

BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



Bài giảng:

**PHÁT TRIỂN ỨNG DỤNG CHO CÁC
THIẾT BỊ DI ĐỘNG**

Biên soạn: Th.S. Nguyễn Hoàng Anh

2019

LỜI GIỚI THIỆU

Môn học PHÁT TRIỂN ỨNG DỤNG CHO CÁC THIẾT BỊ DI ĐỘNG là môn học thuộc phần kiến thức chuyên ngành trong chương trình đào tạo đại học ngành Công nghệ thông tin, chuyên ngành Công nghệ phần mềm của Học viện Công nghệ Bưu chính Viễn thông.

Bài giảng môn học PHÁT TRIỂN ỨNG DỤNG CHO CÁC THIẾT BỊ DI ĐỘNG được kết cấu thành 4 chương:

Chương 1: TỔNG QUAN PHÁT TRIỂN ỨNG DỤNG CHO CÁC THIẾT BỊ DI ĐỘNG

Chương 2: LẬP TRÌNH ỨNG DỤNG ANDROID CƠ BẢN

Chương 3: LẬP TRÌNH ỨNG DỤNG ANDROID NÂNG CAO

Chương 4: PHÁT TRIỂN ỨNG DỤNG ĐA NỀN TẢNG

Bài giảng được biên soạn mới lần đầu với mong muốn cập nhật những kiến thức và công nghệ mới nhất cho lĩnh vực phát triển ứng dụng trên thiết bị di động cho sinh viên ngành Công nghệ thông tin.

Mặc dù, tập thể tác giả đã có nhiều cố gắng trong quá trình biên soạn, nhưng do biên soạn lần đầu nên tập bài giảng khó tránh khỏi những thiếu sót nhất định. Chúng tôi rất mong nhận được ý kiến đóng góp của bạn đọc để tiếp tục hoàn thiện và nâng cao hơn nữa chất lượng bài giảng.

Chúng tôi xin chân thành cảm ơn các bạn đồng nghiệp đã góp ý, giúp đỡ trong quá trình biên soạn bài giảng này.

Tập thể tác giả

MỤC LỤC

LỜI GIỚI THIỆU	11
MỤC LỤC.....	12
DANH MỤC HÌNH	16
CHƯƠNG 1: TỔNG QUAN PHÁT TRIỂN ỨNG DỤNG CHO CÁC THIẾT BỊ DI ĐỘNG	17
1.1. GIỚI THIỆU VỀ THIẾT BỊ DI ĐỘNG.....	17
1.1.1. Khái niệm thiết bị di động	17
1.1.2. Phân loại Đầu cuối di động	17
1.1.3. Lịch sử hình thành và phát triển của thiết bị di động	22
1.1.4. Các đặc điểm của thiết bị di động.....	25
1.2. GIỚI THIỆU VỀ HỆ SINH THÁI DI ĐỘNG.....	26
1.2.1. Nhà mạng (Operators)	26
1.2.2. Mạng lưới (Networks).....	27
1.2.3. Thiết bị (Devices).....	28
1.2.4. Nền tảng (Platforms)	28
1.2.5. Hệ điều hành (Operating systems).....	28
1.2.6. Khung ứng dụng (Application frameworks).....	28
1.2.7. Ứng dụng (Applications)	28
1.2.8. Dịch vụ (Services)	28
1.3. GIỚI THIỆU QUY TRÌNH PHÁT TRIỂN ỨNG DỤNG DI ĐỘNG	28
1.3.1. Ý tưởng (Concept).....	29
1.3.2. Phân tích yêu cầu (Requirement Analysis).....	29
1.3.3. Lập kế hoạch (Planning)	30
1.3.4. Phát triển (Development)	30
1.3.5. Kiểm thử (Testing)	30
1.3.6. Phân phối ứng dụng (Delivery).....	31
1.4. GIỚI THIỆU HỆ ĐIỀU HÀNH DI ĐỘNG.....	31

1.4.1. Giới thiệu.....	31
1.4.2. Phân loại	31
1.4.3. Các đặc điểm cơ bản	33
1.5. GIỚI THIỆU NGÔN NGỮ LẬP TRÌNH CHO ỨNG DỤNG DI ĐỘNG	33
1.5.1. C++	33
1.5.2. Java.....	33
1.5.2. C#.....	34
1.6. THIẾT LẬP MÔI TRƯỜNG LẬP TRÌNH ỨNG DỤNG DI ĐỘNG	35
1.6.1. Khái niệm hệ điều hành Android	35
1.6.2. Đặc điểm cơ bản của Hệ điều hành Android	37
1.6.3. Kiến trúc của Hệ điều hành Android	38
1.6.4. Các thành phần ứng dụng Android.....	42
1.7. KIẾN THỨC CƠ BẢN VỀ NGÔN NGỮ LẬP TRÌNH TRÊN THIẾT BỊ DI ĐỘNG (JAVA)	43
1.7.1. Tập ký tự, từ khóa, định danh.....	43
1.7.2. Cấu trúc chung.....	45
1.7.3. Khái niệm, Phân loại kiểu dữ liệu.....	50
1.7.4. Biến, Hằng, Toán tử.....	51
1.7.5. Lệnh vào ra và điều khiển	51
1.7.6. Lớp, đối tượng.....	52
1.7.7. Kế thừa và đa hình.....	53
1.7.8. Thư viện cơ bản.....	56
1.8. TIẾN TRÌNH LẬP TRÌNH ỨNG DỤNG	56
1.8.1. Khởi chạy ứng dụng.....	56
1.8.2. Viết chương trình	62
1.8.3. Build ứng dụng.....	63
1.8.4. Chạy chương trình và Debug.....	64
CÂU HỎI CHƯƠNG 1	68
CHƯƠNG 2: LẬP TRÌNH ỨNG DỤNG ANDROID CƠ BẢN	69

2.1. QUẢN LÝ VÒNG ĐỜI LẬP TRÌNH ỨNG DỤNG DI ĐỘNG.....	69
2.1.1. Vòng đời ứng dụng di động.....	69
2.1.2. Quản lý vòng đời lập trình ứng dụng di động.....	70
2.2. ACTIVITIES VÀ INTENTS.....	76
2.2.1. Hiểu về Activities.....	76
2.2.2. Liên kết giữa các Activities sử dụng Intents.....	76
2.2.3. Fragments.....	81
2.3. LẬP TRÌNH GIAO DIỆN.....	87
2.3.1. Tìm hiểu giao diện người dùng Android.....	87
2.3.2. Thiết kế giao diện với Views và ViewGroup.....	96
2.3.3. Xử lý Sự kiện trong Android.....	118
2.4. QUẢN LÝ DỮ LIỆU.....	122
2.4.1. Shared Preferences.....	123
2.4.2. Đọc và Ghi file.....	124
2.4.3. Cơ sở dữ liệu quan hệ.....	128
CÂU HỎI CHƯƠNG 2.....	132
CHƯƠNG 3: LẬP TRÌNH ỨNG DỤNG ANDROID NÂNG CAO.....	133
3.1. MESSAGING VÀ NETWORKING.....	133
3.1.1. SMS Messaging.....	133
3.1.2. Sending Email.....	138
3.1.3. Wifi.....	140
3.1.4. Bluetooth.....	142
3.2. CÁC DỊCH VỤ DỰA TRÊN VỊ TRÍ (LOCATION - BASED SERVICES).....	151
3.2.1. Hiển thị bản đồ.....	151
3.2.2. Lấy dữ liệu định vị.....	155
3.3. AUDIO, VIDEO, VÀ SỬ DỤNG CAMERA.....	159
3.3.1. Audio & Video.....	159
3.3.2. Camera.....	163

3.4. SENSORS PHẦN CỨNG	175
3.4.1. <i>Quản lý Sensors</i>	175
3.4.2. <i>Giám sát hướng và chuyển động của thiết bị</i>	180
CÂU HỎI CHƯƠNG 3	182
CHƯƠNG 4: PHÁT TRIỂN ỨNG DỤNG ĐA NỀN TẢNG	184
4.1. ĐỊNH NGHĨA, PHÂN LOẠI ỨNG DỤNG DI ĐỘNG ĐA NỀN TẢNG	184
4.1.1. ĐỊNH NGHĨA ỨNG DỤNG DI ĐỘNG ĐA NỀN TẢNG	184
4.1.2. PHÂN LOẠI ỨNG DỤNG DI ĐỘNG ĐA NỀN TẢNG	185
4.2. GIỚI THIỆU CÁC KHUNG LÀM VIỆC (FRAMEWORK) PHÁT TRIỂN ỨNG DỤNG ĐA NỀN TẢNG	186
4.3. XÂY DỰNG MỘT ỨNG DỤNG ĐA NỀN TẢNG SỬ DỤNG MỘT FRAMEWORK	186
4.3.1. <i>Thiết lập môi trường lập trình ứng dụng đa nền tảng</i>	187
4.3.2. <i>Các kỹ thuật chia sẻ mã nguồn đa nền tảng</i>	188
4.3.3. <i>Lập trình ứng dụng di động đa nền tảng</i>	196
CÂU HỎI VÀ BÀI TẬP CHƯƠNG 4	205
TÀI LIỆU THAM KHẢO	207

DANH MỤC HÌNH

PDF

CHƯƠNG 1: TỔNG QUAN PHÁT TRIỂN ỨNG DỤNG CHO CÁC THIẾT BỊ DI ĐỘNG

Nội dung chương này nhằm giới thiệu một cách tổng quát về các vấn đề liên quan tới việc phát triển ứng dụng trên đầu cuối di động. Nội dung trình bày bao gồm:

- Giới thiệu về đầu cuối di động.
- Giới thiệu về hệ sinh thái di động.
- Giới thiệu về quy trình phát triển ứng dụng di động.
- Giới thiệu về hệ điều hành di động.
- Giới thiệu về các kỹ thuật phát triển ứng dụng di động.
- Giới thiệu một số ngôn ngữ lập trình cho ứng dụng di động.

1.1. GIỚI THIỆU VỀ THIẾT BỊ DI ĐỘNG

1.1.1. Khái niệm thiết bị di động

1.1.1.1. Khái niệm

Đầu cuối di động là bất kỳ thiết bị tính toán cầm tay di động. Một đầu cuối di động thông thường có một số đặc tính chính như sau:

- Có màn hình và bàn phím cho phép người dùng tương tác và nhập liệu.
- Cho phép người dùng mang theo dễ dàng.
- Có khả năng kết nối mạng.
- Cho phép thực thi nhiều tác vụ.

1.1.2. Phân loại Đầu cuối di động

Có nhiều cách phân loại đầu cuối di động. Trong bài giảng sẽ tập trung vào phân loại đầu cuối di động theo tiêu chí dựa trên mục đích sử dụng của đầu cuối di động. Theo mục đích sử dụng đầu cuối di động, từ những năm 1990, đầu cuối di động được phân chia năm chủng loại chính sau:

1.1.2.1. Thiết bị kỹ thuật số hỗ trợ cá nhân (PDA) / Thiết bị kỹ thuật số hỗ trợ doanh nghiệp (EDA)

PDA / EDA là đầu cuối di động hỗ trợ người dùng quản lý các thông tin người dùng / doanh nghiệp dưới dạng số hóa, như: tài liệu, ca nhạc, hình ảnh, video, sổ địa chỉ, email, tin nhắn, lịch nhắc nhở...



Hình 1.1. P835 WVGA PDA của hãng ASUS



Hình 1.2. MC55 EDA của hãng Motorola

PDA/ EDA xây dựng trên nền tảng là một máy tính cá nhân bỏ túi đầy đủ phần cứng và phần mềm. Sự kết hợp giữa sức mạnh của máy tính để bàn và khả năng di chuyển cao đã khiến PDA/ EDA trở nên khá gần gũi và dễ dàng sử dụng với người dùng.

1.1.2.2. Điện thoại di động (Mobile phones)

Điện thoại di động, hay còn gọi là điện thoại cầm tay, là thiết bị viễn thông liên lạc có thể sử dụng trong không gian rộng, phụ thuộc vào nơi phủ sóng của nhà cung cấp dịch vụ. Chất lượng sóng phụ thuộc phần lớn vào thiết bị mạng và địa hình nơi sử dụng máy, chứ ít khi giới hạn về không gian. Trong thời đại ngày nay, điện thoại di động đang là một thiết bị không thể thiếu trong cuộc sống.

Điện thoại di động sử dụng được nhờ khả năng thu phát sóng. Ngày nay, ngoài chức năng thực hiện và nhận cuộc gọi, điện thoại di động còn được tích hợp các chức năng khác như: nhắn tin, duyệt web, nghe nhạc, chụp ảnh, quay phim, xem truyền hình...

Máy điện thoại di động hiện nay được chia làm hai chủng loại sau:

Điện thoại di động tính năng (feature phones) : là điện thoại di động không sử dụng hệ điều hành hoặc sử dụng hệ điều hành của riêng hãng. Điện thoại loại này có tính năng rất đơn giản.



Hình 1.3. Một số điện thoại di động tính năng của hãng Nokia chạy hệ điều hành Series 40 (S40)

Điện thoại di động thông minh (smartphones):



Hình 1.4. Một số điện thoại thông minh

Điện thoại thông minh là điện thoại di động tích hợp một nền tảng hệ điều hành di động với nhiều tính năng hỗ trợ tiên tiến về điện toán và kết nối dựa trên nền tảng cơ bản của điện thoại di động thông thường.

Thời kỳ đầu điện thoại thông minh bao gồm các tính năng của điện thoại thông thường kết hợp với các thiết bị phổ biến khác như PDA, thiết bị điện tử cầm tay, máy ảnh kỹ thuật số, hệ thống định vị toàn cầu GPS... Ngày nay điện thoại thông minh bao gồm tất cả các chức năng của laptop như kết nối mạng wifi, 3G/4G, Bluetooth, chạy các ứng dụng di động (Ví dụ: Web browsers, games, cameras, video players), đồng bộ hóa dữ liệu với máy tính và các thiết bị khác...

Những điện thoại thông minh hiện nay dựa trên nền tảng của một số hệ điều hành phổ biến như : Android của Google, iOS của Apple, Windows phone của Microsoft

1.1.2.3. Máy tính bảng (Tablet computer)



Hình 1.5. Máy tính bảng

Máy tính bảng là một loại thiết bị máy tính tất cả trong một với màn hình cảm ứng, sử dụng bút cảm ứng (nếu có) hay ngón tay để nhập dữ liệu thông tin thông qua bàn phím ảo, thay cho bàn phím và chuột máy tính của máy tính thông thường. Máy tính bảng thường khá nhỏ gọn và tiện lợi cho người sử dụng mang theo bên mình dùng cho các mục đích như : lướt web, xem video, đọc sách, chơi game.

Ví dụ một số máy tính bảng: Samsung Galary Tab 3 Lite, Asus Memo Pad HD 8, Huawei MediaPad 7 Youth, Archos 80 Titanium, Amazon Kindle Fire 2013, Apple iPad Air, Apple iPad Mini 2 Retina...

Một số đặc tính chính của máy tính bảng :

- Chạy hệ điều hành di động (Mobile OS): Cũng giống như thiết bị máy tính khác, máy tính bảng có thể chạy một số hệ điều hành. Các loại máy tính bảng của các hãng khác nhau sẽ chạy các hệ điều hành di động khác nhau. Các hệ điều hành phổ biến nhất hiện nay bao gồm : Apple iOS, Mac OS X, Microsoft Windows, Google Android. Các hệ điều hành khác ít phổ biến hơn bao gồm Windows C, Chrome OS và các phiên bản khác nhau của hệ điều hành Linux.
- Tích hợp ổ đĩa lưu trữ trạng thái (Solid-state drives): cho phép máy tính khởi động và chạy chương trình nhanh chóng.
- Có khả năng kết nối mạng Wifi, 3G/4G.
- Có khả năng kết nối Bluetooth.

1.1.2.4. Máy tính cá nhân siêu di động (Ultra-Mobile PC, viết tắt UMPC)

UMPC là một phiên bản của máy tính bút, được thiết kế nhỏ gọn hơn subnotebook và hoạt động như một máy tính bảng. UMPC có màn hình cảm ứng, bàn phím vật lý, có thể có thêm bút cảm ứng. Không có ranh giới rõ rệt giữa subnotebook và UMPC.



Hình 1.6. Ultra-Mobile PC sản xuất bởi Gigabyte

1.1.2.5. Thiết bị vi tính gắn theo người (Wearable computer)

Thiết bị vi tính gắn theo người là thiết bị công nghệ chúng ta có thể đeo trên người như kính, đồng hồ, dây đeo...



Hình 1.7. Smartwatch của Sony Fan



Hình 1.8. Kính thông minh Google glass

Thiết bị vi tính gắn theo người được đưa ra nhằm mục đích định hình lại thói quen của con người trong cuộc sống với sự hỗ trợ của công nghệ mọi lúc mọi nơi, với một số tính năng tiêu biểu như:

- Thông báo cho người dùng tin tức (ví dụ tin về : thời tiết, cổ phiếu), tin nhắn, tình trạng đồ dùng của bạn (ví dụ pin điện thoại sắp hết).
- Hỗ trợ thực sự khi tích hợp màn hình hiển thị, giọng nói, trí tuệ nhân tạo.
- Giao tiếp với người dùng thông qua văn bản, giọng nói, hình ảnh, video.

1.1.3. Lịch sử hình thành và phát triển của thiết bị di động

Mỗi loại đầu cuối di động đều có lịch sử hình thành và phát triển riêng. Trong phạm vi bài giảng này sẽ nêu ra lịch sử hình thành và phát triển của điện thoại di động, một loại đầu cuối di động phổ biến không thể thiếu trong cuộc sống con người hiện nay.

Ngày 10/3/1876 được coi là mốc son đánh dấu sự ra đời của điện thoại. Cha đẻ của chiếc điện thoại đầu tiên là Alexander Graham Bell người Scottish-Scôt lan, đã được trao bằng sáng chế đầu tiên của Mỹ cho việc phát minh ra điện thoại. Chiếc máy thô sơ có thể truyền được giọng nói này đã mở ra một kỷ nguyên phát triển mới trong lịch sử thông tin liên lạc, thay thế cho điện tín.



Hình 1.9. Chiếc điện thoại đầu tiên ra đời năm 1876

Vào năm 1973, Martin Cooper đã sáng chế ra chiếc điện thoại di động đầu tiên. Điện thoại thời kỳ này gọi là điện thoại dạng cục gạch (Brick phone).



Hình 1.10. Motorola Dyna-Tac - Cell phone đầu tiên năm 1973

Motorola Dyna Tac mang hình dáng gần giống điện thoại di động ngày nay mặc dù vẫn còn khá cồng kềnh, không phổ biến và chi phí đắt hơn điện thoại có dây. Một số đặc tính của điện thoại này như sau:

- Kích thước 9 x 5 x 1.75 inches.
- Trọng lượng: 2.5 pounds.
- Màn hình: không có.
- Số bảng mạch điện: 30.

- thời gian nói chuyện: 35 phút.
- Thời gian sạc pin: 10 giờ.
- Đặc điểm: Nói, nghe, quay số.

Giai đoạn những năm 1988-1998 được gọi là kỷ nguyên của điện thoại dạng thanh kẹo (Candy Bar phone).



Hình 1.11. Nokia candy bar phone

Điện thoại loại này có một số đặc tính sau:

- Hỗ trợ kết nối mạng 2G như GSM, CDMA.
- Tiêu thụ ít năng lượng, nhỏ gọn với sự hỗ trợ của nhiều trạm thu phát sóng ra đời.
- Chất lượng thoại tốt hơn điện thoại thời kỳ trước.
- Có thêm chức năng nhắn tin qua SMS.

Giai đoạn những năm 1998-2008 được gọi là kỷ nguyên của điện thoại tính năng (Feature phone).



Hình 1.12. Điện thoại tính năng Motorola RAZR

Điện thoại loại này có một số đặc tính sau:

- Hỗ trợ kết nối mạng 2.5G như GPRS.
- Hỗ trợ camera.
- Hỗ trợ thoại và nhắn tin qua SMS, MMS.
- Hỗ trợ kết nối Internet tốc độ thấp với giá thành cao và khả năng quảng bá thông tin còn hạn hẹp.

Giai đoạn đầu những năm 2002 đến nay được gọi là kỷ nguyên của điện thoại thông minh (smartphone).



Hình 1.13. Điện thoại thông minh của hãng BlackBerry chạy hệ điều hành Symbian

Điện thoại loại này có một số đặc tính sau:

- Hỗ trợ kết nối mạng 3G, HSDPA, wifi.
- Cho phép thực thi các tác vụ tương tự như trên máy tính thông thường như máy tính (PC).
- Chạy hệ điều hành riêng (Ví dụ: Symbian, Android, iPhone, Windows phone)
- Màn hình rộng, cảm biến.
- Nền tảng di động này đã và đang được quan tâm phát triển rất nhiều bởi các hãng phổ biến như Apple, Samsung, Motorola, Nokia...

Giai đoạn cuối những năm 2010 đến nay được gọi là kỷ nguyên của điện thoại thông minh cảm biến (touch phone).

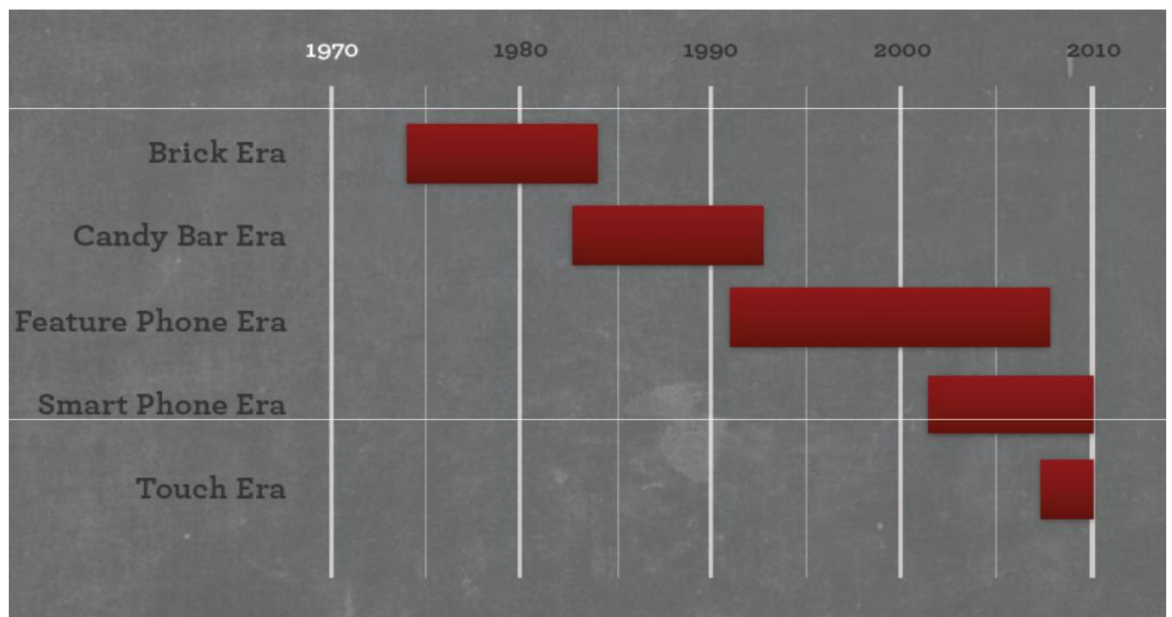


Hình 1.14. Một số điện thoại thông minh cảm biến

Điện thoại loại này có một số đặc tính sau:

- Hỗ trợ kết nối mạng 3G, 4G.
- Hỗ trợ kết nối GPS.
- Giao diện ứng dụng thân thiện hướng tới người dùng.
- Hỗ trợ cảm biến gia tốc.
- Cho phép truy cập Internet tốc độ cao.
- Cho phép lập trình viên phát triển ứng dụng và đưa lên kho ứng dụng.

Tóm lại lịch sử hình thành và phát triển của điện thoại di động trải qua các thời kỳ sau:



Hình 1.15. Lịch sử hình thành và phát triển của điện thoại di động

1.1.4. Các đặc điểm của thiết bị di động

Các đầu cuối di động có rất nhiều loại với kích cỡ, giao diện hiển thị, khả năng xử lý khác nhau. Nhưng nhìn chung chúng có một số đặc trưng cơ bản như sau:

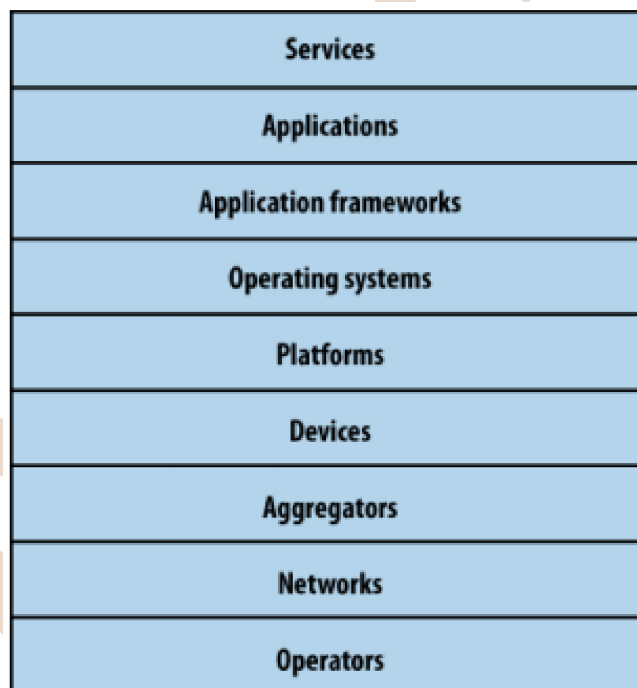
- *Kích thước đầu cuối di động nhỏ gọn*: giúp thuận tiện cho người dùng trong việc sử dụng và di chuyển. Chính vì vậy khi phát triển ứng dụng di động, người phát triển cần xây dựng giao diện ứng dụng có kích thước giao diện phù hợp với đầu cuối di động được chọn.
- *Dung lượng bộ nhớ và tốc độ xử lý của đầu cuối di động thường nhỏ hơn so với máy tính thông thường, xét với cả bộ nhớ trong (RAM, ROM) và bộ nhớ ngoài (disk)*. Chính vì vậy khi phát triển ứng dụng di động, người phát triển cần cân nhắc bộ nhớ dành cho ứng dụng phải hợp lý với loại đầu cuối di động được chọn.
- *Kết nối mạng hạn chế, băng thông thấp*: một số đầu cuối di động không cho phép kết nối được hết các loại mạng (ví dụ: mạng không dây wifi, 3G, 4G...). Ngày nay hầu hết các đầu cuối di động có khả năng kết nối với mạng không dây wifi. Chính vì vậy khi phát triển ứng dụng di động có khả năng kết nối mạng, người phát triển cần cân nhắc tới khả năng kết nối mạng và yêu cầu băng thông của loại đầu cuối di động được chọn.
- *Thời gian phục hồi ngắn*: do đầu cuối di động là loại thiết bị thường xuyên hoạt động và di chuyển, chính vì vậy yêu cầu đặt ra với đầu cuối di động là có thể khởi động trong vài giây (ví dụ PDA), thậm chí một số đầu cuối luôn luôn bật (ví dụ điện thoại di động). Chính vì vậy khi phát triển ứng dụng di động, người phát triển cần đưa ra các xử lý ở từng trạng thái ứng dụng tương ứng với hành vi sử dụng của người dùng đầu

cuối (theo mô hình vòng đời ứng dụng di động).

- *Thời gian sử dụng pin của đầu cuối di động giới hạn.* Chính vì vậy khi phát triển ứng dụng di động, người phát triển cần tối ưu hiệu năng sử dụng ứng dụng để tiết kiệm pin cho đầu cuối.
- *Sự đa dạng về nhập dữ liệu.* Khi phát triển ứng dụng di động, người phát triển cần xác định và xử lý dữ liệu tương ứng với các thao tác nhập liệu từ người dùng (ví dụ nhập liệu thông qua màn hình cảm ứng, thông qua bàn phím...)

1.2. GIỚI THIỆU VỀ HỆ SINH THÁI DI ĐỘNG

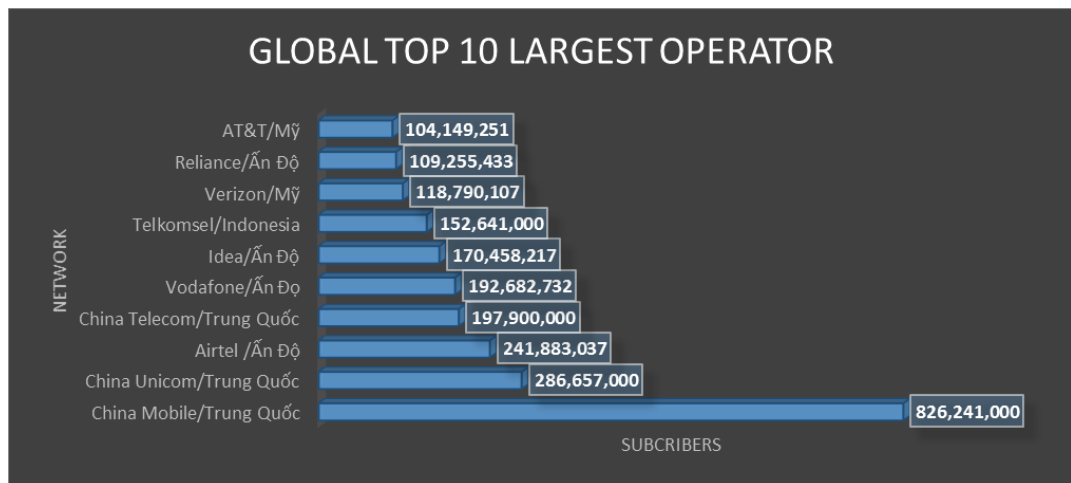
Hệ sinh thái di động là tập hợp các thực thể hoạt động gắn kết thống nhất với nhau nhằm cung cấp các ứng dụng, dịch vụ tới người dùng thiết bị di động. Các thực thể đó được bố trí trong các lớp của hệ sinh thái với mỗi lớp có một vai trò nhất định như sau:



Hình 1.16. Chồng các lớp trong hệ sinh thái di động

1.2.1. Nhà mạng (Operators)

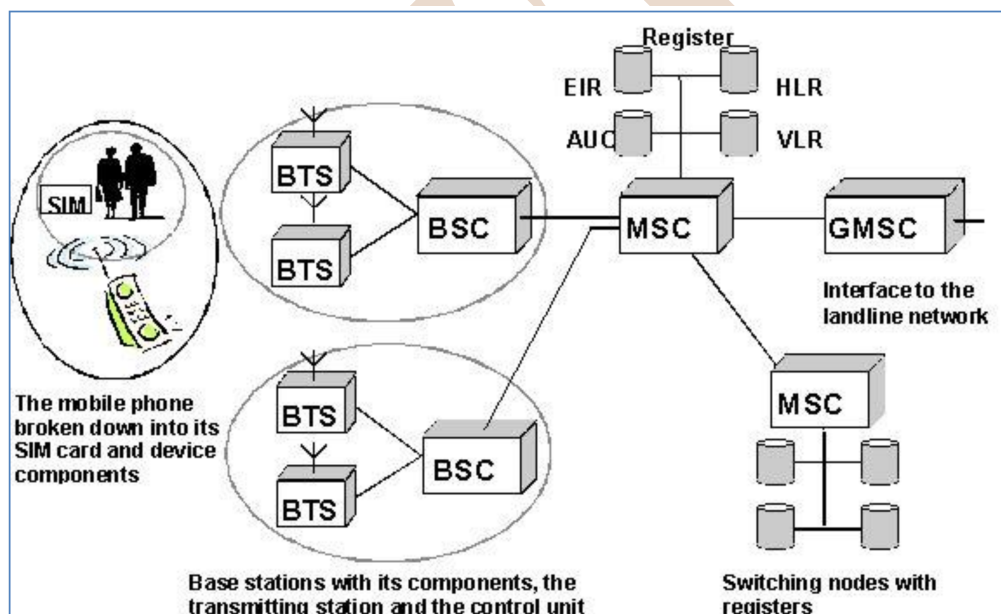
Nhà mạng di động (MNO) là các công ty viễn thông chuyên cung cấp các dịch vụ không dây (wireless) cho các thuê bao di động (Ví dụ dịch vụ thoại, tin nhắn, truy cập Internet).



Hình 1.17. Top 10 nhà mạng lớn nhất thế giới năm 2016 (Nguồn: GSMA Intelligence)

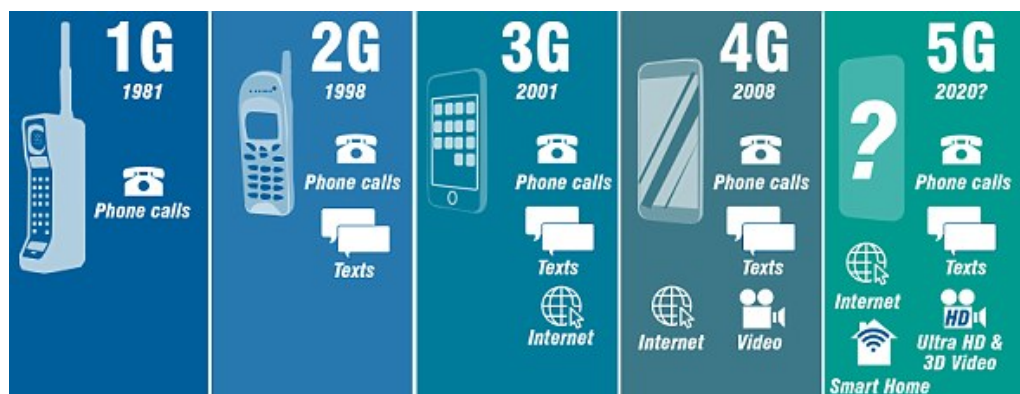
1.2.2. Mạng lưới (Networks)

Mạng di động giao tiếp thông qua sóng vô tuyến điện từ phát ra từ các trạm thu phát sóng.



Hình 1.18. Giao tiếp giữa các thành phần trong mạng di động

Mạng di động được phân chia theo các thế hệ mạng theo các tiêu chuẩn khác nhau như sau:



Hình 1.19. Các thể hệ mạng

1.2.3. Thiết bị (Devices)

Đầu cuối di động là bất kỳ thiết bị tính toán cầm tay di động như điện thoại di động, máy tính bảng, thiết bị đọc sách điện tử... Trong nội dung 1.1 đã có giới thiệu chi tiết về đầu cuối di động.

1.2.4. Nền tảng (Platforms)

Nền tảng di động hỗ trợ ứng dụng trong việc giao tiếp với thiết bị và là môi trường để phát triển và chạy ứng dụng di động.

Nền tảng di động được phân chia thành ba loại:

- Nền tảng di động độc quyền được cung cấp bởi nhà sản xuất thiết bị như : iPhone, BlackBerry, Palm.
- Nền tảng di động cấp phép (bán cho nhà sản xuất thiết bị) : JavaME, Windows
- Mobile, Brew.

Nền tảng di động mã nguồn mở (cho phép người dùng tải, chỉnh sửa) như Android.

1.2.5. Hệ điều hành (Operating systems)

Hệ điều hành di động chạy trên các thiết bị di động và điều hành hoạt động của thiết bị di động. Một số hệ điều hành di động phổ biến như : Symbian, Windows Phone, Android, Linux, Mac OS X, Palm OS...

Chú ý rằng không phải tất cả các thiết bị di động đều có hệ điều hành di động (ví dụ điện thoại dạng tính năng), ngày nay các điện thoại thông minh đều có hệ điều hành di động chạy trên nó.

1.2.6. Khung ứng dụng (Application frameworks)

Khung ứng dụng chạy trên hệ điều hành di động, cung cấp và chia sẻ các dịch vụ cho các ứng dụng di động phía trên nó, như các dịch vụ: tin nhắn, đồ họa, vị trí, bảo mật, xác thực, giao tiếp... Tùy thuộc vào mỗi nền tảng di động thì sẽ có các khung ứng dụng cho nó.

Ví dụ một số khung ứng dụng như : Java, BREW, Flash Lite, Windows Mobile 6, Cocoa Touch, Web.

1.2.7. Ứng dụng (Applications)

Ứng dụng di động là những chương trình phần mềm cài đặt trên thiết bị di động, ví dụ: game, web browser, camera, media player...

Ứng dụng di động có thể được cài đặt sẵn có trên thiết bị di động bán ra hoặc được tạo ra và phân phối tới người dùng di động thông qua kho ứng dụng di động (như Apple store, Google play, Windows phone store...)

1.2.8. Dịch vụ (Services)

Dịch vụ di động là tất cả những gì người dùng đầu cuối sử dụng như truy cập Internet, gửi tin nhắn văn bản, lấy thông tin vị trí,...

1.3. GIỚI THIỆU QUY TRÌNH PHÁT TRIỂN ỨNG DỤNG DI ĐỘNG

Có thể nói mỗi đơn vị phát triển ứng dụng di động thường có quy trình phát triển riêng tùy thuộc vào cách thức tổ chức và nghiệp vụ trong đơn vị mình. Nhưng nhìn chung quy trình phát triển ứng dụng di động phổ biến thường theo mô hình cơ bản sau:



Hình 1.20. Quy trình phát triển ứng dụng di động

Miêu tả cụ thể các bước trong quy trình trên như sau:

1.3.1. Ý tưởng (Concept)

Để tạo ra ứng dụng di động thì đầu tiên cần có ý tưởng cho ứng dụng. Ý tưởng cho ứng dụng di động có thể được những người phát triển tự đưa ra hoặc làm theo yêu cầu đặt hàng của phía đối tác.

1.3.2. Phân tích yêu cầu (Requirement Analysis)

Sau khi có ý tưởng cho ứng dụng, người phát triển cần phân tích các yêu cầu cần thiết cho việc phát triển ứng dụng. Quá trình phân tích yêu cầu xét trên một số phương diện chính sau:

- Phân tích tương tác người dùng với ứng dụng.

- Phân tích thói quen sử dụng của người dùng với ứng dụng.
- Phân tích môi trường hoạt động của ứng dụng (ví dụ: tương tác giữa ứng dụng với môi trường mạng, thiết bị...)

1.3.3. Lập kế hoạch (Planning)

Sau khi phân tích yêu cầu ứng dụng, bước tiếp theo là lập kế hoạch phát triển ứng dụng để biến ý tưởng thành sản phẩm ứng dụng di động cụ thể. Quá trình lập kế hoạch gồm một số khâu chính:

- Xác định mục đích, phạm vi của dự án.
- Ước lượng chi phí dự án.
- Phân tích rủi ro dự án.
- Lập lịch.

1.3.4. Phát triển (Development)

Sau giai đoạn lập kế hoạch, bước tiếp theo là phân tích, thiết kế và lập trình ứng dụng. Giai đoạn này gồm ba khâu:

- Phân tích thiết kế phần mềm (Analysis & Design): ngôn ngữ mô hình hóa thống nhất thường sử dụng để phân tích thiết kế phần mềm hiện nay là ngôn ngữ UML.
- Thiết kế đồ họa và thiết kế tương tác (Designing).
- Lập trình (Coding): tùy thuộc vào ứng dụng phát triển trên nền tảng di động nào thì lập trình viên sử dụng ngôn ngữ và công cụ lập trình phù hợp cho nền tảng đó. Ví dụ: để lập trình ứng dụng cho Android thì sử dụng ngôn ngữ lập trình Java và công cụ lập trình là Android Studio, cho iOS thì sử dụng ngôn ngữ lập trình Objective C và công cụ lập trình là Xcode, cho Windows phone thì ngôn ngữ lập trình là C#/ VB.NET và công cụ lập trình là Visual studio.

1.3.5. Kiểm thử (Testing)

Sau giai đoạn phân tích, thiết kế và lập trình ứng dụng, bước tiếp theo là kiểm thử phần mềm. Giai đoạn này gồm hai khâu:

- Kiểm thử nội bộ phần mềm (Testing): Giai đoạn này nhằm kiểm tra xem ứng dụng hoạt động tốt ở các kịch bản khác nhau trong phạm vi nội bộ. Các loại kiểm thử cần xét đến đó là:
 - Kiểm thử chức năng.
 - Kiểm thử hiệu năng.
 - Kiểm thử bảo mật.
 - Kiểm thử thói quen sử dụng của người dùng.
- Kho ứng dụng kiểm thử phần mềm (UAC – User Acceptance Testing): Sau bước kiểm thử nội bộ phần mềm, người phát triển tiến hành đưa ứng dụng lên kho ứng dụng để người dùng đầu cuối khác có thể tải ứng dụng từ kho về dùng. Khi nhận được ứng dụng do người phát triển đưa lên, kho ứng dụng tiến hành kiểm thử phần mềm.

1.3.6. Phân phối ứng dụng (Delivery)

Sau giai đoạn kiểm thử ứng dụng, nếu ứng dụng không có lỗi thì phần mềm sẽ xuất hiện trên kho ứng dụng và người dùng có thể vào kho tải và cài đặt ứng dụng về máy của mình.

1.4. GIỚI THIỆU HỆ ĐIỀU HÀNH DI ĐỘNG

1.4.1. Giới thiệu

Hệ điều hành di động (mobile OS) là phần mềm hệ thống chạy trên thiết bị di động, dùng để điều hành, quản lý các thiết bị phần cứng và tài nguyên phần mềm của thiết bị di động. Hệ điều hành di động đóng vai trò trung gian trong việc giao tiếp giữa người sử dụng và thiết bị di động, cung cấp một môi trường cho phép người sử dụng phát triển và thực hiện các ứng dụng của họ một cách dễ dàng. Một số thiết bị di động chạy hệ điều hành di động như điện thoại thông minh, máy tính bảng, PDA...

1.4.2. Phân loại

Phân loại hệ điều hành di động theo nền tảng di động thì có một số dòng hệ điều hành di động phổ biến như sau:

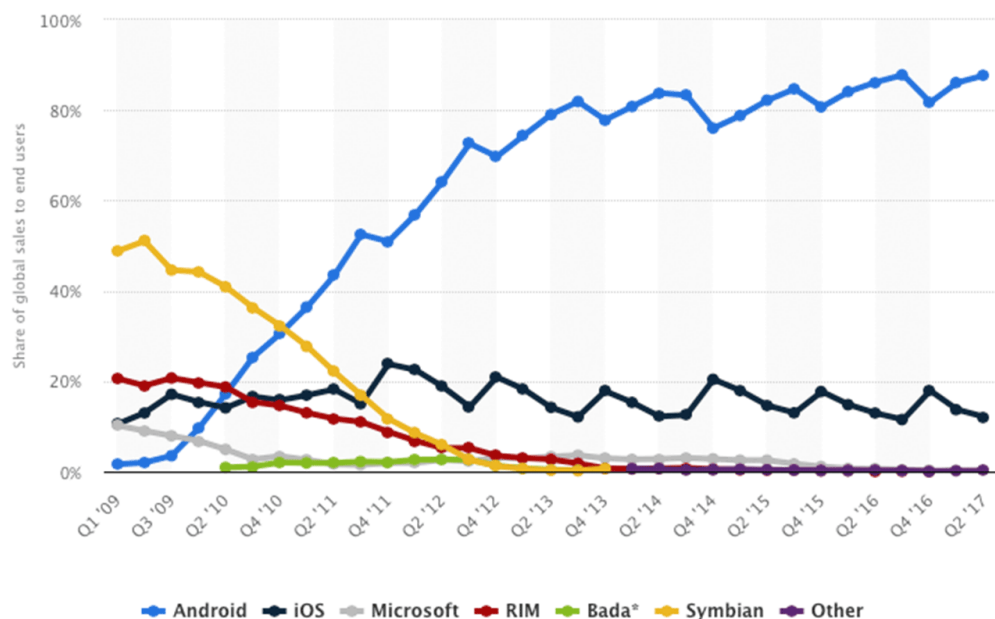
- Java ME OS
- Palm OS
- Symbian OS
- Linux OS
- Windows phone OS
- BlackBerry OS
- iPhone OS
- Android OS



Hình 1.21. Một số giao diện hệ điều hành di động: (a) Android 4.4 Kit Kat; (b) iOS 7; (c) Windows Phone 8; (d) BlackBerry OS 10; (e) HP web OS; (f) Nokia Symbian

3

Global mobile OS market share in sales to user from 1st quarter 2009 to 2nd quarter 2017



Hình 1.22. Thị phần của các hệ điều hành di động trên thế giới tới hết năm 2017

1.4.3. Các đặc điểm cơ bản

- Hỗ trợ xử lý đa nhiệm: hệ điều hành di động cho phép nhiều ứng dụng xử lý cùng một lúc trên nó.
- Có khả năng lập lịch: cho phép thực hiện phân bổ và gán quy trình thực thi cho các tác vụ trong bộ xử lý đúng thời điểm yêu cầu.
- Quản lý bộ nhớ: cho phép cấp phát và quản lý bộ nhớ cấp cho ứng dụng.
- Quản lý hệ thống tập tin, như các thao tác đọc, viết, truy xuất tập tin của hệ thống.
- Hỗ trợ giao diện bàn phím: cho phép người dùng nhập liệu vào thiết bị.
- Hỗ trợ giao diện vào ra (I/O): cung cấp các giao diện khác nhau cho việc vào ra dữ liệu như bàn phím cứng, bàn phím cảm biến, màn hình, ...
- Quản lý bảo mật thông tin.
- Xử lý dữ liệu đa phương tiện, như video, image, audio.

1.5. GIỚI THIỆU NGÔN NGỮ LẬP TRÌNH CHO ỨNG DỤNG DI ĐỘNG

1.5.1. C++

Lịch sử phát triển của C++

C++ ra đời vào giữa những năm 1980, là một ngôn ngữ lập trình hướng đối tượng được mở rộng từ ngôn ngữ lập trình cấu trúc C. Cho nên, C++ là ngôn ngữ lập trình nửa hướng đối tượng, nửa hướng cấu trúc.

Đặc trưng của ngôn ngữ C++

- Cho phép định nghĩa lớp đối tượng.
- Cho phép đóng gói dữ liệu vào các lớp đối tượng. Cho phép định nghĩa phạm vi truy nhập dữ liệu của lớp bằng các từ khóa phạm vi.
- Cho phép kế thừa lớp với các kiểu kế thừa khác nhau tùy vào từ khóa dẫn xuất.
- Cho phép lớp kế thừa sử dụng các phương thức của lớp bị kế thừa (trong phạm vi quy định).
- Cho phép định nghĩa chồng phương thức trong lớp kế thừa.

Một số nền tảng di động sử dụng ngôn ngữ C++

Nền tảng di động phổ biến nhất hiện nay sử dụng ngôn ngữ C++ là Symbian, Qualcomm's BREW.

1.5.2. Java

Lịch sử phát triển của Java:

Năm 1991, một nhóm kỹ sư của Sun Microsystems muốn lập trình để điều khiển các thiết bị điện tử như ti vi, máy ảnh, lò nướng...Ban đầu, họ định dùng C và C++ nhưng trình biên dịch C/C++ lại phụ thuộc vào từng loại CPU. Do đó, họ đã bắt tay vào xây dựng một ngôn ngữ chạy nhanh, gọn, hiệu quả, độc lập thiết bị và ngôn ngữ “Oak” ra đời và vào năm 1995, sau đó đổi tên thành Java.

Ngôn ngữ lập trình Java được Sun Microsystems đưa ra giới thiệu vào tháng 6 năm 1995 và đã nhanh chóng trở thành một ngôn ngữ lập trình của các lập trình viên chuyên nghiệp. Java được xây dựng trên nền tảng của C và C++ nghĩa là Java sử dụng cú pháp của C và đặc trưng hướng đối tượng của C++. Java là ngôn ngữ vừa biên dịch vừa thông dịch. Đầu tiên mã nguồn được biên dịch thành mã bytecode. Sau đó được thực thi trên từng loại máy nhờ trình thông dịch. Mục tiêu của các nhà thiết kế Java là cho phép người lập trình viết chương trình một lần nhưng có thể chạy trên các nền phần cứng khác nhau.

Ngày nay, Java được sử dụng rộng rãi, không chỉ để viết các ứng dụng trên máy cục bộ hay trên mạng mà còn để xây dựng các trình điều khiển thiết bị tự động, PDA,...

Đặc trưng của ngôn ngữ Java:

- Đơn giản
- Hướng đối tượng
- Độc lập phần cứng và hệ điều hành
- Mạnh mẽ
- Bảo mật
- Phân tán
- Đa luồng
- Linh động

Một số nền tảng di động sử dụng ngôn ngữ Java

Nền tảng di động phổ biến nhất hiện nay sử dụng ngôn ngữ Java là Android. Ngoài ra còn một số nền tảng khác như J2ME, BlackBerry.

1.5.2. C#

Lịch sử phát triển của C#

Các ngôn ngữ lập trình .NET của Microsoft, trong đó có C# ra đời vào cuối những năm 1990 để cạnh tranh với ngôn ngữ lập trình Java. C# là một ngôn ngữ hoàn toàn hướng đối tượng, hơn nữa nó còn cung cấp một giao diện lập trình đồ họa thân thiện và đẹp mắt với truyền thống lập trình kéo thả của Microsoft.

Đặc trưng của ngôn ngữ C#

- C# là một ngôn ngữ hoàn toàn hướng đối tượng: tất cả các thành phần, các thực thể trong chương trình đều được mô hình dưới dạng một lớp nhất định. Không có dữ liệu tự do và hàm tự do trong chương trình.
- Cung cấp giao diện lập trình đồ họa: lập trình viên chỉ cần kéo và thả các đối tượng đồ họa cho ứng dụng của mình.
- Cho phép lập trình viên tự tạo ra các thư viện UserControl của mình. Đây là một thư viện bao gồm các thành phần được người dùng tự thiết kế giao

điện, viết mã nguồn, đóng gói và có thể sử dụng lại trong nhiều ứng dụng khác nhau, tùy theo chức năng của các thành phần.

Một số nền tảng di động sử dụng ngôn ngữ C#

Nền tảng di động phổ biến nhất hiện nay sử dụng ngôn ngữ C# là Windows phone.

1.6. THIẾT LẬP MÔI TRƯỜNG LẬP TRÌNH ỨNG DỤNG DI ĐỘNG

1.6.1. Khái niệm hệ điều hành Android



Android là một hệ điều hành dựa trên nền tảng Linux được thiết kế dành cho các thiết bị di động có màn hình cảm ứng (ví dụ: điện thoại thông minh, máy tính bảng,...). Ban đầu, Android được phát triển bởi tổng công ty Android, với sự hỗ trợ tài chính từ Google và sau này được Google mua lại vào năm 2005. Android ra mắt vào năm 2007 cùng với tuyên bố thành lập liên minh thiết bị cầm tay mở: một hiệp hội gồm các công ty phần cứng, phần mềm và viễn thông với mục tiêu đẩy mạnh các tiêu chuẩn mở cho các thiết bị di động. Chiếc điện thoại đầu tiên chạy Android được bán vào tháng 10 năm 2008.



Hình 1.23. Điện thoại Android, HTC Dream (trái) và HTC Magic (phải)



Hình 1.24. Máy tính bảng chạy hệ điều hành Android

Android có mã nguồn mở và Google phát hành mã nguồn theo giấy phép Apache. Chính mã nguồn mở cùng với một giấy phép không có nhiều ràng buộc đã cho phép các nhà phát triển thiết bị, mạng di động và các lập trình viên nhiệt huyết được điều chỉnh và phân phối Android một cách tự do. Ngoài ra, Android còn có một cộng đồng lập trình viên đông đảo chuyên viết các ứng dụng để mở rộng chức năng của thiết bị, bằng một loại ngôn ngữ lập trình Java có sửa đổi. Vào tháng 2 năm 2014, có khoảng 1.200.000 ứng dụng trên Android, và số lượt tải ứng dụng từ Google Play, cửa hàng ứng dụng chính của Android, ước tính khoảng 29 tỷ lượt.

Những yếu tố này đã giúp Android trở thành nền tảng điện thoại thông minh phổ biến nhất thế giới, vượt qua Symbian vào quý 4 năm 2010 và được các công ty công nghệ lựa chọn khi họ cần một hệ điều hành không nặng nề, có khả năng tinh chỉnh, và giá rẻ chạy trên các thiết bị công nghệ cao thay vì tạo dựng từ đầu. Kết quả là mặc dù được thiết kế để chạy trên điện thoại và máy tính bảng, Android đã xuất hiện trên TV, máy chơi game và các thiết bị điện tử khác.



Hình 1.25. Thiết bị đọc sách điện tử (e-book readers) chạy hệ điều hành Android



Hình 1.26. Tivi chạy hệ điều hành Android



Hình 1.27. Đồng hồ thông minh chạy hệ điều hành Android

Bản chất mở của Android cũng khích lệ một đội ngũ đông đảo lập trình viên và những người đam mê sử dụng mã nguồn mở để tạo ra những dự án do cộng đồng quản lý. Những dự án này bổ sung các tính năng cao cấp cho những người dùng thích tìm tòi hoặc đưa Android vào các thiết bị ban đầu chạy hệ điều hành khác.

Android chiếm 80% thị phần điện thoại thông minh trên toàn thế giới vào thời điểm quý 3 năm 2013, với tổng cộng 1 nghìn tỉ thiết bị đã được kích hoạt. Sự thành công của hệ điều hành cũng khiến nó trở thành mục tiêu trong các vụ kiện liên quan đến bằng phát minh, góp mặt trong cái gọi là “cuộc chiến điện thoại thông minh” giữa các công ty công nghệ.

1. Android 1.5: Android Cupcake
2. Android 1.6: Android Donut
3. Android 2.0: Android Eclair
4. Android 2.2: Android Froyo
5. Android 2.3: Android Gingerbread
6. Android 3.0: Android Honeycomb
7. Android 4.0: Android Ice Cream Sandwich
8. Android 4.1 to 4.3.1: Android Jelly Bean
9. Android 4.4 to 4.4.4: Android KitKat
10. Android 5.0 to 5.1.1: Android Lollipop
11. Android 6.0 to 6.0.1: Android Marshmallow
12. Android 7.0 to 7.1: Android Nougat
13. Android 8.0 to Android 8.1: Android Oreo
14. Android 9.0: Android Pie

1.6.2. Đặc điểm cơ bản của Hệ điều hành Android

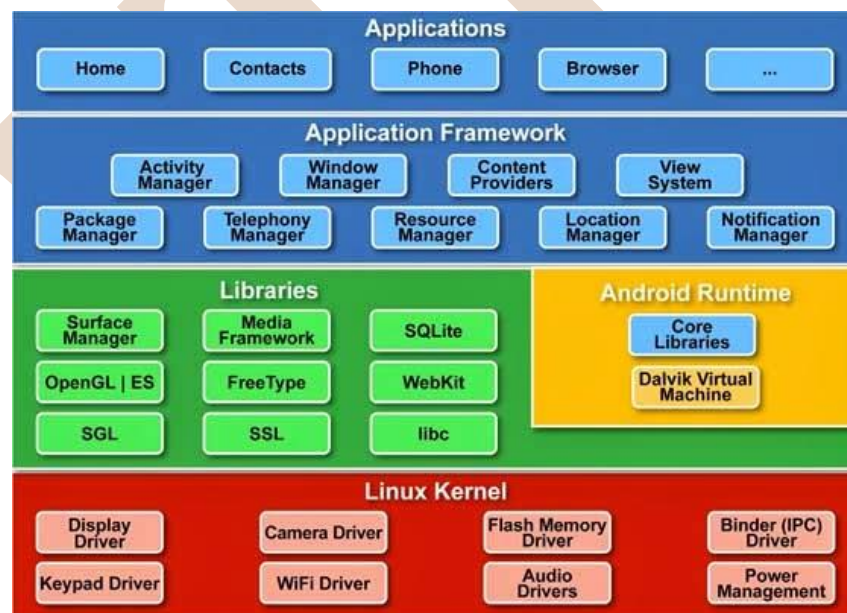
Các đặc trưng cơ bản của Hệ điều hành Android

Bảng 1.1. Các đặc trưng cơ bản của Hệ điều hành Android

Đặc trưng	Miêu tả
Kết nối	GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wifi, LTE, NFC, WiMAX
Lưu trữ	Sử dụng cơ sở dữ liệu quan hệ SQLite ; Lưu trữ dữ liệu trong bộ nhớ thiết bị ; Lưu trữ dữ liệu kiểu key/value
Hỗ trợ media	Theo chuẩn: H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, BMP
Tin nhắn	SMS, MMS
Web browser	Dựa trên bộ thư viện mã nguồn mở WebKit
Hỗ trợ thiết bị phần cứng	Accelerometer sensor, camera, digital compass, Proximity sensor, GPS
Đa chạm	Hỗ trợ màn hình đa chạm
Đa nhiệm	Ứng dụng có khả năng hoạt động đa nhiệm
Chia sẻ kết nối Internet	Hỗ trợ chia sẻ kết nối Internet như điểm truy nhập có dây / không dây
Hỗ trợ flash	Android 2.3 hỗ trợ Flash 10.1
Hỗ trợ widget tùy biến	Widget có khả năng tùy biến cho phép người dùng mở rộng hoặc thu nhỏ lại để xem thông tin cho phù hợp.
Đa ngôn ngữ	Hỗ trợ hiển thị bằng đa ngôn ngữ

1.6.3. Kiến trúc của Hệ điều hành Android

Hệ điều hành Android có các thành phần phần mềm chia thành 5 phần với 4 tầng chính trong sơ đồ kiến trúc như hình dưới đây:



Hình 1.28. Kiến trúc của hệ điều hành Android

Chức năng chi tiết của từng tầng trong kiến trúc của hệ điều hành Android như sau:

Hạt nhân của hệ điều hành



Hình 1.29. Nhân Linux trên Android

Android sử dụng nhân Linux 2.6. Tầng này cung cấp chức năng hệ thống cơ bản như giao tiếp với tầng trên, bảo mật, quản lý bộ nhớ, quản lý các tiến trình, ngăn xếp mạng và trình điều khiển thiết bị (máy ảnh, bàn phím, màn hình hiển thị, giao tiếp USB, giao tiếp hồng ngoại, không dây, v.v...). Ngoài ra, nhân Linux này cũng có vai trò như một lớp trừu tượng giữa phần cứng và phần mềm.

Thư viện và các giao diện lập trình ứng dụng

Android có một thư viện khá phong phú cung cấp sẵn để người lập trình có thể sử dụng. Sơ đồ tóm tắt về hệ thống thư viện này như sau:



Hình 1.30. Thư viện và các giao diện lập trình ứng dụng Android

- Bộ thư viện: Android cung cấp một tập hợp các thư viện C/C++ như:
 - o OpenGL: thư viện dùng để tạo ra các đồ họa 3D dựa vào chuẩn OpenGL ES 1.0.
 - o FreeType: thư viện hỗ trợ xử lý bitmap, font, vector.
 - o SGL: thư viện cơ bản cung cấp các engine đồ họa 2D.
 - o Libc : thư viện C chuẩn, được tối ưu cho các thiết bị Linux-based.
 - o SQLite: thư viện thao tác với cơ sở dữ liệu quan hệ nhỏ gọn SQLite.
 - o SSL : thư viện hỗ trợ sử dụng giao thức mã hóa SSL (Secure Sockets Layer) trong bảo mật truyền thông Internet
- Android runtime:

Android runtime cung cấp một tập các thư viện lõi cho phép các nhà phát triển viết các ứng dụng Android sử dụng ngôn ngữ lập trình Java. Danh sách các gói API cơ bản sau được cung cấp bởi tất cả các thiết bị trên nền Android:

Bảng 1.2. Các gói thư viện trong Java

Gói thư viện	Miêu tả
android.util	Gói tiện ích cơ bản bao gồm nhiều lớp mức thấp như các lớp quản lý (List, Stack...), lớp xử lý chuỗi, lớp xử lý XML.
android.os	Gói hệ điều hành cho phép truy cập đến các dịch vụ cơ bản như: tin nhắn, đồng hồ và gỡ lỗi.

android.graphics	Gói đồ họa cung cấp các lớp đồ họa mức thấp thực hiện các chức năng đồ họa, màu, vẽ cơ bản.
android.text	Công cụ hiển thị và xử lý văn bản
android.database	Cung cấp các lớp mức thấp làm việc với cơ sở dữ liệu như SQLite.
android.content	Các giao tiếp lập trình nội dung được dùng để quản lý truy cập dữ liệu và xuất bản bằng cách cung cấp các dịch vụ thao tác với tài nguyên, nhà cung cấp nội dung và các gói.
android.view	Cung cấp các lớp giao diện người dùng cơ bản nhất. Tất cả giao diện người dùng được tạo ra đều phải sử dụng một tập các view.
android.widget	Xây dựng kế thừa từ gói android.view. Widget bao gồm những thành phần giao diện tạo sẵn để tạo nên giao diện người dùng. Các widget bao gồm danh sách, nút bấm, hộp nhập, các kiểu trình bày (layout).
com.google.android.maps	Bộ API mức cao cho phép ứng dụng truy cập đến bản đồ Google sẵn có của Android, bao gồm các lớp MapView, Overlay, MapController để tương tác với bản đồ bên trong ứng dụng
android.app	Gói ứng dụng là một gói thư viện bậc cao, cho phép truy cập đến dữ liệu của ứng dụng. Gói ứng dụng cũng bao gồm Activity, Service là thành phần cơ bản của mọi ứng dụng Android.
android.provider	Bao gồm một tập các lớp cho phép ứng dụng truy cập đến dữ liệu tiêu chuẩn trong tất cả các bản phân phối Android (như dữ liệu danh bạ).
android.telephony	Cho phép ứng dụng tạo, nhận, theo dõi cuộc gọi, tình trạng cuộc gọi và tin nhắn SMS.
android.webkit	Bao gồm một lớp WebView để tạo giao diện web nhúng trong ứng dụng, làm việc với các nội dung web và một trình quản lý cookie.
Ngoài các thư viện chuẩn của Android, để đáp ứng tiêu chí phù hợp với nhiều thiết bị khác nhau, Android còn có thể có các API phụ thuộc thiết bị như: android.location, android.media, android.opengl, android.hardware, android.bluetooth, android.net.wifi.	

Ngoài ra, Android runtime cung cấp máy ảo Dalvik – một loại máy ảo Java được thiết kế đặc biệt và tối ưu hóa cho Android. Máy ảo Dalvik sử dụng các tính năng cốt lõi của Linux như quản lý bộ nhớ và đa luồng, những tính năng này đều có trong ngôn ngữ Java. Máy ảo Dalvik cho phép tất cả các ứng dụng Android chạy trong tiến trình riêng của nó.

Khung ứng dụng Android



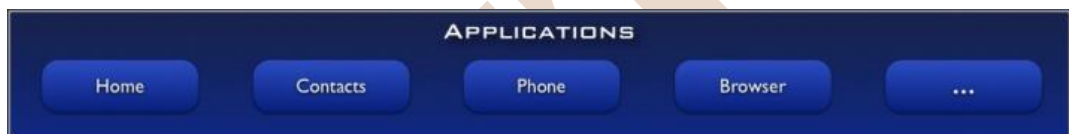
Hình 1.31. Khung ứng dụng Android

Khung ứng dụng Android cung cấp nhiều dịch vụ cấp cao hơn thông qua các API ở dạng các lớp Java từ Android SDK cho các ứng dụng phía lớp trên. Các nhà phát triển ứng dụng được phép sử dụng các dịch vụ này trong các ứng dụng của họ (Ví dụ: dịch vụ kết nối Internet, dịch vụ SMS...), điều này cho phép các nhà phát triển khả năng xây dựng các ứng dụng cực kỳ phong phú và sáng tạo.

Cơ bản ứng dụng Android bao gồm một bộ các dịch vụ do hệ thống cung cấp, bao gồm:

- Activity manager, fragment manager: quản lý chu trình sống và điều hướng các activity, fragment. Đồng thời quản lý activity stack.
- View system : một tập hợp rất nhiều các view sử dụng để tạo lập giao diện người dùng cho các activity, fragments.
- Notification manager: cung cấp cơ chế ổn định và quy củ cho việc gửi các thông báo đến người dùng thông qua các alerts trong thanh trạng thái (status bar).
- Content provider: cho phép các ứng dụng có thể truy cập và chia sẻ dữ liệu với ứng dụng khác (như ứng dụng danh bạ sẵn có trong điện thoại).
- Resource manager: cho phép truy xuất các thành phần không thuộc mã nguồn, chẳng hạn như: chuỗi ký tự (localized strings), đồ họa (graphics), file giao diện (layout files).

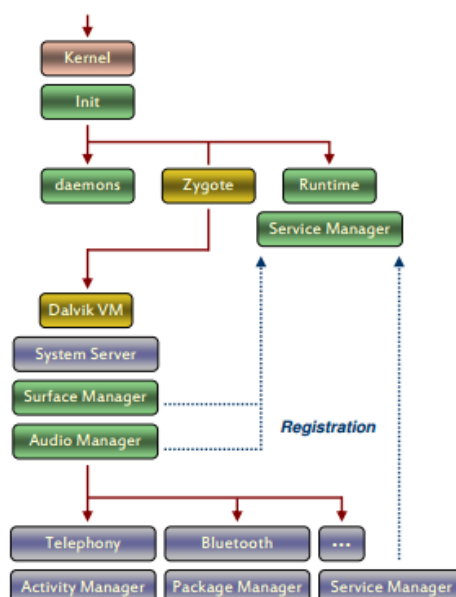
Ứng dụng



Hình 1.32. Các ứng dụng trên Android

Đây là tầng trên cùng trong kiến trúc của hệ điều hành Android. Mặc định Android tích hợp sẵn một số ứng dụng cần thiết cơ bản như: home, contacts, phone, browser, camera... Tất cả các ứng dụng chạy trên hệ điều hành Android đều được viết bằng ngôn ngữ Java và được quản lý bởi máy ảo Dalvik. Các ứng dụng này có thể được cung cấp sẵn hoặc được phát triển bởi lập trình viên. Khi ứng dụng được phát triển bởi lập trình viên cần đưa ứng dụng lên kho ứng dụng (ví dụ: Google Play, Amazon Appstore), người dùng đầu cuối có thể vào kho tải ứng dụng về máy.

Sau khi có cái nhìn tổng quát về kiến trúc của hệ điều hành Android, chúng ta cần hiểu quá trình khởi động các thành phần trong kiến trúc trên vào điện thoại Android diễn ra như thế nào?



Hình 1.33. Quá trình khởi động hệ điều hành vào điện thoại Android

Quá trình trên được miêu tả cụ thể như sau:

Khi nhấn nút trên điện thoại để khởi động hệ điều hành Android thì phần nhân (Kernel) sẽ được khởi động trước tiên và gọi đến tiến trình khởi tạo (Init). Tại tiến trình khởi tạo này sẽ chạy phần Daemons, Daemons làm nhiệm vụ quản lý phần cứng cấp thấp (ví dụ: usb, adb, debugger, radio...). Sau đó tiến trình mỗi Zygote và Runtime sẽ được gọi đồng thời.

- Tiến trình mỗi Zygote được gọi nhằm khởi động máy ảo Dalvik (Dalvik VM) để chạy các ứng dụng.
- Tiến trình Runtime được gọi nhằm khởi động trình quản lý dịch vụ (Service Manager). Một điều cần chú ý ở đây là bất cứ một dịch vụ chạy ngầm nào đều phải được đăng ký với Service Manager (Ví dụ: Surface Manager, Audio Manager...)

Từ đây tất cả các thành phần khác trong hệ điều hành Android sẽ được khởi động.

1.6.4. Các thành phần ứng dụng Android

Một ứng dụng trên Android được cấu thành từ bốn thành phần chính sau:

Bảng 1.3. Các thành phần của ứng dụng Android

Thành phần	Miêu tả
Activities	Là thành phần tối quan trọng của bất kỳ một ứng dụng Android. Màn hình giao diện và xử lý tương tác của người dùng được quản lý bởi Activity. Thông thường trong một ứng dụng Android có thể có một hoặc nhiều Activity, trong đó có một main Activity để khởi chạy ứng dụng, Activity này sẽ gọi các Activity khác trong quá trình ứng dụng hoạt động.
Services	Dịch vụ chạy ngầm trong hệ thống mà người sử dụng không thấy được. Service sử dụng để cập nhật dữ liệu, đưa ra thông báo (Notification).

	Ví dụ: khi bạn yêu cầu phát một bài nhạc, sẽ có một Service chạy bên dưới để phát bài nhạc. Khi bạn download một tập tin, sẽ có một Service thực hiện tác vụ download
Broadcast Receivers	Quản lý sự kiện, chuyển tiếp sự kiện trong hệ thống (Ví dụ: xử lý truyền thông giữa hệ điều hành và ứng dụng Android).
Content Provider	Cung cấp cách thức quản lý và chia sẻ tài nguyên giữa các ứng dụng.

Ngoài ra, có một số thành phần khác được sử dụng trong ứng dụng android, đó là:

Bảng 1.4. Các thành phần khác

Thành phần	Miêu tả
Fragments	Là thành phần giao diện được nhúng vào Activity, có thể hiểu Fragment là một sub Activity. Fragment được thêm vào phiên bản Android 3.0 (API level 11). Việc sử dụng Fragment giải quyết vấn đề khi cho phép ứng dụng chạy trên các dòng thiết bị có độ phân giải khác nhau có màn hình giao diện ứng dụng phù hợp.
Views	Là các đối tượng hiển thị trên giao diện ứng dụng (Ví dụ: button, list, textbox...). View chỉ được hiển thị lên màn hình khi được đặt trong Activity.
Layouts	Cho phép bố trí sắp xếp các views lên giao diện ứng dụng theo nhu cầu.
Intents	Dùng để truyền các thông báo từ Activity này tới service hoặc Activity khác (Ví dụ: khi mở 1 trang web, bạn gửi 1 Intent đi để tạo 1 Activity mới hiển thị trang web đó).
Resources	Lưu trữ tài nguyên của ứng dụng, như: strings, constants, drawables pictures.
Manifest	File cấu hình của ứng dụng.

Các thành phần trên không nhất thiết phải có mặt đầy đủ trong ứng dụng Android. Mỗi một thành phần trên muốn được triệu gọi trong ứng dụng thì bắt buộc nó phải được khai báo trong AndroidManifest.xml. Chúng ta có thể xem các thành phần nào được sử dụng trong ứng dụng bằng việc xem khai báo trong file AndroidManifest.xml

1.7. KIẾN THỨC CƠ BẢN VỀ NGÔN NGỮ LẬP TRÌNH TRÊN THIẾT BỊ DI ĐỘNG (JAVA)

1.7.1. Tập ký tự, từ khóa, định danh

1.7.1.1. Tập ký tự trong Java

- Tập các chữ cái in hoa: A, B, C, D, . . . , Z
- Tập các chữ cái in thường: a, b, c, d, . . . , z
- Tập các chữ số: 0, 1, 2, 3, . . . , 9
- Các dấu chấm câu: , . ; : / ? [] { } ! @ # \$ ^ & * () + = - < > “
- Các ký tự không nhìn thấy: dấu trống (Space), dấu Tab, dấu xuống dòng (Enter),
- Dấu gạch dưới _

1.7.1.2. Tập từ khóa trong Java

- Từ khóa là những từ có ý nghĩa xác định.

- Thường dùng để khai báo các kiểu dữ liệu, viết các toán tử và câu lệnh.
- Chú ý:
 - i. Không được dùng các từ khóa để đặt tên cho hằng, biến, mảng, hàm,...
 - ii. Từ khóa phải viết bằng chữ thường

Tập hợp tất cả các từ khoá có thể được phân thành các nhóm từ khoá sau đây để dễ nhớ, dễ tiếp cận:

Bảng 1.5. Các kiểu dữ liệu trong Java

STT	Loại	Từ khóa
1	Kiểu dữ liệu cơ bản	byte, short, int, long, float, double, char, boolean
2	Phát biểu lặp do, while, for, break, continue	Phát biểu lặp do, while, for, break, continue
3	Phát biểu rẽ nhánh if, else, switch, case, default, break	Phát biểu rẽ nhánh if, else, switch, case, default, break
4	Đặc tính của một phương thức private, protected, public, final, static, abstract, synchronized, volatile	Đặc tính của một phương thức private, protected, public, final, static, abstract, synchronized, volatile
5	Hằng true, false, null	Hằng true, false, null
6	Liên quan tới phương thức return, void	Liên quan tới phương thức return, void
7	Liên quan tới gói package, import	Liên quan tới gói package, import
8	Liên quan tới đối tượng new, extends, implements, class, instance of, this super	Liên quan tới đối tượng new, extends, implements, class, instance of, this super

2.2.1.3. Tên và cách đặt tên

Trong Java, việc đặt tên lớp, các biến, các thuộc tính, các phương thức, hằng... phải tuân theo các quy tắc sau:

- Chỉ được bắt đầu bằng một kí tự (chữ), hoặc một dấu gạch dưới, hoặc một kí tự \$.
- Không có khoảng trắng giữa tên.
- Bắt đầu từ kí tự thứ hai, có thể dùng các kí tự (chữ), chữ số, dấu \$, dấu gạch dưới.
- Không trùng với các từ khoá.
- Có phân biệt chữ hoa và chữ thường

2.2.1.4. Lời giải thích

Trong chương trình cần có có đoạn đặc tả chú thích để giải thích công việc hoặc cách thức thực hiện để tiện cho người đọc hiểu và dễ dàng theo dõi.

Java hỗ trợ 3 loại chú thích:

Chú thích trên một dòng: sử dụng ký hiệu “//”. Tất cả ký tự sau dấu “//” đều là chú thích cho đến cuối dòng.

Ví dụ: // Dòng này đang được chú thích

Chú thích trên nhiều dòng: tương tự như trong C phần nằm giữa cặp dấu /* và */ là chú thích.

Ví dụ:

```
/*  
 * Chú thích này đang được giải  
 * thích trên nhiều dòng.  
 */
```

Chú thích trong tư liệu (javadoc): Đây là loại chú thích đặc biệt được đặt vào những chỗ thích hợp trong chương trình để javadoc có thể đọc và sử dụng để tạo ra tư liệu dạng html cho chương trình. Chúng được đặt vào phần trước định nghĩa các lớp, interface, phương thức và biến. Phần chú thích trong tư liệu được bắt đầu bằng /** và kết thúc bằng */

Ví dụ:

```
/**  
 * Cài đặt lớp KhamPhaCntt  
 * Tác giả: khamphacntt.com.vn  
 * Phiên bản: 1.0  
 */
```

1.7.2. Cấu trúc chung

Dạng cơ bản của một tập tin mã nguồn Java có cấu trúc như sau:

```
package packageName; //Khai báo tên gói, nếu có  
  
import java.awt.*; //Khai báo tên thư viện sẵn có, nếu cần dùng  
  
class className //Khai báo tên lớp  
{  
  
    /* Đây là dòng ghi chú */  
  
    int var; //Khai báo biến  
  
    public void methodName() //Khai báo tên phương thức  
    {  
  
        /* Phần thân của phương thức */  
  
        statement(s); //Lệnh thực hiện  
  
    }  
  
}
```

Một tệp mã nguồn Java có thể có ba phần chính:

- Phần khai báo tên gói (khởi) bằng từ khóa package.
- Phần khai báo thư viện tham khảo bằng từ khóa import
- Phần khai báo nội dung lớp bằng từ khóa class.

Khai báo package

Package được dùng để đóng gói các lớp trong chương trình lại với nhau thành một khối. Đây là một cách hữu hiệu để lưu trữ các lớp gần giống nhau hoặc có cùng một module thành một khối thống nhất.

Cú pháp khai báo tên gói bằng từ khóa package:

```
package <Tên gói>;
```

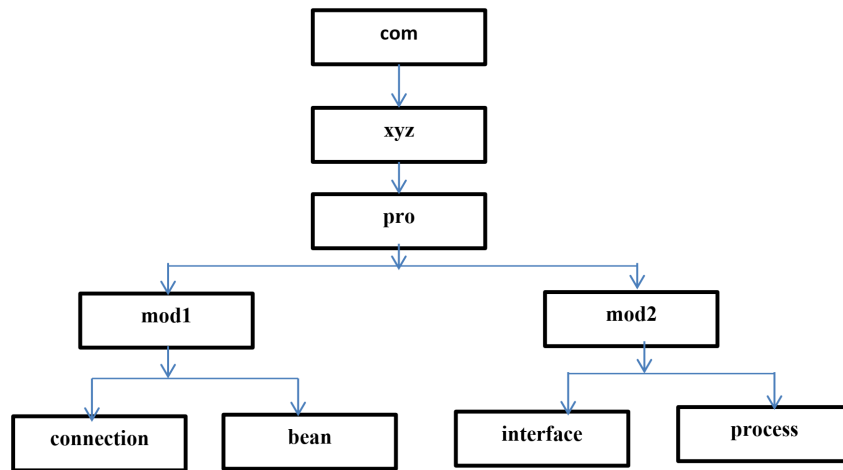
Để đặt tên package trong chương trình, người ta có thể tiến hành như đặt tên thư mục trên ổ đĩa. Nghĩa là bắt đầu bằng tên có phạm vi lớn, cho đến các tên có phạm vi nhỏ, cuối cùng là tên các gói trực tiếp chứa các lớp. Trên thực tế phạm vi đặt tên gói được tiến hành theo thứ tự phạm vi lớn đến nhỏ như sau:

- Tên tổ chức
- Tên công ty
- Tên dự án
- Tên module trong dự án
- Tên các chức năng trong modul.

Ví dụ:

- Tên miền của công ty là syz.com
- Tên dự án là pro
- Dự án có hai modul là mod1 và mod2
- Modul mod1 có hai chức năng là kết nối cơ sở dữ liệu connection và biểu diễn dữ liệu bean.
- Modul mod2 có hai chức năng là giao tiếp interface và xử lý yêu cầu process.

Khi đó, cấu trúc khối của dự án được mô tả như hình:



Hình 1.34. Kiến trúc khối của dự án

Khi đó, trong chức năng bean có lớp User, thì phải khai báo tên khối trong lớp này như sau:

```
package com.xyz.pro.mod1.bean;
```

Ưu điểm của package:

- Cho phép nhóm các lớp vào với nhau thành các đơn vị nhỏ hơn. Việc thao tác trên các đơn vị khối sẽ gọn hơn thao tác trên một tập các lớp.
- Tránh xung đột khi đặt tên lớp. Vì các lớp không cùng package thì có thể đặt tên trùng nhau. Khi số lượng lớp của chương trình quá lớn ta có thể tránh phải đặt tên khác nhau cho các lớp bằng cách đặt chúng vào các package khác nhau.
- Cho phép bảo vệ các lớp. Khi chương trình lớn, việc chia nhỏ chương trình thành các package sẽ thuận lợi hơn cho việc quản lý và phát triển.
- Tên gói còn được dùng để định danh lớp cho ứng dụng.

Lưu ý:

- Dòng lệnh khai báo tên khối phải được đặt đầu tiên trong tệp tin mã chương trình.
- Chỉ được khai báo tối đa một tên khối đối với mỗi tệp mã nguồn Java.
- Các tệp tin của các lớp nằm cùng gói ứng dụng phải được lưu trong cùng một thư mục (tên thư mục là tên khối) theo cấu trúc khối của dự án.
- Tên khối nên đặt theo chữ thường vì tên khối sẽ là tên thư mục tương ứng trong ổ đĩa, tránh nhầm lẫn với tên các tệp tin là tên các lớp của chương trình.
- Khi không phân chia chương trình thành khối (chương trình đơn giản), không cần thiết phải khai báo tên khối ở đầu chương trình.

Khai báo thư viện

Khai báo thư viện để chỉ ra những thư viện đã được định nghĩa sẵn mà chương trình sẽ tham khảo tới. Cú pháp khai báo thư viện với từ khóa import như sau:

```
import <Tên thư viện>;
```

Ví dụ, nếu trong chương trình cần đến các thao tác chuyển đổi kiểu dữ liệu tường minh (từ kiểu string sang kiểu int), thì ta sẽ phải tham khảo thư viện **java.lang**

```
import java.lang.*;
```

Lưu ý:

- Nếu muốn khai báo tham khảo nhiều thư viện, phải khai báo tham khảo mỗi thư viện với một từ khóa import.
- Nếu chỉ tham khảo một vài lớp trong một thư viện, nên chỉ rõ tham khảo lớp nào, thay vì phải khai báo tham khảo cả gói (bằng kí hiệu “*”) vì tham khảo cả gói sẽ tăng kích thước tệp tin class sau khi biên dịch.
- Nếu không tham khảo thư viện nào, không nhất thiết phải khai báo các tham khảo với từ khóa import.

Khai báo lớp

Phần thứ ba là phần khai báo lớp và nội dung của lớp, phần này luôn bắt buộc phải có đối với một tệp mã nguồn Java:

- Khai báo tên lớp với từ khóa class.
- Khai báo các thuộc tính của lớp.
- Khai báo các phương thức của lớp.

Ví dụ chương trình java cơ bản cho phép hiển thị thông điệp “Hello World”:

```
package vidu;
```

```
//Đây là chương trình “Hello.java”
```

```
class Hello
```

```
{
```

```
Public static void main(String args[])
```

```
{
```

```
System.out.println("Hello World");
```

```
}
```

```
}
```

Để biên dịch mã nguồn, ta sẽ dùng trình biên dịch javac. Trình biên dịch xác định tên của file nguồn tại dòng lệnh như mô tả dưới đây (giả sử đang ở thư mục chứa package vidu và biến môi trường PATH đã được thiết lập đúng qui cách):

>javac vidu/Hello.java

Trình dịch javac tạo ra file Hello.class chứa các mã “bytecodes”. Những mã này chưa thể thực thi được. Để chương trình thực thi được ta cần dùng trình thông dịch “java interpreter” với lệnh java. Lệnh được thực hiện như sau:

>javac vidu/Hello

Kết quả sẽ hiển thị trên màn hình như sau:

Hello World

Phân tích chương trình đầu tiên:

- Trong Java, tất cả mã lệnh đều phải được tổ chức vào trong một lớp nhất định. Do đó, mỗi tệp tin mã nguồn xác định ít nhất một lớp Java và tên tệp tin phải trùng với tên lớp. Java phân biệt chữ hoa và chữ thường, cho nên tên tệp tin của chương trình trên phải trùng với tên lớp: Hello.java.
- `package vidu;` Đây là dòng khai báo tên khối của chương trình, tệp tin Hello.java sẽ nằm trong thư mục: ../vidu/.
- Chương trình này không tham khảo thư viện nào nên không cần lệnh import nào. // Đây là chương trình “First.java”. Ký hiệu “//” dùng để chú thích dòng lệnh. Trình biên dịch sẽ bỏ qua dòng chú thích này. Java hỗ trợ hai loại chú thích:
 - o Loại chú thích trên một dòng, dùng “//”. Trình biên dịch sẽ bỏ qua nội dung bắt đầu từ kí hiệu “//” cho đến hết dòng lệnh chứa nó.
 - o Loại chú thích trên nhiều dòng có thể bắt đầu với “/*” và kết thúc với “*/”. Trình biên dịch sẽ bỏ qua nội dung nằm giữa hai kí hiệu này.
- Dòng kế tiếp khai báo lớp có tên First: Bắt đầu với từ khoá class, kế đến là tên lớp:
`class First`
- Một định nghĩa lớp nằm trọn vẹn giữa hai ngoặc móc mở “{” và đóng “}”. Các ngoặc này đánh dấu bắt đầu và kết thúc một khối lệnh.
- `public static void main(String args[ ])`
- Đây là phương thức chính, từ đây chương trình bắt đầu việc thực thi của mình. Tất cả các ứng dụng java đều sử dụng một phương thức main này.
- Từ khoá public là một chỉ định truy xuất. Nó cho biết thành viên của lớp có thể được truy xuất từ bất cứ đâu trong chương trình.
- Từ khoá static cho phép main được gọi tới mà không cần tạo ra một thể hiện (instance) của lớp. Nó không phụ thuộc vào các thể hiện của lớp

được tạo ra.

- Từ khoá void thông báo cho máy tính biết rằng phương thức sẽ không trả lại bất cứ giá trị nào khi thực thi chương trình.
- String args[] là tham số dùng trong phương thức main. Khi không có một thông tin nào được chuyển vào main, phương thức được thực hiện với các dữ liệu rỗng – không có gì trong dấu ngoặc đơn.
- System.out.println(“Hello World”); Dòng lệnh này hiển thị chuỗi “Hello World” được truyền vào lên màn hình.

1.7.3. Khái niệm, Phân loại kiểu dữ liệu

2.2.3.1. Khái niệm kiểu dữ liệu

- Dữ liệu : là tất cả những gì được máy tính xử lý. Dữ liệu được chứa trong bộ nhớ máy tính với 1 số lượng ô nhớ nhất định, tính theo đơn vị byte.
- Kiểu dữ liệu được định nghĩa với 2 điểm chính:
 - o Một tập hợp các giá trị mà 1 biến thuộc kiểu đó có thể nhận được.
 - o Trên đó xác định 1 số phép toán.

2.2.3.2. Phân loại kiểu dữ liệu

Trong Java, kiểu dữ liệu được chia thành hai loại:

- Các kiểu dữ liệu cơ bản
- Các kiểu dữ liệu đối tượng

Bảng 1.5. Kiểu dữ liệu cơ bản

Kiểu dữ liệu	Miêu tả
byte	Dùng để lưu dữ liệu kiểu số nguyên có kích thước một byte (8 bit). Phạm vi biểu diễn giá trị từ -128 đến 127. Giá trị mặc định là 0.
char	Dùng để lưu dữ liệu kiểu kí tự hoặc số nguyên không âm có kích thước 2 byte (16 bit). Phạm vi biểu diễn giá trị từ 0 đến u\ffff. Giá trị mặc định là 0.
boolean	Dùng để lưu dữ liệu chỉ có hai trạng thái đúng hoặc sai (độ lớn chỉ có 1 bit). Phạm vi biểu diễn giá trị là {“True”, “False”}. Giá trị mặc định là False.
short	Dùng để lưu dữ liệu có kiểu số nguyên, kích cỡ 2 byte (16 bit). Phạm vi biểu diễn giá trị từ -32768 đến 32767. Giá trị mặc định là 0.
int	Dùng để lưu dữ liệu có kiểu số nguyên, kích cỡ 4 byte (32 bit). Phạm vi biểu diễn giá trị từ -2,147,483,648 đến 2,147,483,647. Giá trị mặc định là 0.
float	Dùng để lưu dữ liệu có kiểu số thực, kích cỡ 4 byte (32 bit). Giá trị mặc định là 0.0f.
double	Dùng để lưu dữ liệu có kiểu số thực có kích thước lên đến 8 byte. Giá trị mặc định là 0.00d.
long	Dùng để lưu dữ liệu có kiểu số nguyên có kích thước lên đến 8 byte. Giá trị mặc định là 0l.

Bảng 1.5. Các kiểu dữ liệu đối tượng:

Kiểu dữ liệu	Miêu tả
Array	Một mảng của các dữ liệu cùng kiểu.

class	Dữ liệu kiểu lớp đối tượng do người dùng định nghĩa. Chứa tập các thuộc tính và phương thức.
interface	Dữ liệu kiểu lớp giao tiếp do người dùng định nghĩa. Chứa các phương thức của giao tiếp.

1.7.4. Biến, Hằng, Toán tử

Dữ liệu được chứa trong máy tính có thể là biến hoặc hằng

- Biến là đại lượng có thể thay đổi được giá trị.
- Hằng là đại lượng có giá trị không thay đổi.

Biến và hằng được lưu trữ trong bộ nhớ và thường được đặt tên để cho dễ sử dụng, thay vì phải sử dụng địa chỉ ô nhớ cũng như giá trị cụ thể của chúng rất khó sử dụng.

Khai báo biến

Cú pháp khai báo biến:

Kiểu_dữ_liệu_tên_biến;

Ví dụ khai báo biến x kiểu số nguyên :

int x;

Phạm vi hoạt động của biến

Một biến có phạm vi hoạt động trong toàn bộ khối lệnh mà nó được khai báo. Một khối lệnh bắt đầu bằng dấu "{" và kết thúc bằng dấu "}":

Khai báo hằng

Cú pháp khai báo hằng:

final kiểu_dữ_liệu_tên_hằng = giá_trị_cần_gán

Ví dụ khai báo hằng a kiểu số nguyên có giá trị bằng 2:

final int a = 2;

Các phép toán

- Java cung cấp các dạng phép toán (toán tử) sau:
- Toán tử số học
- Toán tử bit
- Toán tử quan hệ
- Toán tử logic
- Toán tử điều kiện
- Toán tử gán

1.7.5. Lệnh vào ra và điều khiển

Các lệnh vào ra

Java cung cấp sẵn một bộ thư viện chứa các hàm xử lý vào/ra trên các thiết bị chuẩn và các thiết bị ngoại vi là java.io.

Miêu tả chi tiết về các lệnh vào ra trong Java:

<http://docs.oracle.com/javase/tutorial/essential/io/index.html>

Các lệnh điều khiển

Java cung cấp hai loại cấu trúc điều khiển:

Điều khiển rẽ nhánh

- Mệnh đề if-else
- Mệnh đề switch-case

Vòng lặp (Loops)

- Vòng lặp while
- Vòng lặp do-while
- Vòng lặp for

1.7.6. Lớp, đối tượng

Trong ngôn ngữ Java, lớp là một đơn vị mẫu có chứa dữ liệu và mã lệnh liên quan đến một thực thể nào đó. Khi xây dựng một lớp, thực chất là tạo ra một kiểu dữ liệu. Kiểu dữ liệu mới này được sử dụng để xác định các biến mà ta thường gọi là “đối tượng”. Đối tượng là các thể hiện (instance) của lớp. Tất cả các đối tượng đều thuộc về một lớp có chung đặc tính và hành vi. Mỗi lớp xác định một thực thể, trong khi đó mỗi đối tượng là một thể hiện thực sự.

Hay nói cách khác : Java coi lớp là một khuôn mẫu (Template) của một đối tượng, trong đó lớp chứa các thuộc tính và các phương thức hoạt động của đối tượng.

Khai báo lớp

Một lớp được khai báo với cú pháp:

```
<tính      chất>      class      <tên      lớp>
{
}
```

Lớp trong java có ba tính chất đặc trưng bởi ba từ khóa:

- **public**: Lớp thông thường, có thể được truy cập từ các gói (package) khác, public là giá trị mặc định cho tính chất của lớp.
- **final**: Khai báo lớp hằng, lớp này không thể tạo dẫn xuất. Tức là không có lớp nào kế thừa được từ các lớp có tính chất final.
- **abstract**: Khai báo lớp trừu tượng, lớp này chỉ được phép chứa các phương thức trừu tượng. Hơn nữa, không thể tạo các thể hiện (Instance) của các lớp trừu tượng bằng toán tử new như các lớp thông thường.

Ví dụ khai báo một lớp thông thường với kiểu mặc định là public với dòng khai báo

```
class Person
{ }
```

Sử dụng lớp

Lớp được sử dụng khi chương trình cần một đối tượng có kiểu của lớp đó. Khi đó, đối tượng được khai báo dựa vào toán tử new:

```
<tên lớp> <tên đối tượng> = new <tên lớp>();
```

Ví dụ, muốn tạo một đối tượng có kiểu là lớp Person ta dùng lệnh sau:

```
Person myClass = new Person();
```

Khai báo thuộc tính của lớp

```
<tính chất> <kiểu dữ liệu> <tên thuộc tính>;
```

Khai báo phương thức của lớp

```
<tính chất> <kiểu trả về> <tên phương thức> ([<các tham số>]) [throws <các ngoại lệ>]  
{  
}
```

Phương thức khởi tạo của lớp

Phương thức khởi tạo (Constructor) được dùng để khởi tạo một thể hiện cụ thể của một lớp, nghĩa là gán các giá trị khởi đầu cho các thuộc tính, nếu có, và tạo ra một đối tượng cụ thể. Phương thức khởi tạo phải cùng tên với lớp.

1.7.7. Kế thừa và đa hình

1.7.7.1. Kế thừa

Sự kế thừa được sử dụng khi muốn tạo một lớp mới từ một lớp đã biết. Khi đó, tất cả các thuộc tính và phương thức của lớp cũ đều trở thành thuộc tính và phương thức của lớp mới. Lớp cũ được gọi là lớp cha, lớp mới được gọi là lớp con.

Khai báo lớp kế thừa

```
<thuộc tính> <tên lớp con> extends <tên lớp cha>  
{  
}
```

Khai báo phương thức nạp chồng

Khi muốn thay đổi nội dung của các phương thức được kế thừa từ lớp cha, ta dùng cách khai báo phương thức nạp chồng. Thực ra là khai báo lại một phương thức mới có cùng tên và kiểu với một phương thức đã có trong lớp cha.

Ví dụ:

```
class Employee extends Person  
{
```

```

public float salary;
// Phương thức khởi dựng
public Employee(String name, int age, float salary)
{
    super(name, age);
    this.salary = salary;
}
// Khai báo nạp chồng
public void show()
{
    System.out.println( name + "has a salary of" +
    salary + "$/month");
}
}

```

Quy tắc truy nhập trong kế thừa

Các quy tắc này quy định khả năng truy nhập của lớp con đối với các thuộc tính và phương thức của lớp cha:

- **private:** chỉ được truy nhập trong phạm vi lớp cha, lớp con không truy nhập được. Tất cả các lớp ngoài lớp cha đều không truy nhập được.
- **protected:** lớp con có thể truy nhập được. Tất cả các lớp không kế thừa từ lớp cha đều không truy nhập được.
- **final:** lớp con có thể sử dụng được nhưng không thể khai báo nạp chồng được.
- **public:** lớp con có thể sử dụng và nạp chồng được. Tất cả các lớp bên ngoài đều sử dụng được.

Lớp trừu tượng

Lớp trừu tượng là một dạng lớp đặc biệt, trong đó các phương thức chỉ được khai báo ở dạng khuôn mẫu (template) mà không được cài đặt chi tiết. Việc cài đặt chi tiết các phương thức chỉ được thực hiện ở các lớp con kế thừa lớp trừu tượng đó.

Lớp trừu tượng được sử dụng khi muốn định nghĩa một lớp mà không thể biết và định nghĩa ngay được các thuộc tính và phương thức của nó.

Ví dụ về khai báo

```

[public] abstract class <tên lớp>
{
}

```

1.7.7.2. Đa hình

Nạp chồng

Java cho phép trong cùng một lớp, có thể khai báo nhiều phương thức có cùng tên. Nạp chồng là hiện tượng các phương thức có cùng tên. Có hai kiểu nạp chồng trong Java:

Các phương thức của cùng một lớp có cùng tên. Khi hai phương thức của một lớp có cùng tên thì bắt buộc chúng phải có:

- Hoặc danh sách các tham số khác nhau.
- Hoặc kiểu trả về khác nhau.
- Hoặc kết hợp hai điều kiện trên.

Đa hình

Đa hình là việc triệu gọi đến các phương thức nạp chồng của đối tượng. Khi một phương thức nạp chồng được gọi, chương trình sẽ dựa vào kiểu các tham số và kiểu trả về để gọi phương thức của đối tượng cho phù hợp.

Chương trình sau minh họa việc khai báo nhiều hàm add() để cộng hai số hoặc cộng hai chuỗi kí tự.

```
public class Operator
{
    // Cộng hai số nguyên
    public int add(int x, int y)
    {
        return (x + y);
    }
    // Cộng hai số thực
    public float add(float x, float y)
    {
        return (x + y);
    }
    // Cộng hai chuỗi kí tự
    public String add(String a, String b)
    {
        return (a + b);
    }
    // Hàm main
    public static void main(String args[])
    {
        Operator myOperator = new Operator();
        System.out.println("The (5+19) is " +
            myOperator.add(5, 19));
        System.out.println("The (\"ab\" + \"cd\") is \""
            + myOperator.add("ab", "cd") + "\"");
    }
}
```

Chương trình 4.13 sẽ hiển thị ra hai dòng thông báo:

```
The (5+19) is 24
The ('ab' + 'cd') is 'abcd'
```

Trong lớp Operator có hai phương thức cùng tên và cùng có hai tham số đầu vào là add(). Khi chương trình thực thi lệnh myOperator.add(5, 19), chương trình sẽ tự đối chiếu các kiểu tham số, thấy 5 và 19 có dạng gần với kiểu int nhất, nên phương thức add(int, int) sẽ được gọi và trả về giá trị là 24.

Khi chương trình thực thi lệnh `myOperator.add("ab", "cd")`, chương trình sẽ tự đổi chiều các kiểu tham số, thấy 'ab' và 'cd' có dạng gần với kiểu String nhất, nên phương thức `add(String, String)` sẽ được gọi và trả về giá trị là 'abcd'.

1.7.8. Thư viện cơ bản

Java chuẩn cung cấp một số thư viện như sau:

- `java.lang`: cung cấp các hàm thao tác trên các kiểu dữ liệu cơ bản, xử lý lỗi và ngoại lệ, xử lý vào ra trên các thiết bị chuẩn như bàn phím và màn hình.
- `java.applet`: cung cấp các hàm cho xây dựng các applet
- `java.awt`: cung cấp các hàm cho xây dựng các ứng dụng đồ họa với các thành phần giao diện multi media
- `java.io`: cung cấp các hàm xử lý vào/ra trên các thiết bị chuẩn và các thiết bị ngoại vi.
- `java.util`: cung cấp các hàm tiện ích trong xử lý liên quan đến các kiểu dữ liệu có cấu trúc như Date, Stack, Vector.

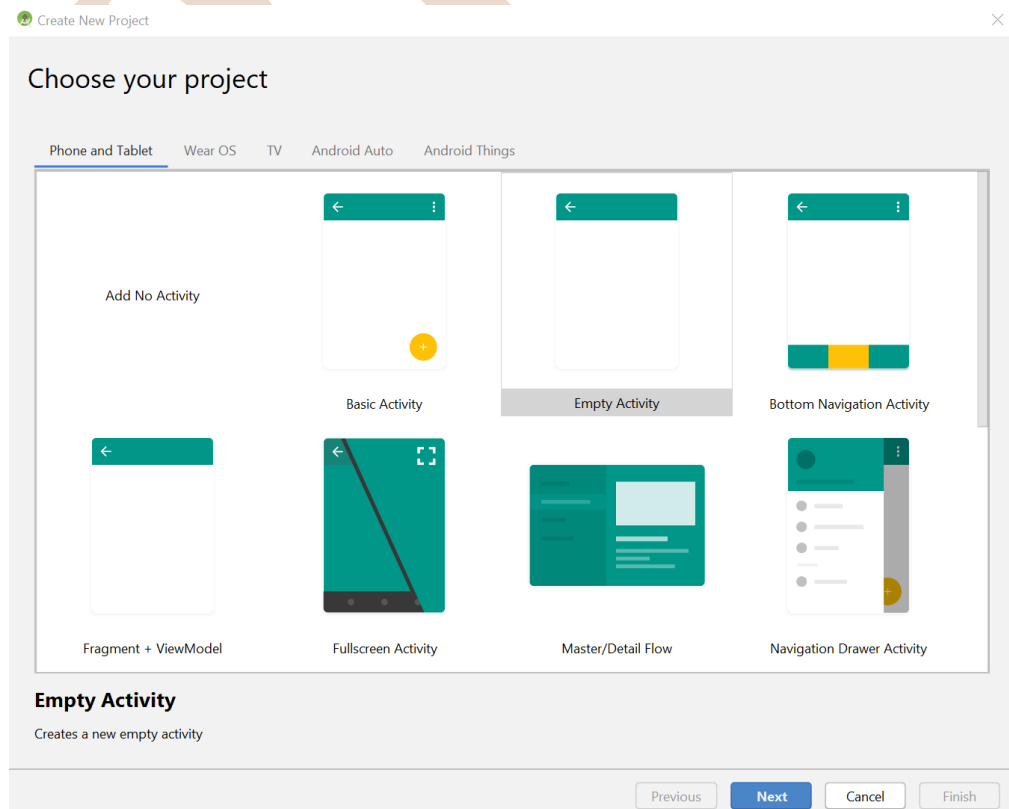
1.8. TIẾN TRÌNH LẬP TRÌNH ỨNG DỤNG

Trong phần này sẽ hướng dẫn tạo ra ứng dụng Android "Hello world" theo 4 bước trong quy trình lập trình ứng dụng di động.

1.8.1. Khởi chạy ứng dụng

Bật visual studio vào chọn New-> NewProject

Chọn các Loại Project khác nhau: Basic Activity, Empty Activity, Bottom Navigation Activity...



Hình 1.35. Màn hình tạo mới một Android Project

Create New Project

Configure your project

Name
HelloWorld

Package name
com.example.helloworld

Save location
~\Works\2018\Khoa CNTT\Lap Trinh Di Dong\2018.2019\Project\HelloWorld

Language
Java

Minimum API level
API 14: Android 4.0 (IceCreamSandwich)

Basic Activity

Creates a new basic activity with an app bar.

Information: Your app will run on approximately **100%** of devices.
[Help me choose](#)

☐ This project will support instant apps

☒ Use androidx.* artifacts

Warning: project location should not contain whitespace, as this can cause problems with the NDK tools.

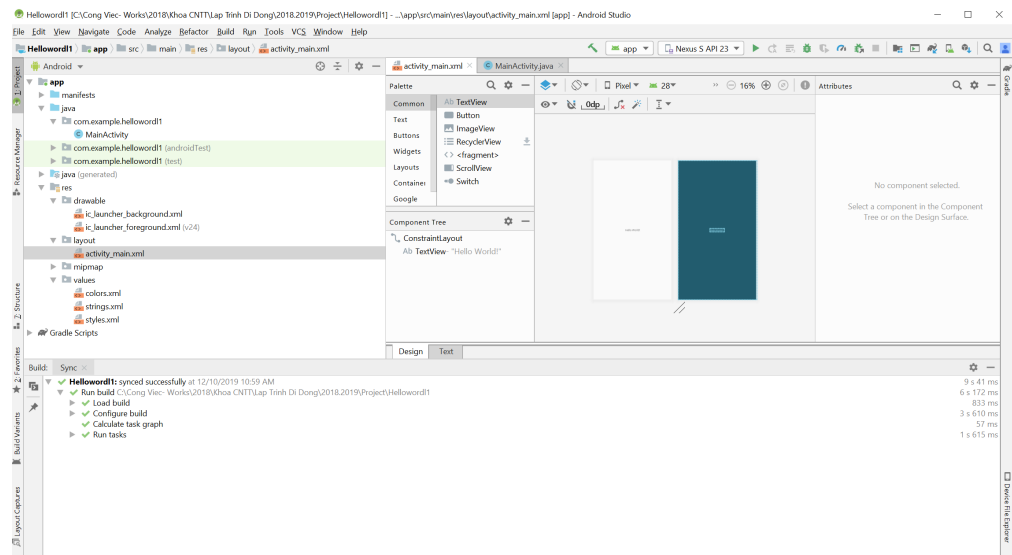
[Previous](#) [Next](#) [Cancel](#) [Finish](#)

Hình 1.36. Màn hình nhập thông số khi tạo mới một Android Project

- **Application Name:** tên ứng dụng.
- **Package Name:** nên viết thường hết và phải ít nhất có 1 dấu chấm ngăn cách. Thường tên ứng dụng đặt như thế nào thì tên gói sẽ tương ứng là com.exemple.ApplicationName
- **Save Location:** Chọn vị trí lưu dự án
- **Minimum API level:** chọn giới hạn API thấp nhất mà ứng dụng có thể cài đặt (Điện thoại đó có API $\geq$ Minimum Required).
- **Language:** Chọn ngôn ngữ thực hiện là Java

Chọn Next cho tới khi Finish.

Bấm Finish, quan sát Package Explorer trong Android Studio



Hình 1.37. Cấu trúc cây trong Android project.

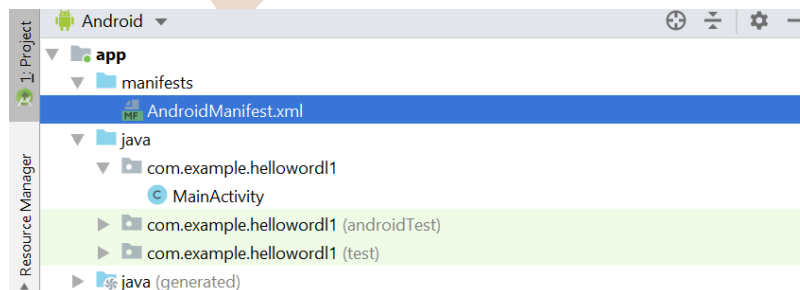
Giải thích ý nghĩa của các file / folder được tạo ra khi ứng dụng được khởi tạo:

Dự án android chứa các loại mô-đun ứng dụng, tệp mã nguồn và tệp tài nguyên khác nhau được liệt kê dưới đây:

1. Manifests Folder
2. Java Folder
3. res (Resources) Folder
 - Drawable Folder
 - Layout Folder
 - Mipmap Folder
 - Values Folder
4. Gradle Scripts

Thư mục Manifests

Thư mục Manifests chứa file AndroidManifest.xml khi tạo ra ứng dụng android. File này chứa các thông tin về ứng dụng như phiên bản android, siêu dữ liệu (metadata), và các thành phần khác của ứng dụng. Nó đóng vai trò trung gian giữa hệ điều hành và ứng dụng.



Hình 1.38. Thư mục Manifest

File AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.helloworld1">

    <application
```

```

        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>

```

Ý nghĩa của các thành phần trong file AndroidManifest.xml ở ví dụ trên như sau:

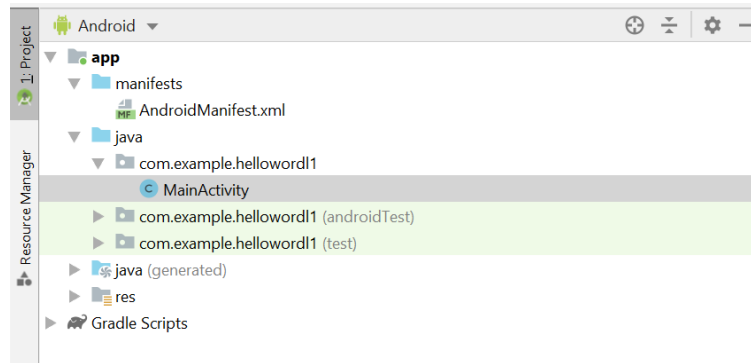
- Node gốc bắt buộc của bất kỳ file AndroidManifest.xml nào là <manifest>
- Tên package của ứng dụng là multi.com, , được thiết lập thông qua thuộc tính package của <manifest> (package = "multi.com")
- Version code của ứng dụng là 1, được thiết lập thông qua thuộc tính versionCode (android:versionCode = "1"). Giá trị của version code cho biết ứng dụng đang ở phiên bản thứ mấy, thay đổi giá trị này khi muốn nâng cấp ứng dụng.
- Phần tử <application>...</application> dùng để thiết lập các thông tin mà ứng dụng sử dụng qua các thuộc tính . Cụ thể:
 - o Icon ứng dụng là file ảnh ic_launcher.png chứa trong thư mục drawable (android:icon="@drawable/ic_launcher").
 - o Tên của ứng dụng được thiết lập thông qua thuộc tính label (android:label="@string/app_name"), khai báo @string/app_name nhằm tham chiếu tới chuỗi app_name có giá trị "SampleProject" trong file string.xml (ở thư mục res/values).
 - o Theme của ứng dụng được thiết lập thông qua thuộc tính theme (android:theme="@style/AppTheme"), khai báo @style/AppTheme nhằm tham chiếu tới style có tên AppTheme trong file styles.xml (ở thư mục res/values).
 - o Phần tử <activity>...</activity> dùng để khai báo activity sẽ được gọi trong ứng dụng. Phần tử activity này chứa phần tử con <intent-filter> với khai báo: Tên action của intent filter là android.intent.action.MAIN cho biết activity này được thực thi đầu tiên khi chạy ứng dụng; Tên category của intent filter là android.intent.category.LAUNCHER cho phép hiển thị icon của ứng dụng. Các activity khác không được thực thi đầu tiên khi chạy ứng dụng thì khi khai báo nó không cần khai báo phần tử con <intent-filter>.

Ngoài ra, tùy yêu cầu của ứng dụng, lập trình viên sẽ lựa chọn các thành phần

cần thiết trong file AndroidManifest.xml để thiết lập cho ứng dụng. Thông tin chi tiết về các thành phần này trên website: <http://developer.android.com/guide/topics/manifest/manifestintro.html>

Thư mục Java

Thư mục chứa toàn bộ các file mã nguồn được tạo ra trong suốt quá trình phát triển bao gồm cả thư mục Test. Khi tạo ra một dự án mới, mặc định lớp MainActivity sẽ được tạo ra trong gói com.example.helloworld1



Hình 1.39. Thư mục Java

```
package com.example.helloworld1;

import androidx.appcompat.app.AppCompatActivity;

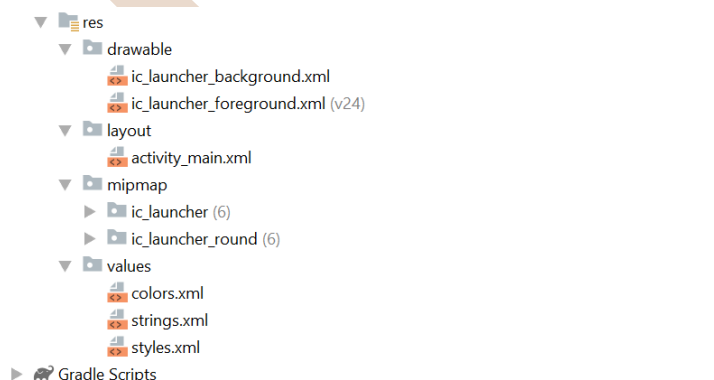
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Thư mục Tài nguyên (res)

Thư mục tài nguyên là thư mục quan trọng nhất vì nó chứa tất cả các nguồn không phải mã như hình ảnh, bố cục XML, chuỗi UI cho ứng dụng Android.



Hình 1.40. Thư mục res

Thư mục res/drawable

Chứa các loại hình ảnh khác nhau được sử dụng để phát triển ứng dụng

Thư mục res/layout

Thư mục Layout chứa toàn bộ các file XML được dùng để định nghĩa các giao diện ứng dụng. Nó bao gồm tệp activity_main.xml thông thường để xác định giao diện chính.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Thư mục res/mipmap folder

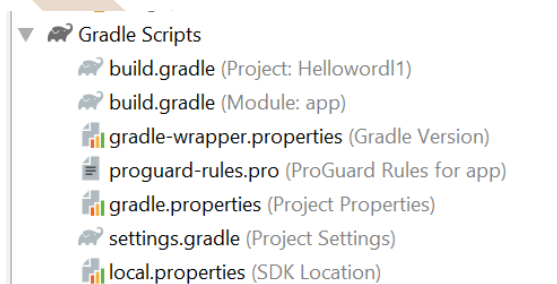
Thư mục chứa file launcher.xml xác định các icon nào được sử dụng để hiển thị ứng dụng trong màn hình chính của điện thoại (home screen). Nó chứa nhiều các độ phân giải khác nhau của icons dựa trên kích cỡ của thiết bị như hdpi, mdpi, xhdpi.

Thư mục res/values folder

Thư mục Values chứa một số các file xml như chuỗi (string), độ mờ (dimens), màu sắc và định nghĩa kiểu. Một trong những tệp quan trọng nhất là tệp String.xml chứa tài nguyên.

```
<resources>
    <string name="app_name">HelloWorld</string>
    <string name="checked">Checked</string>
    <string name="unchecked">Unchecked</string>
</resources>
```

Thư mục Gradle Scripts



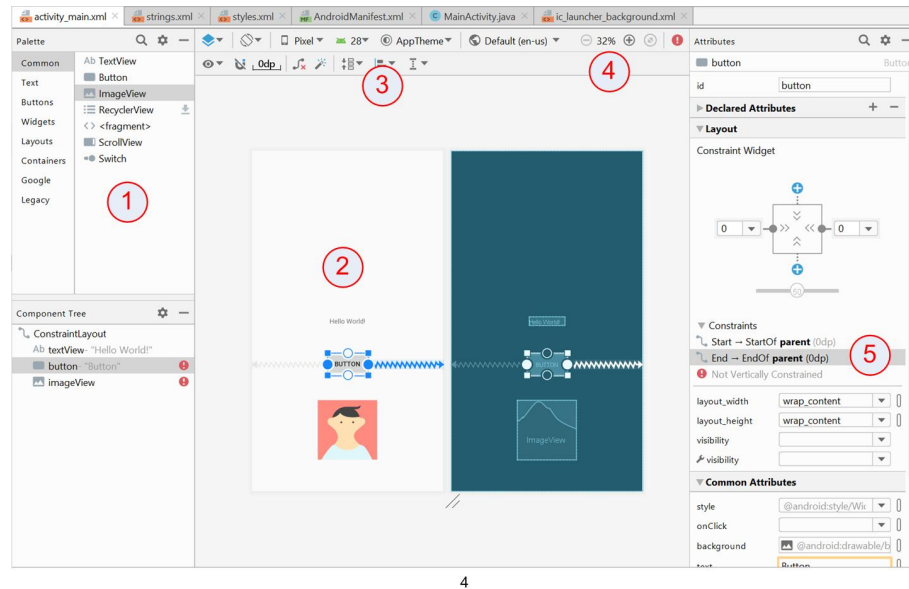
Hình 1.41. Thư mục Gradle Scripts

Gradle có nghĩa là hệ thống sẽ tự xây dựng và thư mục chứa một số các tệp được sử dụng để xác định cấu hình build được áp dụng cho các modules trong ứng dụng. Trong build.gradle (Project) có các script và trong build.gradle (Module) có các plugins và các thực thi (implementations) được sử dụng để cấu hình build.

1.8.2. Viết chương trình

Tạo giao diện ứng dụng

Nhấp đúp chuột vào activity_main.xml để thiết kế giao diện ứng dụng:



Hình 1.42. Giao diện thiết kế ứng dụng

Quan sát màn hình ở trên có 2 phần (GraphicalLayout và activity_main.xml), khi chỉnh sửa 1 trong 2 phần thì phần còn lại sẽ cập nhật tương ứng, với:

- Vùng số 1: chính là nơi chứa các đối tượng hiển thị giao diện (control, layout, component...). Bạn muốn sử dụng đối tượng nào thì kéo thả nó vào vùng số 2. Ví dụ trong hình trên, kéo thả một Button vào và đặt Id nó là "btnXinChao".
- Vùng số 2: là giao diện, nơi mà bạn cần thiết kế.
- Vùng số 3: cho phép thiết kế theo chiều đứng hay chiều ngang.
- Vùng số 4: chức năng zoom in, zoom out để dễ thiết kế.
- Vùng số 5: nơi thiết lập các thuộc tính cho các view được kéo thả vào giao diện.

File activity_main.xml cho phép tạo giao diện bằng mã xml

```
?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
```

```

app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent" />

<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    tools:layout_editor_absoluteY="434dp" />

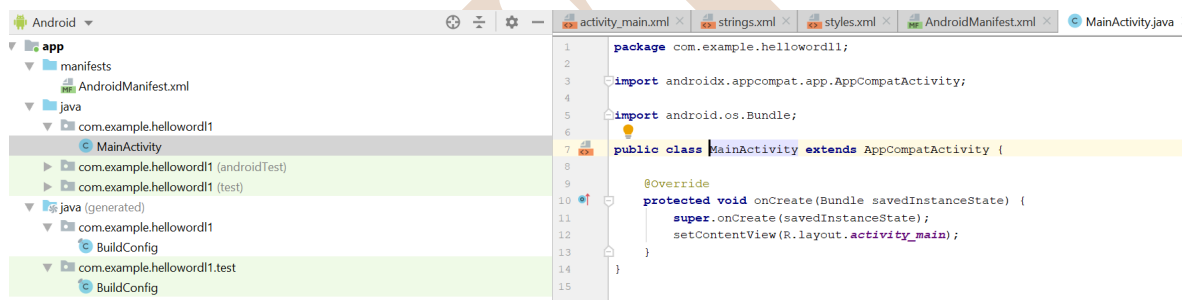
<ImageView
    android:id="@+id/imageView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    tools:layout_editor_absoluteY="536dp"
    tools:srcCompat="@tools:sample/avatars" />

</androidx.constraintlayout.widget.ConstraintLayout>

```

Viết code xử lý:

- Nhấp đúp chuột vào MainActivity.java, viết code xử lý cho ứng dụng như sau:



Hình 1.43. Viết code java xử lý

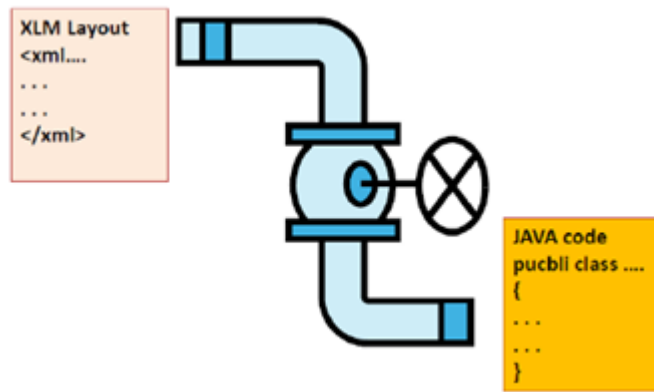
Trong hàm onCreate có một số lệnh:

- setContentView(R.layout.activity_main): đây là lệnh dùng để kết nối giao diện thiết kế activity_main.xml vào Activity. Với activity_main là định danh của activity_main.xml, được tự động sinh ra trong class Layout của file R.java.
- findViewById(R.id.btnXinChao): truy xuất đối tượng Button trên giao diện có định danh btnXinChao lấy từ R.java.

1.8.3. Build ứng dụng

Bấm chuột phải vào ứng dụng trong Package Explorer, chọn Run As, chọn Android Application. Khi đó Android Project sẽ được tự động biên dịch và chuyển qua Android Executable (.dex).

Tại thời điểm biên dịch ứng dụng, Android sẽ dịch mã XML thành lớp giao diện Android dưới dạng mã Java (Java bytecode).



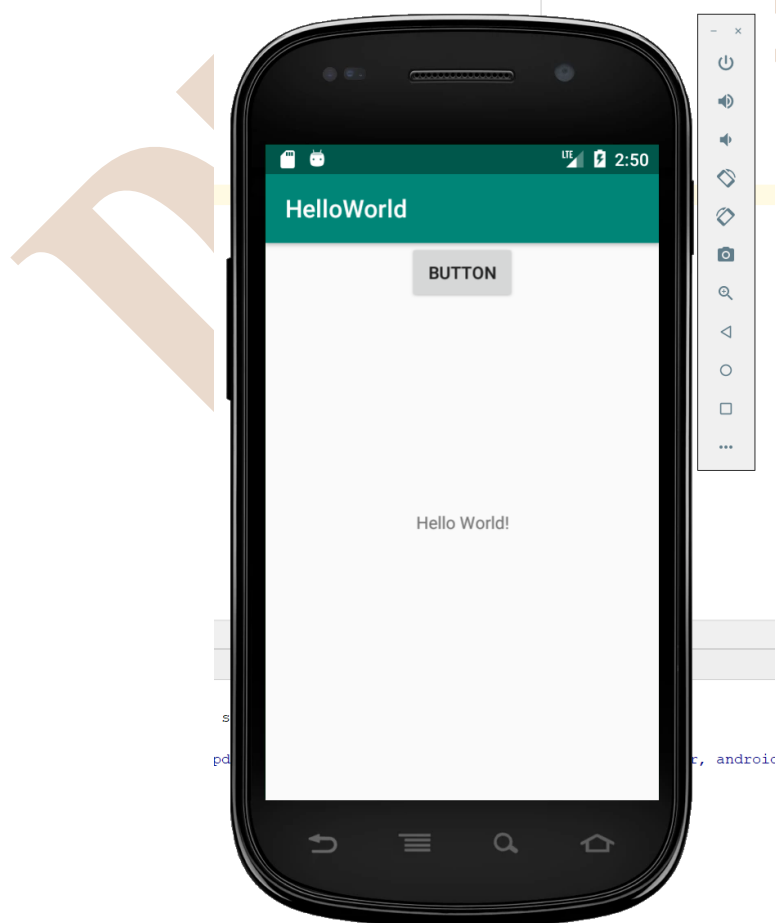
Hình 1.44. Cơ chế dịch XML thành Java code

1.8.4. Chạy chương trình và Debug

2.3.4.1. Chạy chương trình

Sau bước biên dịch, ứng dụng sẽ được đóng gói thành tập tin .apk. Đồng thời dựa vào loại thiết bị/ emulator, file .apk sẽ được tải và cài đặt vào thiết bị android được chọn.

Tại thời điểm chạy ứng dụng (runtime): đầu tiên phương thức onCreate() của activity được thực thi cho phép giao diện ứng dụng được kết nối vào activity qua hàm setContentView().



Hình 1.45. Ứng dụng Hello chạy trên Android Virtual Device

Chọn nút Home để quay về màn hình Home của điện thoại. Sau đó chọn nút

launcher để xem danh sách các ứng dụng được cài đặt trên thiết bị, trong đó có ứng dụng SampleProject vừa tạo.



Hình 1.46. Ứng dụng Hello cài đặt trên AVD.

2.3.4.2. Gỡ lỗi (debug)

Công cụ Android Studio Debugger cho phép theo dõi ứng dụng theo thời gian thực. Ngoài ra nó còn cung cấp thêm thông tin về thiết bị đã kết nối hoặc máy ảo đang được sử dụng.

Breakpoint

Breakpoint là một thuật ngữ lập trình có nghĩa là vị trí ở dòng code nào mà ta muốn chương trình khi sẽ dừng tại đó.

Chúng ta có thể chèn thêm một breakpoint (hoặc nhiều breakpoint tùy ý muốn) ở vị trí nào mà chúng ta muốn chương trình khi chạy đến đấy thì dừng lại. Có nhiều cách để thêm breakpoint.

Cách thêm Breakpoint

Click vào vị trí muốn thêm breakpoint. Sau đó chọn Run > Toggle Line Breakpoint. Một vòng tròn màu đỏ sẽ xuất hiện trực tiếp ngay bên trái của dòng code này. Vùng vòng tròn màu đỏ này nằm trong người ta gọi là gutter.

```

1 package com.example.helloworld1;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.os.Bundle;
6 import android.widget.Button;
7
8 public class MainActivity extends AppCompatActivity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) { savedInstanceState: nu
12         super.onCreate(savedInstanceState); savedInstanceState: null
13         setContentView(R.layout.activity_main);
14         Button btn= (Button) findViewById(R.id.button);
15         btn.setText("Demo");
16
17     }
18 }
19

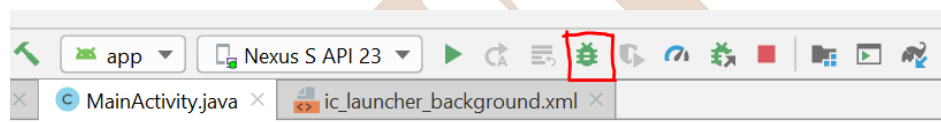
```

Hình 1.47. Ví dụ về thêm Breakpoint

Để xóa điểm breakpoint, có thể chọn dòng tương tự, sau đó chọn Run > Toggle Line Breakpoint một lần nữa để tắt nó đi. Hoặc đơn giản là click đúp vào vòng tròn đỏ.

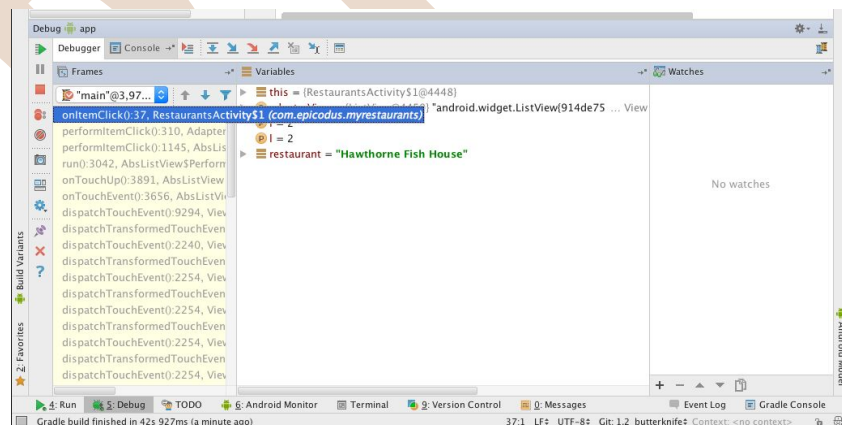
Cách sử dụng Breakpoints

Sau đó, có thể chạy debug bằng cách chọn icon “Debug” ở trên thanh công cụ. Việc này sẽ chạy ứng dụng ở chế độ debug lên thiết bị hay máy ảo.



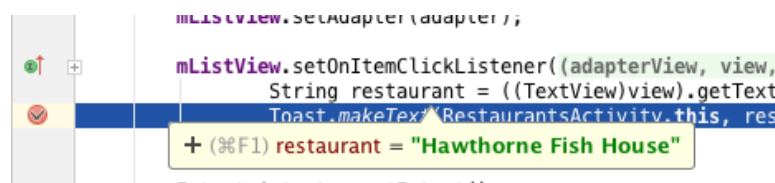
Hình 1.48. Debug

Một khi ứng dụng chạy, bạn mở đến màn hình ứng dụng mà dòng code được đặt breakpoint trước đó có thể được gọi. Nếu dòng code đó được thực hiện, ứng dụng sẽ dừng lại và màn hình console debugger sẽ xuất hiện.



Hình 1.49. Giao diện Debug

Khi ứng dụng dừng lại tại breakpoint, để có thể kiểm tra giá trị của các biến thì chúng ta có một cách. Cách đơn giản là di chuột qua các biến để thấy các giá trị hiện tại của nó:

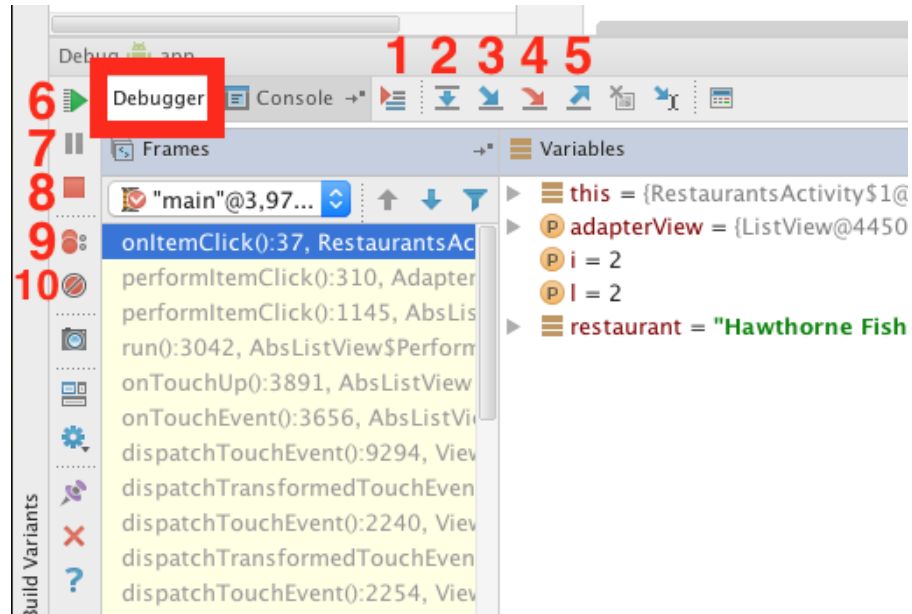


Hình 1.50. Chi tiết Debug

Nếu chúng ta click vào biểu tượng “+” ở bên trái của cửa sổ màu vàng xuất hiện khi chúng ta di chuột qua một biến, chúng ta có thể xem thêm chi tiết hơn

Điều hướng các dòng code khi debug

Không chỉ cần xem tại vị trí debug, ta cần xem code chạy có đúng không. Do đó cần phải xem chi tiết các bước trong Debug.



Hình 1.51. Các nút cần nhớ khi Debug

1. Nút hiển thị Breakpoint đang active

Nút này có tác dụng đặt con trỏ trở lại vào vị trí mà đang debug lỗi.

2. Step Over

Nút này sẽ giúp debug nhảy xuống dòng code tiếp theo

3. Step Into

Nút này sẽ nhảy vào bên trong hàm

4. Force Step Into

Nút này sẽ cho phép nhảy thẳng đến dòng đầu tiên bên trong của hàm được gọi

5. Thoát ra ngoài

Nhảy ra khỏi hàm đang debug.

6. Tiếp tục chương trình(Resume Program)

Nút này sẽ tiếp tục chạy ứng dụng một cách bình thường. Tạm thời bỏ qua debug

7. Tạm dừng chương trình(Pause Program)

Nút này sẽ được greyed-out trước tiên bởi vì chương trình đã sẵn sàng tạm dừng. Nếu bạn chọn tiếp tục chương trình, bạn sẽ tạm dừng nó lại với tùy chọn này.

8. Dừng ứng dụng (Stop App)

Nút này sẽ làm ứng dụng đang chạy trên thiết bị hoặc máy ảo dừng lại hoàn toàn.

Hiệu năng là tắt ứng dụng và không debug nữa.

9. Xem các Breakpoints

Nút này sẽ mở một cửa sổ hiển thị các tắt cả breakpoint đã được chèn. Ngoài ra, nó cho phép tinh chỉnh cài đặt cho từng breakpoint riêng biệt.

10. Mute Breakpoint

Tùy chọn này giúp cho có thể bật – tắt tạm thời tắt cả các breakpoint.

CÂU HỎI CHƯƠNG 1

1. Viết chương trình Java tính tổng các số chẵn nằm trong khoảng 1 đến N rồi in ra kết quả, với N được nhập từ bàn phím.
2. Viết chương trình Java hiển thị tổng các bội số của 7 nằm giữa 1 và 200.
3. Viết chương trình Java đưa ra màn hình các số nguyên <60 theo định dạng

1	3	5	7	9	11	13	15	17	19
21	23	25	27	29	31	33	35	37	39
41	43	45	47	49	51	53	55	57	59

4. Viết một chương trình quản lý nhân viên của một công ty. Bao gồm các lớp chính:
 - Lớp Human là một lớp trừu tượng, chỉ có một phương thức duy nhất là show().
 - Lớp Person là lớp kế thừa từ lớp Human, có hai thuộc tính là tên (name) và tuổi (age). Để đóng gói dữ liệu các thuộc tính này có dạng private và các phương thức truy nhập chúng (get và set). Ngoài ra lớp này còn cài đặt phương thức show() kế thừa từ lớp trừu tượng Human.
 - Lớp Employee là lớp kế thừa từ lớp Person, có thêm thuộc tính là lương (salary). Thuộc tính này cũng có dạng private để đóng gói dữ liệu và cần các phương thức truy nhập get/set. Lớp này cài đặt lại phương thức show(). Hơn nữa, lớp Employee có thêm hai phương thức addSalary() và addSalary(float) để tính tăng lương cho nhân viên: một phương thức tăng lương theo tỉ lệ mặc định là 10% (không cần tham số), và một phương thức tăng theo giá trị cụ thể đưa vào (cần tham số).
5. Nêu kiến trúc của hệ điều hành Android, các thành phần tạo nên ứng dụng Android.
6. Viết ứng dụng Android tính tổng, hiệu, tích, thương cho 2 số nguyên theo quy trình lập trình ứng dụng di động.

CHƯƠNG 2: LẬP TRÌNH ỨNG DỤNG ANDROID CƠ BẢN

Nội dung chương này tập trung trình bày về các kỹ thuật lập trình ứng dụng di động cơ sở trên nền tảng Android sử dụng ngôn ngữ Java. Nội dung trình bày bao gồm:

- Lập trình quản lý vòng đời ứng dụng di động.
- Lập trình giao diện ứng dụng di động.
- Lập trình quản lý dữ liệu.

2.1. QUẢN LÝ VÒNG ĐỜI LẬP TRÌNH ỨNG DỤNG DI ĐỘNG

2.1.1. Vòng đời ứng dụng di động

Ta biết rằng Android cung cấp máy ảo Dalvik cho phép tất cả các ứng dụng chạy trong tiến trình riêng của nó. Đồng thời Android có cơ chế quản lý các tiến trình (process) riêng này theo chế độ ưu tiên (priority). Các tiến trình có độ ưu tiên thấp sẽ bị Android giải phóng mà không hề cảnh báo nhằm đảm bảo tài nguyên.

Một sự khác biệt và đặc tính cơ bản của Android là thời gian sống của tiến trình ứng dụng không được điều khiển trực tiếp bởi chính nó. Thay vào đó, nó được xác định bởi hệ thống qua sự kết hợp của:

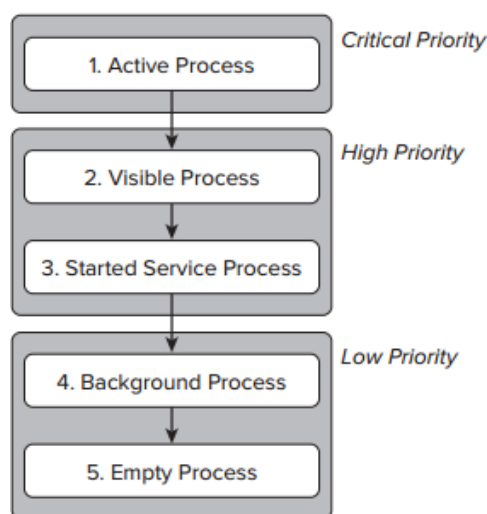
- Những phần của ứng dụng mà hệ thống biết đang chạy.
- Những phần đó quan trọng như thế nào đối với người dùng.
- Bao nhiêu vùng nhớ chiếm lĩnh trong hệ thống.

Các loại tiến trình (process) của Android bao gồm:

1. **Active process:** là process của ứng dụng hiện thời đang được người dùng tương tác. Active process bao gồm:
 - Các Activity đang ở trạng thái hoạt động (active).
 - Broadcast Receivers đang thực thi sự kiện onReceive.
 - Các service đang thực thi các sự kiện onStart, onCreate, or onDestroy.
 - Các service được chỉ định để chạy ở chế độ hiển thị trên màn hình (foreground).
2. **Visible process:** là process của ứng dụng mà Activity đang hiển thị đối với người dùng, nhưng không chạy ở chế độ foreground cũng như không nhận tương tác từ người dùng. Điều này xảy ra khi Activity không hiển thị toàn màn hình hoặc trong suốt với người dùng (Khi sự kiện onPause của Activity được gọi).
3. **Started Service process:** là Service đang chạy. Những service này bị Android giải phóng khi active process hoặc visible process cần sử dụng tài nguyên của hệ thống.
4. **Background process:** là process của ứng dụng mà các activity của nó không hiển thị với người dùng và không có bất kỳ service nào đang chạy (Khi sự kiện onStop của Activity được gọi).
5. **Empty process:** process không có bất cứ 1 thành phần nào active. Để cải thiện hiệu suất tổng thể hệ thống, Android thường giữ lại một ứng dụng trong bộ nhớ ngay cả khi nó không còn sử dụng, nhằm duy trì vùng nhớ tạm

thời để quá trình khởi động lại ứng dụng được nhanh chóng. Các process này sẽ bị giải phóng khi có yêu cầu.

Theo chế độ ưu tiên thì khi cần tài nguyên, Android sẽ tự động giải phóng process có độ ưu tiên từ thấp lên cao, trước tiên là các empty process.



Hình 2.1 Thứ tự ưu tiên của các tiến trình trong hệ điều hành Android

Mỗi ứng dụng chạy trong tiến trình riêng của mình do máy ảo của Android quản lý đều có chu kỳ sống của nó. Chu kỳ sống của ứng dụng Android phụ thuộc vào chu kỳ sống của các thành phần tạo nên ứng dụng Android, mỗi thành phần trong ứng dụng Android đều có một chu kỳ sống riêng. Các ứng dụng chỉ được gọi là kết thúc khi tất cả các thành phần trong ứng dụng kết thúc.

Trong các thành phần tạo nên ứng dụng Android như đã đề cập trong mục 2.1.4 thì một thành phần tối quan trọng của bất kỳ một ứng dụng Android là Activity. Activity là một thành phần cho phép người dùng giao tiếp với ứng dụng. Tuy nhiên, khi tất cả các Activity kết thúc và người dùng không còn giao tiếp được với ứng dụng nữa nhưng không có nghĩa là ứng dụng đã kết thúc. Bởi vì ngoài Activity là thành phần có khả năng tương tác với người dùng thì còn có các thành phần không có khả năng tương tác với người dùng như Service, Broadcast receiver. Có nghĩa là những thành phần không tương tác với người dùng có thể chạy background dưới sự giám sát của hệ điều hành cho đến khi người dùng tự tắt chúng.

Mô hình vòng đời ứng dụng Android phụ thuộc chặt chẽ vào mô hình vòng đời của Activity. Chính vì vậy việc hiểu vòng đời của Activity là rất quan trọng trong việc xử lý thông tin của ứng dụng Android.

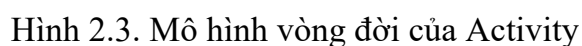
2.1.2. Quản lý vòng đời lập trình ứng dụng di động

2.1.2.1. Mô hình vòng đời Activity

Nội dung phần này sẽ trình bày về mô hình vòng đời của Activity trong ứng dụng Android.

Như đã đề cập ở các phần trước, trong mỗi ứng dụng có thể có một hoặc nhiều Activity. Mỗi một Activity sẽ có một vòng đời riêng độc lập hoàn toàn với Activity khác. Bên trong hệ thống, các Activity được quản lý bởi một ngăn xếp Activity (Activity

Hình 2.2. Activity stack



Các phương thức quản lý vòng đời Activity được định nghĩa sẵn trong lớp Activity gồm:

- onCreate() : hàm này được gọi khi Activity được khởi tạo, dùng để thiết lập giao diện ứng dụng và thực thi những thao tác cơ bản.
- onStart() : hàm này được gọi khi Activity xuất hiện trên màn hình.
- onResume() : hàm này được gọi ngay sau onStart() hoặc khi người dùng tương tác với Activity.
- onPause() : hàm này được gọi khi Activity hiện thời bị đóng, hệ thống đang gọi đến một Activity trước đó.
- onStop() : hàm này được gọi khi Activity hiện thời không còn được hiển thị tới người dùng hoặc Activity khác đang được hiển thị tới người dùng.
- onDestroy() : hàm này được gọi trước khi Activity bị giải phóng khỏi hệ thống.
- onRestart() : hàm này được gọi khi Activity đang ở trạng thái dừng (sự kiện onStop() được gọi) và muốn khởi động lại.

2.1.2.2. Lập trình quản lý vòng đời ứng dụng di động

Nội dung phần này sẽ trình bày về việc lập trình quản lý vòng đời của Activity trong ứng dụng Android.

Bước 1: từ AndroidStudio, tạo Android project có tên Activity101

Bước 2: tạo một activity mới bằng một trong hai cách sau:

Cách 1: Tạo Activity bằng mã nguồn

Tạo lớp Java kế thừa từ lớp Activity cơ sở

```
package net.learn2develop.Activity101;
import android.app.Activity;
import android.os.Bundle;
public class Activity101Activity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Activity này tải giao diện từ file XML có tên main.xml, được định nghĩa trong folder res/layout, thông qua câu lệnh:

```
setContentView(R.layout.main);
```

Khai báo activity trên trong file AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
manifest
xmlns:android="http://schemas.android.com/apk/res/android"
package="net.learn2develop.Activity101"
android:versionCode="1"
```

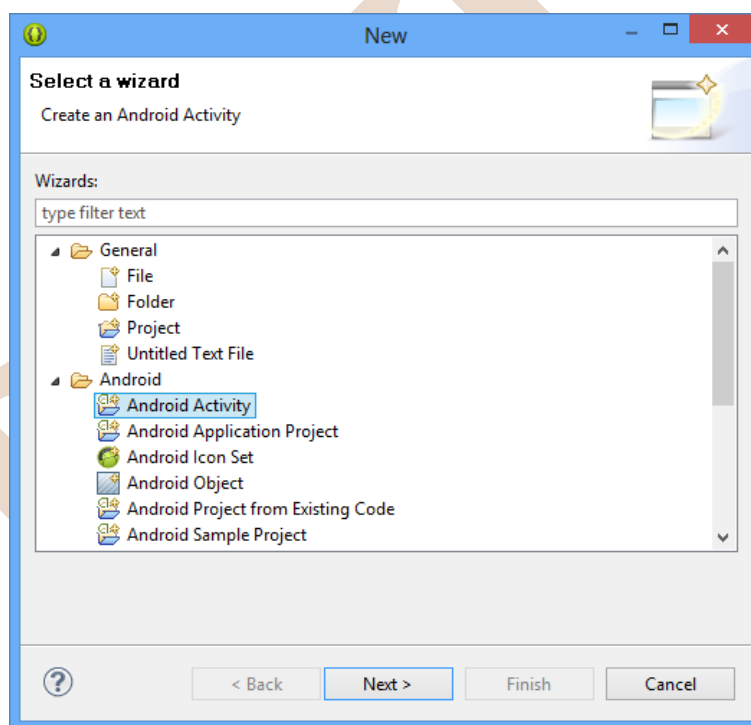
```

android:versionName="1.0" >
<uses-sdk android:minSdkVersion="14" />
<application
android:icon="@drawable/ic_launcher"
android:label="@string/app_name" >
<activity
android:label="@string/app_name"
android:name=".Activity101Activity" >
<intent-filter >
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>

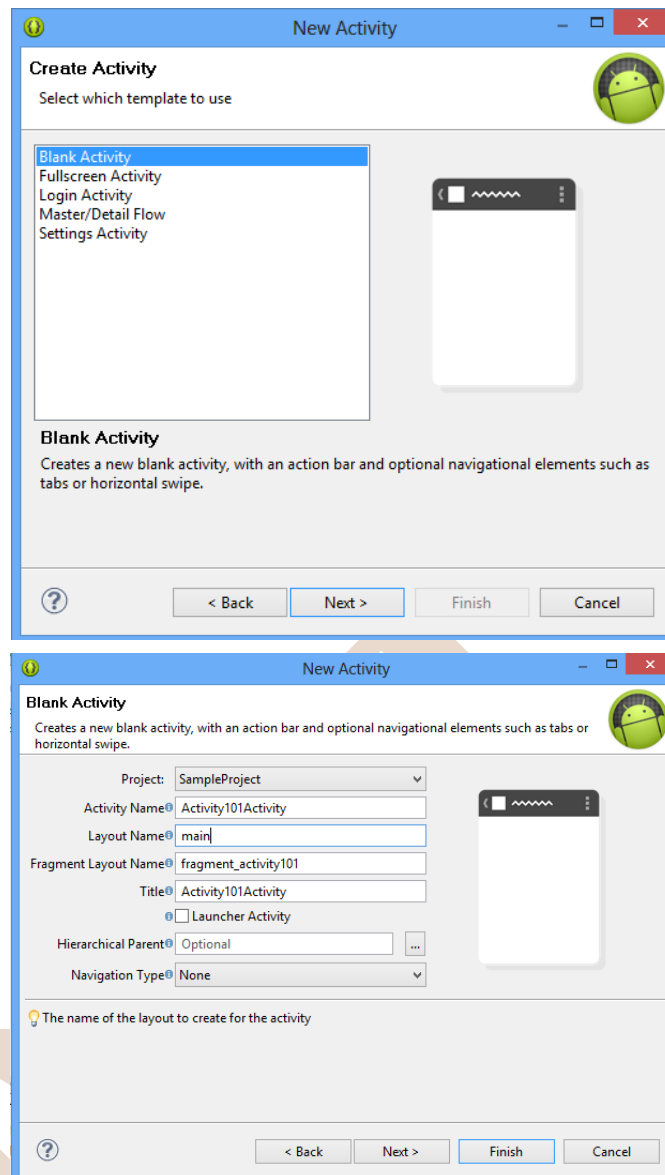
```

Cách 2: Tạo mới Activity bằng thao tác trên giao diện

Nháy chuột phải vào project Activity101, chọn new -> Other -> Android -> Android Activity, nhập thông số cho Activity



Hình 2.4. Tạo mới Activity



Khi đó Activity mới sẽ được tạo ra gồm 2 file : file xử lý logic (Activity101Activity.java) và file giao diện lưu trong res/layout (main.xml). Khai báo activity trên trong file AndroidManifest.xml giống như cách 1

Bước 3: Quản lý vòng đời Activity qua các hàm sự kiện trong mô hình vòng đời Activity.

Thêm mã nguồn in đậm sau trong file Activity101Activity.java :

```
package net.learn2develop.Activity101;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
public class Activity101Activity extends Activity {
    String tag = "Lifecycle";
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Log.d(tag, "In the onCreate() event");
    }
}
```

```

    }
    public void onStart()
    {
        super.onStart();
        Log.d(tag, "In the onStart() event");
    }
    public void onRestart()
    {
        super.onRestart();
        Log.d(tag, "In the onRestart() event");
    }
    public void onResume()
    {
        super.onResume();
        Log.d(tag, "In the onResume() event");
    }
    public void onPause()
    {
        super.onPause();
        Log.d(tag, "In the onPause() event");
    }
    public void onStop()
    {
        super.onStop();
        Log.d(tag, "In the onStop() event");
    }
    public void onDestroy()
    {
        super.onDestroy();
        Log.d(tag, "In the onDestroy() event");
    }
}

```

Bước 4: Bấm F11 để debug ứng dụng trên Emulator

- ❖ Khi ứng dụng chạy trên Emulator, quan sát cửa sổ debug LogCat:

```

11-16 06:25:59.396: D/Lifecycle(559): In the onCreate() event
11-16 06:25:59.396: D/Lifecycle(559): In the onStart() event
11-16 06:25:59.396: D/Lifecycle(559): In the onResume() event

```

- ❖ Bấm nút Back từ Android Emulator, những nội dung sau sẽ được in ra:

```

11-16 06:29:26.665: D/Lifecycle(559): In the onPause() event
11-16 06:29:28.465: D/Lifecycle(559): In the onStop() event
11-16 06:29:28.465: D/Lifecycle(559): In the onDestroy() event

```

- ❖ Bấm giữ nút Home và bấm icon của ứng dụng để mở lại ứng dụng, những nội dung sau sẽ được in ra:

```

11-16 06:31:08.905: D/Lifecycle(559): In the onCreate() event
11-16 06:31:08.905: D/Lifecycle(559): In the onStart() event
11-16 06:31:08.925: D/Lifecycle(559): In the onResume() event

```

- ❖ Bấm nút Phone trên Android Emulator để chuyển Activity hoạt động ở trạng thái nền (background), những nội dung sau sẽ được in ra:

```

11-16 06:32:00.585: D/Lifecycle(559): In the onPause() event
11-16 06:32:05.015: D/Lifecycle(559): In the onStop() event

```

- ❖ Bấm nút Back từ Android Emulator để thoát chương trình gọi điện trên, khi đó Activity trên sẽ được gọi lại, những nội dung sau sẽ được in ra:

11-16 06:32:50.515: D/Lifecycle(559): In the onRestart() event
11-16 06:32:50.515: D/Lifecycle(559): In the onStart() event
11-16 06:32:50.515: D/Lifecycle(559): In the onResume() event

❖ Chú ý rằng sự kiện OnDestroy() không được gọi cho biết rằng Activity vẫn còn tồn tại trong bộ nhớ.

2.2. ACTIVITIES VÀ INTENTS

2.2.1. Hiểu về Activities

Mục này bắt đầu bằng cách xem xét cách tạo ra một Activity. Để tạo một hoạt động, ta tạo một Java lớp mở rộng lớp cơ sở Activity:

```
package net.learn2develop.Activity101;
import android.app.Activity;
import android.os.Bundle;
public class Activity101Activity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Lớp Activity tải thành phần UI của nó bằng tệp XML được xác định trong thư mục res / layout. Trong ví dụ này, sẽ tải UI từ tệp main.xml:

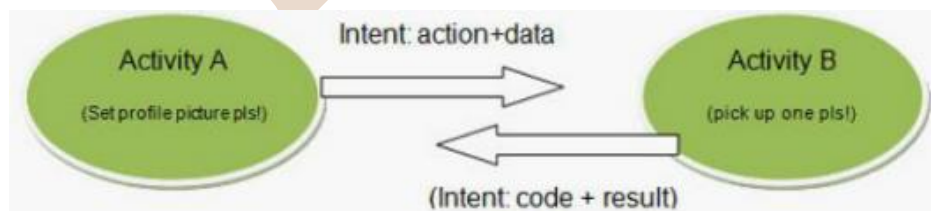
```
setContentView(R.layout.main);
```

Tất cả các Activity có trong ứng dụng phải được khai báo trong file Androidmanifest.xml như dưới đây

2.2.2. Liên kết giữa các Activities sử dụng Intents

Khái niệm Intents

Ứng dụng Android thường bao gồm nhiều Activity, mỗi Activity hoạt động độc lập với nhau và thực hiện những công việc khác nhau. Intent giống như người đưa thư, giúp Activity này có thể triệu gọi cũng như truyền dữ liệu cần thiết tới một Activity khác. Điều này cũng giống như việc di chuyển qua lại giữa các Form trong lập trình Windows Form.



Hình 2.5. Dùng Intent để truyền dữ liệu giữa 2 Activity

❖ Sử dụng Intent để triệu gọi Activity

- Khai báo Activity tường minh: cần cung cấp chính xác thông tin của activity cần gọi (nếu 2 activity muốn gọi nhau thuộc cùng 1 ứng dụng thì chỉ cần cung cấp tên class, khác ứng dụng thì cần cung cấp thêm package và tên class).

Ví dụ: đoạn code dưới đây sẽ khởi động Activity tên là TargetActivity

```
Intent i = new Intent(getApplicationContext(), TargetActivity.class);
```

```
startActivity(i);
```

- Khai báo Activity không tường minh: cần cung cấp thao tác cần làm gì, với loại dữ liệu nào, thao tác thuộc nhóm nào,... hệ thống sẽ tìm activity tương ứng để khởi động.

Ví dụ: đoạn code dưới đây sẽ khởi động Activity xem ảnh

```
Intent i = new Intent(Intent.ACTION_VIEW);
```

```
i.setData(MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
```

```
startActivity(i);
```

Với cách khởi động Activity không tường minh thì khi khai báo Activity trong AndroidManifest.xml cần khai báo Intent-filter bên trong Activity. Intent-filter sẽ giúp một Activity đăng ký với hệ thống Activity đó có làm được thao tác gì, trong nhóm nào, với loại dữ liệu nào. Như vậy với việc tạo ra đối tượng Intent trong code và khai báo Intent-filter cho Activity trong AndroidManifest.xml thì Activity sẽ được hệ thống khởi động.

❖ Sử dụng Intent để truyền dữ liệu giữa các Activity

- Khi khởi động một Activity, ta có thể gửi kèm dữ liệu trong đối tượng Intent như:

```
i.putExtra("value1", new String("Hello")); // i là 1 đối tượng của lớp Intent
```

```
i.putExtra("value2", new Long(100));
```

- Bên phía Activity được triệu gọi, ta có thể lấy dữ liệu được gửi tới như sau:

```
getIntent().getExtras().getString("value1");
```

```
getIntent().getExtras().getLong("value2");
```

- Có thể khởi động một Activity với một yêu cầu nào đó và Activity được triệu gọi khi làm xong công việc sẽ trả lại kết quả cho Activity trước

Mình họa:

- Từ IDE tạo Android project có tên PassingData
- Chỉnh sửa file main.xml trong thư mục res/layout, chú ý thêm các dòng in đậm sau:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" android:orientation="vertical" >
    <Button
        android:id="@+id/btn_SecondActivity"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Click to go to Second Activity"
        android:onClick="onClick"/>
</LinearLayout>
```

- Tạo file secondactivity.xml trong thư mục res/layout như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical" >
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Welcome to Second Activity" />
<Button
android:id="@+id/btn_MainActivity" android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Click to return to main activity"
android:onClick="onClick"/>
</LinearLayout>
```

- Tạo class SecondActivity. Viết code cho SecondActivity.java như sau:

```
package net.learn2develop.PassingData;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
public class SecondActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.secondactivity);
        //---get the data passed in using getStringExtra()---
        Toast.makeText(this, getIntent().getStringExtra("str1"),
        Toast.LENGTH_SHORT).show();
        //---get the data passed in using getIntExtra()---
        Toast.makeText(this, Integer.toString(
        getIntent().getIntExtra("age1", 0)),
        Toast.LENGTH_SHORT).show();
        //---get the Bundle object passed in---
        Bundle bundle = getIntent().getExtras();
        //---get the data using the getString()---
        Toast.makeText(this, bundle.getString("str2"),
        Toast.LENGTH_SHORT).show();
        //---get the data using the getInt() method---
        Toast.makeText(this, Integer.toString(bundle.getInt("age2")),
        Toast.LENGTH_SHORT).show();
    }
    public void onClick(View view) {
        //---use an Intent object to return data---
        Intent i = new Intent();
        //---use the putExtra() method to return some
        // value---
        i.putExtra("age3", 45);
        //---use the setData() method to return some value---
```

```

i.setData(Uri.parse(
    "Something passed back to main activity"));
//---set the result with OK and the Intent object---
setResult(RESULT_OK, i);
//---destroy the current activity---
finish();
}
}

```

- Thêm mã in đậm sau vào AndroidManifest.xml

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="net.learn2develop.PassingData"
android:versionCode="1"
android:versionName="1.0" >
<uses-sdk android:minSdkVersion="14" />
<application
android:icon="@drawable/ic_launcher"
android:label="@string/app_name" >
<activity
android:label="@string/app_name"
android:name=".PassingDataActivity" >
<intent-filter >
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity
android:label="Second Activity"
android:name=".SecondActivity" >
<intent-filter >
<action android:name="net.learn2develop.PassingDataSecondActivity"
/>
<category android:name="android.intent.category.DEFAULT" />
</intent-filter>
</activity>
</application>
</manifest>

```

- Viết code cho file PassingDataActivity.java, chú ý thêm các dòng in đậm sau:

```

package net.learn2develop.PassingData;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
public class PassingDataActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    public void onClick(View view) {
        Intent i = new
        Intent("net.learn2develop.PassingDataSecondActivity");
    }
}

```

```

//---use putExtra() to add new name/value pairs---
i.putExtra("str1", "This is a string");
i.putExtra("age1", 25);
//---use a Bundle object to add new name/values
// pairs---
Bundle extras = new Bundle();
extras.putString("str2", "This is another string");
extras.putInt("age2", 35);
//---attach the Bundle object to the Intent object---
i.putExtras(extras);
//---start the activity to get a result back---
startActivityForResult(i, 1);
}
public void onActivityResult(int requestCode,
int resultCode, Intent data)
{
//---check if the request code is 1---
if (requestCode == 1) {
//---if the result is OK--- if (resultCode == RESULT_OK) {
//---get the result using getIntExtra()---
Toast.makeText(this, Integer.toString(
data.getIntExtra("age3", 0)),
Toast.LENGTH_SHORT).show();
//---get the result using getData()---
Toast.makeText(this, data.getData().toString(),
Toast.LENGTH_SHORT).show();
}
}
}
}

```

- F11 để debug ứng dụng trên Android emulator. Nhấn button trên mỗi Activity và quan sát kết quả hiển thị.

Giải thích cách thức hoạt động của Intent trong ví dụ minh họa trên:

- Hàm putExtra() dùng để gửi kèm dữ liệu (name/value) trong đối tượng Intent:

```

//---use putExtra() to add new name/value pairs---
i.putExtra("str1", "This is a string");
i.putExtra("age1", 25);

```

Giá trị của dữ liệu (value) truyền vào có thể là kiểu nguyên thủy như: string, integer,... hoặc kiểu Bundle.

Bundle giống như 1 từ điển lưu một tập dữ liệu với mỗi dữ liệu là cặp name/value:

```

//---use a Bundle object to add new name/values pairs---
Bundle extras = new Bundle();
extras.putString("str2", "This is another string");
extras.putInt("age2", 35);
//---attach the Bundle object to the Intent object---
i.putExtras(extras);

```

- Activity được triệu gọi muốn nhận được thông tin dữ liệu do Activity thứ nhất gửi đến thì cần gọi lấy lại đối tượng Intent thông qua phương thức getIntent(). Sau đó gọi hàm getStringExtra(), getIntExtra(), getExtras() để lấy về giá trị do hàm putExtra() gửi đi tương ứng với các kiểu dữ liệu string, integer, Bundle từ Activity thứ nhất:

```
//---get the data passed in using getStringExtra()---
Toast.makeText(this, getIntent().getStringExtra("str1"),
Toast.LENGTH_SHORT).show();
//---get the data passed in using getIntExtra()---
Toast.makeText(this, Integer.toString(
getIntent().getIntExtra("age1", 0)),
Toast.LENGTH_SHORT).show();
//---get the Bundle object passed in---
Bundle bundle = getIntent().getExtras();
//---get the data using the getString()---
Toast.makeText(this, bundle.getString("str2"),
Toast.LENGTH_SHORT).show();
//---get the data using the getInt() method---
Toast.makeText(this, Integer.toString(bundle.getInt("age2")),
Toast.LENGTH_SHORT).show();
```

- Cách khác để truyền dữ liệu giữa 2 Activity là sử dụng phương thức setData() :

```
//---use the setData() method to return some value---
i.setData(Uri.parse( "Something passed back to main activity"));
```

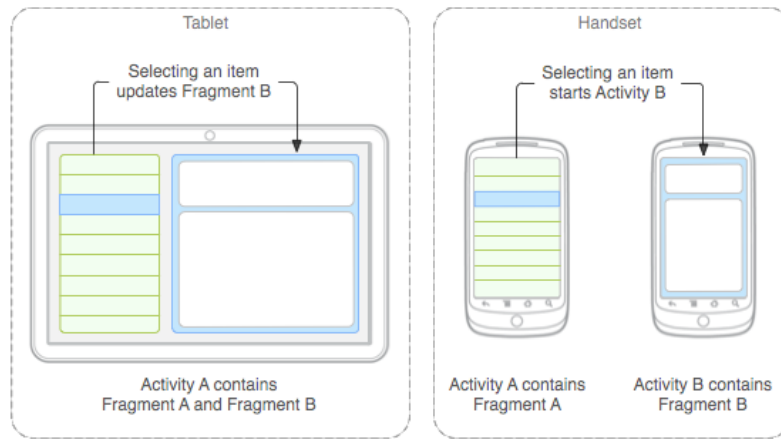
Thường sử dụng phương thức setData() được sử dụng để truyền dữ liệu cho các ứng dụng khác cài đặt sẵn trên thiết bị (build-in application), như truyền URL từ Activity tới ứng dụng web browser để xem nội dung web. Khi đó Activity được triệu gọi muốn nhận được dữ liệu do Intent gửi đến thông qua phương thức setData() thì cần sử dụng phương thức getData():

```
//---get the result using getData()---
Toast.makeText(this, data.getData().toString(),
Toast.LENGTH_SHORT).show();
```

2.2.3. Fragments

Fragment là một thành phần giao diện được nhúng vào Activity cho phép thiết kế Activity linh động. Fragment có thể coi như sub-activity.

Nếu chỉ sử dụng Activity không chứa Fragment để xây dựng ứng dụng thì không thể thay đổi giao diện Activity khi chạy ứng dụng trên các thiết bị di động khác nhau. Với việc sử dụng Fragment trong Activity sẽ giúp Activity có giao diện tùy biến linh hoạt trên các thiết bị di động có kích thước màn hình khác nhau.



Hình 2.6. Ví dụ kết hợp các Fragment trên một Activity

Phân loại Fragment

- ListFragment : dùng để hiển thị danh sách mục tin.
- DialogFragment : fragment hiển thị dưới dạng dialog động phía trên Activity.
- PreferenceFragment : dùng để hiển thị phân cấp các đối tượng

Đặc điểm của Fragment

- Fragment có layout và vòng đời hoạt động riêng. Vòng đời hoạt động của Fragment phụ thuộc vào vòng đời của Activity chứa nó.
- Fragment có thể được thêm vào hoặc gỡ bỏ khỏi Activity trong khi Activity đang hoạt động.
- Một Activity có thể được hình thành từ nhiều Fragment, một Fragment có thể được sử dụng bởi nhiều Activity.
- Fragment được đưa vào từ Android API 11.

Cách tạo và sử dụng Fragment

- Tạo Fragment bằng cách tạo 1 lớp kế thừa từ class Fragment cơ sở.
- Tạo giao diện cho Fragment dưới dạng mã XML
- Nhúng Fragment vào Activity bằng cách khai báo thành phần <fragment> trong file giao diện của Activity.

Minh họa:

- Trong thư mục res/layout, thêm file mới 2 file :
- File fragment1.xml có nội dung như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#00FF00"
>
<TextView
```



```

android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="This is fragment #1"
android:textColor="#000000"
android:textSize="25sp" />
</LinearLayout>

```

- File fragment2.xml có nội dung như sau:

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#FFFE00"
>
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="This is fragment #2"
android:textColor="#000000"
android:textSize="25sp" />
</LinearLayout>

```

- Chỉnh sửa file main.xml trong thư mục res/layout, chú ý thêm các dòng in đậm sau:

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent" android:orientation="horizontal" >
<!--
<fragment
android:name="net.learn2develop.Fragments.Fragment1"
android:id="@+id/fragment1"
android:layout_weight="1"
android:layout_width="0px"
android:layout_height="match_parent" />
<fragment
android:name="net.learn2develop.Fragments.Fragment2"
android:id="@+id/fragment2"
android:layout_weight="1"
android:layout_width="0px"
android:layout_height="match_parent" />
-->
</LinearLayout>

```

- Thêm 2 file Java class có tên Fragment1.java , Fragment2.java

File Fragment1.java có nội dung sau:

```

package net.learn2develop.Fragments;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
public class Fragment1 extends Fragment {

```

```

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {
    //---Inflate the layout for this fragment---
    return inflater.inflate(
        R.layout.fragment1, container, false);
}
}

```

- File Fragment2.java có nội dung sau:

```

package net.learn2develop.Fragments;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
public class Fragment2 extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {
        //---Inflate the layout for this fragment---
        return inflater.inflate(
            R.layout.fragment2, container, false);
    }
}

```

- Viết code cho file FragmentsActivity.java, chú ý thêm các dòng in đậm sau:

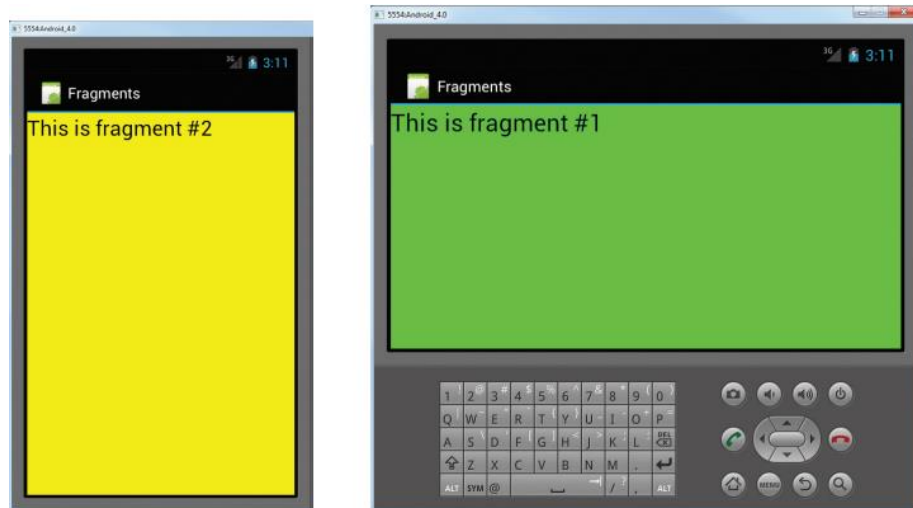
```

package net.learn2develop.Fragments;
import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentTransaction;
import android.os.Bundle;
import android.view.Display;
import android.view.WindowManager;
public class FragmentsActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        FragmentManager fragmentManager = getFragmentManager();
        FragmentTransaction fragmentTransaction =
            fragmentManager.beginTransaction();
        //---get the current display info---
        WindowManager wm = getWindowManager();
        Display d = wm.getDefaultDisplay();
        if (d.getWidth() > d.getHeight())
        {
            //---landscape mode---
            Fragment1 fragment1 = new Fragment1();
            // android.R.id.content refers to the content
            // view of the activity
            fragmentTransaction.replace(android.R.id.content, fragment1);
        }
        else
        {

```

```
//---portrait mode---
Fragment2 fragment2 = new Fragment2();
fragmentTransaction.replace(android.R.id.content, fragment2);
}
fragmentTransaction.commit();
}
}
```

- Nhấn F11 để debug ứng dụng trên Android emulator theo chiều thẳng đứng (portrait mode), Ctrl_F11 để thay đổi hướng màn hình emulator theo chiều nằm ngang (landscape mode)



Hình 2.7. Fragment

Giải thích:

Vì Fragment có thể coi như sub-activity, gồm 1 class Java tải giao diện từ file giao diện XML và xử lý tương tác từ người dùng.

Tạo Fragment bằng cách tạo 1 class Java kế thừa từ lớp Fragment cơ sở

```
public class Fragment1 extends Fragment {
}
```

Để tải giao diện cho Fragment từ file XML, trong Fragment1.java cần override phương thức onCreateView(). Phương thức này trả về 1 đối tượng View.

```
public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
//---Inflate the layout for this fragment---
return inflater.inflate(R.layout.fragment1, container, false);
}
```

Sử dụng phương thức inflate() của đối tượng LayoutInflater để tải giao diện từ fragment1.xml vào Fragment1. Phương thức này có tham số container dùng để tham chiếu tới ViewGroup của Activity chứa Fragment, tham số savedInstanceState cho phép khôi phục lại trạng thái trước đó của Fragment. Chú ý mỗi Fragment cần có 1 định danh (id) duy nhất.

Nhúng Fragment vào Activity sử dụng một trong 2 cách:

- *Cách 1: Thêm <fragment> vào main.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal" >
    <!--
    <fragment
        android:name="net.learn2develop.Fragments.Fragment1"
        android:id="@+id/fragment1"
        android:layout_weight="1"
        android:layout_width="0px"
        android:layout_height="match_parent" />
    <fragment
        android:name="net.learn2develop.Fragments.Fragment2"
        151
        android:id="@+id/fragment2"
        android:layout_weight="1"
        android:layout_width="0px"
        android:layout_height="match_parent" />
    -->
</LinearLayout>
```

Tuy nhiên để linh động trong việc nhúng Fragment vào Activity ta sử dụng cách thứ 2 sau:

- *Cách 2: Nhúng Fragment vào Activity tại thời điểm ứng dụng chạy*

Lấy về đối tượng FragmentManager:

```
FragmentManager fragmentManager = getSupportFragmentManager();
```

Lấy về đối tượng FragmentTransaction cho phép thực hiện các giao dịch của Fragment trong Activity

```
FragmentTransaction fragmentTransaction =
    fragmentManager.beginTransaction();
```

Lấy về đối tượng Display để xác định thiết bị ở chế độ portrait hay landscape

```
WindowManager wm = getWindowManager();
Display d = wm.getDefaultDisplay();
```

Để có thể tải Fragment mong muốn ở chế độ landscape cần lấy về đối tượng Fragment1 và gọi phương thức replace() của đối tượng FragmentTransaction với tham số truyền thứ nhất là android.R.id.content tham chiếu tới giao diện hiển thị của Activity, tham số thứ 2 là fragment1 là đối tượng của lớp Fragment1.

```
//---landscape mode---
Fragment1 fragment1 = new Fragment1();
// android.R.id.content refers to the content
```

```
// view of the activity
fragmentTransaction.replace(android.R.id.content, fragment1);
```

Ngoài ra bạn có thể sử dụng phương thức `remove()` của lớp `FragmentManager` để loại bỏ `Fragment` khỏi `Activity`. Cuối cùng để đảm bảo các phương thức trên có hiệu lực, cần gọi đến phương thức `commit()`:

```
fragmentTransaction.commit();
```

2.3. LẬP TRÌNH GIAO DIỆN

Trong phần trước, ta đã tìm hiểu về lớp `Activity` và vòng đời của nó. Ta đã học được rằng một `Activity` là một phương tiện mà người dùng tương tác với ứng dụng. Tuy nhiên, một `Activity` của bản thân nó không có sự thể hiện trên màn hình. Thay vào đó, nó phải vẽ màn hình bằng cách sử dụng `Views` và `Viewgroup`. Trong phần này, ta sẽ tìm hiểu chi tiết về cách tạo giao diện người dùng trong `Android` và cách người dùng tương tác với chúng. Ngoài ra, ta sẽ học cách xử lý các thay đổi trong định hướng màn hình trên các thiết bị `Android`.

2.3.1. Tìm hiểu giao diện người dùng *Android*

2.3.1.1. Hiểu về giao diện người dùng

Trong phần trước, ta biết được thành phần cơ bản của ứng dụng `android` là `activity`. `Activity` sẽ hiển thị giao diện ứng dụng bao gồm các widgets như `button`, `labels`, `textboxes`... Và xác định giao diện sử dụng file `XML` (ví dụ `mail.xml` trong thư mục `res/layout` trong dự án. Có dạng như sau

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        tools:layout_editor_absoluteY="434dp" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
```

```

tools:layout_editor_absoluteY="536dp"
tools:srcCompat="@tools:sample/avatars" />

</androidx.constraintlayout.widget.ConstraintLayout>

```

Trong suốt quá trình chạy, Giao diện UI sẽ được tải qua phương thức onCreate() trong lớp Activity, sử dụng phương thức setContentView() của lớp Activity:

```

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btn= (Button) findViewById(R.id.button);
        btn.setText("Demo");
    }
}

```

Trong quá trình biên dịch, mỗi phần tử trong file XML được biên dịch thành những lớp GUI Android tương ứng với các thuộc tính được biểu diễn bởi các phương thức. Hệ thống Android sau đó tạo giao diện người dùng của Activity khi nó được tải.

2.3.1.2. Views and ViewGroups

Một Activity chứa các Views và ViewGroups. Một View là một cấu thành giao diện (widget) xuất hiện trên màn hình. Ví dụ buttons, labels, và textboxes. View được dẫn xuất từ lớp cơ sở là android.view.View.

Một hoặc nhiều view có thể được nhóm với nhau trong ViewGroup. Một ViewGroup (bản thân nó cũng là một View) cung cấp các bố cục (layout) trong đó ta có thể sắp xếp sự xuất hiện và thứ tự của các views. Ví dụ về ViewGroup bao gồm LinearLayout và FrameLayout. Một ViewGroup dẫn xuất từ lớp cơ sở android.view.ViewGroup.

Android hỗ trợ những ViewGroup sau

- LinearLayout
- AbsoluteLayout
- TableLayout
- RelativeLayout
- FrameLayout
- ScrollView

Các phần sau đây mô tả từng Nhóm này một cách chi tiết hơn. Lưu ý rằng trong thực tế, nó là phổ biến để kết hợp các loại bố cục khác nhau để tạo giao diện người dùng mong muốn.

LinearLayout

LinearLayout sắp xếp các views trong một cột đơn hoặc hàng đơn. Các view con (child views) có thể được sắp xếp hoặc là theo chiều dọc, hoặc theo chiều ngang. Để thấy được cách LinearLayout bố cục, ta xem các phần tử được bố trí trong file mail.xml sau:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />
</LinearLayout>
```

Trong file mail.xml, ta thấy phần tử gốc là <LinearLayout> và có phần tử <TextView> là phần tử được chứa trong nó. Phần tử <LinearLayout> sẽ điều khiển thứ tự các view xuất hiện trong nó.

Mỗi View và ViewGroup đều có tập các thuộc tính chung, một số thuộc tính được mô tả trong bảng sau:

Bảng 2.1 Thuộc tính của View và ViewGroup

Thuộc tính	Mô tả
layout_width	Độ rộng của View hoặc ViewGroup
layout_height	Chiều dài của View hoặc ViewGroup
layout_marginTop	Chỉ định không gian thừa ở phía trên cùng của View hoặc Viewgroup
layout_marginBottom	Chỉ định không gian thừa ở phía dưới cùng của View hoặc Viewgroup
layout_marginLeft	Chỉ định không gian thừa ở phía trái của View hoặc Viewgroup
layout_marginRight	Chỉ định không gian thừa ở phía phải của View hoặc Viewgroup
layout_gravity	Chỉ định Vị trí View được định vị
layout_weight	Chỉ định bao nhiêu không gian thừa trong layout sẽ được phân bổ cho View
layout_x	Chỉ định tọa độ trục x của View hoặc ViewGroup
layout_y	Chỉ định tọa độ trục y của View hoặc ViewGroup

Ví dụ: chiều rộng của phần tử <TextView> sẽ lấp đầy toàn bộ chiều rộng của phần tử mẹ của nó (là phần tử màn hình trong trường hợp này) bằng cách sử dụng hằng số fill_parent. Chiều cao của nó được biểu thị bằng Wrap_content không đổi, có nghĩa là chiều cao của nó là chiều cao của nội dung của nó (trong trường hợp này là văn bản chứa trong đó). Nếu không muốn View <TextView> chiếm toàn bộ hàng, bạn có thể đặt thuộc tính layout_width thành Wrap_content, như sau:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
```

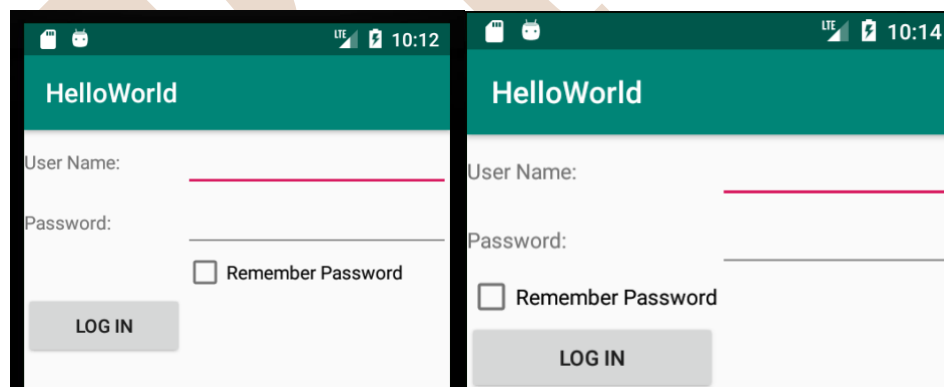
Tablelayout

Tablelayout nhóm các View vào trong các hàng và các cột. Ta có thể sử dụng phần tử <TableRow> để chỉ định một hàng trong bảng. Mỗi row chứa một hoặc nhiều view, mỗi view đặt trong row tạo thành một cell. Độ rộng của mỗi cột được xác định bằng độ rộng lớn nhất của Cell trong cột đó.

Ví dụ một layout main.xml có nội dung như sau:

```
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent" >
    <TableRow>
        <TextView
            android:text="User Name:"
            android:width="120dp"
        />
        <EditText
            android:id="@+id/txtUserName"
            android:width="200dp" />
    </TableRow>
    <TableRow>
        <TextView
            android:text="Password:"
        />
        <EditText
            android:id="@+id/txtPassword"
            android:password="true"
        />
    </TableRow>
    <TableRow>
        <TextView />
        <CheckBox android:id="@+id/chkRememberPassword"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Remember Password"
        />
    </TableRow>
    <TableRow>
        <Button
            android:id="@+id/buttonSignIn"
            android:text="Log In" />
    </TableRow>
</TableLayout>
```

Trong ví dụ trên, giao diện hình thành 2 cột và 4 hàng. Ô trực tiếp dưới ô Password là <TextView/> với phần tử trống. Nếu không làm thế, ô Remember Password sẽ chèn vào ô trống như hình:



Hình 2.8 Table Layout

RelativeLayout

RelativeLayout cho phép xác định cách các view được định vị trong mối quan hệ với các view khác trong bố cục. Xem xét ví dụ sau:

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    android:id="@+id/RLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android" >

    <TextView
        android:id="@+id/lblComments"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Comments"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true" />

    <EditText
        android:id="@+id/txtComments"

        android:layout_width="fill_parent"
        android:layout_height="170px"
        android:textSize="18sp"
        android:layout_alignLeft="@+id/lblComments"
        android:layout_below="@+id/lblComments"
        android:layout_centerHorizontal="true" />

    <Button
        android:id="@+id/btnSave"
        android:layout_width="125px"
        android:layout_height="wrap_content"
        android:text="Save"
        android:layout_below="@+id/txtComments"
        android:layout_alignRight="@+id/txtComments" />

    <Button
        android:id="@+id/btnCancel"
        android:layout_width="124px"
        android:layout_height="wrap_content"
        android:text="Cancel"
        android:layout_below="@+id/txtComments"
        android:layout_alignLeft="@+id/txtComments" />
</RelativeLayout>

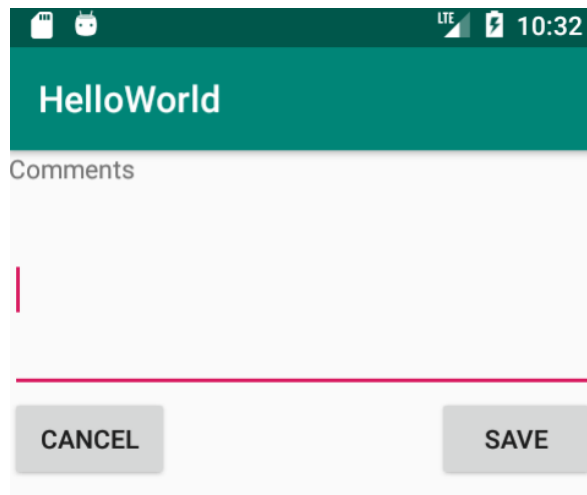
```

Ghi nhớ là mỗi View được nhúng trong Relative layout có các thuộc tính cho phép chúng căn thẳng với các View khác. Những thuộc tính đó như sau:

- layout_alignParentTop
- layout_alignParentLeft
- layout_alignLeft
- layout_alignRight
- layout_below
- layout_centerHorizontal

Giá trị của các thuộc tính này là ID của mỗi View được tham chiếu.

Giao diện được tạo ra của đoạn mã trên như hình dưới:



Hình 2.9. Relative Layout

FrameLayout

FrameLayout là một trình giữ chỗ trên màn hình mà có thể sử dụng để hiển thị một View. View mà ta thêm vào FrameLayout luôn được neo ở phía trên bên trái của bố trí.

Ví dụ trong đoạn mã xml sau:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    android:id="@+id/RLayout"

    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android" >

    <TextView
        android:id="@+id/lblComments"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, Android!"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true" />

    <FrameLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/lblComments"
        android:layout_below="@+id/lblComments"
        android:layout_centerHorizontal="true" >

        <ImageView
            android:src = "@drawable/droid"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

    </FrameLayout>
</RelativeLayout>
```

Ở đó, ta thấy FrameLayout ở bên trong RelativeLayout. Trong FrameLayout, ta nhúng ImageView, và giao diện được thể hiện trong hình sau:



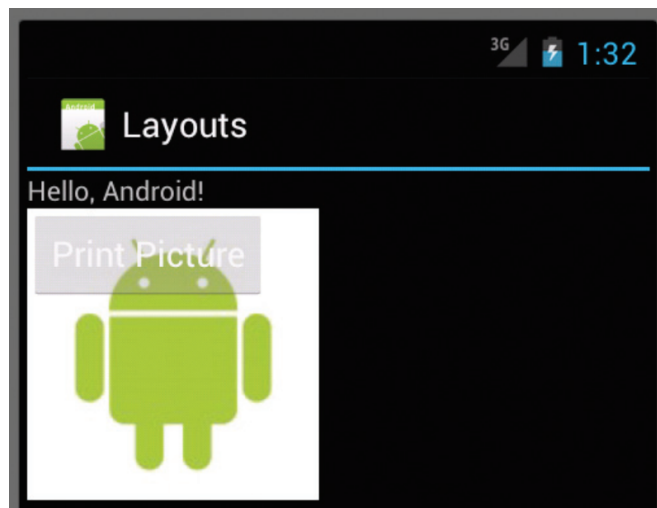
Hình 2.10. Frame Layout

Nếu chèn thêm một View khác (ví dụ Button) trong FrameLayout, View đó sẽ bị chồng lên View cũ

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    android:id="@+id/RLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
    <TextView
        android:id="@+id/lblComments"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, Android!"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
    />
    <FrameLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/lblComments"
        android:layout_below="@+id/lblComments"
        android:layout_centerHorizontal="true" >
        <ImageView
            android:src = "@drawable/droid"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

        <Button
            android:layout_width="124dp"
            android:layout_height="wrap_content"
            android:text="Print Picture" />

    </FrameLayout>
</RelativeLayout>
```



Hình 2.11. Frame Layout bị chồng View

ScrollView

ScrollView là một trường hợp đặc biệt của FrameLayout, trong đó cho phép người dùng có thể cuộn qua danh sách các view và cho phép có nhiều không gian hơn màn hình hiển thị. ScrollView chỉ có thể chứa một View con hoặc Viewgroup và thường là LinearLayout.

Đoạn code sau hiển thị một ScrollView trong một LinearLayout, trong đó có chứa các Button và các EditText

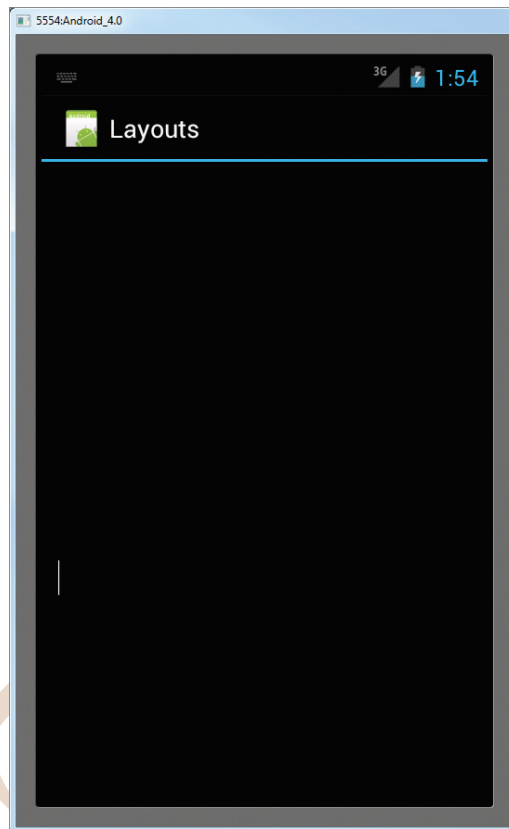
```
<ScrollView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android" >
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >
        <Button
            android:id="@+id/button1"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 1" />
        <Button
            android:id="@+id/button2"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 2" />
        <Button
            android:id="@+id/button3"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 3" />
        <EditText
            android:id="@+id/txt"
            android:layout_width="fill_parent"
            android:layout_height="600dp" />
        <Button
            android:id="@+id/button4"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 4" />
        <Button
            android:id="@+id/button5"
```

```

        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 5" />
    </LinearLayout>
</ScrollView>

```

Khi chạy sẽ có giao diện như hình dưới



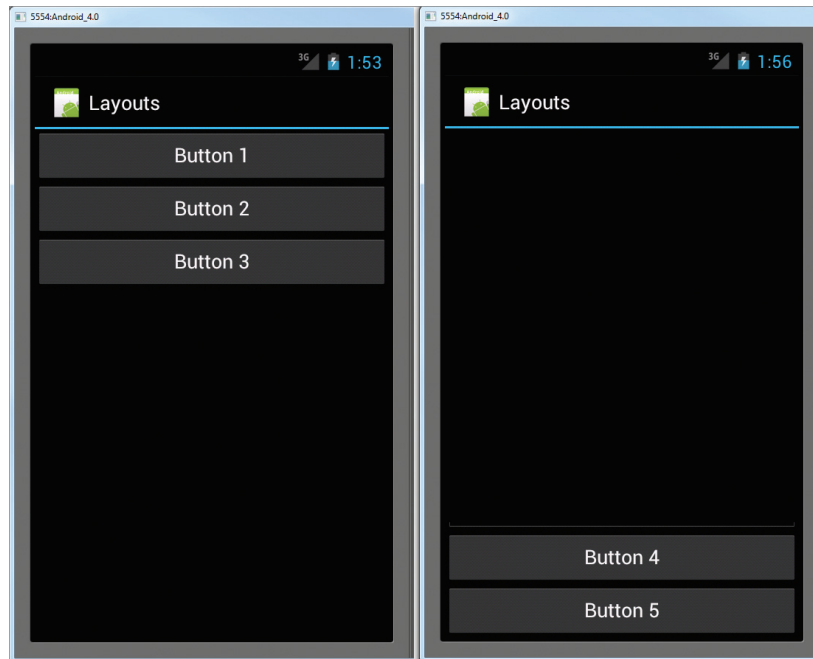
Hình 2.12. Scroll Layout

Vì EditText được tự động giữ con trỏ nên nó tràn ra toàn bộ các Activity (được xác định chiều cao là 600dp). Để tránh việc giữ con trỏ này, thêm các thuộc tính vào linearLayout

```

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:focusable="true"
    android:focusableInTouchMode="true" >

```



Hình 2.13. ScrollLayout đã được xử lý

Đôi khi, có thể ta muốn EditText sẽ được tự động trở con trỏ, nhưng lại không muốn bàn phím tự động xuất hiện, thêm thuộc tính vào phần tử <Activity> trong file Androidmanifest.xml:

```
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

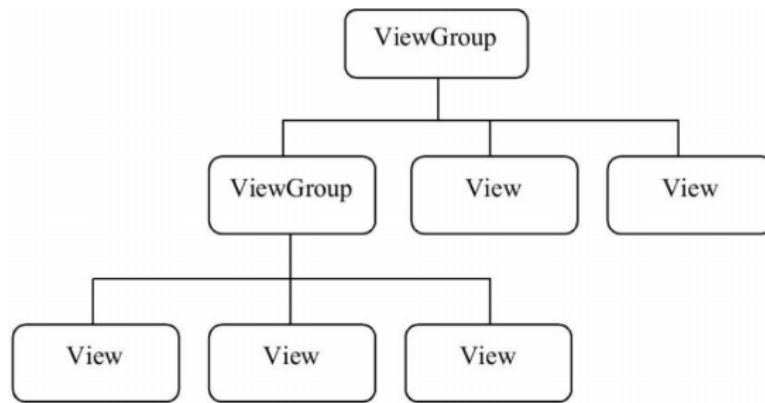
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
    android:windowSoftInputMode="stateHidden" >
</activity>
```

Phù hợp với chiều màn hình hiển thị

Một trong những tính năng chính của thiết bị thông minh hiện đại là khả năng thay đổi chiều màn hình hiển thị và Android cũng không phải là ngoại lệ. Android hỗ trợ 2 chiều màn hình là dọc (portrait) và ngang (landscape). Theo mặc định, khi ta thay đổi chiều màn hình hiển thị, cho thiết bị Android, activity hiện tại sẽ vẽ lại nội dung theo chiều màn hình mới. Điều này là do phương thức onCreate() của Activity được kích hoạt bất cứ khi nào có sự thay đổi trong hướng hiển thị.

2.3.2. Thiết kế giao diện với Views và ViewGroup

Các màn hình giao diện ứng dụng Android luôn được bố trí theo một kiểu cấu trúc phân cấp như hình dưới. Một màn hình giao diện ứng dụng hình thành từ một tập hợp các layout và các widget được bố trí có thứ tự. Để thể hiện một màn hình giao diện ứng dụng thì trong hàm onCreate() của mỗi Activity cần phải được gọi hàm setContentView(R.layout.main); hàm này sẽ tải giao diện từ file XML lên để phân tích thành mã bytecode.



Hình 2.14. View và ViewGroup

View

Trong phần này sẽ trình bày về lập trình ứng dụng với các view: tạo giao diện và xử lý sự kiện khi người dùng tương tác với các view trong ứng dụng Android.

Các view trong Android được định nghĩa sẵn trong lớp thư viện `android.view.View`. Các view này được chia thành các loại chính sau :

- Các view cơ bản
- Các picker view
- Các view hiển thị danh sách

Các view cơ bản

Các view cơ bản trong Android nhằm hiển thị thông tin văn bản hoặc cho phép lựa chọn thông tin, bao gồm : `TextView`, `EditText`, `Button`, `ImageButton`, `CheckBox`, `ToggleButton`, `RadioButton`, `RadioGroup`.

TextView

Khi tạo mới Android project, Eclipse sẽ tạo ra file giao diện có tên `main.xml` đặt trong thư mục `res/layout`. File giao diện này có chứa đối tượng `<TextView>`:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />
</LinearLayout>
```

`TextView` được sử dụng chỉ để hiển thị thông tin văn bản lên màn hình giao diện. Trong ví dụ trên `TextView` dùng hiển thị dòng chữ có nội dung tham chiếu tới chuỗi `hello` trong file `strings.xml` thông qua thuộc tính `android:text`.

Chi tiết về tất cả các thuộc tính của `TextView` được miêu tả trên:

<http://developer.android.com/reference/android/widget/TextView.html>

Thiết lập giá trị cho các thuộc tính của TextView có thể sử dụng một trong hai cách:

- Viết mã XML trực tiếp: gán giá trị cho các thuộc tính của TextView trong thẻ <TextView>.

- Gán giá trị cho thuộc tính thông qua cửa sổ giao diện : Trong phần Layout hiển thị giao diện đồ họa của activity, nhấn chuột phải vào đối tượng TextView, chọn show in -> Properties khi đó giao diện thiết lập giá trị cho thuộc tính cho đối tượng TextView sẽ được hiển thị.

Button, ImageButton, EditText, CheckBox, ToggleButton, RadioButton, RadioGroup

- Button: widget nút nhấn.
- ImageButton: có tác dụng tương tự như Button nhưng nội dung hiển thị có thể là hình ảnh.
- EditText: là lớp con của TextView cho phép người dùng soạn thảo nội dung văn bản.
- RadioButton: nút chọn, có hai trạng thái checked và unchecked.
- RadioGroup: dùng để nhóm các RadioButton lại với nhau, trong đó chỉ có 1 RadioButton ở trạng thái checked, các RadioButton khác ở trạng thái unchecked.
- ToggleButton: nút chọn, có hai trạng thái checked và unchecked nhận biết thông qua màu sắc sáng-tối.

Ví dụ tạo giao diện ứng dụng Android sử dụng một số view cơ bản: Button, ImageButton, EditText, CheckBox, ToggleButton, RadioButton, RadioGroup

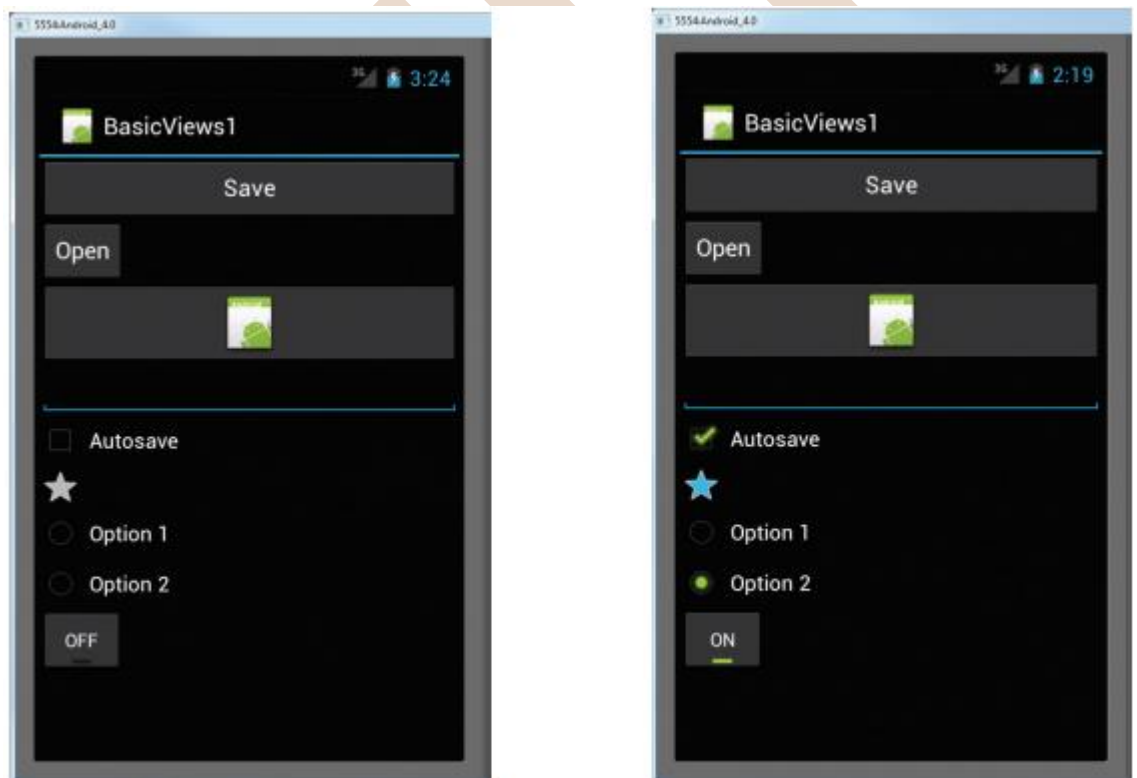
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <Button android:id="@+id/btnSave"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="save" />
    <Button android:id="@+id/btnOpen"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Open" />
    <ImageButton android:id="@+id/btnImg1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_launcher" />
    <EditText android:id="@+id/txtName"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
    <CheckBox android:id="@+id/chkAutosave"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
```

```

        android:text="Autosave" />
<CheckBox android:id="@+id/star"
    style="?android:attr/starStyle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
<RadioGroup android:id="@+id/rdbGp1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" >
    <RadioButton android:id="@+id/rdb1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Option 1" />
    <RadioButton android:id="@+id/rdb2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Option 2" />
</RadioGroup>
<ToggleButton android:id="@+id/toggle1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
</LinearLayout>

```

Debug trên thiết bị, quan sát sự thay đổi của các đối tượng sau khi được chọn



Hình 2.15. Các View cơ bản

Trước khi các đối tượng được chọn (trái) , sau khi các đối tượng được chọn (phải)

Giải thích cách các view cơ bản được tạo ra trong file giao diện main.xml:

- Do giao diện chứa các view được sắp xếp từ trên xuống dưới nên ta dùng ViewGroup là LinearLayout để bố trí các view này (các view được khai báo trong <LinearLayout>).

- Khai báo Button đầu tiên với thuộc tính `layout_width` có giá trị `fill_parent`, tức là Button có chiều rộng bằng chiều rộng màn hình:

```
<Button android:id="@+id/btnSave"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="save" />
```

- Khai báo Button thứ hai với thuộc tính `layout_width` có giá trị `wrap_content`, tức là có Button này có chiều rộng bằng chiều rộng nội dung button (“Open”).

```
<Button android:id="@+id/btnOpen"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Open" />
```

- Khai báo ImageButton có ảnh hiển thị được thiết lập thông qua thuộc tính `src`:

```
<ImageButton android:id="@+id/btnImg1"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:src="@drawable/ic_launcher" />
```

- Khai báo EditText cho phép người dùng nhập liệu thông tin dưới dạng văn bản. Thiết lập thuộc tính `layout_height` có giá trị `wrap_content` nhằm mục đích khi người dùng gõ vào nội dung văn bản dài thì chiều cao của đối tượng này sẽ tự động dẫn ra bằng nội dung văn bản muốn hiển thị:

```
<EditText android:id="@+id/txtName"
           android:layout_width="fill_parent"
           android:layout_height="wrap_content" />
```

- Khai báo CheckBox :

```
<CheckBox android:id="@+id/chkAutosave"
           android:layout_width="fill_parent"
           android:layout_height="wrap_content"
           android:text="Autosave" />
```

Có thể tùy chỉnh dạng hiển thị của checkbox thông qua thuộc tính `style` nhằm cho phép checkbox hiển thị dưới dạng ảnh (style có giá trị với format :

?[package:] [type:]name)

```
<CheckBox android:id="@+id/star"
           style="?android:attr/starStyle"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content" />
```

- Khai báo RadioGroup gom nhóm 2 RadioButton

```
<RadioGroup android:id="@+id/rdbGp1"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical" >
    <RadioButton android:id="@+id/rdb1"
                 android:layout_width="fill_parent"
```

```

        android:layout_height="wrap_content"
        android:text="Option 1" />
<RadioButton android:id="@+id/rdb2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Option 2" />
</RadioGroup>

```

Các RadioButton được sắp xếp theo chiều thẳng đứng thông qua việc gán giá trị vertical cho thuộc tính orientation. Khi muốn các RadioButton sắp xếp theo chiều nằm ngang thì gán giá trị cho thuộc tính orientation là horizontal

- Khai báo ToogleButton, button hình chữ nhật cho phép người dùng nhấn chọn on/off kèm chỉ thị về màu sắc.

```

<ToggleButton android:id="@+id/toggle1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

```

- Một điều quan trọng cần chú ý là tất cả các view được khai báo ở trên đều được đặt id cho view đó, mục đích để xử lý một số công việc theo yêu cầu. Dựa vào id ta sẽ lấy được view theo đúng id này thông qua phương thức View.findViewById() hoặc Activity.findViewById().

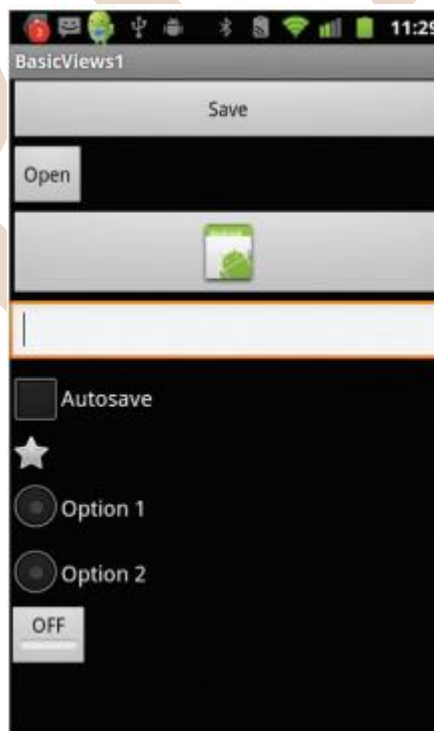
- Hình 3.9 là hình ảnh ứng dụng chạy trên thiết bị Android 4.0. Thay đổi giá trị cho thuộc tính android:minSdkVersion trong file AndroidManifest.xml là 10 và chạy lại ứng dụng trên thiết bị Google Nexus S (Android 2.3.6)

```

<uses-sdk android:minSdkVersion="10" />

```

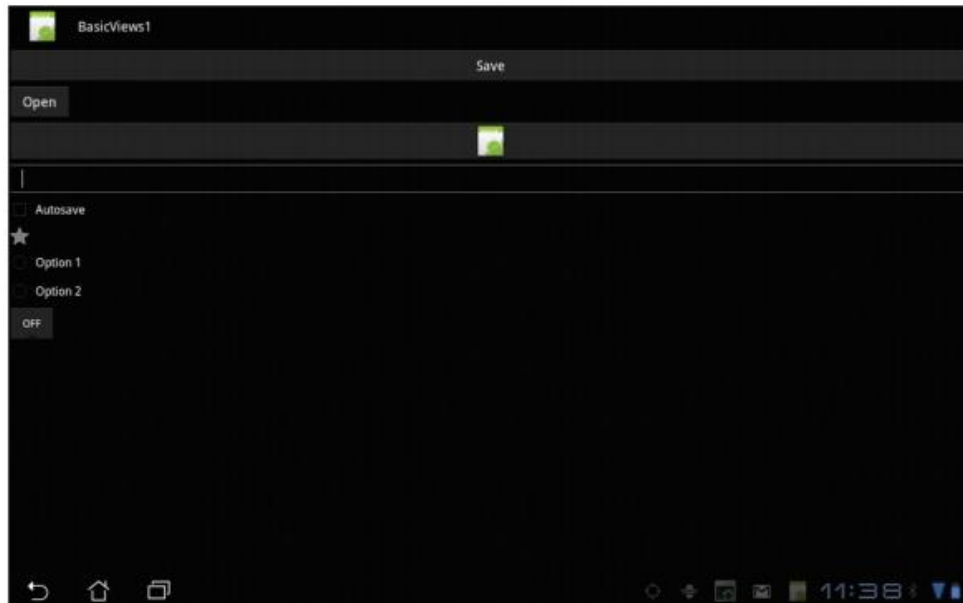
Khi đó hình ảnh cho ứng dụng như sau:



Hình 2.16. Radio Button

- Thay đổi giá trị cho thuộc tính android:minSdkVersion trong file AndroidManifest.xml là 13 và chạy lại ứng dụng trên thiết bị Asus Eee Pad

Transformer (Android 3.2.1). Khi đó hình ảnh cho ứng dụng như sau:



Hình 2.17. Khi thay đổi giá trị

ProgressBar

ProgressBar hiển thị thông tin trực quan về trạng thái nhiệm vụ trong ứng dụng đang thực hiện được tới đâu.

Ví dụ khi ứng dụng đang thực hiện một nhiệm vụ ở chế độ nền (background) như: trong khi ứng dụng tải (download) dữ liệu từ trang web thì ứng dụng cần cập nhật tới người dùng về tình trạng tải (tải được bao nhiêu %). Khi đó ProgressBar là một lựa chọn tốt để làm được điều này.

Minh họa cách sử dụng ProgressBar trong ứng dụng Android như sau:

- Từ Eclipse tạo Android project có tên BasicView2
- Chỉnh sửa file main.xml trong thư mục res/layout, chú ý thêm các dòng in đậm sau:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <ProgressBar android:id="@+id/progressbar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Viết code cho file BasicViews2Activity.java, chú ý thêm các dòng in đậm sau:

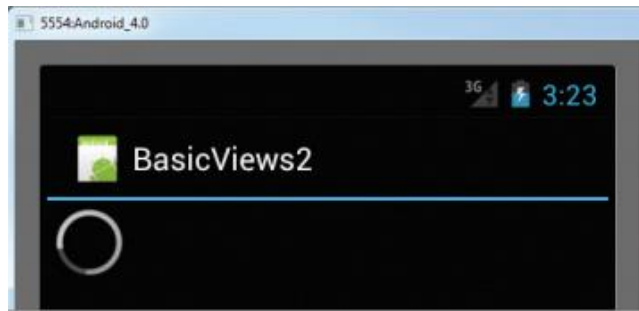
```
package net.learn2develop.BasicViews2;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.widget.ProgressBar;
public class BasicViews2Activity extends Activity {
```

```

static int progress;
ProgressBar progressBar;
int progressStatus = 0;
Handler handler = new Handler();
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    progress = 0;
    progressBar = (ProgressBar) findViewById(R.id.progressbar);
    //---do some work in background thread---
    new Thread(new Runnable()
    {
        public void run()
        {
            //---do some work here---
            while (progressStatus < 10)
            {
                progressStatus = doSomeWork();
            }
            //---hides the progress bar---
            handler.post(new Runnable()
            {
                public void run()
                {
                    //---0 - VISIBLE; 4 - INVISIBLE; 8 - GONE---
                    progressBar.setVisibility(View.GONE);
                }
            });
        }
        //---do some long running work here---
        private int doSomeWork()
        {
            try {
                //---simulate doing some work---
                Thread.sleep(500);
            } catch (InterruptedException e)
            {
                e.printStackTrace();
            }
            return ++progress;
        }
    }).start();
}

```

Khi Debug project trên Android emulator. Quá trình hoạt động của ProgressBar diễn ra như hình bên dưới (sau 5 giây, progressbar sẽ biến mất)



Hình 2.18. Hoạt động của ProgressBar trong ứng dụng Android

Thay đổi dạng hiển thị của ProgressBar bằng cách gán thuộc tính style của ProgressBar trong file giao diện xml. Các giá trị mà thuộc tính style có thể nhận được là :

- Widget.ProgressBar.Horizontal
- Widget.ProgressBar.Small
- Widget.ProgressBar.Large
- Widget.ProgressBar.Inverse
- Widget.ProgressBar.Small.Inverse
- Widget.ProgressBar.Large.Inverse

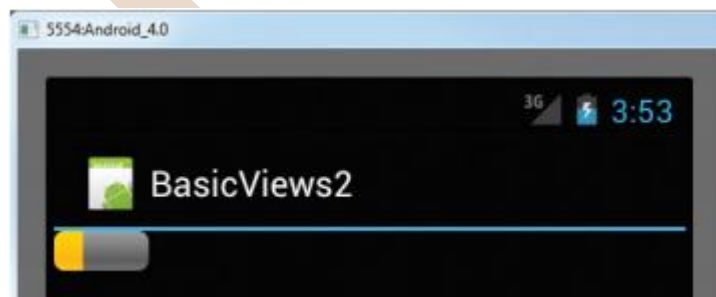
Ví dụ gán giá trị cho thuộc tính style của ProgressBar là Widget.ProgressBar.Horizontal

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <ProgressBar android:id="@+id/progressbar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        style="@android:style/Widget.ProgressBar.Horizontal" />

</LinearLayout>
```

Khi đó ProgressBar sẽ có dạng hiển thị như sau:



Hình 2.19. Dạng khác của ProgressBar

Giải thích:

Mặc định khi ProgressBar được gọi trong ứng dụng Android, nó sẽ hiển thị dưới dạng một hình ảnh chuyển động liên tục theo chu kỳ (ví dụ như biểu tượng hình tròn quay liên tục). ProgressBar ở chế độ này sử dụng phù hợp cho các ứng dụng cần thực hiện nhiệm vụ mà không xác định được thời gian hoàn thành của nhiệm vụ đó, như : gửi

dữ liệu của ứng dụng qua mạng và chờ đáp ứng từ server trả về. Khi nhận được đáp ứng trả về cho biết nhiệm vụ đã thực hiện xong, ProgressBar cần dừng hoạt động. Để làm được việc này, lập trình viên cần thực hiện một số công việc sau:

Sử dụng một đối tượng Thread kết hợp với một đối tượng Runnable. Phương thức run() thực thi thread này sẽ gọi phương thức doSomeWork() để thực hiện nhiệm vụ khi ProgressBar đang chạy. Khi nhiệm vụ hoàn thành (sau 5 giây), đối tượng Handler được sử dụng để gửi một thông điệp yêu cầu ProgressBar biến mất.

```
//---do some work in background thread---
new Thread(new Runnable()
{
    public void run()
    {
        //---do some work here---
        while (progressStatus < 10)
        {
            progressStatus = doSomeWork();
        }
        //---hides the progress bar---
        handler.post(new Runnable()
        {
            public void run()
            {
                //---0 - VISIBLE; 4 - INVISIBLE; 8 - GONE---
                progressBar.setVisibility(View.GONE);
            }
        });
        //---do some long running work here---
        private int doSomeWork()
        {
            try {
                //---simulate doing some work---
                Thread.sleep(500);
            } catch (InterruptedException e)
            {
                e.printStackTrace();
            }
            return ++progress;
        }
    }
}).start();
```

Chú ý rằng phương thức setVisibility() có tham số truyền vào nhận một trong ba giá trị : View.VISIBLE, View.INVISIBLE, View.GONE. Trong đó:

- Dùng tham số View.VISIBLE khi muốn hiển thị ProgressBar.
- Dùng tham số View.INVISIBLE khi muốn ẩn ProgressBar tuy nhiên ProgressBar vẫn tồn tại như một thành phần của activity.
- Dùng tham số View.GONE khi muốn ProgressBar biến mất, khi đó ProgressBar bị giải phóng hoàn toàn và không còn là một thành phần của activity.

AutoCompleteTextView

AutoCompleteTextView tương tự EditText nhưng nó cho phép hiển thị danh sách gợi ý tự động ứng với các cụm từ người dùng nhập vào.

Minh họa

- Từ Eclipse tạo Android project có tên BasicView3
- Chỉnh sửa file main.xml trong thư mục res/layout, chú ý thêm các dòng in đậm sau:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" | >

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Name of President" />

<AutoCompleteTextView android:id="@+id/txtCountries"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />

</LinearLayout>
```

- Viết code cho file BasicViews3Activity.java, chú ý thêm các dòng in đậm sau:

```
package net.learn2develop.BasicViews3;
import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
public class BasicViews3Activity extends Activity {
    String[] presidents = {
        "Dwight D. Eisenhower",
        "John F. Kennedy",
        "Lyndon B. Johnson",
        "Richard Nixon",
        "Gerald Ford",
        "Jimmy Carter",
        "Ronald Reagan",
        "George H. W. Bush",
        "Bill Clinton",
        "George W. Bush",
        "Barack Obama"
    };
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_dropdown_item_1line, presidents);
        AutoCompleteTextView textView = (AutoCompleteTextView)
            findViewById(R.id.txtCountries);
        textView.setThreshold(3);
        textView.setAdapter(adapter);
    }
}
```

}

Debug ứng dụng trên Virtual Device



Hình 2.20. Autocomplete Textbox

Giải thích:

Trong lớp BasicViews3Activity, trước tiên khai báo một mảng String nhằm chứa một danh sách các vị tổng thống.

```
String[] presidents = {  
    "Dwight D. Eisenhower",  
    "John F. Kennedy",  
    "Lyndon B. Johnson",  
    "Richard Nixon",  
    "Gerald Ford",  
    "Jimmy Carter",  
    "Ronald Reagan",  
    "George H. W. Bush",  
    "Bill Clinton",  
    "George W. Bush",  
    "Barack Obama"  
};
```

Trong phương thức onCreate() của activity, tạo ra đối tượng ArrayAdapter quản lý mảng chuỗi trên đồng thời được hiển thị bởi AutoCompleteTextView. Trong ví dụ này, thiết lập chế độ hiển thị của AutoCompleteTextView ở chế độ simple_dropdown_item_1line:

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
    android.R.layout.simple_dropdown_item_1line, presidents);
```

Phương thức setThreshold() dùng để cài đặt số lượng ký tự tối thiểu mà người dùng phải gõ vào nếu muốn danh sách gợi ý ứng với cụm từ gõ vào xuất hiện. Danh sách gợi ý này xuất hiện dưới dạng menu xổ xuống (drop-down menu)

```
textView.setThreshold(3);
```

Danh sách gợi ý của AutoCompleteTextView được chứa trong đối tượng ArrayAdapter:

```
textView.setAdapter(adapter);
```

Các picker view

Picker được chia làm 2 loại : TimePicker, DatePicker. Việc sử dụng 2 loại picker này cho phép người dùng chọn thời gian, ngày tháng năm trên giao diện ứng dụng Android.

TimePicker

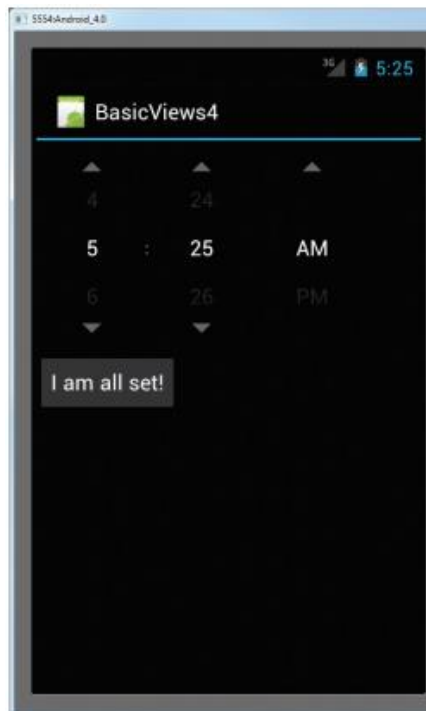
TimePicker cho phép người dùng ứng dụng chọn thời gian trong ngày với 2 chế độ: chế độ 24 giờ và chế độ 12 giờ (AM/PM).

Minh họa cách sử dụng TimePicker trong ứng dụng Android như sau:

- Từ Eclipse tạo Android project có tên BasicView4
- Chỉnh sửa file main.xml trong thư mục res/layout, chú ý thêm các dòng in đậm sau:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical" >
<TimePicker android:id="@+id/timePicker"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
<Button android:id="@+id/btnSet"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="I am all set!" android:onClick="onClick" />
</LinearLayout>
```

- Nhấn F11 để debug ứng dụng trên Android emulator. Để lựa chọn thời gian trong ngày, ngoài việc nhấn mũi tên lên xuống để thay đổi giá trị thời gian, bạn có thể sử dụng phím số trên bàn phím và nhấn nút AM để chuyển chế độ giữa AM và PM.



Hình 2.21. Hoạt động của TimePicker trong activity ở chế độ 12 giờ
Quay trở lại IDE, viết code cho file BasicViews4Activity.java, chú ý thêm các dòng in đậm sau:

```
package net.learn2develop.BasicViews4;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.TimePicker;
import android.widget.Toast;

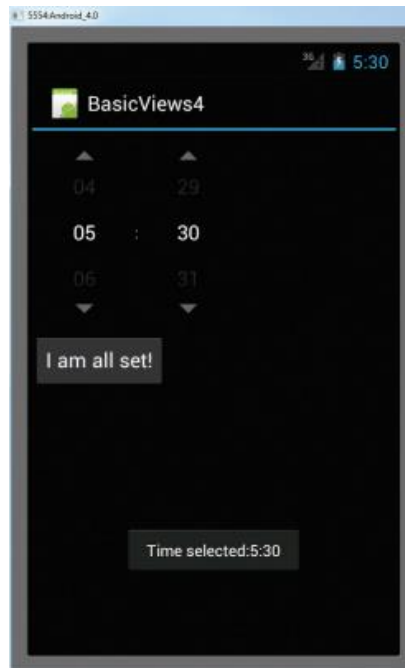
public class BasicViews4Activity extends Activity {
    TimePicker timePicker;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        timePicker = (TimePicker) findViewById(R.id.timePicker);
        timePicker.setIs24HourView(true);
    }

    public void onClick(View view) {
        Toast.makeText(getApplicationContext(),
            "Time selected: " +
            timePicker.getCurrentHour() +
            ":" + timePicker.getCurrentMinute(),
            Toast.LENGTH_SHORT).show();
    }
}
```

Nhấn F11 để debug lại ứng dụng trên Android emulator. TimePicker lúc này sẽ hiển thị ở chế độ 24 giờ. Nhấn vào Button sẽ hiển thị thời gian mà bạn vừa thiết lập cho TimePicker.



Hình 2.22. Hoạt động của TimePicker trong activity ở chế độ 24 giờ

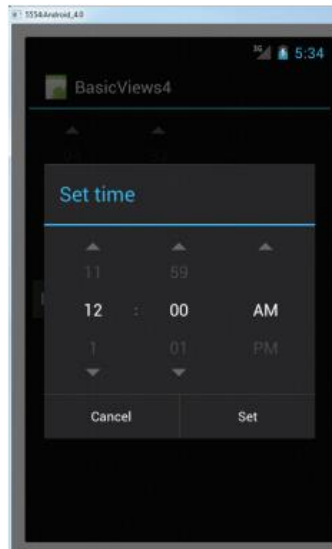
Giải thích

TimePicker hiển thị giao diện cho phép người dùng ứng dụng thiết lập thời gian theo chuẩn. Mặc định nó hiển thị ở chế độ 12 giờ (AM/PM). Nếu muốn hiển thị TimePicker ở chế độ 24 giờ, lập trình viên cần gọi phương thức `setIs24HourView()` của nó.

Phương thức `getCurrentHour()` và `getCurrentMinute()` cho phép lấy về thời gian thiết lập cho TimePicker trên giao diện.

```
Toast.makeText(getApplicationContext(),
    "Time selected:" +
    timePicker.getCurrentHour() +
    ":" + timePicker.getCurrentMinute(),
    Toast.LENGTH_SHORT).show();
```

Ví dụ minh họa ở trên, TimePicker được sử dụng trong Activity. Trên thực tế, TimePiker thường được hiển thị trong cửa sổ dialog, nhằm mục đích khi người dùng thiết lập xong thời gian trên dialog thì nó sẽ biến mất cũng như không chiếm không gian hiển thị trên activity.



Hình 2.23. Hoạt động của TimePicker trong dialog ở chế độ 12 giờ

DatePicker

DatePicker cho phép người dùng ứng dụng chọn ngày tháng năm trên giao diện ứng dụng.

Minh họa:

Sử dụng lại Android project có tên BasicView4.

Chỉnh sửa file main.xml trong thư mục res/layout, chú ý thêm các dòng in đậm sau:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <Button android:id="@+id/btnSet"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="I am all set!"
        android:onClick="onClick" />

    <DatePicker android:id="@+id/datePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <TimePicker android:id="@+id/timePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Nhấn F11 để debug ứng dụng trên Android emulator. Nhấn tiếp Ctrl+F11 để thay đổi hướng hiển thị của Android emulator nằm ngang (landscape)

Viết code cho file BasicViews4Activity.java, chú ý thêm các dòng in đậm sau:

```
package net.learn2develop.BasicViews4;
import java.text.SimpleDateFormat;
import java.util.Date;
import android.app.Activity;
```

```

import android.app.Dialog;
import android.app.TimePickerDialog;
import android.os.Bundle;
import android.view.View;
import android.widget.DatePicker;
import android.widget.TimePicker;
import android.widget.Toast;
public class BasicViews4Activity extends Activity {
    TimePicker timePicker; DatePicker datePicker;
    int hour, minute;
    static final int TIME_DIALOG_ID = 0;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        timePicker = (TimePicker) findViewById(R.id.timePicker);
        timePicker.setIs24HourView(true);
        // showDialog(TIME_DIALOG_ID);
        datePicker = (DatePicker) findViewById(R.id.datePicker);
    }
    public void onClick(View view) {
        Toast.makeText(getApplicationContext(),
            "Date selected: " + (datePicker.getMonth() + 1) +
            "/" + datePicker.getDayOfMonth() +
            "/" + datePicker.getYear() + "\n" +
            "Time selected: " + timePicker.getCurrentHour() +
            ":" + timePicker.getCurrentMinute(),
            Toast.LENGTH_SHORT).show();
    }
}

```

Giải thích

Phương thức `getMonth()`, `getDayOfMonth()`, `getYear()` của `DatePicker` cho phép lấy về tháng, ngày, năm thiết lập cho `DatePicker` trên giao diện.

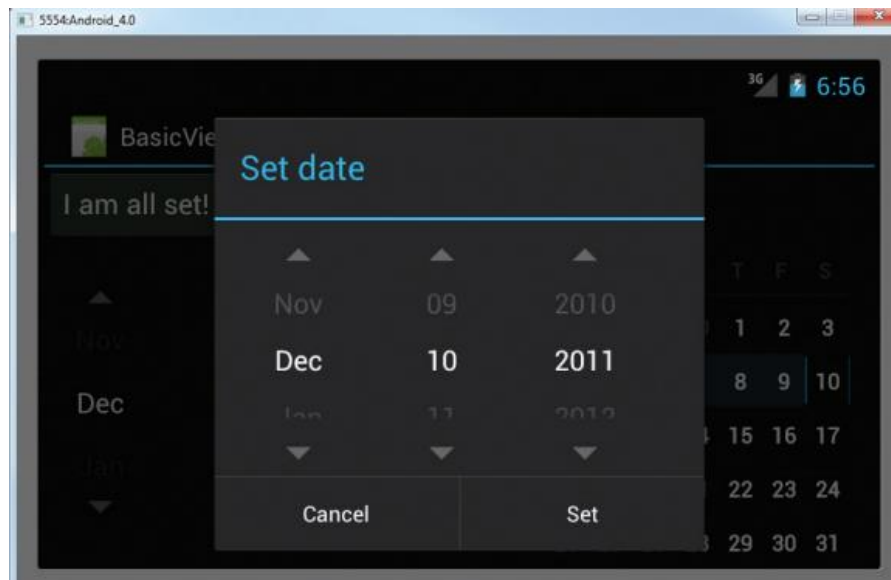
```

"Date selected: " + (datePicker.getMonth() + 1) +
"/" + datePicker.getDayOfMonth() +
"/" + datePicker.getYear() + "\n" +

```

Chú ý rằng `getMonth()` trả về giá trị 0 nếu là tháng 1, 1 nếu là tháng 2... Do đó cần tăng giá trị nhận về của phương thức này để hiển thị đúng tháng tới người dùng.

Ví dụ minh họa ở trên, `DatePicker` được sử dụng trong `Activity`. Trên thực tế, `DatePiker` thường được hiển thị trong cửa sổ dialog, nhằm mục đích khi người dùng thiết lập xong thời gian trên dialog thì nó sẽ biến mất cũng như không chiếm không gian hiển thị trên activity.



Hình 2.24. Hoạt động của DatePicker trong dialog

Các view hiển thị danh sách

Các view hiển thị danh sách trong Android nhằm hiển thị danh sách mục thông tin. Có 2 loại view hiển thị danh sách: ListView, SpinnerView.

ListView

ListView hiển thị danh sách mục thông tin có thể cuộn theo chiều dọc.

Mình họa cách sử dụng ListView trong ứng dụng Android như sau:

- Từ IDE tạo Android project có tên BasicView5
- Viết code cho file BasicViews5Activity.java, chú ý thêm các dòng in đậm sau:

```
package net.learn2develop.BasicViews5;
import android.app.ListActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;
public class BasicViews5Activity extends ListActivity {
String[] presidents = {
"Dwight D. Eisenhower",
"John F. Kennedy",
"Lyndon B. Johnson",
"Richard Nixon",
"Gerald Ford",
"Jimmy Carter",
"Ronald Reagan",
"George H. W. Bush",
"Bill Clinton",
"George W. Bush", "Barack Obama"
};
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
//---no need to call this---
```

```

//setContentView(R.layout.main);
setListAdapter(new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, presidents));
public void onItemClick(
ListView parent, View v, int position, long id)
{
    Toast.makeText(this,
    "You have selected " + presidents[position],
    Toast.LENGTH_SHORT).show();
}
}

```

Debug ứng dụng:



Hình 2.25. Hoạt động của ListView

Giải thích

Chú ý lớp BasicViews5Activity kế thừa từ lớp ListActivity, lớp ListActivity kế thừa từ lớp Activity. Khi đó bản thân lớp ListActivity chứa ListView, vì vậy trong phương thức onCreate() không cần gọi phương thức setContentView() để tải giao diện từ main.xml.

Phương thức onCreate() sử dụng phương thức setListAdapter() để kết nối danh sách mục dữ liệu vào activity. Phương thức setListAdapter() có tham số truyền vào là 1 đối tượng của lớp ArrayAdapter. Lớp ArrayAdapter quản lý mảng các mục thông tin mà bạn muốn hiển thị lên ListView ở chế độ simple_list_item1:

```

setListAdapter(new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_1, presidents));

```

Phương thức onItemClick() được gọi khi 1 mục thông tin của danh sách được chọn

```

public void onListItemClick(
    ListView parent, View v, int position, long id)
{
    Toast.makeText(this,
        "You have selected " + presidents[position],
        Toast.LENGTH_SHORT).show();
}

```

Spinner

Khi activity không có nhiều không gian để hiển thị danh mục thông tin dài như ListView, thì có thể dùng Spinner thay thế. Spinner chứa danh sách mục thông tin cho phép người dùng lựa chọn, nhưng tại một thời điểm chỉ có một mục tin được hiển thị trên giao diện.

Minh họa

- Từ Eclipse tạo Android project có tên BasicView6
- Chỉnh sửa file main.xml trong thư mục res/layout, chú ý thêm các dòng in đậm sau:

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    >
    <Spinner
        android:id="@+id/spinner1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:drawSelectorOnTop="true"
        />
</LinearLayout>

```

- Tạo ra 1 mảng String chứa danh sách mục thông tin trong file strings.xml ở folder res/value

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, BasicViews6Activity!</string>
    <string name="app_name">BasicViews6</string>
    <string-array name="presidents_array">
        <item>Dwight D. Eisenhower</item>
        <item>John F. Kennedy</item>
        <item>Lyndon B. Johnson</item>
        <item>Richard Nixon</item>
        <item>Gerald Ford</item>
        <item>Jimmy Carter</item>
        <item>Ronald Reagan</item>
        <item>George H. W. Bush</item>
        <item>Bill Clinton</item>
        <item>George W. Bush</item>
        <item>Barack Obama</item>
    </string-array>
</resources>

```

Viết code cho file BasicViews6Activity.java, chú ý thêm các dòng in đậm sau:

```

package net.learn2develop.BasicViews6;
import android.app.Activity;
import android.os.Bundle;

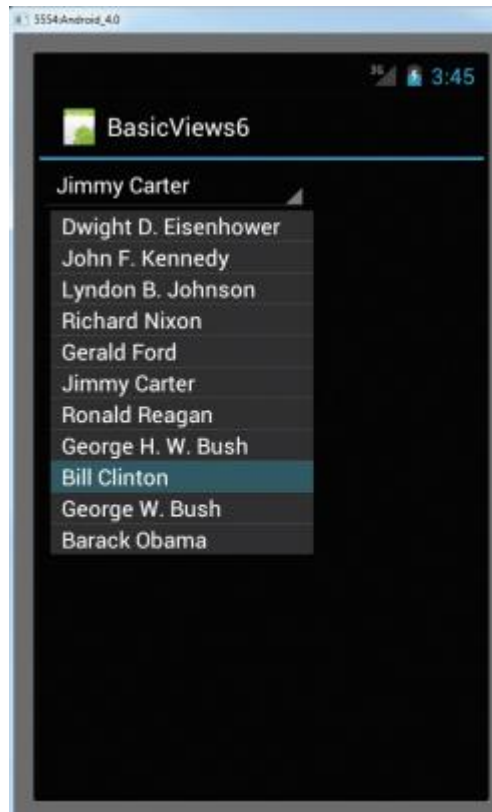
```

```

import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;
public class BasicViews6Activity extends Activity {
    String[] presidents;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        presidents =
            getResources().getStringArray(R.array.presidents_array);
        Spinner s1 = (Spinner) findViewById(R.id.spinner1);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_spinner_item, presidents);
        s1.setAdapter(adapter);
        s1.setOnItemClickListener(new OnItemClickListener()
        {
            @Override
            public void onItemClick(AdapterView<?> arg0,
                View arg1, int arg2, long arg3)
            {
                int index = arg0.getSelectedItemPosition();
                Toast.makeText(getApplicationContext(),
                    "You have selected item : " + presidents[index],
                    Toast.LENGTH_SHORT).show();
            }
            @Override
            public void onNothingSelected(AdapterView<?> arg0) { }
        });
    }
}

```

Debug và Nhấn vào SpinnerView sẽ xổ xuống danh sách tên các vị tổng thống



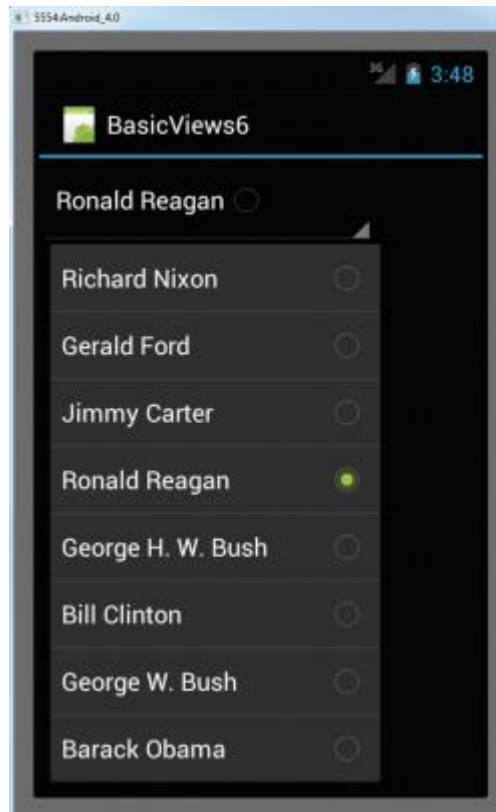
Hình 2.26. Hoạt động của SpinnerView

Giải thích

Phương thức `onNothingSelected()` được gọi khi người dùng nhấn nút back, khi đó danh sách mục tin sẽ không xuất hiện trên màn hình. Trong trường hợp này không có mục tin nào được chọn.

Ngoài ra, thay vì hiển thị danh mục tin dưới dạng danh sách đơn giản ở chế độ `simple_spinner_item`, lập trình viên có thể hiển thị danh mục tin dưới dạng danh sách radio button. Để làm việc này, chỉnh sửa tham số thứ 2 trong hàm khởi tạo của `ArrayAdapter` từ chế độ `simple_spinner_item` thành `simple_list_item_single_choice`:

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
    android.R.layout.simple_list_item_single_choice, presidents);
```

Hình 2.27. Spinner

2.3.3. Xử lý Sự kiện trong Android

Phần trên vừa trình bày về cách tạo giao diện ứng dụng Android với các view. Khi người dùng sử dụng ứng dụng, người dùng sẽ tương tác với các view này, các view nhận được các tương tác từ người dùng sẽ xử lý các sự kiện tương ứng. Các sự kiện này được Android Framework quản lý bởi 1 hàng đợi (queue) hoạt động theo cơ chế FIFO (First In First Out).

- Các bước cần làm để xử lý sự kiện cho các đối tượng giao diện:
- Đăng ký lắng nghe sự kiện (Event Listeners Registration).
- Khai báo đối tượng nhận thông báo khi có sự kiện xuất hiện (Event Listeners).
- Xử lý sự kiện (Event Handlers)

Đăng ký lắng nghe sự kiện

Có 3 cách để đăng ký lắng nghe sự kiện cho đối tượng giao diện trong ứng dụng:

- Sử dụng Anonymous Inner Class
- Khai báo thuộc tính sự kiện cho view trong file giao diện XML.
- Khai báo lớp Activity thực thi giao diện Listener.

Bảng 2.2. Các Event Listener và Event Handler

Event Handler	Event Listener
onClick()	OnClickListener() Được gọi khi người dùng nhấn/ chạm vào widget. Event Listener này sẽ gọi phương thức onClick() để xử lý sự kiện.

onLongClick()	OnLongClickListener() Được gọi khi người dùng nhấn/ chạm li vài giây vào widget. Event Listener này sẽ gọi phương thức onLongClick() để xử lý sự kiện.
onFocusChange()	OnFocusChangeListener() Được gọi khi người dùng không tập trung vào widget. Event Listener này sẽ gọi phương thức onFocusChange() để xử lý sự kiện.
onKey()	OnFocusChangeListener() Được gọi khi người dùng thao tác với các phím bấm của thiết bị. Event Listener này sẽ gọi phương thức onKey() để xử lý sự kiện.
onTouch()	OnTouchListener() Được gọi khi người dùng chạm vào màn hình thiết bị. Event Listener này sẽ gọi phương thức onTouch() để xử lý sự kiện.
onMenuItemClick()	OnMenuItemClickListener() Được gọi khi người dùng chọn menu item. Event Listener này sẽ gọi phương thức onMenuItemClick () để xử lý sự kiện.

Minh họa việc xử lý sự kiện cho view trong ứng dụng Android theo 3 cách đăng ký sự kiện

Cách 1: Đăng ký và xử lý sự kiện sử dụng Anonymous Inner Class

• Sử dụng project BasicView1 đã tạo ra ở trên. Chỉnh sửa mã nguồn cho file BasicView1Activity.java bằng cách thêm vào các nội dung in đậm sau:

```
package net.learn2develop.BasicViews1;
import android.app.Activity;
import android.os.Bundle; import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.RadioGroup.OnCheckedChangeListener;
import android.widget.Toast;
import android.widget.ToggleButton;
public class BasicViews1Activity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //---Button view---
        Button btnOpen = (Button) findViewById(R.id.btnOpen);
        btnOpen.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                DisplayToast("You have clicked the Open button");
            }
        });
        //---Button view---
        Button btnSave = (Button) findViewById(R.id.btnSave);
```

```

btnSave.setOnClickListener(new View.OnClickListener()
{
    public void onClick(View v) {
        DisplayToast("You have clicked the Save button");
    }
});
//---CheckBox---
CheckBox checkBox = (CheckBox) findViewById(R.id.chkAutosave);
checkBox.setOnClickListener(new View.OnClickListener()
{
    public void onClick(View v) {
        if (((CheckBox)v).isChecked())
            DisplayToast("CheckBox is checked");
        else
            DisplayToast("CheckBox is unchecked");
    }
});
//---RadioButton---
RadioGroup radioGroup = (RadioGroup) findViewById(R.id.rdbGp1);
radioGroup.setOnCheckedChangeListener(new
OnCheckedChangeListener()
{
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        RadioButton rbl = (RadioButton) findViewById(R.id.rdb1);
        if (rbl.isChecked()) {
            DisplayToast("Option 1 checked!");
        } else {
            DisplayToast("Option 2 checked!");
        }
    }
});
//---ToggleButton---
ToggleButton toggleButton =
(ToggleButton) findViewById(R.id.toggle1);
toggleButton.setOnClickListener(new View.OnClickListener()
{
    public void onClick(View v) {
        if (((ToggleButton)v).isChecked())
            DisplayToast("Toggle button is On");
        else
            DisplayToast("Toggle button is Off");
    }
});
private void DisplayToast(String msg)
{
    Toast.makeText(getApplicationContext(), msg, Toast.LENGTH_SHORT).show();
}
}

```

Bấm F11 để debug ứng dụng trên Android emulator.

Bấm chọn các view và quan sát nội dung thông điệp hiển thị ở cửa sổ Toast.

Giải thích:

Trong hàm onCreate() cần lấy về các view theo đúng id của view đó thông qua

hàm findViewById():

```
//---Button view---
Button btnOpen = (Button) findViewById(R.id.btnOpen);
```

Phương thức setOnClickListener() đăng ký phương thức callback onClick(), onClick() được gọi khi Button btnOpen được click.

```
btnOpen.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        DisplayToast("You have clicked the Open button");
    }
});
```

CheckBox

Để xác định trạng thái của CheckBox, ta ép kiểu tham số của hàm onClick() thành CheckBox, sau đó gọi phương thức isChecked()

```
CheckBox checkBox = (CheckBox) findViewById(R.id.chkAutosave);
checkBox.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        if (((CheckBox)v).isChecked())
            DisplayToast("CheckBox is checked");
        else
            DisplayToast("CheckBox is unchecked");
    }
});
```

RadioButton

Phương thức setOnCheckedChangeListener() của RadioGroup đăng ký phương thức callback onCheckedChanged().

```
//---RadioButton---
RadioGroup radioGroup = (RadioGroup) findViewById(R.id.rdbGp1);
radioGroup.setOnCheckedChangeListener(new OnCheckedChangeListener() {
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        RadioButton rdb1 = (RadioButton) findViewById(R.id.rdb1);
        if (rdb1.isChecked()) {
            DisplayToast("Option 1 checked!");
        } else {
            DisplayToast("Option 2 checked!");
        }
    }
});
```

Hàm onCheckedChanged() được gọi khi một trong số RadioButton chuyển sang trạng thái checked. Trong hàm này, RadioButton được lấy về theo id (rdb1), sau đó gọi hàm isChecked() để xác định RadioButton rdb1 được chọn hay không. Ngoài ra hàm onCheckedChanged() có chứa tham số thứ 2 là checkedId chính là id của RadioButton được chọn.

Cách 2: Đăng ký sự kiện cho view trong file giao diện, cài đặt phương thức xử lý sự kiện trong source code

Thêm thuộc tính sự kiện cho view trong file giao diện .xml. Cụ thể : để xử lý sự kiện click vào Button, ta thêm thuộc tính onClick của <Button> trong main.xml

```
<Button android:id="@+id/btnSave"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/save"
        android:onClick="btnSaved_clicked"/>
```

Cách khai báo này xác định sự kiện click của button. Giá trị của thuộc tính onClick

chính là tên của sự kiện. Để xử lý sự kiện cho button này, thêm hàm `btnSaved_clicked()` trong file `BasicViews1Activity.java` (chú ý hàm `btnSaved_clicked()` có tham số truyền vào là một đối tượng của lớp `View`).

```
public class BasicViews1Activity extends Activity {

    public void btnSaved_clicked (View view) {
        DisplayToast("You have clicked the Save button1");
    }

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //...
    }

    private void DisplayToast(String msg)
    {
        Toast.makeText(getApplicationContext(), msg,
            Toast.LENGTH_SHORT).show();
    }
}
```

Cách 3: Đăng sự kiện bằng cách khai báo lớp Activity thực thi giao diện Listener

Viết code cho `BasicViews1Activity.java` như sau

```
package net.learn2develop.BasicViews1;
import android.app.Activity;
import android.os.Bundle; import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.RadioGroup.OnCheckedChangeListener;
import android.widget.Toast;
import android.widget.ToggleButton;
public class BasicViews1Activity extends Activity implements
OnClickListener{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button btnOpen = (Button) findViewById(R.id.btnOpen);
        // -- register click event with button ---
        btnOpen.setOnClickListener(this);
    }
    //--- Implement the OnClickListener callback
    public void onClick(View v) {
        if(v.getId() == R.id.button_s)
        {
            DisplayToast("You have clicked the Open button");
        }
    }
}
```

2.4. QUẢN LÝ DỮ LIỆU

Lưu trữ dữ liệu là việc làm cần thiết trong hầu hết các ứng dụng. Trong phần này,

bài giảng trình bày về 3 kiểu quản lý dữ liệu trong ứng dụng Android:

- Quản lý dữ liệu dạng key-value
- Quản lý dữ liệu từ file trong bộ nhớ thiết bị
- Quản lý dữ liệu với cơ sở dữ liệu quan hệ
- Quản lý dữ liệu với Content Providers

2.4.1. Shared Preferences

Android cung cấp tập các API SharedPreferences cho phép quản lý lượng nhỏ dữ liệu đơn giản của ứng dụng dạng key-value. SharedPreferences có thể lưu trữ dữ liệu cho một Activity cụ thể hoặc chia sẻ dữ liệu giữa tất cả các Activity trong/ ngoài ứng dụng của bạn, dữ liệu khi đó được lưu trữ trong một file xml. Cách quản lý dữ liệu này phù hợp khi muốn lưu trạng thái ứng dụng.

Ví dụ: trong ứng dụng có tùy chọn cho phép người dùng cài đặt kích cỡ chữ hiển thị. Yêu cầu khi người dùng chọn cài đặt xong và tắt ứng dụng đi thì lần sau mở lại ứng dụng vẫn hiển thị nội dung với kích cỡ chữ đã được cài đặt trước đó. Để làm được việc này thì lưu trữ thông tin vào SharedPreferences là lựa chọn tốt hơn so với lưu dữ liệu vào file hay cơ sở dữ liệu.

Tạo và kết nối tới file SharedPreferences

Android cung cấp 2 phương thức để tạo mới file SharedPreferences hoặc tham chiếu tới file SharedPreferences có sẵn:

- **getSharedPreferences():** sử dụng để tạo mới/ tham chiếu tới nhiều file SharedPreferences với tên file là tham số truyền vào cho phương thức. Phương thức này có thể được gọi từ bất cứ Context nào trong ứng dụng.
- **getPreferences():** Activity gọi phương thức này để tạo mới / tham chiếu tới một file SharedPreferences. Hàm này tham chiếu tới 1 file SharedPreferences với tên mặc định do đó không cần truyền tham số là tên file cho hàm.

Giá trị trả về của 2 phương thức này là đối tượng SharedPreferences, dùng đối tượng này để đọc/ ghi dữ liệu vào file SharedPreferences trên thiết bị.

Đoạn code sau thực thi bên trong class Fragment dùng để tham chiếu tới các file SharedPreferences, sử dụng phương thức getSharedPreferences()

```
Context context = getActivity();
SharedPreferences sharedPref = context.getSharedPreferences(
    getString(R.string.preference_file_key), Context.MODE_PRIVATE);
```

- Tên các file SharedPreferences xác định thông qua preference_file_key được khai báo trong strings.xml.
- File được mở ở chế độ private (MODE_PRIVATE) cho biết file SharedPreferences chỉ được truy cập bên trong nội bộ ứng dụng. Nếu tạo file ở chế độ MODE_WORLD_READABLE, MODE_WORLD_WRITEABLE thì bất cứ ứng dụng trên thiết bị đều

có thể truy cập dữ liệu trong file.

Để tham chiếu tới 1 file SharedPreferences, sử dụng phương thức `getPreferences()`

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
```

Ghi dữ liệu vào SharedPreferences

Tạo đối tượng `SharedPreferences.Editor` bằng cách gọi phương thức `edit()` của đối tượng `SharedPreferences`. Sau đó gọi phương thức như `putInt()`, `putString()`... với tham số truyền vào là giá trị dạng key-value để lưu dữ liệu vào file.

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPref.edit();
editor.putInt(getString(R.string.saved_high_score), newHighScore);
editor.commit();
```

Gọi phương thức `commit()` của đối tượng `SharedPreferences.Editor` để hoàn tất việc ghi dữ liệu.

Đọc dữ liệu từ SharedPreferences

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
int defaultValue = getResources().getInteger(R.string.saved_high_score_default);
long highScore = sharedPref.getInt(getString(R.string.saved_high_score), defaultValue);
```

2.4.2. Đọc và Ghi file

Android cung cấp tập các API File cho phép ứng dụng đọc/ ghi lượng lớn dữ liệu vào file trong bộ nhớ thiết bị. Tập các API File được cung cấp bởi gói `java.io`.

Các thiết bị Android có 2 bộ nhớ dữ liệu: bộ nhớ trong (Internal storage) và bộ nhớ ngoài như thẻ SD (External storage).

Bảng 2.3. Bộ nhớ trong và Bộ nhớ ngoài

Bộ nhớ trong (Internal storage)	Bộ nhớ ngoài (External storage)
Luôn sẵn sàng	Không phải lúc nào cũng sẵn sàng vì người dùng có thể thêm vào hoặc gỡ bỏ bộ nhớ ngoài khỏi thiết bị.
Các file lưu trữ ở bộ nhớ trong chỉ được phép truy cập bởi các ứng dụng tạo ra chúng.	Các file lưu trữ ở bộ nhớ ngoài có thể được truy cập tự do.
Khi gỡ bỏ ứng dụng khỏi thiết bị thì hệ thống cũng gỡ bỏ luôn những file do ứng dụng tạo ra ở bộ nhớ trong.	Khi gỡ bỏ ứng dụng khỏi thiết bị thì hệ thống chỉ gỡ bỏ những file do ứng dụng tạo ra ở bộ nhớ ngoài khi trước đó ứng dụng lưu trữ dữ liệu vào đường dẫn trả về từ phương thức <code>getExternalFilesDir()</code> .
Phù hợp khi không muốn người dùng hoặc ứng dụng khác truy cập vào file dữ liệu của ứng dụng	Phù hợp khi ứng dụng cho phép file dữ liệu của mình được truy cập tự do từ người dùng, ứng dụng khác hay từ máy tính

Cấp quyền cho ứng dụng truy cập bộ nhớ thiết bị

- Trong file `AndroidManifest.xml` cần khai báo quyền để cho phép ứng dụng ghi dữ liệu vào bộ nhớ ngoài (`WRITE_EXTERNAL_STORAGE`):


```
<manifest ...>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>
```

- Hiện tại Android cho phép tất cả các ứng dụng có khả năng đọc dữ liệu từ bộ nhớ ngoài mà không cần cấp quyền cho ứng dụng. Tuy nhiên trong thời gian tới điều này có thể thay đổi. Có nghĩa là ứng dụng cần được cấp quyền để đọc dữ liệu từ bộ nhớ ngoài (READ_EXTERNAL_STORAGE)

```
<manifest ...>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    ...
</manifest>
```

- Tuy nhiên nếu ứng dụng được cấp quyền ghi thì mặc định được cấp quyền đọc dữ liệu từ bộ nhớ ngoài.

Chú ý: mặc định ứng dụng được phép truy cập bộ nhớ trong.

Đọc, ghi file dữ liệu từ bộ nhớ trong

Các chế độ cấp cho file :

- MODE_WORLD_READABLE : cho phép tất cả ứng dụng có thể đọc dữ liệu từ file.
- MODE_PRIVATE : cho phép file chỉ được truy cập bởi ứng dụng tạo ra nó.
- MODE_WORLD_WRITEABLE : cho phép tất cả ứng dụng có thể ghi dữ liệu vào file.

Ghi file dữ liệu vào bộ nhớ trong

- Gọi phương thức `openFileOutput()` mở file để ghi dữ liệu, tham số truyền vào của phương thức này gồm: tên file, chế độ cấp cho file. Giá trị trả về của phương thức này là đối tượng kiểu luồng ký tự `FileOuputStream`.
- Chuyển đổi đối tượng kiểu luồng ký tự `FileOuputStream` sang đối tượng kiểu luồng byte `OutputStreamWriter` bằng cách gọi phương thức khởi tạo của lớp `OutputStreamWriter` với tham số truyền vào là đối tượng kiểu luồng ký tự.
- Sử dụng phương thức `write ()` của đối tượng `OutputStreamWriter` để ghi dữ liệu vào bộ nhớ trong.

```

FileOutputStream fOut =
    openFileOutput("textfile.txt",
        MODE_WORLD_READABLE);
OutputStreamWriter osw = new
    OutputStreamWriter(fOut);

//---write the string to the file---
osw.write(str);
osw.flush();
osw.close();

```

Chú ý: đường dẫn tới file lưu ở bộ nhớ trong của ứng dụng được xác định bởi tên package của ứng dụng trong hệ thống tập tin Android, có thể thấy được thông qua Device File Explorer của Android Studio.

Đọc file dữ liệu từ bộ nhớ trong

- Gọi phương thức `openFileInput()` mở file để đọc dữ liệu, tham số truyền vào của phương thức này là tên file. Giá trị trả về của phương thức này là đối tượng kiểu luồng ký tự `FileInputStream`.
- Chuyển đổi đối tượng kiểu luồng ký tự `FileInputStream` sang đối tượng kiểu luồng byte `InputStreamReader` bằng cách gọi phương thức khởi tạo của lớp `InputStreamReader` với tham số truyền vào là đối tượng kiểu luồng ký tự.
- Lặp gọi phương thức `read()` của đối tượng `InputStreamReader` để đọc lần lượt khối 100 ký tự vào bộ đệm (buffer) cho tới khi không còn ký tự trong luồng. Tại mỗi lần phương thức `read()` được gọi, ký tự đọc được được gán cho biến kiểu `String`. Cộng dồn các giá trị của biến kiểu `String` này sẽ thu được nội dung file.

```

FileInputStream fIn =
    openFileInput("textfile.txt");
InputStreamReader isr = new
    InputStreamReader(fIn);

char[] inputBuffer = new char[READ_BLOCK_SIZE];
String s = "";

int charRead;
while ((charRead = isr.read(inputBuffer)) > 0)
{
    //---convert the chars to a String---
    String readString =
        String.valueOf(inputBuffer, 0,
            charRead);
    s += readString;

    inputBuffer = new char[READ_BLOCK_SIZE];
}

```

Đọc, ghi dữ liệu từ bộ nhớ ngoài

Không phải thiết bị nào cũng có sẵn bộ nhớ ngoài. Để kiểm tra xem thiết bị có bộ nhớ ngoài hay không ta gọi phương thức `getExternalStorageState()` của lớp `Environment`. Nếu giá trị trả về của phương thức này là `MEDIA_MOUNTED` thì có nghĩa thiết bị có bộ

nhớ ngoài cho phép bạn đọc/ ghi file.

```
/* Checks if external storage is available for read and write */
public boolean isExternalStorageWritable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}

/* Checks if external storage is available to at least read */
public boolean isExternalStorageReadable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        return true;
    }
    return false;
}
```

Các file lưu trữ trong bộ nhớ ngoài do ứng dụng tạo ra thuộc 1 trong 2 kiểu sau:

- Public files: là những file lưu trong bộ nhớ ngoài cho phép bất kỳ người dùng, ứng dụng trên thiết bị có thể truy cập. Khi người dùng gỡ bỏ ứng dụng thì các file này vẫn còn tồn tại (Ví dụ: các file download từ ứng dụng, các ảnh do ứng dụng chụp được).

Phương thức `getExternalStorageDirectory()` trả về đường dẫn của public file trong bộ nhớ ngoài.

- Private files: là những file lưu trong bộ nhớ ngoài chỉ được phép truy cập bởi ứng dụng tạo ra nó. Khi người dùng gỡ bỏ ứng dụng thì các file này cũng bị xóa (Ví dụ những file tạm do ứng dụng tạo ra).

Phương thức `getExternalFilesDir()` trả về đường dẫn của private file trong bộ nhớ ngoài.

Ghi file dữ liệu vào bộ nhớ ngoài

```
//---SD Card Storage---
File sdCard = Environment.getExternalStorageDirectory();
File directory = new File (sdCard.getAbsolutePath() +
    "/MyFiles");
directory.mkdirs();
File file = new File(directory, "textfile.txt");
FileOutputStream fOut = new FileOutputStream(file);
/*
FileOutputStream fOut = openFileOutput("textfile.txt",
MODE_WORLD_READABLE);
*/
OutputStreamWriter osw = new OutputStreamWriter(fOut);
//---write the string to the file---
osw.write(str);
osw.flush(); osw.close();
```

Đọc file dữ liệu từ bộ nhớ ngoài

```
//---SD Storage---
File sdCard = Environment.getExternalStorageDirectory();
```

```

File directory = new File (sdCard.getAbsolutePath() +
"/MyFiles");
File file = new File(directory, "textfile.txt");
FileInputStream fIn = new FileInputStream(file);
InputStreamReader isr = new InputStreamReader(fIn);
/*
FileInputStream fIn =
openFileInput("textfile.txt");
InputStreamReader isr = new InputStreamReader(fIn);
*/
char[] inputBuffer = new char[READ_BLOCK_SIZE];
String s = "";
int charRead;
while ((charRead = isr.read(inputBuffer))>0)
{
    //--convert the chars to a String--
    String readString =
String.valueOf(inputBuffer, 0, charRead);
    s += readString;
    inputBuffer = new char[READ_BLOCK_SIZE];
}

```

Xóa file: Xóa file lưu ở bộ nhớ trong ứng dụng bằng cách gọi hàm `deleteFile()` của đối tượng `Context` của ứng dụng

```
myContext.deleteFile(fileName);
```

Xóa file lưu ở bộ nhớ ngoài của thiết bị bằng cách gọi hàm `delete()` của đối tượng `File`:

```
myFile.delete();
```

2.4.3. Cơ sở dữ liệu quan hệ

Android cung cấp hệ quản trị cơ sở dữ liệu SQLite để quản lý cơ sở dữ liệu quan hệ của ứng dụng. Cơ sở dữ liệu quan hệ SQLite tạo ra cho của ứng dụng này thì chỉ được truy cập bởi chính nó mà không cho phép ứng dụng khác truy cập vào, cơ sở dữ liệu này được lưu trữ trong thư mục `/data/data/<package_name>/databases` của hệ thống (Sử dụng `Device File Explorer`, vào thư mục trên sẽ thấy file cơ sở dữ liệu của ứng dụng).

Định nghĩa lược đồ cơ sở dữ liệu

Vấn đề chính của cơ sở dữ liệu là định nghĩa đồ cơ sở dữ liệu. Lược đồ cơ sở dữ liệu là khai báo cách tổ chức cơ sở dữ liệu. Lược đồ cơ sở dữ liệu được khai báo thông qua lớp chứa định nghĩa các thành phần của cơ sở dữ liệu.

```

public final class FeedReaderContract {
    // To prevent someone from accidentally instantiating the contract class,
    // give it an empty constructor.
    public FeedReaderContract() {}

    /* Inner class that defines the table contents */
    public static abstract class FeedEntry implements BaseColumns {
        public static final String TABLE_NAME = "entry";
        public static final String COLUMN_NAME_ENTRY_ID = "entryid";
        public static final String COLUMN_NAME_TITLE = "title";
        public static final String COLUMN_NAME_SUBTITLE = "subtitle";
        ...
    }
}

```

Trong đó: định nghĩa các thành phần trong lược đồ cơ sở dữ liệu thông qua lớp

FeedReaderContract. Lớp FeedReaderContract định nghĩa tên bảng cùng với tên cột của các bảng trong cơ sở dữ liệu. Trong lớp FeedReaderContract tạo mỗi lớp con cho mỗi bảng tương ứng, trong mỗi lớp con sẽ khai báo tên các cột của bảng.

Tạo cơ sở dữ liệu sử dụng SQL Helper

Cơ sở dữ liệu ứng dụng Android tạo ra được lưu vào bộ nhớ trong do thiết bị cấp phát riêng cho ứng dụng và không được phép truy cập từ ứng dụng khác.

Tạo và duy trì cơ sở dữ liệu sử dụng các thành phần được khai báo trong lược đồ cơ sở dữ liệu ở trên, qua các bước sau:

- Khai báo các câu lệnh SQL để tạo và xóa các bảng trong cơ sở dữ liệu

```

private static final String TEXT_TYPE = " TEXT";
private static final String COMMA_SEP = ",";
private static final String SQL_CREATE_ENTRIES =
    "CREATE TABLE " + FeedEntry.TABLE_NAME + " (" +
    FeedEntry._ID + " INTEGER PRIMARY KEY," +
    FeedEntry.COLUMN_NAME_ENTRY_ID + TEXT_TYPE + COMMA_SEP +
    FeedEntry.COLUMN_NAME_TITLE + TEXT_TYPE + COMMA_SEP +
    ... // Any other options for the CREATE command
    " )";

private static final String SQL_DELETE_ENTRIES =
    "DROP TABLE IF EXISTS " + FeedEntry.TABLE_NAME;

```

- Tạo và duy trì cơ sở dữ liệu sử dụng tập API của lớp SQLiteOpenHelper bằng cách: tạo ra 1 lớp java kế thừa từ lớp SQLiteOpenHelper, override các phương thức onCreate(), onUpgrade(), onOpen() do SQLiteOpenHelper cung cấp.

```

public class FeedReaderDbHelper extends SQLiteOpenHelper {
    // If you change the database schema, you must increment the database version.
    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "FeedReader.db";

    public FeedReaderDbHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    public void onCreate(SQLiteDatabase db) {
        db.execSQL(SQL_CREATE_ENTRIES);
    }

    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // This database is only a cache for online data, so its upgrade policy is
        // to simply to discard the data and start over
        db.execSQL(SQL_DELETE_ENTRIES);
        onCreate(db);
    }

    public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        onUpgrade(db, oldVersion, newVersion);
    }
}

```

Tạo đối tượng FeedReaderDBHelper để truy xuất tới cơ sở dữ liệu đã tạo:

```

FeedReaderDbHelper mDbHelper = new FeedReaderDbHelper(getApplicationContext());

```

Ghi dữ liệu vào trong cơ sở dữ liệu

- Sử dụng phương thức insert() của đối tượng SQLiteDatabase để ghi dữ liệu vào cơ sở dữ liệu.

```

// Gets the data repository in write mode
SQLiteDatabase db = mDbHelper.getWritableDatabase();

// Create a new map of values, where column names are the keys
ContentValues values = new ContentValues();
values.put(FeedEntry.COLUMN_NAME_ENTRY_ID, id);
values.put(FeedEntry.COLUMN_NAME_TITLE, title);
values.put(FeedEntry.COLUMN_NAME_CONTENT, content);

// Insert the new row, returning the primary key value of the new row
long newRowId;
newRowId = db.insert(
    FeedEntry.TABLE_NAME,
    FeedEntry.COLUMN_NAME_NULLABLE,
    values);

```

Phương thức insert() có 3 tham số :

- Tham số thứ nhất c tên bảng.
- Tham số thứ 2 là tên của của cột trong bảng mà giá trị ứng với cột này có thể nhận giá trị NULL trong trường hợp đối tượng ContentValues trống. Nếu truyền giá trị cho tham số thứ 2 này là “null” thì dòng dữ liệu mới không được thêm vào.
- Tham số thứ 3 là đối tượng ContentValues tương ứng với 1 dòng dữ liệu trong bảng.

Đọc dữ liệu từ cơ sở dữ liệu

Sử dụng phương thức query() của đối tượng SQLiteDatabase để đọc dữ liệu từ cơ sở dữ liệu. Phương thức này có tham số truyền vào là tên bảng, mảng các tên cột mà bạn cần lấy dữ liệu, điều kiện. Giá trị trả về của phương thức này là đối tượng Cursor.

```
SQLiteDatabase db = mDbHelper.getReadableDatabase();

// Define a projection that specifies which columns from the database
// you will actually use after this query.
String[] projection = {
    FeedEntry._ID,
    FeedEntry.COLUMN_NAME_TITLE,
    FeedEntry.COLUMN_NAME_UPDATED,
    ...
};

// How you want the results sorted in the resulting Cursor
String sortOrder =
    FeedEntry.COLUMN_NAME_UPDATED + " DESC";

Cursor c = db.query(
    FeedEntry.TABLE_NAME, // The table to query
    projection,            // The columns to return
    selection,             // The columns for the WHERE clause
    selectionArgs,         // The values for the WHERE clause
    null,                  // don't group the rows
    null,                  // don't filter by row groups
    sortOrder              // The sort order
);
```

Đối tượng Cursor do phương thức query() trả về chứa các dữ liệu được yêu cầu. Sử dụng phương thức moveToFirst() của đối tượng Cursor để đặt vị trí đọc dữ liệu ở phần tử đầu tiên trong kết quả. Tại mỗi dòng, đọc giá trị từng phần tử của dòng tương ứng với các cột trong bảng dữ liệu, bằng cách gọi phương thức get của Cursor, ví dụ: getString() để lấy về giá trị của phần tử có kiểu String, getLong() để lấy về giá trị của phần tử có kiểu Long.... Với mỗi lần gọi phương thức get cần truyền vào tham số là chỉ số cột trong bảng.

Chỉ số cột của bảng có được bằng cách gọi phương thức getColumnIndex() hoặc getColumnIndexOrThrow().

```
cursor.moveToFirst();
long itemId = cursor.getLong(
    cursor.getColumnIndexOrThrow(FeedEntry._ID)
);
```

Xóa dữ liệu trong cơ sở dữ liệu

Sử dụng phương thức delete() của đối tượng SQLiteDatabase để xóa các dòng dữ liệu từ một bảng trong cơ sở dữ liệu. Phương thức delete() có 3 tham số :

- Tham số thứ nhất là tên bảng.
- Tham số thứ 2 là điều kiện xóa.
- Tham số thứ 3 là chỉ số dòng cần xóa (rowId).


```
// Define 'where' part of query.
String selection = FeedEntry.COLUMN_NAME_ENTRY_ID + " LIKE ?";
// Specify arguments in placeholder order.
String[] selectionArgs = { String.valueOf(rowId) };
// Issue SQL statement.
db.delete(table_name, selection, selectionArgs);
```

Cập nhật dữ liệu trong cơ sở dữ liệu

Sử dụng phương thức update() của đối tượng SQLiteDatabase để cập nhật dữ liệu đã có trong cơ sở dữ liệu. Phương thức update() có 3 tham số :

- Tham số thứ nhất là tên bảng.
- Tham số thứ 2 là giá trị mới muốn cập nhật.
- Tham số thứ 3 là điều kiện cập nhật.
- Tham số thứ 4 là chỉ số dòng cần cập nhật (rowId).

```
SQLiteDatabase db = mDbHelper.getReadableDatabase();

// New value for one column
ContentValues values = new ContentValues();
values.put(FeedEntry.COLUMN_NAME_TITLE, title);

// Which row to update, based on the ID
String selection = FeedEntry.COLUMN_NAME_ENTRY_ID + " LIKE ?";
String[] selectionArgs = { String.valueOf(rowId) };

int count = db.update(
    FeedReaderDbHelper.FeedEntry.TABLE_NAME,
    values,
    selection,
    selectionArgs);
```

CÂU HỎI CHƯƠNG 2

1. Xây dựng ứng dụng theo dõi các trạng thái trong vòng đời ứng dụng Android và đưa ra thông báo tương ứng khi mỗi trạng thái được gọi.
2. Sử dụng các đối tượng giao diện của Android để xây dựng ứng dụng máy tính.
3. Xây dựng ứng dụng To-do list quản lý các công việc của người dùng điện thoại theo 3 cách: 1/ Lưu trữ dữ liệu dạng key-value; 2/ Lưu trữ dữ liệu dưới dạng file trong bộ nhớ thiết bị; 3/ Lưu trữ dữ liệu trong cơ sở dữ liệu quan hệ SQLite.
4. Xây dựng ứng dụng thư viện hiển thị ảnh sử dụng các View như: ImageView, Gallery, ImageSwicher, GridView.

CHƯƠNG 3: LẬP TRÌNH ỨNG DỤNG ANDROID NÂNG CAO

3.1. MESSAGING VÀ NETWORKING

3.1.1. SMS Messaging

SMS (Short Messaging Service) là công nghệ gửi tin nhắn ngắn giữa các điện thoại di động. Tin nhắn ngắn tồn tại dưới 2 dạng: dạng văn bản người dùng đọc được và dạng dữ liệu (như dạng nhị phân) dùng để truyền tín hiệu trao đổi giữa các ứng dụng. Android cung cấp 1 tập API SMS hỗ trợ lập trình chức năng nhắn tin ngắn.

Có 2 cách cho phép gửi tin nhắn từ ứng dụng Android: 1/ Bằng cách sử dụng Intent, ứng dụng xây dựng sẽ gọi ứng dụng nhắn tin có sẵn trên điện thoại ; 2/ Bằng cách sử dụng lớp SmsManager do Android cung cấp để quản lý việc gửi nhận tin nhắn ngay trong chính ứng dụng xây dựng.

Gửi tin nhắn SMS thông qua ứng dụng nhắn tin có sẵn trên điện thoại

Sử dụng phương thức startActivity() với tham số truyền vào là đối tượng Intent có hành vi Intent.ACTION_SENDTO để gọi ứng dụng nhắn tin có sẵn trên điện thoại từ ứng dụng xây dựng như sau:

```
Intent smsIntent = new Intent(Intent.ACTION_SENDTO,
    Uri.parse("sms:55512345"));
smsIntent.putExtra("sms_body", "Press send to send me");
startActivity(smsIntent);
```

Trong đó: dữ liệu Intent đặc tả số điện thoại nhận tin dưới dạng sms:schema và nội dung tin nhắn gửi đi “Press send to send me” được gán cho sms_body.

Gửi tin nhắn trong chính ứng dụng

- Cấp quyền SEND_SMS cho phép gửi tin nhắn ngay trong ứng dụng, trong AndroidManifest.xml như sau:

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

- Tạo đối tượng SmsManager để xử lý việc gửi nhận tin nhắn:

```
SmsManager smsManager = SmsManager.getDefault();
```

- Cài đặt lệnh gửi tin nhắn SMS trong chính ứng dụng:

Gửi tin nhắn văn bản:

Phương thức sendTextMessages() của SmsManager cho phép gửi tin nhắn SMS, phương thức này gồm 5 tham số:

- destinationAddress: Số điện thoại nhận tin.
- scAddress: Địa chỉ trung tâm dịch vụ tin nhắn(SMSC), giá trị này bằng null cho biết ứng dụng sử dụng dịch vụ mặc định của SMSC.
- text: Nội dung tin nhắn.
- sentIntent: tham số Pending intent thứ nhất cho biết trạng thái gửi tin từ

bên gửi.

- `deliveryIntent`: tham số Pending intent thứ 2 cho biết trạng thái nhận tin nhắn bởi bên nhận.

```
String sendTo = "5551234";  
String myMessage = "Android supports programmatic SMS messaging!";  
smsManager.sendTextMessage(sendTo, null, myMessage, null, null);
```

Gửi tin nhắn dữ liệu dạng nhị phân

Sử dụng `sendDataMessage()` của `SmsManager` cho phép gửi dữ liệu dạng nhị phân. Phương thức này có tham số truyền vào tương tự như phương thức `sendTextMessage()` nêu trên, đồng thời bổ sung 2 tham số: cổng lắng nghe kết nối dữ liệu (destination port) và 1 mảng byte chứa các dữ liệu nhị phân.

Giám sát trạng thái gửi tin nhắn từ bên gửi tới bên nhận

Để giám sát trạng thái tin nhắn gửi đi cần đăng ký Broadcast Receivers lắng nghe các sự kiện với giá trị đặc tả được truyền vào tham số Pending Intents của phương thức `sendTextMessage()`.

Tham số Pending Intent đầu tiên cho biết trạng thái tin nhắn gửi đi từ bên gửi là thành công hay thất bại thông qua 1 trong các mã đặc tả sau:

- `Activity.RESULT_OK`: cho biết tin nhắn gửi thành công.
- `SmsManager.RESULT_ERROR_GENERIC_FAILURE`: cho biết quá trình truyền tin nhắn lỗi.
- `SmsManager.RESULT_ERROR_RADIO_OFF`: cho biết tin nhắn không truyền đi do không có kết nối mạng.
- `SmsManager.RESULT_ERROR_NULL_PDU`: lỗi PDU (protocol description unit).
- `SmsManager.RESULT_ERROR_NO_SERVICE`: cho biết dịch vụ mạng hiện thời không sẵn sàng.

Tham số Pending Intent thứ 2 cho biết trạng thái tin nhắn gửi đến bên nhận là thành công hay thất bại thông qua 1 trong 2 đặc tả sau:

- `Activity.RESULT_OK`: cho biết bên nhận đã nhận được tin nhắn.
- `Activity.RESULT_CANCELED`: cho biết bên nhận không nhận được tin nhắn.

Việc cài đặt lệnh giám sát trạng thái gửi tin nhắn từ bên gửi tới bên nhận được thực hiện như sau:

```
String SENT_SMS_ACTION = "com.paad.smssnippets.SENT_SMS_ACTION";  
String DELIVERED_SMS_ACTION = "com.paad.smssnippets.DELIVERED_SMS_ACTION";  
// Create the sentIntent parameter  
Intent sentIntent = new Intent(SENT_SMS_ACTION);  
PendingIntent sentPI = PendingIntent.getBroadcast(getApplicationContext(),  
0,  
sentIntent,  
PendingIntent.FLAG_UPDATE_CURRENT)  
;
```

```

// Create the deliveryIntent parameter
Intent deliveryIntent = new Intent(DELIVERED_SMS_ACTION);
PendingIntent deliverPI =
PendingIntent.getBroadcast(getApplicationContext(),
0,
deliveryIntent,
PendingIntent.FLAG_UPDATE_CURRENT);
//--- Register the BroadcastReceiver when the SMS is sent---
registerReceiver(new BroadcastReceiver() {
@Override
public void onReceive(Context _context, Intent _intent)
{
String resultText = "UNKNOWN";
switch (getResultCode()) {
case Activity.RESULT_OK:
resultText = "Transmission successful"; break;
case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
resultText = "Transmission failed"; break;
case SmsManager.RESULT_ERROR_RADIO_OFF:
resultText = "Transmission failed: Radio is off";
break;
case SmsManager.RESULT_ERROR_NULL_PDU:
resultText = "Transmission Failed: No PDU specified";
break;
case SmsManager.RESULT_ERROR_NO_SERVICE:
resultText = "Transmission Failed: No service";
break;
}
Toast.makeText(_context, resultText,
Toast.LENGTH_LONG).show();
}
},
new IntentFilter(SENT_SMS_ACTION));
//---Register the BroadcastReceiver when the SMS is delivered---
registerReceiver(new BroadcastReceiver() {
@Override
public void onReceive(Context _context, Intent _intent)
{
switch (getResultCode()){
case Activity.RESULT_OK:
resultText = "SMS delivered";
break;
case Activity.RESULT_CANCELED:
resultText = "SMS not delivered ";
break;
}
Toast.makeText(_context, resultText,
Toast.LENGTH_LONG).show();
}
},
new IntentFilter(DELIVERED_SMS_ACTION));

```

```
// Send the message
SmsManager smsManager = SmsManager.getDefault();
String sendTo = "5551234";
String myMessage = "Android supports programmatic SMS messaging!";
smsManager.sendTextMessage(sendTo, null, myMessage, sentPI, deliverPI);
```

Phân đoạn tin nhắn SMS gửi đi

Chiều dài tối đa của tin nhắn SMS gửi đi được qui định là 160 kí tự. Nếu bên gửi gửi đi tin nhắn có độ dài lớn hơn 160 kí tự thì tin nhắn sẽ được cắt ra thành các tin nhắn có độ dài ngắn hơn, với mỗi tin nhắn có độ dài dưới 160 kí tự. Để làm được điều này, Android cung cấp phương thức một số API để làm điều này:

- `divideMessage()`: đầu vào là đoạn văn bản muốn gửi đi, đầu ra là mảng `ArrayList` chứa các tin nhắn có chiều dài dưới 160 kí tự.
- `sendMultipartTextMessage()`: cho phép truyền 1 mảng tin nhắn đã phân đoạn trên.

```
ArrayList<String> messageArray = smsManager.divideMessage(myMessage);
ArrayList<PendingIntent> sentIntents = new ArrayList<PendingIntent>();
for (int i = 0; i < messageArray.size(); i++)
    sentIntents.add(sentPI);
smsManager.sendMultipartTextMessage(sendTo, null, messageArray,
    sentIntents, null);
```

Nhận tin nhắn trong chính ứng dụng

Để nhận tin nhắn SMS trong chính ứng dụng cần tạo ra 1 lớp Java kế thừa từ lớp `BroadcastReceiver` cho phép ứng dụng nhận các Intent gửi từ ứng dụng khác thông qua `sendBroadcast()`. Việc cài đặt như sau:

Cấp quyền cho phép nhận tin nhắn trong chính ứng dụng, trong `AndroidManifest.xml` như sau:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.interceptsmsmessages"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />
    <uses-permission
        android:name="android.permission.RECEIVE_SMS" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
                />
            <category
```

```

        android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<receiver android:name=".SMSReceiver" >
    <intent-filter android:priority="100" >
        <action android:name=
            "android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
</receiver>
</application>
</manifest>

```

Trong đó : thuộc tính android:priority được thiết lập có giá trị càng cao thì ứng dụng càng có khả năng cao nhận được tin nhắn.

Cài đặt lệnh nhận tin nhắn SMS trong chính ứng dụng:

```

package net.learn2develop.interceptsmsmessages;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.util.Log;
import android.widget.Toast;
public class SMSReceiver extends BroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent intent)
    {
        //---get the SMS message passed in---
        Bundle bundle = intent.getExtras();
        SmsMessage[] msgs = null; String str = "SMS from ";
        if (bundle != null)
        {
            //---retrieve the SMS message received---
            Object[] pdus = (Object[]) bundle.get("pdus");
            msgs = new SmsMessage[pdus.length];
            for (int i=0; i<msgs.length; i++){
                msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);
                if (i==0) {
                    //---get the sender address/phone number---
                    str += msgs[i].getOriginatingAddress();
                    str += ": ";
                }
                //---get the message body---
                str += msgs[i].getMessageBody().toString();
            }
            //---display the new SMS message---
            Toast.makeText(context, str, Toast.LENGTH_SHORT).show();
            Log.d("SMSReceiver", str);
        }
    }
}

```

}

Khi ứng dụng nhận được tin nhắn gửi tới, `onReceive()` được thực thi. Tin nhắn SMS được chứa trong đối tượng `Intent` (tham số thứ 2 của `onReceive()`) thông qua đối tượng `Bundle`.

Mỗi tin nhắn SMS được chứa trong 1 mảng `Object` theo định dạng PDU. Nếu tin nhắn SMS có độ dài nhỏ hơn 160 kí tự thì mảng đó chứa 1 phần tử, ngược lại nếu tin nhắn có độ dài lớn hơn 160 kí tự thì mảng sẽ chứa các phần tử, mỗi phần tử là 1 đoạn trong nội dung tin nhắn có độ dài nhỏ hơn 160 kí tự.

Đọc ra mỗi tin nhắn sử dụng phương thức tĩnh `createFromPdu()` của lớp `SmsMessage`. Số điện thoại bên gửi thu được thông qua phương thức `getOriginatingAddress()`, nội dung mỗi tin nhắn thu được thông qua phương thức `getMessageBody()`.

3.1.2. Sending Email

Giống như nhắn tin SMS, Android cũng hỗ trợ e-mail. Ứng dụng Gmail / Email trên Android cho phép xác định cấu hình tài khoản email bằng POP3 hoặc IMAP. Bên cạnh việc gửi và nhận e-mail sử dụng ứng dụng Gmail / Email, bạn cũng có thể gửi tin nhắn e-mail theo chương trình từ chính trong ứng dụng Android.

Các bước thực hiện:

1. Tạo dự án có tên là Email
2. Thêm các dòng sau vào file `mail.xml`

```
?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">
    <Button
        android:id="@+id/btnSendEmail"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Send Email"
        android:onClick="onClick" />
</LinearLayout>
```

3. Thêm các đoạn sau vào file `Activity.java`

```
package net.learn2develop.Emails;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
public class EmailsActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    public void onClick(View v) {
```



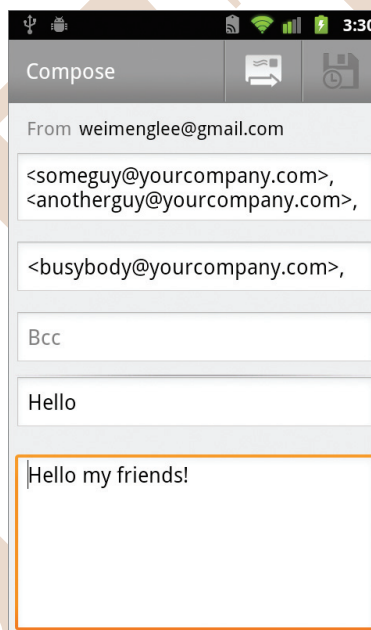
```

//---replace the following email addresses with real ones---
String[] to =

{"someguy@example.com",
"anotherguy@example.com"};
String[] cc = {"busybody@example.com"};
sendEmail(to, cc, "Hello", "Hello my friends!");
}
//---sends an SMS message to another device---
private void sendEmail(String[] emailAddresses, String[] carbonCopies,
String subject, String message)
{
Intent emailIntent = new Intent(Intent.ACTION_SEND);
emailIntent.setData(Uri.parse("mailto:"));
String[] to = emailAddresses;
String[] cc = carbonCopies;
emailIntent.putExtra(Intent.EXTRA_EMAIL, to);
emailIntent.putExtra(Intent.EXTRA_CC, cc);
emailIntent.putExtra(Intent.EXTRA_SUBJECT, subject);
emailIntent.putExtra(Intent.EXTRA_TEXT, message);
emailIntent.setType("message/rfc822");
startActivity(Intent.createChooser(emailIntent, "Email"));
}
}

```

Build ứng dụng và nhận kết quả



Hình 3.1. Kết quả ứng dụng gửi Email

Trong ví dụ này, bạn đang khởi chạy ứng dụng Email tích hợp để gửi thông điệp email. Để làm thế, bạn sử dụng một đối tượng Intent và đặt các tham số khác nhau bằng cách sử dụng setData (), putExtra () và các phương thức setType ():

```

Intent emailIntent = new Intent(Intent.ACTION_SEND);
emailIntent.setData(Uri.parse("mailto:"));
String[] to = emailAddresses;
String[] cc = carbonCopies;
emailIntent.putExtra(Intent.EXTRA_EMAIL, to);
emailIntent.putExtra(Intent.EXTRA_CC, cc);
emailIntent.putExtra(Intent.EXTRA_SUBJECT, subject);
emailIntent.putExtra(Intent.EXTRA_TEXT, message);
emailIntent.setType("message/rfc822");
startActivity(Intent.createChooser(emailIntent, "Email"));

```

3.1.3. Wifi

Android cung cấp lớp WifiManager cho phép cấu hình kết nối wifi, giám sát và chỉnh sửa cài đặt cho kết nối wifi đã có, quản lý kết nối wifi hiện thời, cũng như quét xem có những điểm truy cập wifi nào xung quanh thiết bị.

❖ **Để ứng dụng có thể làm việc với wifi, đầu tiên cần cấu hình cấp quyền cho ứng dụng trong AndroidManifest.xml**

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
```

❖ **Tạo ra đối tượng WifiManager cho phép ứng dụng quản lý wifi:** sử dụng phương thức getSystemService()

```
String service = Context.WIFI_SERVICE;
WifiManager wifi = (WifiManager) getSystemService(service);
```

❖ **Kích hoạt hoặc hủy bỏ kết nối wifi:** sử dụng phương thức setWifiEnabled() của đối tượng WifiManager

```
if (!wifi.isWifiEnabled())
if (wifi.getWifiState() != WifiManager.WIFI_STATE_ENABLING)
wifi.setWifiEnabled(true);
```

❖ **Lấy về thông tin kết nối wifi hiện thời**

```
WifiInfo info = wifi.getConnectionInfo();
if (info.getBSSID() != null) {
int strength = WifiManager.calculateSignalLevel(info.getRssi(), 5);
int speed = info.getLinkSpeed();
String units = WifiInfo.LINK_SPEED_UNITS;
String ssid = info.getSSID();
String cSummary = String.format("Connected to %s at %s%. Strength %s/5",
ssid, speed, units, strength);
Log.d(TAG, cSummary);
}
```

Trong đó: getConnectionInfo() lấy về thông tin trạng thái kết nối, có giá trị trả về là đối tượng WifiInfo cho biết các thông tin : SSID, BSSID, địa chỉ MAC, địa chỉ IP của điểm truy cập wifi hiện thời, cũng như tốc độ kết nối và cường độ tín hiệu.

Quét những điểm truy cập wifi xung quanh thiết bị

```
registerReceiver(new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
List<ScanResult> results = wifi.getScanResults();
ScanResult bestSignal = null;
```

```

for (ScanResult result : results) {
    if (bestSignal == null ||
        WifiManager.compareSignalLevel(
            bestSignal.level, result.level) < 0)
        bestSignal = result;
}
String connSummary = String.format("%s networks found. %s is
the strongest.", results.size(), bestSignal.SSID);
Toast.makeText(MyActivity.this, connSummary, Toast.LENGTH_LONG).show();
}
}, new IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));
// Initiate a scan.
wifi.startScan();

```

Trong đó:

- startScan() của đối tượng WifiManager được gọi cho phép quét những điểm truy cập wifi xung quanh thiết bị.
- Khi phương thức này thực thi xong, đối tượng IntentFilter với tham số truyền vào là SCAN_RESULTS_AVAILABLE_ACTION cho phép quảng bá thông báo kết quả quét, lúc này phương thức callback onReceive() được gọi để lấy về thông tin những điểm truy cập wifi xung quanh thiết bị.
- getScanResults() của đối tượng WifiManager được gọi cho phép lấy về danh sách các đối tượng ScanResult. Mỗi đối tượng ScanResult sẽ chứa các thông tin chi tiết về điểm truy cập wifi mà thiết bị quét được, gồm: tốc độ truyền tải, độ mạnh tín hiệu, SSID và các kỹ thuật xác thực được hỗ trợ.

Quản lý thông tin cấu hình wifi

Lấy về thông tin cấu hình của mạng wifi mà thiết bị kết nối tới qua đối tượng WifiManager.

```

// Get a list of available configurations
List<WifiConfiguration> configurations = wifi.getConfiguredNetworks();
// Get the network ID for the first one.
if (configurations.size() > 0) {
    int netID = configurations.get(0).networkId; // Enable that network.
    boolean disableAllOthers = true;
    wifi.enableNetwork(netID, disableAllOthers);
}

```

Trong đó: getConfiguredNetworks() sử dụng để lấy về thông tin cấu hình của mạng wifi mà thiết bị kết nối tới. Phương thức này có giá trị trả về là danh sách các đối tượng WifiConfiguration chứa các thông tin về network ID, SSID và một số thông tin cấu hình chi tiết khác.

Thiết lập cấu hình cho mạng wifi mà thiết bị kết nối tới

Để ứng dụng kết nối được với mạng wifi thì cần phải thiết lập và đăng ký cấu hình wifi.

Mặc định quá trình thiết lập và đăng ký cấu hình sẽ sử dụng những thông tin cấu hình wifi mặc định trên thiết bị. Tuy nhiên, ứng dụng có thể thiết lập cấu hình riêng cho mạng wifi mà ứng dụng muốn kết nối tới.

Thông tin cấu hình mạng được lưu trữ trong đối tượng WifiConfiguration gồm:

- BSSID của điểm truy cập wifi.
- SSID của một mạng cụ thể.
- networkId : định danh mạng.
- priority: mức độ ưu tiên của cấu hình mạng.
- status: trạng thái hiện thời của mạng được kết nối tới nhận 1 trong số trạng thái sau:

WifiConfiguration.Status.ENABLED, WifiConfiguration.Status.DISABLED, WifiConfiguration.Status.CURRENT.

WifiConfiguration cung cấp một số phương thức:

- addNetworkmethod (),updateNetwork() để thiết lập mới cấu hình, cập nhật cấu hình mạng wifi. Tham số truyền vào cho các phương thức này là networkID và các giá trị muốn thiết lập hoặc cập nhật.
- removeNetwork() dùng để gỡ bỏ cấu hình wifi, tham số truyền vào là networkID.
- saveConfiguration() để các thiết lập, cấu hình, gỡ bỏ mạng wifi trên có hiệu lực trên thiết bị.

3.1.4. Bluetooth

Bluetooth là giao thức hỗ trợ cho thiết bị ngang hàng (peer to peer) ở khoảng cách gần nhau có thể kết nối không dây được với nhau với băng thông thấp.

Android cung cấp một tập các API bluetooth trong gói android.bluetooth cho phép xây dựng ứng dụng có khả năng kiểm tra, bật tắt, giám sát, tìm kiếm, truyền dữ liệu tới các thiết bị khác có hỗ trợ bluetooth trong phạm vi cho phép.

Gói android.bluetooth gồm các lớp và giao diện sau:

Bảng 3.1. Các lớp và giao diện trong gói Android.bluetooth

Lớp/ giao diện	Miêu tả
BluetoothAdapter	Cho phép thăm dò, truy vấn, lắng nghe kết nối bluetooth.
BluetoothDevice	Cho phép gửi yêu cầu, truy vấn thông tin của các thiết bị được kết nối tới thông qua bluetooth.
BluetoothSocket	Để 2 thiết bị Android có thể kết nối với nhau thông qua bluetooth thì 1 thiết bị phải đóng vai trò là thiết bị nguồn gửi yêu cầu kết nối và 1 thiết bị đóng vai trò là thiết bị đích lắng nghe yêu cầu kết nối gửi tới thông qua bluetooth. Lớp này cho phép mở socket yêu cầu kết nối tới thiết bị đích và nhận hồi đáp trả về. Sau khi quá trình kết nối giữa thiết bị nguồn và đích được thiết lập, dữ liệu được truyền qua 2 thiết bị thông qua InputStream, OutputStream.

BluetoothServerSocket	Lớp này cho phép mở socket lắng nghe kết nối và hồi đáp lại thiết bị nguồn thông qua kết nối bluetooth.
BluetoothClass	Cho biết thông tin đặc tả và khả năng của thiết bị hỗ trợ bluetooth.
BluetoothProfile	Cho biết thông tin đặc tả giao tiếp bluetooth giữa 2 thiết bị.
BluetoothHeadset	Cho phép bluetooth headset sử dụng được với điện thoại di động.
BluetoothA2dp	Định nghĩa chất lượng audio giữa 2 thiết bị kết nối với nhau thông qua bluetooth.
BluetoothHealth	Cho biết thông tin tình trạng của cổng điều khiển dịch vụ kết nối bluetooth.
BluetoothHealthCallback	Là lớp trừu tượng thực thi các phương thức callback từ BluetoothHealth. Ứng dụng khai báo 1 lớp kế thừa từ lớp này sẽ cài đặt các phương thức callback để nhận thông tin cập nhật về những thay đổi trạng thái đăng ký và trạng thái kênh kết nối bluetooth trong ứng dụng.
BluetoothHealthAppConfiguration	Cho biết cấu hình ứng dụng được phát triển bởi bên thứ 3 đăng ký giao tiếp với thiết bị khác thông qua kết nối bluetooth.
BluetoothProfile.ServiceListener	Dùng để thông báo khi thiết bị thiết lập hay ngắt kết nối với các dịch vụ thông qua kết nối bluetooth.

Cấp quyền để cho phép ứng dụng làm việc với bluetooth

Trong AndroidManifest.xml cần cấp quyền BLUETOOTH và BLUETOOTH_ADMIN cho ứng dụng:

```
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
```

Kiểm tra thiết bị có hỗ trợ bluetooth hay không

Dùng phương thức getDefaultAdapter() của đối tượng BluetoothAdapter để kiểm tra thiết bị có hỗ trợ bluetooth hay không. Giá trị trả về của phương thức này là null cho biết thiết bị không hỗ trợ bluetooth. Đoạn code dưới đây phục vụ cho mục đích này:

```
package net.learn2develop.bluetooth;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.os.Bundle;
import android.widget.Toast;
public class MainActivity extends Activity {
    BluetoothAdapter bluetoothAdapter;
    //---check if bluetooth is available on the device---
    private boolean BluetoothAvailable()
    {
        if (bluetoothAdapter == null)
            return false;
        else
            return true;
    }
}
```

```

    }
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        Toast.makeText(this, "Bluetooth available: " +
            BluetoothAvailable(),
            Toast.LENGTH_LONG).show();
    }
}

```

Bật tắt bluetooth trên thiết bị

Sau khi đã kiểm tra được thiết bị có hỗ trợ bluetooth. Để bật tắt bluetooth trên thiết bị sử dụng phương thức isEnabled() của đối tượng BluetoothAdapter để kiểm tra thiết bị có đang bật bluetooth hay không.

- Nếu giá trị trả về của phương thức này là true thì có nghĩa bluetooth đã được bật trên thiết bị.
- Nếu giá trị trả về là false thì có nghĩa thiết bị đang tắt bluetooth. Khi đó sử dụng đối tượng Intent để bật tính năng bluetooth.

Đoạn code dưới đây phục vụ cho mục đích này:

```

package net.learn2develop.bluetooth;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.content.Intent;
import android.os.Bundle;
import android.widget.Toast;
public class MainActivity extends Activity {
    BluetoothAdapter bluetoothAdapter;
    static final int REQUEST_ENABLE_BT = 0;
    //---check if bluetooth is available on the device---
    private boolean BluetoothAvailable()
    {
        if (bluetoothAdapter == null)
            return false;
        else
            return true;
    }
    //---enable bluetooth on the device---
    private void EnableBluetooth() {
        if (BluetoothAvailable() && !bluetoothAdapter.isEnabled()) {
            Intent i = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(i, REQUEST_ENABLE_BT);
        }
    }
    public void onActivityResult(int requestCode,
        int resultCode, Intent data) {
        if (requestCode == REQUEST_ENABLE_BT) {

```

```

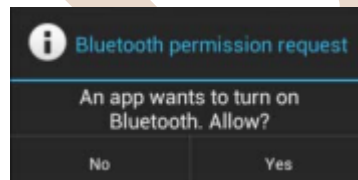
if (resultCode == RESULT_OK)
{
    Toast.makeText(this, "Bluetooth turned on!",
    Toast.LENGTH_SHORT).show();
}
}
}

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
    Toast.makeText(this, "Bluetooth available: " +
    BluetoothAvailable(),
    Toast.LENGTH_LONG).show();
    if (BluetoothAvailable())
        EnableBluetooth();
}
}

```

Khi ứng dụng chạy, phương thức `startActivityForResult()` được gọi yêu cầu hệ điều hành sẽ bật hộp thoại yêu cầu người dùng xác thực việc bật bluetooth trên thiết bị.

Phương thức này có hai tham số truyền vào: tham số đầu tiên là đối tượng `Intent`, tham số thứ hai là hằng số `REQUEST_ENABLE_BT` hoặc `REQUEST_DISABLE_BT` để hiển thị thông báo tương ứng tới người dùng bật tắt bluetooth.



Hình 3.2. Giao diện ứng dụng bluetooth

Tắt bluetooth trên thiết bị:

Sử dụng hàm `disable()` của đối tượng `BluetoothAdapter` để tắt bluetooth trên thiết bị.

Giám sát trạng thái bluetooth của thiết bị

Các trạng thái bluetooth của thiết bị gồm có: `STATE_TURNING_ON`; `STATE_ON`; `STATE_TURNING_OFF`; `STATE_OFF`

Sử dụng lớp `BroadcastReceiver` kết hợp với đối tượng `IntentFilter` để giám sát trạng thái bluetooth của thiết bị. Đoạn code dưới đây phục vụ cho mục đích này:

```

package net.learn2develop.bluetooth;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

```



```

import android.content.IntentFilter;
import android.os.Bundle;
import android.widget.Toast;
public class MainActivity extends Activity {
    BluetoothAdapter bluetoothAdapter;
    static final int REQUEST_ENABLE_BT = 0;
    //---check if bluetooth is available on the device---
    private boolean BluetoothAvailable()
    {
        if (bluetoothAdapter == null)
            return false;
        else
            return true;
    }
    //---enable bluetooth on the device---
    private void EnableBluetooth() {
        if (BluetoothAvailable() && !bluetoothAdapter.isEnabled()) {
            Intent i = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(i,
                REQUEST_ENABLE_BT);
        }
    }
    public void onActivityResult(int requestCode,
        int resultCode, Intent data) {
        if (requestCode == REQUEST_ENABLE_BT) {
            if (resultCode == RESULT_OK)
            {
                Toast.makeText(this, "Bluetooth turned on!",
                    Toast.LENGTH_SHORT).show();
            }
        }
    }
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        Toast.makeText(this, "Bluetooth available: " +
            BluetoothAvailable(),
            Toast.LENGTH_LONG).show();
        if (BluetoothAvailable())
            EnableBluetooth();
        MyBTBroadcastReceiver mReceiver = new MyBTBroadcastReceiver();
        IntentFilter intentFilter = new
            IntentFilter("android.bluetooth.adapter.action.STATE_CHANGED");
        registerReceiver(mReceiver, intentFilter);
    }
    public class MyBTBroadcastReceiver extends BroadcastReceiver {
        @Override
        public void onReceive(Context context, Intent intent) {

```

```

int state =
intent.getExtras().getInt(BluetoothAdapter.EXTRA_STATE);
switch (state) {
case BluetoothAdapter.STATE_OFF:
Toast.makeText(context, "Off", Toast.LENGTH_SHORT).show();
break;
case BluetoothAdapter.STATE_TURNING_OFF:
Toast.makeText(context, "Turning Off",
Toast.LENGTH_SHORT).show();
break;
case BluetoothAdapter.STATE_ON:
Toast.makeText(context, "On", Toast.LENGTH_SHORT).show();
break;
case BluetoothAdapter.STATE_TURNING_ON:
Toast.makeText(context, "Turning On",
Toast.LENGTH_SHORT).show();
break;
}
}
}
}

```

Đối tượng `IntentFilter` nghe sự kiện `android.bluetooth.adapter.action.STATE_CHANGED`. Khi trạng thái bluetooth của thiết bị thay đổi, đối tượng `MyBTBroadcastReceiver` sẽ được gọi. Lớp `MyBTBroadcastReceiver` kế thừa từ lớp `BroadcastReceiver`, trong đó cài đặt phương thức callback `onReceive()`, phương thức này được gọi khi trạng thái bluetooth của thiết bị thay đổi nhằm đưa ra các xử lý tương ứng của ứng dụng.

Thăm dò các thiết bị thông qua kết nối bluetooth

Thiết lập cho thiết bị địa phương có thể thăm dò bởi thiết bị khác thông qua kết nối bluetooth.

Sử dụng phương thức `getScanMode()` để lấy về trạng thái thiết lập bluetooth của thiết bị địa phương. Phương thức này có giá trị trả về như sau:

- `SCAN_MODE_CONNECTABLE_DISCOVERABLE`: cho biết thiết bị địa phương có thể thăm dò thấy bởi bất kỳ thiết bị nào thông qua kết nối bluetooth.
- `SCAN_MODE_CONNECTABLE`: cho biết những thiết bị đã từng kết nối bluetooth với thiết bị địa phương có thể tiếp tục thăm dò thấy thiết bị địa phương thông qua kết nối bluetooth, tuy nhiên các thiết bị khác thì không.
- `SCAN_MODE_NONE`: cho biết thiết bị địa phương không thăm dò thấy bởi bất kỳ thiết bị nào thông qua kết nối bluetooth.

Mặc định thiết bị địa phương không được thăm dò bởi bất kỳ thiết bị nào thông qua kết nối bluetooth. Để cho phép thiết bị địa phương có thể thăm dò bởi thiết bị khác thì cần có sự cấp quyền từ phía người dùng thiết bị địa phương, bằng cách triệu gọi 1 Activity với tham số truyền vào là `ACTION_REQUEST_DISCOVERABLE` của lớp

BluetoothAdapter như sau:

```
startActivityForResult(  
    new Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE),  
    DISCOVERY_REQUEST);
```

Với việc cài đặt này, khi ứng dụng chạy, hộp thoại thông báo yêu cầu người dùng cấp quyền cho phép thiết bị địa phương được thăm dò bởi thiết bị khác như sau:



Hình 3.3. Giao diện bật tìm kiếm Bluetooth

Để kiểm tra trạng thái xác thực trên từ phía người dùng, cài đặt nạp chồng phương thức `onActivityResult()`. Giá trị `resultCode` trong phương thức này cho biết khoảng thời gian mà thiết bị địa phương có thể thăm dò bởi các thiết bị khác.

```
@Override  
protected void onActivityResult(int requestCode,  
    int resultCode, Intent data) {  
    if (requestCode == DISCOVERY_REQUEST) {  
        if (resultCode == RESULT_CANCELED) {  
            Log.d(TAG, "Discovery canceled by user");  
        }  
    }  
}
```

Ngoài ra, có thể theo dõi sự thay đổi trạng thái thăm dò của thiết bị địa phương khi sự kiện quảng bá `ACTION_SCAN_MODE_CHANGED` được gọi như sau:

```
registerReceiver(new BroadcastReceiver() {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        String prevScanMode = BluetoothAdapter.EXTRA_PREVIOUS_SCAN_MODE;  
        String scanMode = BluetoothAdapter.EXTRA_SCAN_MODE;  
        int currentScanMode = intent.getIntExtra(scanMode, -1);  
        int prevMode = intent.getIntExtra(prevScanMode, -1);  
        Log.d(TAG, "Scan Mode: " + currentScanMode +  
            ". Previous: " + prevMode);  
    }  
},  
    new IntentFilter(BluetoothAdapter.ACTION_SCAN_MODE_CHANGED));
```

Thăm dò thiết bị khác thông qua kết nối bluetooth

```
private ArrayList<BluetoothDevice> deviceList =  
    new ArrayList<BluetoothDevice>();  
private void startDiscovery() {  
    registerReceiver(discoveryResult,  
        new IntentFilter(BluetoothDevice.ACTION_FOUND));  
    if (bluetooth.isEnabled() && !bluetooth.isDiscovering())
```

```

deviceList.clear();
bluetooth.startDiscovery();
}
BroadcastReceiver discoveryResult = new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
String remoteDeviceName =
intent.getStringExtra(BluetoothDevice.EXTRA_NAME);
BluetoothDevice remoteDevice =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
deviceList.add(remoteDevice);
Log.d(TAG, "Discovered " + remoteDeviceName);
}
};

```

- isDiscovering() kiểm tra thiết bị địa phương có đang được thăm dò bởi thiết bị khác thông qua kết nối bluetooth hay không.
- startDiscovery() để thiết bị địa phương bắt đầu thăm dò thiết bị khác thông qua kết nối bluetooth.
- cancelDiscovery() để hủy bỏ quá trình thăm dò bluetooth của thiết bị địa phương.
- Giám sát quá trình thăm dò thiết bị thông qua kết nối bluetooth bằng cách tạo đối tượng BroadcastReceiver lắng nghe sự kiện do hệ thống gửi đến. Khi thiết bị địa phương thăm dò thấy bất kỳ thiết bị nào thì phương thức nạp chồng onReceive() sẽ được gọi để có những xử lý tương ứng.

Truyền dữ liệu giữa 2 thiết bị hỗ trợ bluetooth

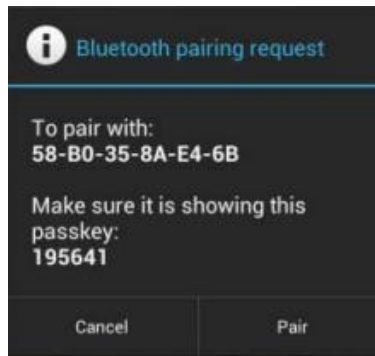
Thiết lập kênh kết nối RFCOMM giữa 2 thiết bị hỗ trợ bluetooth

- Mở socket lắng nghe kết nối trên thiết bị đích (Bluetooth Server Socket)

Gọi phương thức listenUsingRfcommWithServiceRecord() để mở socket lắng nghe sự kiện kết nối bluetooth gửi tới. Phương thức này có tham số truyền vào của là tên, định danh UUID của thiết bị đích và trả về đối tượng BluetoothServerSocket. Thiết bị nguồn muốn kết nối bluetooth tới thiết bị đích thì cần biết UUID của thiết bị đích.

Gọi phương thức accept() của đối tượng BluetoothServerSocket cho phép socket trên thiết bị đích (server socket) chấp nhận yêu cầu kết nối bluetooth từ thiết bị nguồn. Server socket chấp nhận kết nối nếu thiết bị nguồn biết UUID của thiết bị đích. Giá trị trả về của phương thức accept() là đối tượng BluetoothSocket để kết nối với thiết bị nguồn, socket này được sử dụng để truyền dữ liệu giữa 2 thiết bị.

Khi có yêu cầu kết nối gửi tới, hộp thoại hiển thị yêu cầu người dùng cần xác nhận thông tin kết nối:



Hình 3.4. Giao diện passkey

```
private BluetoothSocket transferSocket;
private UUID startServerSocket(BluetoothAdapter bluetooth) {
    UUID uuid = UUID.fromString("a60f35f0-b93a-11de-8a39-08002009c666");
    String name = "bluetoothserver";
    try {
        final BluetoothServerSocket btserver =
            bluetooth.listenUsingRfcommWithServiceRecord(name, uuid);
        Thread acceptThread = new Thread(new Runnable() {
            public void run() {
                try {
                    // Block until client connection established.
                    BluetoothSocket serverSocket = btserver.accept(); // Start
                    listening for messages. listenForMessages(serverSocket);
                    // Add a reference to the socket used to send messages.
                    transferSocket = serverSocket;
                } catch (IOException e) {
                    Log.e("BLUETOOTH", "Server connection IO Exception", e);
                }
            }
        });
        acceptThread.start();
    } catch (IOException e) {
        Log.e("BLUETOOTH", "Socket listener IO Exception", e);
    }
    return uuid;
}
```

Mở socket kết nối từ thiết bị nguồn (Client Bluetooth Socket)

Gọi phương thức `createRfcommSocketToServiceRecord()` để tạo đối tượng `BluetoothSocket` dùng để kết nối bluetooth từ thiết bị nguồn. Tham số truyền vào của phương thức này là UUID của thiết bị đích.

Sau khi đối tượng `BluetoothSocket` được tạo, gọi phương thức `connect()` của đối tượng này để khởi tạo kết nối tới thiết bị đích.

```
private void connectToServerSocket(BluetoothDevice device, UUID uuid)
{
    try{
        BluetoothSocket clientSocket
            = device.createRfcommSocketToServiceRecord(uuid);
        // Block until server connection accepted.
        clientSocket.connect();
        // Start listening for messages. listenForMessages(clientSocket);
        // Add a reference to the socket used to send messages.
    }
```

```
transferSocket = clientSocket;
} catch (IOException e) {
Log.e("BLUETOOTH", "Bluetooth client I/O Exception", e);
}
}
```

Truyền dữ liệu giữa 2 thiết bị qua kết nối bluetooth đã được thiết lập

Sau khi kênh kết nối bluetooth giữa 2 thiết bị được thiết lập. Bluetooth socket được tạo ra trên cả thiết bị nguồn và thiết bị đích. Khi đó 2 thiết bị này có thể truyền dữ liệu với nhau. Việc truyền dữ liệu giữa 2 thiết bị thông qua đối tượng `InputStream`, `OutputStream`.

```
private void listenForMessages(BluetoothSocket socket,
StringBuilder incoming) {
listening = true;
int bufferSize = 1024;
byte[] buffer = new byte[bufferSize];
try {
InputStream instream = socket.getInputStream();
int bytesRead = -1; while (listening) {
bytesRead = instream.read(buffer);
if (bytesRead != -1) {
String result = "";
while ((bytesRead == bufferSize) &&
(buffer[bufferSize-1] != 0)){
result = result + new String(buffer, 0, bytesRead - 1);
bytesRead = instream.read(buffer);
}
result = result + new String(buffer, 0, bytesRead - 1);
incoming.append(result);
}
socket.close();
}
} catch (IOException e) {
Log.e(TAG, "Message received failed.", e);
}
finally {
}
}
```

3.2. CÁC DỊCH VỤ DỰA TRÊN VỊ TRÍ (LOCATION - BASED SERVICES)

3.2.1. *Hiển thị bản đồ*

Google Maps là một trong nhiều ứng dụng đi kèm với nền tảng Android. Ngoài chỉ cần sử dụng ứng dụng Bản đồ, bạn cũng có thể nhúng nó vào các ứng dụng của riêng mình và làm cho nó hoạt động là nội dung rất thú vị. Phần này mô tả cách sử dụng Google Maps trong các ứng dụng Android của bạn và lập trình thực hiện như sau:

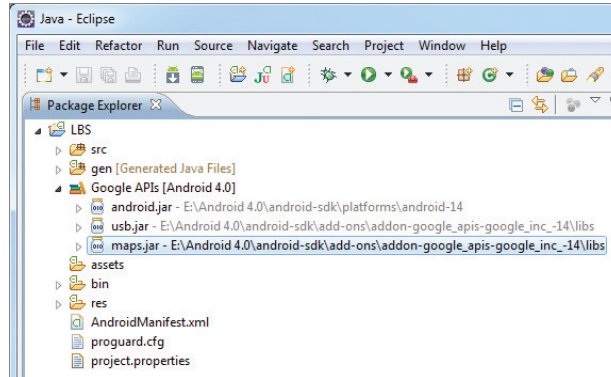
- Thay đổi quan điểm của Google Maps.
- Lấy vĩ độ và kinh độ của các vị trí trong Google Maps.
- Thực hiện mã hóa địa lý và mã hóa địa lý đảo ngược (dịch địa chỉ thành

vĩ độ và kinh độ và ngược lại).

- Thêm điểm đánh dấu vào Google Maps

Các bước thực hiện

1. Tạo dự án và đặt tên là LBS
2. Khi dự án được tạo ra, thêm tệp maps.jar vào thư mục API như hình vẽ



Hình 3.5. Thêm thư viện vào Google API

Lấy Maps API Key

Bắt đầu với phiên bản Android SDK v1.0, bạn cần đăng ký khóa API Google Maps miễn phí trước khi bạn có thể tích hợp Google Maps vào ứng dụng Android của mình. Khi bạn đăng ký Key, bạn cũng phải đồng ý với các điều khoản sử dụng của Google, vì vậy hãy nhớ đọc hướng dẫn cẩn thận.

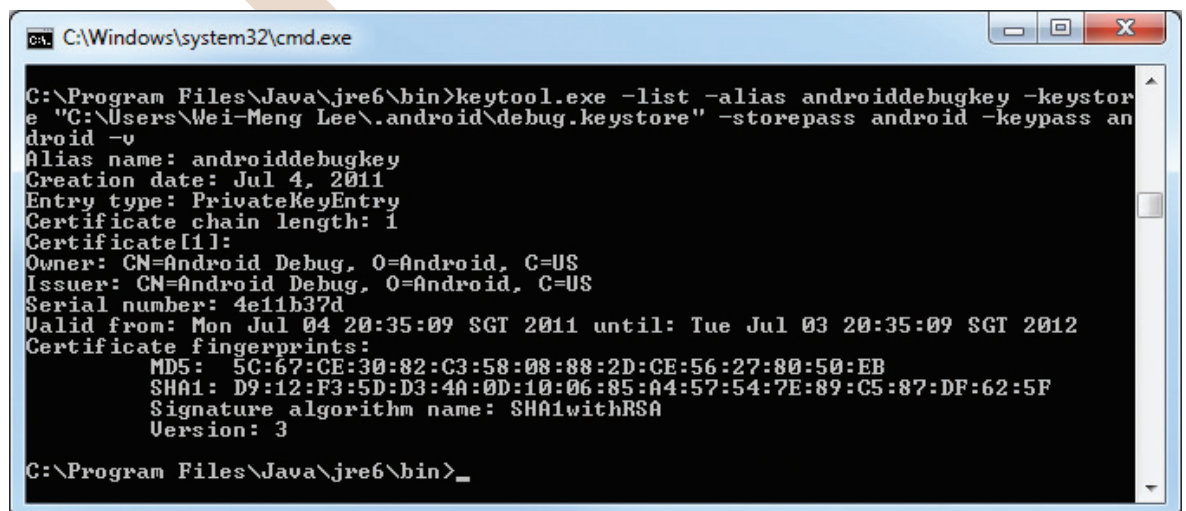
Xem chi tiết tại:

<http://code.google.com/android/add-ons/google-apis/mapkey.html>.

Tên file chứa Debug Keystore là debug.keystore

Để sử dụng Keystore này, cần giải nén nó ra sử dụng Keytool.exe trong phần cài đặt SDK

```
keytool.exe -list -alias androiddebugkey -keystore "C:\Users\<username>\.android\debug.keystore" -storepass android -keypass android -v
```

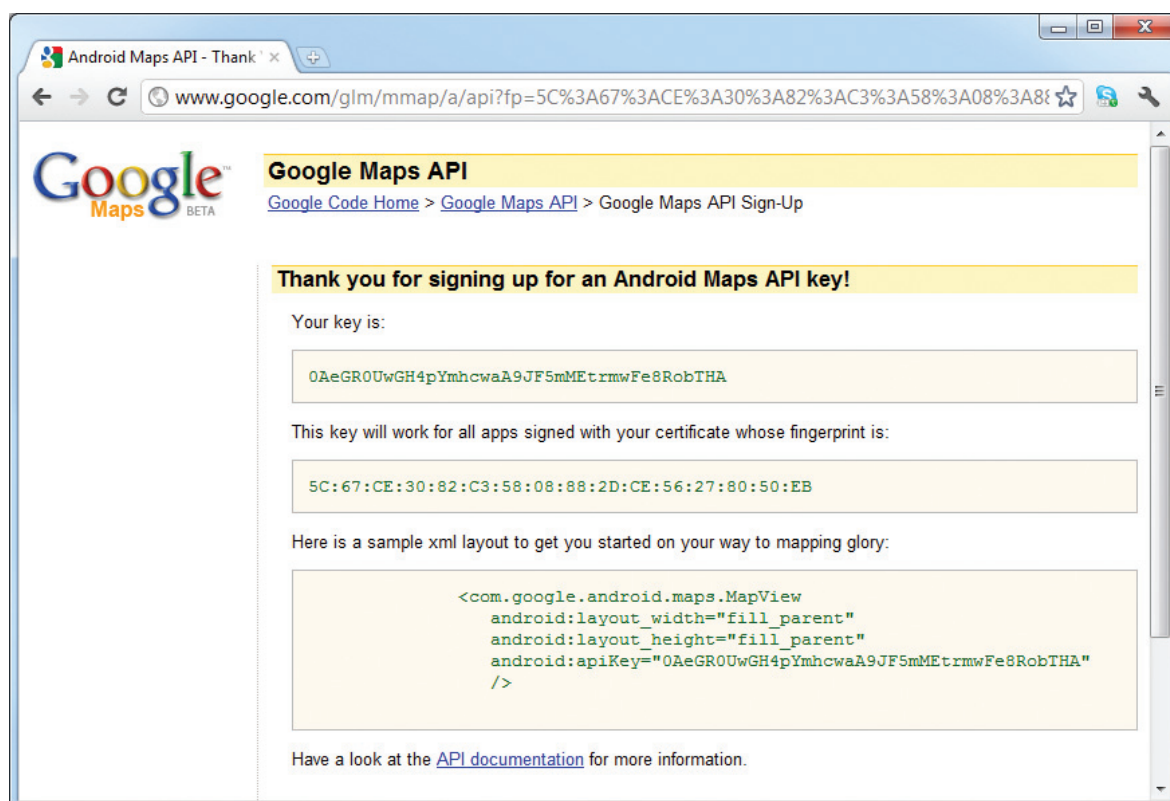


Hình 3.6. Giao diện lấy Keystore

Trong ví dụ trên, MD5 fingerprint là 5C:67:CE:30:82:C3:58:08:88:2D:CE:56:27:80:50:EB.

Sao chép MD5 đến trình duyệt web tại địa chỉ

<http://code.google.com/android/maps-api-signup.html> và làm theo hướng dẫn



Hình 3.7. Giao diện nhập đăng ký lên google

Hiện thị bản đồ

Việc hiện thị bản đồ cần hai nhiệm vụ chính

- Thay đổi file Androidmanifest.xml, thêm phần tử < uses-library > và cho phép Internet.
- Thêm phần tử MapView vào Giao diện UI

1. Tạo dự án và thêm Mapview

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <com.google.android.maps.MapView
        android:id="@+id/mapView"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:enabled="true"
        android:clickable="true"
        android:apiKey="0AeGR0UwGH4pYmhcwaA9JF5mMEtrmwFe8RobTHA" />
</LinearLayout>
```

2. Thêm 2 dòng in đậm

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.LBS"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="14" />
    <uses-permission android:name="android.permission.INTERNET"/>
```

```

<application
android:icon="@drawable/ic_launcher"
android:label="@string/app_name" >
<uses-library android:name="com.google.android.maps" />
<activity
android:label="@string/app_name"
android:name=".LBSActivity" >
<intent-filter >
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>

```

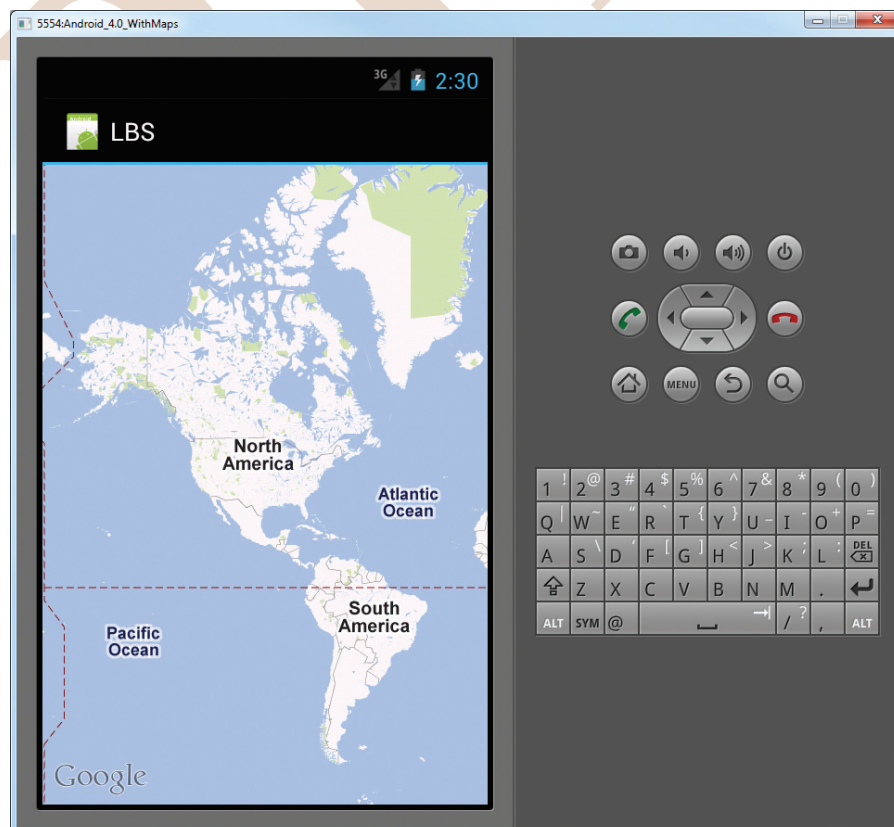
3. Thêm đoạn sau vào file LBSActivity.java

```

package net.learn2develop.LBS;
import com.google.android.maps.MapActivity;
import android.os.Bundle;
public class LBSActivity extends MapActivity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
}
@Override
protected boolean isRouteDisplayed() {
// TODO Auto-generated method stub
return false;
}
}

```

4. Chạy ứng dụng



3.2.2. Lấy dữ liệu định vị

Ngày nay, các thiết bị di động thường được trang bị máy thu GPS. Vì nhiều vệ tinh quay quanh trái đất, bạn có thể sử dụng máy thu GPS để tìm vị trí của mình một cách dễ dàng. Tuy nhiên, GPS yêu cầu một trời quang để làm việc và do đó không phải lúc nào cũng hoạt động trong nhà hoặc nơi các vệ tinh có thể xuyên thủng (như một đường hầm xuyên qua một ngọn núi).

Một cách hiệu quả khác để xác định vị trí của bạn là thông qua tam giác tháp di động. Khi di động Điện thoại được bật, nó liên tục tiếp xúc với các trạm cơ sở xung quanh nó. Bằng cách biết danh tính của các tháp di động, có thể dịch thông tin này vào một vị trí thực tế thông qua việc sử dụng các cơ sở dữ liệu khác nhau có chứa các bản sắc của tháp di động địa điểm. Ưu điểm của tam giác tháp di động là nó hoạt động trong nhà, không cần có được thông tin từ các vệ tinh. Tuy nhiên, nó không chính xác như GPS vì độ chính xác của nó phụ thuộc vào trên vùng phủ sóng tín hiệu chằng chéo, thay đổi khá nhiều. Tam giác tháp di động hoạt động tốt nhất trong khu vực đông dân cư, nơi các tháp di động nằm gần nhau.

Phương pháp thứ ba để xác định vị trí của bạn là dựa vào tam giác Wi-Fi. Thay vì kết nối với các tháp di động, thiết bị kết nối với mạng Wi-Fi và kiểm tra nhà cung cấp dịch vụ cơ sở dữ liệu để xác định vị trí được phục vụ bởi nhà cung cấp. Trong ba phương pháp được mô tả ở đây, Tam giác Wi-Fi là ít chính xác nhất.

Trong nền tảng Android, SDK cung cấp lớp LocationManager để giúp thiết bị xác định vị trí vật lý. Đoạn code sau minh chứng điều đó:

```
import java.io.IOException;
import java.util.List;
import java.util.Locale;
import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Point;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.widget.Toast;
public class LBSActivity extends MapActivity {
    MapView mapView;
    MapController mc;
    GeoPoint p;
    LocationManager lm;
    LocationListener locationListener;
```

```

private class MapOverlay extends com.google.android.maps.Overlay
{
//...
}
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
mapView = (MapView) findViewById(R.id.mapView);
mapView.setBuiltInZoomControls(true);
mapView.setSatellite(true);
mapView.setTraffic(true);
mc = mapView.getController();
/*
String coordinates[] = {"1.352566007", "103.78921587"};
double lat = Double.parseDouble(coordinates[0]);
double lng = Double.parseDouble(coordinates[1]);
p = new GeoPoint(
(int) (lat * 1E6),
(int) (lng * 1E6));
mc.animateTo(p);
mc.setZoom(13);
*/
//---geo-coding---
Geocoder geoCoder = new Geocoder(this, Locale.getDefault());
try {
//...
}
//---Add a location marker---
MapOverlay mapOverlay = new MapOverlay();
List<Overlay> listOfOverlays = mapView.getOverlays();
listOfOverlays.clear();
listOfOverlays.add(mapOverlay);
mapView.invalidate();
//---use the LocationManager class to obtain locations data---
lm = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
locationListener = new MyLocationListener();
}
@Override
public void onResume() {
super.onResume();
//---request for location updates---
lm.requestLocationUpdates(
LocationManager.GPS_PROVIDER,
0,
0,
locationListener);
}
@Override
public void onPause() {
super.onPause();
//---remove the location listener---
lm.removeUpdates(locationListener);
}
private class MyLocationListener implements LocationListener
{
public void onLocationChanged(Location loc) {
if (loc != null) {

```

```

Toast.makeText(getApplicationContext(),
    "Location changed : Lat: " + loc.getLatitude() +
    " Lng: " + loc.getLongitude(),
    Toast.LENGTH_SHORT).show();
p = new GeoPoint(
    (int) (loc.getLatitude() * 1E6),
    (int) (loc.getLongitude() * 1E6));
mc.animateTo(p);
mc.setZoom(18);
}
}

public void onProviderDisabled(String provider) {
}

public void onProviderEnabled(String provider) {
}

public void onStatusChanged(String provider, int status,
    Bundle extras) {
}
}

public boolean onKeyDown(int keyCode, KeyEvent event)
{
    //...
}

@Override
protected boolean isRouteDisplayed() {
    //...
}

```

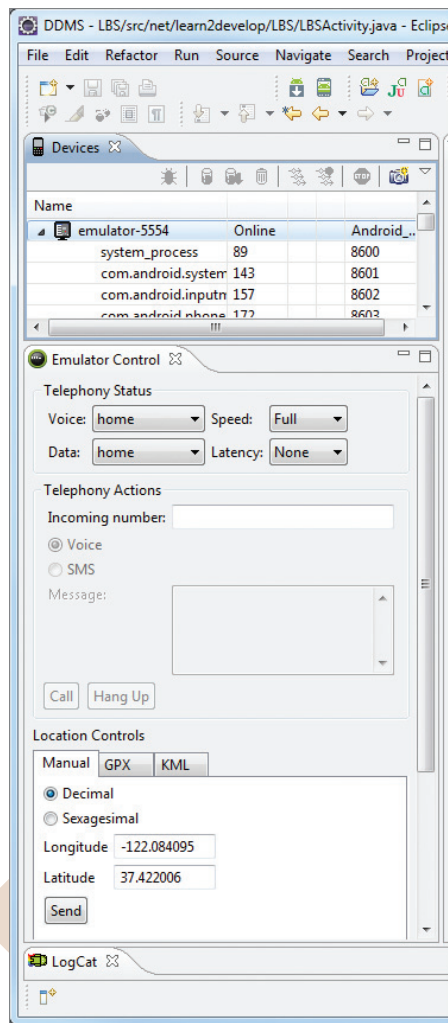
Thêm 2 dòng sau vào Androidmanifest.xml

```

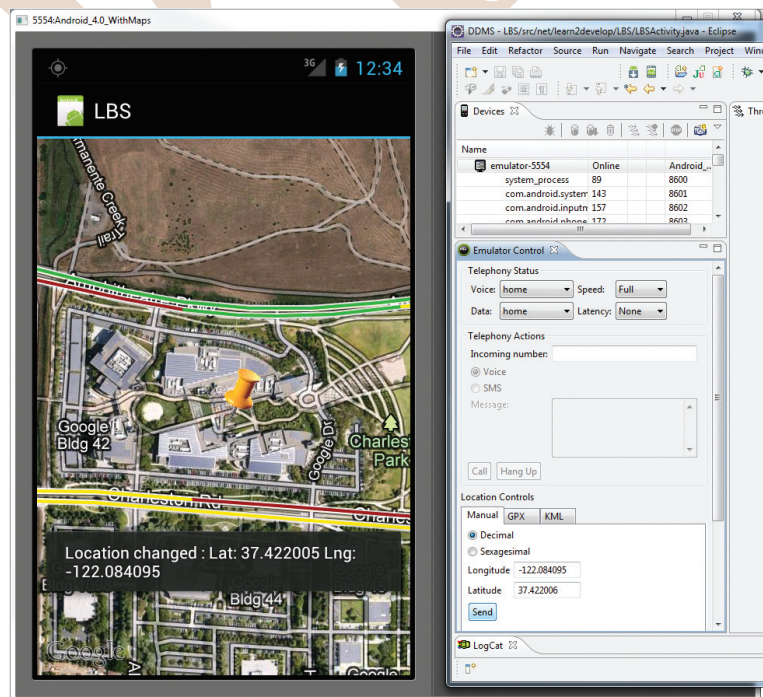
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.LBS"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="14" />
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <uses-library android:name="com.google.android.maps" />
        <activity
            android:label="@string/app_name"
            android:name=".LBSActivity" >
            <intent-filter >
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>

```

Có thể sử dụng DDMS để xem kết quả



Hình 3.9. Kết quả xác định vị trí của thiết bị
Có thể thay đổi vị trí của thiết bị thông qua DDMS



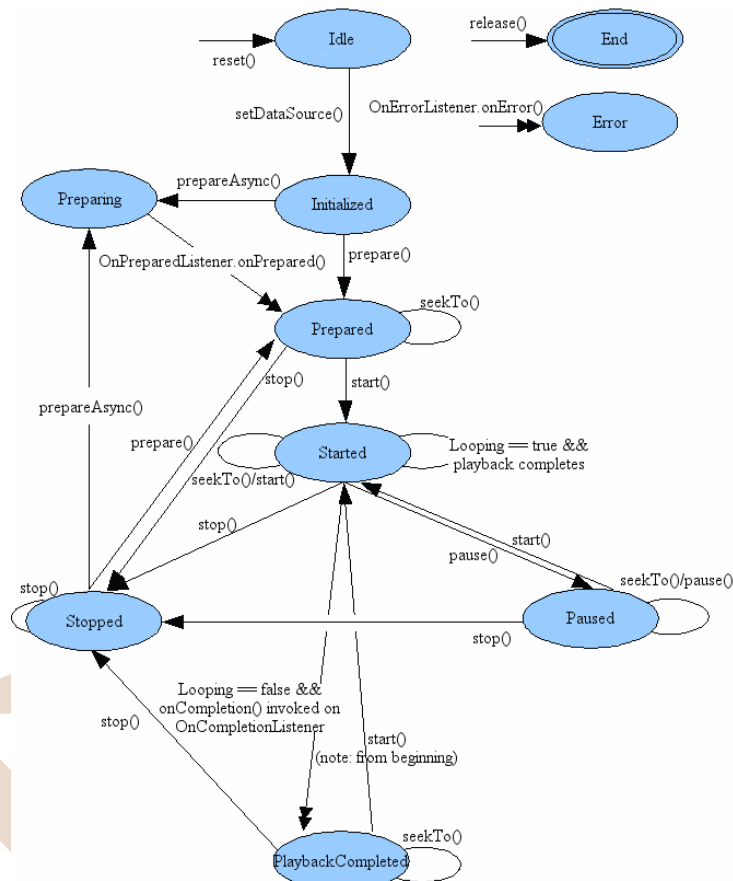
Hình 3.10. Thay đổi vị trí thiết bị qua DDMS

3.3. AUDIO, VIDEO, VÀ SỬ DỤNG CAMERA

3.3.1. Audio & Video

Android cung cấp lớp MediaPlayer gồm 1 tập hợp các API cho phép phát các file audio, video được lưu trữ trong chính ứng dụng, trên hệ thống file của thiết bị hoặc truyền tải trực tiếp từ trên mạng (streamming audio/ video).

Máy trạng thái (state machine) quản lý việc phát audio và video thông qua MediaPlayer như sau:



Hình 3.11. Máy trạng thái quản lý việc phát audio và video thông qua MediaPlayer

Tóm tắt quá trình chuyển tiếp các trạng thái trong máy trạng thái qua 5 bước sau:

1. Khởi tạo đối tượng MediaPlayer với nguồn thông tin đầu vào là file audio/video.
2. Chuẩn bị phát audio/ video.
3. Bắt đầu phát audio/ video.
4. Dừng tạm thời hoặc ngừng phát audio/ video.
5. Hoàn thành phát audio/ video.

Để phát audio/ video trực tiếp từ mạng sử dụng MediaPlayer cần cấp quyền truy cập INTERNET cho ứng dụng như sau:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Chuẩn bị phát audio

File audio có thể được xác định thông qua: vị trí trực tiếp lưu trong project (thư mục res/raw); URI của file trên hệ thống (file://schema); URI trực tuyến lưu file (URL);

URI Content Provider.

Việc chuẩn bị phát audio được thực hiện thông qua đối tượng MediaPlayer. Quá trình này được thực hiện theo 1 trong 2 cách sau:

Cách 1:

```
// Load an audio resource from a package resource.
MediaPlayer resourcePlayer = MediaPlayer.create(this, R.raw.my_audio);
// Load an audio resource from a local file.
MediaPlayer filePlayer = MediaPlayer.create(this,
Uri.parse("file:///sdcard/localfile.mp3"));
// Load an audio resource from an online resource.
MediaPlayer urlPlayer = MediaPlayer.create(this,
Uri.parse("http://site.com/audio/audio.mp3"));
// Load an audio resource from a Content Provider.
MediaPlayer contentPlayer = MediaPlayer.create(this,
Settings.System.DEFAULT_RINGTONE_URI);
```

Cách 2:

```
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setDataSource("/sdcard/mydopetunes.mp3");
mediaPlayer.prepare();
```

Chuẩn bị phát video

Để phát video cần có giao diện hiển thị video. Giao diện hiển thị video được tạo ra theo 1 trong 2 cách: 1/ Sử dụng lớp VideoView do Android cung cấp; 2/ Tự tạo ra giao diện phát video.

Phát video sử dụng VideoView:

- Khai báo đối tượng VideoView trong giao diện hiển thị.
- Thiết lập nguồn dữ liệu video hiển thị trong VideoView:

```
// Get a reference to the Video View.
final VideoView videoView =
(VideoView)findViewById(R.id.videoView);
// Assign a local file to play
videoView.setVideoPath("/sdcard/mycatvideo.3gp");
// Assign a URL of a remote video stream
videoView.setVideoUri(myAwesomeStreamingSource);
```

Điều khiển việc phát video với VideoView thông qua MediaController như sau:

```
// Attach a Media Controller
MediaController mediaController = new MediaController(this);
videoView.setMediaController(mediaController);
```

Tự tạo giao diện phát video riêng với SurfaceView

MediaPlayer sử dụng phương thức setDisplay() để hiển thị nội dung video với tham số truyền vào là đối tượng SurfaceHolder.

- Tạo giao diện hiển thị video với SurfaceView:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
```

```

android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical" >
<SurfaceView
android:id="@+id/surfaceView"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_weight="30" />
<LinearLayout
android:id="@+id/linearLayout1"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_weight="1">
<Button
android:id="@+id/buttonPlay"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Play" />
<Button
android:id="@+id/buttonPause"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Pause" />
<Button
android:id="@+id/buttonSkip"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Skip" />
</LinearLayout>
</LinearLayout>

```

- Khởi tạo và gán SurfaceView với MediaPlayer

```

import java.io.IOException;
import android.app.Activity;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.util.Log;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
public class SurfaceViewVideoViewActivity extends Activity
implements SurfaceHolder.Callback {
    static final String TAG = "SurfaceViewVideoViewActivity";
    private MediaPlayer mediaPlayer;
    public void surfaceCreated(SurfaceHolder holder) {
        try {
            // When the surface is created, assign it as the
            // display surface and assign and prepare a data
            // source.
            mediaPlayer.setDisplay(holder);

```

```

        mediaPlayer.setDataSource("/sdcard/test2.3gp");
        mediaPlayer.prepare();
    }
}

public void surfaceDestroyed(SurfaceHolder holder) {
    mediaPlayer.release();
}

public void surfaceChanged(SurfaceHolder holder,
    int format, int width, int height) { }

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.surfaceviewvideoviewer);
    // Create a new Media Player.
    mediaPlayer = new MediaPlayer();
    // Get a reference to the Surface View.
    final SurfaceView surfaceView =
        (SurfaceView)findViewById(R.id.surfaceView);
    // Configure the Surface View.
    surfaceView.setKeepScreenOn(true);
    // Configure the Surface Holder and register the callback.
    SurfaceHolder holder = surfaceView.getHolder();
    holder.addCallback(this);
    holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    holder.setFixedSize(400, 300);
    // Connect a play button.
    Button playButton = (Button)findViewById(R.id.buttonPlay);
    playButton.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            mediaPlayer.start();
        }
    });
    // Connect a pause button.
    Button pauseButton = (Button)findViewById(R.id.buttonPause);
    pauseButton.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            mediaPlayer.pause();
        }
    });
    // Add a skip button.
    Button skipButton = (Button)findViewById(R.id.buttonSkip);
    skipButton.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            mediaPlayer.seekTo(mediaPlayer.getDuration()/2);
        }
    });
}
}

```

Phát audio/ video

Sau khi MediaPlayer đã được chuẩn bị, gọi phương thức start() của MediaPlayer

để khởi động phát audio/ video :

```
mediaPlayer.start();
```

3.3.2. Camera

Android cung cấp 1 tập các API camera và camera Intent cho phép tạo ra các ứng dụng làm việc với các loại camera và các tính năng camera khác nhau có sẵn trên thiết bị.

Một số lớp hỗ trợ làm việc với camera được Android cung cấp bao gồm:

- Camera: cung cấp những API điều khiển camera. Lớp này dùng để xây dựng ứng dụng có khả năng chụp ảnh, video.
- SurfaceView: sử dụng để hiển thị màn hình camera trong ứng dụng.
- MediaRecorder: sử dụng để ghi video từ camera.
- Intent: dùng để gọi ứng dụng camera có sẵn trên thiết bị trong ứng dụng xây dựng, thông qua kiểu hành động: `MediaStore.ACTION_IMAGE_CAPTURE` hoặc `MediaStore.ACTION_VIDEO_CAPTURE`

Trước khi phát triển ứng dụng với API camera cần khai báo quyền cho ứng dụng có khả năng làm việc với camera trên thiết bị:

- Cấp quyền cho ứng dụng sử dụng các loại camera trên thiết bị:

```
<uses-permission android:name="android.permission.CAMERA" />
```

- Cấp quyền cho ứng dụng sử dụng các đặc tính camera trên thiết bị:

```
<uses-feature android:name="android.hardware.camera" />
```

Có thể cấp quyền cho ứng dụng sử dụng các đặc tính cụ thể như:

- `android.hardware.camera.autofocus`: sử dụng tính năng autofocus của thiết bị camera trong ứng dụng.
- `android.hardware.camera.flash`: sử dụng tính năng flash của thiết bị camera trong ứng dụng.
- `android.hardware.camera.front`: sử dụng camera trước của thiết bị camera trong ứng dụng.
- `android.hardware.camera.any`: sử dụng ít nhất 1 camera trên thiết bị hoặc camera kết nối với thiết bị.
- `android.hardware.camera.external`: sử dụng camera ngoài nếu nó được kết nối tới.

- Cấp quyền lưu trữ ảnh/ video trên bộ nhớ ngoài (SD card) cho ứng dụng:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

- Cấp quyền thu audio cho ứng dụng:

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
```

- Cấp quyền cho phép ứng dụng có khả năng chụp ảnh kèm theo ghi lại thông tin vị trí GPS:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Xây dựng ứng dụng camera từ ứng dụng camera có sẵn trên thiết bị

Đây là cách xây dựng ứng dụng có khả năng chụp ảnh/ video bằng cách triệu gọi ứng dụng camera có sẵn trên thiết bị thông qua Intent. Các bước thực hiện:

- Tạo 1 Camera Intent sử dụng tính năng chụp ảnh/ video của ứng dụng camera có sẵn với kiểu hành động chụp ảnh (MediaStore.ACTION_IMAGE_CAPTURE) hoặc quay video (MediaStore.ACTION_VIDEO_CAPTURE).
- Thực thi Camera Intent có sẵn bằng cách gọi phương thức startActivityForResult(). Khi phương thức này được gọi, giao diện ứng dụng Camera có sẵn trên thiết bị hiển thị cho phép người dùng chụp ảnh / video.
- Lấy lại kết quả Camera Intent trả về thông qua phương thức callback onActivityResult() sau khi chụp ảnh / video.

Gọi Camera Intent sử dụng tính năng chụp ảnh

```
private static final int CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE = 100;
private Uri fileUri;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    // create Intent to take a picture and return control to the calling application
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

    fileUri = getOutputMediaFileUri(MEDIA_TYPE_IMAGE); // create a file to save the image
    intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri); // set the image file name

    // start the image capture Intent
    startActivityForResult(intent, CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE);
}
```

Trong đó: thiết lập một số thông tin liên quan tới Camera Intent thông qua phương thức putExtra() như sau:

- MediaStore.EXTRA_OUTPUT: thiết lập tên và vị trí lưu trữ file với fileUri truyền vào. Nếu thông tin này không được thiết lập thì file ảnh sẽ được lưu vào vị trí mặc định với tên mặc định.

Khi phương thức startActivityForResult() được thực thi, người dùng sẽ nhìn thấy giao diện ứng dụng camera có sẵn trên thiết bị. Sau khi người dùng sử dụng / hủy bỏ chụp ảnh, ứng dụng camera thoát và phương thức onActivityResult() trong ứng dụng được gọi cho phép thu lại kết quả sử dụng camera có sẵn đó trong ứng dụng xây dựng.

Gọi Camera Intent sử dụng tính năng quay video

```
private static final int CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE = 200;
private Uri fileUri;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    //create new Intent
    Intent intent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);

    fileUri = getOutputMediaFileUri(MEDIA_TYPE_VIDEO); // create a file to save the vi
    intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri); // set the image file name

    intent.putExtra(MediaStore.EXTRA_VIDEO_QUALITY, 1); // set the video image quality

    // start the Video Capture Intent
    startActivityForResult(intent, CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE);
}
```

Trong đó: thiết lập một số thông tin liên quan tới Camera Intent thông qua phương thức putExtra() như sau:

- MediaStore.EXTRA_OUTPUT: thiết lập tên và vị trí lưu trữ file với fileUri truyền vào. Nếu thông tin này không được thiết lập thì file video sẽ được lưu vào vị trí mặc định với tên mặc định.
- MediaStore.EXTRA_VIDEO_QUALITY: thiết lập chất lượng ảnh. Với giá trị là 0 thì chất lượng và kích thước ảnh thấp nhất, giá trị là 1 thì chất lượng và kích thước ảnh cao nhất.
- MediaStore.EXTRA_DURATION_LIMIT: thiết lập độ dài giới hạn của đoạn video theo giây.
- MediaStore.EXTRA_SIZE_LIMIT: thiết lập kích thước dung lượng giới hạn của file video theo byte.

Khi phương thức startActivityForResult() được thực thi, người dùng sẽ nhìn thấy giao diện ứng dụng camera có sẵn trên thiết bị. Sau khi người dùng sử dụng / hủy bỏ thu video, ứng dụng camera thoát và phương thức onActivityResult() trong ứng dụng được gọi cho phép thu lại kết quả sử dụng camera có sẵn đó trong ứng dụng xây dựng.

Lấy lại kết quả Camera Intent trả về

```

private static final int CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE = 100;
private static final int CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE = 200;

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE) {
        if (resultCode == RESULT_OK) {
            // Image captured and saved to fileUri specified in the Intent
            Toast.makeText(this, "Image saved to:\n" +
                data.getData(), Toast.LENGTH_LONG).show();
        } else if (resultCode == RESULT_CANCELED) {
            // User cancelled the image capture
        } else {
            // Image capture failed, advise user
        }
    }

    if (requestCode == CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE) {
        if (resultCode == RESULT_OK) {
            // Video captured and saved to fileUri specified in the Intent
            Toast.makeText(this, "Video saved to:\n" +
                data.getData(), Toast.LENGTH_LONG).show();
        } else if (resultCode == RESULT_CANCELED) {
            // User cancelled the video capture
        } else {
            // Video capture failed, advise user
        }
    }
}

```

Xây dựng ứng dụng camera không phụ thuộc vào ứng dụng camera có sẵn trên thiết bị

Các bước thực hiện:

- Kiểm tra camera tồn tại trên thiết bị và gửi yêu cầu truy cập.
- Tạo lớp xem trước ảnh/ video từ camera, lớp này kế thừa từ lớp SurfaceView và thực thi giao diện SurfaceHolder.
- Xây dựng giao diện hiển thị và điều khiển camera.
- Xử lý sự kiện chụp ảnh/ video.
- Lưu file.
- Giải phóng camera.

Kiểm tra camera tồn tại trên thiết bị

Sử dụng phương thức PackageManager.hasSystemFeature() để kiểm tra camera có tồn tại trên thiết bị không.


```

/** Check if this device has a camera */
private boolean checkCameraHardware(Context context) {
    if (context.getPackageManager().hasSystemFeature(PackageManager.FEATURE_CAMERA)) {
        // this device has a camera
        return true;
    } else {
        // no camera on this device
        return false;
    }
}

```

Ngoài ra 1 thiết bị đầu cuối di động có thể gắn nhiều camera như camera trước, camera sau, camera ngoài. Để kiểm tra số lượng camera gắn trên thiết bị sử dụng phương thức: `Camera.getNumberOfCameras()`.

Truy cập camera

Truy cập camera thông qua đối tượng `Camera` được tạo ra thông qua phương thức `open()` như sau:

```

/** A safe way to get an instance of the Camera object. */
public static Camera getCameraInstance(){
    Camera c = null;
    try {
        c = Camera.open(); // attempt to get a Camera instance
    }
    catch (Exception e){
        // Camera is not available (in use or does not exist)
    }
    return c; // returns null if camera is unavailable
}

```

Kiểm tra các đặc tính camera

Sử dụng phương thức `Camera.getParameters()` để lấy về thông tin các đặc tính camera.

Tạo lớp xem trước ảnh/ video từ camera

Lớp này thực thi lớp `SurfaceHolder.Callback`, cài đặt phương thức callback `surfaceCreated()` và `surfaceDestroyed()` để tạo và hủy giao diện xem trước ảnh/ video từ camera.

```

/** A basic Camera preview class */
public class CameraPreview extends SurfaceView implements
    SurfaceHolder.Callback {
    private SurfaceHolder mHolder;
    private Camera mCamera;
    public CameraPreview(Context context, Camera camera) {
        super(context);
        mCamera = camera;
        // Install a SurfaceHolder.Callback so we get notified when the
        // underlying surface is created and destroyed.
        mHolder = getHolder();
        mHolder.addCallback(this);
        // deprecated setting, but required on Android versions prior to

```

```

3.0
mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
}
public void surfaceCreated(SurfaceHolder holder) {
    // The Surface has been created, now tell the camera where to
    draw
    the preview.
    try {
        mCamera.setPreviewDisplay(holder);
        mCamera.startPreview();
    } catch (IOException e) {
        Log.d(TAG, "Error setting camera preview: " + e.getMessage());
    }
}
public void surfaceDestroyed(SurfaceHolder holder) {
    // empty. Take care of releasing the Camera preview in your
    activity.
}
public void surfaceChanged(SurfaceHolder holder, int format, int
w, int h)
{
    // If your preview can change or rotate, take care of those
    events here.
    // Make sure to stop the preview before resizing or reformatting
    it.
    if (mHolder.getSurface() == null){
        // preview surface does not exist
        return;
    }
    // stop preview before making changes
    try {
        mCamera.stopPreview();
    } catch (Exception e){
        // ignore: tried to stop a non-existent preview
    }
    // set preview size and make any resize, rotate or
    // reformatting changes here
    // start preview with new settings
    try {
        mCamera.setPreviewDisplay(mHolder);
        mCamera.startPreview();
    } catch (Exception e){
        Log.d(TAG, "Error starting camera preview: " + e.getMessage());
    }
}
}

```

Xây dựng giao diện hiển thị và điều khiển camera

Lớp xem trước ảnh/ video từ camera được triệu gọi ở code thực thi của activity trong ứng dụng để điều khiển việc hiển thị camera.

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <FrameLayout
        android:id="@+id/camera_preview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
    />

    <Button
        android:id="@+id/button_capture"
        android:text="Capture"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
    />
</LinearLayout>

```

Trong AndroidManifest.xml khai báo việc thiết lập hiển thị activity theo chiều ngang (landscape) như sau:

```

<activity android:name=".CameraActivity"
    android:label="@string/app_name"

    android:screenOrientation="landscape">
    <!-- configure this activity to use landscape orientation -->

    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

```

Code thực thi của activity, triệu gọi xem trước ảnh/ video từ camera:

```

public class CameraActivity extends Activity {

    private Camera mCamera;
    private CameraPreview mPreview;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // Create an instance of Camera
        mCamera = getCameraInstance();

        // Create our Preview view and set it as the content of our activity.
        mPreview = new CameraPreview(this, mCamera);
        FrameLayout preview = (FrameLayout) findViewById(R.id.camera_preview);
        preview.addView(mPreview);
    }
}

```

Xử lý sự kiện chụp ảnh

Sử dụng phương thức `Camera.takePicture()` để chụp ảnh thông qua sự kiện nhấn nút chụp:

```

// Add a listener to the Capture button
Button captureButton = (Button) findViewById(id.button_capture);
captureButton.setOnClickListener(
    new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // get an image from the camera
            mCamera.takePicture(null, null, mPicture);
        }
    }
);

```

Để file ảnh có định dạng JPEG, ứng dụng cần thực thi giao diện `Camera.PictureCallback` để lấy và lưu ảnh chụp được dưới dạng file từ camera như sau:

```

private PictureCallback mPicture = new PictureCallback() {

    @Override
    public void onPictureTaken(byte[] data, Camera camera) {

        File pictureFile = getOutputMediaFile(MEDIA_TYPE_IMAGE);
        if (pictureFile == null){
            Log.d(TAG, "Error creating media file, check storage permissions: " +
                e.getMessage());
            return;
        }

        try {
            FileOutputStream fos = new FileOutputStream(pictureFile);
            fos.write(data);
            fos.close();
        } catch (FileNotFoundException e) {
            Log.d(TAG, "File not found: " + e.getMessage());
        } catch (IOException e) {
            Log.d(TAG, "Error accessing file: " + e.getMessage());
        }
    }
};

```

Chú ý: khi ứng dụng không sử dụng camera nữa cần giải phóng bằng cách sử dụng phương thức `Camera.release()`.

Xử lý sự kiện ghi video

Các bước thực hiện để ghi video qua camera trong ứng dụng như sau:

- Cấu hình `MediaRecorder`

Khi sử dụng `MediaRecorder` để ghi video, lập trình viên cần thực hiện thiết lập các thông số cần thiết và gọi phương thức `MediaRecorder.prepare()` như sau:

```

private boolean prepareVideoRecorder() {
    mCamera = getCameraInstance();
    mMediaRecorder = new MediaRecorder();
    // Step 1: Unlock and set camera to MediaRecorder
    mCamera.unlock();
    mMediaRecorder.setCamera(mCamera);
    // Step 2: Set sources
    mMediaRecorder.setAudioSource(MediaRecorder.AudioSource.CAMCORDER);
    mMediaRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);
    // Step 3: Set a CamcorderProfile (requires API Level 8 or higher)
    mMediaRecorder.setProfile(CamcorderProfile.get(CamcorderProfile.QUALITY_HIGH));
    // Step 4: Set output file
    mMediaRecorder.setOutputFile(getOutputMediaFile(MEDIA_TYPE_VIDEO).toString());
    // Step 5: Set the preview output
    mMediaRecorder.setPreviewDisplay(mPreview.getHolder().getSurface());
}

```

```
// Step 6: Prepare configured MediaRecorder
try {
    mMediaRecorder.prepare();
} catch (IllegalStateException e) {
    Log.d(TAG, "IllegalStateException preparing MediaRecorder: " +
        e.getMessage());
    releaseMediaRecorder();
    return false;
} catch (IOException e) {
    Log.d(TAG, "IOException preparing MediaRecorder: " +
        e.getMessage());
    releaseMediaRecorder();
    return false;
}
return true;
```

Với Android 2.2 (API level 8) trở về trước, lập trình viên cần thiết lập các thông số định dạng đầu ra và thực hiện mã hóa định dạng trực tiếp thay vì sử dụng CamcorderProfile, như sau:

```
// Step 3: Set output format and encoding (for versions prior to API Level 8)
mMediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
mMediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT);
mMediaRecorder.setVideoEncoder(MediaRecorder.VideoEncoder.DEFAULT);
```

Ngoài ra, Android cung cấp một số phương thức cho việc thiết lập tham số cho MediaRecorder sau (Nếu các tham số này không được thiết lập thì nó sẽ nhận giá trị mặc định):

```
setVideoEncodingBitRate()
setVideoSize()
setVideoFrameRate()
setAudioEncodingBitRate()
setAudioChannels()
setAudioSamplingRate()
• Khởi động và dừng MediaRecorder
```

Các bước thực hiện:

1. Mở khóa camera sử dụng Camera.unlock()
2. Cấu hình MediaRecorder như trên.
3. Khởi động ghi âm video sử dụng MediaRecorder.start().
4. Ghi âm video.
5. Dừng ghi âm video sử dụng MediaRecorder.stop()
6. Giải phóng media recorder sử dụng MediaRecorder.release().
7. Khóa camera sử dụng Camera.lock()

Mình họa dưới đây thực hiện bắt sự kiện nút nhấn để khởi động và dừng ghi âm với camera :

```
private boolean isRecording = false;
// Add a listener to the Capture button
```

```

Button captureButton = (Button) findViewById(id.button_capture);
captureButton.setOnClickListener(
    new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (isRecording) {
                // stop recording and release camera
                mMediaRecorder.stop(); // stop the recording
                releaseMediaRecorder(); // release the MediaRecorder object
                mCamera.lock(); // take camera access back from MediaRecorder
                // inform the user that recording has stopped
                setCaptureButtonText("Capture");
                isRecording = false;
            } else {
                // initialize video camera
                if (prepareVideoRecorder()) {
                    // Camera is available and unlocked, MediaRecorder is prepared,
                    // now you can start recording
                    mMediaRecorder.start();
                    // inform the user that recording has started
                    setCaptureButtonText("Stop");
                    isRecording = true;
                } else {
                    // prepare didn't work, release the camera
                    releaseMediaRecorder();
                    // inform user
                }
            }
        }
    }
);

```

Giải phóng camera

Camera là tài nguyên dùng chung bởi tất cả các ứng dụng trên thiết bị, do đó khi ứng dụng không cần sử dụng camera nữa cần giải phóng camera để hệ thống thu hồi lại tài nguyên cấp cho ứng dụng khác. Để giải phóng camera gọi phương thức `Camera.release()` như sau:

```

public class CameraActivity extends Activity {
    private Camera mCamera;
    private SurfaceView mPreview;
    private MediaRecorder mMediaRecorder;
    ...
    @Override
    protected void onPause() {
        super.onPause();
        super.onPause();
        releaseMediaRecorder(); // if you are using MediaRecorder,
        release
        //it first
        releaseCamera(); // release the camera immediately on pause
        event
    }
    private void releaseMediaRecorder(){
        if (mMediaRecorder != null) {
            mMediaRecorder.reset(); // clear recorder configuration
            mMediaRecorder.release(); // release the recorder object
            mMediaRecorder = null;

```



```

mCamera.lock(); // lock camera for later use
}
}
private void releaseCamera(){
if (mCamera != null){
mCamera.release(); // release the camera for other applications
mCamera = null;
}
}
}
}

```

Lưu file ảnh/ video chụp được từ camera

Các tập tin media được tạo ra bởi người sử dụng như hình ảnh và video sẽ được lưu vào thư mục lưu trữ bên ngoài (SD Card) của thiết bị để bảo tồn không gian hệ thống và cho phép người dùng truy cập vào các tập tin ngay cả khi không có thiết bị. Có rất nhiều vị trí thư mục có thể lưu các tập tin media trên thiết bị, tuy nhiên chỉ có hai địa điểm tiêu chuẩn được khuyến nghị cho việc lưu file media, đó là:

- `Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES)` – Phương thức này trả về vị trí lưu trữ file media khuyến nghị trên thiết bị, các file media được tạo ra từ ứng dụng xây dựng này được phép xem, thay đổi và xóa file bởi các ứng dụng khác. Khi ứng dụng xây dựng bị gỡ bỏ thì file tạo ra bởi ứng dụng đó cũng không bị xóa khỏi vị trí lưu trữ đó.
- `Context.getExternalFilesDir(Environment.DIRECTORY_PICTURES)` – Phương thức này trả về vị trí lưu trữ file media kết hợp với ứng dụng xây dựng, các file media được tạo ra từ ứng dụng xây dựng này được phép xem, thay đổi và xóa file bởi các ứng dụng khác. Khi ứng dụng xây dựng bị gỡ bỏ thì file tạo ra bởi ứng dụng đó cũng bị xóa khỏi vị trí lưu trữ đó.

Đoạn mã ví dụ sau đây minh họa cách lưu file ảnh/ video chụp được từ camera:

```

public static final int MEDIA_TYPE_IMAGE = 1;
public static final int MEDIA_TYPE_VIDEO = 2;
/** Create a file Uri for saving an image or video */
private static Uri getOutputMediaFileUri(int type){
return Uri.fromFile(getOutputMediaFile(type));
}
/** Create a File for saving an image or video */
private static File getOutputMediaFile(int type){
// To be safe, you should check that the SDCard is mounted
// using Environment.getExternalStorageState() before doing this.
File mediaStorageDir = new
File(Environment.getExternalStoragePublicDirectory(
Environment.DIRECTORY_PICTURES), "MyCameraApp");
// This location works best if you want the created images to be
shared
// between applications and persist after your app has been
uninstalled.
// Create the storage directory if it does not exist
if (! mediaStorageDir.exists()){
if (! mediaStorageDir.mkdirs()){
Log.d("MyCameraApp", "failed to create directory");
}
}
}
}

```

```

return null;
}
}
// Create a media file name
String timeStamp = new
SimpleDateFormat("yyyyMMdd_HH:mm:ss").format(new
Date());
File mediaFile;
if (type == MEDIA_TYPE_IMAGE) {
mediaFile = new File(mediaStorageDir.getPath() + File.separator +
"IMG_" + timeStamp + ".jpg");
} else if (type == MEDIA_TYPE_VIDEO) {
mediaFile = new File(mediaStorageDir.getPath() + File.separator +
"VID_" + timeStamp + ".mp4");
} else {
return null;
}
return mediaFile;
}

```

3.4. SENSORS PHẦN CỨNG

Hầu hết các thiết bị Android đều gắn sẵn các cảm biến (sensor)) để theo dõi chuyển động, hướng và môi trường xung quanh thiết bị. Các cảm biến này có khả năng thu thập dữ liệu thô với độ chính xác cao. Việc khai thác các thông tin thu được từ cảm biến gắn trên thiết bị cho phép lập trình viên tạo ra các ứng dụng đa dạng, ví dụ như ứng dụng thời tiết thu thập thông tin từ cảm biến nhiệt độ và cảm biến độ ẩm để tính toán và đưa ra thông báo về thời tiết tại một địa điểm nào đó.

3.4.1. Quản lý Sensors

Android hỗ trợ 3 loại cảm biến sau:

- Loại cảm biến chuyển động (Motion sensors): đây là các cảm biến đo lực gia tốc và lực quay dọc theo 3 trục tọa độ. Nhóm này bao gồm gồm 1 số cảm biến như: cảm biến gia tốc (accelerometer sensor), cảm biến trọng lực (gravity sensor), con quay hồi chuyển (gyroscopes sensor), cảm biến vector quay (rotational vector sensor).
- Loại cảm biến môi trường (Environment sensors): đây là các cảm biến đo các thông số môi trường như nhiệt độ, độ ẩm, ánh sáng. Nhóm này bao gồm 1 số cảm biến như: cảm biến đo khí áp (barometer sensor), cảm biến trắc quang (photometer sensor), cảm biến nhiệt kế (thermometer sensor).
- Loại cảm biến vị trí (Position sensors): đây là các cảm biến đo vị trí vật lý của thiết bị. Nhóm này bao gồm gồm 1 số cảm biến như: cảm biến định hướng (orientation sensor) và cảm biến từ kế (magnetometer sensor).

Các cảm biến này có thể dựa trên phần cứng hoặc dựa trên phần mềm. Cảm biến dựa trên phần cứng là những phần cứng vật lý gắn trực tiếp trên thiết bị dùng thu thập dữ liệu trực tiếp từ môi trường xung quanh thiết bị, cảm biến dựa trên phần mềm lấy

được dữ liệu từ 1 hoặc nhiều các cảm biến dựa trên phần cứng hay còn gọi là cảm biến tổng hợp.

Bảng 3.2. Các loại cảm biến

Cảm biến	Loại	Miêu tả	Sử dụng
TYPE_ACCELEROMETER	Cảm biến phần cứng	Đo lực gia tốc của thiết bị theo 3 hướng vật lý (x,y,z), bao gồm cả lực hấp dẫn. Đơn vị μ/σ^2	Phát hiện chuyển động của thiết bị (lắc, nghiêng...)
TYPE_AMBIENT_TEMPERATURE	Cảm biến phần cứng	Đo nhiệt độ môi trường xung quanh thiết bị với đơn vị $^{\circ}\text{C}$	Giám sát nhiệt độ không khí.
TYPE_GRAVITY	Cảm biến phần cứng hoặc phần mềm	Đo trọng lực thiết bị theo cả 3 trục vật lý (x,y,z). Đơn vị μ/σ^2	Phát hiện chuyển động của thiết bị (lắc, nghiêng...)
TYPE_GYROSCOPE	Cảm biến phần cứng	Đo tốc độ quay của thiết bị theo 3 trục vật lý (x,y,z). Đơn vị rad/s.	Phát hiện chuyển động quay của thiết bị.
TYPE_LIGHT	Cảm biến phần cứng	Đo mức độ chiếu sáng của môi trường xung quanh thiết bị. Đơn vị lx.	Kiểm soát độ sáng màn hình thiết bị.
TYPE_LINEAR_ACCELERATION	Cảm biến phần cứng hoặc phần mềm	Đo lực gia tốc của thiết bị theo 3 hướng vật lý (x,y,z), không bao gồm cả lực hấp dẫn. Đơn vị	Theo dõi gia tốc dọc theo 1 trục.
TYPE_MAGNETIC_FIELD	Cảm biến phần cứng	Đo từ trường xung quanh thiết bị theo cả 3 trục vật lý (x,y,z). Đơn vị μT .	Tạo la bàn.
TYPE_ORIENTATION	Cảm biến phần mềm	Đo tốc độ quay của thiết bị theo 3 trục vật lý (x,y,z). Từ Android API level 3 cho phép thu được ma trận nghiêng và ma trận xoay của thiết bị bằng cách sử dụng cảm biến trọng lực và cảm biến từ trường bằng cách sử dụng phương thức <code>getRotationMatrix()</code>	Xác định vị trí thiết bị.

TYPE_PRESSURE	Cảm biến phần cứng	Đo áp suất không khí xung quanh thiết bị. Đơn vị hPa hoặc mbar.	Giám sát sự thay đổi áp suất không khí xung quanh thiết bị.
TYPE_PROXIMITY	Cảm biến phần cứng	Đo khoảng cách của 1 đối tượng với thiết bị. Đơn vị: cm.	Xác định vị trí điện thoại trong khi có cuộc gọi.
TYPE_ROTATION_VECTOR	Cảm biến phần cứng hoặc phần mềm	Đo hướng thiết bị thông qua vector chuyển động quay của thiết bị.	Phát hiện chuyển động quay của thiết bị.
TYPE_TEMPERATURE	Cảm biến phần cứng	Đo nhiệt độ của thiết bị. Đơn vị °C	Giám sát nhiệt độ thiết bị.

Android sensor framework

Android sensor framework là 1 phần của gói android.hardware, bao gồm các lớp và giao tiếp sau:

- **SensorManager:** Được sử dụng để tạo ra thể hiện của dịch vụ cảm biến. Lớp này cung cấp các phương thức cho phép truy cập, đăng ký/ hủy bỏ sự kiện, thu thập thông tin định hướng các cảm biến trên thiết bị. Đồng thời lớp này cũng cho phép báo cáo, thiết lập và hiệu chỉnh các thông số cảm biến.
- **Sensor:** Được sử dụng để tạo ra thể hiện của 1 cảm biến cụ thể. Lớp này cung cấp các phương thức khác nhau cho phép xác định khả năng của cảm biến.
- **SensorEvent:** Được sử dụng để tạo ra đối tượng sự kiện cảm biến nhằm cung cấp các thông tin về sự kiện cảm biến. Đối tượng sự kiện cảm biến cung cấp các thông tin như: dữ liệu cảm biến thô, loại cảm biến tạo ra sự kiện, tính chính xác của dữ liệu và các mốc thời gian xuất hiện sự kiện.
- **SensorEventListener:** Giao tiếp này cung cấp 2 phương thức callback cho phép nhận thông báo từ sự kiện cảm biến khi giá trị cảm biến thay đổi hoặc khi độ chính xác cảm biến thay đổi.

Thông thường ứng dụng sử dụng các API cảm biến để thực hiện 2 nhiệm vụ: 1/ Xác định các cảm biến trên thiết bị và khả năng của các cảm biến đó; 2/ Theo dõi các sự kiện cảm biến nhằm thu thập dữ liệu cảm biến.

Xác định các cảm biến trên thiết bị và khả năng của các cảm biến đó

• Để xác định các cảm biến có trên thiết bị, đầu tiên cần tham chiếu tới dịch vụ cảm biến bằng cách tạo ra thể hiện của lớp **SensorManager** (bằng cách gọi **getSystemService()** với tham số truyền vào là **SENSOR_SERVICE**). Cách làm như sau:

```
private SensorManager mSensorManager;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
```

- Lấy về danh sách các cảm biến có trên thiết bị bằng cách gọi phương thức `getSensorList()` với tham số truyền vào là `Sensor.TYPE_ALL` như sau:

```
List<Sensor> deviceSensors = mSensorManager.getSensorList(Sensor.TYPE_ALL);
```

Để lấy về danh sách các cảm biến theo chủng loại với tham số truyền vào của phương thức `getSensorList` là `TYPE_GYROSCOPE`, `TYPE_LINEAR_ACCELERATION`, `TYPE_GRAVITY`...

Để kiểm tra 1 loại cảm biến có tồn tại trên thiết bị không thì dùng phương thức `getDefaultSensor()` với tham số truyền vào là kiểu của loại cảm biến đó.

```
private SensorManager mSensorManager;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
if (mSensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD) != null) {
    // Success! There's a magnetometer.
}
else {
    // Failure! No magnetometer.
}
```

- Xác định khả năng của các cảm biến trên thiết bị:

Lớp `Sensor` cung cấp 1 số phương thức cho phép xác định khả năng của các cảm biến trên thiết bị, như `getResolution()` lấy về độ phân giải cảm biến, `getMaximumRange()` lấy về phạm vi tối đa đo lường của cảm biến, `getPower()` lấy về yêu cầu nguồn cho cảm biến...

```

private SensorManager mSensorManager;
private Sensor mSensor;

...

mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);

if (mSensorManager.getDefaultSensor(Sensor.TYPE_GRAVITY) != null) {
    List<Sensor> gravSensors = mSensorManager.getSensorList(Sensor.TYPE_GRAVITY);
    for(int i=0; i<gravSensors.size(); i++) {
        if ((gravSensors.get(i).getVendor().contains("Google Inc.)) &&
            (gravSensors.get(i).getVersion() == 3)){
            // Use the version 3 gravity sensor.
            mSensor = gravSensors.get(i);
        }
    }
}
else{
    // Use the accelerometer.
    if (mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER) != null){
        mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    }
    else{
        // Sorry, there are no accelerometers on your device.
        // You can't play this game.
    }
}
}

```

Theo dõi các sự kiện cảm biến

Để theo dõi dữ liệu cảm biến, ứng dụng cần cài đặt hai phương thức callback của giao tiếp `SensorEventListener`: `onAccuracyChanged()` và `onSensorChanged()`. Hai phương thức này được hệ thống gọi bất cứ khi nào giá trị cảm biến hoặc độ chính xác cảm biến thay đổi.

- Theo dõi sự thay đổi độ chính xác của cảm biến thông qua đối tượng `Sensor` trong phương thức `onAccuracyChanged()`. Độ chính xác cảm biến được biết đến thông qua 4 hằng số trạng thái: `SENSOR_STATUS_ACCURACY_LOW`, `SENSOR_STATUS_ACCURACY_MEDIUM`, `SENSOR_STATUS_ACCURACY_HIGH`, `SENSOR_STATUS_UNRELIABLE`.
- Theo dõi sự thay đổi giá trị cảm biến thông qua đối tượng `SensorEvent` trong phương thức `onSensorChanged()`. Đối tượng `SensorEvent` cung cấp 1 số thông tin về dữ liệu cảm biến như: tính chính xác của dữ liệu, cảm biến tạo ra dữ liệu là gì, mốc thời gian dữ liệu được sinh ra và các dữ liệu mới được cảm biến ghi lại.

Ví dụ dưới đây minh họa cách theo dõi dữ liệu của cảm biến ánh sáng thông qua phương thức `onSensorChanged()`:

```

public class SensorActivity extends Activity implements
    SensorEventListener
{

```

```

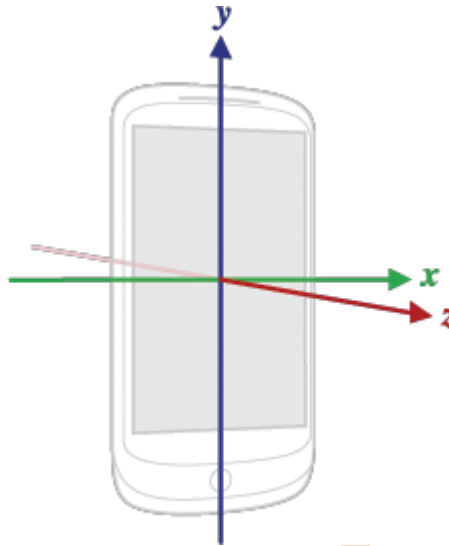
private SensorManager mSensorManager;
private Sensor mLight;
@Override
public final void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    mSensorManager = (SensorManager)
        getSystemService(Context.SENSOR_SERVICE);
    mLight = mSensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
}
@Override
public final void onAccuracyChanged(Sensor sensor, int accuracy)
{
    // Do something here if sensor accuracy changes.
}
@Override
public final void onSensorChanged(SensorEvent event) {
    // The light sensor returns a single value.
    // Many sensors return 3 values, one for each axis.
    float lux = event.values[0];
    // Do something with this sensor value.
}
@Override
protected void onResume() {
    super.onResume();
    mSensorManager.registerListener(this, mLight,
        SensorManager.SENSOR_DELAY_NORMAL);
}
@Override
protected void onPause() {
    super.onPause();
    mSensorManager.unregisterListener(this);
}
}

```

Chú ý 2 phương thức callback `onResume()` và `onPause()` dùng để đăng ký/ hủy bỏ đăng ký lắng nghe sự kiện cảm biến.

3.4.2. Giám sát hướng và chuyển động của thiết bị

Android sensor framework sử dụng hệ tọa độ 3 chiều tiêu chuẩn (x,y,z) để thể hiện giá trị dữ liệu thu thập qua cảm biến, trong đó: trục x là trục ngang với giá trị bên phải, trục y là trục thẳng đứng với giá trị phía trên, trục z là trục ngoài khuôn màn hình. Hệ trục tọa độ có biểu diễn như sau:



Hình 3.12. Hệ trục tọa độ sử dụng bởi Android sensor API

Một số cảm biến sử dụng hệ trục tọa độ này để thu thập dữ liệu: Acceleration sensor, Gravity sensor, Gyroscope sensor, Linear acceleration sensor, Geomagnetic field sensor.

Chú ý:

- Hệ trục tọa độ này không thay đổi khi thiết bị di chuyển.
- Hệ tọa độ cảm biến luôn căn cứ trên định hướng tự nhiên của thiết bị, hầu hết các thiết bị có định hướng tự nhiên theo chiều ngang (landscape).
- Sử dụng phương thức `getRotation()` để xác định chiều xoay của màn hình, `remapCoordinateSystem()` để chuyển hệ trục tọa độ cảm biến về hệ trục tọa độ màn hình.

Các thông tin thu được qua cảm biến chuyển động trên thiết bị được lưu trữ trong mảng số thực đa chiều của mỗi `SensorEvent`. Ví dụ khi sự kiện cảm biến gia tốc xuất hiện sẽ trả về dữ liệu lực gia tốc theo 3 chiều tọa độ.

Lấy về một thể hiện của cảm biến gia tốc

```
private SensorManager mSensorManager;
private Sensor mSensor;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

Về mặt khái niệm, gia tốc thiết bị thu được thông qua cảm biến gia tốc (A_d) đo được thông qua trọng lực lên cảm biến (F_s) như sau:

$$A_d = - \sum F_s / mass$$

Tuy nhiên khi gia tốc thiết bị chịu ảnh hưởng của lực hấp dẫn thì:

$$A_d = -g - \sum F / mass$$

Khi thiết bị di động được đặt yên trên bàn (không có gia tốc): $g = 9.81 \text{ m/s}^2$, khi thiết bị rơi tự

do, $g = 0 \text{ m/s}^2$. Do đó, để đo gia tốc thực của thiết bị, cần loại bỏ lực hấp dẫn khỏi dữ liệu gia tốc thu được, điều này được thực hiện bằng cách sử dụng một bộ lọc để cô lập các lực hấp dẫn. Minh họa dưới đây thực hiện điều này:

```
public void onSensorChanged(SensorEvent event) {
    // In this example, alpha is calculated as t / (t + dT),
    // where t is the low-pass filter's time-constant and
    // dT is the event delivery rate.

    final float alpha = 0.8;

    // Isolate the force of gravity with the low-pass filter.
    gravity[0] = alpha * gravity[0] + (1 - alpha) * event.values[0];
    gravity[1] = alpha * gravity[1] + (1 - alpha) * event.values[1];
    gravity[2] = alpha * gravity[2] + (1 - alpha) * event.values[2];

    // Remove the gravity contribution with the high-pass filter.
    linear_acceleration[0] = event.values[0] - gravity[0];
    linear_acceleration[1] = event.values[1] - gravity[1];
    linear_acceleration[2] = event.values[2] - gravity[2];
}
```

Lưu ý: có nhiều kỹ thuật khác nhau được sử dụng để lọc dữ liệu cảm biến. Đoạn code phía trên lọc dữ liệu cảm biến đơn giản thông qua hằng $\alpha = 0.8$ minh họa.

Thu thập thông tin từ các cảm biến khác thuộc loại cảm biến chuyển động được thực hiện tương tự như cảm biến gia tốc. Thông tin chi tiết:

http://developer.android.com/guide/topics/sensors/sensors_motion.html#sensors-motion-accel

CÂU HỎI CHƯƠNG 3

1. Xây dựng ứng dụng kết nối mạng thông qua wifi có các chức năng: 1/ Kích hoạt kết nối wifi; 2/ Lấy về thông tin kết nối wifi hiện thời; 3/ Quyetts những điểm truy cập wifi xung quanh thiết bị; 4/ Thiết lập cấu hình cho mạng wifi mà thiết bị kết nối tới.
2. Xây dựng ứng dụng kết nối mạng thông qua bluetooth có các chức năng: 1/ Bật bluetooth trên thiết bị; 2/ Giám sát trạng thái bluetooth của thiết bị; 3/ Thăm dò các thiết bị thông qua kết nối bluetooth; 4/ Truyền dữ liệu giữa 2 thiết bị hỗ trợ bluetooth.
3. Xây dựng ứng dụng nhắn tin ngắn SMS với các chức năng: 1/ Gửi tin nhắn SMS thông qua ứng dụng tin nhắn có sẵn trên điện thoại; 2/ Gửi tin nhắn trong chính ứng dụng; 3/ Nhận tin nhắn trong chính ứng dụng.
4. Xây dựng ứng dụng phát audio, video.
5. Xây dựng ứng dụng camera theo 2 cách: 1/ Từ ứng dụng camera có sẵn trên thiết bị; 2/ Xây dựng ứng dụng camera không phụ thuộc vào ứng

dụng camera có sẵn trên thiết bị.

6. Xây dựng ứng dụng thu thập thông tin từ các cảm biến trên thiết bị tương ứng với 3 loại cảm biến : 1/ Cảm biến chuyển động; 2/ Cảm biến vị trí; 3/ Cảm biến môi trường.

PDF

CHƯƠNG 4: PHÁT TRIỂN ỨNG DỤNG ĐA NỀN TẢNG

Trong chương 2 và chương 3 của bài giảng tập trung vào trình bày về lập trình ứng dụng di động trên một nền tảng di động cụ thể là Android sử dụng ngôn ngữ Java. Trên thực tế chúng ta biết rằng có rất nhiều nền tảng di động đang tồn tại và mỗi nền tảng sử dụng một ngôn ngữ riêng. Điều này gây mất nhiều thời gian phát triển, nguồn lực và phức tạp trong việc bảo trì ứng dụng. Vậy làm thế nào để ứng dụng di động của chúng ta chỉ viết một lần nhưng có khả năng chạy trên các nền tảng khác nhau là vấn đề đang được cộng đồng lập trình viên rất quan tâm. Nội dung chương 4 này tập trung trình bày về các kiến thức lập trình ứng dụng di động đa nền tảng. Nội dung trình bày bao gồm:

- Khái niệm, phân loại ứng dụng di động đa nền tảng.
- Giới thiệu một số framework phát triển ứng dụng di động đa nền tảng.
- Xây dựng ứng dụng di động đa nền tảng sử dụng một framework (trong phạm vi bài giảng sẽ tập trung trình bày về framework lập trình đa nền tảng có tên là Mono).

4.1. ĐỊNH NGHĨA, PHÂN LOẠI ỨNG DỤNG DI ĐỘNG ĐA NỀN TẢNG

4.1.1. Định nghĩa ứng dụng di động đa nền tảng

Ứng dụng di động đa nền tảng (cross-platform mobile apps) là thuật ngữ để chỉ những ứng dụng di động được thiết kế chạy trên nhiều nền tảng di động khác nhau như iOS, Android, Windows Phone, BlackBerry, WebOS,... mà không cần phải lập trình nhiều lần cho từng nền tảng.

Ưu nhược điểm của việc phát triển ứng dụng di động đa nền tảng:

Ưu điểm:

- Ứng dụng có thể tiếp cận được nhiều khách hàng trên các nền tảng hơn.
- Tiết kiệm được nhiều thời gian phát triển và khiến cho ứng dụng dễ duy trì và nâng cấp đồng bộ trên các nền tảng.
- Việc thiết kế và cảm nhận của ứng dụng thống nhất trên các nền tảng.
- Sử dụng lại được các nguồn lực và công nghệ có sẵn, thống nhất.
- Tất cả các yếu tố trên cũng đồng nghĩa với chi phí phát triển ứng dụng sẽ giảm đi.

Nhược điểm:

- Có rất nhiều công cụ phát triển ứng dụng di động đa nền tảng như Unity, Ramp, PhoneGap, Grapple, Open Plug, Rhomobile, Titanium, Xamarin... Tuy vậy, việc thiết kế các ứng dụng theo ý riêng của khách hàng sẽ có thể gặp khó khăn trong việc đối phó với sự khác biệt giữa các công cụ và ngôn ngữ trong mỗi nền tảng khác nhau.
- Thiết kế giao diện tương tác người dùng phù hợp với phong cách riêng trên nhiều nền tảng là một công việc không hề đơn giản.
- Việc lưu trữ dữ liệu trên các nền tảng khác nhau rất đa dạng. Đồng thời

việc tích hợp các ứng dụng với thiết lập cục bộ, ưu tiên và khai báo cũng có thể sẽ phải đối mặt với những rắc rối trong việc sắp xếp nhiều nền tảng.

- Mỗi nền tảng đều cung cấp sự linh hoạt riêng, đó là lý do vì sao chúng có mặt trên thị trường. Khi thiết kế một ứng dụng đa nền tảng thì phải nhìn vào sự tương đồng khiến cho mất đi phần nào sự linh hoạt trên mỗi nền tảng.

4.1.2. Phân loại ứng dụng di động đa nền tảng

Hiện tại có 2 giải pháp thiết kế ứng dụng di động đa nền tảng là ứng dụng web được thiết kế cho di động (web-apps) và ứng dụng được xây dựng dành riêng cho nền tảng mục tiêu (mobile native apps).

Ứng dụng web được thiết kế cho di động (web-apps)

Web apps được xây dựng trên nền web với các ngôn ngữ web như HTML, Javascript,... Các ứng dụng dạng này được triển khai tại 1 nơi gọi là web server và các thiết bị di động chạy bất kỳ nền tảng nào có thể kết nối tới nó thông qua trình duyệt web mà không cần sự chuẩn bị thêm đặc biệt nào.

Ứng dụng được xây dựng dành riêng cho nền tảng mục tiêu (native apps)

Native app là ứng dụng được viết để chạy trên từng nền tảng riêng biệt (iOS, android, window phone, RIM-OS ...) và trên từng thiết bị khác nhau để cùng thực hiện một chức năng cụ thể như: phần mềm nghe nhạc, lịch, hay các ứng dụng mạng xã hội ... và các trò chơi trên thiết bị di động. Ví dụ như Apple cho ra Apple App Store – kho lưu trữ phần mềm trực tuyến, RIM cho ra đời kho lưu trữ BB App World – kho lưu trữ phần mềm trực tuyến cho điện thoại BlackBerry, và Google với nền tảng Android cho ra Google Play.

Native app được tải từ các kho ứng dụng di động và cài đặt trực tiếp lên các thiết bị di động. Native apps có lợi thế hơn web app là được xây dựng như một ứng dụng nền của hệ điều hành mục tiêu (iOS, Android,...) và có thể hoạt động cả khi không có kết nối internet.

Giao diện của native app được thiết kế ấn tượng, đáp ứng được nhu cầu của người sử dụng. Tuy vậy, native app phụ thuộc nhiều vào hệ điều hành cũng như phần cứng của thiết bị (cpu, rom, ram, màn hình hiển thị ...), tính đa phương tiện chưa được cao, ví dụ như Adobe Flash chỉ hỗ trợ thiết bị có RIM OS7 trở lên, Android, iOS 2.1 trở lên ... Font ứng dụng còn hạn chế, chưa thể hỗ trợ tối đa các hệ điều hành và còn nhiều khó khăn trong việc chia sẻ ứng dụng giữa 2 hệ điều hành có nền tảng khác nhau như giữa iOS và Android, iOS và Blackberry. Native app có thể yêu cầu cấu hình phần cứng để chạy trên thiết bị một cách hiệu quả, đồng thời người phát triển native app phải chờ đợi sự kiểm duyệt của nhà cung cấp để có thể đưa ứng dụng của mình lên những kho lưu trữ trực tuyến như AppStore, GooglePlay hay Windows phone store... Nhưng khó khăn đáng kể nhất ở đây, một native app chỉ có thể chạy trên một hệ điều hành nhất định, do đó với mỗi hệ điều hành thì người lập trình phải viết native code riêng cho nó, dẫn tới việc không đồng nhất giữa các phiên bản ứng dụng, chi phí phát triển cao và đòi hỏi sự thành

thạo nhiều ngôn ngữ ở lập trình viên, đôi khi phải sử dụng công cụ độc quyền và chỉ làm việc trên được một môi trường duy nhất như Xcode trên MacOS

4.2. GIỚI THIỆU CÁC KHUNG LÀM VIỆC (FRAMEWORK) PHÁT TRIỂN ỨNG DỤNG ĐA NỀN TẢNG

Vấn đề đặt ra ở đây là cần những kỹ thuật giúp chúng ta có thể phát triển ứng dụng đa nền tảng, chỉ cần phát triển ứng dụng 1 lần, sau đó sẽ sử dụng được cốt lõi của ứng dụng đó trên nhiều nền tảng hệ điều hành khác nhau. Hiện nay có khá nhiều kỹ thuật có thể giúp ta phát triển ứng dụng đa nền tảng, có thể kể ra một số như : PhoneGap, Titanium mobile, Corona SDK, Mono, Xamarin, Unity3D, Cocos2D-x...

- PhoneGap: PhoneGap là một mã nguồn mở cho phép các nhà phát triển ứng dụng di động đa nền tảng cho nhiều loại thiết bị và hệ điều hành di động bao gồm cả iPhone, Android, Windows Phone, BlackBerry, Symbian, Bada và webOS, bằng cách sử dụng ngôn ngữ như HTML5, JavaScript and CSS3, thay vì dùng những ngôn ngữ phức tạp như iOS Objective-C và một số loại khác.
- Appcelerator Titanium: Appcelerator Titanium là một nền tảng cho phép phát triển những ứng dụng native đa nền tảng. Ngày nay, đối với các doanh nghiệp, nền tảng này đã trở nên quan trọng trên tất cả mọi nơi – từ trực tuyến, điện thoại đến máy tính bảng và máy tính bàn. Appcelerator Titanium giúp các nhà phát triển tạo ra các ứng dụng đầy đủ tính năng, vượt ra mọi ranh giới cũng như luôn kết nối với người dùng bất kể khi nào và đang ở đâu
- MoSync: Kết hợp sức mạnh tính toán của C++, JavaScript and HTML5, MoSync là một SDK cho phép các nhà phát triển thiết kế và phát triển phong phú hơn các giải pháp đa nền tảng tương thích cho iOS, Android và các thiết bị di động khác.
- Mono: Đây là framework cho phép phát triển ứng dụng native cho iOS, Android windows, Windows phone sử dụng ngôn ngữ C#. Framework này cho phép lập trình viên tạo ra giao diện ứng dụng đặc thù riêng cho mỗi nền tảng, đồng thời sử dụng lại mã ứng dụng chia sẻ dùng chung cho các nền tảng đó.

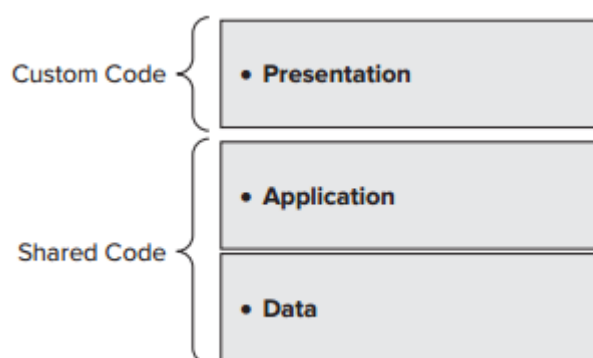
4.3. XÂY DỰNG MỘT ỨNG DỤNG ĐA NỀN TẢNG SỬ DỤNG MỘT FRAMEWORK

Trong phần này bài giảng sẽ trình bày về xây dựng ứng dụng di động đa nền tảng dạng native. Framework sử dụng là Mono. Lợi điểm của framework này là cho phép xây dựng ứng dụng đa nền tảng với giao diện ứng dụng vẫn mang phong cách đặc thù riêng trên mỗi nền tảng, trong khi đó sử dụng lại các mã nguồn logic chung cho các nền tảng đó mà không phải viết lại nhiều lần.

	iOS	Android	Windows Phone
Runtime	Mono	Mono	.NET
Business logic	C#	C#	C#
User interface	MonoTouch	Mono for Android	Silverlight

Hình 4.1. Giải pháp xây dựng ứng dụng đa nền tảng của Mono framework

Giải pháp phát triển ứng dụng di động đa nền tảng của Mono là cho phép lập trình viên tạo ra giao diện ứng dụng đặc thù riêng cho mỗi nền tảng di động, kết hợp với việc sử dụng lại những mã ứng dụng chia sẻ dùng chung cho các nền tảng đó (như mã ứng dụng xử lý logic, dữ liệu).



Hình 4.2. Kiến trúc chia sẻ mã nguồn của Mono framework

4.3.1. Thiết lập môi trường lập trình ứng dụng đa nền tảng

Thiết lập môi trường lập trình ứng dụng di động iOS trên nền tảng Mono

Cài đặt những gói phần mềm sau trên Mac OSX:

- Xcode4 : IDE của Apple bao gồm một số công cụ như: iOS simulator, công cụ thiết kế giao diện ứng dụng, SDK cho phép phát triển ứng dụng trên Mac OS X và iOS. Gói này được download từ OS X App Store hoặc Apple's iOS Dev Center. Trong khi phát triển ứng dụng cho iOS sử dụng giải pháp đa nền tảng Mono tuy không sử dụng trực tiếp Xcode nhưng việc cài đặt nó là cần thiết.
- MonoDevelop: IDE cung cấp môi trường phát triển ứng dụng iOS tương tự như Microsoft's Visual Studio.
- MonoTouch: plug-in của MonoDevelop cho phép phát triển ứng dụng iOS sử dụng ngôn ngữ C#.

MonoDevelop và MonoTouch được download từ website của Xamarin:

<http://docs.xamarin.com/ios>

Thiết lập môi trường lập trình ứng dụng di động Android trên nền tảng Mono

Cài đặt những gói phần mềm sau trên Windows hoặc Mac OSX:

- Java JDK
- Android SDK
- MonoDevelop hoặc Visual studio 2010

- Mono for Android: plug-in của MonoDevelop hoặc Visual Studio cho phép phát triển ứng dụng Android sử dụng ngôn ngữ C#.

Các gói phần mềm này được download từ : <http://docs.xamarin.com/android>

Thiết lập môi trường lập trình ứng dụng di động cho Windows phone trên nền tảng Mono

Cài đặt gói phần mềm sau trên Windows:

- Visual studio 2017
- Windows phone SDK

Các gói phần mềm này được download từ: <http://create.msdn.com/>

4.3.2. Các kỹ thuật chia sẻ mã nguồn đa nền tảng

Nội dung phần này trình bày về cách tạo ra thư viện trên các nền tảng và các kỹ thuật chia sẻ mã nguồn đa nền tảng.

Tạo thư viện trên các nền tảng

Thư viện chia sẻ bao gồm tập hợp các mã nguồn tạo ra độc lập, thực thi logic chung của ứng dụng. Các ứng dụng trên các nền tảng khác nhau sẽ tham chiếu tới thư viện dùng chung này để sử dụng các chức năng đã được xây dựng mà không phải viết lại mã nguồn cho nó.

Các bước thực hiện như sau:

Tạo thư mục SharedLibrary để chứa những project thư viện chia sẻ. Khi phát triển ứng dụng từ môi trường Windows và Mac OS X cho cả 3 nền tảng di động Android, Windows Phone, iOS thì cần đảm bảo đặt thư mục này tại nơi được phép truy cập bởi cả 2 hệ thống Windows và Mac OS X.

Tạo ra project với tên SharedLibrary trong thư mục SharedLibrary để chứa code chia sẻ đa nền tảng, bằng cách:

- Mở Visual Studio (trên môi trường Windows), chọn New Project -> Windows -> Class Library, đặt tên cho project là SharedLibrary.
- - Hoặc mở MonoDevelop (trên môi trường Mac OS X), chọn New Project -> C# -> Library in MonoDevelop, đặt tên cho project là SharedLibrary.

Loại bỏ bất cứ 1 lớp nào mặc định được thêm vào project để đảm bảo project hoàn toàn trống.

Tạo các project thư viện độc lập trên mỗi nền tảng trong thư mục SharedLibrary.

- Tạo thư viện cho ứng dụng iOS: Mở MonoDevelop (trên môi trường Mac OS X), chọn new solution -> MonoTouch -> MonoTouch Library Project, đặt tên là SharedLibrary.MonoTouch. Loại bỏ bất cứ 1 lớp nào mặc định được thêm vào project để đảm bảo project hoàn toàn trống.
- Tạo thư viện cho ứng dụng Android: Mở Visual Studio (trên môi trường Windows) hoặc MonoDevelop (trên môi trường Mac OS X) chọn new solution -> Mono for Android -> Mono for Android Library Project, đặt

tên là SharedLibrary.MonoAndroid. Loại bỏ bất cứ 1 lớp nào mặc định được thêm vào project để đảm bảo project hoàn toàn trống.

- Tạo thư viện cho ứng dụng Windows Phone: Mở Visual Studio (trên môi trường Windows), chọn new solution -> Silverlight for Windows Phone -> Windows Phone Class Library, đặt tên là SharedLibrary.WindowsPhone. Loại bỏ bất cứ 1 lớp nào mặc định được thêm vào project để đảm bảo project hoàn toàn trống.

Các kỹ thuật chia sẻ mã nguồn đa nền tảng

Để thuận tiện cho việc trình bày, giả sử yêu cầu đặt ra là xây dựng ứng dụng theo dõi chỗ đậu xe. Ứng dụng này cho phép xác định vị trí hiện thời đậu xe của người dùng và đưa ra thông báo khi người dùng trở lại cách vị trí đó 1 khoảng cách nhất định. Nội dung trình bày dưới đây sẽ không đi xây dựng hoàn thiện ứng dụng mà tập trung vào việc làm thế nào để áp dụng các kỹ thuật chia sẻ đa nền tảng vào việc viết code logic cho ứng dụng yêu cầu.

File Linking

Phương pháp đơn giản nhất để chia sẻ mã nguồn cho các nền tảng là tạo ra bản sao riêng biệt của mỗi tập tin trên các nền tảng đó, nhưng cách làm này sẽ là một phương pháp tồi khi chúng ta muốn duy trì sự đồng bộ của các bản sao.

Để khắc phục nhược điểm này, kỹ thuật chia sẻ mã nguồn đa nền tảng “File Linking” được đưa ra. Đây là kỹ thuật chia sẻ mã nguồn trên các nền tảng bằng cách duy trì một bản sao file duy nhất, các project trên các nền tảng khác nhau liên kết tới file này và biên dịch qua những project trên các nền tảng khác nhau. Về bản chất, khi liên kết một file xác định vào một project, file đó vẫn chỉ tồn tại ở nơi nó được tạo ra, nhưng vẫn có vai trò giúp project đó hoạt động bình thường. Khi cần cập nhật file mã nguồn chia sẻ, chỉ cần tìm file đó chỉnh sửa và việc chỉnh sửa đó sẽ có tác dụng đến bất kỳ project liên kết tới file đó.

Minh họa áp dụng các kỹ thuật chia sẻ đa nền tảng vào việc viết code logic cho ứng dụng yêu cầu trên theo các bước sau:

- Project SharedLibrary được coi là nguồn nơi chứa các tập tin, được liên kết vào các lớp thư viện khác nhau cho iOS, Android, Windows Phone. Tạo ra một thư mục có tên Chapter3 trong project SharedLibrary và tạo ra một class mới có tên ParkingSpotTracker
- Mở SharedProject.MonoTouch solution (trên môi trường Mac OS X), tạo ra folder Chapter3. Nhấp chuột phải vào folder Chapter3, chọn Add -> Add Files, và chọn ParkingSpotTracker.cs từ project thư viện nguồn. MonoDevelop sẽ yêu cầu lập trình viên chọn cách thêm file vào project bằng cách đưa ra các lựa chọn. Ở đây ta sẽ chọn “Add a link to the file” để liên kết file trong project nguồn SharedLibrary vào project SharedProject.MonoTouch.
- Mở SharedProject.WindowsPhones trong Visual Studio và tạo ra folder Chapter3. Nhấp chuột phải vào folder, chọn Add -> Existing Item và

chọn `ParkingSpotTracker.cs` từ project thư viện nguồn. Tại nút Add, ấn mũi tên xuống chọn Add as Link nhằm liên kết file trong project nguồn `SharedLibrary` vào project `SharedProject.WindowsPhones` như làm ở `MonoDevelop`. Điều này được làm tương tự với `SharedProject.MonoAndroid` sử dụng Visual Studio hoặc `MonoDevelop`.

Abstraction

Trong thực tế mỗi nền tảng hoạt động theo một cách riêng, nhưng vẫn có những điểm chung tồn tại giữa chúng. Mã cài đặt các chức năng của ứng dụng trên mỗi nền tảng là khác nhau, mặc dù kết quả trả về khi thực thi các chức năng trên các nền tảng là như nhau nhưng không thể dùng chung mã nguồn này để chia sẻ giữa các nền tảng (Ví dụ: mỗi nền tảng khác nhau có những API riêng để lấy về vị trí hiện thời của thiết bị). Vậy làm thế nào để chia sẻ mã nguồn giữa các nền tảng trong trường hợp này?

Trong hầu hết các trường hợp, lớp nghiệp vụ của ứng dụng không cần quan tâm đến cài đặt cụ thể của chức năng trong ứng dụng mà chỉ quan tâm tới đầu vào và đầu ra của dữ liệu để thực thi chức năng đó. Để làm được việc này kỹ thuật chia sẻ mã nguồn đa nền tảng “Abstraction” được đưa ra.

Về bản chất kỹ thuật “Abstraction” tiến hành trừu tượng hóa dữ liệu và chức năng ứng dụng bằng cách xác định và nhóm các thuộc tính và hành động liên quan đến một thực thể đặc thù trong ứng dụng, làm dịch vụ thay đổi theo sự thay đổi của dữ liệu.

Trong C#, điều này được biết đến là interface. Interface đưa ra định nghĩa chung nhất về các thuộc tính, phương thức của lớp thực thể mà không có cài đặt cụ thể. Các lớp thực thi interface phải tiến hành cài đặt cụ thể cho các phương thức đã định nghĩa trong interface. Một lớp có thể thực thi nhiều interface.

Với việc sử dụng kỹ thuật “Abstraction” sẽ cho phép tách biệt phần xử lý logic chung của ứng dụng trong lớp chia sẻ mà không phụ thuộc vào nền tảng nào, việc thực thi những chức năng này sẽ được cài đặt cụ thể trên nền tảng liên kết tới lớp chia sẻ này.

Minh họa sử dụng kỹ thuật này trong việc xây dựng ứng dụng yêu cầu ở trên như sau:

Đối với các ứng dụng theo dõi chỗ đậu xe, chức năng định vị địa lý được trừu tượng vào lớp chia sẻ. Trong thư mục `Chapter3` của `SharedLibrary`, thêm một lớp mới có tên `LocationInfo` sẽ lưu trữ vĩ độ và kinh độ của một điểm cụ thể trên bản đồ và chức năng tính khoảng cách giữa giữa 2 điểm (để minh họa chức năng này trả về một giá trị ví dụ). Tiếp theo, định nghĩa một interface có tên `ILocationProvider` chứa một phương pháp duy nhất, `GetCurrentLocation ()`, trả về một đối tượng `LocationInfo` đại diện cho vị trí hiện tại của người dùng. Cuối cùng, viết code cho `ParkingSpotTracker.cs`, hàm khởi tạo của lớp `ParkingSpotTracker` có tham số truyền vào là 1 thể hiện của `ILocationProvider`, thêm một phương thức có tên `ParkHere ()` vào lớp này để việc kiểm tra vị trí hiện tại.

```
// LocationInfo.cs
namespace SharedLibrary.Chapter3
{
```

```

public class LocationInfo
{
    public double Latitude { get; private set; }
    public double Longitude { get; private set; }
    public LocationInfo (double latitude, double longitude)
    {
        Latitude = latitude;
    }
    public double DistanceInMetersFrom(LocationInfo point)
    {
        return 42;
    }
}

// ILocationProvider.cs
namespace SharedLibrary.Chapter3
{
    public interface ILocationProvider
    {
        LocationInfo GetCurrentLocation();
    }
}

// ParkingSpotTracker.cs
using System;
namespace SharedLibrary.Chapter3
{
    public class ParkingSpotTracker
    {
        private readonly ILocationProvider _locationProvider;
        public ParkingSpotTracker(ILocationProvider locationProvider)
        {
            _locationProvider = locationProvider;
        }
        public void ParkHere()
        {
            var currentLocation = _locationProvider.GetCurrentLocation();
            // save a new parking spot using currentLocation
        }
    }
}

```

Tiếp theo tiến hành liên kết các file được tạo ra ở đây với các thư viện cho mỗi nền tảng như kỹ thuật “File Linking” đã trình bày ở trên. Việc thực thi những chức năng trong interface `ILocationProvider` sẽ được cài đặt cụ thể trên nền tảng liên kết tới lớp chia sẻ này.

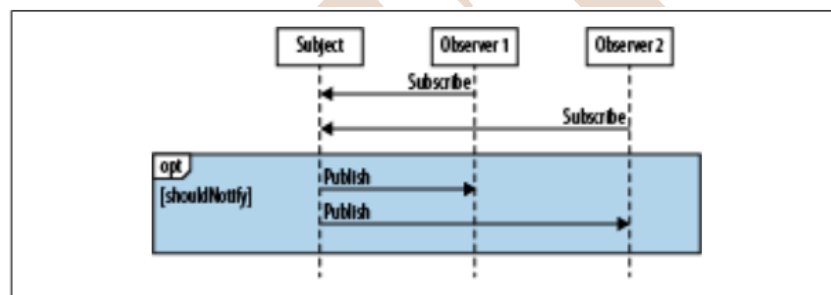
Observer Pattern

Khi thiết kế một ứng dụng, bạn muốn ứng dụng của bạn có thể đáp ứng bất kì phát sinh nào từ lớp chia sẻ. Tuy nhiên vì lớp chia sẻ được tạo ra không xác định nền tảng cụ thể, điều này làm cho việc kết nối trực tiếp ngày trở nên khó khăn.

Một lựa chọn được đưa ra là sử dụng kỹ thuật “Abstraction” ở trên nhằm trừu tượng hóa đưa ra xử lý giống nhau cho các những chức năng tương tác với người dùng trên giao diện, ví dụ đưa ra hộp thoại thông báo hay chỉ thị tiến độ. Việc làm này không phải lúc nào cũng phù hợp với bất kỳ chức năng giao diện người dùng trong ứng dụng chạy trên các nền tảng khác nhau, vì khi đó logic nghiệp vụ ứng dụng sẽ phụ thuộc vào giao diện người dùng (ví dụ ứng dụng chạy trên Windows phone và Android đều cùng đưa hộp thoại thông báo khi thực thi 1 chức năng trong ứng dụng, trong khi điều này có thể cần thiết khi ứng dụng chạy trên Windows phone nhưng không cần khi ứng dụng chạy trên Android). Để giải quyết vấn đề này 1 kỹ thuật mới được đưa ra là mô hình quan sát (Observer pattern) nhằm xử lý tách biệt giữa logic nghiệp vụ và giao diện ứng dụng.

Theo mô hình này, một đối tượng đóng vai trò là chủ thể (subject) sẽ phát ra những thông điệp cập nhật về trạng thái của nó tới những nhân tố phụ thuộc (observer). Các observer đưa ra yêu cầu cập nhật và nhận các thông báo từ subject khi có bất kỳ sự thay đổi nào xuất hiện.

Mô hình này thường được dùng phổ biến khi xử lý các sự kiện liên quan tới giao diện người dùng, ví dụ xử lý sự kiện khi nút bấm trên giao diện được nhấn.



Hình 4.3. Quá trình xuất bản và đăng ký sự kiện trong mô hình quan sát

Mình họa sử dụng kỹ thuật này trong việc xây dựng ứng dụng yêu cầu trên với chức năng kiểm tra và thông báo khi người dùng trở lại cách vị trí đậu xe 1 khoảng cách nhất định:

- Tạo lớp SpotDetectedNearbyEventArgs, lớp này xác định khoảng cách giữa người dùng và địa điểm đậu xe:

```

// SpotDetectedNearbyEventArgs.cs
using System;
namespace SharedLibrary.Chapter3
{
    public class SpotDetectedNearbyEventArgs : EventArgs
    {
        public double Distance { get; private set; }
        public SpotDetectedNearbyEventArgs(double distance)
        {
            Distance = distance;
        }
    }
}

```

Cập nhật mã nguồn cho ParkingSpotTracker.cs, trong đó khai báo sự kiện

SpotDetectedNearby để khi có bất kỳ sự thay đổi nào ở class SpotDetectedNearbyEventArgs thì event SpotDetectedNearby sẽ thông báo tương ứng lại cho class PackingSpotTracker.

```
// ParkingSpotTracker.cs (updates only)
using System;

namespace SharedLibrary.Chapter3
{
    public class ParkingSpotTracker
    {
        // only including updated parts to this class
        public event EventHandler<SpotDetectedNearbyEventArgs>
            SpotDetectedNearby;
        private void checkCurrentLocation()
        {
            var currentLocation =
                _locationProvider.GetCurrentLocation();
            var newYorkCity = new LocationInfo(40.716667, -74);
            double distance =
                currentLocation.DistanceInMetersFrom(newYorkCity);
            if (distance < 100 && SpotDetectedNearby != null)
            {
                SpotDetectedNearby(this, new
                    SpotDetectedNearbyEventArgs(distance));
            }
        }
    }
}
```

Với việc đưa ra thông báo điểm đậu xe như là việc xuất hiện 1 sự kiện, việc xử lý logic nghiệp vụ ứng dụng trong lớp chia sẻ sẽ không phụ thuộc vào giao diện người dùng và ứng dụng trên mỗi nền tảng sẽ chủ động đưa ra các xử lý sự kiện khi cần.

Partial Classes

Kỹ thuật “Abstraction” hữu ích trong việc cung cấp sự khác biệt trong xử lý chức năng ứng dụng giữa các nền tảng trong khi vẫn cho phép tái sử dụng được nhiều mã nguồn. Tuy nhiên, trong nhiều trường hợp lập trình viên muốn thực hiện mở rộng từ lớp chia sẻ cho 1 nền tảng cụ thể thì việc sử dụng kỹ thuật “Abstraction” thuần túy có vẻ trở nên cồng kềnh.

Một giải pháp khác đưa ra là sử dụng kỹ thuật “Abstraction” để đưa ra lớp chia sẻ và trên mỗi nền tảng cụ thể sẽ liên kết và kế thừa để mở rộng lớp chia sẻ này. Tuy nhiên trong một số trường hợp, giải pháp này có thể trở lên phức tạp.

Để thực hiện mở rộng lớp chia sẻ cho một số nền tảng cụ thể được đơn giản, kỹ thuật chia sẻ tạo ra các lớp thành phần (Partial Classes) được đưa ra. Kỹ thuật này được thực hiện như sau: Với 1 namespace thì 1 lớp chỉ khai báo 1 tên duy nhất đóng vai trò là lớp chia sẻ đa nền tảng, nhưng lập trình viên có thể tạo ra các lớp thành phần nhằm bổ sung phần khai báo cho lớp đó trên nhiều file khác nhau của mỗi nền tảng cụ thể. Như vậy kỹ thuật “Partial Classes” cho phép thực hiện khai báo một lớp được phân bổ trên

nhiều file khác nhau, nhưng khi trình biên dịch hoạt động, nó sẽ sáp nhập các file này và chạy như một file duy nhất. Sự kết hợp giữa kỹ thuật “Partial classes” với kỹ thuật “File linking” giúp cho việc tái sử dụng được nhiều mã nguồn trong khi vẫn cho phép khai báo mở rộng lớp chia sẻ khi cần thiết.

Minh họa sử dụng kỹ thuật này trong việc xây dựng ứng dụng yêu cầu trên với giả sử lớp thư viện trên nền tảng Android mở rộng thêm chức năng từ lớp chia sẻ ParkingSpotTracker mà nền tảng khác không có. Cách làm như sau:

- Bổ sung từ khóa partial trước định nghĩa lớp ParkingSpotTracker

```
// ParkingSpotTracker.cs (updates only)
using System;
namespace SharedLibrary.Chapter3
{
    public partial class ParkingSpotTracker
    {
        // only including updated parts to this class
    }
}
```

Trong project SharedLibrary.MonoAndroid, tạo file có tên ParkingSpotTracker.Extensions.cs trong thư mục Chapter3 nhằm chứa các phần khai báo mở rộng riêng trên nền tảng Android. Lưu ý rằng file này được đặt cùng thư mục với file liên kết với lớp chia sẻ và nó không ảnh hưởng tới các nền tảng khác.

```
// ParkingSpotTracker.Extensions.cs
namespace SharedLibrary.Chapter3
{
    public partial class ParkingSpotTracker
    {
        public void HonkHorn()
        {
            // honk the car horn
        }
    }
}
```

Conditional compilation

Kỹ thuật biên dịch có điều kiện (Conditional compilation) cho phép trình biên dịch đoạn mã lệnh tương ứng với chỉ thị miêu tả môi trường biên dịch, các đoạn mã lệnh khác không thuộc môi trường đang biên dịch sẽ bị bỏ qua.

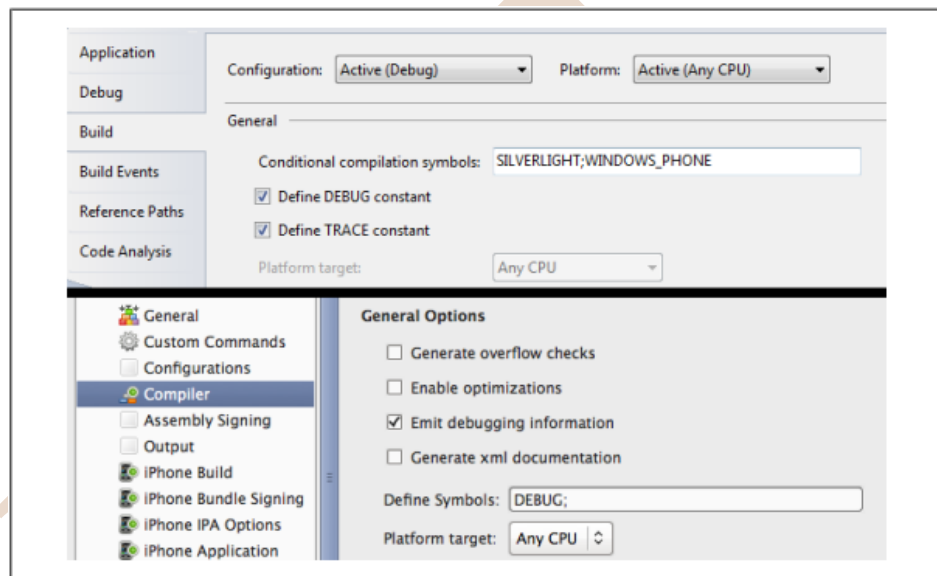
Lưu ý việc lạm dụng kỹ thuật này có thể khiến mã nguồn khó đọc và bảo trì. Trong trường hợp này việc sử dụng kỹ thuật “Abstraction” có thể sẽ là lựa chọn tốt hơn.

Dưới đây là chỉ thị miêu tả môi trường biên dịch được định nghĩa bởi Mono:

	iOS	Android	Windows Phone
Applications	None	__ANDROID__	SILVERLIGHT
		__ANDROID_1__	WINDOWS_PHONE
		__ANDROID_2__	
		...	
Class Libraries	None	__ANDROID__	SILVERLIGHT
		__ANDROID_1__	WINDOWS_PHONE
		__ANDROID_2__	

Hình 4.4. Chỉ thị miêu tả môi trường biên dịch được định nghĩa bởi Mono

Ngoài ra, lập trình viên có thể tự tạo ra chỉ thị miêu tả biên dịch trong project của mình bằng cách: trong Visual Studio, mở properties của project, chọn mục Build, điền chỉ thị mong muốn vào textbox, trong MonoDevelop, mở properties của project, chọn mục mở mục Compiler.



Hình 4.5. Định nghĩa chỉ thị biên dịch trong Visual Studio (phía trên) và MonoDevelop (phía dưới)

Minh họa sử dụng kỹ thuật này trong việc xây dựng ứng dụng yêu cầu trên với chức năng cập nhật vị trí người dùng (UpdateIntervalSeconds) có các giá trị tương ứng với mỗi nền tảng khác nhau như sau: Android: 30, iOS: 45, Windows Phone: 60. Việc cài đặt được thực hiện như sau:

Example 3-9. ParkingSpotTracker.cs (updates only)

```
using System;
namespace SharedLibrary.Chapter3
{
    public class ParkingSpotTracker
    {
        // only including updated parts to this class
        public int UpdateIntervalSeconds
        {
            get
            {

```

```

    #if __ANDROID__
    return 30;
    #elif MONOTOUCH
    return 45;
    #elif WINDOWS_PHONE
    return 60;
    #else
    return 120;
    #endif
}
}
}
}

```

4.3.3. Lập trình ứng dụng di động đa nền tảng

Nội dung phần này nhằm áp dụng các kỹ thuật chia sẻ đa nền tảng đã trình bày ở trên vào việc xây dựng ứng dụng đa nền tảng. Cụ thể: xây dựng ứng dụng có tên Twitter client kết nối mạng để tải về và hiển thị danh sách các tweet trên trang web Twitter của người dùng.

Mô tả về giải pháp xây dựng ứng dụng này như sau: Xây dựng thư viện chia sẻ đa nền tảng chứa mã nguồn truy cập mạng và lấy về thông tin các tweet trên trang web Twitter của người dùng; Xây dựng ứng dụng trên mỗi nền tảng (iOS, Android, Windows phone) được chia làm 2 phần: phần giao diện ứng dụng được tạo riêng cho mỗi nền tảng, phần xử lý logic ứng dụng sử dụng lại từ thư viện chia sẻ đa nền tảng đã tạo ra trước đó. Các bước thực hiện như sau:

Tạo thư viện chia sẻ đa nền tảng

Mở project SharedLibrary và tạo thư mục Chapter4 để chứa các lớp chia sẻ đa nền tảng. Thêm mới lớp Tweet vào thư mục Chapter4, lớp này xác định thông tin của đối tượng tweet, bao gồm: ID, thời gian tạo, nội dung thông tin.

```

// Tweet.cs
using System;
namespace SharedLibrary.Chapter4
{
    public class Tweet
    {
        public long Id { get; set; }
        public DateTime CreatedAt { get; set; }
        public string Text { get; set; }
    }
}

```

Twitter cho phép lấy dữ liệu thông qua các REST API từ URL sau:

https://api.twitter.com/1/statuses/user_timeline.xml?screen_name=OReillyMe
dia

Kết quả trả về của các API này dưới dạng file XML dạng sau:

```
// Sample XML response from Twitter (simplified)
<?xml version="1.0" encoding="UTF-8"?>
<statuses type="array">
<status>
<created_at>Mon Dec 12 11:28:54 +0000 2011</created_at>
<id>111111111111111111</id>
<text>...</text>
</status>
<status>
<created_at>Sun Dec 11 16:20:10 +0000 2011</created_at>
<id>222222222222222222</id>
<text>...</text>
</status>
</statuses>
```

Thêm lớp mới có tên `TwitterClient` vào thư mục `Chapter4`.

```
// TwitterClient.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Threading;
using System.Xml.Linq;
namespace SharedLibrary.Chapter4
{
public class TwitterClient
{
private const string baseUrl =
https://api.twitter.com/1/statuses/";
public void GetTweetsForUser(string user, Action<IList<Tweet>>
callback)
{
string url = _baseUrl + "user_timeline.xml?screen_name=" +
Uri.EscapeUriString(user);
var client = new WebClient();
client.DownloadStringCompleted +=
(sender, args) =>
{
var tweets = XDocument.Parse(args.Result).Root.
Elements("status").Select(status => new Tweet
{
Id = long.Parse(status.Element("id").Value),
CreatedAt =
DateTime.ParseExact(status.Element("created_at").Value,
"ddd MMM dd HH:mm:ss zz00 yyyy", null),
Text = status.Element("text").Value
})
.ToList();
callback(tweets);
};
}
```

```

        client.DownloadStringAsync(new Uri(url));
    }
}
}

```

Trong đó:

- Phương thức GetTweetsForUser() dùng để lấy dữ liệu từ website Twitter thông qua Twitter API. Phương thức này có tham số truyền vào của là tên người dùng (user) và phương thức callback xử lý kết quả trả về là danh sách tweet của người dùng đó.
- Quá trình tải dữ liệu từ Twitter là quá trình không đồng bộ với tiến trình chính, khi quá trình tải dữ liệu kết thúc, kết quả sẽ được trả về thông qua phương thức callback.

Sau khi thư viện chia sẻ được tạo ra, tiến hành liên kết file này trong project SharedLibrary.MonoTouch, SharedLibrary.MonoAndroid SharedLibrary.WindowsPhone để sử dụng thư viện đa nền tảng.

Xây dựng ứng dụng trên nền tảng iOS

Tạo ra project trống tên là Chapter4.MonoTouchApp. Thêm project SharedLibrary.MonoTouch vào solution và tham chiếu project Chapter4.MonoTouchApp tới project SharedLibrary. MonoTouch để được phép truy cập tới thư viện TwitterClient đã xây dựng ở trên.

- Tạo lớp TwitterViewController xử lý việc load giao diện tới người dùng khi quá trình tải danh sách tweet đang thực hiện, sau đó hiển thị danh sách kết quả trả về trên giao diện tới người dùng.

```

// TwitterViewController.cs
using MonoTouch.UIKit;
using SharedLibrary.Chapter4;
namespace Chapter4.MonoTouchApp
{
    public class TwitterViewController : UITableViewController
    {
        private TwitterClient _client;
        public override void ViewDidLoad()
        {
            base.ViewDidLoad();
            _client = new TwitterClient();
            var loading = new UIAlertView("Downloading Tweets",
                "Please wait...", null, null, null);
            loading.Show();
            var indicator = new
                UIActivityIndicatorView(UIActivityIndicatorViewStyle.WhiteLarge);
            indicator.Center = new
                System.Drawing.PointF(loading.Bounds.Width/2,
                loading.Bounds.Size.Height - 40);
            indicator.StartAnimating();
            loading.AddSubview(indicator);
            _client.GetTweetsForUser("OReillyMedia", tweets =>
            {
                InvokeOnMainThread(() =>

```

```

{
    TableView.Source = new TwitterTableViewSource(tweets);
    TableView.ReloadData();
    loading.DismissWithClickedButtonIndex(0, true);
});
});
}
}
}
// TwitterTableViewSource.cs
using System.Collections.Generic;
using MonoTouch.Foundation;
using MonoTouch.UIKit;
using SharedLibrary.Chapter4;
namespace Chapter4.MonoTouchApp
{
    public class TwitterTableViewSource : UITableViewSource
    {
        private readonly IList<Tweet> _tweets;
        private const string TweetCell = "TweetCell";
        public TwitterTableViewSource(IList<Tweet> tweets)
        {
            _tweets = tweets;
        }
        public override int RowsInSection(UITableView tableView, int section)
        {
            return _tweets.Count;
        }
        public override float GetHeightForRow(UITableView tableView, NSIndexPath indexPath)
        {
            return 60;
        }
        public override UITableViewCell GetCell(UITableView tableView, NSIndexPath indexPath)
        {
            var cell = tableView.DequeueReusableCell(TweetCell)
            ?? new UITableViewCell(UITableViewCellStyle.Subtitle, TweetCell);
            var tweet = _tweets[indexPath.Row];

```

Bổ sung mã nguồn cho AppDelegate.cs như dưới đây để TwitterViewController được load khi ứng dụng khởi động.

```

// AppDelegate.cs
using MonoTouch.Foundation;
using MonoTouch.UIKit;
namespace Chapter4.MonoTouchApp
{
    [Register ("AppDelegate")]
    public partial class AppDelegate : UIApplicationDelegate
    {
        private UIWindow _window;
        private TwitterViewController _twitterViewController;

```

```

public override bool FinishedLaunching(UIApplication app,
NSDictionary options)
{
    _window = new UIWindow (UIScreen.MainScreen.Bounds);
    _twitterViewController = new TwitterViewController();
    _window.RootViewController = _twitterViewController;
    _window.MakeKeyAndVisible ();
    return true;
}
}
}

```

Kết quả chạy ứng dụng:



Hình 4.6. Ứng dụng Twitter trên iOS

Xây dựng ứng dụng trên nền tảng Android

Tạo ra project trống tên là Chapter4.MonoAndroid. Thêm project SharedLibrary.MonoAndroid vào solution và tham chiếu project Chapter4.MonoAndroid tới project SharedLibrary.MonoAndroid để được phép truy cập tới thư viện TwitterClient đã xây dựng ở trên.

- Cấp quyền cho ứng dụng Twitter trên Android truy cập Internet bằng cách: trong Visual studio mở properties của project hoặc trong MonoDevelop chọn Mono for Android Application, chọn Android Manifest, trong mục Required permissions, tích chọn INTERNET.
- Từ thư mục Resources/Layout trong project, thêm mới file layout của Android tên là Twitter.axml, trong đó thêm mới ListView vào layout để hiển thị danh sách tweet. Tiếp theo thêm mới file layout tên là Tweet.axml để hiển thị thông tin chi tiết khi người dùng chọn 1 tweet trong danh sách, layout này chứa 2 TextView để hiển thị nội dung và thời gian của mỗi tweet.

//Twitter.axml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
<ListView
android:id="@+id/Tweets"
android:layout_width="match_parent"
android:layout_height="wrap_content" />
</LinearLayout>
//Tweet.xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
<TextView
android:id="@+id/Text"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:ellipsize="end"
android:singleLine="true"
android:textAppearance="?android:attr/textAppearanceMedium"
android:padding="5dip" />
<TextView
android:id="@+id/CreatedAt"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:padding="5dip" />
</LinearLayout>

```

Thêm Activity tên là **TwitterActivity** vào project dùng để điều khiển việc hiển thị danh sách tweet tới người dùng.

```

// TwitterActivity.cs
using Android.App;
using Android.OS;
using Android.Widget;
using SharedLibrary.Chapter4;
namespace Chapter4.MonoAndroidApp
{
    [Activity (Label = "\\@OReillyMedia", MainLauncher = true)]
    public class TwitterActivity : Activity
    {
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate (bundle);
            SetContentView (Resource.Layout.Twitter);
            var client = new TwitterClient();
            var loading = ProgressDialog.Show(this,
                "Downloading Tweets",
                "Please wait...", true);
            client.GetTweetsForUser("OReillyMedia", tweets =>
            {

```



```

RunOnUiThread (() =>
{
var tweetList = FindViewById<ListView>(Resource.Id.Tweets);
tweetList.Adapter = new TweetListAdapter(this, tweets);
tweetList.ItemClick += (object sender, ItemEventArgs e) =>
{
var selectedTweet = tweets[e.Position];
new AlertDialog.Builder(this)
.SetTitle("Full Tweet")
.SetMessage(selectedTweet.Text)
.SetPositiveButton("Ok", delegate { })
.Show();
};
loading.Hide();
});
});
}
}

// TweetListAdapter.cs
using System.Collections.Generic;
using Android.App;
using Android.Views;
using Android.Widget;
using SharedLibrary.Chapter4;
namespace Chapter4.MonoAndroidApp
{
public class TweetListAdapter : BaseAdapter<Tweet>
{
private readonly Activity _context;
private readonly IList<Tweet> _tweets;
public TweetListAdapter(Activity context, IList<Tweet> tweets)
{
_context = context;
_tweets = tweets;
}
public override View GetView(int position, View convertView,
ViewGroup parent)
{
var view = convertView
?? _context.LayoutInflater.Inflate(Resource.Layout.Tweet,
null);
var tweet = _tweets[position];
view.FindViewById<TextView>(Resource.Id.Text).Text = tweet.Text;
view.FindViewById<TextView>(Resource.Id.CreatedAt).Text =
tweet.CreatedAt.ToLocalTime().ToString();
return view;
}
public override int Count
{
get { return _tweets.Count; }
}
public override long GetItemId(int position)
{
return position;
}
}
}

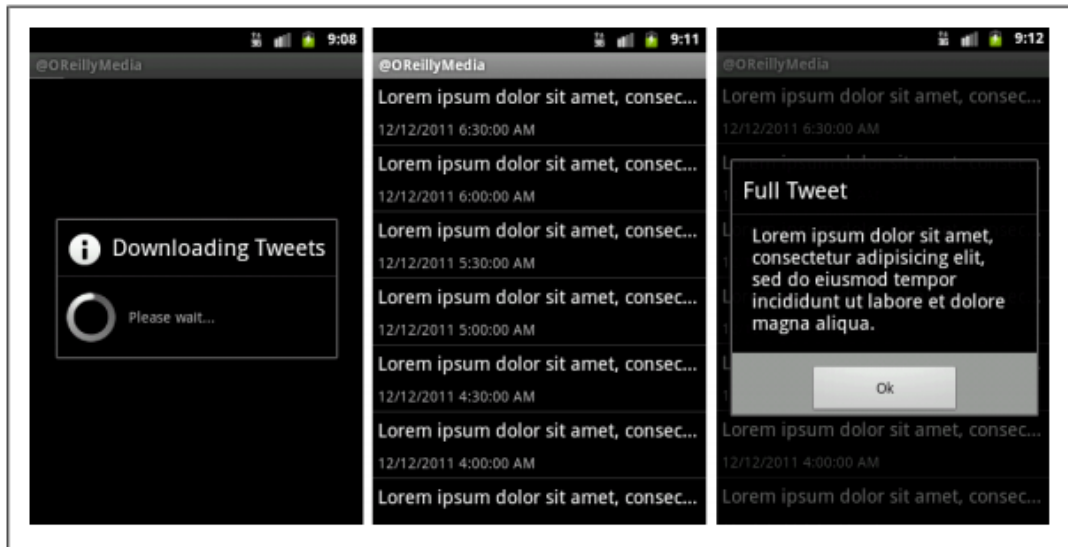
```

```

    }
    public override Tweet this[int position]
    {
        get { return _tweets[position]; }
    }
}
}

```

Kết quả chạy ứng dụng:



Hình 4.7. Ứng dụng Twitter trên Android

Xây dựng ứng dụng trên nền tảng Windows Phone

Tạo ra project trống tên là Chapter4.WindowsPhoneApp. Thêm project SharedLibrary.WindowsPhone vào solution và tham chiếu project Chapter4.WindowsPhoneApp tới project SharedLibrary.WindowsPhone để được phép truy cập tới thư viện TwitterClient đã xây dựng ở trên.

Tạo giao diện ứng dụng bằng cách chỉnh sửa mã nguồn MainPage.xaml:

```

// MainPage.xaml
<phone:PhoneApplicationPage
    x:Class="Chapter4.WindowsPhoneApp.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clrnamespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clrnamespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">
    <Grid x:Name="LayoutRoot" Background="Transparent">
    <Grid.RowDefinitions>

```

```

<RowDefinition Height="Auto"/>
<RowDefinition Height="*/>
</Grid.RowDefinitions>
<StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
<TextBlock x:Name="ApplicationTitle" Text="Twitter"
Style="{StaticResource
PhoneTextNormalStyle}"/>
<TextBlock x:Name="PageTitle" Text="@OReillyMedia" Margin="9,-
7,0,0"
Style="{StaticResource PhoneTextTitle1Style}"/>
</StackPanel>
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
<StackPanel x:Name="Loading">
<ProgressBar IsIndeterminate="True" />
<TextBlock TextAlignment="Center" Text="Downloading tweets,
please
wait..." />
</StackPanel>
<ListBox x:Name="Items" Margin="0,0,-12,0"
ItemsSource="{Binding}"
SelectionChanged="TweetSelected">
<ListBox.ItemTemplate>
<DataTemplate>
<StackPanel Margin="0,15,0,15">
<TextBlock Text="{Binding Text}"
TextTrimming="WordEllipsis"
Style="{StaticResource PhoneTextNormalStyle}"/>
<TextBlock Text="{Binding CreatedAt}"
Style="{StaticResource
PhoneTextSmallStyle}"/>
</StackPanel>
</DataTemplate>
</ListBox.ItemTemplate>
</ListBox>
</Grid>
</Grid>
</phone:PhoneApplicationPage>

```

Viết code điều khiển việc hiển thị danh sách tweet tới người dùng trong MainPage.xaml.cs:

```

// MainPage.xaml.cs
using System.Windows;
using System.Windows.Controls;
using Microsoft.Phone.Controls;
using SharedLibrary.Chapter4;
namespace Chapter4.WindowsPhoneApp
{
public partial class MainPage : PhoneApplicationPage
{
public MainPage()
{
InitializeComponent();
}
protected override void
OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs

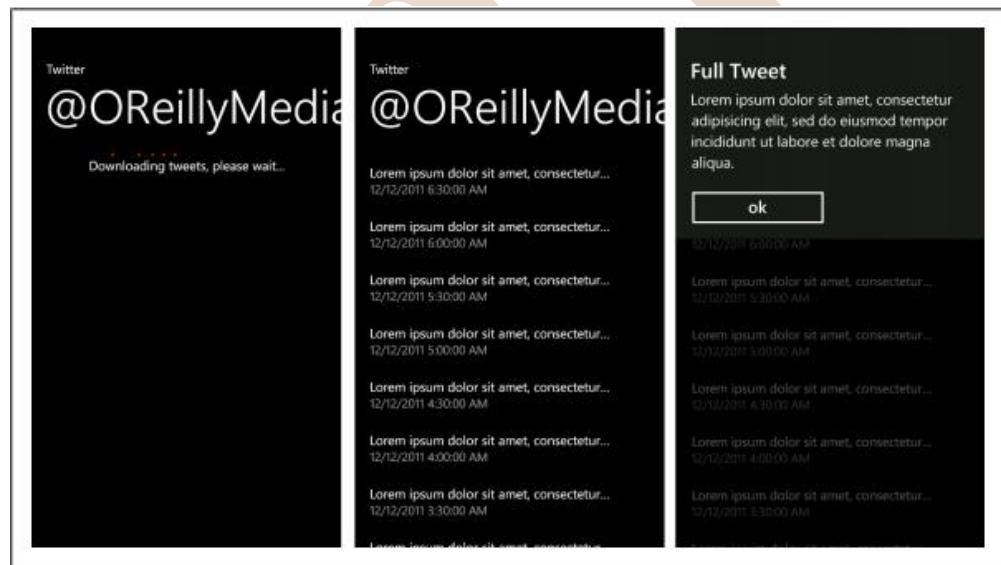
```

```

e)
{
base.OnNavigatedTo(e);
var client = new TwitterClient();
client.GetTweetsForUser("OReillyMedia", tweets =>
{
Deployment.Current.Dispatcher.BeginInvoke(() =>
{
DataContext = tweets;
Loading.Visibility = Visibility.Collapsed;
});
});
}
private void TweetSelected(object sender,
SelectionChangedEventArgs e)
{
{
var tweet = (Tweet)e.AddedItems[0];
MessageBox.Show(tweet.Text, "Full Tweet", MessageBoxButton.OK);
}
}
}

```

Kết quả chạy ứng dụng:



Hình 4.8. Ứng dụng Twitter trên Windows Phone

CÂU HỎI VÀ BÀI TẬP CHƯƠNG 4

1. Ứng dụng di động đa nền tảng là gì? Ưu nhược điểm của việc phát triển ứng dụng di động đa nền tảng so với việc phát triển ứng dụng di động đơn nền tảng là gì?
2. Ứng dụng di động đa nền tảng được phân chia thành các loại nào? Mỗi loại ứng dụng đa nền tảng đó phù hợp với mục đích xây dựng ứng dụng như thế nào?
3. Nêu một số framework phát triển ứng dụng đa nền tảng. Việc lựa chọn framework để phát triển ứng dụng đa nền tảng căn cứ trên các tiêu chí nào?
4. Nêu các kỹ thuật chia sẻ mã nguồn đa nền tảng và đưa ra mục đích sử dụng

của mỗi loại kỹ thuật chia sẻ đó.

5. Xây dựng ứng dụng máy tính đa nền tảng.

6. Phát triển tiếp ứng dụng Twitter đa nền tảng nhằm mục đích đưa ra thông tin báo tới người dùng khi người dùng đó được một ai đó trong cộng đồng mạng Twitter đề cập tới.

7. Xây dựng ứng dụng quản lý sinh viên trên đa nền tảng.

PREL

TÀI LIỆU THAM KHẢO

Tiếng Anh

1. Wei-Meng Lee, Beginning Android 4 Application Development. John Wiley & Sons, Inc, 2012
2. Brian Fling, Mobile design and development. 1st edition. O'Reilly Media, 2009
3. Reto Meier, Professional Android 4 Application Development. John Wiley & Sons, Inc, 2012
4. Scott Olson, John Hunter, Ben Horgen, Kenny Goers, Professional Cross-Platform Mobile Development in C#. John Wiley & Sons, Inc, 2012

PDF