

TRƯỜNG ĐẠI HỌC QUỐC TẾ HỒNG BÀNG
KHOA CÔNG NGHỆ THÔNG TIN
oo

LẬP TRÌNH DI ĐỘNG TRÊN MÔI TRƯỜNG ANDROID

(TÀI LIỆU THAM KHẢO)

LÊ VĂN HẠNH
2014

MỤC LỤC

1 Phần I: KHỞI ĐẦU.....	1
1.1. CẤU HÌNH MÔI TRƯỜNG PHÁT TRIỂN Android (Android Integrated Development Environment (Android IDE)).....	1
1.1.1. Hệ điều hành hỗ trợ	1
1.1.2. Môi trường phát triển hỗ trợ	1
1.1.3. Cài đặt ADT Bundle - “bộ thu gọn”	6
1.1.4. Tạo Android Emulator (AVD)	7
1.2. TẠO ỨNG DỤNG ĐƠN GIẢN	9
1.2.1. Tạo mới Project	9
1.2.2. Cấu trúc một project	12
1.2.3. Tìm hiểu về XML Layout và Java Code	13
1.2.4. Chạy ứng dụng	14
1.2.5. Tùy biến ứng dụng đầu tiên vừa xây dựng	16
1.3. TÌM HIỂU VỀ Android Project & Android Activity	18
1.3.1. Application	18
1.3.2. Giới thiệu activity	19
1.3.3. Tạo mới 1 activity.....	20
1.3.4. Khai báo activity trong file AndroidManifest.xml	20
1.3.5. Khởi chạy một activity	20
1.3.6. Chuyển thông tin giữa hai Activities.....	28
1.3.7. Intents	35
1.3.8. Vận dụng kiểu mở Activity mới khi lập trình	41
1.3.9. Vòng đời của Activity	41
1.4. SỬ DỤNG TÀI NGUYÊN TRONG Android	44
1.4.1. Các file về tài nguyên trong Android Project.....	44
1.4.2. Các tài nguyên thường dùng.....	45
1.5. BÀI TẬP TỔNG HỢP PHẦN I.....	55
1.5.1. Mở rộng từ những bài thực hành đã thực hiện	55
1.5.2. Bài tập 2.....	55
2 Phần II: GIAO DIỆN NGƯỜI DÙNG	56
2.1. SỬ DỤNG Layouts.....	56
2.1.1. Một số khái niệm	56
2.1.2. Một số vấn đề cần quan tâm khi thiết kế giao diện và viết mã lệnh.....	57
2.1.3. Xây dựng ứng dụng	59
2.1.4. ViewGroup	63
2.2. MỘT SỐ CONTROL CƠ BẢN.....	81
2.2.1. Sử dụng các điều khiển nhập liệu cơ bản	81
2.2.2. Sử dụng các điều khiển với Adapters	89
2.2.3. ProgressBars and SeekBars	94
2.2.4. ImageViews	105
2.3. ActionBar & Menu Navigation.....	108
2.3.1. Tìm hiểu về Options Menu.....	108
2.3.2. Sử dụng Action Bar	114
2.4. Activities & Fragments	122
2.4.1. Fragments	122
2.4.2. Di chuyển giữa các Fragments	128
2.4.3. Tương tác giữa Fragment và Activity	134
2.5. CÁC DẠNG Dialogs	137
2.5.1. Dialog Fragment	137

2.5.2. Dialogs dùng cho việc chọn ngày và giờ.....	144
2.5.3. Sử dụng các Dialogs dạng cảnh báo (Alert dialog).....	147
2.6. LISTS, GRIDS, GALLERIES, & FLIPPERS	152
2.6.1. ListFragments	152
2.6.2. Grids và Galleries	158
2.6.3. Sử dụng AdapterViewFlipper.....	164
2.6.4. Một số control dùng trong việc phân trang dạng cuộn ngang	167
2.7. BÀI TẬP TỔNG HỢP PHẦN II	167
2.7.1. Thiết kế giao diện	167
2.7.2. Bài tập về fragment	169
2.7.3. Bài tập về các xử lý cơ bản.....	170
2.7.4. ListView kết hợp với các cấu trúc dữ liệu.....	176
2.7.5. Tạo ứng dụng định dạng file chữ.....	178
2.7.6. ActionBar	178
2.7.7. TimePickerDialog kết hợp với dialog fragment.....	178
3 Phần III: LƯU TRỮ DỮ LIỆU	179
3.1. QUẢN LÝ CÁC TÙY CHỌN (Preferences).....	179
3.1.1. Sử dụng SharedPreferences	179
3.1.2. Thiết lập các tùy chọn của người dùng (<i>User Preferences</i>)	181
3.2. SQLite	193
3.2.1. Giới thiệu.....	193
3.2.2. Cài đặt SQLite trên Windows.....	195
3.2.3. Kiểu dữ liệu trong SQLite	197
3.2.4. Lệnh SQLite	197
3.2.5. Toán tử trong SQLite(SQLite Operators).....	200
3.2.6. Biểu thức trong SQLite (<i>SQLite Expressions</i>)	203
3.2.7. Các lệnh liên quan đến CSDL	203
3.2.8. Các lệnh liên quan đến cấu trúc của TABLE	205
3.2.9. Lệnh Insert Into	209
3.2.10. Lệnh truy vấn dữ liệu	209
3.2.11. Update.....	218
3.2.12. Delete.....	218
3.2.13. Lệnh VACUUM	219
3.2.14. Sub Queries	219
3.2.15. Views	221
3.2.16. Trigger	222
3.2.17. Transactions.....	224
3.2.18. Indexes.....	225
3.2.19. Các hàm thường dùng trong SQLite.....	227
3.2.20. SQLite PRAGMA	230
3.3. THAO TÁC VỚI FILE CỦA ỨNG DỤNG ĐƯỢC TẠO RA TRÊN AVD.....	234
3.4. SỬ DỤNG CSDL SQLite TRONG Android.....	236
3.4.1. Package.....	236
3.4.2. Class	236
3.4.3. Cursor	236
3.4.4. SQLiteOpenHelper class	240
3.4.5. SQLiteDatabase class	241
3.5. Content Provider	252
3.5.1. Giới thiệu.....	252
3.5.2. Truy cập Content Provider	252
3.5.3. Custom content provider	252
3.5.4. UriMatcher	253

3.5.5. Thread Safety.....	253
3.6. LUỒU TRỮ DỮ LIỆU DƯỚI DẠNG FILE.....	264
3.6.1. Sử dụng bộ nhớ trong	264
3.6.2. Sử dụng cache file	266
3.6.3. Sử dụng bộ nhớ ngoài.....	266
3.6.4. Một số phương thức hữu dụng đối với file:.....	268
3.7. BÀI TẬP TỔNG HỢP CHƯƠNG 3.....	277
3.7.1. Game.....	277
4 Phàn IV: NETWORK & TELEPHONY	290
4.1. NETWORK.....	290
4.1.1. Giao thức HTTP	290
4.1.2. Sử dụng kết nối mạng trong Android.....	290
4.1.3. WebView	295
4.2. Telephony	300
4.2.1. SMS	300
4.3. Sending e-mail	323
4.4. BÀI TẬP TỔNG HỢP CHƯƠNG 4.....	330
4.4.1. 330	

Phần I: KHỞI ĐẦU

- (i). Cấu hình môi trường phát triển Android
- (ii). Tạo ứng dụng đơn giản
- (iii). Tìm hiểu về Android Project và Android Activity
- (iv). Sử dụng tài nguyên (Resources) trong Android

1.1. CẤU HÌNH MÔI TRƯỜNG PHÁT TRIỂN Android (Android Integrated Development Environment (Android IDE))

Do Android được hỗ trợ đáng kể trong môi trường Eclipse, nên tài liệu này sẽ sử dụng môi trường phát triển là Eclipse để minh họa.

1.1.1. Hệ điều hành hỗ trợ

- Windows XP (32-bit), Vista (32- 64-bit), Windows 7 (32- 64-bit)
- Mac OS X 10.4.8 trở lên
- Linux (Ubuntu Linux, Lucid Lynx)

1.1.2. Môi trường phát triển hỗ trợ

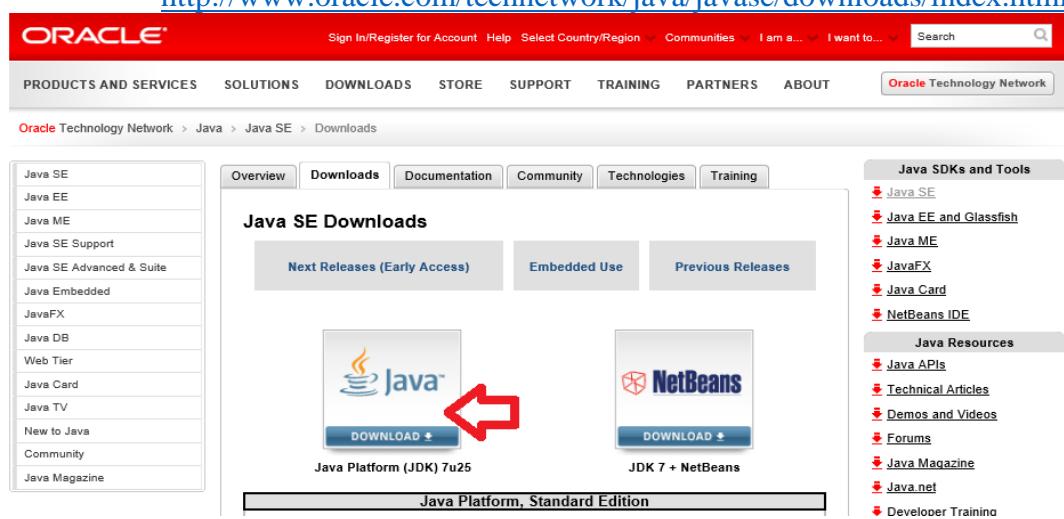
- Môi trường Android có sự phân biệt giữa Android SDK tools và platform tools. Tools là thành phần trung tâm của Software Development Kit (SDK). Chúng được sử dụng trong tất cả các version của Android. Platform tools được kết hợp với từng version riêng biệt của Android.
- Để phát triển ứng dụng Android cần cài đặt các software sau:
 - [1] JDK: (java development kit) 1.7 hoặc lớn hơn
 - [2] IDE (môi trường phát triển): Eclipse (Eclipse 3.5 - Galileo hoặc lớn hơn)
 - [3] ADT (Android Development Tools plugin)
 - [4] SDK (Android Software Development Kit)
- Yêu cầu máy đang cài phải kết nối mạng trong quá trình cài đặt.
- Có thể xem hướng dẫn cài đặt tại: <http://developer.android.com/sdk/installing.html>

1.1.2.1. JDK

1.1.2.1.1. Download JDK

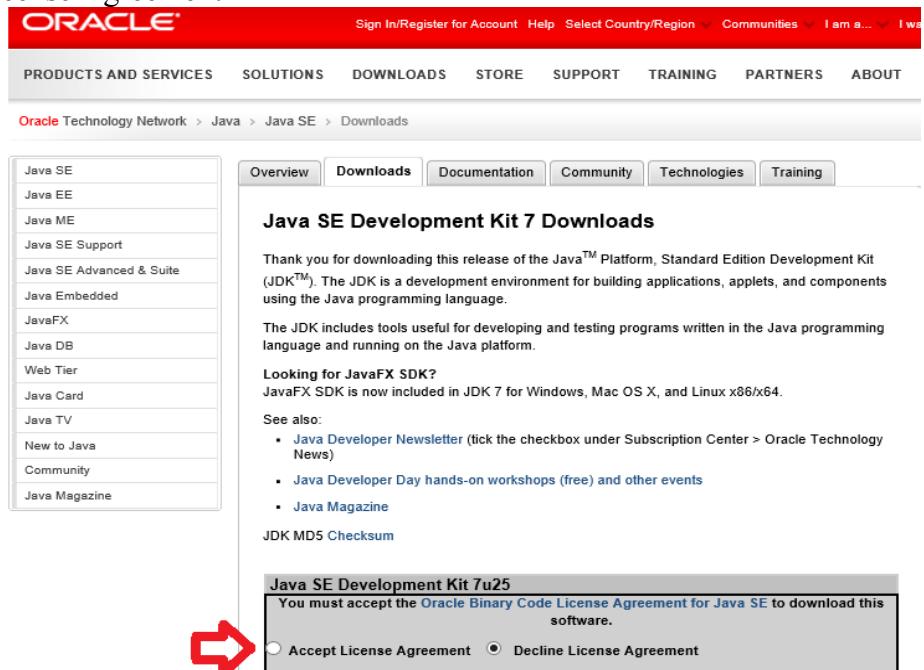
- Vào địa chỉ sau để download

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>



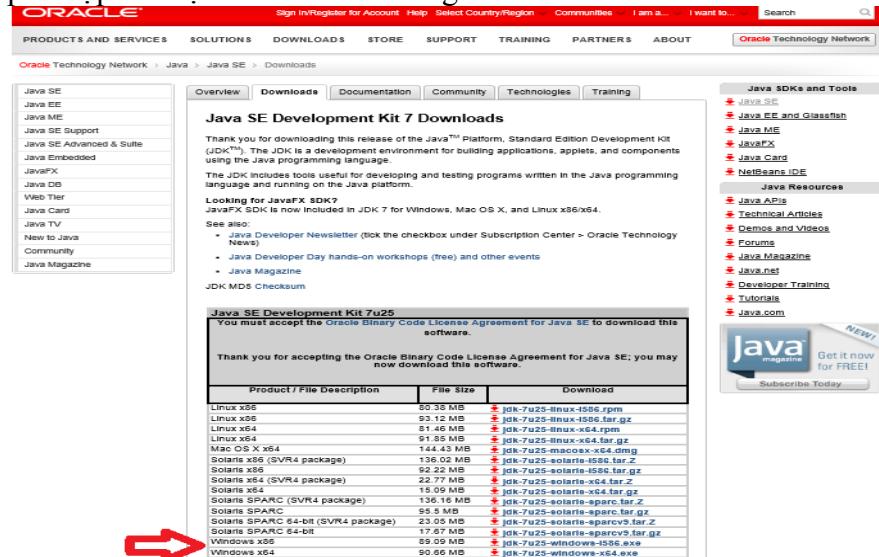
Hình 1-1 download JDK

- Chọn Accept License Agreement



Hình 1-2 Chọn Accept License Agreement

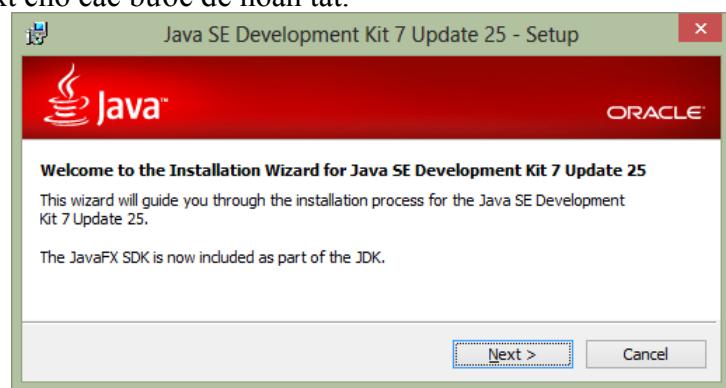
- Chọn phiên bản phù hợp với hệ điều hành cần dùng



Hình 1-3 Chọn phiên bản phù hợp với hệ điều hành cần dùng

1.1.2.1.2. Cài đặt JDK

- Run file vừa download, chọn Next cho các bước để hoàn tất.



Hình 1-4 Cài đặt JDK

1.1.2.2. Eclipse

1.1.2.2.1. Download:

- Download tại địa chỉ <http://www.eclipse.org/downloads/>
- Phiên bản đề nghị download là Eclipse Kepler SR2 (4.3.2)



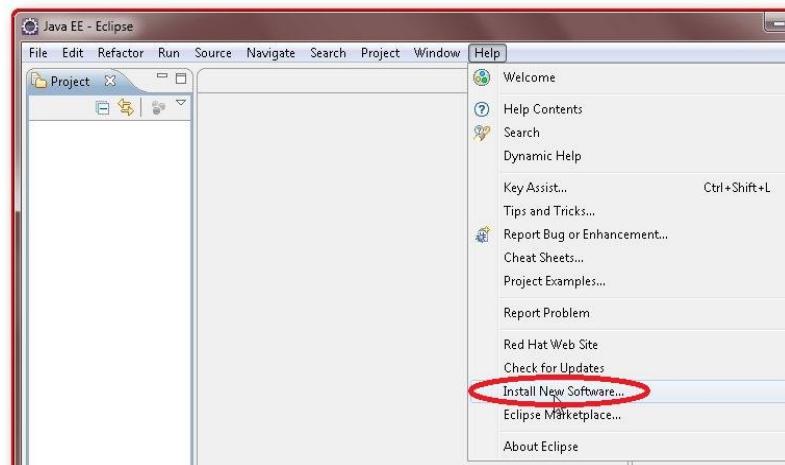
Hình 1-5 Một số màn hình giới thiệu của eclipse (tùy thuộc phiên bản cài đặt)

1.1.2.2.2. Sử dụng

- Sau khi download hoàn tất, thực hiện giải nén file vừa có.
- Run file `eclipse.exe` (`eclipse.exe`) trong folder vừa giải nén để mở ứng dụng Eclipse.

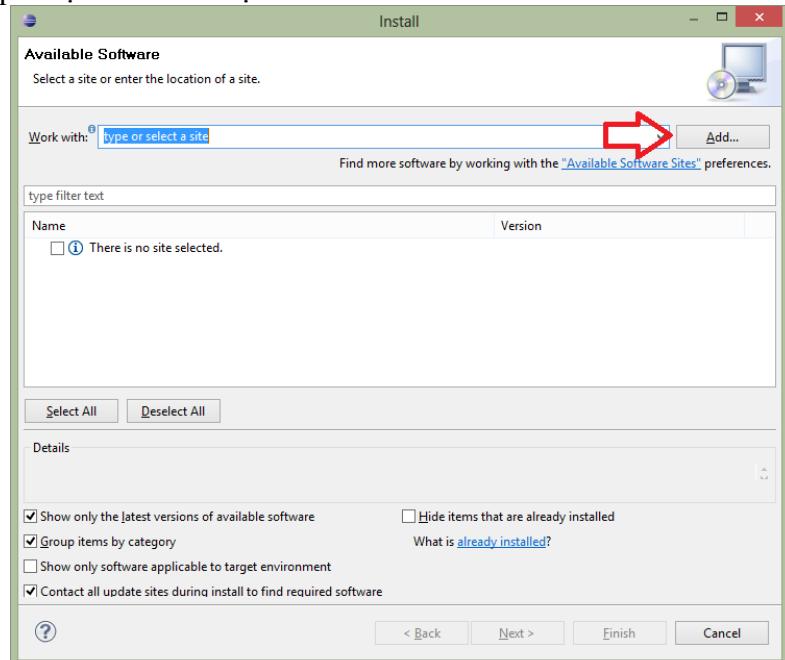
1.1.2.2.3. SDK

Cài đặt ADT plugin: Mở eclipse. Chọn *Help → Install new software.*



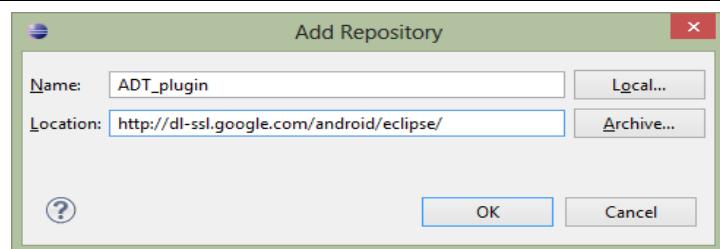
Hình 1-6 chọn menu để cài đặt ADT

- Chọn button Add trong hộp thoại vừa xuất hiện



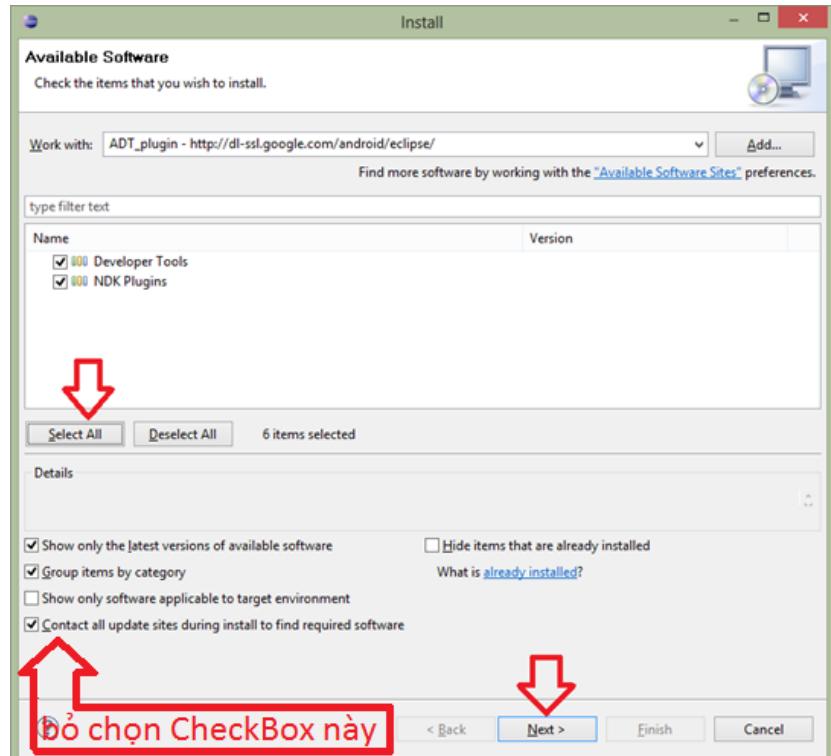
Hình 1-7 chọn button Add

- Ở mục Name, đặt tên cho ADT, ví dụ: ADT plugin
- Ở mục Location, nhập link: <https://dl-ssl.google.com/android/eclipse/>
- Click button OK để hoàn tất và đóng hộp thoại này



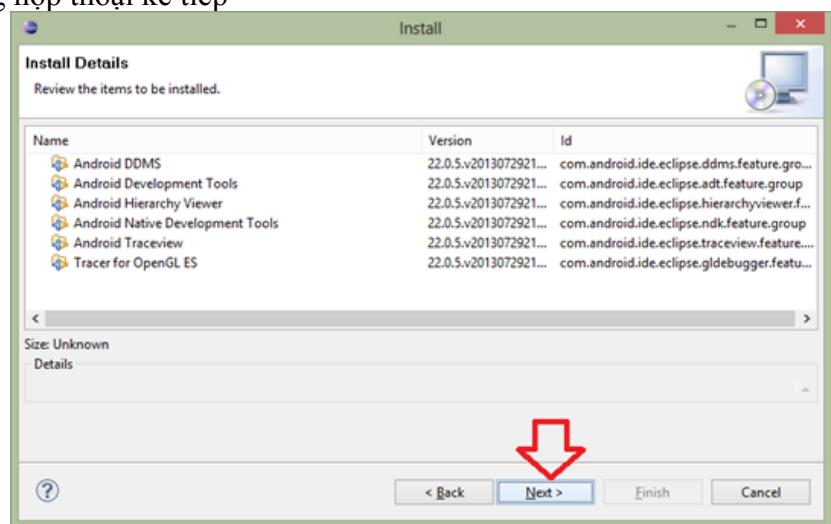
Hình 1-8 nhập tên và liên kết để cài đặt ADT

- Trở về hộp thoại đầu tiên. Chọn button *Select All*
- Bỏ chọn checkbox *Contact all ...*
- Chọn button *Next*



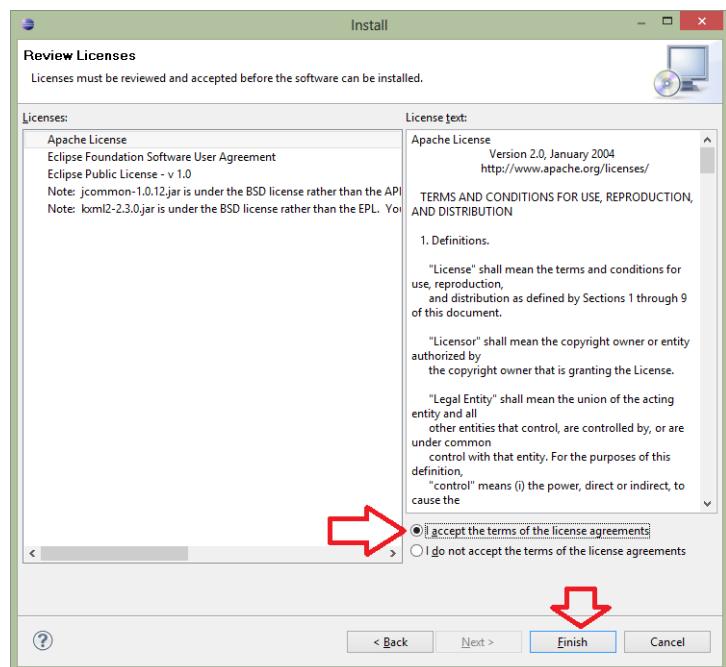
Hình 1-9 Chọn lựa các mục khi cài đặt ADT

- Chọn button *Next* trong hộp thoại kế tiếp



Hình 1-10 Xác nhận các công cụ sẽ được cài đặt

- Chọn *I accept ...*
- Chọn button *Finish* để bắt đầu cài đặt



Hình 1-11 Chọn Finish để bắt đầu cài đặt

Trong quá trình cài:

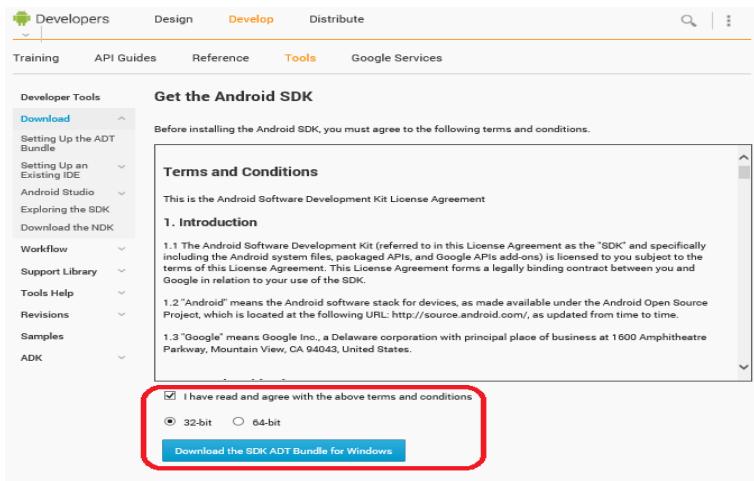
- Yêu cầu máy phải kết nối mạng.
- Có thể xuất hiện một số thông báo, khi đó chọn OK (hoặc nút gì đó của thông báo).

Sau khi cài đặt hoàn tất, eclipse sẽ yêu cầu khởi động lại.

1.1.2.3. Android SDK

1.1.2.3.1. Download Android SDK

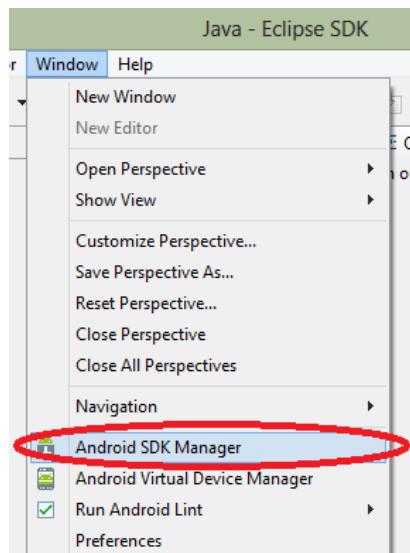
Download tại link <http://developer.android.com/sdk/index.html> (chú ý đến hệ điều hành tương thích)



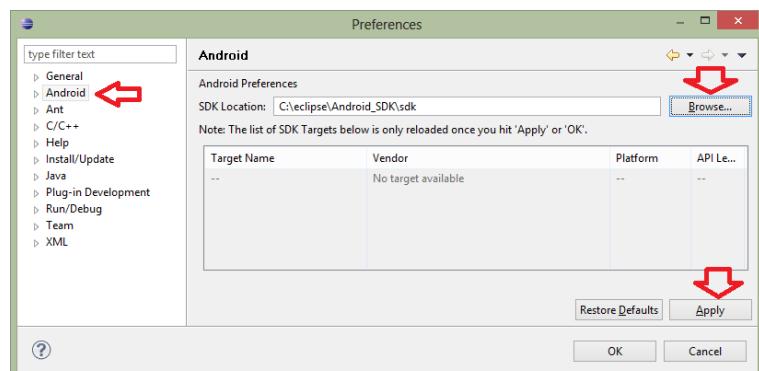
Hình 1-12 Chú ý đến hệ điều hành tương thích

1.1.2.3.2. Xác lập cho Eclipse đường dẫn đến Android SDK

- Giải nén file SDK vừa download
- Trong Eclipse, chọn menu Window → Preferences
- Trong hộp thoại Preferences, chọn Android (bên trái)
- Trong mục SDK location chọn Browse → đưa đường dẫn đến thư mục SDK (thư mục chứa trong thư mục vừa mới giải nén, vì bên trong folder này có chứa folder **TOOLS**) → click Apply.



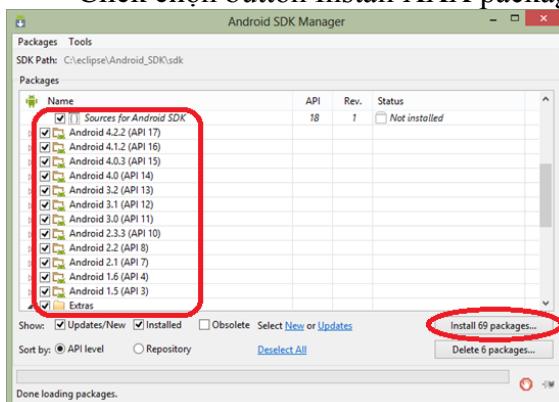
Hình 1-13 Chọn Android SDK Manager



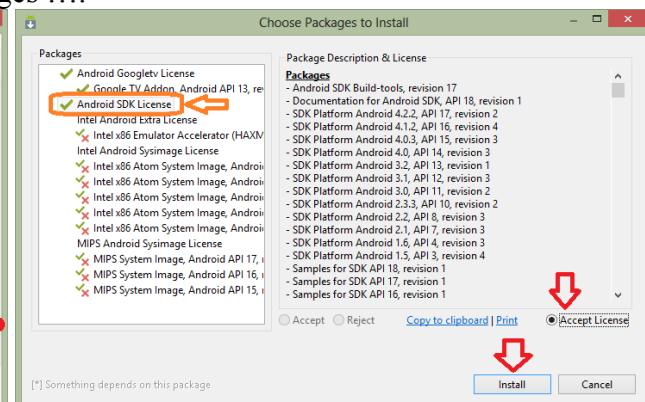
Hình 1-14 Xác lập đường dẫn của folder SDK

1.1.2.3.3. Cài đặt Android SDK

- Trong Eclipse, chọn menu Window → Android SDK Manager
- Trong hộp thoại Android SDK Manager, có thể chọn hết (cho tiện sử dụng sau này) hoặc chỉ chọn Android 1 trong các version của Android (từ 2.1 trở lên)
- Click chọn button Install XXX packages



Hình 1-15 Chọn button XXX packages để bắt đầu cài đặt



Hình 1-16 Chọn Accept License và Install

- Chọn Android SDK License và Accept License
- Click button Install để bắt đầu cài đặt

1.1.3. Cài đặt ADT Bundle - “bộ thu gọn”

Ngoài cách cài đặt trên, ta có thể chọn bộ cài đặt “thu gọn” để sử dụng.

File ADT Bundle cũng cung cấp tất cả những gì bạn cần để phát triển ứng dụng Android là Eclipse IDE và tất cả các công cụ.

Để cài đặt Eclipse và các Eclipse plugin for Android (Android Developer tools), bạn download file cài đặt với tên gọi là ADT Bundle tại một số địa chỉ như:

- <http://developer.android.com/sdk/index.html>
- <http://www.softpedia.com/get/Programming/SDK-DDK/ADT-Bundle.shtml>

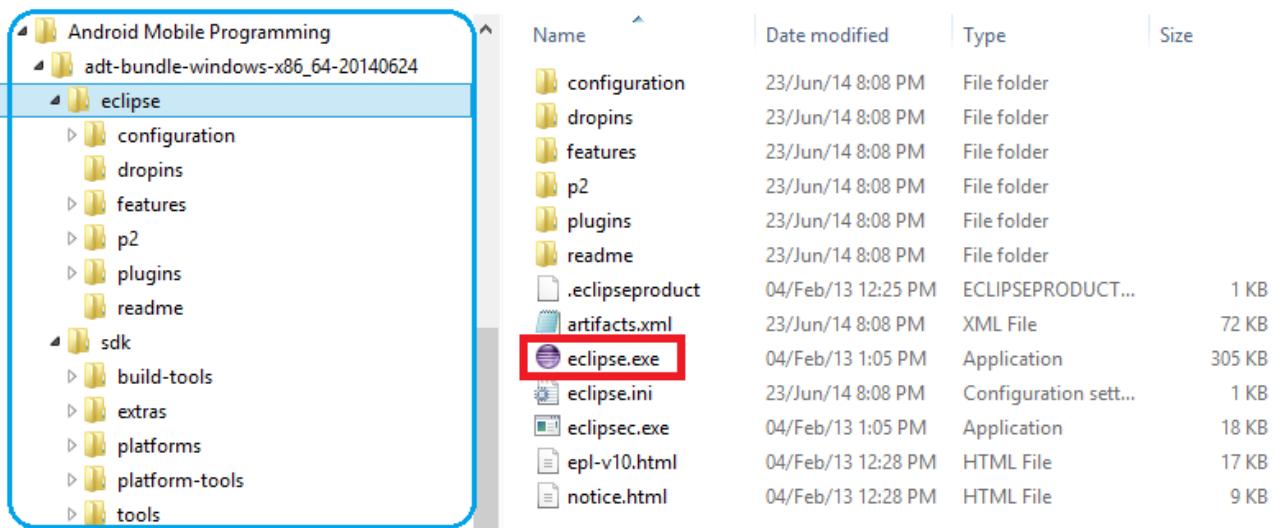
Do ADT Bundle là một file nén nên sau khi download hoàn tất, bạn lặp lại 2 bước sau:

B1: giải nén file có tên là adt-bundle-<os_platform>.zip vào vị trí sao cho phù hợp với mình (ví dụ D:\LapTrinhDiDong)

B2: Cho thực thi file **Eclipse** trong folder adt-bundle-<os_platform>/eclipse/

Lưu ý: đừng di chuyển bất kỳ file hoặc folder nào trong folder `adt-bundle-<os_platform>`.

Vì nếu bạn di chuyển folder `eclipse/` hoặc `sdk/`, ADT sẽ không thể định vị được vị trí của SDK và khi đó cần phải thực hiện cập nhật lại bằng cách thủ công cho ADT.



Hình 1-17 Cấu trúc của folder ADT Bundle

Trong folder này chứa 2 folder chính:

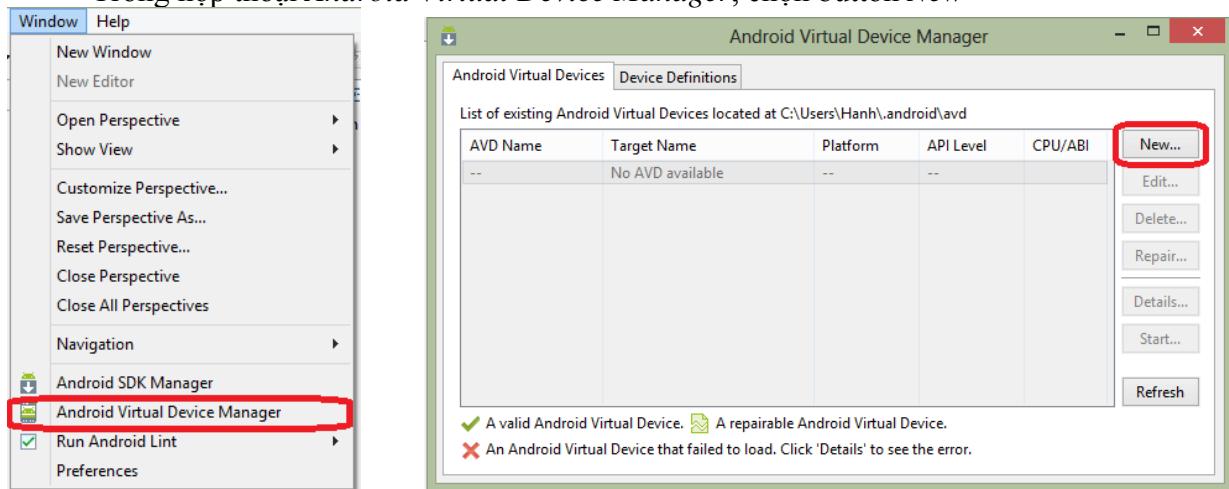
- **Eclipse:** chứa Eclipse IDE và các tools liên quan.
- **sdk:** chứa 1 số folder con chứa các tools, platforms, và platform tools.

1.1.4. Tạo Android Emulator (AVD)

Để chạy thử ứng dụng Android, bạn cần sử dụng môi trường giả lập (emulator). Thiết bị giả lập này được thiết lập trong folder system-images và thiết bị này từ đây trở đi sẽ được gọi là AVD (Android Virtual Device). Bạn có thể tạo AVD bằng cách sử dụng công cụ *Android Virtual Device Manager* từ menu Windows trong Android.

1.1.4.1. Tạo AVD

- Trong Eclipse, chọn menu *Window* → *Android Virtual Device Manager*
- Trong hộp thoại *Android Virtual Device Manager*, chọn button *New...*

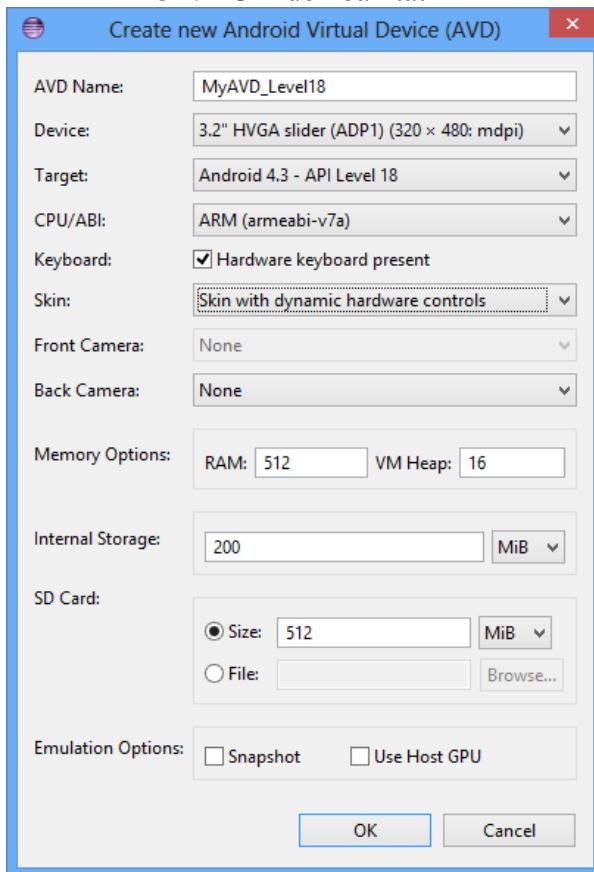


Hình 1-18 Tạo AVD

- Trong hộp thoại Create new Android Virtual Device (AVD):

- **AVD Name:** đặt tên điện thoại ảo (dễ nhớ và nên phân biệt level của AVD là bao nhiêu)
- **Device:** chọn HVGA cho dễ dùng
- **Target:** chọn API mà ứng dụng mong muốn tốt nhất có thể thực thi trên API này (từ API level 11 trở lên là tốt nhất).

- **Memory Options:** bộ nhớ trong nên chọn tối thiểu 256
 - **SD Card:** bộ nhớ ngoài (thẻ nhớ), nên chọn tối thiểu 256MB, có thể đặt dung lượng thẻ nhớ là 2GB
 - Click **OK** để hoàn tất



Hình 1-19 Xác lập thông số cho AVD

Android Virtual Device Manager				
AVD Name	Target Name	Platform	API Level	CPU/ABI
MyAVD_Level18	Android 4.3	4.3	18	ARM (armeabi-v...

A valid Android Virtual Device. A repairable Android Virtual Device.
An Android Virtual Device that failed to load. Click 'Details' to see the error.

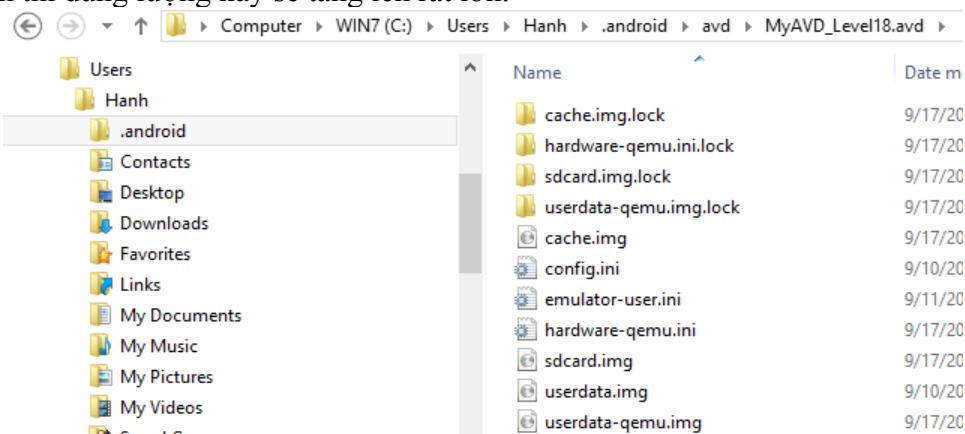
Hình 1-20 Thông tin về các AVD đã tạo thành công

- Điện thoại ảo đã được tạo thành công (hình 1-20)

Lưu ý: máy ảo được tạo ra sẽ được lưu trữ vào user của máy tính đang hoạt động (ở hình trên máy ảo lưu trong “C:\Users\Hanh\.android\avd”)

1.1.4.2. Khảo sát folder chứa máy ảo

- **File sdcards.img:** dung lượng file này chính là dung lượng đã cấp cho SD Card
- **File snapshots.img:** công dụng để lưu trữ lại toàn bộ thông số để các lần khởi chạy sau nhanh hơn. Khi mới tạo, dung lượng file chỉ khoảng 250Kb, nhưng nếu sau lần khởi động đầu tiên thì dung lượng này sẽ tăng lên rất lớn.



Hình 1-21 folder chứa thông tin về AVD

1.1.4.3. Bật DPAD trong AVD

B1.- tìm file cấu hình có dạng:

~/.android/avd/XXXX.avd/config.ini

Trong đó ký hiệu ~ được tượng trưng cho folder Home của từng người dùng (VD: C:\Users\Hanh\.android\avd). Lưu ý folder này chỉ có duy nhất 1 file dạng .INI.

B2.- hiệu chỉnh file config.ini: tìm đến đoạn **hw.dPad=no**, đổi lại thành **hw.dPad=yes**

B3.- Tắt khởi động lại AVD đang dùng. Nhờ thay đổi này, button DETAILS trong VDM cũng sẽ xuất hiện

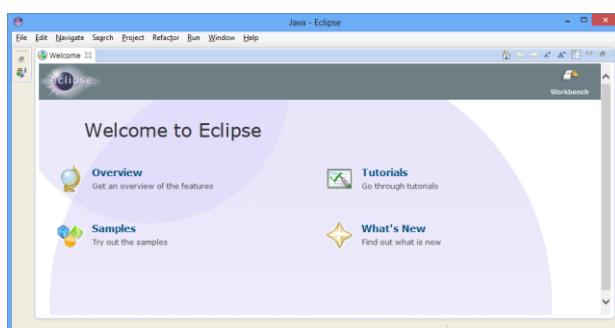
1.2. TẠO ỨNG DỤNG ĐƠN GIẢN

Khi khởi động file Eclipse.exe trong folder Eclipse, màn hình sẽ có dạng như hình sau. Trong đó bạn cần xác định folder nơi sẽ lưu trữ các ứng dụng của mình.

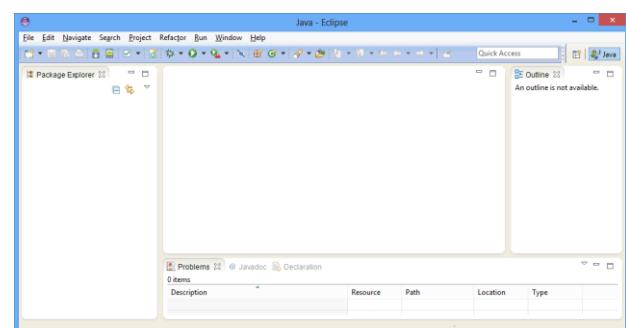


Hình 1-22 Chọn workspace để làm việc trong Eclipse

Sau khi xác định folder nơi sẽ lưu trữ các ứng dụng, sẽ xuất hiện màn hình *Welcome to Eclipse* (hình 1-23).



Hình 1-23 Màn hình Welcome to Eclipse



Hình 1-24 Màn hình làm việc với Eclipse

Sau khi đóng màn hình này, màn hình làm việc với Eclipse sẽ xuất hiện (hình 1-27).

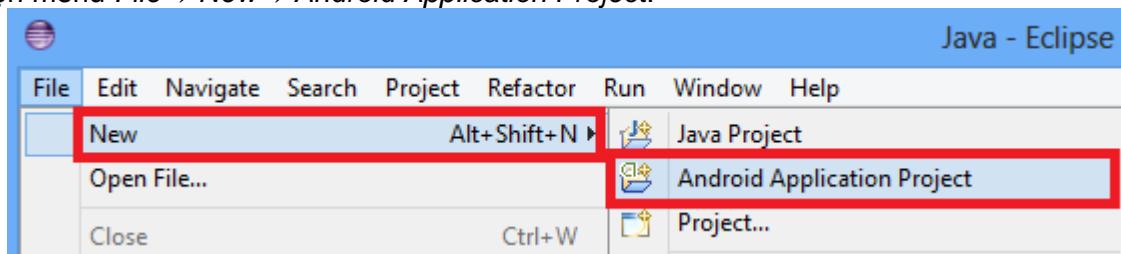
1.2.1. Tạo mới Project

BÀI THỰC HÀNH App_01

☞ **Yêu cầu:** tạo mới 1 ứng dụng đơn giản

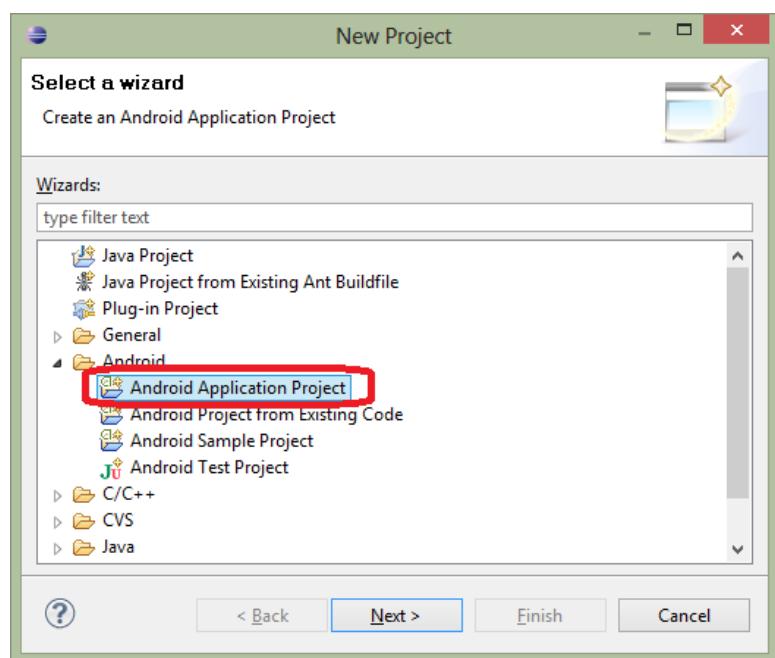
☞ **Thực hiện:** Để tạo ứng dụng, đầu tiên ta phải tạo mới 1 project theo các bước sau:

B1.- Chọn menu **File**→**New**→**Android Application Project**.



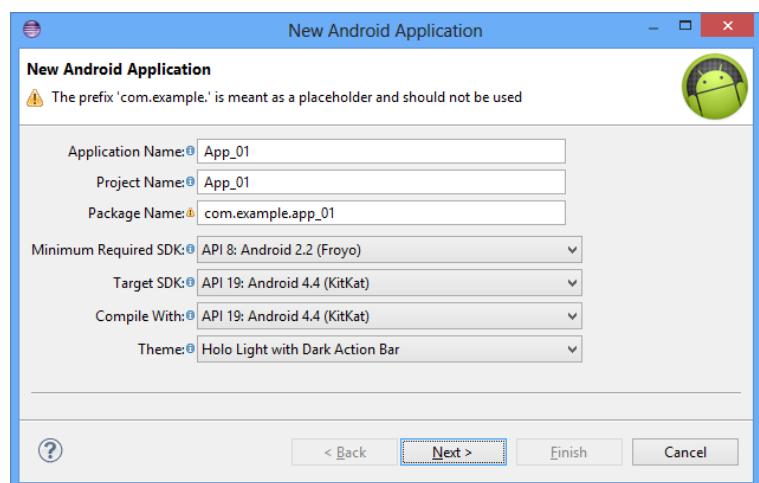
Hình 1-25 Tạo mới project

Nếu trong menu *File* → *New* chỉ có mục *Project...*, chọn mục này. Trên màn hình sẽ xuất hiện tiếp hộp thoại *New Project*, chọn *Android Application Project*. Chọn *Next*

Hình 1-26 Chọn *Android Application Project*

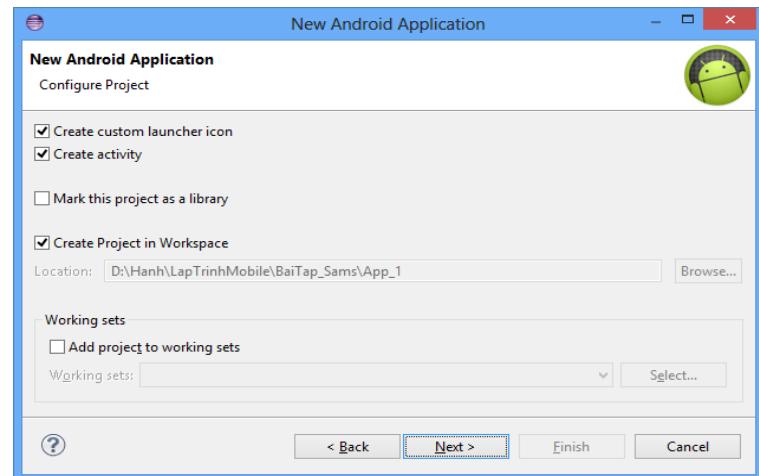
B2.- Nhập tên ***App_01*** trong textbox *Application Name*. Các mục *Project Name* và *Package Name* sẽ được tự động cập nhật. Màn hình lúc này có dạng:

☞ **Package Names:** Tên mặc định của *Package Name* trong hình 1-30 được đặt mặc định là *com.example.App_01*. Bạn có thể thay đổi tên này nhưng nên nhớ tên này là định danh (không được trùng) của ứng dụng. Thông thường, nên giữ nguyên tên do hệ thống tự phát sinh.



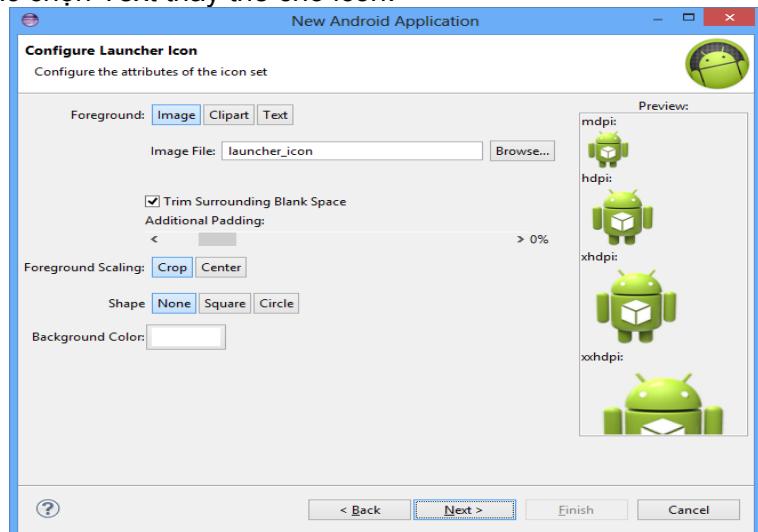
Hình 1-27 Đặt tên cho project

B3.- Click *Next*, trong màn hình kế tiếp, chọn 2 checkbox to *Create custom launcher icon* và *Create activity*.



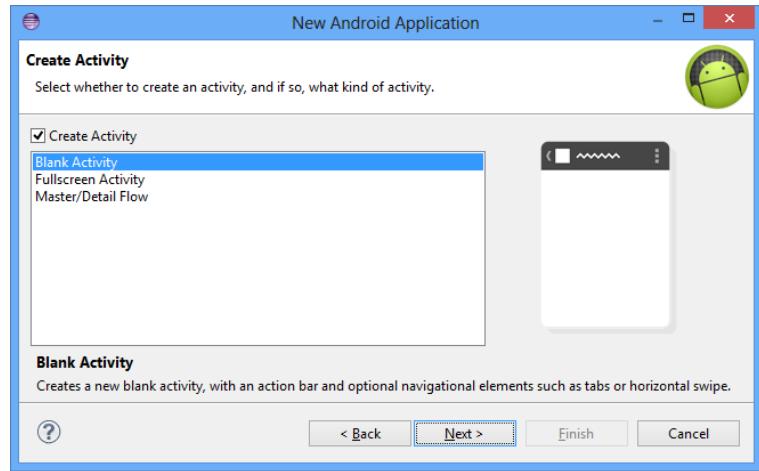
Hình 1-28 Cấu hình Project

- B4.-** Click *Next*. Màn hình cho phép chọn icon xuất hiện. Bạn có thể chấp nhận icon mặc định hoặc icon do mình tự chọn hoặc chọn *Text* thay thế cho icon.



Hình 1-29 Chọn icon cho ứng dụng

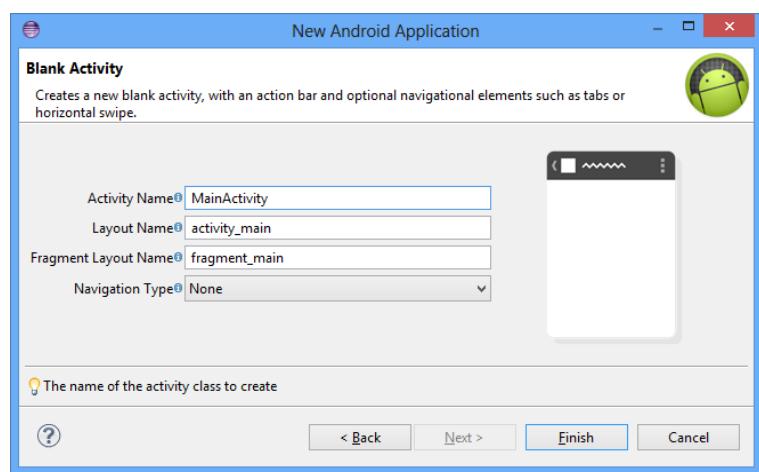
- B5.-** Click *Next*. Màn hình cho phép chọn loại màn hình (Activity) cho ứng dụng. Chọn *Blank Activity* và click *Next*.



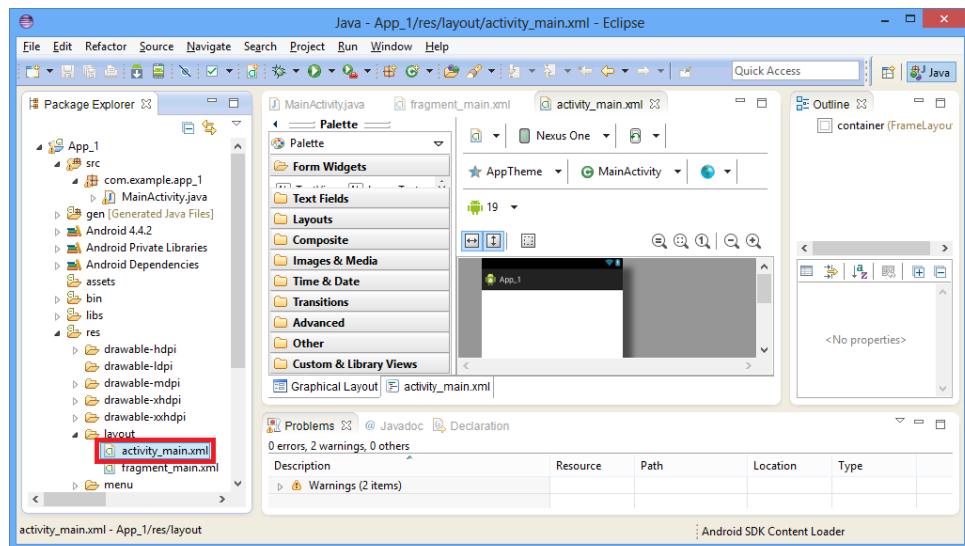
Hình 1-30 Chọn Blank Activity

- B6.-** Click *Next* để chuyển sang bước đặt tên cho file chứa màn hình chính của ứng dụng. Click *Finish* để hoàn tất việc tạo mới Project.

Trong cửa sổ *Package Explorer* (bên trái của màn hình), bạn sẽ thấy tên của project (App_01) vừa tạo. Trong đó có 1 số folder đã được tự động tạo ra. Tìm file *activity_main.xml* theo đường dẫn App_01\res\layout\ *activity_main.xml*. Double click vào file này để xuất hiện giao diện đồ họa của file XML này.



Hình 1-31 Đặt tên cho Activity



Hình 1-32 Giao diện đồ họa của file activity_main.xml

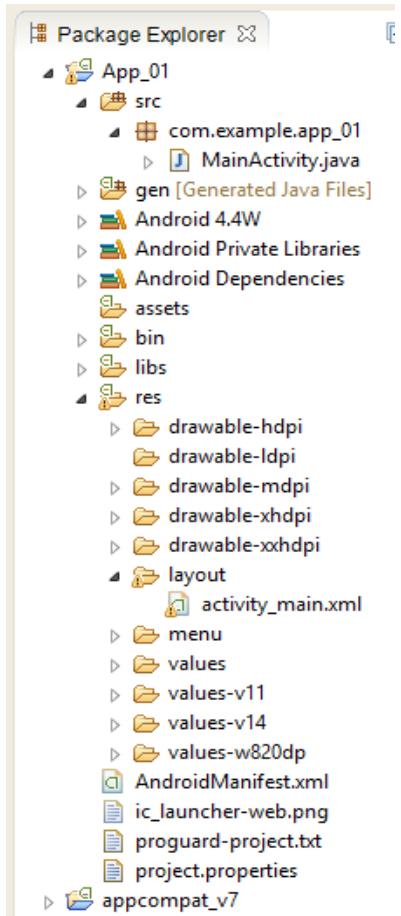
1.2.2. Cấu trúc một project

1.2.2.1. Applications

Mỗi một Android Project khi biên dịch thành công sẽ được đóng gói thành file .apk, file .apk được gọi là một Application (một ứng dụng cụ thể nào đó như ứng dụng tìm đường đi bằng xe bus, ứng dụng nhắn tin liên lạc giữa nhà trường và phụ huynh học sinh, ...)

1.2.2.2. Tổ chức các folder trong một project

- **src:** chứa source code ứng dụng (gồm file chính của ứng dụng main.xml, các package và các class)
- **gen:** chứa các file do Android tự động phát sinh (thường gặp nhất là file R.class). Cho dù bạn có xóa nó thì Android cũng lại tự tạo ra.
- **bin:** chứa ứng dụng sau khi được biên dịch, gồm các folder con:
 - **bin/classes/**: chứa file code java đã thông dịch.
 - **bin/classes.dex** : chứa các file có khả năng thực thi tạo bởi các class Java.
 - **bin/yourapp-debug.apk** hay **bin/yourapp-unsigned.apk**: chứa các file thực thi dùng để debug.
- **libs/**: chứa các Tập JAR sử dụng trong ứng dụng (thư viện của hãng thứ ba).
- **res/**: chứa các tài nguyên của ứng dụng (như các icons, file layout,...) thông qua ID, gồm các folder con:
 - **res/drawable/** : chứa file hình ảnh (PNG, JPEG,...).
 - Gồm các folder con: **Drawable-hdpi**, **Drawable-ldpi**, **Drawable-mdpi**, **Drawable-xdpi**
 - Có thể tự tạo thêm một folder cùng cấp tên là **Drawable**, chứa các file do người lập trình kéo thả trực tiếp vào trong này. Khi chương trình load các Resource sẽ tự động vào đây lấy.
 - Tùy thuộc vào độ phân giải màn hình mà chương trình tự động vào các folder -hdpi, -ldpi, -xdpi để lấy đúng dữ liệu.



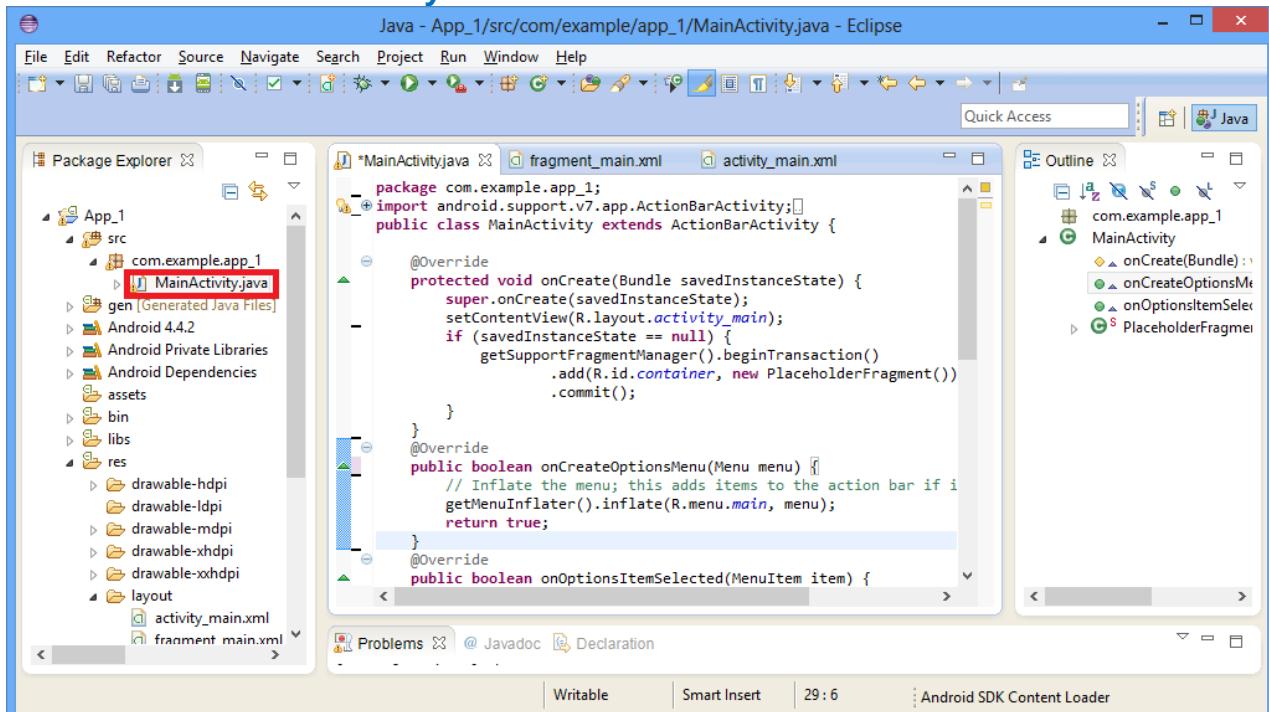
Hình 1-33 Cấu trúc một project

- **res/layout/**: Thư mục chứa các file xml thiết kế giao diện.
- **res/menu/**: Thư mục chứa các file xml đánh giá menu của ứng dụng.
- **res/raw/**: chứa các file khác (chứa thông tin account, tài nguyên raw, ...).
- **res/values/**: chứa các giá trị sử dụng trong ứng dụng được bạn định nghĩa, như các dòng ký tự (string), các màu (color), các themes...
- **res/xml/**: chứa các file XML khác cần cho ứng dụng.
- **assets/**: chứa các resource file mà ứng dụng cần dùng
- Ngoài ra còn có file thư viện.

1.2.2.3. Các file cần quan tâm:

- **MainActivity.java**: là class chứa toàn bộ source code
- **activity_main.xml**: là phần giao diện (layout)
- **AndroidManifest.xml**:
 - Mỗi ứng dụng có một file Manifest giúp khai báo thông tin về ứng dụng với hệ thống như ứng dụng gồm những Activity nào, có service nào..., xin các quyền gì, phiên bản bao nhiêu, dùng từ SDK phiên bản nào, ...
 - Nội dung:
 - Mô tả các thành phần (Activity, Service, Content Provider, Broadcast Receiver) và cách các thành phần này liên kết với nhau. Cụ thể là ứng dụng gồm những Activity nào, có service nào..., xin các quyền gì, phiên bản bao nhiêu, dùng từ SDK phiên bản nào, ...
 - Các Activity muôn được triệu gọi trong ứng dụng phải được khai báo trong file này.
 - Chỉ định các chính sách bảo mật, đơn vị kiểm chứng, các yêu cầu về nền tảng và phần cứng.
- **bin/yourapp.apk**: File cài đặt và thực thi.
- **build.xml**: file chứa mã script Ant (ant.apache.com) để biên dịch và cài đặt ứng dụng lên máy.
- **default.properties**: Tập property tạo bởi script Ant ở trên.

1.2.3. Tìm hiểu về XML Layout và Java Code



Hình 1-34 Mã lệnh của ứng dụng

- **XML layout**: Trong Android, đây được xem là cách để tạo giao diện của ứng dụng. Tuy nhiên, để dễ dàng thiết kế giao diện theo ý của người lập trình, trong tài liệu này chủ yếu sẽ sử dụng mã theo XML file.

- Java code: Để thấy được mã lệnh bằng Java của ứng dụng, trong cửa sổ Package Explorer, mở folder App_01\src\com.example.App_01 để xuất hiện file MainActivity.java. Màn hình lúc này sẽ có dạng như hình 1-34:

Trong hình 1-34, bạn có thể nhìn thấy nội dung của class *MainActivity* và được gọi là *Activity* class. Trong đó có 2 sự kiện *onCreate* và *onCreateOptionsMenu*.

- Phương thức *onCreate()*: liên kết giữa mã lệnh và file XML layout thông qua lời gọi *setContentView()* với tham số là *R.layout.activity_main*. Tên *activity_main* trong tham số chính là tên file XML của layout mà ta đã có trước đó.
- Phương thức *onCreateOptionsMenu()*: giúp phát sinh các xử lý trên các mục của menu (sẽ được giới thiệu trong bài thực hành số 7 về *ActionBar* và *Menu Navigation*).

Resource Files:

- Android chuyển đổi các files trong folder res thành tài nguyên dùng cho ứng dụng. Do file *activity_main.xml* được chứa trong folder res/layout/, vì vậy để chỉ đến tài nguyên này, ta dùng *R.layout.activity_main*.
- Cần lưu ý rằng class R được quản lý tự động, vì vậy bạn không nên tự chỉnh sửa nội dung có trong file R.java. Khi bạn đặt một tập tin vào bất cứ nơi nào trong thư mục res, các plugin thông báo các thay đổi và thêm ID tài nguyên trong R.java trong thư mục gen cho bạn. Nếu bạn loại bỏ hoặc thay đổi một tập tin tài nguyên, R.java sẽ tự động thực hiện việc đồng bộ.

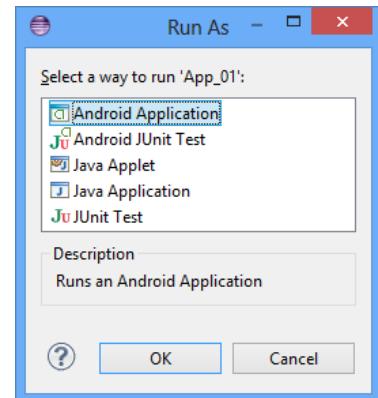
1.2.4. Chạy ứng dụng

Thực hiện các bước sau để chạy ứng dụng:

B1.- Chọn 1 trong các cách sau:

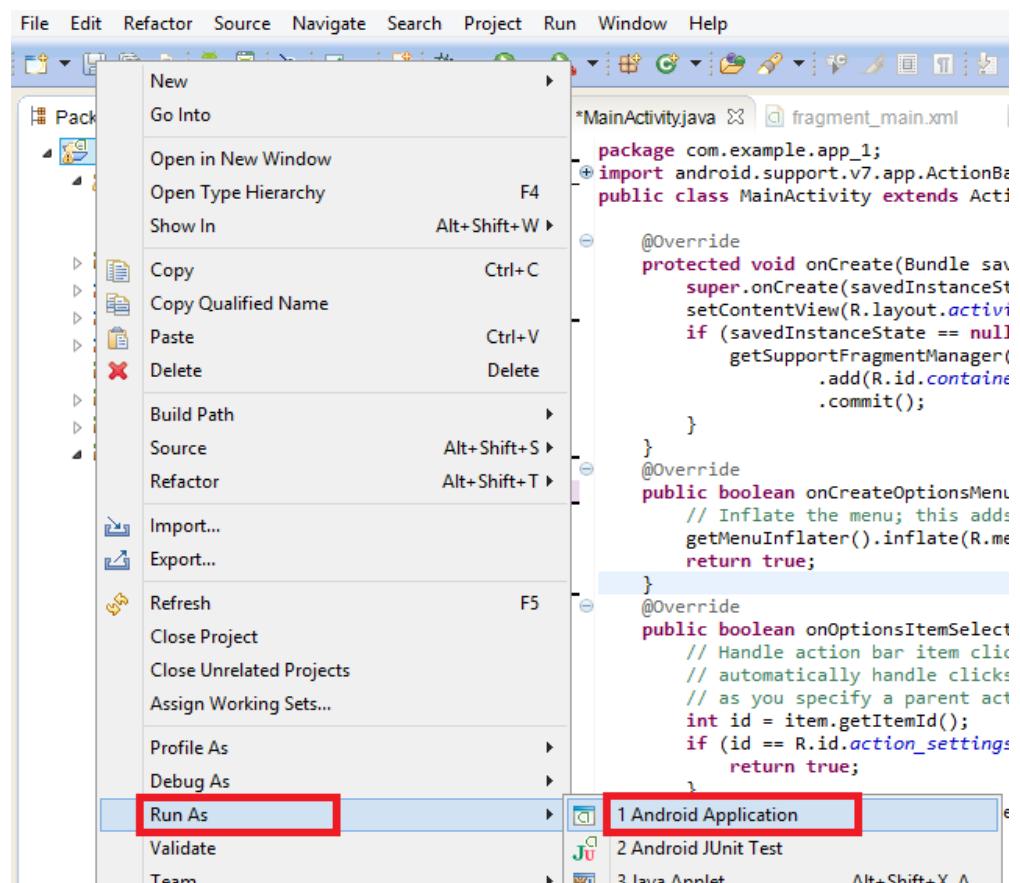
- Click phải vào tên project *App_01* trong cửa sổ Package Explorer, chọn *Run As* → *Android Application*
- Click vào icon *Run* () trên Toolbar
- Chọn menu *Run*→*Run*
- Nhấn tổ hợp phím *Ctrl+F11*

Khi chạy ứng dụng lần đầu bằng 1 trong các lựa chọn trên, sẽ xuất hiện hộp thoại *Run As*



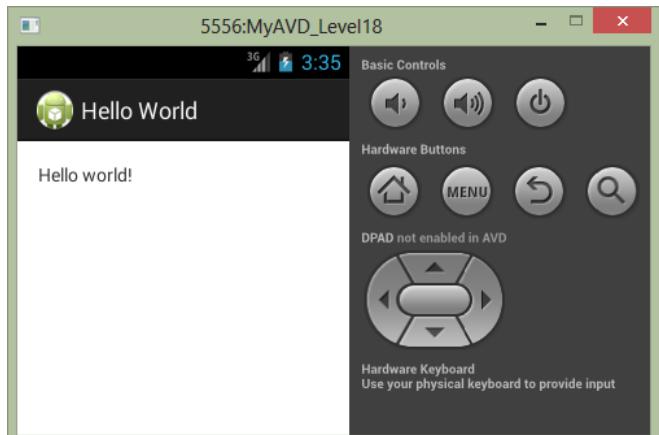
Hình 1-35 hộp thoại *Run As*

- Chọn menu *File*→*Run As*→*Android Application*



Hình 1-36 Chạy ứng dụng Android

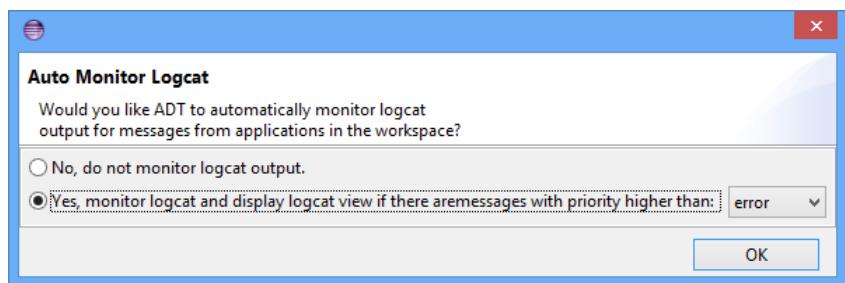
Nếu thành công màn hình sẽ có dạng:



Hình 1-37 Kết quả thực hiện trên AVD

☞ Auto Monitor Logcat:

- Khi chạy ứng dụng trên emulator, Eclipse thường sẽ hỏi bạn có muốn tự động khởi chạy monitor logcat.
- Logcat là một cửa sổ rất tiện dụng khi cần debug chương trình. Logcat ghi lại toàn bộ hoạt động của hệ điều hành như hệ điều hành đang làm gì, gọi đến cái gì, khởi chạy những gì....
- Để mở Logcat, trước tiên các bạn chọn **Window -> Open Perspective -> Debug**. Nếu không thấy option Debug thì chọn Other và tìm Debug trong cửa sổ mới hiện ra.
⇒ Vì vậy, khi đó bạn nên chọn “Yes”.



Hình 1-38 Bật chế độ logcat

1.2.5. Tùy biến ứng dụng đầu tiên vừa xây dựng

BÀI THỰC HÀNH App_01 (tiếp theo)

Yêu cầu: tạo mới 1 ứng dụng đơn giản

Trong ứng dụng trên, trên emulaor sẽ xuất hiện chuỗi “Hello world!”. Để tạo tính tương tác cho ứng dụng, giả sử bạn muốn thêm vào đó 1 khung nhập liệu và 1 button. Khi người dùng nhấn vào button, chuỗi vừa nhập vào sẽ xuất hiện ngay chính giữa màn hình.

Để nội dung người dùng nhập vào EditText sẽ chuyển vào TextView khi người dùng click vào button, cần thực hiện theo 2 bước:

- Thực hiện là liên kết giao diện người dùng với mã lệnh.
- Thêm xử lý cho button.

Thực hiện:

B7.- Cập nhật giao diện của ứng dụng

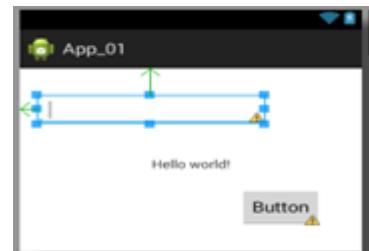
B7.1. Tìm file *activity_main.xml* trong folder *res/layout*. Đây là file đầu tiên được thấy khi ứng dụng mới được tạo lập.

B7.2. Nếu chữ Hello world! Đang không nằm giữa màn hình, click chọn và kéo thả vào giữa màn hình thiết kế (app canvas).

B7.3. Trong khung *Palette*, click vào Button (**Button**) trong folder *Form Widgets* và kéo vào góc dưới bên phải màn hình thiết kế.

B7.4. Tương tự, trong khung *Palette*, click vào Plain Text (**Plain Text**) trong folder *Text Fields* và kéo vào phía trên, bên trái của màn hình thiết kế.

B7.5. Lưu các thay đổi. Lúc này màn hình thiết kế có dạng:



Hình 1-39 Kết quả thiết kế giao diện

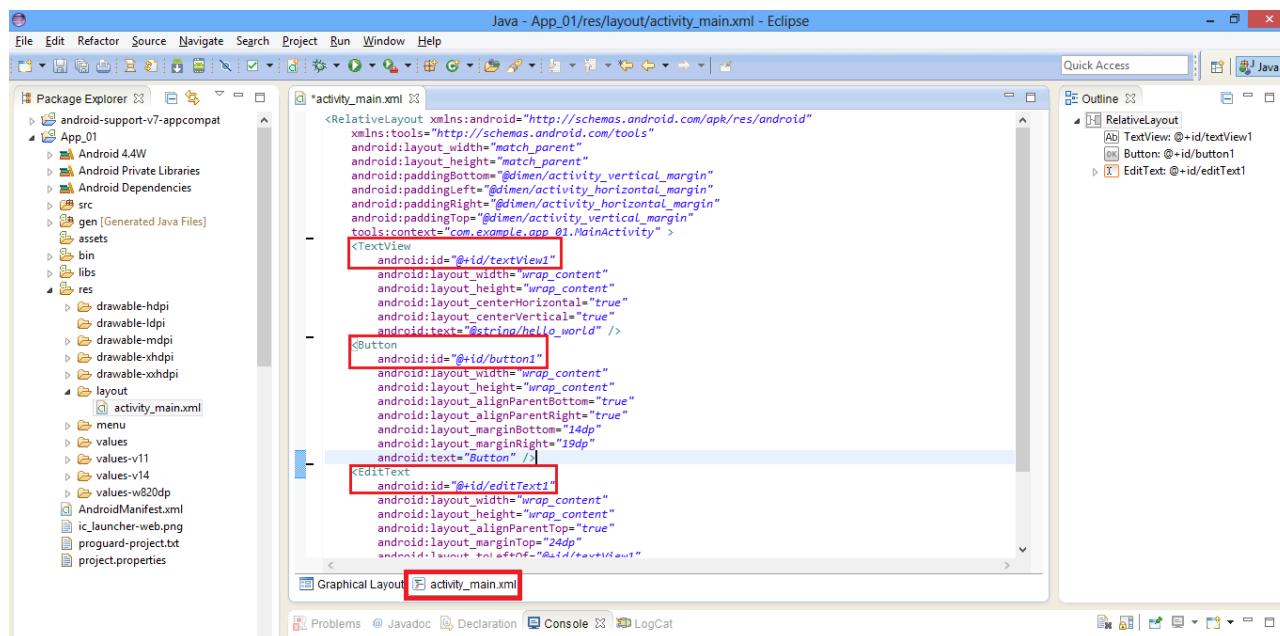
B8.- Thêm mã lệnh cho ứng dụng

Chuyển sang tab mô tả giao diện bằng XML của file *activity_main.xml*:

Thêm khai báo id của TextView hiện chữ Hello World!

`android:id="@+id/textView1"`

Lúc này mã mô tả của 3 widget, trong đó TextBox đã được đặt tên là *textView1* (đã được tạo khi tạo project) và 1 Button đã được đặt tên là *button1* cùng 1 EditText được đặt tên là *editText1*.



Hình 1-40 Nội dung file activity_main.xml được thể hiện gồm 2 tab graphic layout và xml layout

```

<TextView    android:id="@+id/textView1"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:layout_alignParentTop="true"
             android:layout_centerHorizontal="true"
             android:layout_marginTop="207dp"
             android:text="@string/hello_world" />
<Button      android:id="@+id/button1"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:layout_alignParentBottom="true"
             android:layout_alignParentRight="true"
             android:layout_marginBottom="17dp"
             android:layout_marginRight="15dp"
             android:text="Button" />
<EditText    android:id="@+id/editText1"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:layout_alignParentLeft="true"
             android:layout_alignParentTop="true"
             android:layout_marginTop="18dp"
             android:ems="10" >
    <requestFocus />
</EditText>

```

B9.- Mở file MainActivity.java trong folder App_01\src\com.example.app_01.

- Những imports sẵn có trong mã lệnh:

```

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;

```

Thêm các import cho các widget có sử dụng trong chương trình

```

import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

```

- Sau khi thêm hoàn tất, phần import sẽ như sau:

```

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;

```

```

import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

```

B10.- Thêm mã lệnh vào phương thức onCreate() để kết hợp tài nguyên (các widgets) từ layout file với mã lệnh. Việc liên kết này thực hiện được nhờ phương thức findViewById(). Từ khóa final cho biết giá trị của biến sẽ không thay đổi sau khi được gán. Nếu bỏ qua từ khóa này, Eclipse sẽ báo lỗi.

```

final EditText e = (EditText)findViewById(R.id.editText1);
final TextView t = (TextView)findViewById(R.id.textView1);
Button b = (Button)findViewById(R.id.button1);

```

B11.- Thêm phương thức xử lý sự kiện khi người sử dụng nhấn (click) trên button:

```

b.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        t.setText(e.getText());
    }
});

```

Như vậy, toàn bộ nội dung phương thức xử lý sự kiện này sẽ nằm gọn bên trong phương thức OnCreate. Lúc này nội dung sự kiện onCreate như sau:

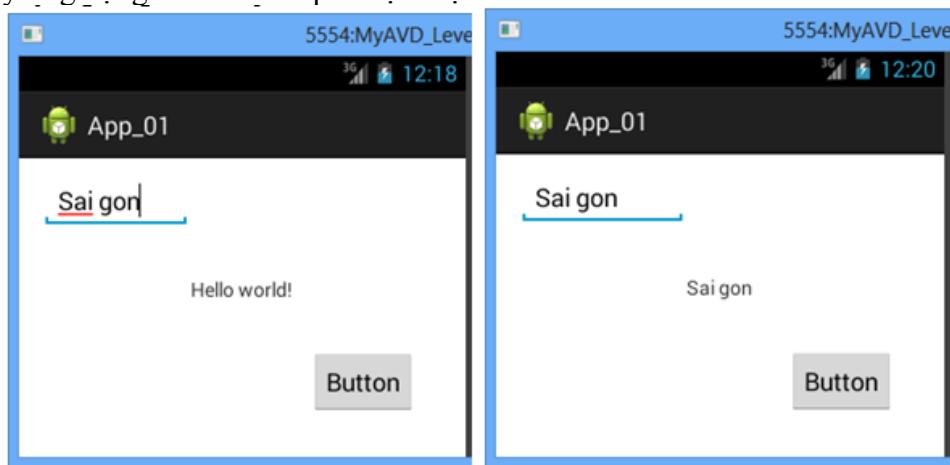
```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    final EditText e = (EditText)findViewById(R.id.editText1);
    final TextView t = (TextView)findViewById(R.id.textView1);
    Button b = (Button)findViewById(R.id.button1);

    b.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub
            t.setText(e.getText());
        }
    });
}

```

Chạy ứng dụng để xem kết quả thực hiện.



Hình 1-41 Kết quả trước và sau khi nhấn Button

1.3. TÌM HIỂU VỀ Android Project & Android Activity

1.3.1. Application

1.3.1.1. Các thành phần trong Application

Một ứng dụng Android gồm 7 thành phần cơ bản tạo thành:

- Activities
- Services
 - Là thành phần chạy bên dưới của ứng dụng.
 - Cập nhật nguồn dữ liệu cho các Activity hiển thị
 - Thường được sử dụng trong các tiến trình không yêu cầu Activity phải hoạt động.
- Content Providers
 - Nơi quản lý và chia sẻ cơ sở dữ liệu của ứng dụng.
 - Thường được dùng để chia sẻ dữ liệu giữa các ứng dụng với nhau.
 - Android cung cấp một vài Content Provider có sẵn như: media, contact details...
- Intents
 - Về khái niệm giống như một “Message” dùng để truyền thông điệp.
 - Intent lưu trữ các mô tả tác vụ và thành phần sẽ tiếp nhận.
 - Hệ thống sẽ tự động xác định thành phần sẽ thực thi tác vụ trong Intent.
 - Có thể dùng Intent để truyền thông điệp đến Activity hay Service.
- Broadcast Receivers
 - Nơi nhận và xử lý các Intent.
 - Broadcast Receiver tự động kích hoạt ứng dụng để phản hồi lại Intent được gửi tới.
- Widgets
 - Thành phần ứng dụng ảo được thể hiện ngoài màn hình chủ của thiết bị.
 - Là một dạng đặc biệt của **Broadcast Receiver**.
 - Cho phép người dùng tương tác như một Activity.
- Notification
 - Cho phép gửi các thông báo đến người dùng mà không cần thể hiện Activity.



Hình 1-42 Kết quả trước và sau khi nhấn Button



Hình 1-43 Notification

1.3.1.2. Quyền ưu tiên của ứng dụng

- Android quản lý các ứng dụng dựa trên độ ưu tiên.
- Nếu hai ứng dụng có cùng trạng thái thì ứng dụng nào đã chạy lâu hơn sẽ có độ ưu tiên thấp hơn.
- Nếu ứng dụng đang chạy một **Service** hay **Content Provider** do một ứng dụng khác hỗ trợ thì sẽ có cùng độ ưu tiên với ứng dụng đó.
- Các ứng dụng sẽ bị đóng mà không có sự báo trước.

1.3.2. Giới thiệu activity

- Mỗi activity đại diện cho một màn hình ứng dụng (giống như một Form trong .NET), giúp người sử dụng tương tác với ứng dụng như gọi điện thoại, xem ảnh, gửi email, tìm kiếm 1 địa điểm trên bản đồ,
- Trong mỗi activity thường sử dụng các **View** để tạo giao diện đồ họa (User Interface) cho ứng dụng.
- Thông thường trong một ứng dụng (Application) sẽ có một hoặc nhiều Activity.
- Một activity có thể mang nhiều dạng khác nhau:
 - Cửa sổ toàn màn hình (full screen window)
 - Cửa sổ floating (với windowsIsFloating)

- Cửa sổ nằm lồng bên trong 1 activity khác (với ActivityGroup).

1.3.3. Tạo mới 1 activity

- Một activity được tạo mới mở rộng từ class Activity của Android.
- Minh họa:

```
public class ActivityA extends ActionBarActivity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_layout);
    }
}
```

1.3.4. Khai báo activity trong file AndroidManifest.xml

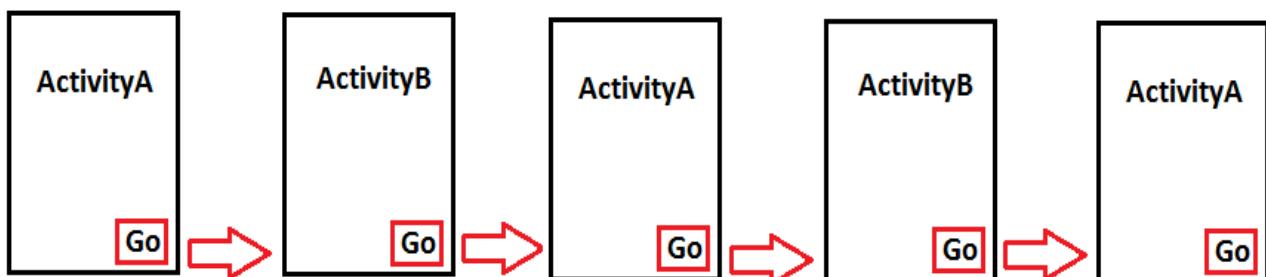
- Để một Activity triệu gọi được trong ứng dụng thì bắt buộc Activity đó phải được khai báo trong file *AndroidManifest.xml* với thẻ <activity>.

```
<application android:allowBackup="true"
            android:icon="@drawable/ic_launcher"
            android:label="@string/app_name"
            android:theme="@style/AppTheme" >
    <activity android:name=".ActivityA"
              android:label="@string/app_name" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".ActivityB"
              android:label="Activity B">
    </activity>
</application>
```

1.3.5. Khởi chạy một activity

Một activity có thể khởi chạy bằng nhiều cách. Trong phần này sẽ hướng dẫn cách khởi chạy 1 activity từ 1 activity khác.

Mục đích của ta là dùng *ActivityA* để khởi chạy *ActivityB* rồi từ *ActivityB* lại cho khởi chạy *ActivityA*. Việc dùng activity này để khởi chạy activity kia có thể thực hiện được nhiều lần theo hình minh họa sau:



Hình 1-44 Gọi activity khác khi nhấn button “Go”

Thông thường, ta sẽ dùng 1 activity này để khởi chạy 1 activity khác, với mỗi activity có một giao diện riêng của mình. Tuy nhiên trong bài thực hành này sẽ sử dụng cùng 1 layout cho cả 2 activity.

1.3.5.1. Khởi chạy một activity từ một activity khác

BÀI THỰC HÀNH App_02

-  **Yêu cầu:** tạo 1 ứng dụng gồm 2 activity, trong đó activity sẽ chứa 1 button, khi được click sẽ mở activity thứ 2.
-  **Thực hiện**

B1: Trong Eclipse, chọn menu *File* → *New* → *Android Application Project*.

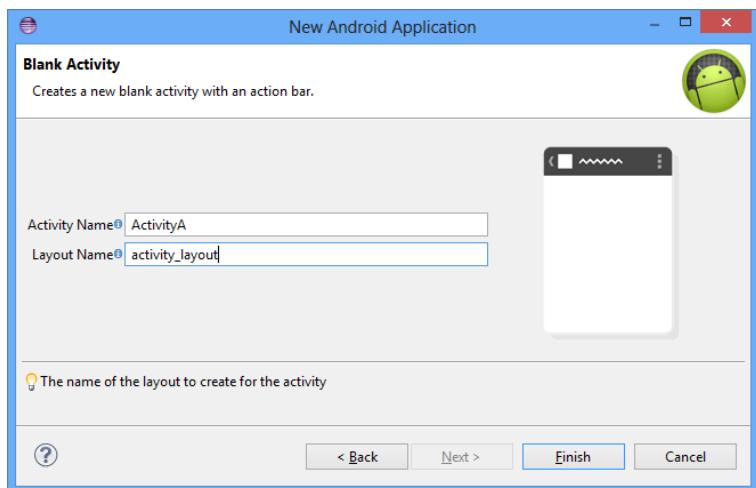
B2: Đặt tên cho project là *App_02*

B3: Chấp nhận icon mặc định và chọn *Blank Activity*.

B4: Sử dụng tên **ActivityA** cho *Activity Name* và tên **activity_layout** cho *Layout Name*.

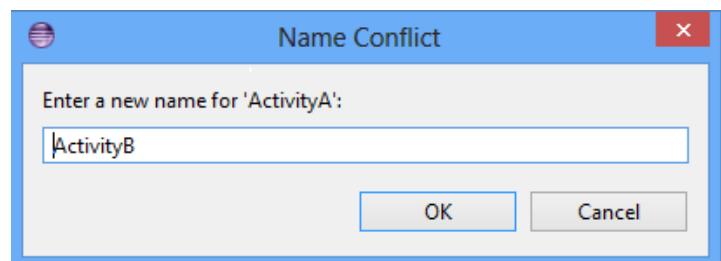
B5: Click *Finish* để hoàn tất tạo project.

Trong project *App_02* hiện chứa 1 activity và 1 layout. Layout vừa có chứa 1 *TextView* hiển thị chuỗi “Hello World!”



Hình 1-45 một trong các bước tạo ứng dụng Android

B6: Để tạo activity thứ 2, tìm file *ActivityA* trong folder *src\com.example.app_02*. Right click vào file *ActivityA*, chọn *Copy*. Right click vào folder *src\com.example.app_02*, chọn *Paste*. Xuất hiện hộp thoại để đặt tên cho activity mới. Ta đặt tên cho activity thứ 2 này là *ActivityB*.



Hình 1-46 Đổi tên cho activity vừa copy

B7: Bổ sung thiết kế cho file layout:

- Mở file *activity_layout.xml*, kéo thả 1 button vào góc dưới bên phải của màn hình thiết kế (canvas).

- Chuyển sang tab mã XML của file *activity_layout.xml*
 - Bổ sung 2 dòng sau vào trong cặp thẻ của *TextView*, mục đích gán id cho widget *TextView* đang chứa chuỗi “Hello World!” và thay đổi kích thước font chữ
 `android:id="@+id/textView1"`
 `android:textSize="40dp"`
 - Đổi chuỗi hiển thị của *TextView* từ “Hello World!” sang “Activity A”
 `android:text="Activity A"`
 - Đổi chuỗi hiển thị của button từ “Button” sang “Go”
 `android:text="Go"`

Sau khi chỉnh sửa hoàn tất, nội dung file *activity_layout.xml* sẽ có nội dung tương tự như sau:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.app_02.ActivityA" >
    <Button android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</RelativeLayout>
```

```

        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_marginBottom="19dp"
        android:layout_marginRight="24dp"
        android:text="Go" />
    <TextView    android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/button1"
        android:layout_alignParentLeft="true"
        android:layout_marginBottom="157dp"
        android:layout_marginLeft="26dp"
        android:text="Activity A"
        android:textSize="40dp" />
</RelativeLayout>
```

B8: Bổ sung mã lệnh cho file ActivityA.java

- Bổ sung các class sẽ sử dụng trong chương trình

```

import android.content.Intent;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.view.View;
```

- Thêm các dòng lệnh sau vào hàm `OnCreate`, ngay sau phương thức `setContentView`:

```

Button b = (Button)findViewById(R.id.button1);
b.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        Intent intent = new Intent(ActivityA.this, ActivityB.class);
        startActivity(intent);
    }
});
```

Sau khi bổ sung hoàn tất, nội dung file ActivityA.java có dạng:

```

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.content.Intent;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.view.View;

public class ActivityA extends ActionBarActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_layout);
        Button b = (Button)findViewById(R.id.button1);
        b.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                Intent intent = new Intent(ActivityA.this, ActivityB.class);
                startActivity(intent);
            }
        });
    }
    // Phần chứa nội dung các phương thức khác sẵn có trong class
}
```

Trong đó:

- Dòng lệnh `Intent intent = new Intent(ActivityA.this, ActivityB.class);`
 - Tham số thứ nhất là activity hiện tại sẽ gọi activity khác, tham số thứ 2 là tên activity được gọi.

- Từ khóa this được dùng ở đây ám chỉ class OnClickListener() ở dòng lệnh b.setOnClickListener(new OnClickListener() {...}) được định nghĩa liền trước đó.
- Dòng lệnh `startActivity(intent);` giúp khởi chạy activity có tên chứa trong tham số thứ 2

B9: Bổ sung ActivityB vào file *AndroidManifest.xml*

Do trong Android quy định tất cả những activity muốn chạy được trong ứng dụng đều phải được khai báo trong file *AndroidManifest.xml* và activity nào được chứa trong cặp thẻ `<intent-filter> ... </intent-filter>` sẽ được chạy đầu tiên.

Nếu quên thêm khai báo Activity vào file *AndroidManifest.xml*, khi chạy ứng dụng sẽ nhận được nội dung lỗi có dạng như sau:

```
Unable to find explicit activity class {XXXXXXXXXX.app_02.ActivityB}; have you
declared this activity in your AndroidManifest.xml?
```

Có 2 cách thêm activity vào file *AndroidManifest.xml*:

- Cách 1:** Mở file *AndroidManifest.xml*, chọn tab *AndroidManifest.xml*, bổ sung đoạn lệnh sau vào sau cặp thẻ `<activity> ... </activity>` của *ActivityA*

```
<activity
    android:name=".ActivityB"
    android:label="Activity B">
</activity>
```

Sau khi thêm xong nội dung nằm giữa cặp thẻ `<application> ... </application>` trong file *AndroidManifest.xml* có dạng.

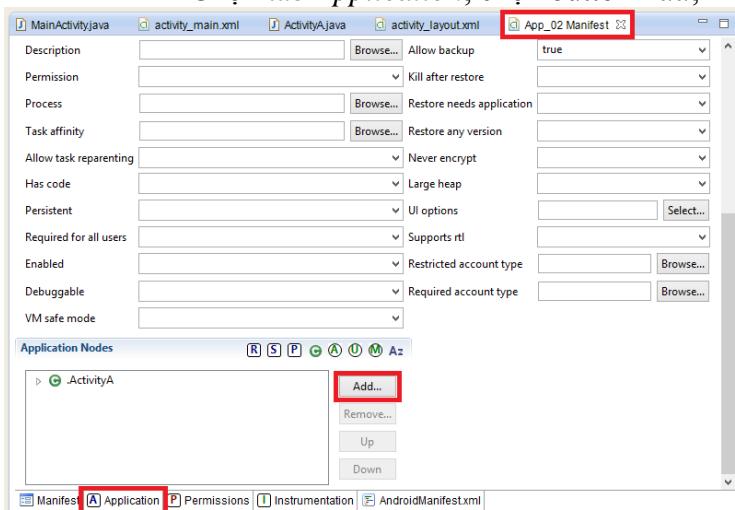
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.app_02"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="21" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".ActivityA"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".ActivityB"
            android:label="Activity B">
        </activity>
    </application>
</manifest>
```

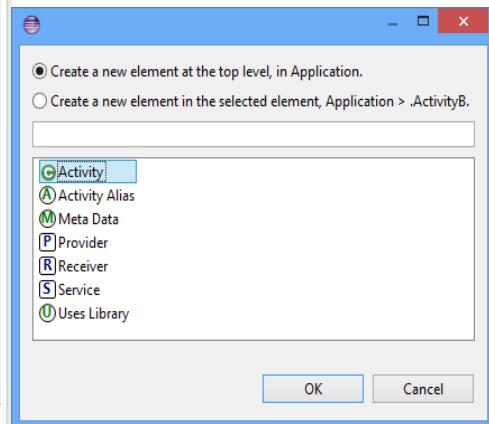
Hình 1-47 Thêm activity vào nội dung file *AndroidManifest* theo cách 1

- Cách 2:** cũng trong file *AndroidManifest.xml*,

- Chọn tab Application, chọn button Add, nhập tên file .ActivityB

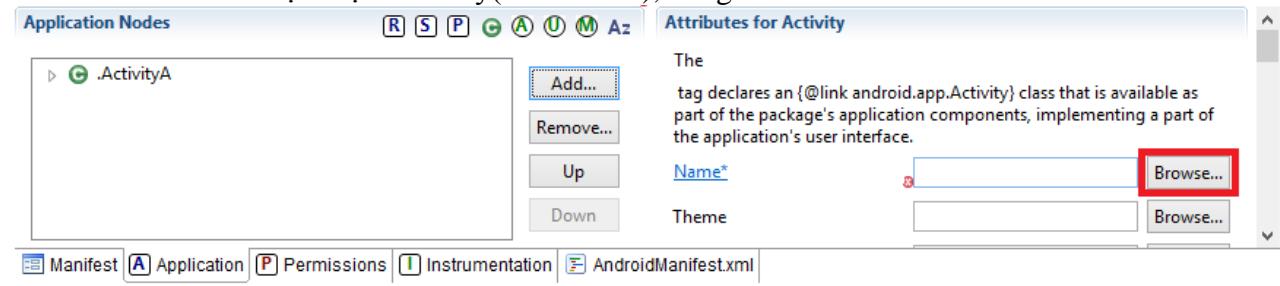


Hình 1-48 Thêm activity vào file AndroidManifest



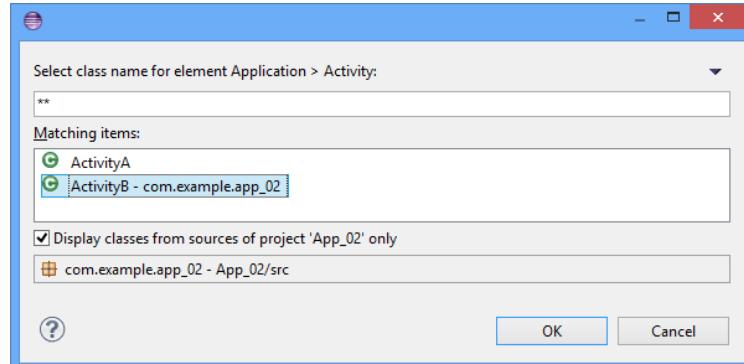
Hình 1-49 Chọn Activity. Xong click OK

- Xuất hiện hộp thoại sau (hình 1-50):
- Chọn mục Activity(), xong click OK để trở về màn hình trước đó.



Hình 1-50 Chọn button “Browse” để mở hộp thoại chứa tên đối tượng cần thêm vào nội dung file AndroidManifest

- Click button Browse bên phải, phía dưới màn hình (ngang với chuỗi Name*) để mở hộp thoại sau:



Hình 1-51 Chọn tên đối tượng cần thêm vào nội dung file AndroidManifest

- Chọn ActivityB, xong nhấn OK để hoàn tất việc bổ sung activity vào file AndroidManifest.xml

B10: Bổ sung mã lệnh cho ActivityB: mở file ActivityB.java,

- Bổ sung các class sẽ sử dụng:

```
import android.content.Intent;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
```

- Bổ sung mã lệnh cho file này như sau:

```

TextView t = (TextView)findViewById(R.id.textView1);
t.setText("This is Activity B");
Button b = (Button)findViewById(R.id.button1);
b.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        Intent intent = new Intent(ActivityB.this, ActivityA.class);
        startActivity(intent);
    }
});

```

Lúc này, file *ActivityB.java* có dạng như sau:

```

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.content.Intent;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
public class ActivityB extends ActionBarActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_layout);
        TextView t = (TextView)findViewById(R.id.textView1);
        t.setText("This is Activity B");
        Button b = (Button)findViewById(R.id.button1);
        b.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                Intent intent = new Intent(ActivityB.this, ActivityA.class);
                startActivity(intent);
            }
        });
    }
    // phần chứa nội dung các phương thức có trong class ActivityB
}

```

Tương tự như trong *activityA*, hai dòng lệnh

```

Intent intent = new Intent(ActivityB.this, ActivityA.class);
startActivity(intent);

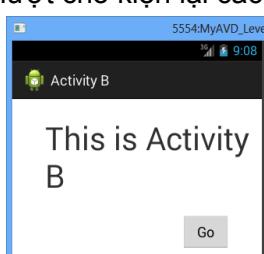
```

Sẽ cho khởi chạy *ActivityA* từ *ActivityB*

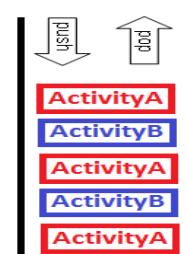
B11: Cho chạy ứng dụng, click button Go trên mỗi activity vài lần để thấy việc chuyển đổi giữa các activity.

Sau khi thực hiện 1 số lần chuyển từ *ActivityA* sang *ActivityB* rồi lại từ *ActivityB* sang

ActivityA, nếu bạn chọn button *Back* (☞ hoặc ↙ tùy thuộc emulator) sẵn có trên emulator, ứng dụng sẽ lẩn lượt cho kiện lại các Activity theo thứ tự ngược lại.



Hình 1-52 ActivityB sẽ xuất hiện khi nhấn button "Go" trên ActivityA

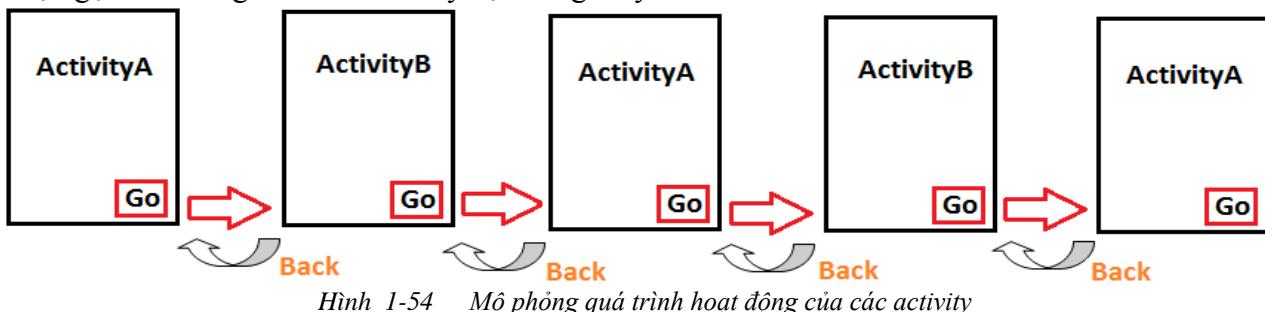


Hình 1-53 Mô phỏng Back Stack

1.3.5.2. Tìm hiểu về Back Stack

Cách xử lý cho button Back như vừa thấy ở trên là cách xử lý mặc định trong Android. Khi activity đầu tiên khởi chạy, activity này sẽ được đưa vào “back stack” của ứng dụng. “back stack” là

một stack chứa các activity, vì vậy mỗi khi bạn click vào button Go, các activity được gọi kế tiếp sẽ lần lượt được đưa vào stack. Một activity có thể được đưa vào stack nhiều lần khác nhau. Activity được gọi cuối cùng chính là activity bạn đang thấy trên màn hình của emulator.



Với cách xử lý mặc định như trên, mỗi activity khi được gọi sẽ được tạo ra 1 thẻ hiện riêng, độc lập.

Nếu bạn muốn mỗi activity chỉ được tạo 1 thẻ hiện duy nhất, bạn cần bổ sung lệnh sau vào sau lệnh khởi tạo intent trong cả 2 activity A và B:

```
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK|Intent.FLAG_ACTIVITY_CLEAR_TOP);
```

Như vậy, sau khi bổ sung mã lệnh, nội dung phương thức onCreate() của ActivityA sẽ như sau:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_layout);
    Button b = (Button)findViewById(R.id.button1);
    b.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            Intent intent = new Intent(ActivityA.this, ActivityB.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK|Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(intent);
        }
    });
}
```

Và nội dung phương thức onCreate() của ActivityB sẽ như sau:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_layout);
    TextView t = (TextView)findViewById(R.id.textView1);
    t.setText("This is Activity B");
    Button b = (Button)findViewById(R.id.button1);
    b.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            Intent intent = new Intent(ActivityB.this, ActivityA.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK|Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(intent);
        }
    });
}
```

Nhờ vậy, nếu xem stack là một tập các phiếu được đê chồng lên nhau. Khi phiếu nào nằm trên cùng thì activity có tên trên phiếu đó sẽ được hiển thị cho người dùng nhìn thấy. Khi tập phiếu này hết (trống), ứng dụng xem như kết thúc.

1.3.5.3. Activity nào sẽ được gọi đầu tiên khi ứng dụng được khởi chạy?

Khi khởi chạy ActivityB từ ActivityA, bạn đã tạo ra class ActivityB rồi thêm nó vào file *AndroidManifest.xml*. Sau đó bạn tạo ra 1 intent của ActivityA và khởi chạy ActivityB.

Câu hỏi đặt ra là ta có 2 activity nhưng tại sao ActivityA luôn là activity đầu tiên được chạy?

Khi bạn tạo mới project, 2 file *ActivityA.java* và *AndroidManifest.xml* cũng được tự động tạo ra. Theo mặc định, trong file *AndroidManifest.xml* phải chỉ định activity nào chạy đầu tiên khi ứng dụng khởi chạy thông qua nội dung chứa trong cặp thẻ `<intent-filter> ... </intent-filter>`, và lúc này toàn bộ project chỉ mới có duy nhất 1 *ActivityA* nên *ActivityA* được chọn. Do đó, ta thấy nội dung trong file *AndroidManifest.xml* có đoạn như sau:

```
<activity
    android:name=".ActivityA"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Nội dung chứa bên trong cặp thẻ `<intent-filter> ... </intent-filter>` là 1 tập các quy định. Khi ứng dụng khởi chạy, hệ thống Android sẽ kiểm tra các quy định này để biết activity nào sẽ được chạy đầu tiên. Nhờ vậy *ActivityA* luôn được khởi chạy trước.

1.3.6. Chuyển thông tin giữa hai Activities

- Hai activity trong App02 ở trên chỉ đơn thuần khởi chạy các activity khác. Trong thực tế, phần lớn các ứng dụng lại có nhu cầu chuyển dữ liệu giữa các activity. Việc chuyển dữ liệu này thường được thực hiện dưới 2 dạng:
 - Chuyển dữ liệu từ activity này đến activity khác, ví dụ ActivityA chuyển dữ liệu cho ActivityB
 - Activity thứ nhất (ActivityA) yêu cầu và nhận dữ liệu của activity thứ 3 (ví dụ ActivityC).

1.3.6.1. Sử dụng Extras và Bundles

Để khởi chạy 1 activity, bạn sử dụng Intent (android.content.Intent). Dữ liệu có thể được thêm vào intent đã được dùng để khởi chạy activity.

Khi activity được khởi chạy, dữ liệu được thêm vào intent dưới dạng dữ liệu mở rộng (“extra” data). Kiểu dữ liệu như kiểu chuỗi, số nguyên, logic có thể được chuyển như là tính năng bổ sung. Gói dữ liệu và các kiểu dữ liệu phức hợp khác cũng có thể được chuyển đi.

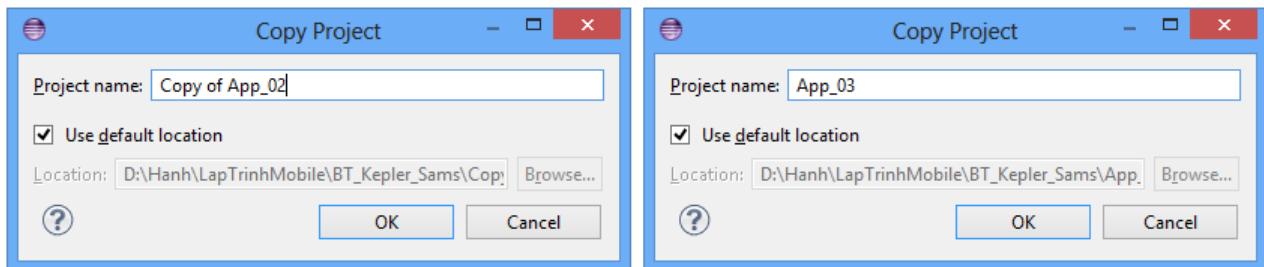
BÀI THỰC HÀNH App_03

Yêu cầu: tạo 1 ứng dụng gồm 2 activity, trong đó activity sẽ chứa 1 button, khi được click sẽ chuyển dữ liệu cho activity thứ 2. Khi activity thứ 2 được mở sẽ hiển thị thông tin nhận được lên màn hình.

Thực hiện

B1: Tạo project mới để thực hiện việc chuyển dữ liệu giữa các activity. Do project mới tạo có nội dung gần giống với App_02 nên ta thực hiện copy project bằng cách:

- Right click vào project App_02, chọn Copy
- Right click lần thứ hai vào project App_02, chọn Paste. Xuất hiện hộp thoại sau, sửa tên project mới copy thành App_03. Xong click OK.



Hình 1-55 Đổi tên project vừa copy

- Tìm và mở file App_03\res\values\string.xml, đổi tên ứng dụng từ App_02 thành App_03 như sau:
`<string name="app_name">App_03</string>`

Trong minh họa tiếp theo, ActivityA sẽ được hiệu chỉnh lại để có thể chuyển dữ liệu cho ActivityB khi ActivityB được khởi chạy và ActivityB sẽ xử lý rồi hiển thị dữ liệu vừa nhận được.

Để thực hiện được việc chuyển dữ liệu ActivityA sang ActivityB, ta dùng Bundle (trong android.os.Bundle). Bundle chứa dữ liệu được định nghĩa thông qua cặp khóa-giá trị (key-value)

B2: Bổ sung các mã lệnh sau vào nội dung file ActivityA.java, ngay sau lệnh `intent.setFlags(...);`:

```

intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TOP);
// khai báo đối tượng Bundle
Bundle b = new Bundle();
// giá trị "Tý" được thêm vào Bundle b với key được đặt là sTen
b.putString("sTen", "Tý");
// thêm Bundle b vào intent
intent.putExtra("bundleExtra", b);
// thêm dữ liệu vào intent
intent.putExtra("sDanhGia", "Khá");
intent.putExtra("bPhaiNu", false);
intent.putExtra("iDiem", 8);

```

Minh họa sự kiện onCreate của file ActivityA.java sau khi bổ sung mã lệnh:

```

import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.content.Intent;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.view.View;
public class ActivityA extends ActionBarActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_layout);
        Button b = (Button)findViewById(R.id.button1);
        b.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                Intent intent = new Intent(ActivityA.this, ActivityB.class);
                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK|Intent.FLAG_ACTIVITY_CLEAR_TOP);
                // khai báo đối tượng Bundle
                Bundle b = new Bundle();
                // giá trị "Tý" được thêm vào Bundle b với key được đặt là sTen
                b.putString("sTen", "Tý");
                // thêm Bundle b vào intent
                intent.putExtra("bundleExtra", b);
                // thêm dữ liệu vào intent
                intent.putExtra("sDanhGia", "Khá");
                intent.putExtra("bPhaiNu", false);
                intent.putExtra("iDiem", 8);
                startActivity(intent);
            }
        });
    }
    // các phương thức sẵn có trong class ActivityA sẽ nằm ngay sau dòng ghi chú này
    // . . .
}

```

B3: Bổ sung mã lệnh cho file ActivityB.java:

- Thực hiện xóa mã lệnh của các widget editText1 và button1 của ứng dụng trước do không còn sử dụng trong minh họa này
- Bổ sung các lệnh sau vào cuối sự kiện onCreate để xử lý dữ liệu nhận được từ ActivityA.

```

// mã lệnh bổ sung mới cho phương thức
Intent intent = getIntent();
Bundle bundle = intent.getBundleExtra("bundleExtra");
String Ten = bundle.getString("sTen");
String DanhGia = intent.getStringExtra("sDanhGia");
Boolean PhaiNu = intent.getBooleanExtra("bPhaiNu", false);
int Diem = intent.getIntExtra("idiem", 8);
String str;
if (PhaiNu==false)
    str="Anh ";
else
    str="Chị ";
t.setText(str + Ten + " nhận xét Android " + DanhGia + ", điểm: " + Diem);

```

Sau khi bổ sung hoàn tất, mã lệnh trong file ActivityB có dạng:

```

import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.content.Intent;
import android.widget.TextView;

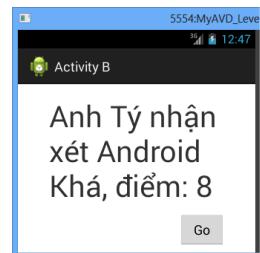
```

```

public class ActivityB extends ActionBarActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_layout);
        TextView t = (TextView) findViewById(R.id.textView1);
        // mã lệnh bổ sung mới cho phương thức
        Intent intent = getIntent();
        Bundle bundle = intent.getBundleExtra("bundleExtra");
        String Ten = bundle.getString("sTen");
        String DanhGia = intent.getStringExtra("sDanhGia");
        Boolean PhaiNu = intent.getBooleanExtra("bPhaiNu", false);
        int Diem = intent.getIntExtra("idiem", 8);
        String str;
        if (PhaiNu==false)
            str="Anh ";
        else
            str="Chị ";
        t.setText(str + Ten + " nhận xét Android " + DanhGia + ", điểm: " + Diem);
    }
    // các phương thức sẵn có trong class ActivityB sẽ nằm ngay sau dòng ghi chú này
    // . .
}

```

Kết quả khi thực hiện chương trình, giao diện sẽ có dạng:



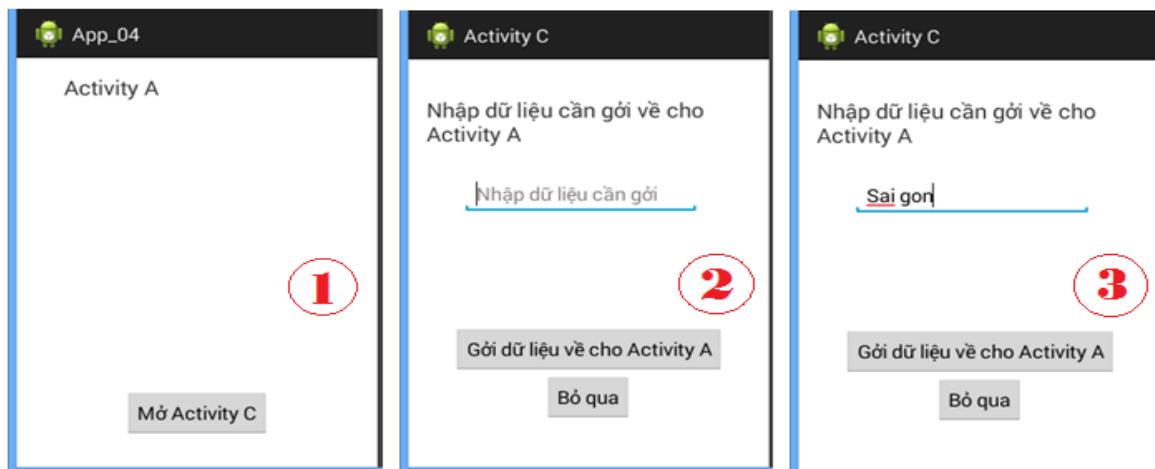
Hình 1-56 ActivityB hiển thị dữ liệu nhận được

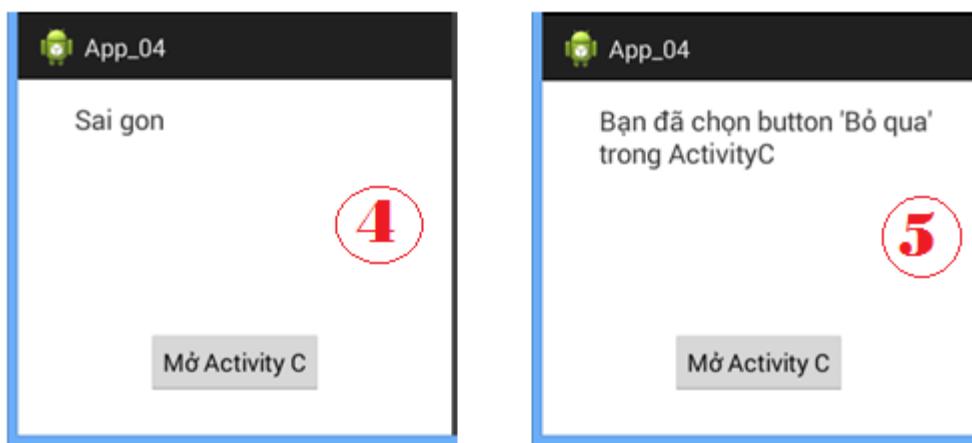
1.3.6.2. Sử dụng phương thức startActivityForResult để nhận kết quả trả về

Trong thực tế, đa phần các ứng dụng cần trao đổi thông tin cho nhau ví dụ như gửi và nhận hình ảnh từ 2 điện thoại

BÀI THỰC HÀNH App_04

☞ Yêu cầu:





Hình 1-57 (gồm 5 hình) minh họa quá trình hoạt động của ứng dụng

Trong bài thực hành này, bạn sẽ xây dựng ứng dụng, trong đó activityA kích hoạt ActivityC và yêu cầu kết quả trả về từ ActivityC nhờ vào phương thức `startActivityForResult()` và xử lý kết quả trả về. Trong hình sau mô phỏng quá trình hoạt động của ứng dụng:

Hình 1-57-① Ứng dụng khi khởi chạy

Hình 1-57-② ActivityC khi mới được gọi

Hình 1-57-③ ActivityC khi người dùng bấm sang thông tin

Hình 1-57-④ ActivityA khi người dùng chọn “Gửi dữ liệu cho ActivityA”

Hình 1-57-⑤ ActivityA khi người dùng chọn “Bỏ qua”

☞ Thực hiện

B1: Để tiết kiệm thời gian, ta tạo App04 bằng cách tương tự như đã thực hiện với App_03:

- Copy và paste App_03 vào của sổ Package Explorer, đổi tên project là App_04.
- Tìm và mở file App_04\res\values\string.xml, đổi tên ứng dụng từ App_03 thành App_04 như sau:

```
<string name="app_name">App_04</string>
```

B2: Copy và paste file App_04\res\layout\activity_layout.xml với tên file mới là activity_c_layout.xml

B3: Sử dụng các widget có trong thanh Palette để thêm 1 edittext và 2 button (1 là “Gửi dữ liệu về cho Activity A” và 1 là “Bỏ qua”) vào activity_c_layout.xml

Với mã xml trong file activity_c_layout.xml có dạng:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    ...
    <EditText    android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/textView1"
        android:layout_below="@+id/textView1"
        android:layout_marginLeft="32dp"
        android:layout_marginTop="27dp"
        android:ems="10"
        android:hint="Nhập dữ liệu cần gửi">
        <requestFocus />
    </EditText>
    <TextView    android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_marginTop="21dp"
        android:text="Nhập dữ liệu cần gửi về cho Activity A"
        android:textSize="20dp" />
    <Button      android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
```

```

        android:layout_centerHorizontal="true"
        android:layout_marginBottom="30dp"
        android:text="Bỏ qua" />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/button1"
        android:layout_centerHorizontal="true"
        android:text="Gửi dữ liệu về cho Activity A" />
</RelativeLayout>
```

B4: Tạo mới ActivityC (có thể copy từ ActivityA cho nhanh)

B5: Cập nhật lại nội dung file ActivityC.java

- Bổ sung class Activity

```
import android.app.Activity;
```

- Sửa tên layout trong phương thức setContentView:

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

- Bổ sung các lệnh gắn kết các widget với các biến của chương trình

```

public class ActivityC extends ActionBarActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_c_layout);
        final EditText e = (EditText)findViewById(R.id.editText1);
        Button send = (Button)findViewById(R.id.button2);
        send.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                // mã lệnh của button "Gửi"
            }
        });
        Button cancel = (Button)findViewById(R.id.button1);
        cancel.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                // mã lệnh của button "Bỏ qua"
            }
        });
    }
    // các phương thức sẵn có trong class ActivityC sẽ nằm ngay sau dòng ghi chú này
    // . .
}
```

B6: Bổ sung mã lệnh cho button “gửi dữ liệu” trong ActivityC.java

Button “gửi dữ liệu” có nhiệm vụ gửi dữ liệu chứa trong editText1 về cho activity đã kích hoạt ActivityC. Để thực hiện việc này, ta cần sử dụng intent

- Bổ sung lệnh sau trong phần import

```
import android.app.Activity;
```

- Bổ sung mã lệnh cho button “gửi dữ liệu”

```

// khởi tạo 1 intent
Intent result = new Intent();
// thêm dữ liệu trong editText1 vào "Data"
result.putExtra("Data", e.getText().toString());
/* sử dụng phương thức setResult với tham số RESULT_OK
để nơi nhận được biết để lấy dữ liệu*/
setResult (Activity.RESULT_OK, result);
// sử dụng phương thức finish để kết thúc activity đang hoạt động
finish();
```

B7: Bổ sung mã lệnh cho button “bỏ qua” trong ActivityC.java

Button “bỏ qua” có nhiệm vụ kết thúc activity và không gửi dữ liệu về cho activity đã kích hoạt.

```

/* sử dụng phương thức setResult với tham số RESULT_CANCELED
để nơi nhận được biết rằng không có dữ liệu gửi về*/
setResult (Activity.RESULT_CANCELED);
```

```
// sử dụng phương thức finish để kết thúc activity đang hoạt động
finish();
```

Sau khi hoàn tất, nội dung class ActivityC có dạng

```
package com.example.app_02;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.app.Activity;
import android.content.Intent;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.view.View;
public class ActivityC extends ActionBarActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_c_layout);
        final EditText e = (EditText)findViewById(R.id.editText1);
        Button send = (Button)findViewById(R.id.button2);
        send.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                // mã lệnh của button "Gởi"
                // khởi tạo 1 intent
                Intent result = new Intent();
                // thêm dữ liệu trong editText1 vào "Data"
                result.putExtra("Data", e.getText().toString());
                /* sử dụng phương thức setResult với tham số RESULT_OK
                để nơi nhận được biết để lấy dữ liệu*/
                setResult(Activity.RESULT_OK, result);
                // sử dụng phương thức finish để kết thúc activity đang hoạt động
                finish();
            }
        });
        Button cancel = (Button)findViewById(R.id.button1);
        cancel.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                // mã lệnh của button "Bỏ qua"
                /* sử dụng phương thức setResult với tham số RESULT_OK
                để nơi nhận được biết rằng không có dữ liệu gửi về*/
                setResult(Activity.RESULT_CANCELED);
                // sử dụng phương thức finish để kết thúc activity đang hoạt động
                finish();
            }
        });
    }
    // các phương thức sẵn có trong class ActivityC sẽ nằm ngay sau dòng ghi chú này
    ...
}
```

B8: Bổ sung ActivityC vào file AndroidManifest.xml file (thực hiện như các bài thực hành trước).

B9: Hiệu chỉnh mã lệnh của ActivityA

- Bổ sung lệnh sau trong phần import
`import android.app.Activity;`

- Bổ sung mã lệnh cho button “Mở Activity C”

```
Button getData = (Button)findViewById(R.id.button1);
getData.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        Intent intent = new Intent(ActivityA.this, ActivityC.class);
```

- Để xử lý dữ liệu gửi về từ ActivityC, bạn cần thực hiện phương thức `onActivityResult()` sẵn có của class Activity. Lưu ý nội dung của phương thức này phải đặt cùng cấp với phương thức `onCreate()` của class ActivityA
- ```

@Override
/* Tham số int requestCode: giá trị do bạn cung cấp khi mở Activity */
/* Tham số int resultCode: là tập kết quả trong ActivityC gồm
Activity.RESULT_OK hoặc Activity.RESULT_CANCELED, cho biết công việc đã hoàn tất
hay bị hủy bởi người dùng */
/* Tham số Intent data: Đối tượng Intent chứa dữ liệu được Activity được gọi sẽ
trả về */
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
 String enteredData="Bạn đã chọn button 'Bỏ qua' trong ActivityC";
 // nếu có dữ liệu trả về
 if (requestCode == 0 && resultCode == Activity.RESULT_OK){
 enteredData = data.getStringExtra("Data");
 }
 // đưa dữ liệu trả về vào textView
 t.setText(enteredData);
 super.onActivityResult(requestCode, resultCode, data);
}

```
- Do đối tượng TextView t được sử dụng trong phương thức nên bạn cần chuyển khai báo của TextView t thành toàn cục.

Sau khi hoàn tất, nội dung class ActivityA có dạng:

```

import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.app.Activity;
import android.content.Intent;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.view.View;
public class ActivityA extends ActionBarActivity {
 TextView t;
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_layout);
 t=(TextView)findViewById(R.id.textView1);
 Button getData = (Button)findViewById(R.id.button1);
 getData.setOnClickListener(new OnClickListener() {
 public void onClick(View v) {
 Intent intent = new Intent(ActivityA.this, ActivityC.class);
 // giá trị 0 được gởi chính là là requestCode
 startActivityForResult(intent, 0);
 }
 });
 }
 @Override
 /* Tham số requestCode: giá trị do bạn cung cấp khi gọi phương thức
startActivityForResult */
 /* Tham số resultCode: là tập kết quả trong ActivityC gồm Activity.RESULT_OK hoặc
Activity.RESULT_CANCELED*/
 // Tham số data: chứa dữ liệu được cập nhật
 protected void onActivityResult(int requestCode, int resultCode, Intent data) {
 String enteredData="Bạn đã chọn button 'Bỏ qua' trong ActivityC";
 // nếu có dữ liệu trả về
 }
}

```

```

if (requestCode == 0 && resultCode == Activity.RESULT_OK){
 enteredData = data.getStringExtra("Data");
}
//đưa dữ liệu trả về vào textView1
t.setText(enteredData);

super.onActivityResult(requestCode, resultCode, data);
}

// các phương thức sẵn có trong class ActivityA sẽ nằm ngay sau dòng ghi chú này
// . .
}

```

### 1.3.7. Intents

#### 1.3.7.1. Phương thức thường được sử dụng trong Intent:

- Phương thức `startActivity()` để mở một Activity
- Phương thức `broadcastIntent` để gửi nó đến một `BroadcastReceiver`
- Phương thức `startService(Intent)`, `bindService(Intent, ServiceConnection, int)` để giao tiếp với các Service chạy dưới nền.

Ngoài ra Intent còn cung cấp một chức năng cho phép kết nối hai chương trình khác nhau trong lúc thực thi (Cung cấp khả năng cho phép hai chương trình khác nhau giao tiếp với nhau).

#### 1.3.7.2. Các thành phần trong Intent

Intent về cơ bản là một cấu trúc dữ liệu, được mô tả trong class `android.content.Intent`.

##### 1.3.7.2.1. Thuộc tính

- Thuộc tính chính:
  - **action**: Tên của hành động cần thực hiện. Các hành động này có thể là hành động được Android định nghĩa (Mở Contact, gọi điện thoại, gửi email) sẵn hoặc hành động do người dùng định nghĩa.
  - **data**: Dữ liệu mà thành phần được gọi (Activity, Service, Broadcast receiver, Content provider). Dữ liệu được lưu trữ dưới định dạng Uri.
- Thuộc tính phụ:
  - **category**: Thông tin về nhóm của hành động cần thực thi.
  - **type**: Định dạng kiểu dữ liệu kèm kèm.
  - **component**: Thành phần (Activity, Service, Broadcast receiver, Content provider) nhận đối tượng Intent. Khi thuộc tính này được chỉ định thì mọi thuộc tính khác trở thành không bắt buộc.
  - **extras**: Chứa các cặp (key,value) được gắn vào Intent.

##### 1.3.7.2.2. Các action định nghĩa sẵn

- Là những hằng String đã được định nghĩa sẵn trong class Intent. Đi kèm với nó là các thành phần (Activity, Broadcast receiver, Service) hoặc ứng dụng được xây dựng sẵn sẽ được triệu hồi mỗi khi Intent tương ứng được gửi.
- Gồm 2 loại:
  - Built-in Standard Actions

|                    |                      |                   |
|--------------------|----------------------|-------------------|
| ACTION_ANSWER      | ACTION_FACTORY_TEST  | ACTION_SEARCH     |
| ACTION_ATTACH_DATA | ACTION_GET_CONTENT   | ACTION_SEND       |
| ACTION_CALL        | ACTION_INSERT        | ACTION_SENDTO     |
| ACTION_CHOOSER     | ACTION_MAIN          | ACTION_SYNC       |
| ACTION_DELETE      | ACTION_PICK          | ACTION_VIEW       |
| ACTION_DIAL        | ACTION_PICK_ACTIVITY | ACTION_WEB_SEARCH |
| ACTION_EDIT        | ACTION_RUN           |                   |

- Built-in Standard Broadcast Actions

|                             |                           |
|-----------------------------|---------------------------|
| ACTION_BATTERY_CHANGED      | ACTION_POWER_CONNECTED    |
| ACTION_BOOT_COMPLETED       | ACTION_POWER_DISCONNECTED |
| ACTION_PACKAGE_ADDED        | ACTION_SHUTDOWN           |
| ACTION_PACKAGE_CHANGED      | ACTION_TIME_CHANGED       |
| ACTION_PACKAGE_DATA_CLEARED | ACTION_TIME_TICK          |
| ACTION_PACKAGE_REMOVED      | ACTION_TIMEZONE_CHANGED   |
| ACTION_PACKAGE_RESTARTED    | ACTION_UID_REMOVED        |

- Một số action thường sử dụng trong Intent:

- ACTION\_ANSWER**: Mở Activity để xử lý cuộc gọi tới, thường là Phone Dialer của Android.
- ACTION\_CALL**: Mở một Phone Dialer (mặc định là PD của Android) và ngay lập tức thực hiện cuộc gọi dựa vào thông tin trong data URI.
- ACTION\_DELETE**: Mở Activity cho phép xóa dữ liệu mà địa chỉ của nó chứa trong data URI
- ACTION\_DIAL**: Mở Phone Dialer (mặc định là Phone Dialer của Android) và điền thông tin lấy từ địa chỉ chứa trong data URI.
- ACTION\_EDIT**: Mở một Activity cho phép chỉnh sửa dữ liệu mà địa chỉ lấy từ data URI.
- ACTION\_SEND**: Mở một Activity cho phép gửi dữ liệu lấy từ data URI, kiểu của dữ liệu xác định trong thuộc tính type.
- ACTION\_SENDTO**: Mở một Activity cho phép gửi thông điệp tới địa chỉ lấy từ data URI.
- ACTION\_VIEW**: Đây là action thông dụng nhất, khởi chạy activity thích hợp để hiển thị dữ liệu trong data URI.
- ACTION\_MAIN**: Sử dụng để khởi chạy một Activity.

- Ví dụ:

- Quay số điện thoại:

```
Intent dialIntent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:123456"));
startActivity(dialIntent);
```

- Mở danh sách contact:

```
Intent listContacts = new
Intent(Intent.ACTION_VIEW, Uri.parse("content://contacts/people/"));
startActivity(listContacts);
```

- Chuỗi data trong hàm Uri.parse(data): là định dạng dữ liệu tương ứng với mỗi action (chuẩn RFC 3986). Một khi sử dụng hành động đã được định sẵn, bạn phải cung cấp data cho nó theo định dạng này. Bảng dưới đây liệt kê một số định dạng và action tương ứng đã được định nghĩa sẵn:

| Định dạng                     | Action            | Mô tả                                            |
|-------------------------------|-------------------|--------------------------------------------------|
| tel:phone_number              | ACTION_VIEW       | Mở Dial form (chưa gọi)                          |
| tel:phone_number              | ACTION_CALL       | Thực hiện gọi tới số phone                       |
| http://web_address            | ACTION_VIEW       | Mở trình duyệt web với địa chỉ được cấp          |
| https://web_address           |                   |                                                  |
| "some_words" (string)         | ACTION_WEB_SEARCH | Thực hiện search                                 |
| http://web_address            |                   |                                                  |
| https://web_address           |                   |                                                  |
| sms://                        | ACTION_SENDTO     | Gửi tin nhắn                                     |
| geo:latitude,longitude        | ACTION_VIEW       | Mở ứng dụng Maps và chỉ tới vị trí được xác định |
| geo:latitude,longitude?z=zoom |                   |                                                  |
| geo:0,0?q=my+street+address   |                   |                                                  |
| geo:0,0?q=business+near+city  |                   |                                                  |

### 1.3.7.3. Phân loại Intent

#### 1.3.7.3.1. Intent tường minh - Explicit Intent:

Xác định rõ một thành phần (Activity, Broadcast Receiver, Service) thông qua phương thức `setComponent(ComponentName)` hoặc `setClass(Context, Class)` sẽ thực thi các hành động được đặc tả trong Intent. Thông thường thì những Intent này không chứa bất kỳ thông tin nào khác (như category, type) mà đơn giản chỉ là cách để ứng dụng mở các Activity khác bên trong một Activity.

#### 1.3.7.3.2. Intent không tường minh - Implicit Intent:

Không chỉ định một thành phần nào cả, thay vào đó, chúng sẽ chứa đủ thông tin để hệ thống có thể xác định component có sẵn nào là tốt nhất để thực thi hiệu quả cho Intent đó.

Trong Android có “*Tiến trình phân giải Intent*”. Khi sử dụng *Implicit intent*, do tính chất chuyên quyền của loại Intent này, “*Tiến trình phân giải Intent*” sẽ chỉ định Intent đến một Activity, *BroadcastReceiver*, hoặc *Service* (hoặc thỉnh thoảng có thể là 2 hay nhiều hơn một activity/receiver) để có thể xử lý các hành động được đặc tả trong Intent.

### 1.3.7.4. Intent Filter

- Bất cứ thành phần nào (Activity, Broadcast Receiver, Service) khi muốn sử dụng trong ứng dụng đều phải được đăng ký trong file `AndroidManifest.xml`. Trong đó cần định nghĩa một thẻ `<intent-filter>` cung cấp các thông tin để hệ thống có thể xác định được cái mà các thành phần này có thể xử lý (action) được.
- Một thành phần (Activity, Service, Broadcast receiver) cung cấp thông tin cho hệ điều hành biết loại đối tượng Intent mà nó có thể “xử lý” bằng cách khai báo một hoặc nhiều *Intent Filter*. Mỗi Intent Filter cung cấp các thông tin về “cái” mà thành phần này có thể xử lý (hiển thị một contact, gọi điện thoại...). Các thành phần khai báo Intent Filter bằng cách sử dụng thẻ `<intent-filter>` đặt trong thẻ khai báo thành phần (`<activity>`, `<service>`, `<receiver>`) như ví dụ sau:
- Ví dụ: khai báo một Activity với thẻ `<intent-filter>` chứa các thông tin dùng để chọn lựa Intent.

```
<activity android:name=".main"
 android:label="@string/app_name">
 <intent-filter>
 <action android:name="android.intent.action.MAIN" />
 <category android:name="android.intent.category.LAUNCHER" />
 </intent-filter>
</activity>
```

- Với Intent tường minh thì thành phần nhận Intent đó đã được xác định cụ thể, tuy nhiên với các Intent không tường minh thì hệ thống sẽ tiến hành so sánh các thông tin phụ được khai báo trong Intent với các thành phần được khai báo trong file `AndroidManifest.xml` của tất cả các ứng dụng cài đặt trên thiết bị để tìm ra thành phần phù hợp.
- Một Intent Filter có các thành phần chính sau:
  - **action** : Tên hành động mà thành phần có thể thực thi.
  - **type** : Kiểu dữ liệu thành phần có thể thực thi.
  - **category** : Phân nhóm các thành phần.

### 1.3.7.5. Sử dụng Intent

Các phương thức để khởi động các thành phần bằng cách sử dụng Intent:

| Tên hàm                        |                          | Mô tả                                                                  |
|--------------------------------|--------------------------|------------------------------------------------------------------------|
| startActivity(intent)          | android.app.<br>Activity | Thực thi activity như mô tả trong intent<br>(không lấy kết quả trả về) |
| startActivityForResult(intent) |                          | Thực thi activity như mô tả trong intent<br>(có lấy kết quả trả về)    |

|                                             |                                |                                                             |
|---------------------------------------------|--------------------------------|-------------------------------------------------------------|
| sendBroadcast(intent)                       | android.content.ContextWrapper | Phát tán intent tới bất kỳ thành phần BroadcastReceiver nào |
| startService(intent)                        |                                | Chạy 1 service                                              |
| bindService(intent, ServiceConnection, int) |                                | Bind 1 service                                              |

### 1.3.7.5.1. Intent tương minh thực thi Activity

Như đã trình bày ở phần trên, intent có thể dùng thuộc tính phụ để chỉ định đích danh tên Activity sẽ được mở. Để thực hiện điều này, class Intent cung cấp các phương thức setComponent(ComponentName) và setClass(Context, Class) và setClassName(Context, String) setClassName(String, String). Cách gọi này chỉ có thể được dùng để gọi các Activities trong cùng một ứng dụng.

Ví dụ:

```
Intent intent = new Intent();
intent.setClassName("ten_package", "ten_lop_activity_ben_trong_package");
startActivity(intent);
```

### 1.3.7.5.2. Intent không tương minh thực thi Activity

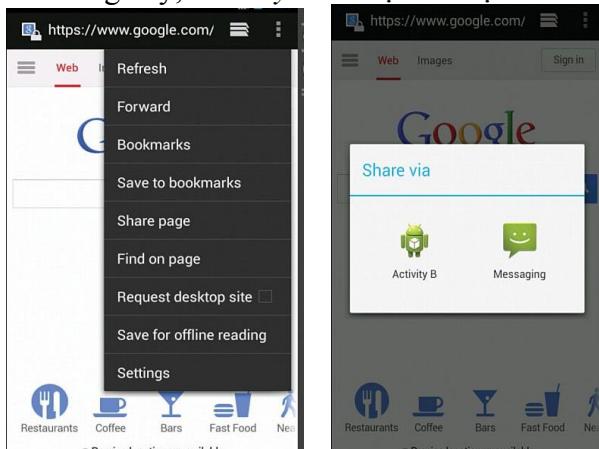
Trong trường hợp này intent không chỉ định một class cụ thể mà thay vào đó hệ thống dùng các dữ liệu khác (action, data, type, etc.) và quyết định xem thành phần nào (Activity, Service, Broadcast receiver) của ứng dụng nào sẽ thích hợp để đáp ứng intent đó.

Như đã nhắc đến ở phần trên, mọi thành phần của một ứng dụng Android phải được khai báo trong file AndroidManifest.xml. Trong đó thông tin action và category của bất kỳ thành phần nào(Activity, Service, Broadcast receiver, Content Provider) được khai báo trong thẻ intent-filter (Tất nhiên nếu chúng ta muốn gọi một hành động được thực thi bởi một thành phần định sẵn thì ta không cần quan tâm đến việc khai báo này).

Ví dụ: khai báo một Activity tên là ActivityExample. Intent-Filter của Activity này có các thuộc tính: <data android:mimeType="video/\*" android:scheme="http"/> xác định ActivityExample có thể xử lý được một đối tượng Intent có đính kèm dữ liệu là file video với đường dẫn kiểu đường dẫn http

```
<activity android:name=".ActivityExample">
 <intent-filter>
 <action android:name="action_name"/>
 <category android:name="category_name"/>
 <data android:mimeType="video/*"
 android:scheme="http"/>
 </intent-filter>
</activity>
```

Trong phần này, chúng ta sẽ cung cấp thêm cho ActivityB xử lý Intent.ACTION\_SEND đối với dữ liệu dạng plain text. Nghĩa là khi bất kỳ một activity nào có yêu cầu xử lý Intent.ACTION\_SEND, ActivityB sẽ được bổ sung vào các lựa chọn sẵn có đưa ra cho người dùng. Với việc thêm khả năng này, ActivityB sẽ nhận dữ liệu từ ActivityA thông qua intent.

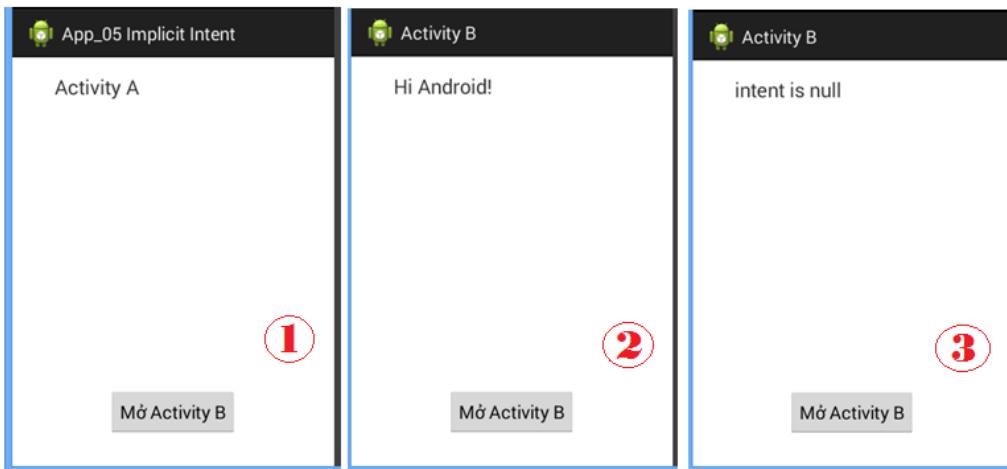


Hình 1-58 Khi người dùng chọn chức năng Share page sẽ xuất hiện thêm lựa chọn ActivityB

Ví dụ trong hình 1-58, trước khi ActivityB được cài đặt, chức năng Share Page chỉ có 1 lựa chọn là *Messaging*. Sau khi được cài đặt trước, ActivityB sẽ được bổ sung vào danh sách các lựa chọn. Nhờ vậy khi người dùng chọn chức năng Share page sẽ xuất hiện ra 2 lựa chọn.

## BÀI THỰC HÀNH App\_05

- ☞ **Yêu cầu:** Trong bài thực hành này, chúng ta chỉ xây dựng ứng dụng minh họa cho xử lý của Intent.ACTION\_SEND. Trong hình sau mô phỏng quá trình hoạt động của ứng dụng:



Hình 1-59 Tùy ActivityA có sử dụng Intent.ACTION\_SEND hay không mà ActivityB sẽ có kết quả khác nhau

- ① Ứng dụng khi khởi chạy.
- ② ActivityB khi ActivityA có sử dụng Intent.ACTION\_SEND.
- ③ ActivityB khi ActivityA không sử dụng Intent.ACTION\_SEND.

- ☞ **Thực hiện**

**B1:** Tạo App05 bằng cách copy và paste App\_04 vào cửa sổ Package Explorer, đổi tên project là App\_05. Tìm và mở file App\_05\res\values\string.xml, đổi tên ứng dụng từ App\_04 thành App\_05 như đã thực hiện trong các bài thực hành trước

**B2:** Chỉ định ActivityB sẽ xử lý intent. Mở file App\_05\AndroidManifest.xml. Chính sửa nội dung file này để ActivityB sẽ đóng vai trò 1 implicit intent (ActivityA vẫn là activity chính (nên không cần chỉnh sửa)

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.app_02" android:versionCode="1" android:versionName="1.0">
 <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="21" />
 <application
 android:allowBackup="true"
 android:icon="@drawable/ic_launcher"
 android:label="@string/app_name"
 android:theme="@style/AppTheme" >
 <activity
 android:name=".ActivityA"
 android:label="@string/app_name" >
 <intent-filter>
 <action android:name="android.intent.action.MAIN" />
 <category android:name="android.intent.category.LAUNCHER" />
 </intent-filter>
 </activity>
 <activity
 android:name=".ActivityB"
 android:label="Activity B">
 <intent-filter>
 <!-- chỉ định ActivityB sẽ xử lý hành động SEND, với SEND là 1
 trong những action được quy định sẵn trong Android-->
 <action android:name="android.intent.action.SEND" />
 <category android:name="android.intent.category.DEFAULT" />
 <!-- chỉ định ActivityB sẽ xử lý dữ liệu plain text -->
 </intent-filter>
 </activity>
 </application></pre>

```

```

 <data android:mimeType="text/plain" />
 </intent-filter>
</activity>
</application>
</manifest>
```

**B3:** Hiệu chỉnh lại nội dung của sự kiện onCreatetrong class ActivityA để có như minh họa sau:

```

public class ActivityA extends ActionBarActivity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_layout);
 Button getData = (Button)findViewById(R.id.button1);
 getData.setOnClickListener(new OnClickListener() {
 public void onClick(View v) {
 Intent intent = new Intent(ActivityA.this, ActivityB.class);
 /* Để tắt action SEND, bạn sử dụng 2 dấu // để che 3 dòng sau rồi cho
 chạy lại ứng dụng */
 intent.setType("text/plain");
 intent.putExtra(Intent.EXTRA_TEXT, "Hi Android!");
 intent.setAction("android.intent.action.SEND");
 startActivity(intent);
 }
 });
 }
 // các phương thức sẵn có trong class ActivityA sẽ nằm ngay sau dòng ghi chú này
 // . .
}
```

**B4:** Hiệu chỉnh lại nội dung của sự kiện onCreatetrong class ActivityB để có như minh họa sau:

```

public class ActivityB extends ActionBarActivity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_layout);
 TextView t = (TextView)findViewById(R.id.textView1);
 Intent intent = getIntent();
 if (intent!=null)
 {
 String action = intent.getAction();
 String type = intent.getType();
 if ((Intent.ACTION_SEND.equals(action)) && ("text/plain".equals(type)))
 {
 t.setText(intent.getStringExtra(Intent.EXTRA_TEXT));
 }
 else
 {
 t.setText("intent is null");
 }
 }
 }
 // các phương thức sẵn có trong class ActivityA sẽ nằm ngay sau dòng ghi chú này
 // . .
}
```

Cho chạy ứng dụng để xem kết quả khác nhau khi thực hiện che 3 dòng lệnh trong ActivityA.

Mỗi activity có thể xử lý intent ACTION\_SEND như trên. Khi phương thức `startActivity()` được gọi với tham số intent, hệ thống Android sẽ tìm intent action với activity phù hợp trong hệ thống của Android và khởi chạy activity tìm thấy.

### 1.3.7.6. Sử dụng Intent trong việc chuyển thông tin

Các intent có thể được dùng để chuyển dữ liệu giữa các activity. Khi sử dụng intent trong trường hợp cần gắn kèm thêm dữ liệu - gọi là extra intent – ta sử dụng phương thức `putExtra()`, kèm theo theo đó sẽ chỉ ra kiểu của dữ liệu được chuyển thông qua phương thức `setType()`.

Khi sử dụng phương thức `startActivityForResult()`, kết quả được trả về dưới dạng tham số là 1 intent cho phương thức `onActivityResult()`. Dữ liệu trong intent này có thể được xử lý bởi activity nơi gọi (activity cha).

### 1.3.7.7. Sử dụng Intents để thực thi các loại ứng dụng khác

- Với việc sử dụng các action của intent, ta có thể thực thi một số ứng dụng như:
  - Thực thi các trình duyệt web và cung cấp địa chỉ URL
  - Thực thi các trình duyệt web và cung cấp chuỗi cần tìm kiếm
  - Thực thi ứng dụng gọi điện thoại và cung cấp cho ứng dụng số điện thoại cần gọi
  - Thực thi ứng dụng bản đồ và cung cấp vị trí cần tìm kiếm
  - Thực thi ứng dụng Google Street View và cung cấp vị trí cần tìm kiếm
  - Thực thi ứng dụng xem hoặc quay video
  - Thực thi ứng dụng phát nhạc
  - ...
- Ví dụ: với việc bổ sung đoạn mã sau khi tạo 1 intent và cung cấp loại action (`ACTION_VIEW`) để thực thi trình duyệt với địa chỉ URL được cung cấp kèm theo
 

```
Uri address = Uri.parse("http://developer.android.com/");
Intent androidDocs = new Intent(Intent.ACTION_VIEW, address);
startActivity(androidDocs);
```

Trong ví dụ này, dữ liệu được gởi kèm theo intent là 1 URI (*Uniform Resource Identifier*), cho biết địa chỉ của tài nguyên cần truy cập để xem. Khi đó, nội dung trang được gọi sẽ được chạy foreground, điều đó dẫn đến trang nguyên thủy (default) của địa chỉ này sẽ bị tạm dừng ở chế độ background. Vì vậy, khi người dùng kết thúc trang cần xem bằng cách click button Back, trang nguyên thủy đó sẽ được chuyển sang chế độ foreground.

Các ứng dụng cũng có thể tạo riêng cho mình các kiểu intent và cho phép các ứng dụng khác gọi chúng thực hiện.

### 1.3.8. Vận dụng kiểu mở Activity mới khi lập trình

- Có 2 kiểu mở Activity mới:
  - Mở Activity mới lên làm che khuất toàn bộ Activity cũ (không nhìn thấy Activity cũ): xảy ra sự kiện `onPause` rồi `onStop` đối với Activity cũ.
  - Mở Activity mới lên làm che khuất một phần Activity cũ (vẫn nhìn thấy Activity cũ): xảy ra sự kiện `onPause` với Activity cũ.
- Vận dụng:
  - Khi quay trở về Activity cũ thì sau khi thực hiện xong các hàm cần thiết, chắc chắn ứng dụng phải gọi hàm `onResume` để phục hồi lại trạng thái ứng dụng.
  - Như vậy ta thường lưu lại trạng thái của ứng dụng trong sự kiện `onPause` và đọc lại trạng thái ứng dụng trong sự kiện `onResume`.

### 1.3.9. Vòng đời của Activity

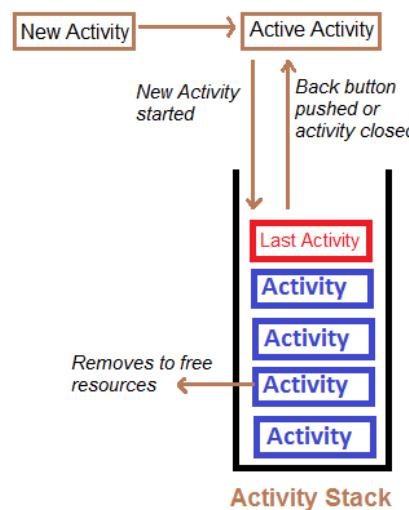
Hầu hết mỗi giao diện người dùng đều được điều khiển bằng 1 class Activity. Mỗi ứng dụng có thể gồm 1 hoặc nhiều Activity.

#### 1.3.9.1. Các trạng thái của activity

- Mỗi activity có một số trạng thái nội tại riêng gồm:
  - **Creating:** Khi ứng dụng được mở lên thì Activity chính sẽ được tạo ra và được đặt lên trên cùng của stack. Lúc này chỉ có duy nhất Activity trên cùng là hiển thị nội dung đến người dùng. Tất cả các Activity còn lại đều chuyển về trạng thái dừng hoạt động.

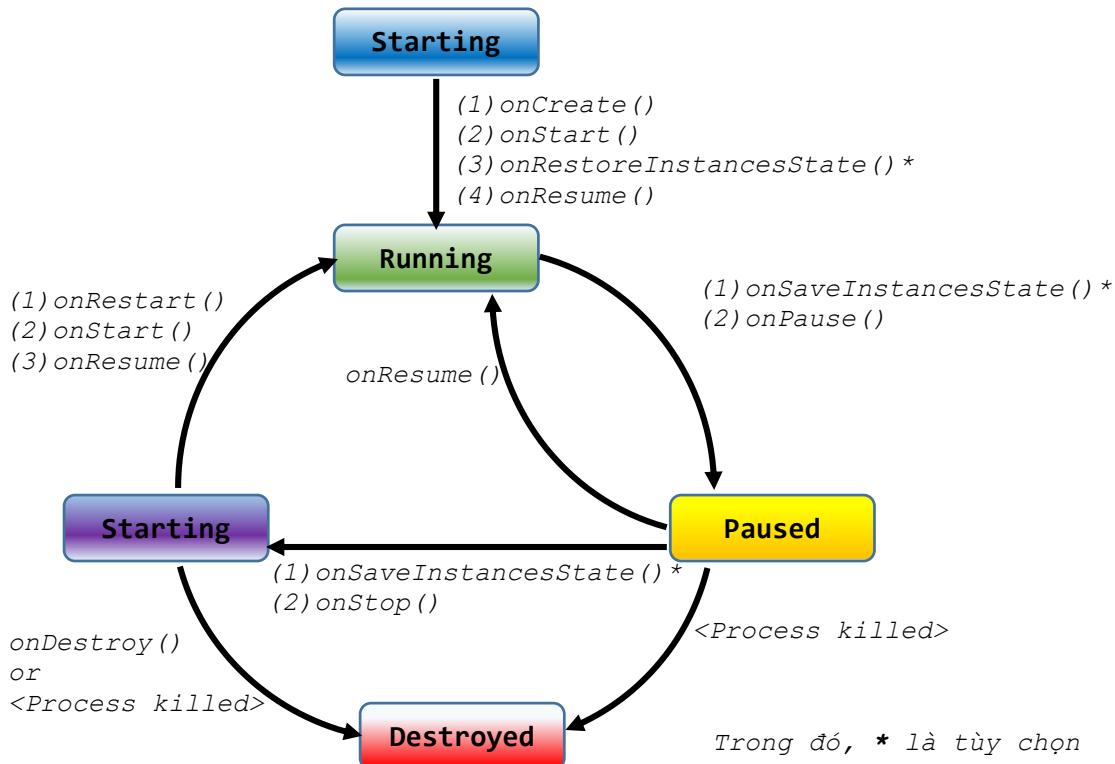
- Running** (đang kích hoạt): Khi màn hình là Foreground (Activity nằm trên cùng ứng dụng và cho phép người sử dụng tương tác)
- Paused** (tạm dừng): Activity bị mất focus nhưng vẫn nhìn thấy được như: Activity bị một activity trong suốt (transparent) hoặc một Activity không chiếm toàn bộ màn hình thiết bị (non-full-sized) đè lên.
- Stopped** (dừng): Khi một activity bị che khuất hoàn toàn bởi một activity khác.
- Destroyed** (chấm dứt)
  - Khi người dùng nhấn phím Back
  - Các “Paused activity” và “Stopped activity” sẽ bị hệ thống bắt chấm dứt (Destroyed) khi hệ thống thiếu bộ nhớ.

⇒ Khi nút “Home” được nhấn ⇒ không thể dùng nút “Back” để quay lại màn hình cũ được.



Hình 1-60 Gỡ bỏ activity khi thiếu tài nguyên

#### - Mô phỏng vòng đời của activity qua các trạng thái



Hình 1-61 Mô phỏng vòng đời của activity qua các trạng thái

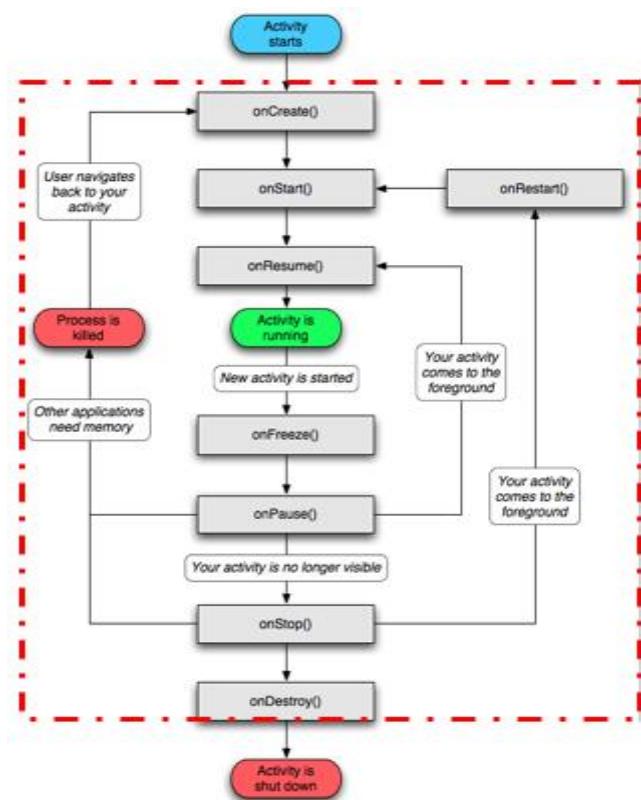
#### 1.3.9.2. Các sự kiện trong vòng đời của activity

- **onCreate**: là nơi dùng để
  - Khởi tạo activity.
  - Người lập trình gọi setContentView(int) để gán giao diện cho Activity.
  - Sử dụng phương thức findViewById(int) giúp gọi về các đối tượng đồ họa (Button, TextView...) đã được khai báo trong file xml để có thể sử dụng sau này.
- **onStart**
  - Được thực hiện khi sau activity được đưa vào background.

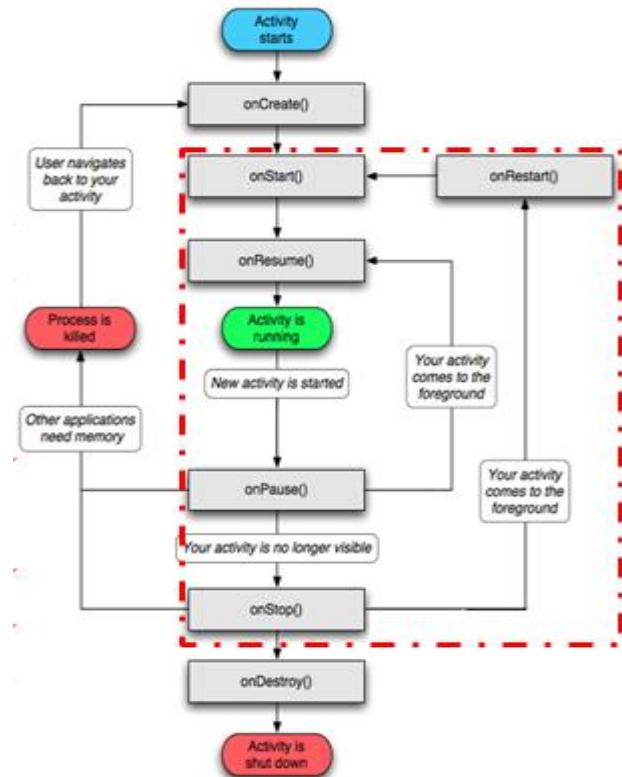
- Nơi sẽ giúp xác định tất cả tài nguyên hệ thống mà ứng dụng yêu cầu đã sẵn sàng. Ví dụ như khi có yêu cầu tài nguyên về GPS, phương thức này là nơi tốt nhất để xác định khả năng sẵn dùng của GPS.
- **onResume**: đưa activity trở lại để người dùng có thể tương tác.
- **onPause**
  - Nơi giải quyết sự kiện người dùng rời khỏi activity.
  - Là nơi tốt để thực hiện các công việc như dùng các hình ảnh động, lưu lại các dữ liệu cần thiết, phóng thích tài nguyên của hệ thống. Bất kỳ tài nguyên nào được phóng thích trong onPause() sẽ được thiết lập lại trong phương thức onResume().
  - Chấm dứt hoạt động của các tiêu trình (thread).
  - Một activity có thể liên tục thay đổi giữa hai trạng thái **Paused** và **Resumed** (VD: như khi thiết bị sleep).
- **onStop**:
  - Đảm bảo việc activity đã được đưa về background.
  - Phóng thích tài nguyên hệ thống không còn cần dùng.
  - Chấm dứt hoạt động của các tiêu trình (thread).
- **onRestart**: nếu phương thức này được gọi, it indicates your activity is being redisplayed to the user from a stopped state.
- **onDestroy**: kết thúc 1 activity, dọn dẹp tài nguyên đã sử dụng
- **onSaveInstanceState(Bundle)**: là phương thức tùy chọn của người lập trình. Android sẽ gọi phương thức này để cho phép activity lưu lại trạng thái của mình (ví dụ như lưu lại vị trí của con trỏ trong khung nhập liệu - EditText). Thông thường, người lập trình sẽ không cần override phương thức này vì theo mặc định việc lưu lại trạng thái cho tất cả các giao diện người dùng sẽ được điều khiển tự động.
- **onRestoreInstanceState(Bundle)**: cũng là phương thức tùy chọn của người lập trình. Phương thức này được gọi khi activity được yêu cầu khôi phục lại trạng thái trước đó đã được lưu bởi phương thức onSaveInstanceState().

#### Mô phỏng vòng đời của activity qua các sự kiện

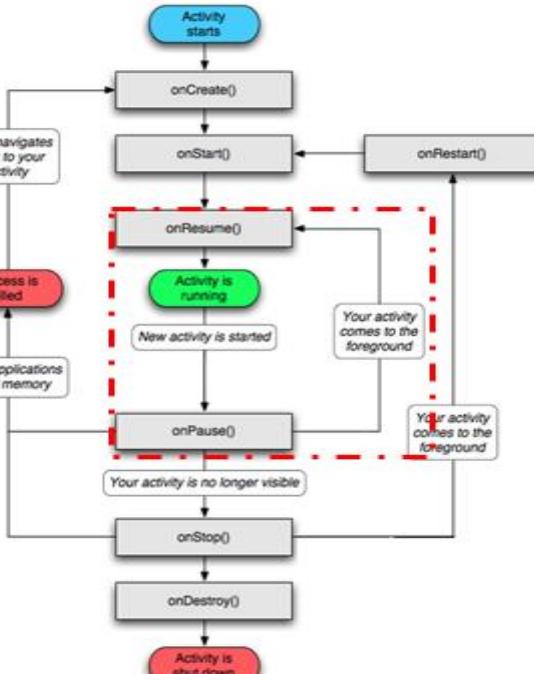
- **Vòng đời toàn diện (Entire Lifetime hay Full Lifetime)**: Diễn ra từ lần gọi `onCreate(Bundle)` đầu tiên và kéo dài tới lần gọi `onDestroy()` cuối cùng.



Hình 1-62 Entire Lifetime



Hình 1-63 Visible Lifetime



Hình 1-64 Foreground Lifetime

- **Vòng đời thấy được (Visible Lifetime):** Diễn ra từ khi gọi `onStart()` và kéo dài tới khi gọi `onStop()`.

Ở vòng đời này, activity được hiển thị trên màn hình mặc dù có thể nó không thể tương tác với người dùng (Activity có thể bị đè bởi một Activity trong suốt hoặc một Activity không chiếm toàn bộ màn hình thiết bị(non-full-sized)).

- **Vòng đời trên nền (Foreground Lifetime hay Active Lifetime):** Diễn ra từ khi gọi `onResume()` và kéo dài tới khi gọi `onPause()`.

Ở vòng đời này, activity nằm trên mọi activity khác và tương tác được với người dùng.

## 1.4. SỬ DỤNG TÀI NGUYÊN TRONG Android

### 1.4.1. Các file về tài nguyên trong Android Project

Hầu hết các tài nguyên trong ứng dụng Android được lưu trữ trong folder /res của project.

#### 1.4.1.1. File R.java

File R.java được lưu trữ trong foleder gen của ứng dụng. Không bao giờ được tự động hiệu chỉnh nội dung có trong file R.java vì nội dung này được tự động phát sinh dựa trên sự liên kết giữa các file tài nguyên và mã lệnh Java.

Khi sử dụng Eclipse để tạo ra project, 2 file về giao diện và mã lệnh được tạo lập. Trong phương thức `onCreate()` của activity, giao diện của ứng dụng được kết hợp với mã lệnh bằng phương thức `setContentView()`.

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

Phương thức `setContentView()` nhận định danh của tài nguyên làm tham số. Định danh này là 1 số nguyên. Khi mở file R.java, bạn có thể nhìn thấy sự kết hợp giữa tên của giao diện được định nghĩa trong xml file và định danh của tài nguyên.

```
public static final class Layout
{
 ...
 public static final int activity_layout=0x7f030019;
```

```
//...
}
```

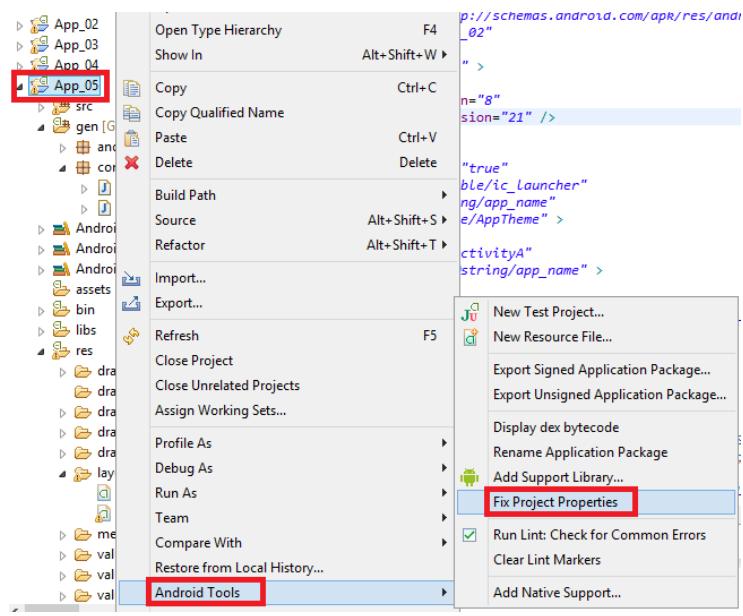
Khi 1 project được biên dịch (buid), file R.java sẽ được phát sinh ra các định danh tài nguyên cẩn cứ vào các tài nguyên có trong folder \res

### 1.4.1.2. Android Platform và Android Dependencies

Khi tạo ra 1 Android project, bạn phải sử dụng Android platform. Theo mặc định, ADT sẽ xác lập platform cho project là Android 4.2. Nội dung của folder Android 4.2 có trong Android.jar và đã được cài đặt với ADT Bundle.

Android dependencies chứa các file jar hỗ trợ Android. Khi Eclipse tạo project, project đã được gắn kèm thư viện Android tương thích (Android Compatibility Library).

Khi có sự cố xảy ra với Android dependencies, chúng ta có thể phát sinh lại bằng cách click phải vào tên project, chọn *Android Tools* và chọn *Fix Project Properties*.



Hình 1-65 Fix Project Properties

## 1.4.2. Các tài nguyên thường dùng

Một project thường dùng 1 số tài nguyên như file layout, hình ảnh, file chứa định nghĩa các chuỗi sử dụng và 1 số tài nguyên khác

### 1.4.2.1. Sử dụng tài nguyên

#### 1.4.2.1.1. Tham chiếu đến tài nguyên của ứng dụng

Tất cả các tài nguyên của ứng dụng được lưu trữ trong folder \res của ứng dụng và được biên dịch đưa vào ứng dụng ở thời điểm biên dịch. Các tài nguyên này có thể được dùng trong lập trình. Chúng cũng có thể tham chiếu đến tài nguyên có trong các ứng dụng khác.

Trong lập trình, để truy cập tài nguyên, các lập trình viên thường sử dụng phương thức *getResource()* kết hợp với các phương thức phù hợp với kiểu của tài nguyên cần dùng. Ví dụ để sử dụng chuỗi có tên là hello đã được định nghĩa trong file string.xml, người lập trình thường dùng như sau:

```
String greeting = getResources().getString(R.string.hello);
```

Để tham chiếu đến tài nguyên của ứng dụng từ tài nguyên khác ví dụ như của file layout, sử dụng cú pháp sau:

```
@[resource type]/[resource name]
```

Ví dụ, thay vì tham chiếu chuỗi có tên là hello theo cách vừa sử dụng ở trên, ta có thể dùng như sau:

```
@string/hello
```

#### 1.4.2.1.2. Sử dụng tài nguyên hệ thống

Các ứng dụng có thể truy cập tài nguyên của hệ thống Android để thêm vào tài nguyên của riêng mình. Tài nguyên hệ thống được lưu trữ trong gói *android.R*. Tồn tại các class cho mỗi kiểu tài nguyên chính, như class *android.R.string* chứa các tài nguyên về chuỗi đã được định nghĩa trước đó. Giả sử khi trong class Activity cần sử dụng tài nguyên chuỗi có tên là *ok* trong Resource, đầu tiên ta

cần gọi phương thức tĩnh của class Resource là `getSystem()` để nhận tài nguyên toàn cục của hệ thống, rồi sử dụng tiếp phương thức phù hợp với tài nguyên kiểu chuỗi là `getString()` như sau:

```
String confirm = Resources.getSystem().getString(android.R.string.ok);
```

Để tham chiếu một tài nguyên hệ thống từ tài nguyên đã được biên dịch khác (như file tài nguyên về layout), cần sử dụng theo định dạng sau:

```
@android:[resource type]/[resource name]
```

Ví dụ khi cần sử dụng tài nguyên kiểu chuỗi có tên là ok giống như trên, bạn dùng tên loại tài nguyên như sau:

```
@android:string/ok
```

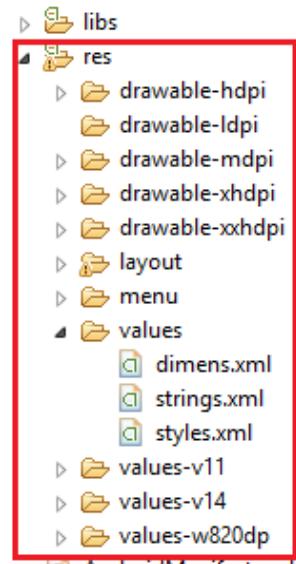
### 1.4.2.2. Sử dụng tài nguyên dạng đơn

Tài nguyên dạng đơn như chuỗi ký tự, màu sắc, giá trị kích thước đều phải được định nghĩa trước đó trong các file XML trong folder \res. Các file tài nguyên này sử dụng các tag của XML được trình bày dưới dạng các cặp *tên-giá trị*. Bạn có thể quản lý các loại tài nguyên này bằng cách hiệu chỉnh trực tiếp vào các file XML tương ứng.

#### 1.4.2.2.1. Sử dụng tài nguyên dạng chuỗi ký tự

- Có thể sử dụng các tài nguyên kiểu chuỗi ở bất kỳ đâu trong ứng dụng của mình khi nó đó cần hiển thị những đoạn văn bản.
- Định nghĩa 1 tài nguyên chuỗi, mở file res\values\string.xml, sử dụng cặp tag `<string>` giữa cặp thẻ này là đoạn văn bản cần dùng. Định danh của chuỗi được gán theo thuộc tính name trong tag mở của `<string>`

Hình 1-66 Folder res chứa tất cả tài nguyên sử dụng trong ứng dụng



- Nội dung file string.xml thường thấy trong các project ngay sau khi mới tạo lập:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
 <string name="app_name">Hour3App</string>
 <string name="hello_world">Hello world!</string>
 <string name="menu_settings">Settings</string>
</resources>
```

- Có thể sử dụng 1 số ký tự đặc biệt trong đoạn văn bản cần hiển thị như dấu nháy đơn (‘), dấu nháy đôi (“), ... bằng cách dùng liền sau ký tự \ và khoảng trắng.

Giá trị nằm giữa cặp thẻ <string>	Chuỗi sẽ hiển thị
Hello, World	Hello, World
“Hello, World”	Hello, World
Allen\'s car	Allen's car
He said, \"No.\\"	He said, "No."

- Truy cập tài nguyên chuỗi

- Sử dụng trong lập trình: sử dụng phương thức `getString()`. Ví dụ:  
`String greeting = getResources().getString(R.string.hello);`
- Sử dụng trong thiết kế layout: khi muốn gán chuỗi hiển thị lên các button, textview, ... sử dụng `@string`. Ví dụ:  
`android:text="@string/hello_world"`

#### 1.4.2.2.2. Sử dụng tài nguyên về màu sắc

Các tài nguyên về màu sắc dùng để áp dụng cho các điều khiển có trong layout.

- Định nghĩa tài nguyên về màu sắc:

- Mặc định, khi ứng dụng mới tạo lập chưa có file này, nên ta phải tự tạo mới file colors.xml bằng cách click phải vào folder res\values, chọn New→Others..., rồi chọn XML file.
- Sử dụng cặp tag `<color>` với thuộc tính name để xác định định danh cho màu.

- Minh họa nội dung file colors.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<resources>
 <color name="background_color">#006400</color>
 <color name="app_text_color">#FFE4C4</color>
</resources>
```

- Android theo định dạng RGB gồm 2 loại 12 và 24 bit. Cách sử dụng cho theo bảng sau:

Định dạng	Mô tả	Ví dụ
#RGB	12 bit color	#00F (blue)
#ARGB	12 bit color và alpha	#800F (blue, alpha 50%)
#RRGGBB	24 bit color	#FF00FF (magenta, alpha 80%)
#AARRGGBB	24 bit color và alpha	#80FF00FF (magenta, alpha 80%)

- Ví dụ: mã lệnh sau sẽ nhận được màu có định danh là *app\_text\_color* trong tài nguyên bằng phương thức *getColor()*:

```
int textColor = getResources().getColor(R.color.app_text_color);
```

- Để tiện sử dụng màu sắc, có thể tạo sẵn file colors.xml lưu các giá trị màu thường dùng:  
VD

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
 <color name="red">#FF0000</color>
 <color name="grey">#808080</color>
 <color name="grey2">#777777</color>
 <color name="grey3">#ACA899</color>
 <color name="white">#FFFFFF</color>
 <color name="black">#000000</color>
 <color name="yellow">#FFFF00</color>
 <color name="yellow1">#F9E60E</color>
 <color name="yellow2">#F9F89D</color>
 <color name="orange4">#F7BE45</color>
 <color name="orange5">#F7D896</color>
 <color name="blue">#0000FF</color>
 <color name="blue2">#19FCDA</color>
 <color name="blue25">#D9F7F2</color>
 <color name="white1">#FFFFFF</color>
 <color name="white2">#DDDDDD</color>
 <color name="green">#000044</color>
 <color name="lightgreen">#66FF33</color>
</resources>
```

- Màu nền (background): sử dụng giá trị alpha cho màu nền

00: tạo nền trong suốt. Ví dụ: #00777777

FF: tạo nền đục (che lấp đôi tượng nằm dưới). Ví dụ: #FF777777

#### 1.4.2.2.3. Sử dụng kích thước

Để chỉ ra kích thước của các điều khiển có trong giao diện như button, textview, ... bạn cần chỉ định sự khác nhau giữa các loại kích thước.

Tài nguyên về kích thước hỗ trợ áp dụng cho cỡ chữ, kích thước hình ảnh, và các vật chất khác hoặc các phép đo lường liên quan đến pixel.

- Định nghĩa tài nguyên:

- Tương tự như màu sắc, trong một số phiên bản cũ của Eclipse, khi ứng dụng mới tạo lập sẽ mặc định chưa có file này, nên ta phải tự tạo mới file *res/values/dimens.xml*.
- Sử dụng cặp tag *<dimen>* với thuộc tính name để xác định định danh cho kích thước.

- Minh họa nội dung file *res/values/dimens.xml*

```
<resources>
 <!-- Default screen margins, per the Android Design guidelines. -->
 <dimen name="activity_horizontal_margin">16dp</dimen>
 <dimen name="activity_vertical_margin">16dp</dimen>
 <dimen name="thumbDim">100px</dimen>
```

&lt;/resources&gt;

- Mỗi giá trị kích thước đều phải có đơn vị đo lường đi liền sau. Bảng sau mô tả các đơn vị đo lường được hỗ trợ trong Android

Đơn vị	Ký hiệu sử dụng	Sử dụng
Pixels	px	Màn hình
Inches	in	Đo lường về vật chất
Mulimeters	mm	Đo lường về vật chất
Points	pt	Font chữ
Density-independent pixels	dp	Độ phân giải (màn hình)
Scale-independent pixels	sp	Font chữ có khả năng thay đổi

- Ví dụ: mã lệnh sau sẽ nhận được màu có định danh là `thumbDim` trong tài nguyên bằng phương thức `getDimension()`:

```
float thumbnailDim = getResources().getDimension(R.dimen.thumbDim);
```

- Trong Android, do kích thước màn hình không cố định, vì vậy việc sử dụng các giá trị tự định nghĩa dựa trên pixel sẽ giúp cho giao diện của bạn hiển thị tốt hơn các thiết bị khác.

### 1.4.2.3. Sử dụng Drawable

- Các tài nguyên Drawable (như các file hình ảnh) đều phải được lưu trong folder `res/drawable` của project. Các thiết bị sử dụng Android có sự khác nhau về kích thước và độ phân giải màn hình.
- Lưu trữ hình ảnh theo độ phân giải:
  - Trong foleder `res`, drawable sẽ được chia nhiều folder con tùy thuộc độ phân giải của hình ảnh như: `drawable-ldpi` (low density), `drawable-mdpi` (medium density), `drawable-hdpi` (high density), ...
  - Khi ứng dụng cung cấp được nhiều phiên bản cho cùng 1 hình ảnh (hình ảnh phải được lưu trùng tên nhau, chỉ khác nhau về folder chứa), tùy thuộc vào độ phân giải màn hình, Android sẽ tự tìm hình ảnh có độ phân giải phù hợp nhất để hiển thị trên màn hình sao cho hình ảnh luôn đẹp nhất (hình ảnh có độ phân giải cao sẽ được sử dụng cho màn hình có độ phân giải cao).
- Bạn có thể kéo thả các file hình ảnh vào folder `res/drawable` trong của sổ Package Explorer hoặc sử dụng cơ chế import để chọn 1 (hoặc nhiều) file.

#### 1.4.2.3.1. Working with Images

- Hầu hết tài nguyên hình ảnh được dùng trong các ứng dụng có kiểu là bitmap như PNG, JPG. Các hình ảnh này thường được dùng cho các icon, button và một số thành phần khác trong giao diện của người dùng.
- Các định dạng thông dụng được Android hỗ trợ:

Định dạng hình ảnh được hỗ trợ	Mô tả	Phân mỏ rộng của file
Portable Networks Graphics	Preferred format (lossless)	.png (PNG)
Nine-Patch Stretchable	Preferred format (lossless)	.9.png (PNG) images
Joint Photographic Experts	Acceptable format (lossy)	.jpg (JPEG/JPG) Group
Graphics Interchange	Discouraged but supported	.gif (GIF) Format (lossless)

#### 1.4.2.3.2. NinePatch?

- Android có thể kèm cả các drawable với kiểu là ninepatch. Class của ninepatch là `android.graphics.drawable.NinePatchDrawable`. Một file ninepatch được ấn định có phần mở rộng là `.9.png`. Android cung cấp một công cụ được gọi là Draw 9-patch phục vụ cho việc tạo lập file hình ảnh dạng này.
- Một file ninepatch là 1 file ảnh bitmap có thể co giãn được với 1 số phần tử cố định. Các phần của ảnh dạng ninepatch có thể co giãn được như phóng to hình ảnh, khi đó 1 vài phần trong ảnh là không thay đổi.
- Các button chuẩn trong Android sử dụng ảnh ninepatch làm nền còn viền bao quanh button lại không thể sử dụng được.

#### 1.4.2.3.3. Sử dụng tài nguyên hình ảnh trong lập trình

Giả sử ta có 1 file ảnh tên là *logo.png* đã được định danh trong tài nguyên với tên gọi là *LogoImage*. Để nạp ảnh này trong mã lệnh của class Activity của ứng dụng, ta dùng như sau:

- Truy cập tài nguyên bằng mã lệnh: do tài nguyên hình ảnh được đóng gói trong class *BitmapDrawable*, nên cần sử dụng phương thức *getDrawable()* để truy cập tài nguyên:

```
BitmapDrawable logoBitmap = (BitmapDrawable) getResources().getDrawable(R.drawable.Logo);
```

- Nạp ảnh cho các widget bằng mã lệnh:

- Đổi với *ImageView*:

```
final ImageView logoView = (ImageView) findViewById(R.id.imageView1);
logoView.setImageResource(R.drawable.LogoImage);
```

- Đổi với *ImageButton*:

```
final ImageButton ib=(ImageButton)findViewById(R.id.imageButton1);
//đổi với ảnh nền (Background)
ib.setBackgroundResource(R.drawable.LogoImage);
//Đổi với ảnh chính (foreground)
ib.setImageResource(R.drawable.bground300);
```

## BÀI THỰC HÀNH App\_06

 **Yêu cầu:** Tạo 1 project chỉ gồm 1 *ImageButton*. Khi ứng dụng khởi chạy, button có hình chính (foreground) là *ic\_launcher*, khi người dùng click chọn sẽ đổi hình chính khác. Cho phép thay đổi xoay vòng 3 hình nền. Trong 3 hình nền có 2 hình có kích thước 48X48 và 1 hình có kích thước 300X300.

 **Thực hiện:**

**B1:** Tạo project mới với tên *App\_06*

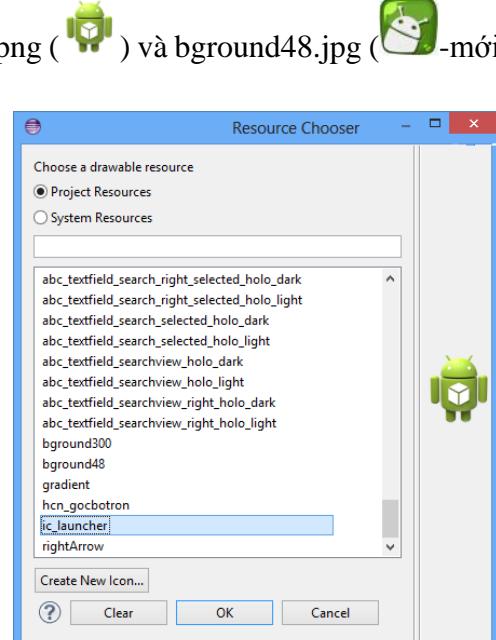
**B2:** Tìm và copy thêm 2 hình ảnh lưu vào folder *res\drawable-mdpi*. Như vậy sau khi copy hoàn tất, trong folder này sẽ có 4 hình ảnh (hình tên *ic\_launcher.png* là hình ảnh đã có sẵn khi project được tạo lập- ):

- Kích thước 48X48 có 2 hình với tên *ic\_launcher.png* ( và *bground48.jpg* ( -mới copy vào).
- Kích thước 300X300 với tên *bground300.jpg* ( mới copy vào).

**B3:** Mở file *activity\_main.xml*

- Kéo và thả 1 *ImageButton* (trong nhóm *Image & Media*) vào canvas. Xuất hiện hộp thoại *Resource Chooser*, chọn tên file ảnh cần dùng cho button(*ic\_launcher*)
- Lúc này nội dung file *activity\_main.xml* có dạng

```
<RelativeLayout
 xmlns:android=
 "http://schemas.android.com/apk/res/android"
 . . . >
<ImageButton
 android:id="@+id/imageButton1"
 android:src="@drawable/ic_launcher" />
```



Hình 1-67 Chọn resource cho *ImageView*

```
 android:layout_width= "wrap_content"
 android:layout_height= "wrap_content"
 android:src="@drawable/ic_launcher" />
</RelativeLayout>
```

**B4:** Mở file *MainActivity.xml*

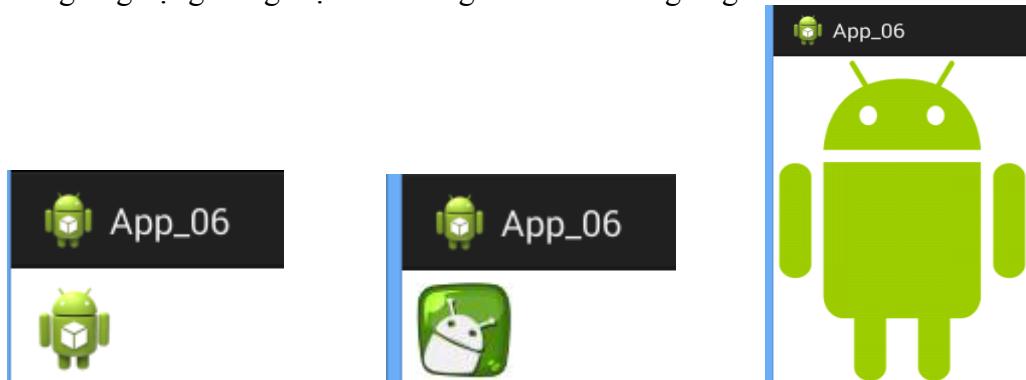
- Bổ sung mã lệnh để có nội dung có dạng như sau:

```

int dem=0;
@Override
protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 final ImageButton ib=(ImageButton)findViewById(R.id.imageButton1);
 ib.setOnClickListener(new OnClickListener() {
 public void onClick(View v) {
 dem=(dem+1)%3;
 switch (dem)
 {
 case 0:
 ib.setImageResource(R.drawable.ic_launcher);
 break;
 case 1:
 ib.setImageResource(R.drawable.bground48);
 break;
 case 2:
 ib.setImageResource(R.drawable.bground300);
 break;
 }
 }
 });
}

```

- B5:** Cho chạy chương trình. Khi click trên *imagebutton*, ngoài việc hình ảnh thay đổi, kích *imagebutton* cũng thay đổi theo kích thước của hình ảnh. Qua đó, các hình ảnh sử dụng trong ứng dụng trong thực tế thường có kích thước giống nhau.



Hình 1-68 Hình ảnh kế tiếp sẽ được nạp khi click vào hình ảnh đang xuất hiện

#### 1.4.2.3.4. Sử dụng các tài nguyên không phải kiểu hình ảnh trong Drawables

Ngoài các hình ảnh như đã nêu, bạn có thể quy định định dạng của file XML để mô tả các subclass của *drawable* như *ShapeDrawable*. Bạn cũng có thể sử dụng class *ShapeDrawable* để định nghĩa ra các hình ảnh theo khuôn dạng khác như hình chữ nhật, hình oval, ...

Cách chung để thực hiện là:

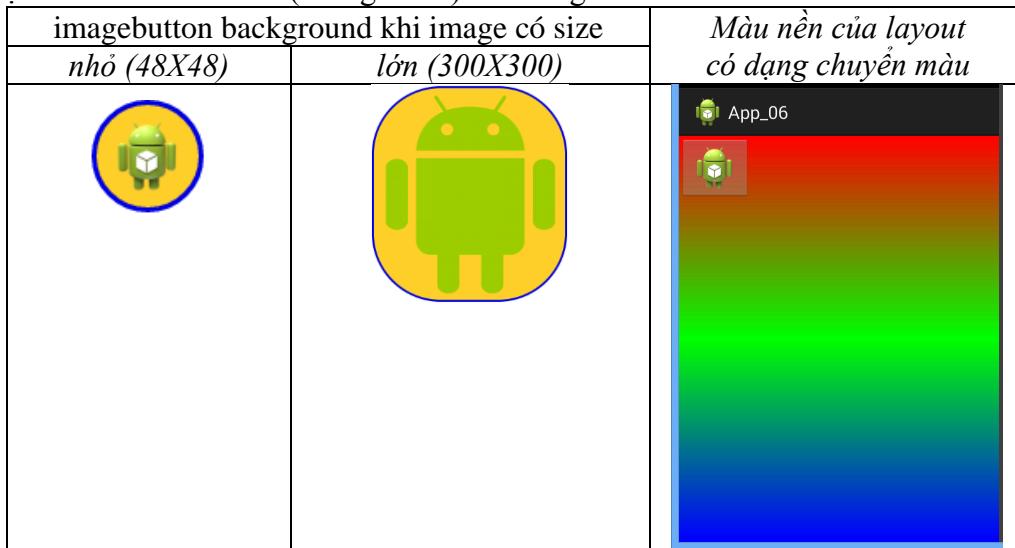
- Tạo file .xml lưu trong folder res\drawable để định dạng shape cần dùng.
- Trong file layout () của project, gán thuộc tính của đối tượng cần dùng shape vào file .xml vừa tạo ở trên, ví dụ:

*android:background="@drawable/Tên file XML vừa tạo"*

## BÀI THỰC HÀNH App\_06 (tiếp theo)

### ☞ Yêu cầu:

- ① Tạo Shape dùng làm khung viền cho button là hình chữ nhật với 4 góc được bo tròn. Minh họa button khi ảnh nền (background) của imagebutton có size



Hình 1-69 Minh họa kết quả của ứng dụng

- ② Tạo nền cho project App\_06 sao cho nền có dạng chuyển màu (gradient) từ xanh dương sang màu đỏ.

### ☞ Thực hiện yêu cầu ①

- B1:** Tạo shape dùng làm khung viền cho button: tạo mới file `res\drawable-mdpi\hcn_gocbotron.xml` với nội dung:

```
<?xml version="1.0" encoding="utf-8"?>
<shape
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:shape="rectangle">
 <!-- Set thuộc tính màu nền-->
 <solid
 android:color="#FACC2E" >
 </solid>
 <!-- set thuộc tính màu và độ rộng của đường viền -->
 <stroke
 android:width="3dp"
 android:color="#0101DF" >
 </stroke>
 <!-- các thuộc tính căn chỉnh-->
 <padding
 android:left="10dp"
 android:top="10dp"
 android:right="10dp"
 android:bottom="10dp" >
 </padding>
 <!-- bán kính đường tròn ở 4 góc -->
 <corners
 android:radius="100dp" >
 </corners>
</shape>
```

- B2:** Trong file `activity_main.xml`, bổ sung thuộc tính nền cho button:

```
 android:background="@drawable/hcn_gocbotron"
```

Sau khi bổ sung xong, nội dung file `activity_main.xml` có dạng:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 . . .
```

```

 tools:context="com.example.app_06.MainActivity" >
<ImageButton
 android:id="@+id/imageButton1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:background="@drawable/hcn_gocbotron"
 android:src="@drawable/ic_launcher" />
</RelativeLayout>

```

**Thực hiện yêu cầu ②**

- B3:** Tạo shape dùng làm nền cho layout: tạo mới file *res\drawable-mdpi\gradient.xml* với nội dung:

```

<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
 android:orientation="vertical"
 android:shape="rectangle">
<gradient android:startColor="#FF0000"
 android:centerColor="#00FF00"
 android:endColor="#0000FF"
 android:angle="90" />
</shape>

```

- B4:** Trong file *activity\_main.xml*, bổ sung thuộc tính nền cho layout. Sau khi bổ sung xong, nội dung file *activity\_main.xml* có dạng:

```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 ...
 android:background="@drawable/gradient"
 tools:context="com.example.app_06.MainActivity" >
<ImageButton
 android:id="@+id/imageButton1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:background="@drawable/hcn_gocbotron"
 android:src="@drawable/ic_launcher" />
</RelativeLayout>

```

- B5:** Cho chạy ứng dụng để thấy các yêu cầu đã được thực hiện.

Nhân xét:

Thuộc tính angle trong file *gradient.xml* cho biết hướng của việc chuyển màu, với quy ước:

- *Angle = "0"* : hướng chuyển màu từ trái sang phải.
- *Angle = "90"* : hướng chuyển màu từ dưới lên trên.
- *Angle = "180"* : hướng chuyển màu từ phải sang trái.
- *Angle = "270"* : hướng chuyển màu từ trên xuống dưới.



Hình 1-70 Hướng chuyển màu từ trên xuống dưới hay từ trái sang phải hoặc ngược lại của nền là do thuộc tính *angle* quyết định

#### 1.4.2.4. Adding Animations

Android cung cấp 3 hệ thống để tạo animation: property animation (từ Android 3.0), view animation và Drawable animation. Trong đó, thuộc tính animation được đánh giá linh hoạt và có nhiều tính năng hơn.

Do các đối tượng animation đều phải được lưu trong folder res\anim, vì vậy trước khi sử dụng animation cần tạo folder này trước.

### BÀI THỰC HÀNH App\_06 (tiếp theo)

- ☛ **Yêu cầu:** tạo hiệu ứng cho imagebutton lặp đi lặp lại việc chuyển từ mờ sang rõ trong 2 giây. Hình sau minh họa 3 trạng thái của imagebutton theo từng thời điểm.



Hình 1-71 Minh họa kết quả thực hiện của App\_06

#### ☛ Thực hiện

- B1:** Tạo mới file res\anim\fadein.xml. Nội dung file này quy định giá trị alpha sẽ chuyển từ 0.0 đến 1.0 (chuyển từ vô hình sang có thể nhìn thấy) được trong khoảng thời gian 2 giây (2,000 milliseconds)

```
<?xml version="1.0" encoding="UTF-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
 <alpha android:fromAlpha="0.0"
 android:toAlpha="1.0"
 android:interpolator="@android:anim/accelerate_interpolator"
 android:duration="2000"
 android:repeatCount="infinite"/>
</set>
```

- B2:** Mở file MainActivity.java để bổ sung 2 lệnh sau lệnh findViewById của ImageButton trong hàm onCreate()

```
Animation myFadeInAnimation = AnimationUtils.LoadAnimation(this, R.anim.fadein);
ib.startAnimation(myFadeInAnimation); //Set animation to ImageButton
```

Sau khi bổ sung hoàn tất class Activity có dạng:

```
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageButton;
import android.view.View.OnClickListener;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
public class MainActivity extends ActionBarActivity {
 int dem=0;
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 final ImageButton ib=(ImageButton)findViewById(R.id.imageButton1);
 Animation myFadeInAnimation = AnimationUtils.LoadAnimation(this, R.anim.fadein);
 ib.startAnimation(myFadeInAnimation); //Set animation to ImageButton
 ib.setOnClickListener(new OnClickListener() {
 public void onClick(View v) {
 dem=(dem+1)%3;
 switch (dem)
 {
 case 0:
 ib.setImageResource(R.drawable.ic_launcher);
```

```
 break;
 case 1:
 ib.setImageResource(R.drawable.bground48);
 break;
 case 2:
 ib.setImageResource(R.drawable.bground300);
 break;
 }
});
```

**B3:** Chạy chương trình để xem kết quả.

#### **1.4.2.5. Sử dụng Styles trong Views**

Thông thường, khi cần định dạng cho textview sử dụng font chữ có màu trắng, chữ nghiêng, cỡ font là 20sp, bạn có thể lặp lại định dạng này lần lượt cho từng textview sử dụng trong ứng dụng của mình như sau:

```
 android:text="(chưa nhập nội dung)"
 android:textColor="#ffffff"
 android:textSize="20sp"
 android:textStyle="italic"
```

Android cung cấp tính năng định dạng style để dùng chung cho các widget mà bạn muốn áp dụng (trong ví dụ nào widget là textview) bằng cách tạo ra 1 style có tên là CustomTextStyle như sau:

- **B1:** xây dựng style: Tạo mới file res\values\style.xml với nội dung như sau:

```
<style name="CustomTextStyle">
 android:text="(chưa nhập nội dung)"
 android:textColor="#ffffffff"
 android:textSize="20sp"
 android:textStyle="italic"
</style>
```
  - **B2:** Áp dụng: gán thuộc tính style cho từng textview như sau:  
style="@style/CustomTextStyle"

Nhờ vậy, khi cần thay đổi định dạng cho style, ta chỉ cần sửa nội dung của style trong file res\values\style.xml, các thay đổi này sẽ được cập nhật trên tất cả các textview đang được gán cho style này

#### **1.4.2.6. Sử dụng tài nguyên trong folder Raw**

- Dịnh dạng Raw:
    - Là định dạng không nén và chưa qua xử lý bởi máy ảnh. Tên gọi *raw* hay *RAW* không phải là một từ viết tắt. Trong tiếng Anh, *Raw* có nghĩa là "thô, sống, nguyên".
    - Có rất nhiều định dạng raw, mỗi loại camera (từ mỗi hãng) sử dụng một định dạng cụ thể của nó.
    - Khác với định dạng JPEG thông thường, định dạng RAW tương tự như phim âm bản nhưng ở dạng kỹ thuật số, cho phép người dùng can thiệp sâu vào các thông số của bức ảnh như ánh sáng, màu sắc, độ chi tiết,... những điều vốn bị hạn chế khi xử lý ảnh JPEG. Việc hỗ trợ lưu ảnh RAW trên di động sẽ giúp người dùng tạo ra những bức ảnh đẹp hơn trước gấp nhiều lần (nếu biết cách tùy chỉnh thông qua các ứng dụng đi kèm).
  - Tài nguyên dạng raw được chứa trong folder *resources\raw*.
  - Truy cập tài nguyên raw: sử dụng cú pháp *R.raw.tên tài nguyên*.

Giả sử bạn đã có 1 file tài nguyên dạng raw với tên là instructions.txt. Để đọc file này, bạn cần sử dụng phương thức *openRawResource()*.

```

InputStream instructions =
 getResources().openRawResource(R.raw.instructions);

Sau đó thực hiện xử lý trên InputStream này để có kết quả như mong muốn. Phương
thức sau nhận tham số là 1 InputStream để xử lý và trả kết quả xử lý về dưới dạng chuỗi
public String inputStreamToString(InputStream is) throws IOException
{
 StringBuffer sBuffer = new StringBuffer();
 DataInputStream dataIO = new DataInputStream(is);
 String strLine = null;
 while ((strLine = dataIO.readLine()) != null)
 {
 sBuffer.append(strLine + "\n");
 }
 dataIO.close();
 is.close();
 return sBuffer.toString();
}

```

#### 1.4.2.7. Sử dụng tài nguyên trong folder Asset

Folder assets thường được dùng để lưu trữ các file cơ sở dữ liệu, các file không có định danh tài nguyên, ...

Sử dụng class *AssetManager* (*android.content.res.AssetManager*) để truy cập các file này.

### 1.5. BÀI TẬP TỔNG HỢP PHẦN I

#### 1.5.1. Mở rộng từ những bài thực hành đã thực hiện

- (i). Thêm tài nguyên về màu sắc với giá trị #00ff00 vào App\_05. Trong file *activity\_main.xml* của App\_05, đổi thuộc tính màu sắc của textview đang sử dụng có màu là màu vừa tạo lập. Chạy ứng dụng để xem kết quả.
- (ii). Tương tự, thêm 1 kích thước mới vào tài nguyên với giá trị là 22pt. Trong file *activity\_main.xml* của App\_05, đổi thuộc tính kích thước của textview đang sử dụng có màu cỡ chữ là cỡ chữ vừa tạo lập. Chạy ứng dụng để xem kết quả.
  - Tạo thêm 1 vài AVD với độ phân giải màn hình khác nhau. Chạy thử ứng dụng để xem kết quả có gì khác biệt với AVD ban đầu hay không?
  - Lần lượt thay đổi đơn vị đang sử dụng từ *pt* sang các đơn vị khác như *px*, *dp*, hoặc *sp*. Chạy thử ứng dụng để xem kết quả có gì khác biệt với AVD ban đầu hay không?
- (iii). Tương tự, thêm 1 tài nguyên hình ảnh vào App\_06 sao cho hình ảnh này có kích thước nhỏ hơn kích thước file đang có (ví dụ 12X12 hay 16X16). Thiết lập thuộc tính *src* của imagebutton sử dụng hình ảnh vừa thêm. Chạy ứng dụng để xem kết quả.

#### 1.5.2. Bài tập 2

- Tạo 1 activity gọi là *InputActivity*, gồm
  - 2 edittext gọi là *numEditText1* và *numEditText2* để người dùng nhập số.
  - 1 button “Add”. Khi người dùng click trên button “Add” sẽ lưu dữ liệu của 2 textbox vào 1 bundle.
- Tạo activity thứ 2 gọi là *AddActivity*, gồm
  - 1 textview chứa giá trị tổng của 2 giá trị có trong bundle của *InputActivity* gửi sang.
  - 1 button “Return”. Khi nhấn sẽ trả kết quả có trong textview về cho *InputActivity*.

## Phần II: GIAO DIỆN NGƯỜI DÙNG

- (i). Sử dụng Layouts
- (ii). Một số control cơ bản
- (iii). ActionBar & Menu Navigation
- (iv). Activities & Fragments
- (v). Các dạng Dialogs
- (vi). Lists, Grids, Galleries, & Flippers

### 2.1. SỬ DỤNG Layouts

#### 2.1.1. Một số khái niệm

##### 2.1.1.1. View

Là các đối tượng đồ họa như button, label, textbox, ... Một View được dẫn xuất từ các class `android.view.View`

##### 2.1.1.2. ViewGroups

- Một hoặc nhiều View có thể được nhóm lại cùng nhau bên trong một ViewGroup. Một ViewGroup cung cấp khả năng cho phép bố trí các đối tượng đồ họa chứa trong nó một cách phù hợp. Có thể sử dụng các ViewGroup để bố trí các đối tượng đồ họa theo dòng, theo cột, theo dạng bảng hay theo số chỉ định tọa độ cụ thể. Mỗi ViewGroup được dẫn xuất từ class `android.view.ViewGroup`.
- Android hỗ trợ các ViewGroup sau đây:
  - `LinearLayout`
  - `AbsoluteLayout`
  - `TableLayout`
  - `RelativeLayout`
  - `FrameLayout`
  - `ScrollView`
  - Một ViewGroup có thể đứng độc lập hoặc chứa trong nó nhiều ViewGroup khác.

##### 2.1.1.3. Layout

- Là 1 trong những giao diện (form) có trong ứng dụng. Một layout có thể chứa một hoặc nhiều ViewGroup.
- Thông thường, để xây dựng giao diện người dùng ta sử dụng file xml trong folder `res/layout` để định nghĩa các view cần dùng cùng với các thuộc tính quyết định cách thức hiển thị view đó trên cửa sổ ứng dụng. Ví dụ: nội dung file `activity_main.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:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="@string/hello"/>
</LinearLayout>
```

- XML layout có hỗ trợ thẻ `<include/>` để nhúng 1 layout A vào trong 1 layout B khác. Thường dùng khi muốn sử dụng cùng 1 header/footer cho nhiều layout. Để sử dụng thẻ này, trước tiên bạn tạo 1 file layout cho A. Sau đó, chèn thẻ sau tại vị trí cần chèn trong file layout B:

```
<include layout="@layout/A"/> device.
```

#### 2.1.1.4. Activity

- Gần như mọi Activity đều tương tác với người dùng nên class Activity đảm nhận việc tạo ra một cửa sổ (window) để người lập trình đặt lên đó một giao diện người dùng (layout) bằng phương thức `setContentView(View)` trong phương thức `onCreate()` của Activity.
- Ví dụ:

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

### 2.1.2. Một số vấn đề cần quan tâm khi thiết kế giao diện và viết mã lệnh

#### 2.1.2.1. Thiết kế giao diện

Việc thiết kế giao diện người dùng trong Android có thể thực hiện bằng một trong hai phương pháp: thủ tục và khai báo.

- Thiết kế giao diện bằng thủ tục: là sử dụng mã lệnh để tạo giao diện. Ví dụ, khi bạn đang lập trình một ứng dụng Swing, bạn viết mã Java để tạo ra và thao tác tất cả các đối tượng giao diện người dùng như JFrame và JButton. Do đó, Swing là thủ tục.
- Thiết kế giao diện bằng khai báo: không liên quan đến bất kỳ mã lệnh nào mà đơn giản chỉ là sử dụng ngôn ngữ đánh dấu dựa trên XML để mô tả giao diện của mình. Google khuyên nên sử dụng XML khai báo càng nhiều càng tốt do các mã XML thường ngắn gọn và dễ hiểu hơn việc đọc các mã lệnh Java tương ứng.

#### 2.1.2.2. Sử dụng độ đo trong Android

Khi lập trình trên máy tính, các lập trình viên luôn thiết kế giao diện dựa trên điểm ảnh (pixels), ví dụ, bạn có định độ rộng của 1 cột là 300 pixels, hay cho phép khoảng cách giữa các cột là 5 pixel, hoặc xác định kích thước của các icon là 16 X 16-pixel, ... Vấn đề xảy ra là nếu bạn chạy chương trình trên màn hình mới với độ phân giải cao hơn (số pixel trên mỗi inch tăng lên – dpi cao), khi đó giao diện người dùng xuất hiện nhỏ hơn kích thước thực của màn hình và có thể dẫn đến một số điểm trở nên quá khó đọc.

Độ đo phân giải độc lập (*Resolution-independent measurements*) giúp giải quyết vấn đề này. Android hỗ trợ cả các đơn vị sau:

- ***px*** (*pixels*): điểm ảnh trên màn hình.
- ***in*** (*inches*): Kích thước được đo bằng đơn vị *inches*.
- ***mm*** (*millimeters*): Kích thước được đo bằng *millimeters*.
- ***pt*** (*points*): 1/72 của 1 inch.
- ***dp*** (*density-independent pixels*): đơn vị đo trùu tượng dựa vào độ phân giải của màn hình. Trên màn hình có dpi là 160 (160 dots/inch) thì  $1dp = 1px$ .
- ***dip***: tương tự như ***dp***, thường được dùng trong các minh họa của Google.
- ***sp*** (*scale-independent pixels*): tương tự như ***dp*** nhưng có thể thu nhỏ lại tùy vào tùy chọn (*preference*) của người dùng.

Để giúp giao diện ứng dụng của bạn có khả năng mở rộng cho bất kỳ loại màn hình, Android khuyên bạn nên sử dụng đơn vị ***sp*** cho kích thước văn bản và đơn vị ***dip*** cho mọi thứ khác. Bạn cũng nên xem xét sử dụng đồ họa vector thay vì dùng bitmap cho ứng dụng của mình.

#### 2.1.2.3. Theme trong Android

Theme được xây dựng dựa trên ý nghĩa của Cascading Style Sheets (CSS) trong việc xây dựng web, ở đó các nội dung của màn hình được tách/chia cắt ra và được trình bày lại theo phong cách của người thiết kế web.

Một Theme là tập hợp các kiểu (styles) được xây dựng để áp dụng một giao diện khác với giao diện chuẩn cho các widget trong Android.

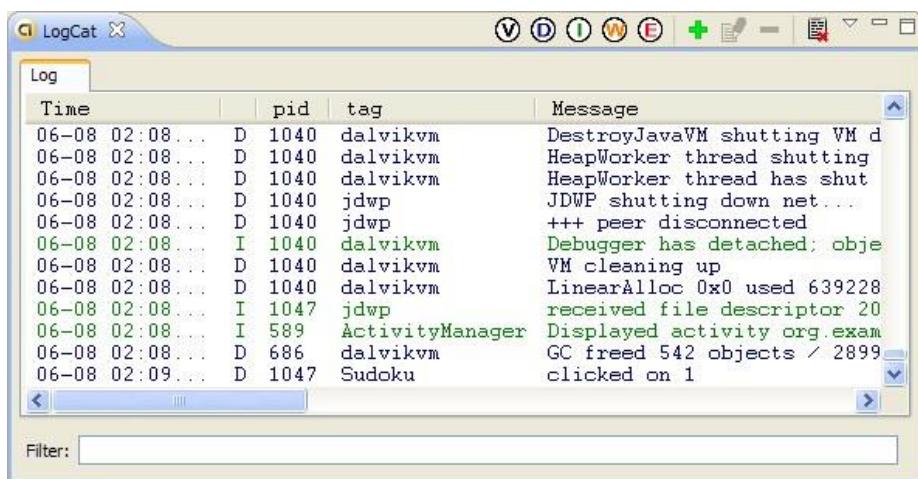
Android được đóng gói sẵn với một số Theme mà bạn có thể tham khảo theo tên, hoặc bạn có thể tạo nên theme của riêng mình bằng cách kế thừa các lớp con hiện có của Android trong file `res/values/styles.xml`.

```
<activity android:name=".About"
 android:label="@string/about_title"
 android:theme="@android:style/Theme.Dialog">
</activity>
```

Tiền tố `@android:style` trước tên kiểu cho biết Theme được dùng là một tham chiếu đến một tài nguyên của Android mà không phải là Theme do người dùng tự tạo.

#### 2.1.2.4. Gõ rối chương trình với Log Messages

- Log được xem như 1 quyển nhật ký ghi nhận quá trình sử dụng hệ thống nhằm giúp người quản trị giám sát được các hoạt động trong hệ thống của mình. Class `Log` trong Android cung cấp một số phương thức tĩnh để in các thông điệp mà người lập trình cần quan tâm. Các phương thức này gồm:
  - `Log.e()` : Errors - lỗi
  - `Log.w()` : Warnings - cảnh báo
  - `Log.i()` : Information - Thông tin
  - `Log.d()` : Debugging – gõ rối
  - `Log.v()` : Verbose – rườm rà, dư thừa
- Xem thông tin do class `Log` cung cấp: Người dùng sẽ không bao giờ nhìn thấy nhật ký này, nhưng là một nhà phát triển, bạn có thể xem nó trong một vài cách:
  - Trong Eclipse, chọn menu `Window>Show View/Others .../Android/LogCat`. Trong màn hình này, bạn có thể lọc thông tin cần xem theo từ khóa hoặc mức độ nghiêm trọng (severity) của sự việc theo ý của mình.



- Nếu bạn không sử dụng Eclipse, bạn có thể thấy kết quả tương tự bằng cách chạy lệnh `adb logcat`.
- Hoạt động của LogCat là trong suốt (người dùng không thấy được) và hoạt động này sẽ không can thiệp đến bất kỳ màn hình khác. Vì vậy bạn nên mở cửa sổ LogCat và để nó chạy trong suốt thời gian mà emulator đang chạy.

#### 2.1.2.5. Kết thúc ứng dụng đang chạy

Các ứng dụng trên di động không thực sự cần một nút Exit, bởi vì người dùng có thể nhấn phím Back và phím Home để kết thúc ứng dụng hoặc chuyển sang sử dụng ứng dụng khác.

Tuy nhiên Android vẫn cung cấp phương thức finish() để chấm dứt một activity. Phương thức này thực hiện tắt (shuts down) activity và trả quyền điều khiển về cho activity kế tiếp trong stack (nếu còn activity trong stack) hoặc màn hình Home (stack rỗng).

Ví dụ: minh họa 1 đoạn mã lệnh ngăn để sử dụng phương thức finish()

```
case R.id.exit_button:
 finish();
 break;
```

### 2.1.3. Xây dựng ứng dụng

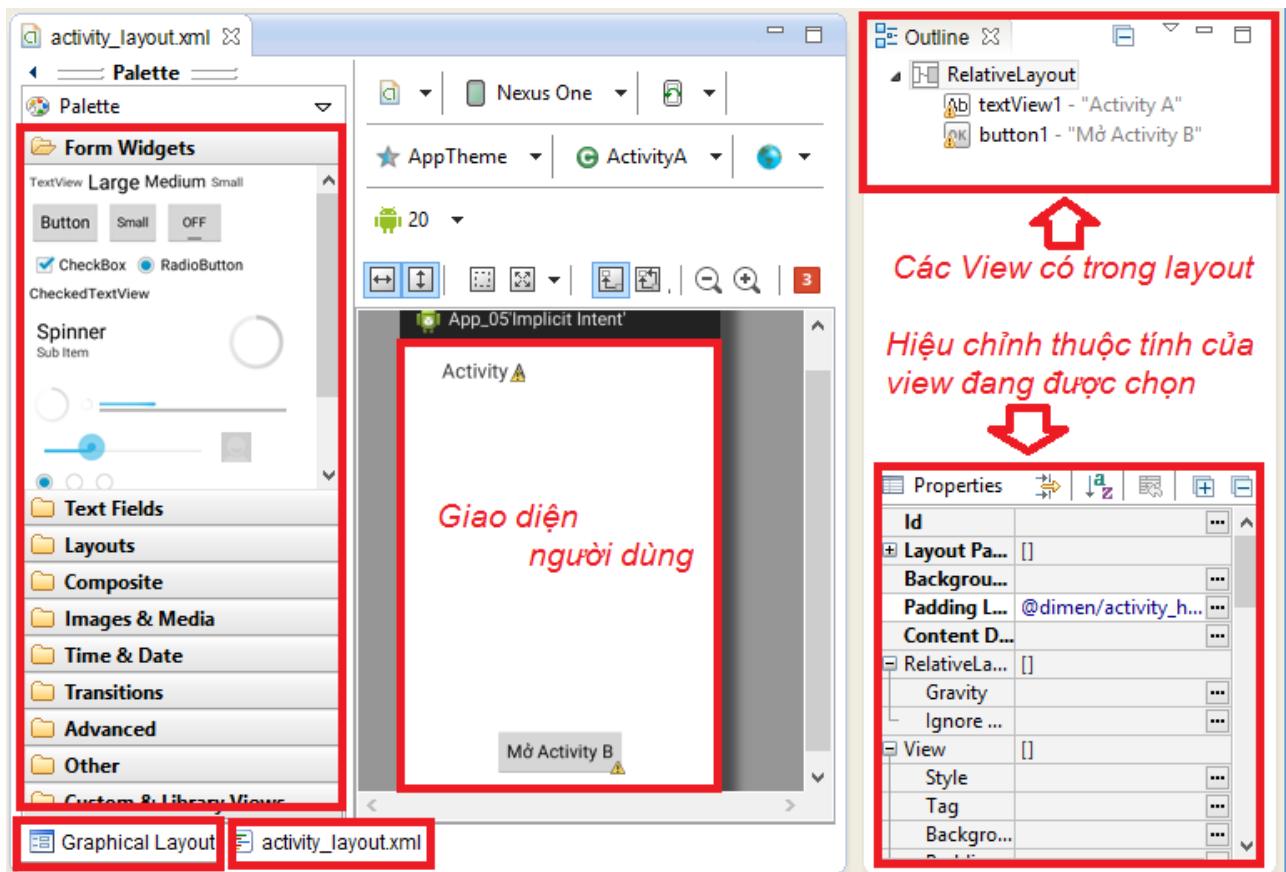
#### 2.1.3.1. Khai báo các chuỗi sử dụng trong ứng dụng

- Để tiện dụng, Android đề nghị các chuỗi cần hiển thị trong giao diện nên được khai báo trước trong file *res/values/strings.xml*.
- Nội dung các chuỗi có thể chứa các tag định dạng HTML đơn giản và có thể kéo dài nhiều dòng.
- Ví dụ: nội dung file *res/values/strings.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
 <string name="app_name">App_18</string>
 <string name="hello_world">Hello <i>world</i>!</string>
 <string name="action_settings">Settings</string>
</resources>
```

#### 2.1.3.2. Sử dụng Layout Resource Editor trong thiết kế giao diện

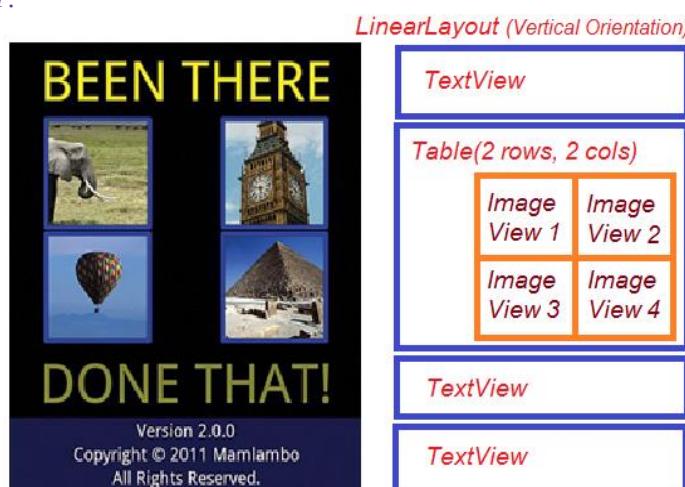
- Sử dụng *Layout Resource Editor* để thiết kế và xem trước giao diện của ứng dụng. Layout Resource Editor gồm 2 tab: Graphical Layout and main.xml.
  - Tab Graphical Layout*: cung cấp tính năng drag-and-drop trong khi thiết kế, nhờ vậy bạn có thể nhìn thấy giao diện của mình 1 cách trực quan.
  - Tab main.xml*: cung cấp tính năng thiết kế, hiệu chỉnh giao diện theo cơ chế dòng lệnh.
- Có thể chuyển đổi qua lại giữa 2 tab trong quá trình thiết kế giao diện. Thông qua việc chuyển đổi qua lại giữa 2 tab này trong quá trình thiết kế sẽ giúp bạn hiểu rõ hơn về mã XML của từng loại View/ViewGroup.



Hình 2-1 Minh họa Layout Resource Editor

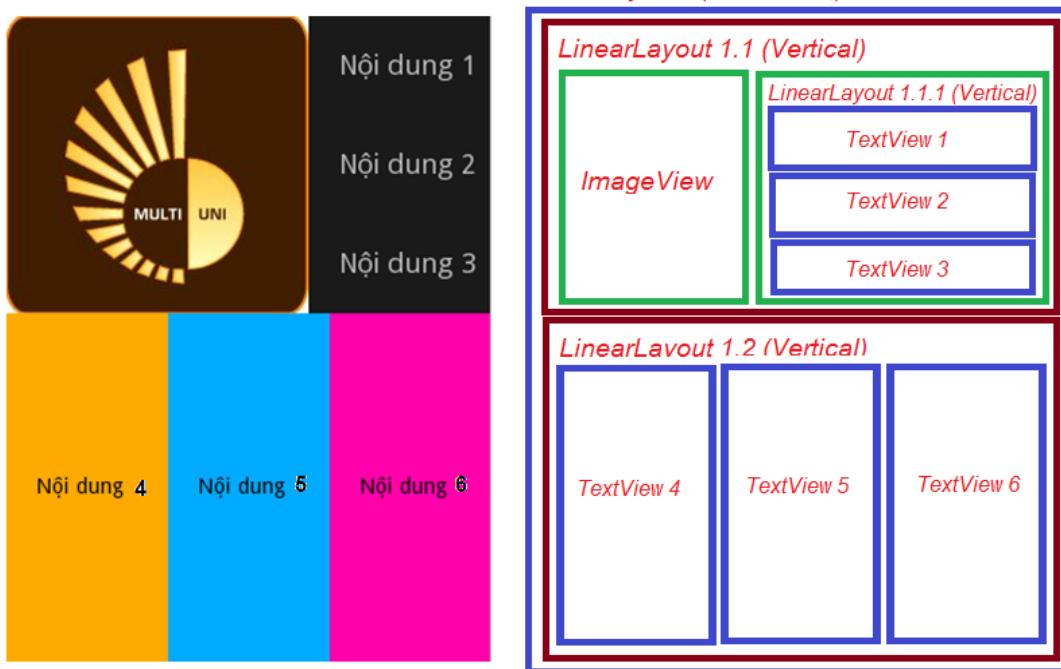
- Bước đầu tiên của việc thiết kế layout là bạn cần dự kiến việc phân bố các view trên màn hình như thế nào. Khi đã có ý tưởng về phân bố các thành phần trên màn hình mới sử dụng *Layout Resource Editor* để chính thức bắt tay vào thiết kế.

#### 2.1.3.2.1. Ví dụ:



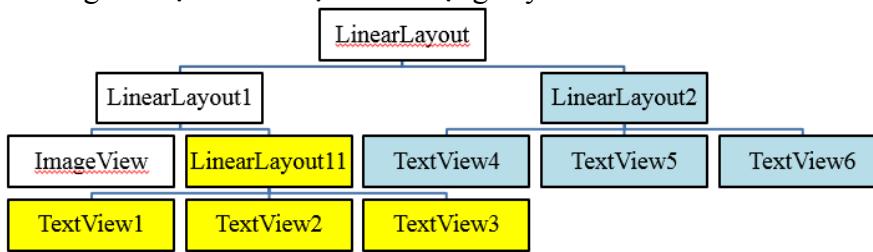
Hình 2-2 Minh họa giao diện cần xây dựng và dự kiến View/ViewGroup sử dụng của ví dụ 1

## 2.1.3.2.2. Ví dụ 2



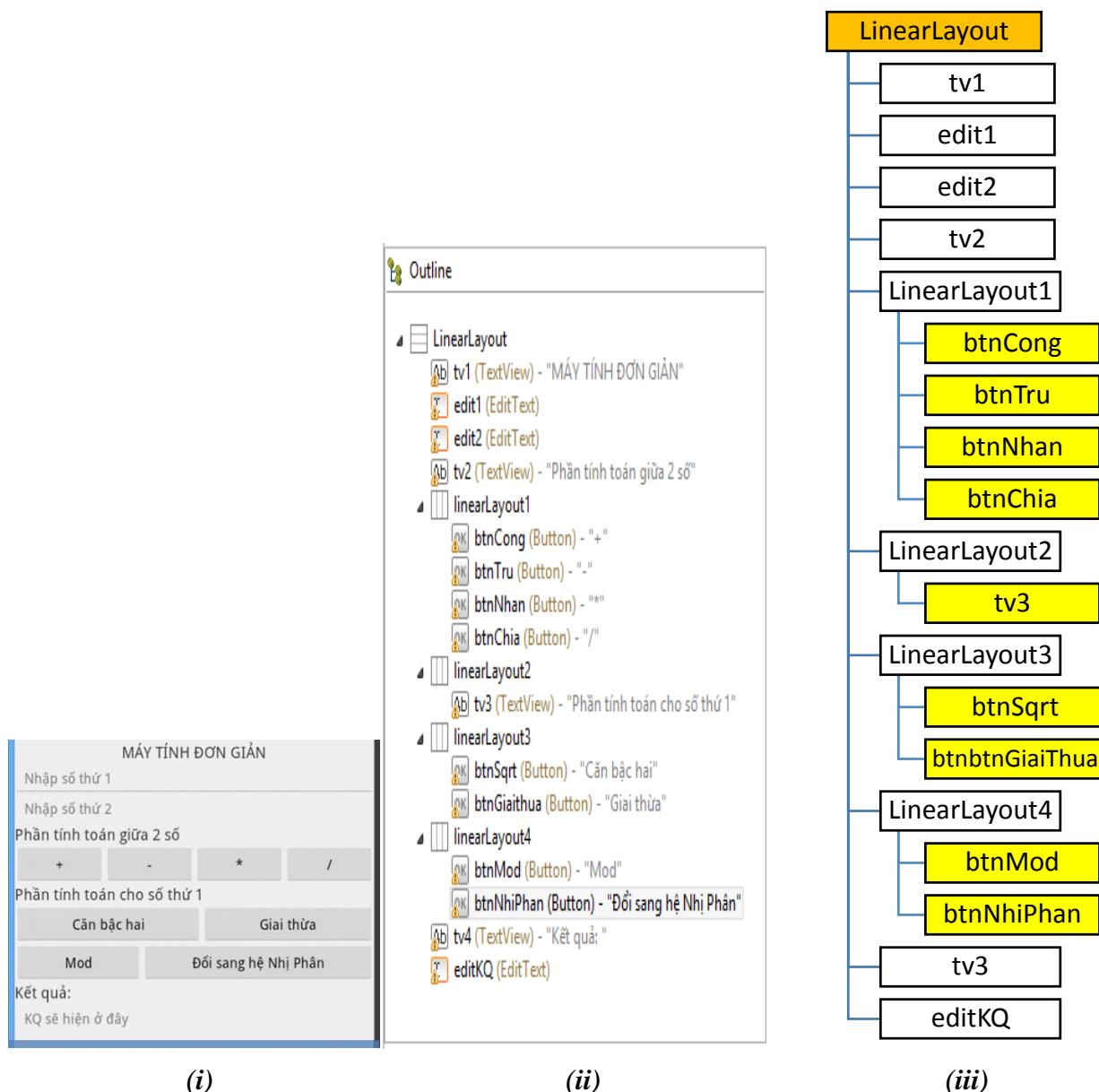
Hình 2-3 Minh họa giao diện cần xây dựng và dự kiến View/ViewGroup sử dụng của ví dụ 2

Có thể tóm tắt chức giao diện của ví dụ 2 dưới dạng cây:



Hình 2-4 Minh họa dưới dạng cây về dự kiến View/ViewGroup sử dụng của ví dụ 2

## 2.1.3.2.3. Ví dụ 3



Hình 2-5 Minh họa các cách nhìn giao diện cần xây dựng

Các cách nhìn giao diện

- (i) Giao diện thực tế cần xây dựng
- (ii) Giao diện nhìn từ cửa sổ Outline (sau khi thiết kế hoàn tất)
- (iii) Giao diện nhìn theo cách của nhà thiết kế.

## 2.1.3.3. Sử dụng Layout khi lập trình

- Để gọi layout vừa xây dựng, sử dụng phương thức theo cú pháp:

```
setContentView(R.layout.Tên file layout);
```

Ví dụ: `setContentView(R.layout.activity_main);`

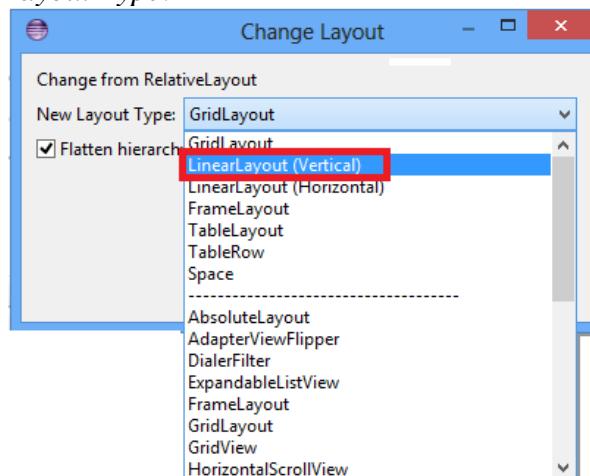
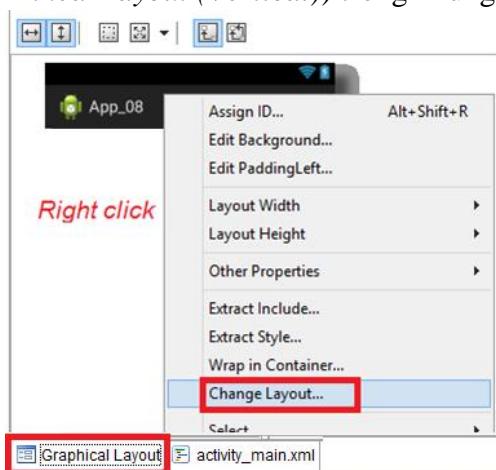
- Để truy cập đến các view có trong layout, sử dụng phương thức `findViewById()` theo cú pháp:

```
TênBiến=(KiểuCủaView) findViewById(R.id.TênCủaViewĐượcĐịnhNghĩaTrong layout.xml);
```

Ví dụ: `Button myButton= (Button) findViewById(R.id.button1);`

### 2.1.4. ViewGroup

- Các dạng ViewGroup thường dùng:
  - LinearLayout
  - TableLayout
  - AbsoluteLayout
  - RelativeLayout (được chọn mặc định cho layout khi mới tạo)
  - FrameLayout
  - ScrollViewLayout
- Bài thực hành trong phần này cần thực hiện chuyển đổi giữa các layout theo yêu cầu của từng layout cụ thể. Để chuyển *RelativeLayout* (đã được mặc định tạo ra trước đó) bằng *LinearLayout* (hay 1 layout khác), trong tab Graphical Layout, click phải vào giao diện project, chọn *Change Layout...*. Trong hộp thoại *Change Layout* vừa xuất hiện, chọn tên layout cần dùng (ví dụ như *LinearLayout (Vertical)*) trong khung *New Layout Type*.

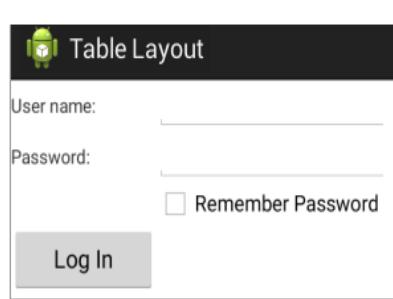


#### 2.1.4.1. TableLayout

- TableLayout** dùng để tổ chức các đối tượng View dưới dạng một bảng gồm nhiều dòng, nhiều cột.
- Dòng trong table:
  - Mỗi dòng được thể hiện bằng một thẻ **<TableRow>**.
  - Mỗi dòng có thể chứa một hoặc nhiều đối tượng View. Mỗi đối tượng View đặt trên một dòng sẽ tạo thành một ô (cột) trong giao diện bảng do *TableLayout* tạo ra.
  - Chiều rộng của mỗi cột được xác định bằng chiều rộng lớn nhất của các ô nằm trên cùng một cột.

## BÀI THỰC HÀNH App\_07

Yêu cầu



- Tạo App\_07 với layout có dạng như hình 2.8. Khi nhấn chọn trên các button sẽ cho mở ra các activity khác (nội dung của các activity này sẽ được mô tả lần lượt trong các phần kế tiếp của tài liệu).
- Tạo tiếp 2 layout mới có giao diện như minh họa trong hình 2.9 và 2.10

**Thực hiện:**

**B1:** Tạo mới project App\_07.

**B2:** Tạo giao diện cho màn hình chính (hình 2.8): Mở file `activity_main.xml`, chỉnh sửa và bổ sung nội dung file này để có như sau:

```

<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 android:orientation="horizontal" >
 <TableRow>
 <TextView android:id="@+id/textView1"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="View Group"
 android:paddingLeft="100dp"
 android:layout_span="2"
 android:textAppearance="?android:attr/textAppearanceLarge" />
 </TableRow>
 <TableRow>
 <Button android:id="@+id/button1"
 android:layout_width="160dp"
 android:layout_height="50dp"
 android:text="@string/Linear" />
 <Button android:id="@+id/button2"
 android:layout_width="160dp"
 android:layout_height="50dp"
 android:text="@string/absolute" />
 </TableRow>
 <TableRow>
 <Button android:id="@+id/button3"
 android:layout_width="match_parent"
 android:layout_height="50dp"
 android:text="@string/table" />
 <Button android:id="@+id/button4"
 android:layout_width="match_parent"
 android:layout_height="50dp"
 android:text="@string/relative" />
 </TableRow>
 <TableRow>
 <Button android:id="@+id/button5"
 android:layout_width="match_parent"
 android:layout_height="50dp"
 android:text="@string/frame" />
 <Button android:id="@+id/button6"
 android:layout_width="match_parent"
 android:layout_height="50dp"
 android:text="@string/scroll" />
 </TableRow>
 <TableRow>
 <TextView android:id="@+id/textView2"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Properties"
 android:paddingLeft="100dp"
 android:layout_span="2"
 android:textAppearance="?android:attr/textAppearanceLarge" />
 </TableRow>

```

```

<TableRow>
 <Button android:id="@+id/button7"
 android:layout_width="match_parent"
 android:layout_height="50dp"
 android:text="span_column" />
 <Button android:id="@+id/button8"
 android:layout_width="match_parent"
 android:layout_height="50dp"
 android:text="Padding" />
</TableRow>
</TableLayout>

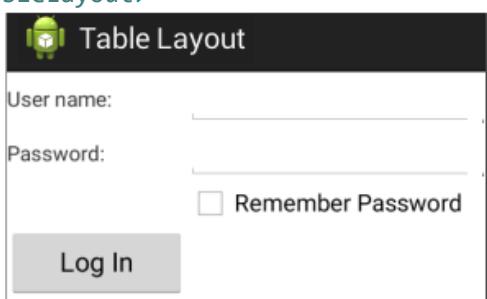
```

**B3:** Tạo mới file giao diện cho activity trong hình 2-9 và đặt tên file cho file này là *app07\_table.xml* với nội dung:

```

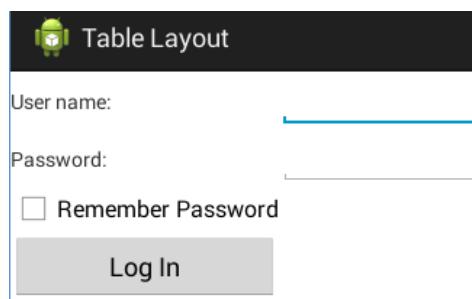
<?xml version="1.0" encoding="UTF-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_height="fill_parent"
 android:layout_width="fill_parent" >
 <TableRow>
 <TextView android:text="User Name:"
 android:width="120px"/>
 <EditText android:id="@+id/txtUserName"
 android:width="200px"/>
 </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>

```



Có dùng thẻ **<TextView>** rỗng

Hình 2-11 Minh họa kết quả của *app07\_table*



Không dùng thẻ **<TextView>** rỗng

Khi nạp phần code xml trên lên Activity, ta sẽ được giao diện như hình bên trái của hình 2-9: Ở ví dụ trên, giao diện đồ họa được tổ chức dưới dạng một bảng gồm có bốn dòng và hai cột. Ở ở ngay dưới ô chứa *EditText* Password có một thẻ **<TextView>** rỗng. Nếu không đặt thẻ này, thì *CheckBox* Remember Password sẽ xuất hiện ngay bên dưới *EditText* Password như hình bên phải.

**B4:** Tạo mới file giao diện cho activity trong hình 2-10 và đặt tên file cho file này là *app07\_span\_column.xml* với nội dung:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:stretchColumns="*"
 android:layout_height="match_parent"
 android:shrinkColumns="*" >
 <!-- Row 1 with single column -->
 <TableRow android:id="@+id/tableRow1"
 android:layout_height="wrap_content"
 android:layout_width="fill_parent"
 android:gravity="center_horizontal">
 <TextView android:id="@+id/TextView01"
 android:layout_width="match_parent"
 android:text="Row1"
 android:layout_height="wrap_content"
 android:background="#FF0000"
 android:gravity="center"
 android:layout_span="3"/>
 </TableRow>
 <!-- Row 2 with 3 columns -->
 <TableRow android:id="@+id/tableRow2"
 android:layout_height="wrap_content"
 android:layout_width="match_parent">
 <TextView android:id="@+id/TextView02"
 android:text="Row2 Col1"
 android:layout_weight="1"
 android:padding="20dp"
 android:background="#808080"
 android:gravity="center"/>
 <TextView android:id="@+id/TextView03"
 android:text="Row2 Col2"
 android:layout_weight="1"
 android:padding="20dp"
 android:background="#777777"
 android:gravity="center"/>
 <TextView android:id="@+id/TextView04"
 android:text="Row2 Col3"
 android:layout_weight="1"
 android:padding="20dp"
 android:background="#ACA899"
 android:gravity="center"/>
 </TableRow>
 <!-- Row 3 with 2 columns -->
 <TableRow android:id="@+id/tableRow3"
 android:layout_height="wrap_content"
 android:layout_width="fill_parent"
 android:gravity="center_horizontal">
 <TextView android:id="@+id/TextView05"
 android:text="Row3 col1"
 android:layout_weight="1"
 android:background="#FFFF00"
 android:padding="20dip"
 android:gravity="center"/>
 <TextView android:id="@+id/TextView06"
 android:text="Row3 Col2"
 android:background="#F7D896"
 android:layout_weight="1"
 android:padding="20dip"
 android:gravity="center"/>
 </TableRow>
</TableLayout>

```



Hình 2-12 Minh họa kết quả của app07\_table

**B5:** Viết mã lệnh cho activity “Table Layout”: copy file src/com.example.app\_07/MainActivity.java rồi paste vào chính folder này (folder src/com.example.app\_07) với tên mới là **MainActivity\_Table.java**. Hiệu chỉnh lại để có nội dung như sau (chú ý, lúc này layout được gọi là layout có tên app07\_table):

```
package com.example.app_07;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity_Table extends ActionBarActivity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.app07_table);
 }
 @Override
 public boolean onCreateOptionsMenu(Menu menu) {
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }
 @Override
 public boolean onOptionsItemSelected(MenuItem item) {
 int id = item.getItemId();
 if (id == R.id.action_settings) {
 return true;
 }
 return super.onOptionsItemSelected(item);
 }
}
```

**B6:** Viết mã lệnh cho activity “span\_column”: tương tự, copy file *MainActivity.java* rồi paste vào chính folder này (folder src/com.example.app\_07) với tên mới là **MainActivity\_Span\_Column.java**. Hiệu chỉnh lại để có nội dung như sau (chú ý, lúc này layout được gọi là layout có tên app07\_span\_column):

```
package com.example.app_07;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity_Span_Column extends ActionBarActivity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.app07_span_column);
 }
 @Override
 public boolean onCreateOptionsMenu(Menu menu) {
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }
 @Override
 public boolean onOptionsItemSelected(MenuItem item) {
 int id = item.getItemId();
 if (id == R.id.action_settings) {
 return true;
 }
 return super.onOptionsItemSelected(item);
 }
}
```

**B7:** Viết mã lệnh cho activity chính: bổ sung mã lệnh trong file *MainActivity.java* để khi người dùng nhấn chọn button “Table Layout” và “span\_column” sẽ xuất hiện các activity tương ứng.

```

package com.example.app_07;
import android.support.v7.app.ActionBarActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MainActivity extends ActionBarActivity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 Button b3 = (Button)findViewById(R.id.button3);
 b3.setOnClickListener(new OnClickListener()
 { public void onClick(View v)
 { Intent intent = new Intent(MainActivity.this, MainActivity_Table.class);
 startActivity(intent);
 }
 });
 Button b7 = (Button)findViewById(R.id.button7);
 b7.setOnClickListener(new OnClickListener()
 { public void onClick(View v)
 { Intent intent = new Intent(MainActivity.this, MainActivity_Span_Column.class);
 startActivity(intent);
 }
 });
 }
 @Override
 public boolean onCreateOptionsMenu(Menu menu) {
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }
 @Override
 public boolean onOptionsItemSelected(MenuItem item) {
 int id = item.getItemId();
 if (id == R.id.action_settings) {
 return true;
 }
 return super.onOptionsItemSelected(item);
 }
}

```

### 2.1.4.2. LinearLayout

#### 2.1.4.2.1. Giới thiệu

Là loại ViewGroup cho phép sắp xếp các đối tượng View (chứa trong nó) trong một cột (hay một hàng) đơn. Ví dụ: với nội dung file main.xml trong folder layout 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">
 <TextView android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="@string/hello" />
</LinearLayout>

```

Ta quan sát thấy rằng phần tử gốc là **<LinearLayout>** có chứa một thành phần **<TextView>**. Phần tử **<LinearLayout>** đó sẽ xác định cách hiển thị của các đối tượng View chứa trong nó thông qua các thuộc tính.

### 2.1.4.2.2. Tập thuộc tính thường dùng của View và ViewGroup:

Thuộc tính	Đặc tả	Giá trị
layout_width	Xác định chiều rộng của một View hoặc ViewGroup	Giá trị: " <b>fill_parent</b> ", " <b>wrap_content</b> ", <b>100px</b> (giá trị xác định)
layout_height	Xác định chiều cao của một View hoặc ViewGroup	
layout_marginTop	Xác định phần mở rộng phía bên trên của View hoặc ViewGroup	
layout_marginBottom	Xác định phần mở rộng phía bên dưới của View hoặc ViewGroup	
layout_marginLeft	Xác định phần mở rộng phía bên trái của View hoặc ViewGroup	
layout_marginRight	Xác định phần mở rộng phía bên phải của View hoặc ViewGroup	
layout_gravity	Xác định cách đặt các đối tượng view con	<ul style="list-style-type: none"> <li>chỉ dùng khi View nằm trong LinearLayout hoặc TableLayout</li> </ul>
layout_weight	Xác định phần kích thước thể hiện của View hoặc ViewGroup (Sử dụng với thuộc tính layout_width hoặc layout_height )	<ul style="list-style-type: none"> <li>chỉ dùng khi View nằm trong LinearLayout hoặc TableLayout</li> <li>Giá trị: "<b>fill_parent</b>", "<b>wrap_content</b>", <b>100px</b> (giá trị xác định)</li> </ul>
layout_x	Xác định toạ độ x của the View hoặc ViewGroup	
layout_y	Xác định toạ độ y của the View hoặc ViewGroup	
orientation	Dùng để xác định các đối tượng đó hoà chúa trong LinearLayout được sắp xếp theo chiều dọc (vertical) hay chiều ngang.(horizontal)	Mặc định là Horizontal

- Chú ý: một vài thuộc tính trên chỉ được sử dụng khi một View được định nghĩa rõ ràng trong một ViewGroup cụ thể.

#### Nhóm các thuộc tính Layout margin

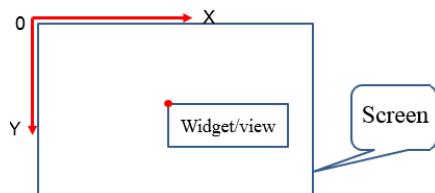
- Xác định khoảng cách giữa view với thành phần (viewgroup hoặc layout) về các phía trên/dưới/trái/phải.
- Một ứng dụng sử dụng layout margin trong lập trình: giả sử ta muốn khi button được “nhấn”, người dùng có “cảm giác” button sẽ hơi lùi xuống và dịch sang phải. Để thực hiện điều này bạn cần thiết lập thuộc tính margin của top và left trong layout như sau:

```
 android:layout_marginTop="40dp"
 android:layout_marginLeft="40dp"
```

Tác dụng này áp dụng được cho LinearLayout với thuộc tính orientation là vertical. Bạn sẽ nhận được kết quả khác khi sử dụng 2 thuộc tính trên trong những dạng layout khác.

– Thuộc tính `layout_x` và `layout_y`:

- `android:layout_x`: Xác định toạ độ x của đối tượng trong toạ độ Oxy.
- `android:layout_y`: Xác định toạ độ y của đối tượng trong toạ độ Oxy.



Hình 2-13 Xác định tọa độ trên màn hình

Lưu ý Gốc toạ được xác định là điểm trên cùng ở góc trái của **ViewGroup** gần nhất chứa nó.

– Thuộc tính `layout_gravity`:

- Sử dụng để canh lè cho view so với vật chứa nó.
- Thuộc tính này có thể nhận 1 trong các giá trị: left, right, center
- Ví dụ: với việc thiết lập thuộc tính `layout_gravity="right"` đối với button1, các button2 và button3 không sử dụng thuộc tính này, ta có kết quả



Hình 2-14 Minh họa thuộc tính gravity

– Thuộc tính `layout_weight`:

- thường dùng với `LinearLayout` và thuộc tính `orientation` là horizontal.
- Quy định tỷ lệ phân chia diện tích sử dụng với các view khác trong cùng `viewgroup`. Khi thuộc tính này không sử dụng, các thuộc tính `width`, `height` sẽ được xét đến.
- Giá trị gán cho thuộc tính này có thể là số lẻ (VD 0.5) và có thể lớn hơn 1. Tuy nhiên, giá trị này càng lớn thì kích thước càng nhỏ. Ví dụ: có 2 view A và B với giá trị `layout_weight` lần lượt của A là 4 và của B là 1. Vậy chiều ngang của màn hình sẽ chia làm 5 phần, A mặc dù có giá trị 4 nhưng chỉ chiếm 1 phần, ngược lại B với giá trị 1 lại chiếm 4 phần.
- Để dễ hình dung thuộc tính `layout_weight` làm việc như thế nào, bạn có thể thực hiện:
  - Tạo mới 1 project sử dụng `LinearLayout` với thuộc tính `orientation` có giá trị là horizontal.
  - Thêm 2 button vào layout với `layout_width` là `match_parent`.
  - Lần lượt thay đổi giá trị của thuộc tính này để kiểm tra kết quả hiển thị.

#### 2.1.4.2.3. Minh họa cách sử dụng các thuộc tính

– Thuộc tính `layout_weight` và `layout_gravity`: chỉ được sử dụng nếu View nằm trong `LinearLayout` hoặc trong `TableLayout`.

- Ví dụ 1: chiều rộng của `TextView` được đặt bằng với chiều rộng của phần tử cha của nó (`fill_parent`), và chiều cao là chiều cao cần thiết để `TextView` hiển thị đoạn text (`wrap_content`)

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

Trong trường hợp này, chiều rộng của `TextView` có thể bằng với chiều rộng màn hình).

- Ví dụ 2: chiều rộng của `TextView` được đặt là giá trị cố định (105 pixels)

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

```

<TextView android:layout_width="105px"
 android:layout_height="wrap_content"
 android:text="@string/hello">
</TextView>
<Button android:layout_width="100px"
 android:layout_height="wrap_content"
 android:text="Button">
</Button>
</LinearLayout>

```

Khi phần code giao diện trên được nạp lên Activity sẽ được giao diện như sau:

- **Thuộc tính Orientation**: có thể gán hai giá trị tương ứng sau:



Hình 2-15 Minh họa sử dụng thuộc tính Orientation

- **vertical**: xác định các đối tượng đồ họa chứa trong **LinearLayout** được sắp xếp theo chiều dọc.
- **horizontal**: (mặc định) xác định các đối tượng đồ họa chứa trong **LinearLayout** được sắp xếp theo chiều ngang.

- Ví dụ 3: Tạo mới 1 layout, trong đó thuộc tính android.Orientation="Vertical"

```

<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayout android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 android:orientation="vertical"
 xmlns:android="http://schemas.android.com/apk/res/android">
 <TextView android:layout_width="105px"
 android:layout_height="wrap_content"
 android:text="@string/hello">
 </TextView>
 <Button android:layout_width="100px"
 android:layout_height="wrap_content"
 android:text="Button">
 </Button>
</LinearLayout>

```

- **Giá trị của thuộc tính kích thước**: một đối tượng đồ họa trong Android được phân thành hai loại:

- Giá trị xác định: Xác định cụ thể giá trị kích thước. Ví dụ: 150px.
- Giá trị linh hoạt. Giá trị này phụ thuộc vào một trong hai yếu tố:
  - Nội dung mà một đối tượng đồ họa muốn hiển thị. Với View thì nội dung này là đoạn text,... Với ViewGroup là các phần tử con mà nó chứa.
  - Phần không gian mà đối tượng cha còng trống.

- **Thuộc tính android:layout\_height và android:layout\_weight**

Cả hai thẻ **Button** và **EditText** đều có thuộc tính **android:layout\_height="wrap\_content"** và **android:layout\_weight** với các giá trị tương ứng: **Button: 0.2** **EditText: 0.8**. Nếu thuộc tính kích thước của một đối tượng đồ họa (**android:layout\_height** và **android:layout\_width**) xác định là **wrap\_content** thì kích thước này sẽ vừa đủ để hiển thị nội dung của nó (với View thì nội dung này là đoạn text,... Với

*ViewGroup* là các phần tử con mà nó chứa). Tuy nhiên nếu sử dụng thêm thuộc tính `android:layout_weight` thì kích thước của đối tượng đồ họa này sẽ phụ thuộc vào giá trị của thuộc tính này. Để giải thích về cách sử dụng thuộc tính `android:layout_weight` ta xem xét lại ví dụ trên như sau:

- Ta có một thẻ gốc *LinearLayout* chứa ba đối tượng *TextView*, *Button*, *EditText*. Cả ba đối tượng này được sắp xếp theo chiều dọc từ trên xuống.
- Chiều cao của *TextView* là "`wrap_content`" nghĩa là nó sẽ có một độ cao vừa đủ để hiển thị một đoạn text cố định "Hello Android" - Ta sẽ bàn về khái niệm `@string/hello` ở phần sau của tài liệu).
- Sau khi xác định được chiều cao của *TextView* thì ta sẽ còn lại một khoảng trống dùng để sắp xếp *Button* và *EditText*. Ta gọi khoảng trống này là H. Chiều cao của 2 đối tượng này là "`wrap_content`" và giá trị `android:layout_weight` lần lượt là 0.2 và 0.8. Lúc này chiều cao của *Button* sẽ là 0.2\*H và chiều cao của *EditText* là 0.8\*H.

Lưu ý: rằng tổng `android:layout_weight` (hoặc `android:layout_height`) của các đối tượng trong cùng một *ViewGroup* luôn là 1.0 (`android:layout_weight` của *Button* + `android:layout_weight` của *EditText* = 1.0)

– Các giá trị có thể gán cho thuộc tính `android:layout_gravity`:

Giá trị	Mô tả
<code>top</code>	Đặt đối tượng đồ họa lên đỉnh trên cùng của <i>ViewGroup</i> chứa nó. Thuộc tính này không thay đổi giá trị kích thước
<code>bottom</code>	Đặt đối tượng đồ họa xuống dưới cùng của <i>ViewGroup</i> chứa nó. Thuộc tính này không thay đổi giá trị kích thước.
<code>left</code>	Đặt đối tượng đồ họa bên trái của <i>ViewGroup</i> chứa nó. Thuộc tính này không thay đổi giá trị kích thước.
<code>right</code>	Đặt đối tượng đồ họa bên phải của <i>ViewGroup</i> chứa nó. Thuộc tính này không thay đổi giá trị kích thước.
<code>center_vertical</code>	Đặt đối tượng đồ họa canh giữa theo chiều dọc của <i>ViewGroup</i> chứa nó. Thuộc tính này không thay đổi giá trị kích thước.
<code>fill_vertical</code>	Tăng kích thước của đối tượng đồ họa tràn đầy chiều cao <i>ViewGroup</i> chứa nó.
<code>center_horizontal</code>	Đặt đối tượng đồ họa canh giữa theo chiều ngang của <i>ViewGroup</i> chứa nó. Thuộc tính này không thay đổi giá trị kích thước.
<code>fill_horizontal</code>	Tăng kích thước của đối tượng đồ họa tràn đầy chiều rộng <i>ViewGroup</i> chứa nó.
<code>center</code>	Đặt đối tượng đồ họa ở chính giữa của <i>ViewGroup</i> chứa nó. Thuộc tính này không thay đổi giá trị kích thước.
<code>fill</code>	Tăng kích thước của đối tượng đồ họa tràn đầy kích thước <i>ViewGroup</i> chứa nó.
<code>clip_vertical</code>	Góc trên hoặc góc dưới của view sẽ bị cắt xén để vừa với <i>ViewGroup</i> chứa nó. Việc xén dựa trên cơ sở của <code>vertical_gravity</code> : top gravity sẽ xén góc dưới và bottom gravity sẽ xén góc trên. 2 cách xén này sẽ không xảy ra đồng thời.
<code>clip_horizontal</code>	Bên trái hoặc bên phải của view sẽ bị xén để vừa với <i>ViewGroup</i> chứa nó. Việc xén dựa trên cơ sở của <code>horizontal_gravity</code> : left gravity sẽ xén bên phải và right gravity sẽ xén bên trái. 2 cách xén này sẽ không xảy ra đồng thời.

Lưu ý Thuộc tính `android:layout_gravity` có thể gán nhiều hơn một thuộc tính.

VD: `android:layout_gravity: "top|left"`

- Thuộc tính `android:gravity`: (khác với `layout_gravity`) canh lè nội dung xuất hiện trong view (trung tâm, trái, phải, trên, dưới...)

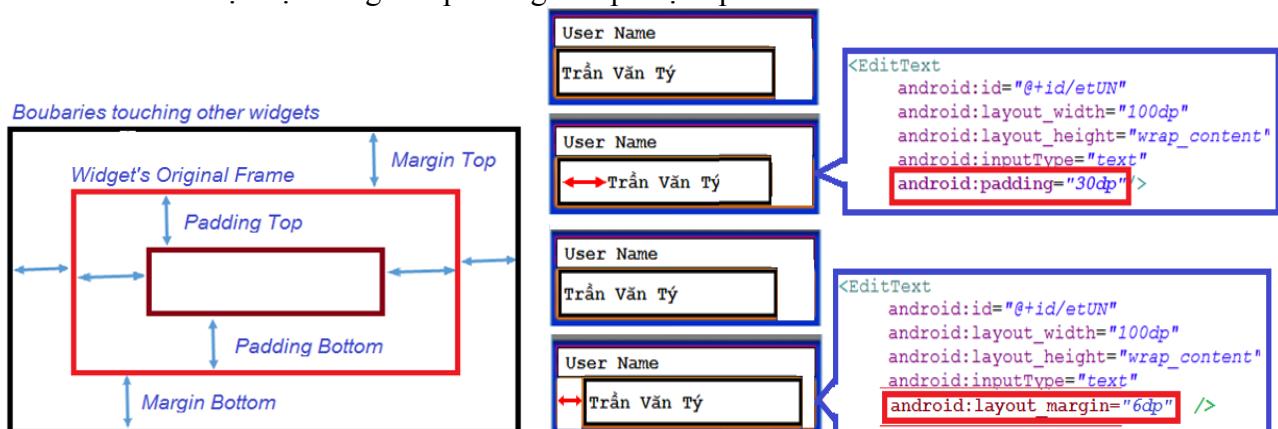
So sánh giữa `android:gravity` và `android:layout_gravity`:



Hình 2-16 Minh họa sự khác biệt giữa 2 thuộc tính `layout_gravity` và `gravity`

- Thuộc tính `android:padding`

- Quy định khoảng cách khung viền ngoài của widget đến nội dung xuất hiện trong widget (ví dụ chuỗi hiển thị trên button).
- Có thể thiết lập padding cho toàn bộ widget hoặc thiết lập riêng cho các lè trên, dưới, trái, phải.
- Đơn vị được dùng cho padding là dip hoặc dp.

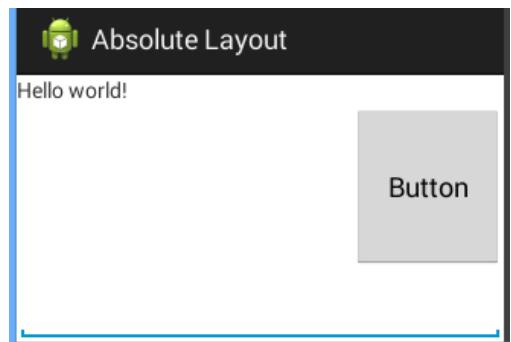


Hình 2-17 Minh họa sự khác biệt giữa 2 thuộc tính `margin` và `Padding`

## BÀI THỰC HÀNH app07\_linear

### ☞ Yêu cầu

- Vẫn trong project App\_07. Tạo thêm 1 activity có dạng như hình 2.18 với tên MainActivity\_Linear. Khi nhấn chọn button “Linear Layout” sẽ gọi activity này.



Hình 2-18 Minh họa kết quả của app07\_linear

### ☞ Thực hiện

- B8:** Tạo mới file giao diện cho activity trong hình 2-18 và đặt tên file cho file này là `app07_linear.xml` với nội dung:

```
<?xml version="1.0" encoding="UTF-8"?>
<LinearLayout android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:orientation="vertical">
 <TextView android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="Text View"/>
 <Button android:layout_width="100dp"
 android:layout_height="wrap_content"
 android:text="Button"/>

```

```

 android:layout_gravity="right"
 android:layout_weight="0.2"/>
 <EditText android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:textSize="18sp"
 android:text="Edit Text"
 android:layout_weight="0.8"/>
</LinearLayout>

```

### Nhận xét về mã lệnh vừa có

- ✓ *Button* được gán vào bên phải của phần tử cha của nó (trong trường hợp này là *LinearLayout*) sử dụng thuộc tính *layout\_gravity*, và sử dụng thuộc tính *layout\_weight* cho các *Button*, *EditText* (tổng giá trị của *layout\_weight* bằng 1). Sau khi nạp phần giao diện trên vào Activity ta sẽ được giao diện như sau. Phân tích phân giao diện vừa có ta nhận thấy:
- ✓ Thẻ gốc của toàn bộ giao diện là *LinearLayout* với hai thuộc tính *android:layout\_width="fill\_parent"* và *android:layout\_height="fill\_parent"* xác định chiều rộng và chiều cao *LinearLayout* sẽ chiếm toàn chiều rộng và chiều cao của đối tượng chứa nó. Nhưng do thẻ *LinearLayout* này là thẻ gốc (Thẻ ngoài cùng) nên nó sẽ được đặt lên màn hình điện thoại khi ứng dụng được mở lên. Vì vậy nó sẽ chiếm toàn bộ chiều rộng và chiều cao của màn hình điện thoại Đây là một trường hợp giá trị kích thước của một đối tượng đồ họa là linh hoạt và phụ thuộc vào không gian còn trống của đối tượng cha (Trong trường hợp này đối tượng cha là màn hình điện thoại, do chưa chứa gì hết nên nó được xem là rỗng).
- ✓ Thẻ *TextView* có thuộc tính *android:layout\_width="fill\_parent"* và *android:layout\_height="wrap\_content"* xác định kính thước chiều ngang của *TextView* sẽ chiếm toàn bộ khoảng trống còn lại của đối tượng chứa nó (trong *LinearLayout*), và chiều cao của *TextView* sẽ phụ thuộc vào nội dung nó cần hiển thị (*TextView* chiếm một khoảng chiều cao bằng với chiều cao cần để hiển thị đoạn text mà nó đang giữ).
- ✓ Thẻ *Button* có thuộc tính *android:layout\_width="100px"* xác định kích thước chiều rộng của button là 100px Đây là một trường hợp mà giá trị kích thước của một đối tượng đồ họa là xác định (chiều rộng = 100px).
- ✓ Thẻ *Button* có thuộc tính *android:layout\_gravity="right"* xác định rằng đối tượng **Button** khi được nạp lên giao diện sẽ canh lề bên phải so với đối tượng chứa nó.

**B9:** Viết mã lệnh cho activity “Linear Layout”: copy file *src/com.example.app\_07/MainActivity.java* rồi paste vào chính folder này (folder *src/com.example.app\_07*) với tên mới là **MainActivity\_Linear.java**. Hiệu chỉnh lại để có nội dung như sau (chú ý, lúc này layout được gọi là layout có tên *app07\_linear*):

```

package com.example.app_07;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
public class MainActivity_Linear extends ActionBarActivity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.app07_linear);
 }
 @Override
 public boolean onCreateOptionsMenu(Menu menu) {
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }
 @Override

```

```

public boolean onOptionsItemSelected(MenuItem item) {
 int id = item.getItemId();
 if (id == R.id.action_settings) {
 return true;
 }
 return super.onOptionsItemSelected(item);
}
}

```

**B10:** Bổ sung mã lệnh trong file MainActivity.java để khi người dùng nhấn chọn button “*Linear Layout*” sẽ xuất hiện activity vừa tạo ở bước trước.

```

package com.example.app_07;
import android.support.v7.app.ActionBarActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
public class MainActivity extends ActionBarActivity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 Button b1 = (Button)findViewById(R.id.button1);
 b1.setOnClickListener(new OnClickListener()
 { public void onClick(View v)
 {
 Intent intent = new Intent(MainActivity.this, MainActivity_Linear.class);
 startActivity(intent);
 }
 });
 Button b3 = (Button)findViewById(R.id.button3);
 b3.setOnClickListener(new OnClickListener() {
 public void onClick(View v) {
 Intent intent = new Intent(MainActivity.this, MainActivity_Table.class);
 startActivity(intent);
 }
 });
 Button b7 = (Button)findViewById(R.id.button7);
 b7.setOnClickListener(new OnClickListener() {
 public void onClick(View v) {
 Intent intent = new Intent(MainActivity.this, MainActivity_Span_Column.class);
 startActivity(intent);
 }
 });
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
 int id = item.getItemId();
 if (id == R.id.action_settings) {
 return true;
 }
 return super.onOptionsItemSelected(item);
}
}

```

**B11:** Bổ sung mã lệnh trong file AndroidManifest.xml để khai báo thêm activity vừa tạo.

## BÀI THỰC HÀNH app07\_padding\_margin

### ☞ Yêu cầu

Vẫn trong project App\_07, tạo mới 1 activity `MainActivity_Padding_Margin`. Activity này sẽ sử dụng file layout `app07_padding_margin` gồm 6 button như hình 2.19. Biết rằng:

- **Button 1:** không sử dụng 2 thuộc tính `android:padding` lẫn `android:margin`
- **Button 2:** có sử dụng thuộc tính `android:padding="30dp"` nhưng không sử dụng thuộc tính `android:margin`.
- **Button 3:** không sử dụng `padding` và có `margin="10dp"`.
- **Button 4:** sử dụng thuộc tính `padding="30dp"` và `margin="10dp"`.
- **Button 5:** sử dụng thuộc tính `android:paddingLeft="30dp"` và `android:paddingTop="30dp"`. Không sử dụng thuộc tính `android:margin`
- **Button 6:** sử dụng thuộc tính `android:paddingRight="30dp"` và `android:paddingBottom="30dp"`. Không sử dụng thuộc tính `android:margin`



Hình 2-19 Minh họa kết quả của `app07_padding_margin`

### ☞ Thực hiện

**B12:** Tạo mới file `app07_padding_margin`, với nội dung như sau:

```
<?xml version="1.0" encoding="UTF-8"?>
<LinearLayout android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:orientation="vertical">
 <Button android:id="@+id/button1"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="@string/btn1" />
 <Button android:id="@+id/button2"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:padding="30dp"
 android:text="@string/btn2" />
 <Button android:id="@+id/button3"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_margin="10dp"
 android:text="@string/btn3" />
 <Button android:id="@+id/button4"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:padding="30dp"
 android:layout_margin="10dp"
 android:text="@string/btn4" />
 <Button android:id="@+id/button5"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:paddingLeft="30dp"
 android:paddingTop="30dp"
 android:text="@string/btn5" />
 <Button android:id="@+id/button6"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:paddingRight="30dp"
 android:paddingBottom="30dp"
 android:text="@string/btn6" />
</LinearLayout>
```

- B13:** Tạo mới file `src/com.example.app_07/MainActivity_Padding_Margin.java`. File này sẽ gọi layout `app07_padding_margin` vừa tạo. (Sinh viên tự thực hiện).
- B14:** Bổ sung mã lệnh trong file `MainActivity.java` để khi người dùng nhấn chọn button Padding|Margin sẽ cho mở activity. (Sinh viên tự thực hiện).
- B15:** Khai báo activity mới trong file `AndroidManifest.xml`. (Sinh viên tự thực hiện).

### 2.1.4.3. AbsoluteLayout

- AbsoluteLayout cho bạn chỉ định chính xác vị trí của các thành phần con

## BÀI THỰC HÀNH app07\_absolute

### ☞ Yêu cầu

Vẫn trong project App\_07, tạo mới 1 activity `MainActivity_Absolute`. Activity này sẽ sử dụng file layout `app07_absolute` như hình 2.20.

### ☞ Thực hiện

- B16:** Tạo mới file layout tên `app07_absolute` với nội dung:

```
<?xml version="1.0" encoding="UTF-8"?>
<AbsoluteLayout android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 xmlns:android=
" http://schemas.android.com/apk/res/android" >
 <Button android:layout_width="188px"
 android:layout_height="wrap_content"
 android:text="Button1"
 android:layout_x="126px"
 android:layout_y="361px" />
 <Button android:layout_width="113px"
 android:layout_height="wrap_content"
 android:text="Button2"
 android:layout_x="12px" />
</AbsoluteLayout>
```



Hình 2-20 Minh họa kết quả của app07\_absolute

- B17:** Tương tự như trên, sinh viên tự thực hiện các công việc sau:

- Tạo mới file `src/com.example.app_07/MainActivity_Absolute.java`.
- Bổ sung mã lệnh trong file `MainActivity.java` để khi người dùng nhấn chọn button Absolute sẽ cho mở activity vừa có.
- Khai báo activity mới trong file `AndroidManifest.xml`.

### 2.1.4.4. RelativeLayout

- RelativeLayout cho phép bạn chỉ định mối quan hệ giữa các đối tượng View/ViewGroup.
- Chú ý rằng mỗi View được nhúng bên trong RelativeLayout có các thuộc tính cho phép xác định mối quan hệ giữa chúng RelativeLayout chứa nó hoặc các View/ViewGroup khác. Một số thuộc tính là:
  - Thuộc tính có giá trị là true/false

Thuộc tính	Vị trí của view đang xét khi giá trị bằng true
<code>layout_alignParentTop</code>	Nằm sát <u>trên cùng</u> của RelativeLayout chứa nó.
<code>layout_alignParentBottom</code>	Nằm sát <u>dưới cùng</u> của RelativeLayout chứa nó.
<code>layout_alignParentLeft</code>	Nằm sát <u>bên trái</u> của RelativeLayout chứa nó.
<code>layout_alignParentRight</code>	Nằm sát <u>bên phải</u> của RelativeLayout chứa nó.
<code>layout_centerInParent</code>	Nằm ở <u>chính giữa</u> của RelativeLayout chứa nó.
<code>layout_centerHorizontal</code>	<u>Canh giữa theo chiều ngang</u> so với RelativeLayout chứa nó.
<code>layout_centerVertical</code>	<u>Canh giữa theo chiều dọc</u> so với RelativeLayout chứa nó.

- Thuộc tính có giá trị là id của View/GroupView hoặc RelativeLayout chứa nó

Thuộc tính	Khi giá trị là id của đối tượng View hoặc ViewGroup khác
layout_alignLeft	View/ViewGroup sẽ nằm ở bên trái của đối tượng đó
layout_alignRight	View/ViewGroup sẽ nằm ở bên phải của đối tượng đó
layout_below	View/ViewGroup sẽ nằm ở bên dưới của đối tượng đó
layout_above	View/ViewGroup sẽ nằm ở bên trên đối tượng đó

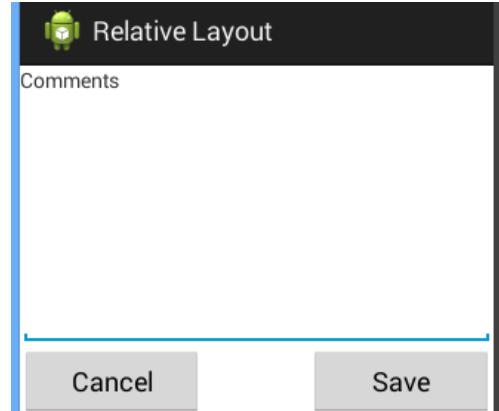
## BÀI THỰC HÀNH app07\_relative

### ☞ Yêu cầu

Vẫn trong project App\_07, tạo mới 1 activity *MainActivity\_Relative*. Activity này sẽ sử dụng file layout **app07\_relative** như hình 2.21.

### ☞ Thực hiện

**B18:** Tạo mới file layout tên **app07\_relative** với nội dung:



Hình 2-21 Minh họa kết quả của *app07\_relative*

```
<?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="170dp"
 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="125dp"
 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="124dp"
 android:layout_height="wrap_content"
 android:text="Cancel"
 android:layout_below="@+id/txtComments"
 android:layout_alignLeft="@+id/txtComments" />
</RelativeLayout>
```

**B19:** Tương tự như trên, sinh viên tự thực hiện các công việc sau:

- Tạo mới file *src/com.example.app\_07/MainActivity\_Relative.java*.
- Bổ sung mã lệnh trong file *MainActivity.java* để khi người dùng nhấn chọn button Relative sẽ cho mở activity vừa có.
- Khai báo activity mới trong file *AndroidManifest.xml*.

### 2.1.4.5. FrameLayout

- *FrameLayout* là ViewGroup đặc biệt có thể sử dụng để hiển thị một View đơn. Tất cả các đối tượng View được đặt trong FrameLayout sẽ luôn ở trên cùng bên trái của FrameLayout. Đây cũng có thể coi là 1 tiện ích như khi cần đặt 1 textView ở góc trái trên của hình ảnh hoặc khi cần tạo bóng (shadow) cho hình ảnh bằng cách dùng 1 số hình ảnh chồng lên nhau, ...
- Để điều chỉnh vị trí của view nằm bất kỳ đâu trong frame thay đổi vị trí mặc định, ta sử dụng bổ sung các thuộc tính trong nhóm layout margin.

## BÀI THỰC HÀNH app07\_frame

### ☞ Yêu cầu

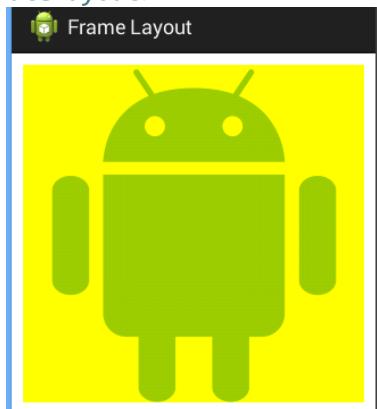
Văn trong project App\_07, tạo mới 1 activity *MainActivity\_Frame*. Activity này sẽ sử dụng file layout *app07\_frame* như hình 2.22.

### ☞ Thực hiện

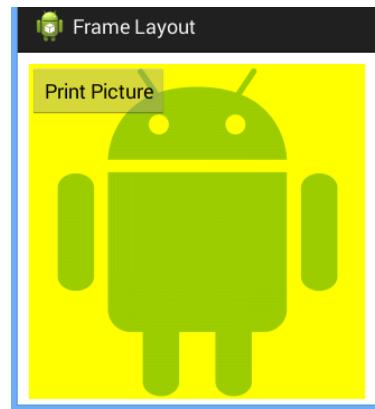
**B20:** Tìm và copy hình ảnh với kích thước khoảng 300X300 (như hình đã dùng trong App\_06) vào các folder tương ứng của project.

**B21:** Tạo mới file layout tên *app07\_frame*. *app07\_frame* gồm một AbsoluteLayout, bên trong có một FrameLayout. Bên trong FrameLayout lại có một ImageView.

```
<?xml version="1.0" encoding="UTF-8"?>
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@+id/widget68"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent" >
 <FrameLayout android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_x="10px"
 android:layout_y="35px">
 <ImageView android:src="@drawable/bground300"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content" />
 </FrameLayout>
</AbsoluteLayout>
```



Giao diện với layout ban đầu



Giao diện sau khi thêm button  
Hình 2-22 Minh họa kết quả của app07\_table

- Thêm một Button bên trong FrameLayout, View mới thêm vào sẽ nằm đè lên View trước đó. Nội dung file xml sau khi bổ sung button:

```
<?xml version="1.0" encoding="UTF-8"?>
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@+id/widget68"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent" >
 <FrameLayout android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_x="10px"
 android:layout_y="35px">
 <ImageView android:src="@drawable/bground300"
 android:layout_width="wrap_content" />
 <Button android:text="Print Picture"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content" />
 </FrameLayout>
</AbsoluteLayout>
```

```

 android:layout_height="wrap_content" />
 <Button android:layout_width="124px"
 android:layout_height="wrap_content"
 android:text="Print Picture" />
</FrameLayout>
</AbsoluteLayout>

```

**B22:** Tương tự như trên, sinh viên tự thực hiện các công việc sau:

- Tạo mới file src/com.example.app\_07/MainActivity\_Frame.java.
- Bổ sung mã lệnh trong file MainActivity.java để khi người dùng nhấn chọn button Frame sẽ cho mở activity vừa có.
- Khai báo activity mới trong file AndroidManifest.xml.

#### 2.1.4.6. ScrollView

*ScrollView* là một trường hợp đặc biệt của *FrameLayout*, nó cho phép người sử dụng cuộn qua một danh sách các đối tượng View hoặc ViewGroup chiếm giữ không gian hiển thị lớn hơn so với màn hình điện thoại hỗ trợ. *ScrollView* chỉ có thể chứa duy nhất một đối tượng View hoặc ViewGroup (thường là *LinearLayout*).

Chú ý: Không sử dụng *ListView* cùng với **ScrollView**. *ListView* được thiết kế để hiển thị một danh sách các đối tượng có liên hệ với nhau. VD Danh sách contact và được tối ưu hóa để phù hợp với những danh sách lớn.

### BÀI THỰC HÀNH app07\_scroll

#### ⦿ Yêu cầu

Vẫn trong project App\_07, tạo mới 1 activity *MainActivity\_Frame*. Activity này sẽ sử dụng file layout **app07\_scroll** như hình 2.22.

#### ⦿ Thực hiện

**B23:** Tìm và copy hình ảnh với kích thước khoảng 300X300 (như hình đã dùng trong App\_06) vào các folder tương ứng của project.

**B24:** Tạo mới layout **app07\_scroll** với *ScrollView* chứa một *LinearLayout*, bên trong có 5 button và 1 edittext.

```

<?xml version="1.0" encoding="utf-8" ?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="fill_parent" >
 <LinearLayout android:layout_width="match_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:hint="Text View"
 android:layout_height="300px" />
 <Button android:id="@+id/button4"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="Button 4" />
 </LinearLayout>
</ScrollView>

```

```

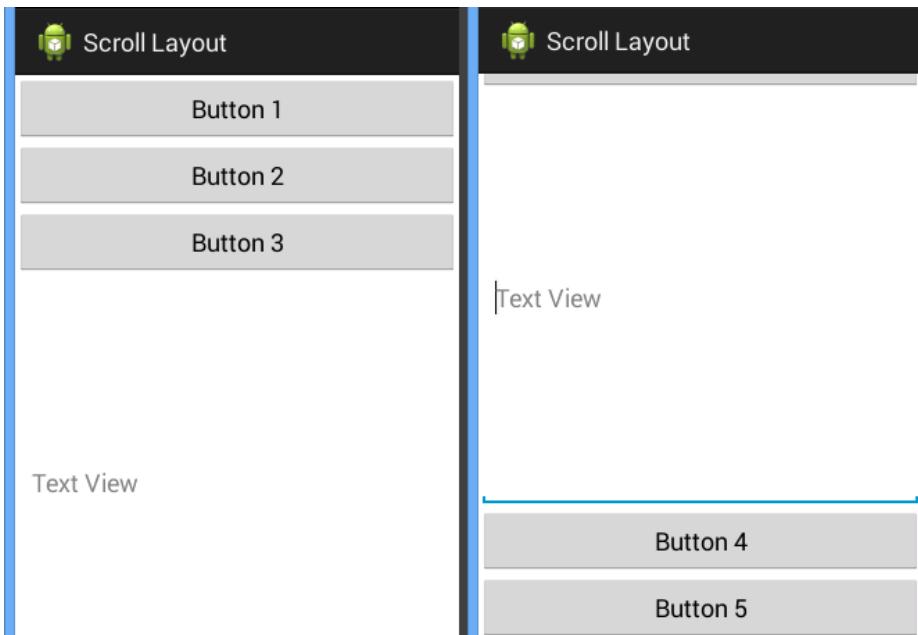
<Button android:id="@+id/button5"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="Button 5" />
 </LinearLayout>
</ScrollView>

```

**B25:** Tương tự như trên, sinh viên tự thực hiện các công việc sau:

- Tạo mới file src/com.example.app\_07/MainActivity\_Scroll.java.
- Bổ sung mã lệnh trong file MainActivity.java để khi người dùng nhấn chọn button ScrollView sẽ cho mở activity vừa có.
- Khai báo activity mới trong file AndroidManifest.xml.

Thực thi đoạn code trên bạn sẽ được kết quả như sau:



Hình 2-23 Minh họa kết quả của app07\_scroll

Giao diện trên cho phép ta dùng ngón tay vuốt lên màn hình theo chiều từ dưới lên để xem các control đang bị khuất phía dưới, sau đó vuốt theo chiều từ trên xuống để xem các control đang bị khuất phía trên. Khi thực hiện trên thiết bị ảo trên máy tính, click và giữ mouse rồi kéo lên (hoặc xuống)

## 2.2. MỘT SỐ CONTROL CƠ BẢN

Để tạo các giao tiếp với người dùng trong Eclipse, bạn chọn các mục trong palette. Palette tổ chức các mục theo từng nhóm như: *Form Widgets*, *Text Fields*, ... Nhóm *Form Widget* bao gồm *TextViews*, *Buttons*, *Spinners*, *ProgressBars*, và *Seekbars*. Nhóm *TextFields* bao gồm *EditText* và *AutoCompleteTextViews*.

Hầu hết các điều khiển trong Eclipse đều cho phép hiệu chỉnh các thuộc tính như kích thước, màu sắc, ...

### 2.2.1. Sử dụng các điều khiển nhập liệu cơ bản

Trong một số ứng dụng cần thu thập thông tin từ người dùng. Hầu hết các thông tin cơ bản được nhập thông qua *edittext*, các *textview* thường để hướng dẫn người dùng về các thông tin liên quan và các *button* để ra lệnh khởi chạy 1 chức năng hoặc công việc nào đó

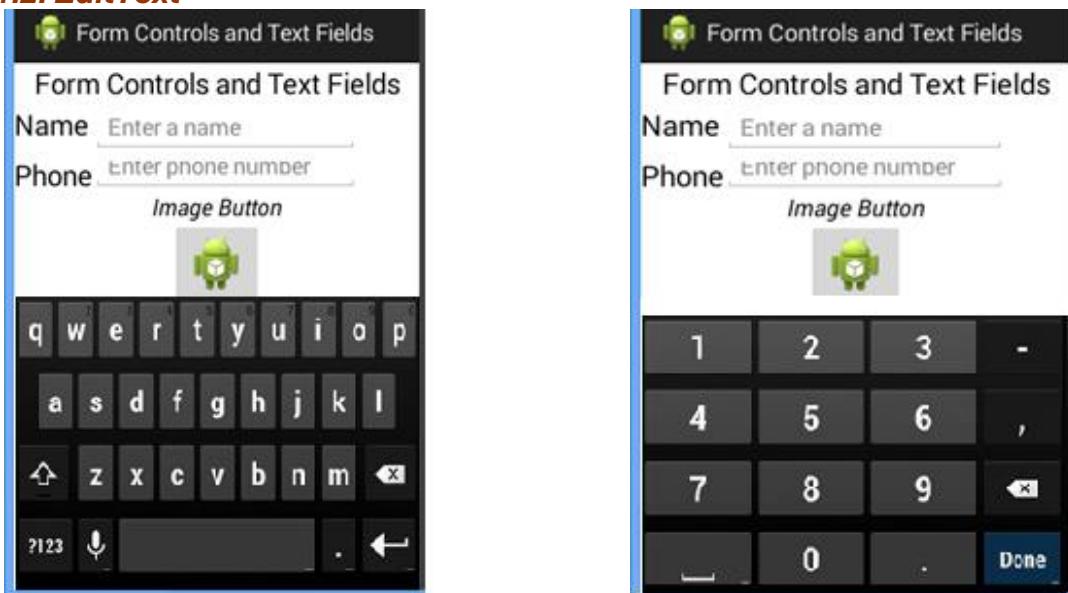
#### 2.2.1.1. TextView

- Công dụng: chỉ cho hiển thị thông tin mà không cho phép người dùng chỉnh sửa trực tiếp.
- Thuộc tính:
  - *android:fontFamily* xác định Font chữ cho *TextView*.

VD: *android:fontFamily = "tahoma"*

- **android:background** xác định màu nền  
VD: android:background = "#ff0000ff"
- **android:textStyle** xác định FontStyle cho TextView.  
VD: android:textStyle = "bold"
- **android:textSize** xác định FontSize cho TextView.  
VD: android:textSize = "25sp"
- **android:textColor** xác định màu chữ cho TextView.  
VD: android:textColor = "@android:color/red-dark"
- **android:text** xác định chuỗi sẽ hiển thị trên TextView.  
VD: android:text = "Nhập số nguyên dương"
- Phương thức:
  - **setText:** hiển thị thông tin lên control TextView  
VD: txt1.setText("Hello world");
  - **getText():** lấy thông tin bên trong control TextView  
VD: String msg=txt1.getText().toString();

### 2.2.1.2. EditText



*EditText view with default inputType property      EditText view with number inputType property*

Hình 2-24 Thuộc tính android:text giúp xác định kiểu bàn phím sẽ xuất hiện

- Công dụng: cho phép chỉnh sửa dữ liệu.
- Thuộc tính:
  - **android:hint**: hiển thị thông tin gợi ý trong vùng nhập dữ liệu khi người dùng chưa nhập bất kỳ dữ liệu nào vào. Khi có dữ liệu trong EditText, phần **hint** sẽ tự động mất đi.
  - **android:textAutoCorrect**: Tự động sửa đúng chính tả.  
VD: nhập “teh” → sẽ tự động sửa thành “the”
  - **android:inputType**: chỉ định dữ liệu cho phép nhập vào TextView. Cho phép kết hợp nhiều giá trị lại với nhau bằng cách dùng toán tử “|”  
VD: android:inputType= “text| textAutoCorrect |textCapWords”
  - **android:text**: giá trị text trong EditText.
- Phương thức:
  - **setText:** hiển thị thông tin lên control TextView  
VD: txtBox.setText("nhập ngày sinh theo dạng dd/mm/yy");
  - **getText():** lấy thông tin bên trong control TextView  
VD: String msg=txtBox.getText().toString();

### 2.2.1.3. Button

2.2.1.3.1. *Công dụng:* gọi thực hiện một sự kiện đã được cài đặt trước.

2.2.1.3.2. *Phân loại:*

- **ImageButton**

- Cần thực hiện import android.widget.ImageButton.
- Được đặt trong nhóm Images and Media của palette.
- Có thể hiển thị hình ảnh có/ không kèm với nội dung văn bản trên button.
- Do class ImageButton được mở rộng từ ImageView nên 1 ImageButton là 1 ImageView nhưng có cách hành động như 1 button. Một ImageButton chỉ chứa được duy nhất 1 hình ảnh.

- **Button**

- Cần thực hiện import android.widget.Button.
- Được đặt trong nhóm Form Widgets của palette.
- Được dùng phổ biến khi chỉ cần hiển thị nội dung văn bản trên button.
- Class Button được mở rộng từ TextView nên Button là một TextView có thể đặt các drawables xung quanh nó.
- Phương thức:
  - `setOnClickListener()`: dùng để viết nội dung chương trình cần xử lý khi button được click.
- Có thể thêm hình ảnh vào button bằng 1 số cách như:
  - Cách 1 Khai báo thuộc tính trong file layout: Gán hình ảnh trong drawable cho 1 trong những thuộc tính: `drawableLeft`, `drawableRight`, `drawableTop`, `drawableBottom`. Ví dụ:
 

```
<Button android:id="@+id/button2"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:drawableLeft="@drawable/ic_launcher"
 android:text="Button" />
```

- Cách 2 sử dụng style sẵn có (hoặc tự tạo):

- (i). Bổ sung hình ảnh cần dùng trong style vào foleder res\drawable tương ứng (nếu tự tạo style riêng)
- (ii). Bổ sung nội dung style cần dùng vào file res\values\styles.xml

```
<resources>
 <!-- Phần đầu file chứa các style sẵn có của project -->
 <!-- Ví dụ về phân thêm mới -->
 <style name="Widget_Button_Inset">
 <item name="android:background">@drawable/button_inset</item>
 </style>
</resources>
```

(iii). Sử dụng trong file layout: ví dụ

```
<Button android:id="@+id/button3"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 style="@style/Widget_Button_Inset" />
```

- Cách 3 Tao background tùy ý: Ví dụ:

- (i). Copy hình ảnh cho mỗi trạng thái của button vào foleder res/drawable tương ứng.
- (ii). Tạo 1 file XML trong folder res\drawable mô tả mỗi trạng thái của button (focus, pressed, ...) sẽ tương ứng với hình ảnh nào? Ví dụ file được tạo có tên với nội dung:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<selector xmlns:android="http://schemas.android.com/apk/res/android">
 <item android:drawable="@drawable/blue_button"
 android:state_pressed="true" />
 <item android:drawable="@drawable/yellow_button"
 android:state_focused="true" />
 <item android:drawable="@drawable/red_button" />
</selector>

```

(iii). Lúc này, Android xem file được tạo cũng là 1 drawable. Vì vậy trong file layout, gán tên file này cho thuộc tính background của button

```

<Button android:id="@+id/button4"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:background="@drawable/button_custom" />

```

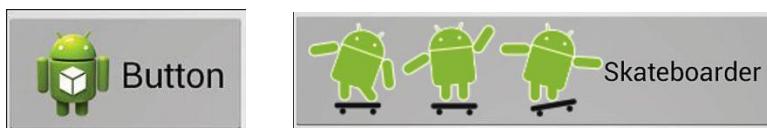
- Cách 4 Thay đổi hình ảnh button khi lập trình: sử dụng phương thức `setCompoundDrawablesWithIntrinsicBounds()` để thay hình ảnh.  
Ví dụ: khi người dùng nhấn trên button sẽ thực hiện thay đổi hình ảnh. Sử dụng 2 hình ảnh có tên lần lượt là `ic_launcher` và `skateboarding_robot120`.

```

final Drawable mDraw1 = getResources().getDrawable(R.drawable.ic_launcher);
final Drawable mDraw2 = getResources().getDrawable(R.drawable.skateboarding_robot120);
final Button swapButton = (Button)findViewById(R.id.button2);
bool flag=true;
swapButton.setOnClickListener(new OnClickListener()
{ public void onClick(View v)
{ if (flag)
 swapButton.setCompoundDrawablesWithIntrinsicBounds(mDraw2, null, null, null);
else
 swapButton.setCompoundDrawablesWithIntrinsicBounds(mDraw1, null, null, null);
}
});

```

Phương thức `setCompoundDrawablesWithIntrinsicBounds()` gồm 4 tham số theo thứ tự: left, top, right, và bottom.



## BÀI THỰC HÀNH App\_08

☛ **Yêu cầu:** tạo 1 ứng dụng gồm 5 activity, trong đó

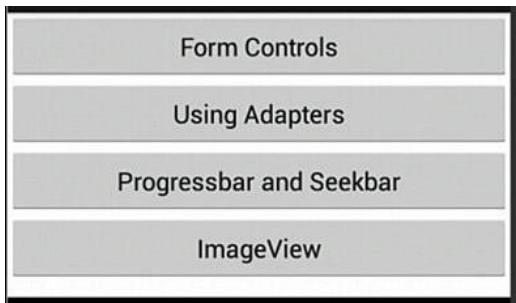
- Activity chính (hình 2-25) chủ yếu làm nhiệm vụ gọi các activity khác thực hiện.

- Activity FormControlActivity để khi người dùng click trên button của chương trình chính, activity này sẽ được mở. layout của FormControlActivity có dạng như hình 2-26, với 1 số yêu cầu:
  - Edittext Name: chỉ cho nhập ký tự.
  - Edittext Phone: chỉ cho nhập ký số.



- Button  khi được click: hình sẽ chuyển đổi qua lại giữa 2 hình ic\_launcher và skateboard; tương tự chữ hiển thị cũng thay đổi qua lại giữa 2 chuỗi "Button" và "Skateboardder".

- Button  : bình thường sẽ có màu đỏ, khi được click sẽ chuyển sang màu xanh, khi nhận focus sẽ chuyển sang màu vàng.



Hình 2-25 Giao diện chính của ứng dụng App\_08

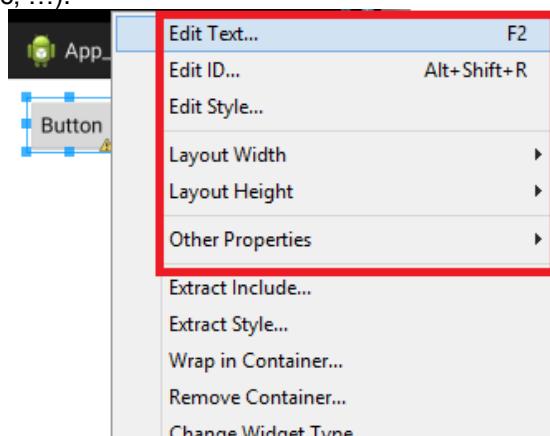


Hình 2-26 Giao diện của activity

FormControlActivity

### Thực hiện

- Để thực hiện bài thực hành này, bạn nên sử dụng LinearLayout với thuộc tính orientation là vertical. Các button sử dụng trong activity chính sẽ sử dụng thuộc tính *layout\_width* với giá trị là *match\_parent*.
- Sau khi đã tạo control trên giao diện, thay vì chuyển sang tab XML để hiệu chỉnh thuộc tính của các view, bạn có thể thực hiện chỉnh sửa trực tiếp các thuộc tính của view bằng click phải lên view, chọn Properties. Trong cửa sổ Properties (xuất hiện bên dưới góc phải của màn hình), chọn thuộc tính cần thay đổi, chọn giá trị gán cho thuộc tính đó (như thay đổi nội dung chuỗi hiển thị, định kích thước theo chiều ngang/dọc, ...).



Hình 2-27 Right click trên control để hiệu chỉnh trực tiếp 1 số thuộc tính

**B1.-** Tạo mới project App\_08.

**B2.-** Chuẩn bị hình ảnh sử dụng cho các button và lưu vào folder res\drawable tương ứng.

- Hình ảnh:

Hình	Tên file	Hình	Tên file
	ic_launcher		skateboarding_robot120
	blue_button		red_button

	yellow_button		
	button_inset		shutdown_button

- Tạo mới file button\_custom.xml trong folder res\drawable tương ứng với nội dung:

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
 <item android:drawable="@drawable/blue_button"
 android:state_pressed="true" />
 <item android:drawable="@drawable/yellow_button"
 android:state_focused="true" />
 <item android:drawable="@drawable/red_button" />
</selector>
```

- Bổ sung vào file res\value\style.xml các style cần dùng

```
<resources>
 <!-- Phần đầu file chứa các style sẵn có của project -->
 <!-- Phân thêm mới -->
 <style name="Widget_Button">
 <item name="android:background">@drawable/shutdown_button</item>
 <item name="android:focusable">true</item>
 <item name="android:clickable">true</item>
 <item name="android:textAppearance">?android:attr/textAppearanceSmallInverse
 </item>
 <item name="android:textColor">@android:color/primary_text_light</item>
 <item name="android:gravity">center_vertical|center_horizontal</item>
 </style>
 <!-- The Widget_Button_Inset has a different background image and some attributes
 are not set. -->
 <style name="Widget_Button_Inset">
 <item name="android:background">@drawable/button_inset</item>
 </style>
</resources></pre>

```

- B3.-** Tạo mới 2 file layout:

- activity\_main.xml:**

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:id="@+id/LinearLayout1"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 tools:context="com.example.app_08.MainActivity" >
 <Button android:id="@+id/button1"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Form Controls" />
 <Button android:id="@+id/button2"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Using Adapters" />
 <Button android:id="@+id/button3"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="ProgressBar And SeekBar" />
 <Button android:id="@+id/button4"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Image View" />
</LinearLayout>
```

▫ **Form\_control\_activity.xml**

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:id="@+id/LinearLayout1"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 tools:context="com.example.app_08.MainActivity" >
 <TextView android:id="@+id/TextView1"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Form Controls and Text Fields"
 android:gravity="center"
 android:textAppearance="?android:attr/textAppearanceLarge" />
 <TableLayout android:layout_width="match_parent"
 android:layout_height="wrap_content">
 <TableRow >
 <TextView android:id="@+id/textView2"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Name"
 android:textAppearance="?android:attr/textAppearanceLarge"/>
 <EditText android:id="@+id/editText1"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:hint="Enter a name"
 android:inputType="text"
 android:ems="10" >
 <requestFocus />
 </EditText>
 </TableRow>
 <TableRow >
 <TextView android:id="@+id/TextView3"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Phone"
 android:textAppearance="?android:attr/textAppearanceLarge"/>
 <EditText android:id="@+id/editText2"
 android:layout_width="113dp"
 android:layout_height="match_parent"
 android:ems="10"
 android:hint="Enter phone number"
 android:inputType="number" />
 </TableRow>
 </TableLayout>
 <TextView android:id="@+id/TextView4"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Image Button"
 android:gravity="center"
 android:textStyle="italic"
 android:textAppearance="?android:attr/textAppearanceMedium"/>
 <ImageButton android:id="@+id/imageButton1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_gravity="center"
 android:src="@drawable/ic_launcher" />
 <TextView android:id="@+id/TextView5"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Standard Button"
 android:gravity="center"
```

```

 android:textStyle="italic"
 android:textAppearance="?android:attr/textAppearanceMedium"/>
 <LinearLayout android:orientation="horizontal"
 android:layout_width="match_parent"
 android:layout_height="wrap_content">
 <Button android:id="@+id/button1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_weight="1"
 android:text="Button" />
 <Button android:id="@+id/button4"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_weight="1"
 android:background="@drawable/button_custom" />
 </LinearLayout>
 <Button android:id="@+id/button2"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:drawableLeft="@drawable/ic_launcher"
 android:text="Button" />
 <Button android:id="@+id/button3"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 style="@style/Widget_Button_Inset" />
</LinearLayout>

```

**B4.-** Tạo mới file Form\_Control\_Activity.java và bổ sung các lệnh cần thiết

- Sửa tên file layout cần nạp thành form\_control\_activity
- Bổ sung các import sau vào đầu file

```

import android.graphics.drawable.Drawable;
import android.widget.Button;
import android.view.View;
import android.view.View.OnClickListener;

```

- Đổi với button cần đổi hình và chữ:

```

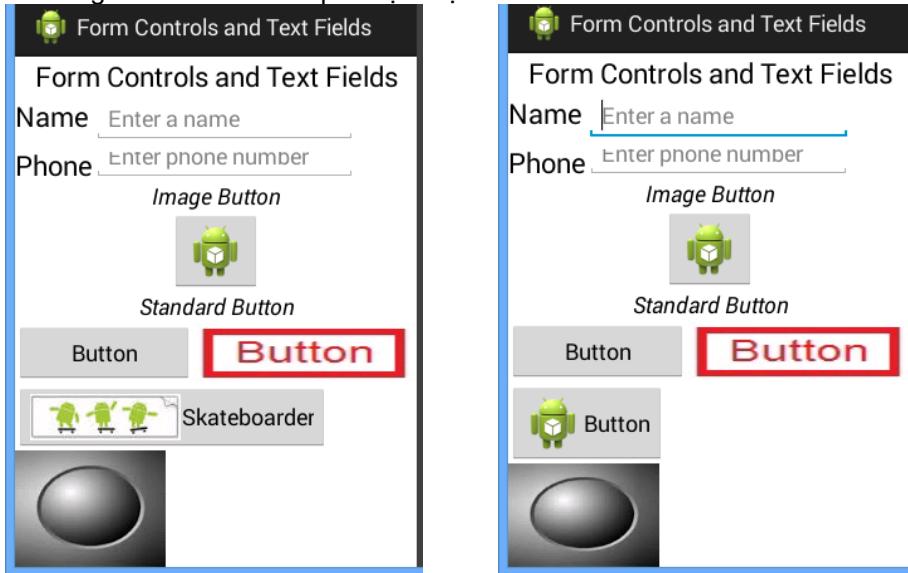
final Drawable mDraw1 = getResources().getDrawable(R.drawable.ic_launcher);
final Drawable mDraw2 = getResources().getDrawable(R.drawable.skateboarding_robot120);
final Button swapButton = (Button)findViewById(R.id.button2);
swapButton.setOnClickListener(new OnClickListener()
{
 public void onClick(View v)
 {
 if (swapButton.getText().equals("Button"))
 { swapButton.setText("Skateboarder");
 swapButton.setCompoundDrawablesWithIntrinsicBounds(mDraw2, null, null, null);
 }
 else
 { swapButton.setText("Button");
 swapButton.setCompoundDrawablesWithIntrinsicBounds(mDraw1, null, null, null);
 }
 }
});

```

**B5.-** Bổ sung lệnh để mở activity form\_control\_activity

- Bổ sung lệnh cho button trong MainActivity.java để mở activity form\_control\_activity.
- Bổ sung khai báo activity trong file AndroidManifest.xml.

**B6.-** Chạy chương trình để xem kết quả thực hiện



Hình 2-28 Kết quả khi thực hiện ứng dụng FormControlActivity

## 2.2.2. Sử dụng các điều khiển với Adapters

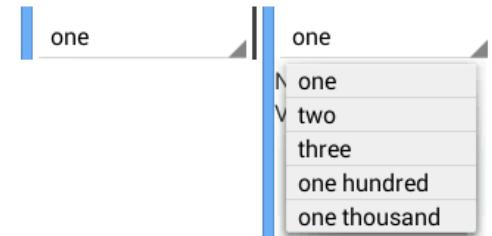
Để hiển thị nội dung, một số control chỉ đơn giản là dùng 1 chuỗi đơn (ví dụ button hiển thị chữ OK hay dấu trừ (-) thể hiện cho phép trừ). Một số control khác lại yêu cầu hiển thị nhiều hơn 1 mẩu dữ liệu (như Spinner hay combobox hay listbox), khi đó bạn cần dùng đến adapter để đưa dữ liệu vào control.

### 2.2.2.1. Adapters

Án Adapter(android.widget.Adapter) giúp kết buộc dữ liệu vào 1 view. Ví dụ Spinner là 1 control cung cấp đến người dùng 1 danh sách các mục chọn. Danh sách này là dữ liệu được yêu cầu cung cấp cho view khi ứng dụng chạy.

### 2.2.2.2. Spinner

Tương tự như control ComboBox, spinner cho phép người dùng lựa chọn 1 mục trong một danh sách cho trước.



Hình 2-29 Hình bên trái là tên mục đã được chọn.  
Hình bên phải liệt kê các mục người dùng được phép chọn

#### 2.2.2.2.1. Thiết lập Spinner

- Các bước xây dựng Spinner:

- Khai báo mảng chuỗi cung cấp dữ liệu cho adapter

VD: String[] values = {"one", "two", "three", "one hundred", "one thousand"};

- Tạo biến mSpinner kết buộc vào control spinner1 trên giao diện

VD: mSpinner = (Spinner) findViewById(R.id.spinner1);

- Tạo adapter

VD: ArrayAdapter<String> spinnerAdapter = new ArrayAdapter<String>(this, android.R.layout.simple\_spinner\_item, values);

- Kết buộc adapter vào Spinner  
VD: `mSpinner.setAdapter(spinnerAdapter);`

### 2.2.2.2. Lấy dữ liệu từ Spinner

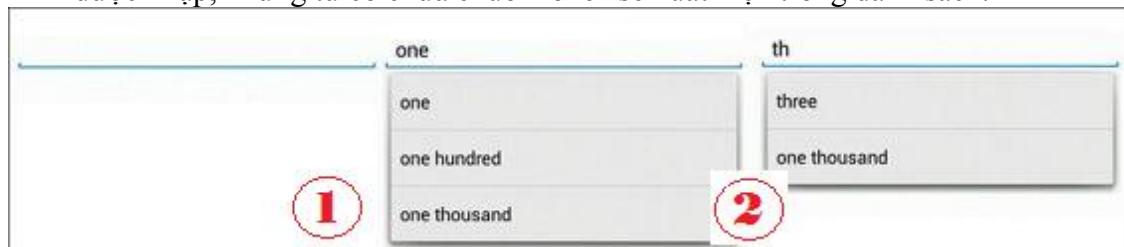
- Để bắt sự kiện khi người dùng chọn trên spinner, sử dụng phương thức `OnItemSelectedListener()`.
- Để đọc dữ liệu đang được chọn trong spinner: sử dụng phương thức `getSelectedItem()` hoặc `getSelectedItemPosition()`. VD:

```
mSpinner.setOnItemSelectedListener(new OnItemSelectedListener()
{
 @Override
 public void onItemSelected(AdapterView<?> arg0, View arg1, int pos, long arg3)
 {
 myString = (String) mSpinner.getSelectedItem();
 }
 @Override
 public void onNothingSelected(AdapterView<?> arg0)
 {
 myString = "";
 }
});
```

### 2.2.2.3. Sử dụng AutoCompleteTextView

#### 2.2.2.3.1. Giới thiệu

- Đối với các thiết bị di động, việc hỗ trợ nhập dữ liệu nhanh cho người sử dụng là điều rất cần thiết.
- `AutoCompleteTextView` là một subclass của `EditText`. Ngoài những chức năng tương tự như `EditText` nó còn có khả năng đưa ra một danh sách các gợi ý tương tự như phần text mà người dùng nhập vào. Người dùng có thể chọn 1 mục trong danh sách đề nghị hoặc tạo ra 1 mục mới.
- Hình bên trái cho thấy tình trạng `AutoCompleteTextView` khi chưa nhập liệu, khi từ “one” được nhập, những từ có chứa chuỗi “one” sẽ xuất hiện trong danh sách.



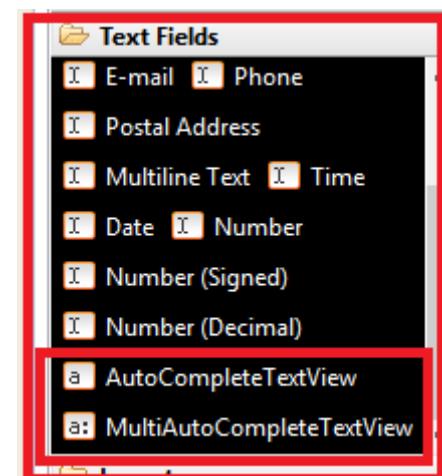
Hình 2-30 Minh họa `AutoCompleteTextView` trước (1) và trong khi nhập dữ liệu

#### 2.2.2.3.2. Phân loại

- `AutoCompleteTextView`: chỉ có thể sử dụng được một lần tự động điền dữ liệu
- `MultiAutoCompleteTextView`: các dữ liệu được tự động điền nhiều lần và cách nhau bằng dấu phẩy.

#### 2.2.2.3.3. Dữ liệu

- Tương tự như nhu trong minh họa Spinner trong phần trên, danh sách cung cấp cho `AutoCompleteTextView` cũng là mảng dữ liệu kiểu chuỗi.



Hình 2-31 `AutoCompleteTextView` và `MultiAutoCompleteTextView`

```
ArrayAdapter<String> textAdapter = new ArrayAdapter<String>
 (this, android.R.layout.simple_spinner_dropdown_item, values);
singleComplete = (AutoCompleteTextView)findViewById(R.id.autoCompleteTextView1);
singleComplete.setAdapter(textAdapter);
```

– Lưu ý:

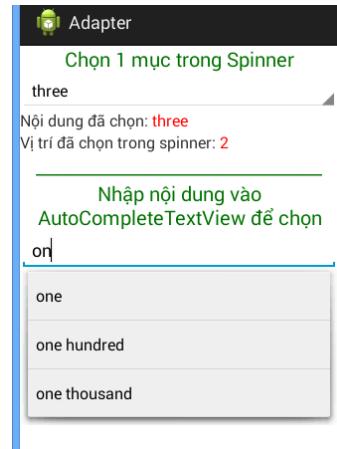
- *values*, *singleComplete* và *multiComplete* sử dụng trong ví dụ đã được khai báo trước đó.
- Riêng với *MultiAutoCompleteTextView*, sau khi kết buộc dữ liệu xong, cần bổ sung lệnh sau:

```
multiComplete.setTokenizer(new
 Multi.AutoCompleteTextView.CommaTokenizer());
```

#### 2.2.2.3.4. Thuộc tính

- *completionThreshold*: thiết lập số ký tự bắt đầu lọc trong AutoComplete. Ví dụ:

```
android:completionThreshold= "1"
```



Hình 2-32 AutoCompleteTextView khi đang nhập dữ liệu

#### 2.2.2.3.5. Phương thức

- *getText()*: lấy dữ liệu trong AutoCompleteTextView
- *onTextChanged*
- *afterTextChanged*
- *beforeTextChanged*

```
//Khi chọn trong AutoCompleteTextView các hàm này sẽ được gọi
public void onTextChanged(CharSequence arg0, int arg1, int arg2, int arg3)
{
 singleComplete.setText(singleComplete.getText());
}

public void afterTextChanged(Editable arg0)
{
}

public void beforeTextChanged(CharSequence arg0, int arg1, int arg2, int arg3)
{
}
```

## BÀI THỰC HÀNH App\_08 (tiếp theo)

☞ **Yêu cầu:** Tạo giao diện có dạng như hình 2.33, trong đó sử dụng 3 control chính là Spinner, AutoCompleteTextView và MultiAutoCompleteTextView. Thêm activity Adapter để khi người dùng click trên button “Using Adapters” của chương trình chính, activity này sẽ được mở.

☞ **Thực hiện**

- Nội dung file adapter.xml

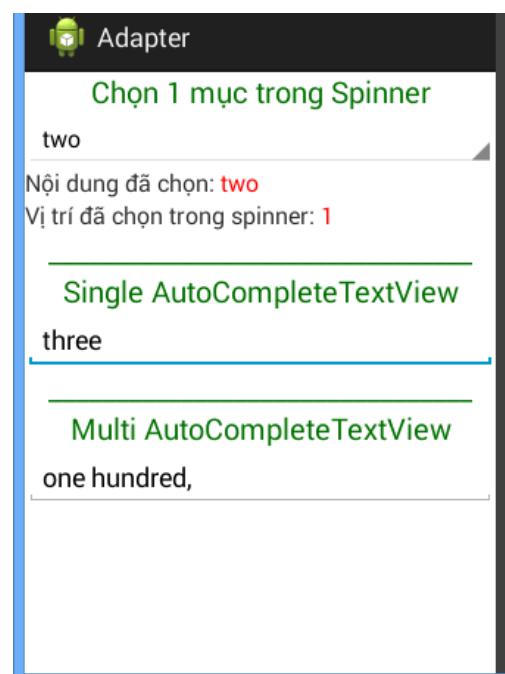
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:id="@+id/LinearLayout1"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 tools:context="com.example.app_08.MainActivity" >
 <TextView
 android:id="@+id/textview1"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="Chọn 1 mục trong Spinner"
 android:gravity="center"
 android:textColor="#007200"
 android:textSize="20sp" />
```

```

<Spinner
 android:id="@+id/spinner1"
 android:layout_width="match_parent"
 android:layout_height="wrap_content" />
<LinearLayout
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:orientation="horizontal" >
 <TextView
 android:id="@+id/textview2"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Nội dung đã chọn: "
 android:textSize="16sp" />
 <TextView
 android:id="@+id/textview3"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text=""
 android:textColor="#FF0000"
 android:textSize="16sp" />
</LinearLayout>

<LinearLayout
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:orientation="horizontal" >
 <TextView
 android:id="@+id/textview4"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Vị trí đã chọn trong spinner: "
 android:textSize="16sp" />
 <TextView
 android:id="@+id/textview5"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text=""
 android:textColor="#FF0000"
 android:textSize="16sp" />
</LinearLayout>
<TextView
 android:id="@+id/textView6"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="_____"
 android:textColor="#007200"
 android:gravity="center"
 android:textSize="20sp" />
<TextView
 android:id="@+id/textView7"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:gravity="center"
 android:text="Single AutoCompleteTextView"
 android:textColor="#007200"
 android:textSize="20sp" />
<AutoCompleteTextView
 android:id="@+id/autoCompleteTextView1"
 android:layout_width="match_parent"
 android:layout_height="wrap_content" />

```



Hình 2-33 Giao diện của ứng dụng Adapter

```

 android:ems="10"
 android:completionThreshold= "1"
 android:text="" >
 <requestFocus />
 </AutoCompleteTextView>
<TextView
 android:id="@+id/textView10"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="_____"
 android:textColor="#007200"
 android:gravity="center"
 android:textSize="20sp" />
<TextView
 android:id="@+id/textView11"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:gravity="center"
 android:text="Multi AutoCompleteTextView"
 android:textColor="#007200"
 android:textSize="20sp" />
<MultiAutoCompleteTextView
 android:id="@+id/multiCompleteTextView1"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:ems="10"
 android:completionThreshold= "1"
 android:text="" >
 <requestFocus />
</MultiAutoCompleteTextView>
</LinearLayout>

```

– Nội dung file Adapter.java

```

import android.support.v7.app.ActionBarActivity;
import android.text.Editable;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.MultiAutoCompleteTextView;
import android.widget.Spinner;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.TextView;

public class adapter extends ActionBarActivity {
 String[] values = {"one", "two", "three", "one hundred", "one thousand" };
 TextView tvText, tvPos, tvComplete;
 Spinner mSpinner;
 AutoCompleteTextView singleComplete;
 MultiAutoCompleteTextView multiComplete;
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.adapter);
 tvText = (TextView) findViewById(R.id.textview3);
 tvPos = (TextView) findViewById(R.id.textview5);
 mSpinner = (Spinner)findViewById(R.id.spinner1);
 }
}

```

```

// tao arrayAdapter để chứa
ArrayAdapter<String> spinnerAdapter = new ArrayAdapter<String>
(this, android.R.layout.simple_spinner_item, values);
mSpinner.setAdapter(spinnerAdapter);
mSpinner.setOnItemSelectedListener(new OnItemSelectedListener() {
 @Override
 public void onItemSelected(AdapterView<?> arg0, View arg1,
 int pos, long arg3)
 {
 tvText.setText(mSpinner.getSelectedItem().toString());
 tvPos.setText(String.valueOf(pos));
 }
 @Override
 public void onNothingSelected(AdapterView<?> arg0) {
 // TODO Auto-generated method stub
 tvText.setText("");
 tvPos.setText("");
 }
});
ArrayAdapter<String> textAdapter = new ArrayAdapter<String>
(this, android.R.layout.simple_spinner_dropdown_item, values);
singleComplete = (AutoCompleteTextView)findViewById(R.id.autoCompleteTextView1);
singleComplete.setAdapter(textAdapter);

//Sử dụng chung dữ liệu với SingleComplete
multiComplete =
(MultiAutoCompleteTextView)findViewById(R.id.multiCompleteTextView1);
multiComplete.setAdapter(textAdapter);
//Đối với MultiAutoCompleteTextView bắt buộc phải gọi dòng lệnh này
multiComplete.setTokenizer(new MultiAutoCompleteTextView.CommaTokenizer());
}
// các phương thức sẵn có trong class ActivityB sẽ nằm ngay sau dòng ghi chú này
// . .
})

```

### 2.2.3. ProgressBars and SeekBars

- ProgressBar (*android.widget.ProgressBar*) and SeekBar (*android.widget.SeekBar*) giúp thông báo tình trạng hiện thời của công việc đang thực hiện.
- *ProgressBar*: cho thấy công việc đang thực hiện đã hoàn thành được bao nhiêu, ví dụ như bạn muốn sử dụng *ProgressBar* để mô tả quá trình đang download file.
- *SeekBar*: đơn giản cũng chỉ là một *ProgressBar* nằm ngang, nhưng người dùng có thể xác định vị trí của *SeekBar*. Ví dụ như sử dụng *SeekBar* trong ứng dụng xem phim, lúc này người dùng có thể xác định/thay đổi vị trí bắt đầu xem phim.
- *AsyncTask*(*android.os.AsyncTask*) được dùng cho những xử lý đa tiến trình và thực hiện dưới nền (background)



Hình 2-34 Minh họa dạng thể hiện của *ProgressBar* và *SeekBar*

#### 2.2.3.1. *AsyncTask*

- *AsyncTask* (*android.os.AsyncTask*) được dùng cho những xử lý đa tiến trình và thực hiện dưới nền (background). Khi phương thức *execute()* thực hiện, class *AsyncTask* thực hiện tiêu trình dưới nền mà không cần ngăn chặn tiêu trình User Interface.

- Các bước sử dụng AsyncTask: cần tạo một class kế thừa từ *AsyncTask*, sau đó từ *MainActivity* ta gọi hàm *execute()* của tiến trình.
- Khai báo class XXX kế thừa từ *AsyncTask*. Khai báo này nằm bên trong class *MainActivity* của chương trình cần sử dụng *AsyncTask*.

```
public class MainActivity extends Activity {
 .
 .
 .
 private class XXX extends AsyncTask<void, integer, integer>
 {
 .
 .
 .
 }
 .
 .
}
```

- Khai báo đối tượng Obj thuộc class XXX  
XXX Obj = new XXX();
- Gọi phương thức execute của Obj và cho thực thi ShowProgressTask bằng các lệnh sau:  
Obj.execute();
- Trong **AsyncTask<Params, Progress, Result>** có 3 đối số là các Generic Type:
  - **Params:** được truyền vào khi gọi thực thi tiến trình và nó sẽ được truyền vào *doInBackground*
  - **Progress:** dùng để update giao diện lúc tiến trình thực thi, biến này sẽ được truyền vào hàm *onProgressUpdate*.
  - **Result:** dùng để lưu trữ kết quả trả về sau khi tiến trình thực hiện xong.

Ba tham số trên là không bắt buộc. để bỏ qua tham số nào, ta dùng từ khóa *void* tại vị trí đó.

Ví dụ khai báo class MyTask không sử dụng cả 3 tham số

```
class MyTask extends AsyncTask<void, void, void>
{
 .
 .
 .
}
```

- Thông thường trong 1 AsyncTask sẽ chứa 4 hàm, đó là :
  - ***onPreExecute()*:** Tự động được gọi đầu tiên khi tiến trình được kích hoạt. Các lệnh truy cập đến giao diện thường đặt trong phương thức này.
  - ***doInBackground()*:** thực hiện các tác vụ chính bằng cách gọi thực thi các tiểu trình khác như gọi hàm *onProgressUpdate* thực hiện cập nhật giao diện (gọi lệnh *publishProgress*) mà không thể cập nhật giao diện trực tiếp trong hàm *doInBackground()*.
  - ***onProgressUpdate ()*:** Dùng để cập nhật giao diện lúc runtime
  - ***onPostExecute()*:** Sau khi tiến trình kết thúc thì hàm này sẽ tự động xảy ra. Các lệnh truy cập đến giao diện hoặc lấy kết quả trả về sau khi thực hiện tiến trình thường đặt trong phương thức này.

Trong 4 hàm trên thì hàm *doInBackground()* bắt buộc phải tồn tại, còn các hàm khác có thể bỏ qua.

Trong ví dụ sau sử dụng *AsyncTask* có tên là *MyAsyncTask*. Trong đó:

- Phương thức *onPreExecute()*, thiết lập cho *ProgressBar* hiển thị, tạo cảm giác cho người công việc đã và đang được thực hiện
- Phương thức *onPostExecute()* thiết lập lại cho *ProgressBar* ẩn đi (INVISIBLE) tạo cảm giác cho người công việc đã hoàn tất.
- Phương thức *doInBackground()* thực hiện tăng giá trị của biến nguyên rồi cung cấp giá trị của biến này cho *publishProgress()* để cập nhật giá trị cho *ProgressBar*. Phương thức *Sleep()* giúp quá trình thực hiện chậm lại để tạo cảm giác cho người dùng, vì vậy chỉ dùng khi cần minh họa cho *ProgressBar*.
- Phương thức *onProgressUpdate()* chấp nhận giá trị và cập nhật lên *ProgressBar* và/hoặc *SeekBar* tương ứng.

Mã lệnh cho class *MyAsyncTask*:

```

private class MyAsyncTask extends AsyncTask<void, integer, integer>
{
 @Override
 protected void onPreExecute ()
 {
 mProgressBar.setVisibility(View.VISIBLE);
 }
 @Override
 protected Integer doInBackground(void... params)
 {
 for (int i=0; i<=100; i++)
 {
 try
 {
 Thread.sleep(100);
 publishProgress(i);
 }
 catch (InterruptedException e)
 {
 return -1;
 }
 }
 return 100;
 }
 @Override
 protected void onProgressUpdate(Integer... progress)
 {
 int progress = progress[0];
 mHorizontalProgressBar.setProgress(progress);
 mSeekBar.setProgress(progress);
 }
 @Override
 protected void onPostExecute (Integer result)
 {
 mProgressBar.setVisibility(View.INVISIBLE);
 }
}

```

### 2.2.3.2. ProgressBar

- Thuộc tính:
  - **android:progress** xác định giá trị ban đầu cho *ProgressBar*.
  - **android:max** dùng để thay đổi giá trị lớn nhất của *ProgressBar* (mặc định là 100).
  - **style** xác định *ProgressBar* được trình bày theo dạng ngang (mặc định) hay đứng. Ví dụ: `style="?android:attr/progressBarStyleHorizontal"`
- Phương thức `setVisibility(View)`: thiết lập trạng thái ẩn/hiện của *ProgressBar*. Ví dụ: `mProgressBar.setVisibility(View.INVISIBLE);`

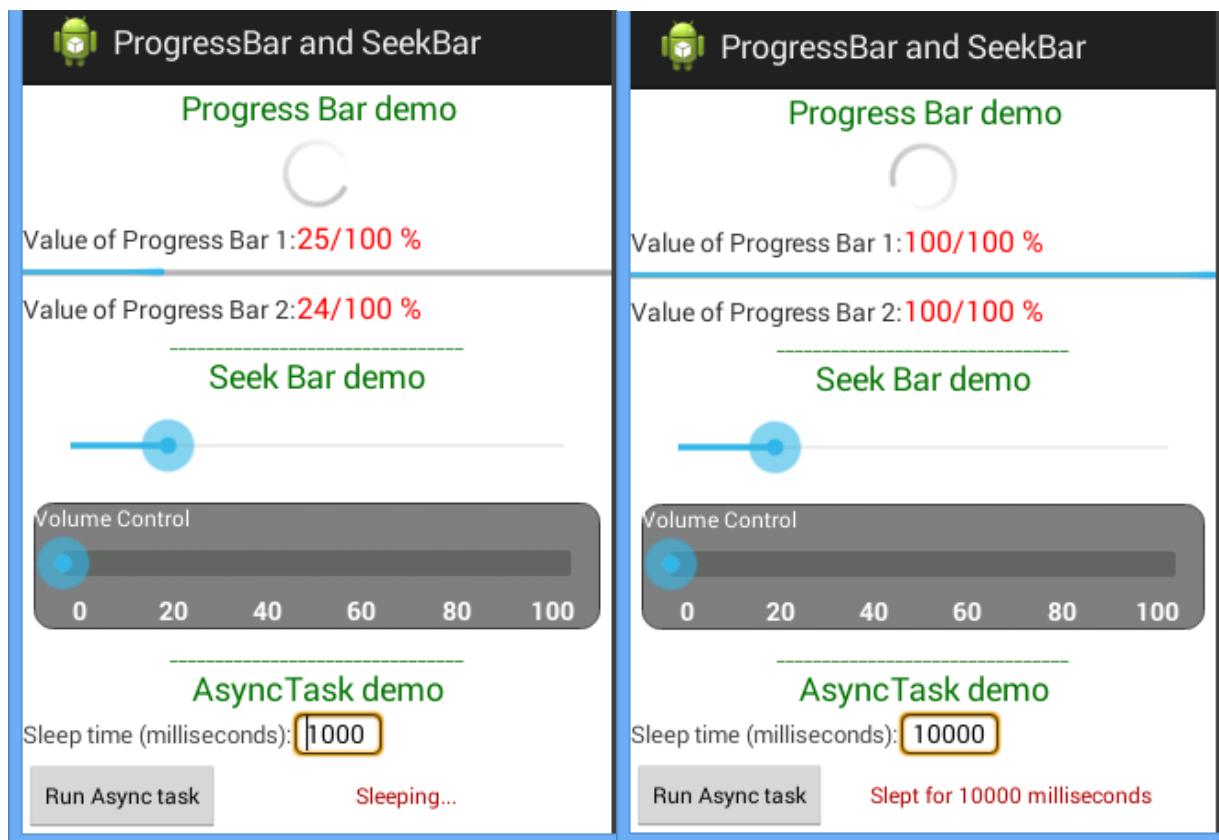
### 2.2.3.3. SeekBar

- Có thể thiết lập 1 listener để lắng nghe và điều chỉnh sự thay đổi trên *SeekBar*
- Thuộc tính: cũng sử dụng 2 thuộc tính **android:progress** và **android:max** như *ProgressBar*

## BÀI THỰC HÀNH App\_08 (tiếp theo)

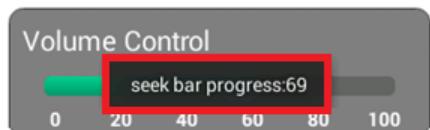
 **Yêu cầu:** Tạo activity mới để khi nhấn trên button thứ 3 (Progress and SeekBar) của MainActivity sẽ xuất hiện activity này. Giao diện của Activity mới có dạng như hình sau, trong đó sử dụng 3 loại control chính là:

- *ProgressBar* (2 cái)
  - *ProgressBar* đầu tiên sử dụng dạng mặc định (vòng tròn xoay -  ).
  - *ProgressBar* thứ 2 sử dụng style nằm ngang.
  - 2 dòng “Value of ...” thể hiện giá trị hiện thời của từng *ProgressBar*
- *AsyncTask*: Khi click trên button “Run Asynctask” sẽ xuất hiện chuỗi “Sleeping ...”. Tùy thuộc thời gian có trong EditText (ví dụ 10000), khi kết thúc, chuỗi “Sleeping ...” sẽ chuyển thành chuỗi “Slept for 10000 milliseconds”.



Hình 2-35 Minh họa hoạt động của AsyncTask trong (trái) và sau (phải) khi thực hiện hoàn tất tác vụ.

- SeekBar (2 cái):
  - SeekBar đầu tiên sử dụng dạng mặc định
  - SeekBar thứ 2 sử dụng drawable riêng.
  - Khi người dùng tự điều chỉnh giá trị trên 1 trong 2 SeekBar sẽ xuất hiện thông báo về giá trị hiện thời của SeekBar tương ứng.



Hình 2-36 Thông báo giá trị hiện thời của SeekBar

## ☛ Thực hiện

### – B1: Chuẩn bị giao diện cho SeekBar thứ 2

- Tạo mới file res\drawable-mdpi\myseekbar.xml, với nội dung:
 

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android" >
 <item android:id="@+id/background">
 <shape android:shape="rectangle" >
 <corners android:radius="2dp" />
 <gradient
 android:angle="270"
 android:endColor="@color/Light_gray_header_color"
 android:startColor="@color/Light_gray_header_color" />
 </shape>
 </item>
 <item android:id="@+id/secondaryProgress">
 <clip>
 <shape android:shape="rectangle" >
 <corners android:radius="2dp" />
 <gradient
 android:angle="270"
 android:endColor="#00996a"
 android:startColor="#00d190" />
 </shape>
 </clip>
 </item>
</layer-list>
```

```

</item>
<item android:id="@+id/progress">
 <clip>
 <shape android:shape="rectangle" >
 <corners android:radius="2dp" />
 <gradient
 android:angle="270"
 android:endColor="#00996a"
 android:startColor="#00d190" />
 </shape>
 </clip>
</item>
</layer-list>

```

- Tạo mới file res\drawable-mdpi\background\_view\_rounded\_single.xml để làm nền choSeekBar thứ 2, với nội dung:

```

<?xml version="1.0" encoding="UTF-8" ?>
<inset android:insetLeft="1.0px"
 android:insetRight="1.0px"
 android:insetTop="0.0px"
 android:insetBottom="1.0px"
 xmlns:android="http://schemas.android.com/apk/res/android">
 <selector>
 <item android:state_pressed="true">
 <shape>
 <gradient android:startColor="@color/rounded_container_bg"
 android:endColor="@color/rounded_container_bg"
 android:angle="270.0" />
 <corners android:radius="11.0dip" />
 </shape>
 </item>
 <item>
 <shape>
 <stroke android:width="1.0px"
 android:color="@color/rounded_container_border" />
 <gradient android:startColor="@color/rounded_container_bg"
 android:endColor="@color/rounded_container_bg"
 android:angle="270.0" />
 <corners android:radius="10.0dip" />
 </shape>
 </item>
 </selector>
</inset>

```

#### **B2: Tạo file định nghĩa các màu sẽ sử dụng trong giao diện**

- Tạo mới file res\values\colors.xml, với nội dung:

```

<?xml version="1.0" encoding="utf-8" ?>
<resources>
 <color name="default_screen_bg">#20324a</color>
 <color name="rounded_container_bg">#80000000</color>
 <color name="rounded_container_border">#3b3f44</color>
 <color name="light_gray_header_color">#646663</color>
</resources>

```

#### **B3: Tạo giao diện:** Tạo mới file res\layout\progressbar\_seekbar.xml, với nội dung:

```

<LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 tools:context="com.example.app_08.MainActivity" >
 <TextView
 android:id="@+id/textview1"
 android:layout_width="fill_parent"

```

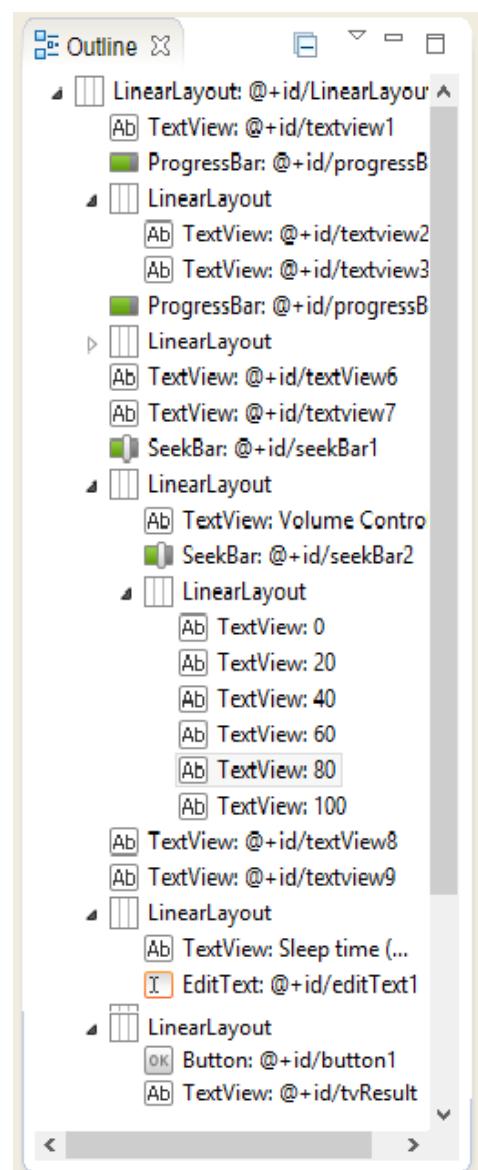
```
 android:layout_height="wrap_content"
 android:text="Progress Bar demo"
 android:gravity="center"
 android:textColor="#007200"
 android:textSize="18sp" />
<ProgressBar android:id="@+id/progressBar1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_gravity="center" />
<LinearLayout android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:orientation="horizontal" >
 <TextView android:id="@+id/textview2"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Value of Progress Bar 1:" />
 <TextView android:id="@+id/textview3"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="0"
 android:textColor="#FF0000"
 android:textSize="16sp" />
</LinearLayout>
<ProgressBar android:id="@+id/progressBar2"
 style="?android:attr/progressBarStyleHorizontal"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:layout_gravity="center" />
<LinearLayout android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:orientation="horizontal" >
 <TextView android:id="@+id/textview4"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Value of Progress Bar 2:"/>
 <TextView android:id="@+id/textview5"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="0"
 android:textColor="#FF0000"
 android:textSize="16sp" />
</LinearLayout>
<TextView android:id="@+id/textView6"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="_____"
 android:textColor="#007200"
 android:gravity="center"
 android:textSize="10sp" />
<TextView android:id="@+id/textview7"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="Seek Bar demo"
 android:gravity="center"
 android:textColor="#007200"
 android:textSize="18sp" />
<SeekBar android:id="@+id/seekBar1"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:layout_margin="10dp"
 android:progress="20"
 android:secondaryProgress="20" />
```

```
<LinearLayout android:layout_width="fill_parent"
 android:layout_height="70dp"
 android:layout_margin="5dp"
 android:background="@drawable/background_view_rounded_single"
 android:orientation="vertical" >
 <TextView android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Volume Control"
 android:textColor="#ffffffff"
 android:textSize="12sp" />
 <SeekBar android:id="@+id/seekBar2"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:progress="0"
 android:max="100"
 android:progressDrawable="@drawable/myseekbar"
 android:secondaryProgress="0" />
<LinearLayout android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:orientation="horizontal" >
 <TextView android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_weight="2"
 android:gravity="center_horizontal"
 android:text="0"
 android:textColor="@android:color/white"
 android:textStyle="bold" />
 <TextView android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_weight="2"
 android:gravity="center_horizontal"
 android:text="20"
 android:textColor="@android:color/white"
 android:textStyle="bold" />
 <TextView android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_weight="2"
 android:gravity="center_horizontal"
 android:text="40"
 android:textColor="@android:color/white"
 android:textStyle="bold" />
 <TextView android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_weight="2"
 android:gravity="center_horizontal"
 android:text="60"
 android:textColor="@android:color/white"
 android:textStyle="bold" />
 <TextView android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_weight="2"
 android:gravity="center_horizontal"
 android:text="80"
 android:textColor="@android:color/white"
 android:textStyle="bold" />
 <TextView android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_weight="2"
 android:gravity="center_horizontal"
 android:text="100"
 android:textColor="@android:color/white"
 android:textStyle="bold" />
```

```

 </LinearLayout>
 </LinearLayout>
<TextView android:id="@+id/textView8"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="_____"
 android:textColor="#007200"
 android:gravity="center"
 android:textSize="10sp" />
<TextView android:id="@+id/textview9"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="AsyncTask demo"
 android:gravity="center"
 android:textColor="#007200"
 android:textSize="18sp" />
<LinearLayout android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:orientation="horizontal" >
<TextView android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:textSize="6pt"
 android:textColor="#444444"
 android:gravity="center"
 android:text="Sleep time
(milliseconds):"/>
<EditText android:id="@+id/editText1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:background=
 "@android:drawable/editbox_background"
 android:inputType="text"
 android:textSize="14sp"
 android:text="1000"/>
</LinearLayout>
<LinearLayout android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:orientation="horizontal" >
<Button android:id="@+id/button1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:gravity="center"
 android:layout_gravity="center"
 android:textSize="12sp"
 android:text="Run Async task" />
<TextView android:id="@+id/tvResult"
 android:layout_width="200dp"
 android:layout_height="wrap_content"
 android:textSize="12sp"
 android:gravity="center"
 android:layout_gravity="center"
 android:textColor="#AA0000"
 android:text="" />
</LinearLayout>
</LinearLayout>

```



Hình 2-37 Cửa sổ Outline sau khi thiết kế hoàn tất

Khi tạo xong, cửa sổ Outline có dạng như hình 2-37

- B4: Tạo file chương trình: Tạo mới file src\com.example.app\_08\ProgressBar\_SeekBar.java, với nội dung:

```

package com.example.app_08;
import android.support.v7.app.ActionBarActivity;
import android.text.Editable;
import android.os.Bundle;
import android.os.Handler;
import android.view.Menu;
import android.view.MenuItem;

import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.SeekBar.OnSeekBarChangeListener;
import android.widget.TextView;
import android.widget.ProgressBar;
import android.widget.SeekBar;
import android.widget.Toast;

public class ProgressBar_SeekBar extends ActionBarActivity {

 private ProgressBar progressBar1, progressBar2;
 private int progressStatus1 = 0, progressStatus2 = 0;
 private SeekBar seekBar1, seekBar2;
 private TextView textView1, textView2;
 private EditText etTime;
 private Handler handler = new Handler();
 private Button button;
 private TextView finalResult;

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

 // Xử lý cho ProgressBar 1
 progressBar1 = (ProgressBar) findViewById(R.id.progressBar1);
 textView1 = (TextView) findViewById(R.id.textview3);
 // Start long running operation in a background thread
 new Thread(new Runnable()
 {
 public void run()
 {
 while (progressStatus1 < 100)
 {
 progressStatus1 += 1;
 // Update progressbar1 and display the current value in the text view
 handler.post(new Runnable()
 {
 public void run()
 {
 progressBar1.setProgress(progressStatus1);
 textView1.setText(progressStatus1+"/"+progressBar1.getMax()+"%");
 }
 });
 }
 }
 });
 try
 {
 Thread.sleep(200);
 }
 catch (InterruptedException e)
 }
}

```

```

 {
 e.printStackTrace();
 }
 }
}).start();
// Xử lý cho ProgressBar 2
progressBar2 = (ProgressBar) findViewById(R.id.progressBar2);
textView2 = (TextView) findViewById(R.id.textview5);
// Start long running operation in a background thread
new Thread(new Runnable()
{
 public void run()
 {
 while (progressStatus2 < 100)
 {
 progressStatus2 += 1;
 // Update progressbar2 and display the current value in the text view
 handler.post(new Runnable()
 {
 public void run()
 {
 progressBar2.setProgress(progressStatus2);
 textView2.setText(progressStatus2+"/"+progressBar2.getMax()+" %");
 }
 });
 }
 try
 {
 Thread.sleep(200);
 }
 catch (InterruptedException e)
 {
 e.printStackTrace();
 }
 }
}).start();
// Xử lý cho SeekBar 1
seekBar1 = (SeekBar) findViewById(R.id.seekBar1);
seekBar1.setOnSeekBarChangeListener(new OnSeekBarChangeListener()
{
 int progressChanged = 0;
 public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser)
 {
 progressChanged = progress;
 }
 public void onStartTrackingTouch(SeekBar seekBar)
 {
 }
 public void onStopTrackingTouch(SeekBar seekBar)
 {
 Toast.makeText(ProgressBar_SeekBar.this,"SeekBar1 progress:" +
 progressChanged,Toast.LENGTH_SHORT).show();
 }
});
// Xử lý cho SeekBar 2
seekBar2 = (SeekBar) findViewById(R.id.seekBar2);
seekBar2.setOnSeekBarChangeListener(new OnSeekBarChangeListener()
{
 int progressChanged2 = 0;
 public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser)
 {

```

```

 progressChanged2 = progress;
 }
 public void onStartTrackingTouch(SeekBar seekBar)
 {
 }
 public void onStopTrackingTouch(SeekBar seekBar)
 {
 Toast.makeText(ProgressBar_SeekBar.this,"SeekBar2 progress:" +
 progressChanged2,Toast.LENGTH_SHORT).show();
 }
});

// Xử lý cho AsyncTask
etTime = (EditText) findViewById(R.id.editText1);
button = (Button) findViewById(R.id.button1);
finalResult = (TextView) findViewById(R.id.tvResult);
button.setOnClickListener(new View.OnClickListener()
{
 @Override
 public void onClick(View v)
 {
 AsyncTaskRunner runner = new AsyncTaskRunner();
 String sleepTime = etTime.getText().toString();
 runner.execute(sleepTime);
 }
});
@Override
public boolean onCreateOptionsMenu(Menu menu) {
 // Inflate the menu; this adds items to the action bar if it is present.
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
 // Handle action bar item clicks here. The action bar will
 // automatically handle clicks on the Home/Up button, so long
 // as you specify a parent activity in AndroidManifest.xml.
 int id = item.getItemId();
 if (id == R.id.action_settings) {
 return true;
 }
 return super.onOptionsItemSelected(item);
}

/* Create Private class which runs the long operation */
private class AsyncTaskRunner extends AsyncTask<String, String, String>
{
 private String resp;
 @Override
 protected String doInBackground(String... params)
 {
 publishProgress("Sleeping..."); // Calls onProgressUpdate()
 try
 {
 // Do your long operations here and return the result
 int time = Integer.parseInt(params[0]);
 // Sleeping for given time period
 Thread.sleep(time);
 resp = "Slept for " + time + " milliseconds";
 }
 catch (InterruptedException e)
 }
}

```

```

 {
 e.printStackTrace();
 resp = e.getMessage();
 }
 catch (Exception e)
 {
 e.printStackTrace();
 resp = e.getMessage();
 }
 return resp;
}
@Override
protected void onPostExecute(String result)
{
 finalResult.setText(result);
}
@Override
protected void onPreExecute()
{
}
@Override
protected void onProgressUpdate(String... text)
{
 finalResult.setText(text[0]);
}
}
}

```

- B5: Khai báo activity trong file Mainifest.xml
- B6: Viết lệnh để gọi activity vừa tạo từ MainActivity.java

## 2.2.4. ImageViews

### 2.2.4.1. Giới thiệu

- ImageView(*android.widget.ImageView*) được dùng để hiển thị hình ảnh, tuy nhiên cũng có những khó khăn nhất định vì độ phân giải màn hình khác nhau và dpi trong các thiết bị Android cũng khác nhau.
- Trong các phiên bản sau này, ImageView được tích hợp trong android.view.View, do đó không cần khai báo *android.widget.ImageView* và biến ImageView lúc này được khai báo lại như sau:
  - Khai báo theo cách cũ:                      ImageView        imageView;
  - Khai báo theo cách mới:                      View             imageView;

### 2.2.4.2. Thuộc tính

- android:size
- android:gravity
- android:src thiết lập ảnh sẽ sử dụng (trong drawable).

### 2.2.4.3. Phương thức

- **setImageResource()**: Hiển thị ảnh. Ví dụ:  
ivLogo.setImageResource(R.drawable.logo1);

### 2.2.4.4. Truy xuất đến các thuộc tính

- Thông qua LayoutParams: chú ý chọn LayoutParams tương ứng với loại Layout chứa View (trong ví dụ này là RelativeLayout)
 

```
LayoutParams ivRocketParams = (LayoutParams) ivRocket.getLayoutParams();
ivRocketParams.width = 200;
ivRocketParams.height = 200;
ivRocket.setLayoutParams(ivRocketParams);
```

### 2.2.4.5. Xử lý sự kiện

- Sử dụng Listener tương ứng. Ví dụ sau xử lý sự kiện chạm vào ảnh sẽ đổi sang ảnh logo2 và di chuyển vị trí lên trên 100px.

```
ivLogo.setOnTouchListener(new OnTouchListener()
{ @Override
public boolean onTouch(View arg0, MotionEvent arg1)
{ LayoutParams ivLogoParams = (LayoutParams) ivLogo.getLayoutParams();
ivLogoParams.bottomMargin += 100;
ivLogo.setImageResource(R.drawable.logo2);
ivLogo.setLayoutParams(ivLogoParams);
return true;
}
})
```

## BÀI THỰC HÀNH App\_08 (tiếp theo)

☞ **Yêu cầu:** Tạo activity imageView để khi người dùng click trên button thứ 4 (Image View) của chương trình chính, activity này sẽ được mở. Layout của imageView có dạng như hình 2-38, với 1 số yêu cầu:

- Các control sử dụng 2 textView (1 là tiêu đề giao diện, 1 sẽ hiển thị tên của hình ảnh) và 1 imageview.
- Giao diện:
  - Ban đầu, giao diện có dạng như hình bên trái của nhóm hình 2-38.
  - Khi click trên imageview sẽ thực hiện chuyển đổi hình với thời gian chuyển đổi giữa các hình là 1 giây và tổng thời gian thực hiện là 10 giây.
- Khi hình ảnh thay đổi thì chuỗi hiển thị tên hình ảnh cũng xuất hiện đồng bộ theo



Hình 2-38 Minh họa giao diện của ứng dụng ImageView

☞ **Thực hiện**

**B1.-** Copy hình ảnh vào res\drawable tương ứng

**B2.-** Tạo mới layout trong res\layout\imageview.xml, với nội dung:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical" >
 <TextView
 android:id="@+id/tv"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Click vào ảnh để bắt đầu xem"
 android:textAppearance="?android:attr/textAppearanceMedium" />
 <ImageView
 android:id="@+id/img"
 android:layout_width="match_parent"
 android:layout_height="0dp" />
```

```

 android:layout_weight="0.96"/>
<TextView
 android:id="@+id/tv2"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="Image Name"
 android:textAppearance="?android:attr/textAppearanceMedium" />
</LinearLayout>

```

B3.- Tạo mới file src\com.example.app\_08\ImageView.java, với nội dung:

```

package com.example.app_08;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.os.CountDownTimer;
import android.view.View;
import android.widget.TextView;
public class ImageView extends ActionBarActivity {
 int i=0;
 int[] R_image={R.drawable.ty1,R.drawable.suu,R.drawable.dan,R.drawable.meo,
 R.drawable.thin,R.drawable.ty5,R.drawable.ngo,R.drawable.mui,
 R.drawable.than,R.drawable.dau,R.drawable.tuat,R.drawable.hoi};
 String[] image_name={"Tý", "Sửu", "Dần", "Mẹo", "Thìn", "Ty", "Ngọ", "Mùi", "Thân",
 "Dậu", "Tuất", "Hợi"};
 View iv;
 TextView tv1, tv2;
 View v;
 int flag=0;
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.image);
 tv1=(TextView)findViewById(R.id.tv);
 tv2=(TextView)findViewById(R.id.tv2);
 iv=(View) findViewById(R.id.img);
 iv.setBackgroundResource(R_image[i]);
 tv2.setText(image_name[i]);
 iv.setOnClickListener(new View.OnClickListener()
 {
 public void onClick(final View v)
 {
 flag=1;
 long delay = 5000;
 long period =1000;
 new CountDownTimer(delay, period)
 {
 @Override
 public void onTick(long millisUntilFinished)
 {
 if (flag==1)
 {
 i=(i+1)%12;
 v.setBackgroundResource(R_image[i]);
 tv2.setText(image_name[i]);
 tv1.setText("Đang thực hiện tự động thay đổi ảnh");
 }
 }
 @Override
 public void onFinish()
 {
 // TODO Auto-generated method stub
 tv1.setText("Click vào hình để xem tiếp");
 }
 }.start();
 }
 });
 }
}

```

```
// các phương thức sẵn có trong class Activity sẽ nằm ngay sau dòng ghi chú này
// . .
}
```

B4.- Khai báo activity trong Mainifest.xml.

## 2.3. ActionBar & Menu Navigation

Android supports two kinds of menus. First, there is the menu you get when you press the physical Menu button. Second, there is a context menu that pops up when you press and hold your finger on the screen (or press and hold the trackball or the D-pad center button).

Từ đầu tài liệu đến giờ, để di chuyển từ activity này sang activity khác, ta đều dùng button. Một cách khác để thực hiện việc này là sử dụng ActionBar (android.app.ActionBar). ActionBar được giới thiệu từ phiên bản Android 3.0, giúp việc di chuyển giữa các activity được thực hiện dễ dàng hơn.

### 2.3.1. Tìm hiểu về Options Menu

- Khi giới thiệu chiếc Samsung Galaxy Nexus cùng với hệ điều hành Android 4.0, Google đã loại bỏ nút Menu ra khỏi cụm phím điều hướng chính của hệ thống và thay thế bằng "Action Bar" (hỗ trợ từ Android level 11 - version 3.0 trở đi), chỉ còn lại nút Quay về (back), nút trở về màn hình chính (Home) và nút liệt kê các ứng dụng đã chạy gần đây (Recents Apps).
- ActionBar là nơi mà người dùng có thể khởi chạy các hoạt động tùy thuộc vào ứng dụng và để người dùng có thể điều khiển hoàn toàn phần mềm bằng màn hình cảm ứng.
- ActionBar là thanh tiêu đề của trang trong ứng dụng hiện tại. Mà mỗi trang như vậy lại có một tiêu đề cũng như có các nút chức năng hoàn toàn khác nhau. Chức năng mỗi nút là do người lập trình quyết định.
- ActionBar có thể được đặt ở đâu đó trên màn hình, chẳng hạn như cạnh trên, cạnh dưới, cạnh trái/phải,... Để tạo sự quen thuộc trong thao tác cho người sử dụng, Google đề nghị Action Bar nên được đặt nằm ở cạnh trên cùng của màn hình, với các thành phần cơ bản như Delete, Share, Settings, Star,... và nút Action Overflow (khi xoay ngang máy, Action Bar thường nằm ở cạnh trái).
- Tất cả những hoạt động quan trọng đều nằm trong ActionBars. Những tính năng nào không thể hiện hết thì sẽ nằm trong dấu ba chấm dạng dọc (⋮) ở cuối Action Bar, gọi là Action Overflow.

#### 2.3.1.1. Hiển thị Options Menu

Khi tạo mới project Android, theo mặc định sẽ tạo ra 1 activity có tên là MainActivity, 1 file có tên là main.xml được tạo lập trong folder res\menu. Trong mã lệnh của file *MainActivity.java* sẽ tự động phát sinh đoạn mã sau:

```
@Override
public boolean onCreateOptionsMenu(Menu menu)
{
 // Inflate the menu; this adds items to the action bar if it is present.
 getMenuInflater().inflate(R.menu.activity_main, menu);
 return true;
}
```

Phương thức inflate trong *onCreateOptionsMenu()* làm tăng thêm các mục menu được định nghĩa trong file tài nguyên menu của *activity\_main.xml*. Đối với những thiết bị không hỗ trợ action bar, những mục này sẽ xuất hiện như các mục trong menu.

Nội dung file *res\menu\main.xml* được tự động hình thành có nội dung tương tự như sau:

```
<menu 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"
 tools:context="com.example.app_08.MainActivity" >
 <item
 android:id="@+id/action_settings"
```

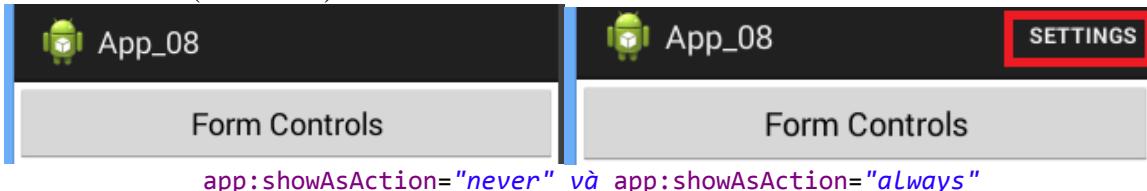
```

 android:orderInCategory="100"
 android:title="@string/action_settings"
 app:showAsAction="never"/>

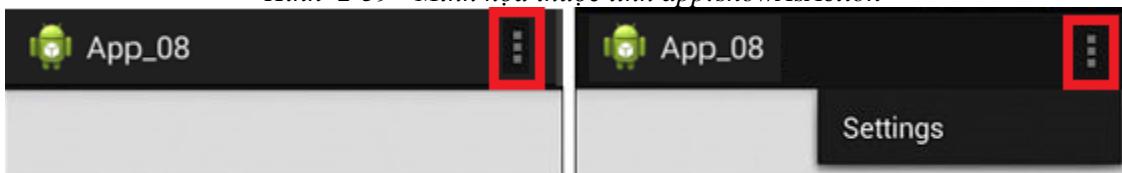
```

Trong đó `android:orderInCategory` xác định vị trí của item trên ActionBar.

Khi action bar không đủ chỗ chứa menu, sẽ xuất hiện ký hiệu 3 dấu chấm dọc biểu thị trạng thái tràn trên menu (hình 2-40).



Hình 2-39 Minh họa thuộc tính `app:showAsAction`



Hình 2-40 Minh họa ký hiệu 3 chấm dọc

#### 2.3.1.1. Thuộc tính `showAsAction`

Do thuộc tính `app:showAsAction` được đặt là `"never"` nên ta không thấy menu xuất hiện. Chuỗi hiển thị trên menu được mặc định là `Setting`.

Khi 1 mục xuất hiện trên action bar, sẽ được gọi là action item và có thể click trực tiếp trên action bar. Khi danh sách các action item trống, chúng sẽ không xuất hiện như tùy chọn Setting trong hình.

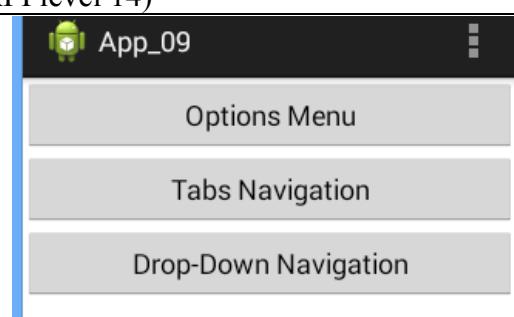
Có thể chọn cho thuộc tính `app:showAsAction` 1 trong những giá trị sau: `never`, `ifRoom`, `always`, `withText`, `collapseActionView`.

Giá trị	Ý nghĩa
<code>never</code>	Không bao giờ hiển thị trên ActionBar
<code>ifRoom</code>	Hiển thị trên ActionBar nếu còn đủ chỗ trống. Thông thường, cách sử dụng này được khuyên nên tránh sử dụng vì về lý thuyết các ActionItem có che lấp giao diện người dùng.
<code>always</code>	Luôn hiển thị trên ActionBar
<code>withText</code>	Hiển thị tên của menu được chỉ định trong thuộc tính <code>android:Title</code> . Có thể sử dụng cùng với các giá trị khác thông qua ký hiệu gạch đứng (more)
<code>collapseActionView</code>	actionView có thể thu lại được (API level 14)

### BÀI THỰC HÀNH App\_09

#### ☛ Yêu cầu:

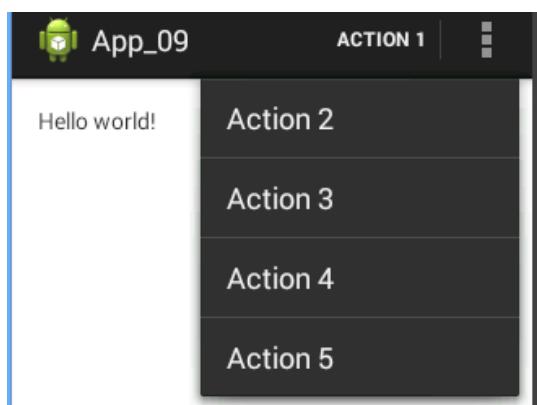
- Tạo mới project App\_09, có giao diện như hình 2-41. Khi click button nào sẽ mở activity tương ứng với activity có tính năng đó.
- Tạo mới thêm 1 activity `Options_Item` để khi click chọn trên button “Options Menu” sẽ mở activity này. Trong activity `Options_Item` thực hiện thêm 5 action item vào action bar, trong đó:



Hình 2-41 Minh họa kết quả cần đạt được

- Chỉnh sửa file res\menu\main.xml để thêm 2 ActionItem vào ActionBar với thuộc tính showAsAction của cả 2 ActionItem là ifRoom.
- Sử dụng mã lệnh để thêm trực tiếp 3 ActionItem lúc Runtime (không cần dùng XML Resource).

Kết quả thực hiện có dạng như hình 2-42:



Hình 2-42 Minh họa kết quả cần đạt được

### ☞ Thực hiện

B1.- Bổ sung đoạn mã XML để thêm 2 item là Action1 và Action2 vào file res\menu\main.xml chưa menu chính của chương trình. Kết quả file res\menu\main.xml có dạng:

```
<menu 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"
 tools:context="com.example.app_09.MainActivity" >
 <item android:id="@+id/action_settings"
 android:title="@string/action_settings"
 android:showAsAction="ifRoom" />
 <item android:id="@+id/action1"
 android:showAsAction="ifRoom"
 app:showAsAction="always"
 android:title="Action 1"/>
 <item android:id="@+id/action2"
 android:showAsAction="ifRoom"
 app:showAsAction="always"
 android:title="Action 2"/>
</menu>
```

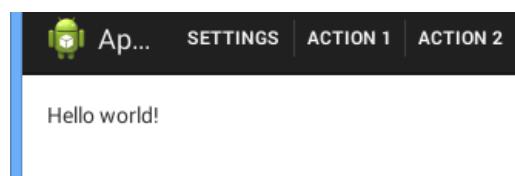
- Nếu sau khi bổ sung đoạn mã trên, Android xuất hiện cảnh báo có dạng tương tự như: “Attribute “showAsAction” is only used in API level 11 and higher (current min is 8)”

Attribute "showAsAction" is only used in API level 11 and higher (current min is 8)

mở file *AndroidManifest.xml*, tìm và chỉnh sửa thuộc tính *android:minSdkVersion* từ số hiện tại (ví dụ là 8) thành số 11 (hoặc cao hơn). Kết quả sau khi thực hiện:

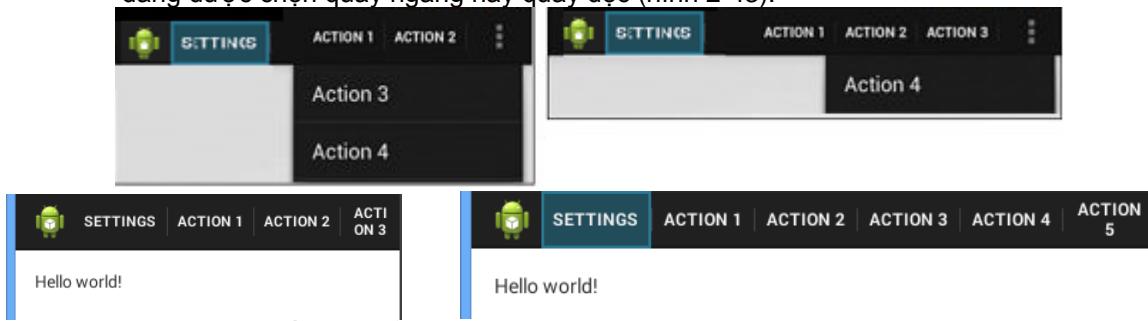
```
<uses-sdk
 android:minSdkVersion="11"
 android:targetSdkVersion="21" />
```

- Chạy ứng dụng để xem kết quả:



Hình 2-43 Minh họa ActionBar

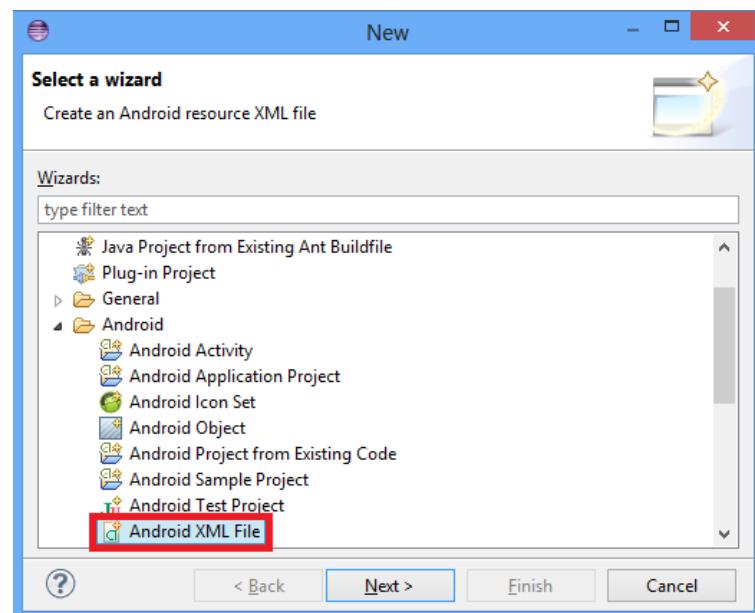
- Nếu bạn thêm nhiều ActionItem nữa vào ActionBar cũng với thuộc tính showAsAction là ifRoom, các item được thêm sẽ xuất hiện trên phần menu tràn. Kết quả hiển thị tùy thuộc vào thiết bị đang được chọn quay ngang hay quay dọc (hình 2-43).



Hình 2-44 Minh họa kết quả hiển thị trên tablet tùy thuộc quay đứng (trái) hay quay ngang (phải)

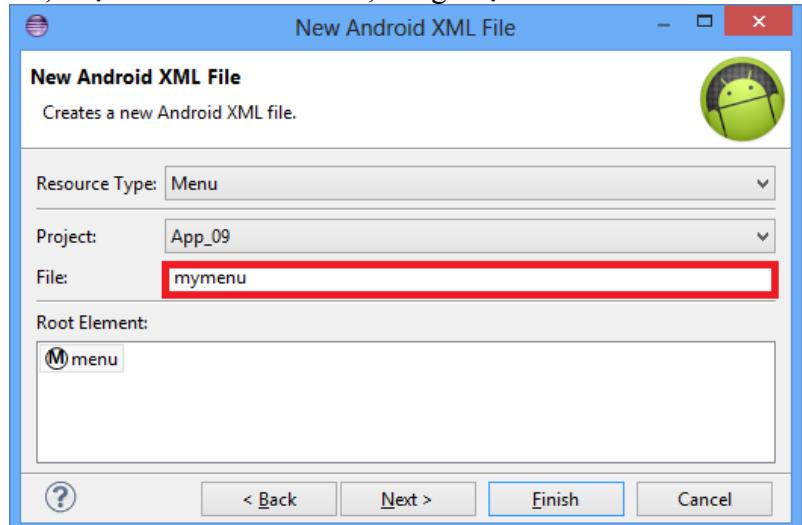
- Có thể thực hiện tương tự như đã giới thiệu ở B1, nhưng thay vì sử dụng file chứa menu chính của chương trình, ta tạo mới file menu cho riêng mình theo các bước:

- Right click vào thư mục `res\menu`, chọn `New\Others` ... để mở hộp thoại New.



Hình 2-45 Chọn Android XML File

- Trong hộp thoại New, chọn *Android XML File*, xong chọn Next.



Hình 2-46 Đặt tên file chứa menu

- Trong hộp thoại New Android XML File vừa xuất hiện, chọn *Resource type* là *Menu*, đặt tên cho file chứa menu là “*mymenu*”. Chọn Finish.
- Quan sát lại thư mục *menu* của ứng dụng sẽ xuất hiện file *mymenu.xml*
- Bổ sung các lệnh như đã giới thiệu ở trên vào file này. Khi hoàn tất, nội dung file *mymenu.xml* có dạng:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
 <item android:id="@+id/action1"
 android:showAsAction="ifRoom"
 app:showAsAction="always"
 android:title="Action 1"/>
 <item android:id="@+id/action2"
 android:showAsAction="ifRoom"
 app:showAsAction="always"
 android:title="Action 2"/>
</menu>
```

- (vi). Chỉ định menu cần dùng: mở file MainActivity.java, tìm đến hàm và chỉ định lại menu cần sử dụng cho activity. Sau khi hoàn tất phương thức này có dạng:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
 /* Inflate the menu; this adds items to the action bar if it is
 * present */
 getMenuInflater().inflate(R.menu.mymenu, menu);
 return super.onCreateOptionsMenu(menu);
}
```

### B2.- Tạo 3 MenuItem bằng Coding (Runtime):

- Sử dụng phương thức *add* của class Menu. Mỗi lần thực hiện phương thức sẽ thêm được 1 MenuItem.
- Phương thức *add* gồm 4 đối số theo thứ tự: nhóm, id của MenuItem, thứ tự xuất hiện của MenuItem, tiêu đề cho MenuItem.

Bổ sung mã lệnh vào hàm onCreateOptionsMenu trong file MainActivity.java

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
 // Inflate the menu; this adds items to the action bar if it is present.
 getMenuInflater().inflate(R.menu.main, menu);

 int itemId=13;
 menu.add(0, itemId, 0, "Action 3");
 itemId=14;
 menu.add(0, itemId, 1, "Action 4");
 itemId=15;
 menu.add(0, itemId, 2, "Action 5");

 return super.onCreateOptionsMenu(menu);
}
```

#### 2.3.1.1.2. Menu Items đối với những thiết bị không hỗ trợ nền đối với ActionBar

Đối với những điện thoại được cài những version cũ, có thể menu sẽ không xuất hiện như mong đợi. Khi đó, bạn cần nhấn button menu (MENU) để chúng xuất hiện.

#### 2.3.1.1.3. Thêm icon cho menu item

Bạn có thể thêm icon vào bất kỳ menu item nào. Icon được hỗ trợ trong cả ActionBar lẫn menu kiểu cũ.

Để thêm icon vào menu item, thêm dòng lệnh sau vào trong khai báo của menu item:

```
android:icon=
 "@+drawable/icon_name"
```

Trong trường hợp này, bạn nên lưu icon thành 2 bản, 1 bản lưu trong *res\drawable-mdpi* và 1 bản lưu trong *res\drawable-mdpi-v11*. Mục đích của icon trong folder sẽ cung cấp drawable cho những phiên bản hỗ trợ Android API từ 11 trở đi.



Hình 2-47 Minh họa App\_09 chạy trên Android 2.3.3

## BÀI THỰC HÀNH App\_09 (tiếp theo)

☞ **Yêu cầu:** Thêm icon cho 2 MenuItem Action1 và Action2

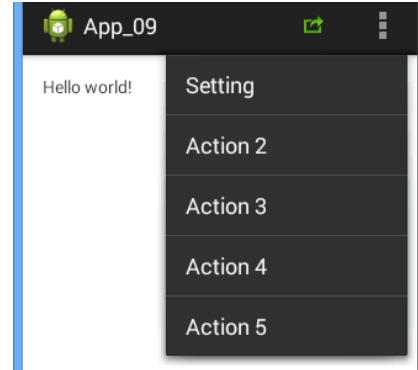
☞ **Thực hiện**

**B3.-** Copy 2 file icon (đã định dùng cho 2 MenuItem) vào folder drawable tương ứng

**B4.-** Bổ sung thuộc tính android:icon cho từng item. Nội dung file res\menu\main.xml lúc này có dạng:

```
<menu 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"
 tools:context="com.example.app_09.MainActivity" >
 <item android:id="@+id/action_settings"
 android:title="Setting"
 android:showAsAction="ifRoom"/>
 <item android:id="@+id/action1"
 app:showAsAction="ifRoom"
 android:icon="@+drawable/send"
 android:title="Action 1"/>
 <item android:id="@+id/action2"
 app:showAsAction="ifRoom"
 android:icon="@+drawable/sun"
 android:title="Action 2"/>
</menu>
```

**B5.-** Chạy chương trình để xem kết quả, thông thường các MenuItem chứa trong “vùng tràn” sẽ không có icon được hiển thị.



Hình 2-48 Minh họa App\_09 chạy trên Android 2.3.3

### 2.3.1.2. Nhận phản hồi từ MenuItem

Sử dụng phương thức `onOptionsItemSelected()` để nhận phản hồi khi 1 MenuItem được click.

## BÀI THỰC HÀNH App\_09 (tiếp theo)

☞ **Yêu cầu:** Khi người dùng click chọn vào MenuItem nào sẽ sử dụng Toast để hiển thị tên nội dung vừa chọn

☞ **Thực hiện**

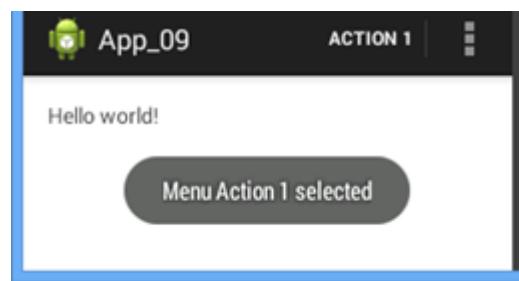
**B6.-** Bổ sung đoạn mã vào phương thức `onOptionsItemSelected()` để thực hiện yêu cầu. Nội dung phương thức `onOptionsItemSelected()` như sau:

```
public boolean onOptionsItemSelected(MenuItem item)
{
 int id = item.getItemId();
 switch (id) {
 case R.id.action_settings:
 Toast.makeText(this, "Menu Setting selected", Toast.LENGTH_SHORT).show();
 break;
 case R.id.action1:
 Toast.makeText(this, "Menu Action 1 selected", Toast.LENGTH_SHORT).show();
 break;
 case R.id.action2:
 Toast.makeText(this, "Menu Action 2 selected", Toast.LENGTH_SHORT).show();
 break;
 case 13:
 Toast.makeText(this, "Menu Action 3 selected", Toast.LENGTH_SHORT).show();
 break;
 case 14:
 Toast.makeText(this, "Menu Action 4 selected", Toast.LENGTH_SHORT).show();
 }
}
```

```

 break;
 case 15:
 Toast.makeText(this, "Menu Action 5
 selected", Toast.LENGTH_SHORT).show();
 break;
 default:
 break;
 }
 return super.onOptionsItemSelected(item);
}

```



Hình 2-49 Kết quả khi click trên menu Action 1

**B7.-** Chạy chương trình để xem kết quả, thông thường các MenuItem chứa trong “vùng tràn” sẽ không có icon được hiển thị.

### 2.3.2. Sử dụng Action Bar

Một số tính năng được bổ sung trong Action Bar là Tabs Navigation và Drop-Down Navigation. Để sử dụng những tính năng này, bạn cần can thiệp trực tiếp bằng mã lệnh.

#### 2.3.2.1. Drop-Down Navigation

Các bước thực hiện

**B1.-** Tương tương ứng với mỗi item, cần tạo:

- File .xml thiết kế layout sẽ dùng khi click chọn item
- Tạo class mới thực thi class Fragment để xử lý file layout đó. VD:

```

public class Fragment1 extends Fragment
{
 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container,
 Bundle savedInstanceState)
 {
 return inflater.inflate(R.layout.tab1,container, false);
 }
}

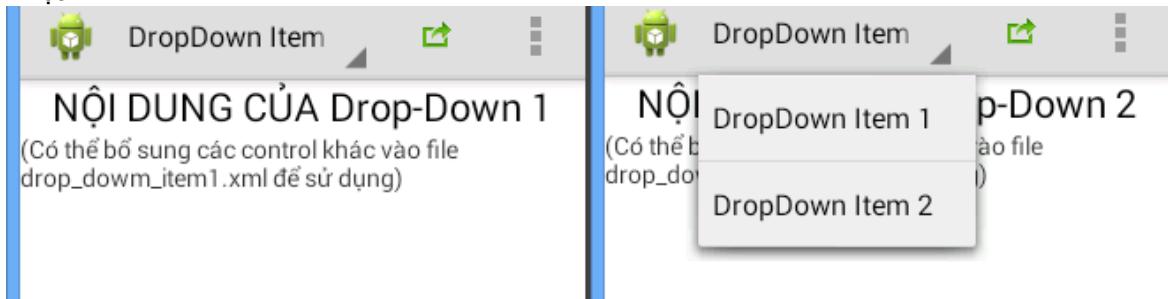
```

**B2.-** Hiệu chỉnh nội dung sự kiện onCreate trong file MainActivity.java

- Bỏ lệnh setContentView(R.layout.activity\_main); vì bạn sẽ không sử dụng activity này nữa.
- Tạo SpinnerAdapter.
- Gán SpinnerAdapter vào ActionBar trước đó để xử lý phương thức OnNavigationListener.

### BÀI THỰC HÀNH App\_09 (tiếp theo)

☞ **Yêu cầu:** Tạo mới activity DropDown\_Navigation, khi chạy sẽ hiển thị 2 drop-down item như hình minh họa.



Hình 2-50 Giao diện cần thực hiện

### Thực hiện

- B1.- Tạo các giao diện và xử lý cho từng item cần dùng

– Item thứ 1:

- Tạo mới file `res\layout\drop_down_item3.xml` với nội dung:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@+id/LinearLayout1"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical" >
 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:textAppearance="?android:attr/textAppearanceLarge"
 android:layout_gravity="center"
 android:text="NỘI DUNG CỦA Drop-Down 1" />
 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="(Có thể bổ sung các control khác vào file
 drop_down_item1.xml để sử dụng)" />
</LinearLayout>
```

- Tạo class mới kế thừa từ class Fragment để xử lý file layout đó. VD:

```
package com.example.app_09;
```

```
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
public class Fragment3 extends Fragment
{
 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
 {
 return inflater.inflate(R.layout.drop_down_item3,container, false);
 }
}
```

– Tab thứ 2:

- Tạo mới file `res\layout\tab2.xml` với nội dung:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@+id/LinearLayout1"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical" >
 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:textAppearance="?android:attr/textAppearanceLarge"
 android:layout_gravity="center"
 android:text="NỘI DUNG CỦA Drop-Down 1" />
 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="(Có thể bổ sung các control khác vào file drop_down_item1.xml để
 sử dụng)" />
</LinearLayout>
```

- Tạo class mới kế thừa từ class Fragment để xử lý file layout đó. VD:

```
package com.example.app_09;
```

```
import android.app.Fragment;
```

```
import android.os.Bundle;
```

```
import android.view.LayoutInflater;
```

```
import android.view.View;
```

```

import android.view.ViewGroup;

public class Fragment4 extends Fragment
{
 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
 savedInstanceState)
 { return inflater.inflate(R.layout. drop_down_item4, container, false);
 }
}

```

**B2.-** Tạo mới file `src\com\example\DropDownNavigation.java` bằng cách copy từ file `src\com\example\MainActivity.java`. Hiệu chỉnh nội dung sự kiện `onCreate` trong file `DropDownNavigation.java` gồm 3 việc:

- Bỏ lệnh `setContentView(R.layout.activity_main)`; vì sẽ không sử dụng activity này nữa.
- Tạo SpinnerAdapter
- Gán SpinnerAdapter vàoActionBar trước đó để xử lý phương thức `OnNavigationListener`.

Sau khi hoàn tất, mã lệnh của file `DropDownNavigation.java` có dạng:

```

package com.example.app_09;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import android.support.v7.app.ActionBarActivity;
import android.app.ActionBar;
import android.app.ActionBar.OnNavigationListener;
import android.app.Fragment;
import android.app.FragmentManager;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.SimpleAdapter;

public class DropDownNavigation extends ActionBarActivity
{
 @Override
 protected void onCreate(Bundle savedInstanceState)
 {
 super.onCreate(savedInstanceState);
 //setContentView(R.layout.option_menu);
 /*set the navigation mode of the ActionBar and remove the title text to allow more
 space for the spinner */
 ActionBar actionBar = getSupportActionBar();
 actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_LIST);
 actionBar.setTitle("");

 //create a SpinnerAdapter
 final List<Map<String, Object>> data = new ArrayList<Map<String, Object>>();
 Map<String, Object> map = new HashMap<String, Object>();

 map.put("title", "DropDown Item 1");
 map.put("fragment", Fragment.instantiate(this, Fragment3.class.getName()));
 data.add(map);

 map = new HashMap<String, Object>();
 map.put("title", "DropDown Item 2");
 map.put("fragment", Fragment.instantiate(this, Fragment4.class.getName()));
 data.add(map);

 SimpleAdapter adapter = new SimpleAdapter(this, data,
 android.R.layout. simple_spinner_dropdown_item,
 new String[] { "title" }, new int[] { android.R.id.text1 });
 }
}

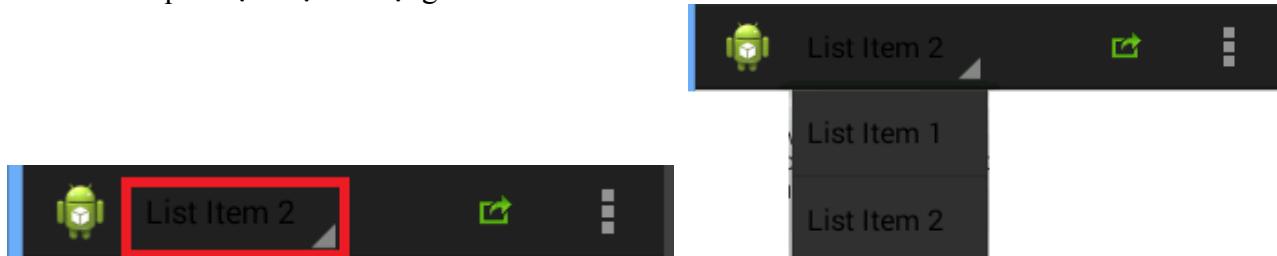
```

```

/*set SpinnerAdapter on the ActionBar with a OnNavigationListener to handle the
navigation callback. i.e. This gets call when the user changes the selection */
actionBar.setListNavigationCallbacks(adapter, new OnNavigationListener()
{
 @Override
 public boolean onNavigationItemSelected(int itemPosition, long itemId)
 {
 Map<String, Object> map = data.get(itemPosition);
 Object o = map.get("fragment");
 if(o instanceof Fragment)
 {
 FragmentTransaction tx = getFragmentManager().beginTransaction();
 tx.replace(android.R.id.content, (Fragment)o);
 tx.commit();
 }
 return true;
 }
});
@Override
public boolean onCreateOptionsMenu(Menu menu)
{
 // Inflate the menu; this adds items to the action bar if it is present.
 getMenuInflater().inflate(R.menu.main, menu);
 return super.onCreateOptionsMenu(menu);
}
@Override
public boolean onOptionsItemSelected(MenuItem item)
{ /* Handle action bar item clicks here. The action bar will automatically handle
clicks on the Home/Up button, so long as you specify a parent activity in
AndroidManifest.xml.*/
 return true;
}
}

```

Kết quả thực hiện có dạng:



Hình 2-51 Giao diện khi chưa thay đổi nền của ActionBar

**B3.-** (Bước bổ sung) Với kết quả trên, ta thấy nội dung hiển thị quá đậm (khó nhìn thấy chữ). Vì vậy ta hiệu chỉnh lại theme cần dùng là [Theme.AppCompat.Light](#) trong file res\values-v14\styles.xml để có nội dung như sau:

```

<resources>
 <!-- Base application theme for API 14+. This theme completely replaces AppBaseTheme
 from BOTH res/values/styles.xml and res/values-v11/styles.xml on API 14+ devices. -->
 <style name="AppTheme" parent="Theme.AppCompat.Light">
 <!-- API 14 theme customizations can go here. -->
 </style>
</resources>

```

**B4.-** Chạy chương trình để xem kết quả.

### 2.3.2.2. Tab Navigation

Các bước thực hiện

**B1.-** Tương tương ứng với mỗi tab cần tạo:

- File .xml thiết kế layout của tab
- Tạo class mới kế thừa từ class Fragment để xử lý file layout đó. VD:

```
public class Fragment1 extends Fragment
{
 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
 {
 return inflater.inflate(R.layout.tab1, container, false);
 }
}
```

**B2.-** Hiệu chỉnh nội dung sự kiện onCreate trong file MainActivity.java

- Bỏ lệnh setContentView(R.layout.activity\_main); vì bạn sẽ không sử dụng activity này nữa.
- Khai báo 1 đối tượng thuộc class ActionBar và khai báo kiểu của action bar

VD:     ActionBar ab = getActionBar();
             ab.setNavigationMode( ActionBar.NAVIGATION\_MODE\_TABS );

- Đổi với mỗi tab cần thêm:

- Khai báo tab mới

VD:     Tab tab = ab.newTab().setText("Tab 1")
             .setTabListener(new MyTabListener(this, Fragment1.class.getName()));

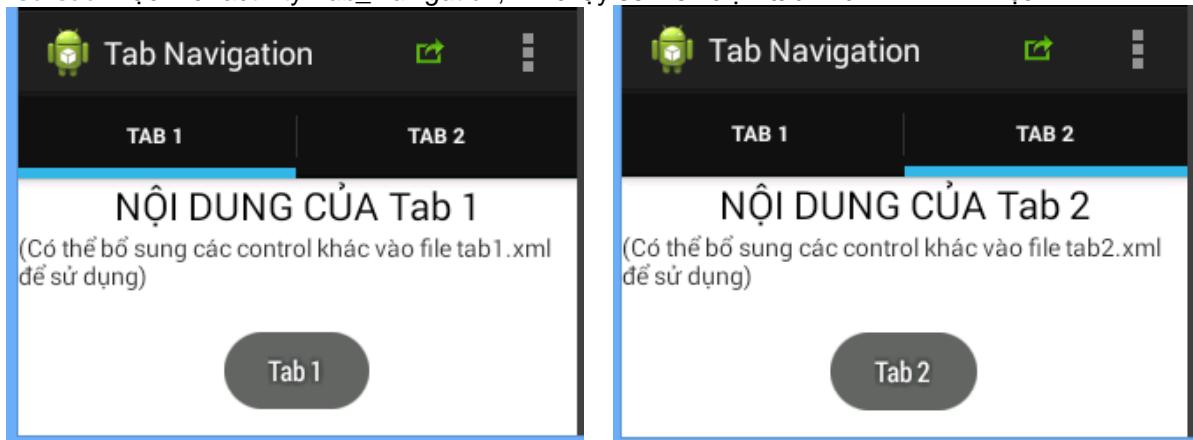
- Thêm tab mới vào ActionBar

VD:     ab.addTab( tab );

**B3.-** Tạo class xử lý sự kiện khi người dùng click chọn trên tab. Class này được đặt nằm trong nằm trong class MainActivity, class này thực thi class ActionBar.TabListener, trong đó định nghĩa một số phương thức như onTabReselected, onTabSelected, onTabUnselected.

## BÀI THỰC HÀNH App\_09 (tiếp theo)

☞ **Yêu cầu:** Tạo mới activity Tab\_Navigation, khi chạy sẽ hiển thị 2 tab như hình minh họa.



Hình 2-52 Giao diện cần thực hiện

☞ **Thực hiện**

**B1.-** Tạo các tab cần dùng

- Tab thứ 1:

- Tạo mới file res\layout\tab1.xml với nội dung:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@+id/LinearLayout1"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical" >
 <TextView
 android:layout_width="wrap_content"
```

```

 android:layout_height="wrap_content"
 android:textAppearance="?android:attr/textAppearanceLarge"
 android:layout_gravity="center"
 android:text="NỘI DUNG CỦA Tab 1" />
 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="(Có thể bổ sung các control khác vào file tab1.xml để sử dụng)"/>
</LinearLayout>

```

- Tạo class mới kế thừa từ class Fragment để xử lý file layout đó. VD:

```

package com.example.app_09;
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)
 {
 return inflater.inflate(R.layout.tab1,container, false);
 }
}

```

- Tab thứ 2:

- Tạo mới file res\layout\tab2.xml với nội dung:

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@+id/LinearLayout2"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical" >
 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:textAppearance="?android:attr/textAppearanceLarge"
 android:layout_gravity="center"
 android:text="NỘI DUNG CỦA Tab 2" />
 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="(Có thể bổ sung các control khác vào file tab2.xml để sử dụng)"/>
</LinearLayout>

```

- Tạo class mới kế thừa từ class Fragment để xử lý file layout đó. VD:

```

package com.example.app_09;
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)
 {
 return inflater.inflate(R.layout.tab2,container, false);
 }
}

```

**B2.-** Hiệu chỉnh nội dung sự kiện onCreate trong file MainActivity.java gồm 3 việc:

- Bỏ lệnh `setContentView(R.layout.activity_main)`
- Khai báo 1 đối tượng thuộc class ActionBar và khai báo kiểu của action bar
- Đối với mỗi tab cần thêm:
  - Khai báo tab mới
  - Thêm tab mới vào ActionBar

Sau khi hoàn tất, mã lệnh của file `MainActivity.java` có dạng:

```
package com.example.app_09;
import android.support.v7.app.ActionBarActivity;
import android.app.ActionBar;
import android.app.ActionBar.Tab;
import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends ActionBarActivity {
 Tab mtab1, mtab2;
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 //setContentView(R.layout.activity_main);

 ActionBar ab = getSupportActionBar();
 ab.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
 // Tab thứ 1
 mtab1 = ab.newTab().setText("Tab 1")
 .setTabListener(new MyTabListener(this, Fragment1.class.getName()));
 ab.addTab(mtab1);
 // Tab thứ 2
 mtab2 = ab.newTab().setText("Tab 2")
 .setTabListener(new MyTabListener(this, Fragment2.class.getName()));
 ab.addTab(mtab2);
 }
 // các phương thức sẵn có trong class sẽ nằm ngay sau dòng ghi chú này
 ...
}
```

**B3.-** Tạo class xử lý sự kiện khi người dùng click chọn trên tab. Trong đó, chỉ chặn sự kiện khi click chọn trên tab. Sau khi bổ sung hoàn tất, nội dung file `MainActivity.java` có dạng:

```
package com.example.app_09;
import android.support.v7.app.ActionBarActivity;
import android.app.ActionBar;
import android.app.ActionBar.Tab;
import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

public class MainActivity extends ActionBarActivity {
 Tab mtab1, mtab2;
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 //setContentView(R.layout.activity_main);

 ActionBar ab = getSupportActionBar();
```

```
ab.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);

 mtab1 = ab.newTab().setText("Tab 1").setTabListener(new MyTabListener(this,
 Fragment1.class.getName()));
 ab.addTab(mtab1);

 mtab2 = ab.newTab().setText("Tab 2").setTabListener(new MyTabListener(this,
 Fragment2.class.getName()));
 ab.addTab(mtab2);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
 int id = item.getItemId();
 if (id == R.id.action_settings) {
 return true;
 }
 return super.onOptionsItemSelected(item);
}
//inner class
private class MyTabListener implements ActionBar.TabListener
{
 private Fragment mFragment;
 private final Activity mActivity;
 private final String mFragName;
 public MyTabListener(Activity activity, String fragName)
 {
 mActivity = activity;
 mFragName = fragName;
 }
 @Override
 public void onTabReselected(Tab tab, FragmentTransaction ft)
 { // TODO Auto-generated method stub
 }
 @Override
 public void onTabSelected(Tab tab, FragmentTransaction ft)
 {
 mFragment = Fragment.instantiate(mActivity, mFragName);
 ft.add(android.R.id.content, mFragment);
 Toast toast = Toast.makeText(getApplicationContext(),"tab",Toast.LENGTH_SHORT);
 if (tab.equals(mtab1))
 toast.setText("Tab 1");
 else
 toast.setText("Tab 2");
 toast.show();
 }
 @Override
 public void onTabUnselected(Tab tab, FragmentTransaction ft)
 {
 ft.remove(mFragment);
 mFragment = null;
 }
}
}
```

Lưu ý: do không đủ chỗ trống nên tab control bị cắt xuống dòng thứ 2. Trong những thiết bị có kích thước lớn hơn hoặc khi bạn chuyển sang màn hình ngang, ActionBar sẽ sắp xếp chúng trên cùng 1 dòng.

## 2.4. Activities & Fragments

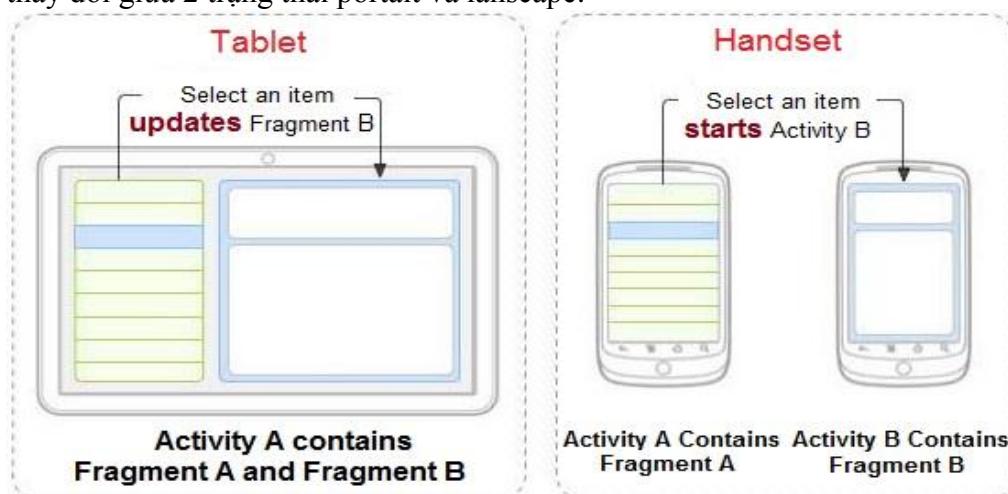
### 2.4.1. Fragments

#### 2.4.1.1. Giới thiệu

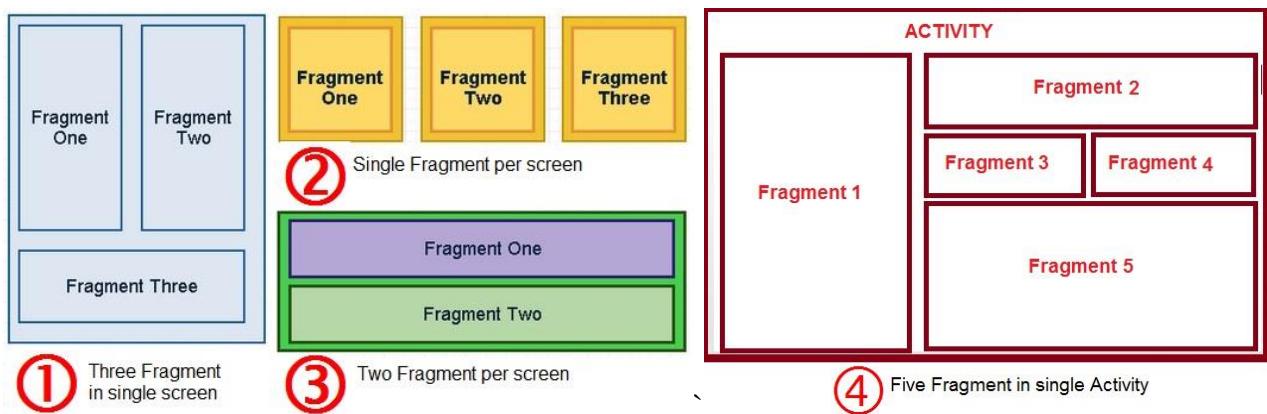
Được giới thiệu từ Android 3.0 (Honeycomb), *Fragment* là một phần giao diện người dùng hoặc hành vi của một ứng dụng. *Fragment* có thể được đặt trong *Activity*, nó có thể cho phép thiết kế *activity* với nhiều module. Có thể nói *Fragment* là một loại *sub-Activity*.

#### 2.4.1.2. Một số tính chất

- *Fragment* phải luôn luôn được nhúng trong *activity*. *Fragment* cũng có layout của riêng của nó, cũng có các hành vi và vòng đời riêng.
- Có thể thêm hoặc xóa *Fragment* trong một *Activity* trong khi *Activity* này đang chạy.
- Có thể kết hợp nhiều *Fragment* trong một *Activity* để xây dựng giao diện người dùng đa khung. Ngược lại, một *Fragment* cũng có thể được sử dụng trong nhiều *Activities*.
- Vòng đời của *Fragment* có quan hệ chặt chẽ với vòng đời của *Activity* đang dùng nó điều này có nghĩa là khi *Activity* bị tạm dừng thì các *Fragment* cũng sẽ dừng hoặc khi *Activity* bị hủy thì các fragment được gắn kèm cũng bị hủy theo.
- *Fragment* có thể thực hiện một hành vi mà không có trong thành phần giao diện người dùng.
- *Fragment* được thêm vào API 11 trở lên.
- Có thể tạo các *Fragments* bằng cách kế thừa class *Fragment* và *Fragment* được thêm vào layout bởi thẻ `<fragment>`
- Như đã biết, tại cùng một thời điểm chúng ta chỉ có thể hiển thị một *Activity* duy nhất trên màn hình. Vì thế, nếu quản lý theo *Activity*, chúng ta không thể chia màn hình thiết bị ra thành nhiều phần và kiểm soát các thành phần khác nhau này một cách riêng biệt. Nhưng với *Fragment* thì màn hình được linh hoạt hơn, nhờ *Activity* có thể chứa nhiều *Fragment* với layout, sự kiện, và vòng đời riêng.
- Kỹ thuật sử dụng nhiều fragment trên cùng 1 màn hình thường được dùng với những thiết bị có màn hình lớn (máy tính bảng, tivi – Tablet, televisions), còn với màn hình nhỏ (Handset) thì thường dùng khi muốn ứng dụng phù hợp trên nhiều kích cỡ màn hình hoặc khi thiết bị thay đổi giữa 2 trạng thái portrait và landscape.

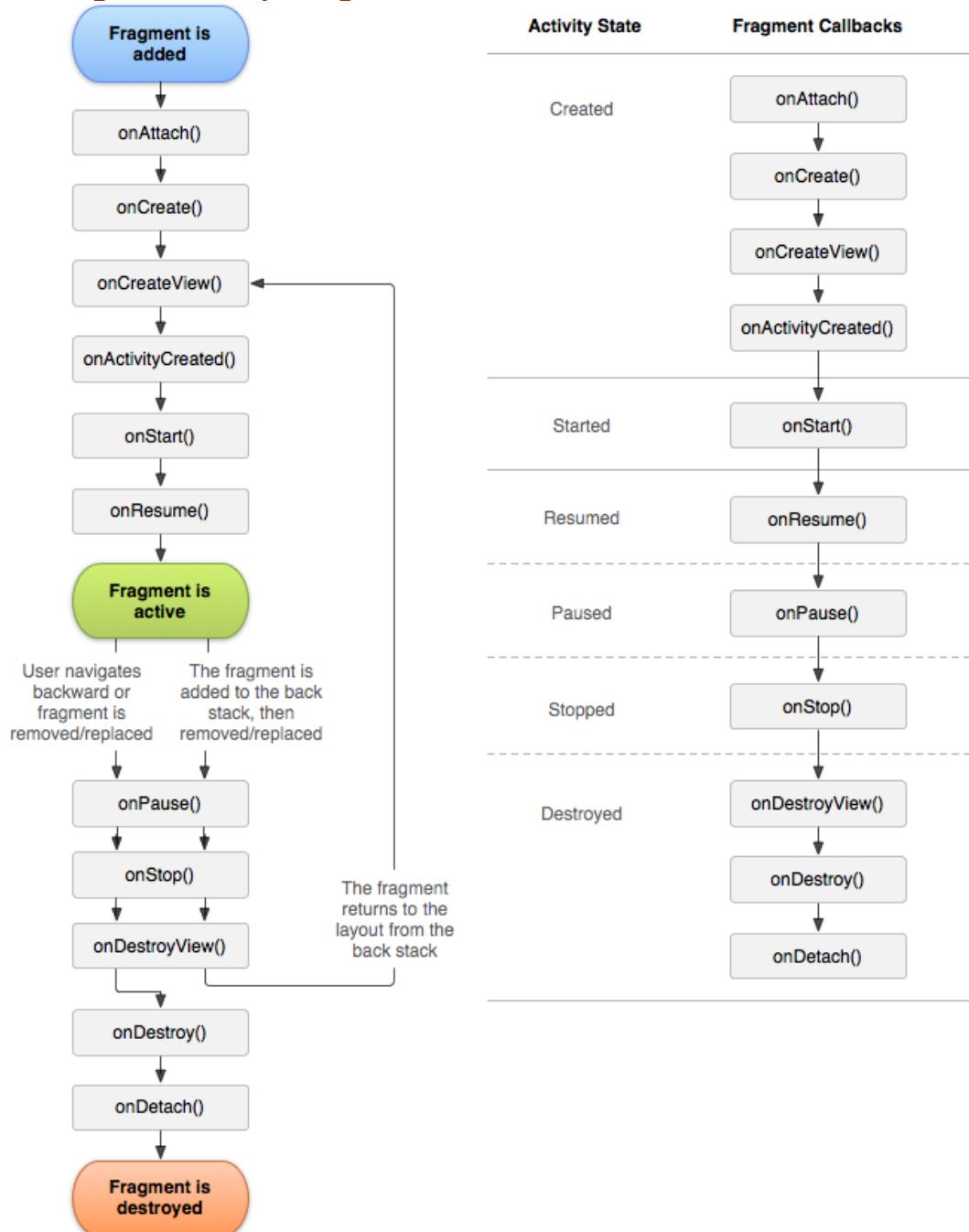


Hình 2-53 Minh họa sử dụng fragment trên thiết bị có màn hình lớn (máy tính bảng, tivi – Tablet, televisions) và với màn hình nhỏ (Handset)



Hình 2-54 Minh họa sử dụng nhiều fragment trên cùng 1 màn hình

#### 2.4.1.3. Vòng đời của một Fragment



Hình 2-55 Vòng đời của fragment

Mỗi Fragment có vòng đời riêng, và vòng đời này giống với vòng đời của một Activity.

- Giai đoạn 1: Khi một Fragment được tạo ra sẽ trải trạng thái:
  - (i). onAttach()
  - (ii). onCreate()
  - (iii). onCreateView()
  - (iv). onActivityCreated()
- Giai đoạn 2: Khi một Fragment được hiển thị:
  - (i). onStart()
  - (ii). onResume()
- Giai đoạn 3: Khi Fragment chạy ẩn dưới nền:
  - (i). onPause()
  - (ii). onStop()
- Giai đoạn 4: Khi hủy một Fragment:
  - (v). onPause()
  - (vi). onStop()
  - (vii). onDestroyView()
  - (viii). onDestroy()
  - (ix). onDetach()

#### 2.4.1.4. Cách tạo một Fragment

B1. Xác định số lượng Fragment cần sử dụng trong một Activity.

B2. Tương ứng với mỗi Fragment, bạn cần phải tạo các file Layout (XML file).

Ví dụ: cần tạo Activity có chứa 1 fragment tên fragment\_a. Ta tạo mới file res\layout\fragment\_a.xml có dạng

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent" >
 <TextView android:id="@+id/textView1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Layout for fragment A"
 android:textAppearance="?android:attr/textAppearanceLarge" >
 </TextView>
</RelativeLayout>
```

B3. Tương ứng với mỗi Fragment, tạo class kế thừa class Fragment. Class Fragment có nhiều phương thức callback cho phép override tùy vào yêu cầu sử dụng.

Ví dụ: tạo class mới trong src\com.example.XXX\Fragment\_A.java có dạng

```
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
public class FragmentA extends Fragment
{
 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container,
 Bundle savedInstanceState)
 {
 View v = inflater.inflate(R.layout.fragment_a, container, false);
 return v;
 }
}
```

B4. Chính sửa file Activity, để xác định vị trí của các Fragment theo yêu cầu.

Trong sự kiện `onCreate` của `MainActivity.java`, tạo ra 1 đối tượng thuộc class vừa tạo (`FragmentA`) ở B3 và 1 đối tượng `FragmentTransaction` (ft) để buộc `FragmentA` vào layout thông qua phương thức `replace()` của `FragmentTransaction`.

```

import com.example.app_10.FragmentA;
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentTransaction;

public class MainActivity extends FragmentActivity {

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

 fragmentManager= getSupportFragmentManager();

 FragmentTransaction transaction= fragmentManager.beginTransaction();
 transaction.add(R.id.fragment_a, new FragmentA(), "myFragment");
 transaction.addToBackStack(null);
 transaction.commit();
 }
 // các phương thức sẵn có trong class sẽ nằm ngay sau dòng ghi chú này
 // . .
}

```

#### 2.4.1.5. Phương thức thường dùng với Fragment

- **`add`**: thêm một fragment vào layout  
`add(int containerViewId, Fragment fragment, String tag)`
- **`remove`** gỡ bỏ một fragment có trong layout  
`remove(Fragment fragment)`
- **`replace`**: thay thế một fragment có trong layout  
`replace(int containerViewId, Fragment fragment, String tag)`
- **`addToBackStack`**: khi click nút back có thể quay về Fragment trước đó. Fragment sẽ không bị hủy khi fragment bị removed hoặc replaced, nó sẽ bị dừng. Phương thức `addToBackStack()` cũng phải được đặt trước phương thức `commit()`. Đối số truyền vào phương thức:
  - **`null`**: tất cả Fragment đều được đưa vào stack, do đó hoạt động theo kiểu của stack
  - **`tên_fragmentTransaction`**: xác định rõ tên của `FragmentTransaction` cần chuyển đến khi click vào back.
- **`SetTransition()`**: thiết lập animation cho `FragmentTransaction`.
- **`commit`**: Hoàn tất transaction. Luôn luôn phải có sau khi dùng các method trên, và luôn đặt ở cuối cùng.  
`commit()`

#### 2.4.1.6. Phương thức được phép override:

- **`onCreate()`**: Hệ thống gọi phương thức này khi tạo `Fragment`. Bạn nên khởi tạo các thành phần thiết yếu của `Fragment` mà bạn muốn giữ lại khi `Fragment` được tạm dừng, dừng lại hoặc tiếp tục.
- **`onCreateView()`**: Hệ thống gọi phương thức này khi `Fragment` vẽ giao diện của nó lần đầu tiên. Để vẽ giao diện cho `Fragment` bạn cần phải **return View** từ phương thức này. Bạn có thể `return null` nếu `Fragment` không cung cấp giao diện.

- **onPause()**: Hệ thống gọi phương thức này như là để đánh dấu lần đầu người dùng rời Fragment. Đây là nơi bạn ghi lại bất kỳ thay đổi nào và thay đổi cần tiếp tục tồn tại ngoài lần dùng này.

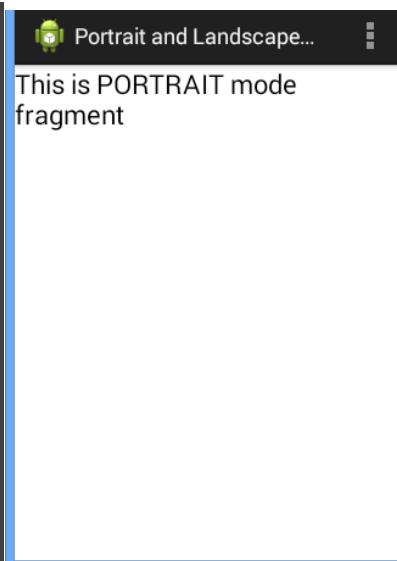
## BÀI THỰC HÀNH App\_10

### ☞ Yêu cầu:

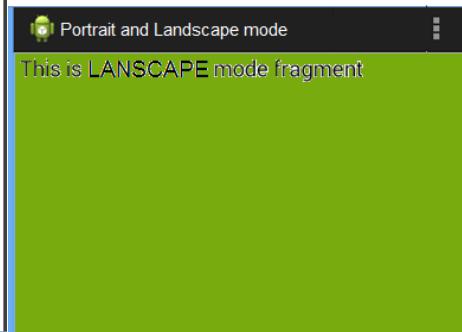
- Tạo mới project App\_10 gồm 1 textview và 3 button, có giao diện như hình



Hình 2-56 Màn hình chính



Hình 2-57 Thiết bị khi ở trạng thái Portrait và Landscape



- Tạo tiếp 1 activity *SwitchBetweenPortraitAndLandscape* để khi người dùng click chọn button “Chuyển đổi giữa Portrait và Landscape mode” sẽ mở ra activity này.
- Activity *SwitchBetweenPortraitAndLandscape* gồm 2 fragment để thay đổi khi người dùng để thiết bị theo chiều đứng hoặc theo chiều ngang

### ☞ Thực hiện

**B1.-** Tạo mới project App\_10

**B2.-** Tạo mới activity và đặt tên là *SwitchBetweenPortraitAndLandscape*.

**B3.-** Tạo 2 file layout *res\layout\landscape\_fragment.xml* và *res\layout\portrait\_fragment.xml* để minh họa giao diện cho 2 trạng thái tương ứng là “nằm” và “đứng” của thiết bị

- Nội dung file *res\layout\landscape\_fragment.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"
 android:background="#7bae16">
 <TextView android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="This is LANDSCAPE mode fragment "
 android:textColor="#000000"
 android:textAppearance="?android:attr/textAppearanceLarge"/>
</LinearLayout>
```

- Nội dung file *res\layout\portrait\_fragment.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"
 android:background="#7bae16">
 <TextView android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="This is PORTRAIT mode fragment "
```

```
 android:textColor="#000000"
 android:textAppearance="?android:attr/textAppearanceLarge"/>
 </LinearLayout>
```

**B4.-** Tạo 2 file class src\com.example.app\_10\Landscape\_Fragment.java và src\com.example.app\_10\Portrait\_Fragment.java

- Nội dung file src\com.example.app\_10\Landscape\_Fragment.java:

```
package com.example.app_10;
import android.support.v4.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
public class Landscape_Fragment extends Fragment
{
 @Override
 public View onCreateView(LayoutInflater inflater,
 ViewGroup container, Bundle savedInstanceState)
 {
 View v = inflater.inflate(R.layout.Landscape_fragment,
 container, false);
 return v;
 }
}
```

- Nội dung file src\com.example.app\_10\Portrait\_Fragment.java:

```
package com.example.app_10;
import android.support.v4.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
public class Portrait_Fragment extends Fragment
{
 @Override
 public View onCreateView(LayoutInflater inflater,
 ViewGroup container, Bundle savedInstanceState)
 {
 View v = inflater.inflate(R.layout.portrait_fragment,
 container, false);
 return v;
 }
}
```

**B5.-** Chỉnh sửa lại nội dung file res\layout\activity\_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="fragments"
 android:id="@+id/Landscape_mode_fragment"
 android:layout_height="match_parent"
 android:layout_weight="1"
 android:layout_width="0dp"/>
 <fragment android:name="fragments"
 android:id="@+id/portrait_mode_fragment"
 android:layout_height="match_parent"
 android:layout_width="0dp"
 android:layout_weight="2"/>
</LinearLayout>
```

B6.- Chỉnh sửa lại nội dung file src\com.example.app\_10\MainActivity.java:

```

package com.example.app_10;
import com.example.app_10.Landscape_Fragment;
import com.example.app_10.Porait_Fragment;
import android.os.Bundle;
import android.app.Activity;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.content.res.Configuration;
import android.view.Menu;
public class SwitchBetweenPortraitAndLandscape extends Activity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 Configuration config = getResources().getConfiguration();
 FragmentManager fragmentManager = getFragmentManager();
 FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

 if(config.orientation == Configuration.ORIENTATION_LANDSCAPE)
 { Landscape_Fragment lm_Fragment = new Landscape_Fragment();
 fragmentTransaction.replace(android.R.id.content, lm_Fragment);
 }
 else
 { Porait_Fragment pm_Fragment = new Porait_Fragment();
 fragmentTransaction.replace(android.R.id.content, pm_Fragment);
 }
 fragmentTransaction.commit();
 }
 @Override
 public boolean onCreateOptionsMenu(Menu menu) {
 // Inflate the menu; this adds items to the action bar if it is present.
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }
}

```

B7.- Khai báo activity *SwitchBetweenPortraitAndLandscape* trong file *AndroidManifest.xml*.

B8.- Viết lệnh bổ sung để khi người dùng click chọn button “*Chuyển đổi giữa Portrait và Landscape mode*” sẽ mở ra activity *SwitchBetweenPortraitAndLandscape* vừa xây dựng xong.

B9.- Chạy ứng dụng để xem kết quả.

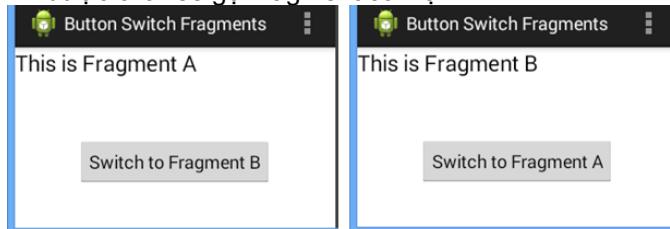
## 2.4.2. Di chuyển giữa các Fragments

### 2.4.2.1. Di chuyển qua lại giữa các Fragments

## BÀI THỰC HÀNH App\_10 (tiếp theo)

### ☞ Yêu cầu

- Tạo mới 1 activity *SwitchFragmentsWithButton* để khi người dùng click chọn button “*Di chuyển giữa các Fragment thông qua button*” sẽ mở ra activity này.
- Activity *SwitchFragmentsWithButton* gồm 2 fragment (FragmentA và FragmentB), trên mỗi fragment sẽ có 1 button để được click sẽ gọi fragment còn lại



Hình 2-58 Minh họa yêu cầu của bài thực hành di chuyển qua lại giữa các fragment

### ☞ Thực hiện

B1.- Tạo mới activity và đặt tên là *SwitchFragmentsWithButton*.

B2.- Tạo 3 file layout *res\layout\switch\_fragments\_with\_button.xml*, *res\layout\fragment\_a.xml* và *res\layout\fragment\_b.xml* với nội dung lần lượt là:

- Nội dung file res\layout\switch\_fragments\_with\_button.xml. Chú ý đến id của LinearLayout, vì tên này sẽ được gọi trong 2 phương thức showFragmentA và showFragmentB có trong file SwitchFragmentsWithButton.java:
 

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 <LinearLayout android:id="@+id/Layout_container"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:layout_centerHorizontal="true"
 android:orientation="horizontal">
 </LinearLayout>
 </RelativeLayout>
```
- Nội dung file res\layout\fragment\_a.xml:
 

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 <TextView android:id="@+id/textView1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="This is Fragment A"
 android:textAppearance="?android:attr/textAppearanceLarge">
 </TextView>
 <Button android:id="@+id/button1a"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_centerInParent="true"
 android:text="Switch to Fragment B"/>
</RelativeLayout>
```
- Nội dung file res\layout\fragment\_b.xml:
 

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 <TextView android:id="@+id/textView1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="This is Fragment B"
 android:textAppearance="?android:attr/textAppearanceLarge">
 </TextView>
 <Button android:id="@+id/button1b"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_centerInParent="true"
 android:text="Switch to Fragment A"/>
</RelativeLayout>
```

**B3.-** Tạo 2 file class src\com.example.app\_10\ FragmentA.java và src\com.example.app\_10\ FragmentB.java

- Nội dung file src\com.example.app\_10\ FragmentA.java:
 

```
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.View.OnClickListener;
import android.widget.Button;
public class FragmentA extends Fragment {
 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container,
 Bundle savedInstanceState)
```

```

 {
 View v = inflater.inflate(R.layout.fragment_a, container, false);
 Button button = (Button) v.findViewById(R.id.button1a);
 button.setOnClickListener(new OnClickListener()
 {
 public void onClick(View v)
 {
 SwitchFragmentsWithButton currentActivity =
 (SwitchFragmentsWithButton) getActivity();
 currentActivity.showFragmentB();
 }
 });
 return v;
 }
}
- Nội dung file src\com.example.app_10\FragmentB.java:
package com.example.app_10;
import com.example.app_10.R;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.View.OnClickListener;
import android.widget.Button;
public class FragmentB extends Fragment
{
 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container,
 Bundle savedInstanceState)
 {
 View v = inflater.inflate(R.layout.fragment_b, container, false);
 Button button = (Button) v.findViewById(R.id.button1b);
 button.setOnClickListener(new OnClickListener()
 {
 public void onClick(View v)
 {
 SwitchFragmentsWithButton currentActivity =
 (SwitchFragmentsWithButton) getActivity();
 currentActivity.showFragmentA();
 }
 });
 return v;
 }
}

```

**B4.-** Copy và paste file src\com.example.app\_10\MainActivity.java để có file src\com.example.app\_10\SwitchFragmentsWithButton.java. Chính sửa nội dung file src\com.example.app\_10\SwitchFragmentsWithButton.java để có nội dung như sau:

```

package com.example.app_10;
import android.app.Activity;
import android.app.FragmentTransaction;
import android.os.Bundle;
import android.view.Menu;
public class SwitchFragmentsWithButton extends Activity
{
 @Override
 protected void onCreate(Bundle savedInstanceState)
 {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.switch_fragments_with_button);
 showFragmentA();
 }
 public void showFragmentA(){
 FragmentA fragmentA = new FragmentA();
 FragmentTransaction ft = getFragmentManager().beginTransaction();
 ft.replace(R.id.layout_container, fragmentA);
 ft.addToBackStack("fragment_a");
 ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
 ft.commit();
 }
 public void showFragmentB(){

```

```

 FragmentB fragmentB = new FragmentB();
 FragmentTransaction ft = getFragmentManager().beginTransaction();
 ft.replace(R.id.layout_container, fragmentB);
 ft.addToBackStack("fragment_b");
 ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
 ft.commit();
 }
 @Override
 public boolean onCreateOptionsMenu(Menu menu) {
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }
}

```

**B5.-** Khai báo activity *SwitchFragmentsWithButton* trong file *AndroidManifest.xml*.

**B6.-** Bổ sung lệnh cho file *MainActivity.java* để khi người dùng click chọn button “*Di chuyển giữa các Fragment thông qua button*” sẽ mở ra activity *SwitchFragmentsWithButton*.

**B7.-** Chạy ứng dụng để xem kết quả.

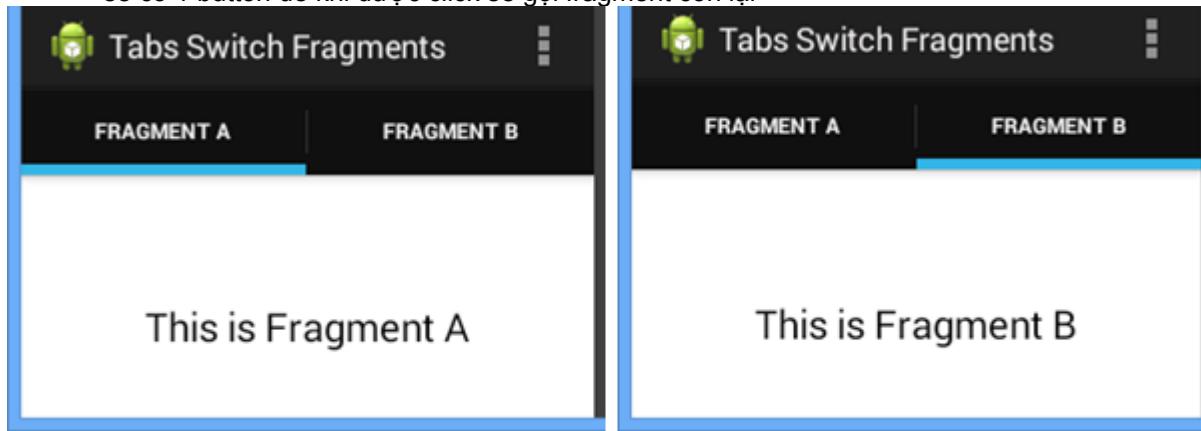
#### 2.4.2.2. Sử dụng Fragments với ActionBar

Trong bài thực hành *App\_09*, chúng ta đã tạo ra một action bar với 2 tab, mỗi tab chứa 1 activity. Trong phần này chúng ta cũng sử dụng Action bar nhưng để chuyển đổi các fragment trong cùng 1 activity.

### BÀI THỰC HÀNH App\_10 (tiếp theo)

#### ☛ Yêu cầu:

- Tạo mới 1 activity *SwitchFragmentsWithTabs* để khi người dùng click chọn button “*Di chuyển giữa các Fragment thông qua Tab*” sẽ mở ra activity này.
- Activity *SwitchFragmentsWithButton* gồm 2 fragment (*FragmentA* và *FragmentB*), trên mỗi fragment sẽ có 1 button để khi được click sẽ gọi fragment còn lại



Hình 2-59 Mỗi fragment được đặt trong 1 Tab

#### ☛ Thực hiện

**B1.-** Tạo mới activity và đặt tên là *SwitchFragmentsWithTabs*.

**B2.-** Mặc dù có thể sử dụng các fragment\_a và fragment\_b của bài thực hành vừa thực hiện, nhưng để tạo thói quen, trong phần này chúng ta vẫn tạo fragment riêng cho bài thực hành đang thực hiện. Tạo 3 file layout res\layout\switch\_fragments\_with\_button.xml res\layout\fragment\_tab\_a.xml và res\layout\fragment\_tab\_b.xml với nội dung lần lượt là:

- Nội dung file res\layout\switch\_fragments\_with\_button.xml. Chú ý đến id của LinearLayout, vì tên này sẽ được gọi trong 2 phương thức *showFragmentA* và *showFragmentB* có trong file *SwitchFragmentsWithTabs.java*:

```

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

```

```

 android:layout_centerHorizontal="true"
 android:orientation="horizontal">
 </LinearLayout>
</RelativeLayout>
- Nội dung file res\layout\fragment_tab_a.xml:
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 <TextView android:id="@+id/textViewTabA"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_centerInParent="true"
 android:text="This is Fragment A"
 android:textAppearance="?android:attr/textAppearanceLarge">
 </TextView>
</RelativeLayout>
- Nội dung file res\layout\fragment_tab_b.xml:
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 <TextView android:id="@+id/textViewTabB"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_centerInParent="true"
 android:text="This is Fragment B"
 android:textAppearance="?android:attr/textAppearanceLarge">
 </TextView>
</RelativeLayout>

```

### B3.- Tạo mới 2 class FragmentTabA.java và FragmentTabB.java:

- Nội dung file class FragmentTabA.java:

```

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
public class FragmentTabA extends Fragment {
 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container,
 Bundle savedInstanceState)
 { View v = inflater.inflate(R.layout.fragment_tab_a, container, false);
 return v;
 }
}

```
- Nội dung file class FragmentTabB.java:

```

package com.example.app_10;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
public class FragmentTabB extends Fragment {
 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container,
 Bundle savedInstanceState)
 { View v = inflater.inflate(R.layout.fragment_tab_b, container, false);
 return v;
 }
}

```

B4.- Copy và paste file `src\com.example.app_10\MainActivity.java` để có file `src\com.example.app_10\SwitchFragmentsWithTabs.java`. Chỉnh sửa nội dung file `src\com.example.app_10\SwitchFragmentsWithTabs.java` để có nội dung như sau:

```

package com.example.app_10;
import android.app.ActionBar;
import android.app.Activity;
import android.app.FragmentTransaction;
import android.app.ActionBar.Tab;
import android.os.Bundle;
import android.view.Menu;

public class SwitchFragmentsWithTabs extends Activity
{
 Tab mTab1, mTab2;
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.switch_fragments_with_tab);
 ActionBar actionBar = getActionBar();
 actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
 mTab1= actionBar.newTab().setText("Fragment A").setTabListener(new
 NavTabListener());
 mTab2= actionBar.newTab().setText("Fragment B").setTabListener(new
 NavTabListener());
 actionBar.addTab(mTab1);
 actionBar.addTab(mTab2);

 showFragmentA();
 }
 public void showFragmentA(){
 FragmentTabA fragmentA = new FragmentTabA();
 FragmentTransaction ft = getFragmentManager().beginTransaction();
 ft.replace(R.id.Layout_container_tab, fragmentA);
 ft.addToBackStack("fragment a");
 ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
 ft.commit();
 }
 public void showFragmentB(){
 FragmentTabB fragmentB = new FragmentTabB();
 FragmentTransaction ft = getFragmentManager().beginTransaction();
 ft.replace(R.id.Layout_container_tab, fragmentB);
 ft.addToBackStack("fragment b");
 ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
 ft.commit();
 }
 @Override
 public boolean onCreateOptionsMenu(Menu menu) {
 //Inflate the menu; this adds items to the action bar if it is present.
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }
 private class NavTabListener implements ActionBar.TabListener
 {
 public NavTabListener() {
 }
 @Override
 public void onTabReselected(Tab tab, FragmentTransaction ft) {
 }
 @Override
 public void onTabSelected(Tab tab, FragmentTransaction ft)
 {//Kiểm tra khi click chọn trên Tab để gọi phương thức tương ứng
 if (tab.equals(mTab1))

```

```

 showFragmentA();
 else
 showFragmentB();
}
@Override
public void onTabUnselected(Tab tab, FragmentTransaction ft) {
}
}
}

```

**B5.-** Khai báo activity *SwitchFragmentsWithTabs* trong file *AndroidManifest.xml*.

**B6.-** Bổ sung lệnh cho file *MainActivity.java* để khi người dùng click chọn button “*Di chuyển giữa các Fragment thông qua Tab*” sẽ mở ra activity *SwitchFragmentsWithTabs*.

**B7.-** Chạy ứng dụng để xem kết quả.

### 2.4.3. Tương tác giữa Fragment và Activity

Qua phần trên, bạn đã thấy fragment có thể gọi 1 phương thức có trong activity. Bạn có thể tiếp tục thêm 1 bước khi bạn tạo fragment và yêu cầu phương thức riêng thực thi việc gọi activity. Để thực hiện được, fragment cần thực thi giao diện (interface) cho activity thực hiện. Fragment bắt buộc thực hiện yêu cầu này bằng việc kiểm tra sự tồn tại của giao diện cần thực thi.

## BÀI THỰC HÀNH App\_10 (tiếp theo)

### ☞ Yêu cầu:



Hình 2-60 Màn hình ban đầu

Hình 2-61 Màn hình khi click chọn Yes hoặc No

- Tạo mới 1 activity *InteractionFragmentActivity* để khi người dùng click chọn button “Tương tác giữa Fragment và Activity” sẽ mở ra activity này.
- Activity *InteractionFragmentActivity* gồm 1 fragment xuất hiện ở góc trái trên của activity với nội dung gồm 1 textview và 2 button (Yes và no). Khi người dùng click chọn button nào sẽ xuất hiện thông báo việc chọn lựa(dùng Toast) trên activity.

### ☞ Thực hiện

**B1.-** Tạo mới 2 file layout là *InteractionFragmentActivity.xml* và *fragment\_yes\_no.xml*.

- Nội dung file *InteractionFragmentActivity.xml*

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 <LinearLayout android:id="@+id/activity_fragment_layout"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_alignParentLeft="true"
 android:layout_alignParentTop="true"
 android:orientation="horizontal">
 <fragment android:id="@+id/fragment"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"/>
 </LinearLayout>
</RelativeLayout>

```
- Nội dung file *fragment\_yes\_no.xml*

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:background="#33777777"
 android:paddingBottom="10dp"
 android:paddingLeft="20dp" android:paddingRight="20dp">
 <TextView android:id="@+id/text"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Question ..."/>
 <Button android:id="@+id/yes_button"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Yes"/>
 <Button android:id="@+id/no_button"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="No"/>

```

```

<TextView android:id="@+id/textQuestion"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:paddingBottom="12dp"
 android:paddingLeft="10dp"
 android:paddingRight="10dp"
 android:text="Question ..."
 android:textAppearance="?android:attr/textAppearanceLarge">
</TextView>
<Button android:id="@+id/buttonYes"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_below="@+id/textQuestion"
 android:layout_centerInParent="false"
 android:text="Yes"
 android:textStyle="bold"
 android:textColor="#ff0000"/>
<Button android:id="@+id/buttonNo"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_below="@+id/textQuestion"
 android:layout_toRightOf="@+id/buttonYes"
 android:text="No"
 android:textStyle="bold"
 android:textColor="#ff0000"/>
</RelativeLayout>

```

**B2.-** Copy 1 file fragmentXXX.java (file nào cũng được) và paste vào folder src\com.example.app\_10 rồi đổi tên thành FragmentYesNo.java, chỉnh sửa nội dung để có kết quả như sau:

```

package com.example.app_10;
import android.app.Activity;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.View.OnClickListener;
import android.widget.Button;
public class FragmentYesNo extends Fragment
{ public interface OnAnswerSelectedListener
{
 public void OnAnswerSelected(String answer);
}
OnAnswerSelectedListener mListener;

@Override
public void onAttach(Activity activity) {
 super.onAttach(activity);
 try
 {
 mListener = (OnAnswerSelectedListener) activity;
 }
 catch (ClassCastException e)
 {
 throw new ClassCastException(activity.toString() + " must
 implement OnAnswerSelectedListener");
 }
}
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
 Bundle savedInstanceState) {
 View v = inflater.inflate(R.layout.fragment_yes_no, container, false);
 Button buttonYes = (Button) v.findViewById(R.id.buttonYes);

```

```

 buttonYes.setOnClickListener(new OnClickListener()
 {
 public void onClick(View v)
 { mListener.OnAnswerSelected("yes");
 }
 });
 Button buttonNo = (Button) v.findViewById(R.id.buttonNo);
 buttonNo.setOnClickListener(new OnClickListener()
 {
 public void onClick(View v)
 { mListener.OnAnswerSelected("no");
 }
 });
 return v;
}
}

```

**B3.-** Copy file MainActivity.java và paste vào folder src\com.example.app\_10 rồi đổi tên thành InteractionFragmentActivity.java, chỉnh sửa nội dung để có kết quả như sau:

```

package com.example.app_10;
import com.example.app_10.InteractionFragmentActivity;
import android.app.Activity;
import android.app.FragmentTransaction;
import android.os.Bundle;
import android.view.Menu;
import android.widget.Toast;

public class InteractionFragmentActivity extends Activity implements
 FragmentYesNo.OnAnswerSelectedListener
{
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.interaction_fragment_activity);
 FragmentYesNo fragmentA = new FragmentYesNo();
 FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
 ft.replace(R.id.activity_fragment_layout, fragmentA);
 ft.addToBackStack("yesno");
 ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
 ft.commit();
 }

 @Override
 public boolean onCreateOptionsMenu(Menu menu) {
 //Inflate the menu; this adds items to the action bar if it is present.
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }
 @Override
 public void OnAnswerSelected(String answer) {
 Toast.makeText(getApplicationContext(), "You choose "+
 answer.toUpperCase(), Toast.LENGTH_SHORT).show();
 }
}

```

**B4.-** Khai báo activity InteractionFragmentActivity trong file AndroidManifest.xml.

**B5.-** Viết lệnh cho button cuối cùng trong MainActivity.java để khi người dùng click chọn button “Tương tác giữa Fragment và Activity” sẽ mở ra activity này.

**B6.-** Chạy ứng dụng để xem kết quả.

## 2.5. CÁC DẠNG Dialogs

Một dialog là 1 cửa sổ giúp hiển thị thông tin đến người dùng. Thông tin đó có thể đơn thuần chỉ là thông tin cần thông báo hay yêu cầu 1 quyết định từ phía người dùng hoặc yêu cầu cung cấp thêm thông tin bổ sung. Dialog thường được dùng dưới dạng modal window, nghĩa là người dùng phải chọn trả lời các thông tin trong dialog trước khi trở lại màn hình chính trước đó. Bạn có thể sử dụng class DialogFragment để tạo ra 1 số loại dialog.

### 2.5.1. Dialog Fragment

#### 2.5.1.1. Hiển thị Dialog

Dialog cho phép tương tác với người dùng mà không ảnh hưởng đến ngữ cảnh đang có. Khi 1 dialog hiển thị, chúng thường chứa những thông tin ngắn gọn hoặc mời người dùng chọn 1 trong những chọn lựa được đề xuất hoặc quyết định 1 vài vấn đề nào đó

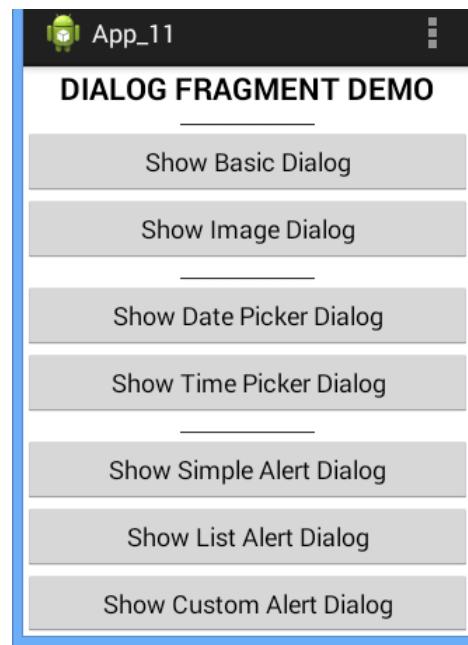
Dialog fragments được giới thiệu từ phiên bản Honeycomb (API level 11, Android 3.0) của Android, và cũng cung cấp sẵn trong các gói hỗ trợ (support package). Các gói hỗ trợ giúp class DialogFragment tương thích với các version trước đó của Android.

Trước khi class DialogFragment được giới thiệu, người lập trình dùng class Dialogs để tạo và hiển thị các modal window. Với khả năng được cung cấp trong class DialogFragment, việc sử dụng trực tiếp các dialog được khuyên không nên sử dụng.

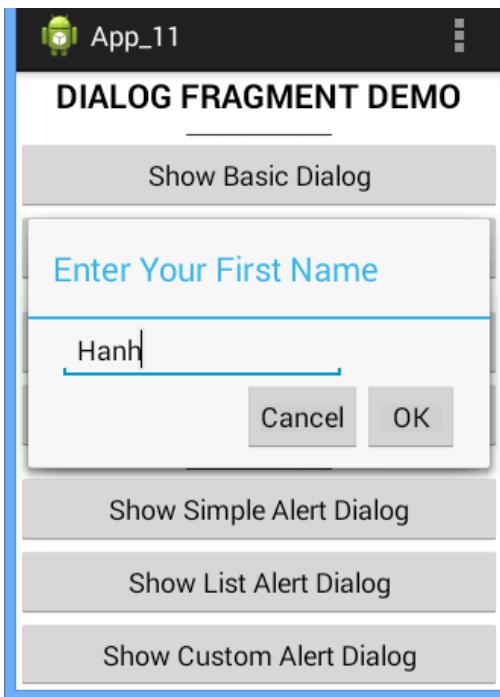
### BÀI THỰC HÀNH App\_11 (tiếp theo)

#### Yêu cầu 1:

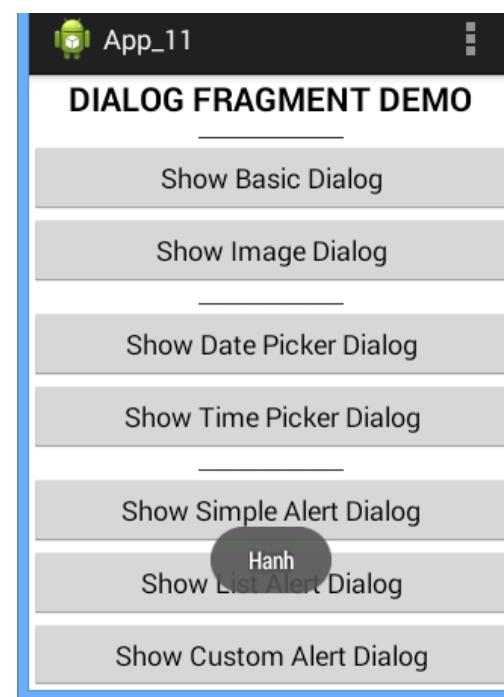
- Tạo mới 1 activity *App\_11* với giao diện như hình minh họa (tạm gọi là *MainForm*)
- Tạo tiếp 1 class kế thừa từ class DialogFragment. Khi người dùng click button đầu tiên “Show Basic Dialog” trên *MainForm* sẽ xuất hiện Dialog như hình 2-62 trong thời gian nhất định (VD 10 giây) nếu người dùng nhập tên và click OK sẽ cho xuất hiện tên trên màn hình (hình 2-63) ngược lại nếu nhấn Cancel hoặc quá thời gian quy định sẽ không xuất hiện gì cả. Sau khi người dùng click chọn, hoặc quá thời gian quy định Dialog sẽ tự tắt đi.



Hình 2-62 Màn hình ban đầu (MainForm)



Hình 2-63 Màn hình khi đang nhập tên



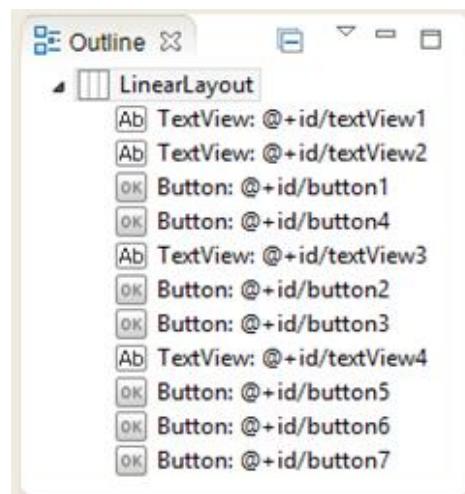
Hình 2-64 Màn hình sau khi click chọn OK

### ☛ Thực hiện

**B1.-** Tạo mới 2 file layout là *activity\_main.xml* và *fragment\_dialog\_name.xml*.

- Nội dung file *activity\_main.xml*

```
<LinearLayout android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 xmlns:android="http://schemas.android.com/apk/res/android">
 <TextView android:id="@+id/textView1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="DIALOG FRAGMENT DEMO"
 android:layout_gravity="center"
 android:textStyle="bold"
 android:textAppearance=
 "?android:attr/textAppearanceLarge"/>
 <TextView android:id="@+id/textView2"
 android:layout_width="wrap_content" Hình 2-65
 android:layout_height="wrap_content"
 android:text="_____"
 android:layout_gravity="center"
 android:textStyle="bold"
 android:textSize="10sp" />
 <Button android:id="@+id/button1"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:height="15dp"
 android:text="Show Basic Dialog"/>
 <Button android:id="@+id/button4"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Show Image Dialog"/>
 <TextView android:id="@+id/textView3"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="_____"
 android:layout_gravity="center"
 android:textStyle="bold"/>
```



Hình 2-64: Layout của MainForm

```

 android:textSize="10sp" />
 <Button android:id="@+id/button2"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Show Date Picker Dialog"/>
 <Button android:id="@+id/button3"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Show Time Picker Dialog"/>
 <TextView android:id="@+id/textView4"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="_____"
 android:layout_gravity="center"
 android:textStyle="bold"
 android:textSize="10sp" />
 <Button android:id="@+id/button5"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Show Simple Alert Dialog"/>
 <Button android:id="@+id/button6"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Show List Alert Dialog"/>
 <Button android:id="@+id/button7"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Show Custom Alert Dialog"/>
</LinearLayout>
```

- Nội dung file fragment\_dialog\_name.xml

```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:paddingBottom="10dp"
 android:paddingLeft="20dp"
 android:paddingRight="20dp">
 <EditText android:id="@+id/editText1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:hint="Enter your first name"
 android:inputType="textPersonName"/>
 <Button android:id="@+id/buttonCancel"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_below="@+id/editText1"
 android:layout_toLeftOf="@+id/buttonOK"
 android:text="Cancel"/>
 <Button android:id="@+id/buttonOK"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_alignBottom="@+id/buttonCancel"
 android:layout_alignParentRight="true"
 android:text="OK"/>
</RelativeLayout>
```

- B2.-** Tạo mới file mã lệnh cho dialogFragment đầu tiên của chương trình và đặt tên là *BasicDialogFragment.java*.

```

package com.example.app_11;
import java.util.Timer;
import java.util.TimerTask;
import android.app.Activity;
import android.app.DialogFragment;
```

```

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;

public class BasicDialogFragment extends DialogFragment
{
 public interface OnNameEnteredListener
 {
 public void OnNameEntered(String nameEntered);
 }
 OnNameEnteredListener mListener;
 @Override
 public void onAttach(Activity activity)
 {
 super.onAttach(activity);
 try
 {
 mListener = (OnNameEnteredListener) activity;
 }
 catch (ClassCastException e)
 {
 throw new ClassCastException(activity.toString() + " must implement
 OnNameEnteredListener");
 }
 }
 Button mOK;
 Button mCancel;
 EditText mNameField;
 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
 savedInstanceState)
 { // gọi layout đã được xây dựng trước đó cho dialog
 View v = inflater.inflate(R.layout.fragment_dialog_name, container, false);
 // gọi phương thức setTitle của class Dialog để gán Title cho dialog
 getDialog().setTitle("First Name");
 /* Nếu không sử dụng Title: gọi ph/thúc requestWindowFeature của class Dialog*/
 // getDialog().requestWindowFeature(Window.FEATURE_NO_TITLE);
 /*Để tìm hiểu thêm về các phương thức khác, có thể truy cập link
 http://developer.android.com/reference/android/app/Dialog.html*/
 mNameField = (EditText)v.findViewById(R.id.editText1);
 mOK = (Button)v.findViewById(R.id.buttonOK);
 mOK.setOnClickListener(new OnClickListener()
 {
 public void onClick(View v)
 {
 mListener.OnNameEntered(mNameField.getText().toString());
 BasicDialogFragment.this.dismiss();
 }
 });
 mCancel = (Button)v.findViewById(R.id.buttonCancel);
 mCancel.setOnClickListener(new OnClickListener()
 {
 public void onClick(View v)
 {
 BasicDialogFragment.this.dismiss();
 }
 });
 }

 final Timer t = new Timer();
 t.schedule(new TimerTask()
 {
 public void run()
 {
 // when the task active then close the dialog
 BasicDialogFragment.this.dismiss();
 // also just stop the timer thread, otherwise, you may receive a crash report
 t.cancel();
 }
 });
}

```

```

 }
 }, 10000); // after 2 second (or 2000 milliseconds), the task will be active.
 return v;
}
}

```

**B3.-** Hiệu chỉnh nội dung file *MainActivity.java* để có kết quả như sau:

```

package com.example.app_11;
import com.example.app_11.R;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity implements
 BasicDialogFragment.OnNameEnteredListener,
 CustomAlertDialogFragment.OnNameEnteredListener,
 DatePickerDialogFragment.OnDateEnteredListener,
 TimePickerDialogFragment.OnTimeEnteredListener
{
 @Override
 protected void onCreate(Bundle savedInstanceState)
 { super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 Button button1 = (Button) findViewById(R.id.button1);
 button1.setOnClickListener(new OnClickListener()
 { public void onClick(View v)
 { BasicDialogFragment basicDialog = new BasicDialogFragment();
 basicDialog.show(getFragmentManager(), "basic");
 }
 });
 }
 @Override
 public boolean onCreateOptionsMenu(Menu menu) {
 // Inflate the menu; this adds items to the action bar if it is present.
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }
 @Override
 public void OnNameEntered(String answer) {
 Toast.makeText(getApplicationContext(), answer, Toast.LENGTH_SHORT).show();
 }
}

```

**B4.-** Chạy chương trình để kiểm tra kết quả.

### 2.5.1.2. Sử dụng Dialog

Để mở một dialog, đơn giản là bạn cần tạo và hiển thị 1 fragment. Trong activity hoặc fragment, bạn khai báo 1 đối tượng thuộc class *BasicDialogFragment*, sau đó gọi phương thức *show* để hiển thị.

```

BasicDialogFragment basicDialog = new BasicDialogFragment();
basicDialog.show(getFragmentManager(), "basic");

```

Trước khi dialog được mở, có thể thiết lập style cho dialog bằng cách gán style cho dialog bằng 1 style đã được định nghĩa trước đó hoặc 1 style do bạn tự tạo.

Trong minh họa sau, hình bên trái sử dụng style mặc định của Android. Khi hiển thị dialog này bao gồm tiêu đề, 1 hình nền bị làm mờ để che lấp các phần tử ở phía sau cửa sổ.

Dialog bên phải sẽ hiển thị dưới dạng không có tiêu đề, không có khung và nền không bị làm mờ. Đây là style có tên *NoFrame* (đã được khai báo trong file res\values\style.xml) và được thiết lập cho dialog như sau:

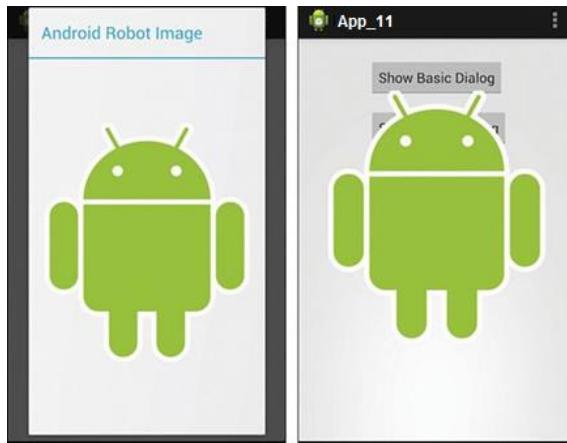
```
ImageDialogFragment imageDialog = new ImageDialogFragment();
imageDialog.setStyle(DialogFragment.STYLE_NO_FRAME, 0);
imageDialog.show(getFragmentManager(), "image");
```

Khi đó, nội dung file res\values-v11\style.xml cần được bổ sung để có nội dung như sau:

```
<resources>
 <!-- Base application theme for API 11+. This theme completely replaces
 AppBaseTheme from res/values/styles.xml on API 11+ devices. -->

 <style name="AppBaseTheme" parent="Theme.AppCompat.Light">
 <!-- API 11 theme customizations can go here. -->
 </style>

 <style name="NoFrame">
 <item name="android:windowBackground">@android:color/transparent</item>
 <item name="android:windowFrame">@null</item>
 <item name="android:windowContentOverlay">@null</item>
 <item name="android:windowAnimationStyle">@null</item>
 <item name="android:backgroundDimEnabled">false</item>
 <item name="android:windowIsTranslucent">true</item>
 <item name="android:windowNoTitle">true</item>
 <item name="android:windowCloseOnTouchOutside">false</item>
 </style>
</resources>
```



Hình 2-66 Sử dụng style mặc định (bên trái) và sử dụng style tự định nghĩa (bên phải)

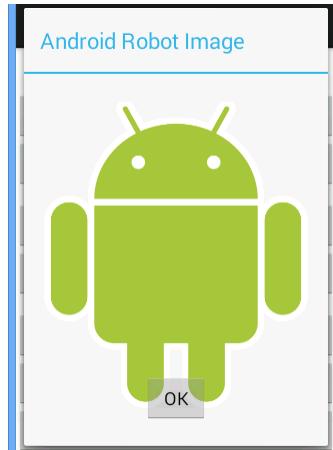
## BÀI THỰC HÀNH App\_11 (tiếp theo)

### ☞ Yêu cầu 2:

- Tạo tiếp 1 class kế thừa từ class DialogFragment.  
Khi người dùng click button đầu tiên “Show Image Dialog” trên MainForm sẽ xuất hiện Dialog trong đó có chứa hình Android Robot (có thể tùy chọn) như hình 2-90. Nhấn phím Esc trên



bàn phím hoặc phím Back ( ) trên emulator để trở về màn hình chính.



Hình 2-67 Màn hình ban đầu (MainForm)

 **Thực hiện**

**B5.-** Tìm và copy file hình có tên là android\_robot.jpg (có thể sử dụng hình ảnh khác) vào file folder res\drawable-mdpi.

**B6.-** Tạo mới file layout là fragment\_dialog\_image.xml.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:paddingBottom="10dp"
 android:paddingLeft="20dp"
 android:paddingRight="20dp">
 <ImageView android:id="@+id/imageView1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_alignParentTop="true"
 android:layout_centerHorizontal="true"
 android:layout_centerInParent="true"
 android:src="@drawable/android_robot"/>
</RelativeLayout>
```

**B7.-** Tạo mới file mã lệnh cho layout vừa tạo với tên ImageDialogFragment.java với nội dung

```
package com.example.app_11;
import com.example.app_11.R;
import android.app.DialogFragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.Button;
public class ImageDialogFragment extends DialogFragment
{
 Button btn;
 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container,
 Bundle savedInstanceState)
 { View v = inflater.inflate(R.layout.fragment_dialog_image, container, false);
 getDialog().setTitle("Android Robot Image");
 btn = (Button)v.findViewById(R.id.button1);
 btn.setOnClickListener(new OnClickListener()
 {
 public void onClick(View v)
 {
 ImageDialogFragment.this.dismiss();
 }
 });
 return v;
 }
}
```

**B8.-** Bổ sung mã lệnh vào file MainActivity.java để có nội dung như sau:

```
package com.example.app_11;
import com.example.app_11.R;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;
public class MainActivity extends Activity implements
 BasicDialogFragment.OnNameEnteredListener,
 CustomAlertDialogFragment.OnNameEnteredListener,
 DatePickerDialogFragment.OnDateEnteredListener,
 TimePickerDialogFragment.OnTimeEnteredListener
{
```

```

@Override
protected void onCreate(Bundle savedInstanceState)
{ super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 Button button1 = (Button)findViewById(R.id.button1);
 button1.setOnClickListener(new OnClickListener()
 { public void onClick(View v)
 { BasicDialogFragment basicDialog = new BasicDialogFragment();
 basicDialog.show(getFragmentManager(), "basic");
 }
 });
 Button button4 = (Button)findViewById(R.id.button4);
 button4.setOnClickListener(new OnClickListener()
 { public void onClick(View v)
 { ImageDialogFragment imageDialog = new ImageDialogFragment();
 //Khi muốn dùng style riêng, bổ sung lệnh liền ngay sau đây
 //imageDialog.setStyle(DialogFragment.STYLE_NO_FRAME, 0);
 imageDialog.show(getFragmentManager(), "image");
 }
 });
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
 // Inflate the menu; this adds items to the action bar if it is present.
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
}
@Override
public void OnNameEntered(String answer) {
 Toast.makeText(getApplicationContext(), answer, Toast.LENGTH_SHORT).show();
}
}

```

**B9.-** Chạy chương trình để xem kết quả.

## 2.5.2. Dialogs dùng cho việc chọn ngày và giờ

Dialog gồm các loại:

- *DatePicker*: Giao diện để chọn ngày.
- *TimePicker*: Giao diện để chọn giờ.
- *AlertDialog*: Có thể được dùng cho 1 số loại dialog.
- *ProgressDialog*: class này được dùng để mở 1 cửa sổ nhỏ, trong đó hiển thị 1 progress bar không xác định. Dialog dạng này được khuyên không nên sử dụng mà thay vào đó nên đưa ProgressBar trực tiếp vào layout

### 2.5.2.1. Sử dụng Date Picker

Để sử dụng dialog *DatePicker*, bạn bao dialog này vào trong 1 dialog fragment bằng cách sử dụng phương thức *onCreateDialog()* của dialog fragment. Trong đó, định nghĩa dialog và truyền các tham số theo dạng sau:

```
DatePickerDialog(Context context, DatePickerDialog.OnDateSetListener callBack,
 int year, int monthOfYear, int dayOfMonth)
```

Trong đó, bạn có thể sử dụng phương thức *onDateSet()* cho tham số *DatePickerDialog.OnDateSetListener*. Các tham số năm, tháng, ngày được dùng để khởi tạo giá trị cho dialog.

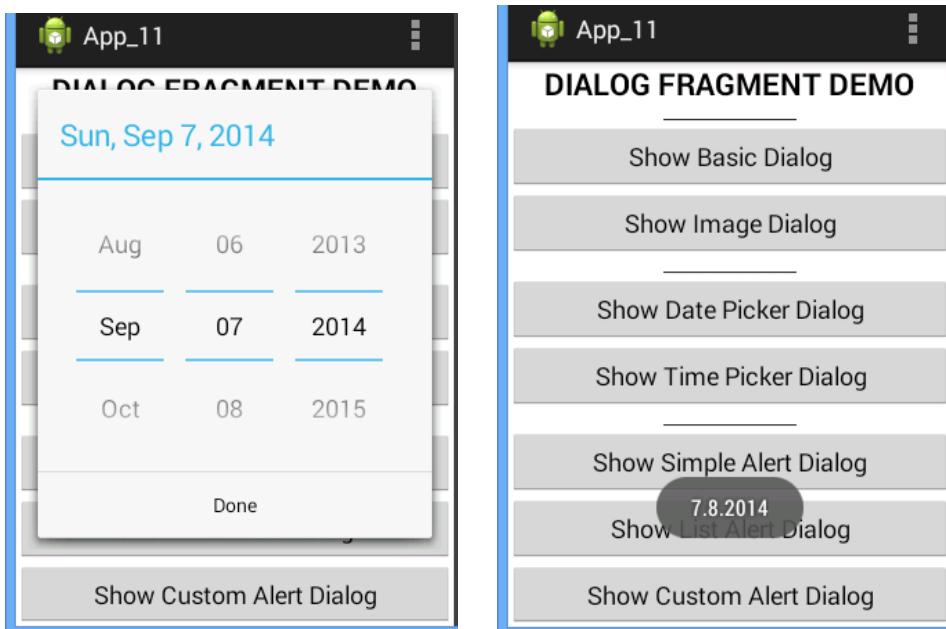
Các bước cần thực hiện để tạo *DateDialogFragment*:

- (i). Tạo class *DateDialogFragment* mở rộng từ *DialogFragment* và thực thi *DatePickerDialog.onDateSetListener()*.
- (ii). Thực thi phương thức *onDateSet()* trong *DialogFragment*.

(iii). Tạo 1 DatePickerDialog trong phương thức onCreateDialog().

## BÀI THỰC HÀNH App\_11 (tiếp theo)

☞ Yêu cầu 3:



Hình 2-68 Khi **DatePickerDialogFragment** được hiển thị và sau khi chọn nút done

☞ Tạo tiếp 1 class DatePickerDialogFragment extends từ DialogFragment. Trong đó khai báo và cho hiển thị 1 đối tượng DatePickerDialog. Khi người dùng click trên button “Show Date Picker Dialog” sẽ cho hiển thị dialog này. Sau đó, khi người dùng click chọn button Done trên DatePickerDialogFragment sẽ đóng dialog và cho hiển thị ngày vừa được chọn bằng Toast.

☞ Thực hiện

**B10.-** Bổ sung phương thức OnDateEntered vào src\com.example.app\_11\MainActivity.java để sau khi đóng DatePickerDialog sẽ cho hiển thị ngày vừa được chọn bằng Toast.

```
@Override
public void OnDateEntered(int year, int monthOfYear, int dayOfMonth)
{
 Toast.makeText(getApplicationContext(), dayOfMonth + "." + monthOfYear + "." +
 year, Toast.LENGTH_SHORT).show();
}
```

**B11.-** Copy file src\com.example.app\_11\BasicDialogFragment.java rồi paste vào src\com.example.app\_11 và đổi tên thành DatePickerDialogFragment.java. Chỉnh sửa để có nội dung như sau:

```
package com.example.app_11;
import java.util.Calendar;
import android.os.Bundle;
import android.widget.DatePicker;
import android.app.Activity;
import android.app.DatePickerDialog;
import android.app.Dialog;
import android.app.DialogFragment;
public class DatePickerDialogFragment extends DialogFragment implements
 DatePickerDialog.OnDateSetListener
{ /* Sử dụng OnDateEnteredListener để thực thi phương thức OnDateEntered trong
 MainActivity*/
 interface OnDateEnteredListener
 { public void OnDateEntered(int year, int monthOfYear, int dayOfMonth);
 }
 OnDateEnteredListener mListener;
```

```

// Get current date and create the DatePickerDialog
Calendar now = Calendar.getInstance();
@Override
public Dialog onCreateDialog(Bundle savedInstanceState)
{ DatePickerDialog dateDialog = new DatePickerDialog(this.getActivity(), this,
 now.get(Calendar.YEAR), now.get(Calendar.MONTH),now.get(Calendar.DAY_OF_MONTH));
return dateDialog;
}
// OnDateSet is required to implement DatePickerDialog.OnDateSetListener
@Override
public void onDateSet(DatePicker view, int year, int monthOfYear,int dayOfMonth)
{ mListener.OnDateEntered(year, monthOfYear, dayOfMonth);
}
@Override
public void onAttach(Activity activity)
{
 super.onAttach(activity);
 /* định nghĩa 1 interface listener để giao tiếp với activity, giá trị của
 listener này sẽ được gán giá trị nhờ phương thức onDateSet ở trên */
 try
 { mListener = (OnDateEnteredListener) activity;
 }
 catch (ClassCastException e)
 { throw new ClassCastException(activity.toString() +
 " must implement OnDateEnteredListener");
 }
}
}
}

```

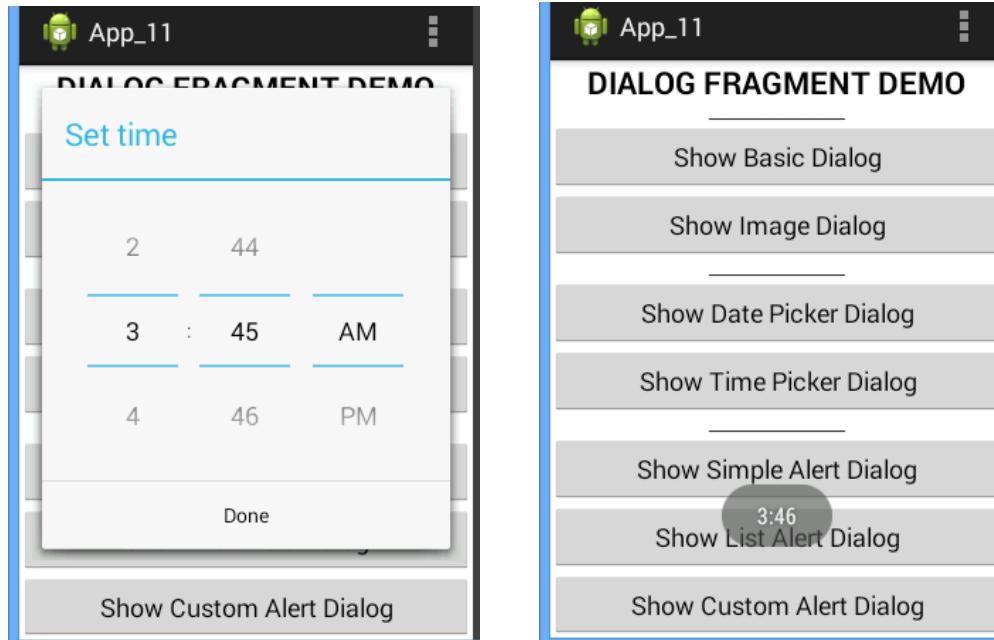
**B12.-** Chạy chương trình để xem kết quả.

### 2.5.2.2. Sử dụng Time Picker

Kỹ thuật sử dụng TimePicker tương tự như khi sử dụng DatePicker.

## BÀI THỰC HÀNH App\_11 (tiếp theo)

### ☞ Yêu cầu 4:



Hình 2-69 Khi TimePickerDialogFragment được hiển thị và sau khi chọn nút done

- Tạo tiếp 1 class TimePickerDialogFragment extends từ DialogFragment. Trong đó khai báo và cho hiển thị 1 đối tượng TimePickerDialog. Khi người dùng click trên button “Show Time

“Picker Dialog” sẽ cho hiển thị dialog này. Sau đó, khi người dùng click chọn button *Done* trên *TimePickerDialogFragment* sẽ đóng dialog và cho hiển thị ngày vừa được chọn bằng Toast.

### ☞ Thực hiện

**B13.-** Bổ sung phương thức *OnTimeEntered* vào *src\com.example.app\_11\MainActivity.java* để sau khi đóng *TimePickerDialog* sẽ cho hiển thị ngày vừa được chọn bằng Toast.

```
@Override
public void OnTimeEntered(int hour, int minute)
{
 Toast.makeText(getApplicationContext(), hour + ":" + minute ,
 Toast.LENGTH_SHORT).show();
}
```

**B14.-** Copy file *src\com.example.app\_11\DatePickerDialogFragment.java* rồi paste vào *src\com.example.app\_11* và đổi tên thành *TimePickerDialogFragment.java*. Chính sửa để có nội dung như sau:

```
package com.example.app_11;
import java.util.Calendar;
import android.os.Bundle;
import android.widget.TimePicker;
import android.app.Activity;
import android.app.Dialog;
import android.app.DialogFragment;
import android.app.TimePickerDialog;
public class TimePickerDialogFragment extends DialogFragment implements
 TimePickerDialog.OnTimeSetListener
{ public interface OnTimeEnteredListener
 { public void OnTimeEntered(int hour, int minute);
 }

 OnTimeEnteredListener mListener;
 Calendar now = Calendar.getInstance();
 @Override
 public Dialog onCreateDialog(Bundle savedInstanceState)
 { final Calendar c = Calendar.getInstance();
 int hour = c.get(Calendar.HOUR_OF_DAY);
 int minute = c.get(Calendar.MINUTE);
 TimePickerDialog dateDialog = new TimePickerDialog(this.getActivity(),
 this, hour, minute, false);
 return dateDialog;
 }
 @Override
 public void onAttach(Activity activity)
 { super.onAttach(activity);
 try
 { mListener = (OnTimeEnteredListener) activity;
 }
 catch (ClassCastException e)
 { throw new ClassCastException(activity.toString() +
 " must implement OnTimeEnteredListener");
 }
 }
 @Override
 public void onTimeSet(TimePicker view, int hour, int minute)
 { mListener.OnTimeEntered(hour, minute);
 }
}
```

**B15.-** Chạy chương trình để xem kết quả.

### 2.5.3. Sử dụng các Dialogs dạng cảnh báo (Alert dialog)

Một AlertDialog được dùng để thêm tính năng vào 1 dialog dạng đơn giản. Để tạo dialog fragment cho việc chọn ngày và giờ bạn cho thực thi phương thức *onCreateDialog()* và trả về 1 dialog.

Tương tự như vậy, bạn sẽ xây dựng 1 AlertDialog trong class *onCreateDialog()* để tạo mới dialog fragment.

Bạn có thể sử dụng các kỹ thuật tạo các listener để tương tác với việc gọi activity và sử dụng phương thức *onTaach()* để bảo đảm rằng listener đã được định nghĩa thành công kèm theo alert dialog.

Một alert dialog có thể bao gồm thanh tiêu đề, vùng chứa nội dung và có thể chứa đến 3 button để người dùng có thể có nhiều chọn lựa. Đối với 1 dialog dạng đơn giản có thể không có sự phân chia giữa vùng chứa nội dung với các phần khác. Khi đó, thanh tiêu đề có thể chứa thông báo nội dung cần xác nhận đến người dùng và các button sẽ thực hiện các xử lý phản hồi từ người dùng. Lúc này các button đại diện cho phản hồi đồng ý (*positive response*), phản hồi không đồng ý (*negative response*) và phản hồi mang tính trung lập (*neutral response*), ví dụ như dialog gồm 3 button Yes, No, Cancel.

Khi 1 trong các button này được chọn, dialog sẽ được đóng lại. Nếu muốn thu thập dữ liệu từ dialog để chuyển về cho activity đã cho mở dialog, bạn cần tạo ra 1 listener interface để thực hiện việc này.

Các bước để tạo 1 dialog fragment dạng alert dialog:

- (i). Tạo dialog fragment
- (ii). Xây dựng và trả về 1 alert dialog trong phương thức *onCreateDialog()*.
- (iii). Cung cấp 1 listener interface để chuyển kết quả của dialog về cho activity thông qua phương thức *AlertDialog.Builder*. VD:

```
AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
```

### 2.5.3.1. Sử dụng Time Picker

Kỹ thuật sử dụng TimePicker tương tự như khi sử dụng DatePicker.

## BÀI THỰC HÀNH App\_11 (tiếp theo)

### ☞ Yêu cầu 5:

- Tạo tiếp 1 class *BasicAlertDialogFragment* extends từ *DialogFragment*. Trong đó khai báo và cho hiển thị câu thông báo “Do you like dialog?” gồm 3 button Yes, No, Cancel. Khi người dùng click trên button “Show Simple Alert Dialog” sẽ cho hiển thị dialog này.



Hình 2-70 Simple Alert Dialog

### ☞ Thực hiện

**B16.-** Copy file *src\com.example.app\_11\DatePickerDialogFragment.java* rồi paste vào *src\com.example.app\_11* và đổi tên thành *BasicAlertDialogFragment.java*. Chỉnh sửa để có nội dung như sau:

```
package com.example.app_11;
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.DialogFragment;
import android.content.DialogInterface;
import android.os.Bundle;
```

```

public class BasicAlertDialogFragment extends DialogFragment
{
 @Override
 public Dialog onCreateDialog(Bundle savedInstanceState) {
 AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
 // Thiết lập nội dung cần thông báo
 builder.setMessage("Do you like Dialogs?");
 /* Thiết lập nội dung hiển thị trên button ĐỒNG Ý và công việc cần thực hiện
 * khi người dùng click chọn*/
 .setPositiveButton("Yes", new DialogInterface.OnClickListener()
 {
 public void onClick(DialogInterface dialog, int id)
 {
 }
 });
 /* Thiết lập nội dung hiển thị trên button KHÔNG ĐỒNG Ý và công việc cần
 * thực hiện khi người dùng click chọn*/
 .setNegativeButton("No", new DialogInterface.OnClickListener()
 {
 public void onClick(DialogInterface dialog, int id)
 {
 }
 });
 /* Thiết lập nội dung hiển thị trên button TRUNG LẬP và công việc cần thực
 * hiện khi người dùng click chọn*/
 .setNeutralButton("Cancel", new DialogInterface.OnClickListener()
 {
 public void onClick(DialogInterface dialog, int id)
 {
 }
 });
 return builder.create();
 }
}

```

### 2.5.3.2. Hiển thị 1 danh sách các lựa chọn trong alert dialog

- Có thể cho hiển thị 1 danh sách các mục chọn trong 1 alert dialog bằng cách sử dụng phương thức setItems. Danh sách này có thể là:
  - Mảng dữ liệu kiểu chuỗi đã được khai báo trước đó. VD
 

```

String[] items = {"Phở", "Mì", "Nui", "Cháo"};
AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
builder.setTitle("Your choices are:");
/* Phương thức setItems nhận 2 tham số tham số đầu là mảng chứa các chuỗi cần
 hiển thị và tham số thứ 2 là một listener sẽ được gọi khi người dùng chọn 1
 trong các mục */
builder.setItems(items, new DialogInterface.OnClickListener() { ... })

```
  - Mảng dữ liệu kiểu chuỗi đã được khai báo trong file strings.xml. VD
    - Khai báo trong file Strings.xml
 

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
 <string-array name="ThucDon">
 <item>Phở</item>
 <item>Mì</item>
 <item>Nui</item>
 <item>Cháo</item>
 </string-array>
</resources>

```
    - Sử dụng trong chương trình

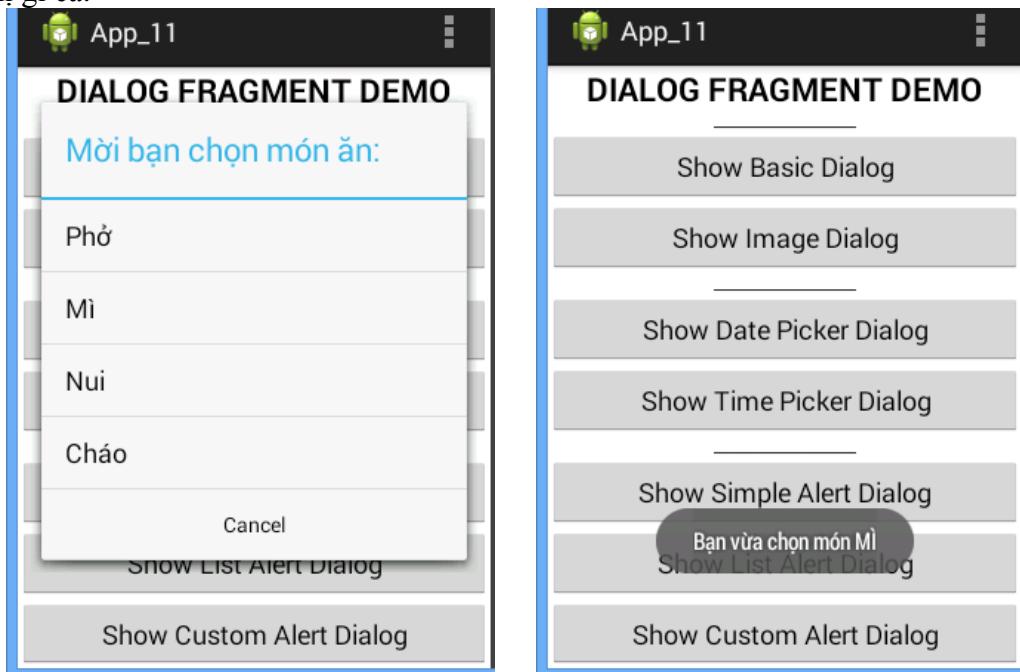
```
AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
builder.setTitle("Your choices are:");
builder.setItems(R.array.ThucDon, new DialogInterface.OnClickListener() { . . . }
```

- Sử dụng `setTitle()` và `setMessage()`
  - `setTitle()` xác định nội dung hiển thị trên thanh tiêu đề của dialog.
  - `setMessage()` xác định nội dung hiển thị trong vùng chứa nội dung của dialog.

## BÀI THỰC HÀNH App\_11 (tiếp theo)

### ☞ Yêu cầu 6:

- Tạo tiếp 1 class `ListAlertDialogFragment` extends từ `DialogFragment`. Trong đó khai báo và cho hiển thị tên 4 món ăn là Phở, mì, nui, cháo và 1 nút “Bỏ qua”. Khi người dùng click trên món ăn nào sẽ hiển thị tên món ăn được chọn, ngược lại nếu chọn nút “Bỏ qua” sẽ không hiển thị gì cả.



Hình 2-71 Simple Alert Dialog

### ☞ Thực hiện

**B17.-** Copy file `src\com.example.app_11\DatePickerDialogFragment.java` rồi paste vào `src\com.example.app_11` và đổi tên thành `ListAlertDialogFragment.java`. Chính sửa để có nội dung như sau:

```
package com.example.app_11;
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.DialogFragment;
import android.content.DialogInterface;
import android.os.Bundle;
import android.widget.Toast;

public class ListAlertDialogFragment extends DialogFragment
{
 @Override
 public Dialog onCreateDialog(Bundle savedInstanceState)
 { final String[] items = {"Phở", "Mì", "Nui", "Cháo"};
 AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
 builder.setTitle("Mời bạn chọn món ăn:");
 builder.setItems(items, new DialogInterface.OnClickListener()
 { public void onClick(DialogInterface dialog, int which)
 { Toast.makeText(getActivity(), "Bạn vừa chọn món "
 +items[which].toUpperCase(), Toast.LENGTH_LONG).show(); }}
```

```

 }
 })
 .setNegativeButton("Cancel", new DialogInterface.OnClickListener()
 {
 public void onClick(DialogInterface dialog, int id)
 {
 }
 });
 return builder.create();
}
}

```

**B18.-** Chạy chương trình để xem kết quả.

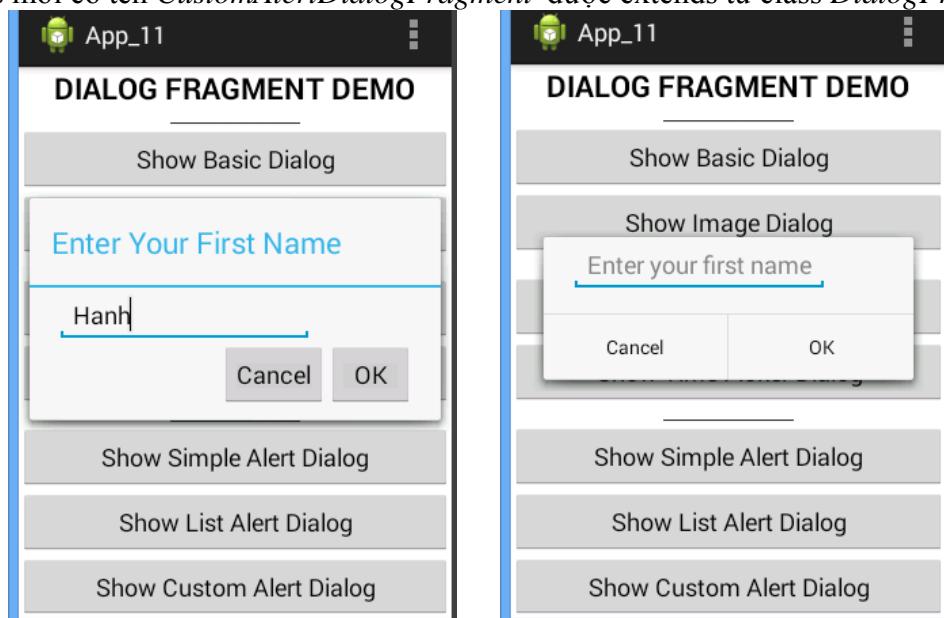
### 2.5.3.3. Adding a Custom View

Trong dialog đầu tiên của bài thực hành App\_11(BasicDialogFragment.java), ta đã tạo ra một layout riêng của mình gồm 1 EditText và 2 button. Khi đó bạn đã sử dụng phương thức onCreateView() chứ không phải là phương thức onCreateDialog(). Bạn có thể thay đổi cách tiếp cận bằng cách sử dụng layout riêng của mình trong 1 alert dialog để cung cấp cùng 1 chức năng như trong dialog đầu tiên. Với cách mới này, bạn chỉ cần cung cấp duy nhất 1 EditText, còn 2 button OK và Cancel là do dialog fragment cung cấp thông qua 2 phản hồi đồng ý và phản hồi không đồng ý.

## BÀI THỰC HÀNH App\_11 (tiếp theo)

### ☞ Yêu cầu 7:

- Có giao diện như dialog đầu tiên của App\_11(BasicDialogFragment.java), nhưng khai báo 1 class mới có tên *CustomAlertDialogFrag* được extends từ class *DialogFragment*.



Hình 2-72 Simple Alert Dialog

### ☞ Thực hiện

**B19.-** Copy file *src\com.example.app\_11\DatePickerDialogFragment.java* rồi paste vào *src\com.example.app\_11* và đổi tên thành *ListAlertDialogFragment.java*. Chính sửa để có nội dung như sau:

```

package com.example.app_11;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.DialogFragment;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.widget.EditText;

```

```

public class CustomAlertDialogFragment extends DialogFragment{
 public interface OnNameEnteredListener
 { public void OnNameEntered(String nameEntered);
 }
 OnNameEnteredListener mListener;
 @Override
 public void onAttach(Activity activity)
 { super.onAttach(activity);
 try
 { mListener = (OnNameEnteredListener) activity;
 }
 catch (ClassCastException e)
 { throw new ClassCastException(activity.toString()
 + " must implement OnNameEnteredListener");
 }
 }
 @Override
 public Dialog onCreateDialog(Bundle savedInstanceState)
 { AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
 LayoutInflater inflater = getActivity().getLayoutInflater();
 builder.setView(inflater.inflate(R.layout.fragment_alert_dialog_name, null))
 .setPositiveButton("OK", new DialogInterface.OnClickListener()
 { public void onClick(DialogInterface dialog, int id)
 { EditText nameText = (EditText)
 dialog.findViewById(R.id.editText1);
 mListener.OnNameEntered(nameText.getText().toString());
 }
 })
 .setNegativeButton("Cancel", new DialogInterface.OnClickListener()
 { public void onClick(DialogInterface dialog, int id)
 {
 }
 });
 return builder.create();
 }
}

```

**B20.-** Chạy chương trình để xem kết quả.

## 2.6. LISTS, GRIDS, GALLERIES, & FLIPPERS

Để thực hiện bài thực hành trong phần này, bạn cần tạo 1 mảng dữ liệu kiểu chuỗi có tên là pie\_array bằng cách khai báo thêm nội dung sau đây trong file res\values\Strings.xml

```

<string-array name="pie_array">
 <item>apple</item>
 <item>blueberry</item>
 <item>cherry</item>
 <item>coconut cream</item>
</string-array>

```

### 2.6.1. ListFragments

*ListFragment(android.app.ListFragment)* là 1 class fragment chuyên biệt, cho phép làm việc với 1 danh sách các mục chọn. Trong phần 2.2.3.2, bạn đã làm việc với spinner, spinner hoạt động tương tự như là 1 drop-down menu và cho hiển thị 1 danh sách các mục. Khi thực thi spinner, bạn cần sử dụng adapter để gắn dữ liệu cần hiển thị trên spinner. Đối với ListFragment, cách thực hiện cũng sẽ tương tự như vậy.

### 2.6.1.1. Tạo 1 ListFragment đơn giản

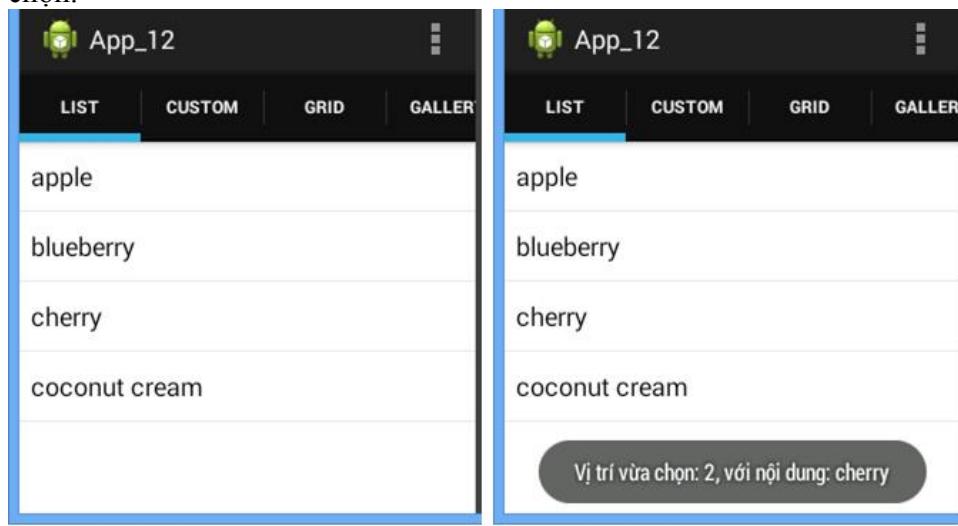
Để tạo 1 *ListFragment* đơn giản, bạn không cần phải tạo layout riêng cho chúng. Một *ListView* (`android.widget.ListView`) được kế thừa từ class *ListFragment*. Nói cách khác, 1 *ListView* được tạo lập bởi hệ thống trong mỗi lần bạn sử dụng *ListFragment*. Không cần tạo *ListView* 1 cách rõ ràng trong phương thức `onCreateView()`.

Trong class *ListFragment* có chứa phương thức `onListClick()` để giúp xử lý khi người dùng click chọn 1 mục trong danh sách. Dựa vào đó, người lập trình có thể cho thực hiện công việc mình muốn trong thân phương thức này.

## BÀI THỰC HÀNH App\_12

### ☞ Yêu cầu:

- Sử dụng toàn bộ bài thực hành trên duy nhất 1 activity với 1 action bar. Trên action bar sẽ thiết lập các tab sao cho mỗi tab sẽ hiển thị riêng 1 fragment.
- Khi người dùng nhấn chọn trên bất kỳ item nào, sẽ sử dụng *Toast* để hiển thị vị trí và nội dung vừa chọn.



Hình 2-73 Giao diện của ứng dụng và kết quả khi người dùng click chọn

### ☞ Thực hiện

- B1.-** Chính sửa nội dung file `res\layout\activity_main.xml` để có nội dung như sau:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 <LinearLayout android:id="@+id/Layout_container"
 android:orientation="horizontal"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 </LinearLayout>
</RelativeLayout>
```

- B2.-** Khai báo 1 mảng dữ liệu kiểu chuỗi tên là `pie_array` trong file `res\values\strings.xml`. nội dung file `strings.xml` sau khi bổ sung khai báo mảng như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
 <string name="app_name">App_12</string>
 <string name="hello_world">Hello world!</string>
 <string name="action_settings">Settings</string>

 <string-array name="pie_array">
 <item>apple</item>
 <item>blueberry</item>
 <item>cherry</item>
 <item>coconut cream</item>
 </string-array>
</resources>
```

```
</string-array>
</resources>
```

- B3.-** Copy file `src\com.example.app_12\MainActivity.java` rồi paste vào folder `src\com.example.app_12`. Đổi tên file mới thành `SimpleListFragment.java`. Chính sửa nội dung file mới này để có nội dung như sau:

```
package com.example.app_12;
import android.app.ListFragment;
import android.content.res.Resources;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;
public class SimpleListFragment extends ListFragment
{ String[] mPies;
@Override
public void onActivityCreated(Bundle savedInstanceState)
{ super.onActivityCreated(savedInstanceState);
Resources resources = getResources();
// Gán dữ liệu đã khai báo trước đó trong res\values\strings.xml cho mPies
mPies = resources.getStringArray(R.array.pie_array);
// - Sử dụng ArrayAdapter<String> để khai báo kiểu dữ liệu.
// - layout có tên android.R.layout.simple_list_item_1 do android xây dựng sẵn
setListAdapter(new ArrayAdapter<String>(this.getActivity(),
 android.R.layout.simple_list_item_1, mPies));
}
@Override
public void onListItemClick(ListView l, View v, int position, long id)
{ Toast.makeText(this.getActivity().getApplicationContext(),"Vị trí vừa chọn: " +
 position +", với nội dung: "+ mPies[position], Toast.LENGTH_SHORT).show();
}
}
```

- B4.-** Hiệu chỉnh nội dung file `src\com.example.app_12\MainActivity.java` để có như sau:

```
package com.example.app_12;
import android.app.ActionBar;
import android.app.Activity;
import android.app.FragmentTransaction;
import android.app.ActionBar.Tab;
import android.os.Bundle;
import android.view.Menu;

public class MainActivity extends Activity
{ Tab mTab1, mTab2, mTab3, mTab4, mTab5;
@Override
protected void onCreate(Bundle savedInstanceState)
{ super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

ActionBar actionBar = getActionBar();
actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
mTab1= actionBar.newTab().setText("List").setTabListener(new NavTabListener());
mTab2= actionBar.newTab().setText("Custom").setTabListener(new NavTabListener());
mTab3= actionBar.newTab().setText("Grid").setTabListener(new NavTabListener());
mTab4= actionBar.newTab().setText("Gallery").setTabListener(new NavTabListener());
mTab5= actionBar.newTab().setText("Flipper").setTabListener(new NavTabListener());

actionBar.addTab(mTab1);
actionBar.addTab(mTab2);
actionBar.addTab(mTab3);
actionBar.addTab(mTab4);
actionBar.addTab(mTab5);
```

```

 showList();
 }
 public void showList()
 {
 SimpleListFragment fragmentA = new SimpleListFragment();
 FragmentTransaction ft = getFragmentManager().beginTransaction();
 ft.replace(R.id.layout_container, fragmentA);
 ft.addToBackStack("fragment a");
 ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
 ft.commit();
 }
 @Override
 public void onTabSelected(Tab tab, FragmentTransaction ft)
 {
 if (tab.equals(mTab1))
 showList();
 }
 @Override
 public void onTabUnselected(Tab tab, FragmentTransaction ft)
 {
 }
 public boolean onCreateOptionsMenu(Menu menu)
 {
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }
}

```

**B5.-** Chạy chương trình để xem kết quả.

### 2.6.1.2. Tùy biến một ListFragment

Trong ListFragment cơ bản đã thực hiện trong phần đầu của bài thực hành App\_12, bạn đã không sử dụng phương thức onCreateView mà đã sử dụng phương thức *onListItemClick()*, kế thừa từ class ListFragment.

Thay vì thực hiện như cách trên, trong phần này, bạn có thể sử dụng phương thức onCreateView của ListFragment để thực thi 1 ListView của riêng mình.

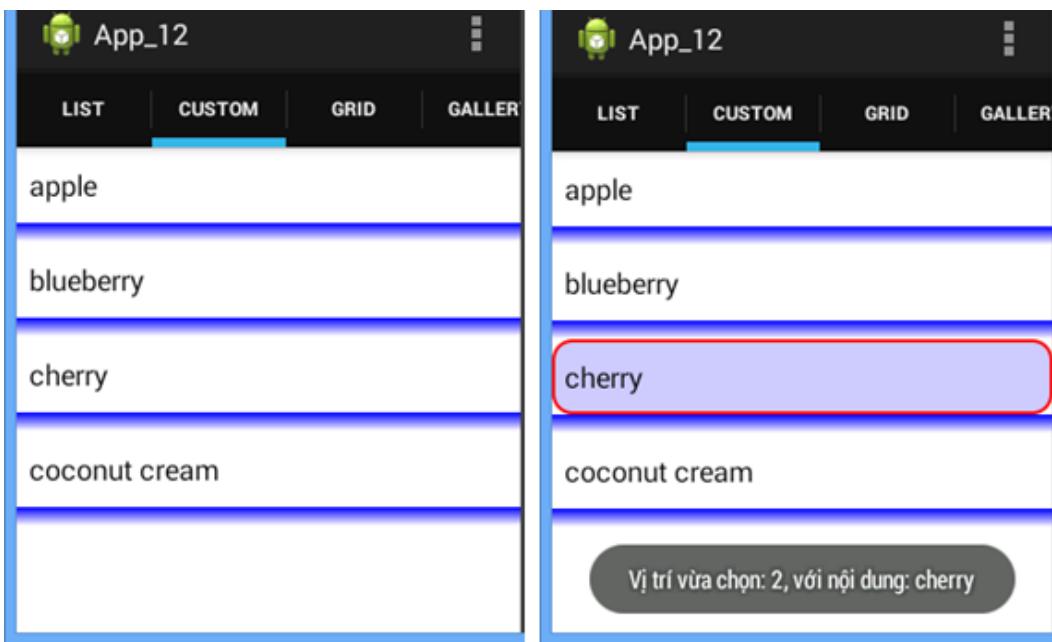
Để sử dụng phương thức onCreateView(), trước tiên bạn cần tạo 1 layout. Khi được sử dụng với ListFragment, layout này cần được định nghĩa 1 cách chính xác với 1 ListView được xác định thuộc tính id như sau:

```
 android:id="@+id/android:list"
```

## BÀI THỰC HÀNH App\_12 (tiếp theo)

### ☞ Yêu cầu 2:

- Tạo 1 danh sách các mục chọn với nội dung tương tự như trong tab “List”. Yêu cầu trang trí đường phân cách giữa các item và mục được chọn theo cách riêng như sau:
  - Đường phân cách: là hình chữ nhật với chiều cao 12dp, màu viền là màu xanh với dạng gradient.
  - Mục chọn: cũng là hình chữ nhật với 4 góc bo tròn, đường viền màu đỏ, nền của mục chọn là màu xanh lợt và hơi trong suốt (#330000FF)
- Khi người dùng nhấn chọn trên bất kỳ item nào, sẽ sử dụng Toast để hiển thị vị trí và nội dung vừa chọn.



Hình 2-74 Giao diện của tab "Custom" và kết quả khi người dùng click chọn

## ☞ Thực hiện

**B6.-** Tạo hình cho đường phân cách và mục chọn:

- Tạo hình cho đường phân cách: tạo mới file res\drawable-mdpi\divider.xml với nội dung:
 

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
 android:shape="rectangle">
 <gradient android:startColor="#0000ff"
 android:endColor="#ffffffff"
 android:angle="270"/>
 <size android:height="12dp"/>
 </shape>
```
- Tạo hình dáng cho mục chọn: tạo mới file res\drawable-mdpi\selector.xml với nội dung:
 

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
 android:shape="rectangle">
 <stroke android:width="2dp"
 android:color="#ff0000"/>
 <corners android:radius="12dp"/>
 <solid android:color="#330000ff"/>
 </shape>
```

**B7.-** Copy file res\layout\activity\_main.xml rồi paste vào folder res\layout với tên mới là custom\_list\_fragment. Chính sửa để có nội dung như sau:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 style="@style/AppTheme"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:background="@style/AppTheme">
 <ListView android:id="@+id/android:list"
 android:layout_width="wrap_content"
 android:layout_height="match_parent"
 android:divider="@drawable/divider" >
 </ListView>
</RelativeLayout>
```

**B8.-** Copy file src\com.example.app\_12\MainActivity.java rồi paste vào folder src\com.example.app\_12 với tên mới là Custom\_List\_Fragment. Chính sửa để có nội dung như sau:

```
package com.example.app_12;
import android.app.ListFragment;
import android.content.res.Resources;
import android.os.Bundle;
```

```

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;
public class CustomListFragment extends ListFragment
{ String[] mPies;
/* Nội dung phương thức onActivityCreated của CustomListFragment
 * và phương thức onActivityCreated của SimpleListFragment giống nhau */
@Override
public void onActivityCreated(Bundle savedInstanceState) {
 super.onActivityCreated(savedInstanceState);
 Resources resources = getResources();
 // Gán dữ liệu đã khai báo trước đó trong res\values\strings.xml cho mPies
 mPies = resources.getStringArray(R.array.pie_array);
 // - Sử dụng ArrayAdapter<String> để khai báo kiểu dữ liệu.
 // - layout có tên android.R.layout.simple_list_item_1 do android xây dựng sẵn
 setListAdapter(new ArrayAdapter<String>(this.getActivity(),
 android.R.layout.simple_list_item_1, mPies));
}
@Override
public void onListItemClick(ListView l, View v, int position, long id)
{ Toast.makeText(this.getActivity().getApplicationContext(),"Vị trí vừa chọn: " +
 position + ", với nội dung: " + mPies[position], Toast.LENGTH_SHORT).show();
}
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
 Bundle savedInstanceState)
{ View v = inflater.inflate(R.layout.custom_list_fragment, container, false);
 return v;
}
}

```

**B9.-** Hiệu chỉnh nội dung file *src\com.example.app\_12\MainActivity.java* để có như sau:

```

package com.example.app_12;
import android.app.ActionBar;
import android.app.Activity;
import android.app.FragmentTransaction;
import android.app.ActionBar.Tab;
import android.os.Bundle;
import android.view.Menu;

public class MainActivity extends Activity
{ Tab mTab1, mTab2, mTab3, mTab4, mTab5;

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

 ActionBar actionBar = getActionBar();
 actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
 mTab1= actionBar.newTab().setText("List").setTabListener(new NavTabListener());
 mTab2= actionBar.newTab().setText("Custom").setTabListener(new NavTabListener());
 mTab3= actionBar.newTab().setText("Grid").setTabListener(new NavTabListener());
 mTab4= actionBar.newTab().setText("Gallery").setTabListener(new NavTabListener());
 mTab5= actionBar.newTab().setText("Flipper").setTabListener(new NavTabListener());

 actionBar.addTab(mTab1);
 actionBar.addTab(mTab2);
 actionBar.addTab(mTab3);
 actionBar.addTab(mTab4);
 }
}

```

```

 actionBar.addTab(mTab5);

 showList();
 }
 public void showList()
 { SimpleListFragment fragmentA = new SimpleListFragment();
 FragmentTransaction ft = getFragmentManager().beginTransaction();
 /* thay thế linearLayout có tên layout_container đã khai báo trong
 file res\layout\activity_main.xml */
 ft.replace(R.id.layout_container, fragmentA);
 ft.addToBackStack("fragment a");
 ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
 ft.commit();
 }
 public void showCustomList()
 { CustomListFragment fragmentE = new CustomListFragment();
 FragmentTransaction ft = getFragmentManager().beginTransaction();
 ft.replace(R.id.layout_container, fragmentE);
 ft.addToBackStack("custom");
 ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
 ft.commit();
 }
 @Override
 public boolean onCreateOptionsMenu(Menu menu)
 { getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }
 private class NavTabListener implements ActionBar.TabListener
 { public NavTabListener()
 {
 }
 @Override
 public void onTabReselected(Tab tab, FragmentTransaction ft)
 {
 }
 @Override
 public void onTabSelected(Tab tab, FragmentTransaction ft)
 { if (tab.equals(mTab1))
 { showList();
 }
 else if (tab.equals (mTab2))
 { showCustomList();
 }
 }
 @Override
 public void onTabUnselected(Tab tab, FragmentTransaction ft)
 {
 }
 }
}

```

**B10.-** Chạy chương trình để xem kết quả

## 2.6.2. Grids và Galleries

Tuy 2 view *GridView*(*android.widget.GridView*) và *Gallery*(*android.widget.Gallery*) không có class đặc trưng như *ListFragment* nhưng bạn vẫn có thể tạo ra fragment hữu dụng với các thành phần.

### 2.6.2.1. Tạo Fragment với GridView

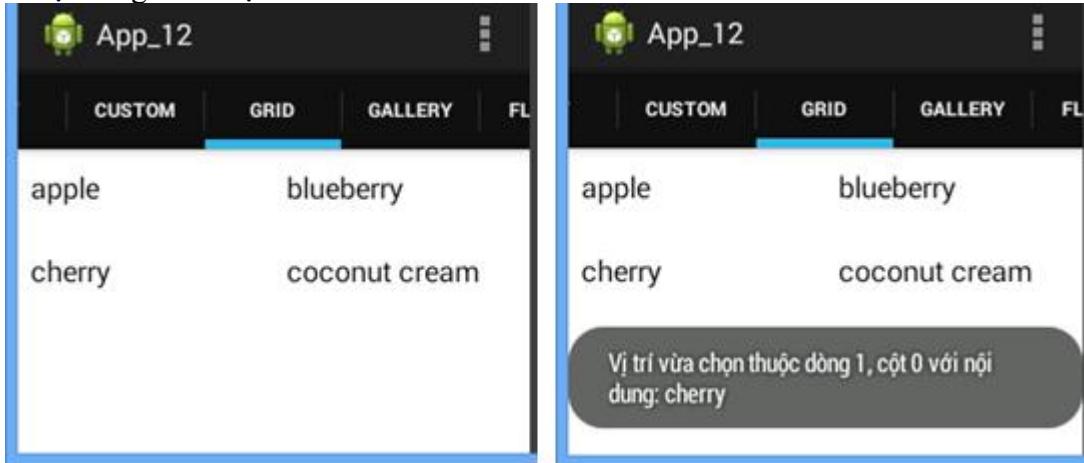
*GridView* hiển thị dữ liệu dưới dạng lưới gồm nhiều dòng và nhiều cột. Để tạo fragment với *GridView*, bạn cần tạo file layout chỉ chứa duy nhất 1 *GridView*. Bạn có thể đặt fragment mới tạo này ở bất cứ đâu trong giao diện của mình. Class fragment sẽ sử dụng layout vừa tạo để định nghĩa 1 view

trong phương thức onCreateView(). Trong phương thức này, dữ liệu sẽ được đưa vào GridView thông qua một adapter.

## BÀI THỰC HÀNH App\_12 (tiếp theo)

### ☞ Yêu cầu 3:

- Tạo 1 danh sách các mục chọn bằng GridView với nội dung tương tự như trong 2 tab “List” và “Custom”.
- Khi người dùng nhấn chọn trên bất kỳ item nào, sẽ sử dụng Toast để hiển thị vị trí (dòng, cột) và nội dung vừa chọn.



Hình 2-75 Giao diện của tab “Grid” và kết quả khi người dùng click chọn

### ☞ Thực hiện

**B11.-** Tạo mới file layout với tên grid\_fragment.xml, trong đó GridView được tổ chức thành 2 cột:

```
<?xml version="1.0" encoding="utf-8"?>
<GridView xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@+id/gridView1"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:numColumns="2"
 android:verticalSpacing="4dp">
</GridView>
```

**B12.-** Tạo mới file layout với tên SimpleGridFragment.java, trong đó GridView được tổ chức thành 2 cột:

```
package com.example.app_12;
import android.app.Fragment;
import android.content.res.Resources;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.GridView;
import android.widget.Toast;
public class SimpleGridFragment extends Fragment
{
 GridView mGrid;
 String[] mPies;
 @Override
 public void onActivityCreated(Bundle savedInstanceState)
 {
 super.onActivityCreated(savedInstanceState);
 Resources resources = getResources();
```

```

mPies = resources.getStringArray(R.array.pie_array);
//Thiết lập dữ liệu cung cấp cho GridView
mGrid.setAdapter(new ArrayAdapter<String>(getActivity(),
 android.R.layout.simple_list_item_1, mPies));
}
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
 Bundle savedInstanceState)
{ mGrid = (GridView) inflater.inflate(R.layout.grid_fragment, container, false);
 mGrid.setOnItemClickListener(new OnItemClickListener()
 { public void onItemClick(AdapterView<?> parent, View v, int position, long id)
 { Toast.makeText(getApplicationContext(),"Vị trí vừa chọn: " +
 position + ", với nội dung: " + mPies[position], Toast.LENGTH_SHORT).show();
 }
 });
 return mGrid;
}
}

```

**B13.-**Hiệu chỉnh nội dung file *src\com.example.app\_12\MainActivity.java* để có như sau:

```

package com.example.app_12;
import android.app.ActionBar;
import android.app.Activity;
import android.app.FragmentTransaction;
import android.app.ActionBar.Tab;
import android.os.Bundle;
import android.view.Menu;
public class MainActivity extends Activity
{ Tab mTab1, mTab2, mTab3, mTab4, mTab5;
 @Override
 protected void onCreate(Bundle savedInstanceState)
 { super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 ActionBar actionBar = getActionBar();
 actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
 mTab1= actionBar.newTab().setText("List").setTabListener(new NavTabListener());
 mTab2= actionBar.newTab().setText("Custom").setTabListener(new NavTabListener());
 mTab3= actionBar.newTab().setText("Grid").setTabListener(new NavTabListener());
 mTab4= actionBar.newTab().setText("Gallery").setTabListener(new NavTabListener());
 mTab5= actionBar.newTab().setText("Flipper").setTabListener(new NavTabListener());
 actionBar.addTab(mTab1);
 actionBar.addTab(mTab2);
 actionBar.addTab(mTab3);
 actionBar.addTab(mTab4);
 actionBar.addTab(mTab5);
 showList();
 }
 public void showList()
 { SimpleListFragment fragmentA = new SimpleListFragment();
 FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
 /* thay thế linearLayout có tên layout_container đã khai báo trong
 * file res\layout\activity_main.xml */
 ft.replace(R.id.layout_container, fragmentA);
 ft.addToBackStack("fragment a");
 ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
 ft.commit();
 }
 public void showCustomList()
 { CustomListFragment fragmentE = new CustomListFragment();
 FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
 ft.replace(R.id.layout_container, fragmentE);
 ft.addToBackStack("custom");
 ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
 ft.commit();
 }
 public void showGrid()
 { SimpleGridFragment fragmentB = new SimpleGridFragment();

```

```

 FragmentTransaction ft = getFragmentManager().beginTransaction();
 ft.replace(R.id.layout_container, fragmentB);
 ft.addToBackStack("fragment b");
 ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
 ft.commit();
 }
 @Override
 public boolean onCreateOptionsMenu(Menu menu)
 { getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }
 private class NavTabListener implements ActionBar.TabListener
 {
 public NavTabListener()
 {
 }
 @Override
 public void onTabReselected(Tab tab, FragmentTransaction ft)
 {
 }
 @Override
 public void onTabSelected(Tab tab, FragmentTransaction ft)
 {
 if (tab.equals(mTab1))
 { showList(); }
 else if (tab.equals (mTab2))
 { showCustomList(); }
 else if (tab.equals (mTab3))
 { showGrid(); }
 }
 @Override
 public void onTabUnselected(Tab tab, FragmentTransaction ft)
 {
 }
 }
}

```

**B14.-** Chạy chương trình để xem kết quả

### 2.6.2.2. Tạo Fragment với Gallery

Gallery được dùng để hiển thị dữ liệu theo chiều ngang, vì vậy người dùng cần nhấn chọn và kéo màn hình qua lại khi xem. Hầu hết control Gallery được dùng để xem hình ảnh, tuy nhiên trong phần giới thiệu các fragment này, chúng ta sử dụng Gallery để xem như đây cũng là 1 trong những cách để liệt kê các mục chọn.

**Lưu ý về Gallery Deprecated:** Nền tảng của Android luôn được cập nhật thường xuyên với những phiên bản mới do đó sẽ có những view, phương thức, ... của những phiên bản trước sẽ được loại bỏ hoặc thay thế trong những phiên bản sau. Class Gallery được đề nghị loại bỏ từ phiên bản API level 16 là Android 4.1 (JellyBean). Vì vậy khi sử dụng Gallery trong lập trình, bạn sẽ nhận được cảnh báo (thường là *The type Gallery is deprecated*). Có 2 cách chọn lựa khi nhận được cảnh báo này:

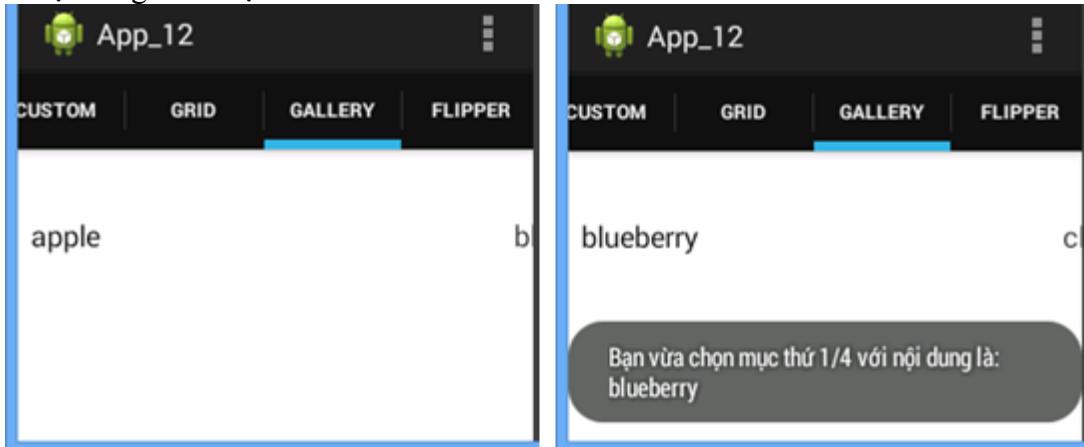
- Thay thế Gallery bằng ViewPager từ gói hỗ trợ và HorizontalScrollView (đề nghị của Android).
- Vẫn sử dụng Gallery: thêm lệnh sau vào sau các lệnh import và nằm ngoài nội dung class của chương trình.

*@SuppressWarnings("deprecation")*

## BÀI THỰC HÀNH App\_12 (tiếp theo)

### ☞ Yêu cầu 4:

- Tạo 1 danh sách các mục chọn bằng Gallery với nội dung tương tự như trong các tab trước đây.
- Khi người dùng nhấn chọn trên bất kỳ item nào, sẽ sử dụng Toast để hiển thị vị trí (dòng, cột) và nội dung vừa chọn.



Hình 2-76 Giao diện của tab “Gallery” và kết quả khi người dùng click chọn

### ☞ Thực hiện

- B15.-** Tạo mới file layout với tên gallery\_fragment.xml, trong đó GridView được tổ chức thành 2 cột:

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

- B16.-** Tạo mới file layout với tên SimpleGalleryFragment.java, với nội dung:

```
package com.example.app_12;
import android.app.Fragment;
import android.content.res.Resources;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Gallery;
import android.widget.Toast;
@SuppressWarnings("deprecation")
public class SimpleGalleryFragment extends Fragment
{
 Gallery mGallery;
 String[] mPies;

 @Override
 public void onActivityCreated(Bundle savedInstanceState)
 {
 super.onActivityCreated(savedInstanceState);
 Resources resources = getResources();
 mPies = resources.getStringArray(R.array.pie_array);
 mGallery.setAdapter(new ArrayAdapter<String>(getActivity(),
 android.R.layout.simple_list_item_1, mPies));
 }
 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container,
```

```

 Bundle savedInstanceState)
{
 mGallery = (Gallery)inflater.inflate(R.layout.gallery_fragment, container, false);
 mGallery.setOnItemClickListener(new OnItemClickListener()
 {
 public void onItemClick(AdapterView<?> parent, View v, int position, long id)
 {
 Toast.makeText(getApplicationContext(),"Bạn vừa chọn mục thứ"
 + position + "/" + mPies.length + " với nội dung là: " + Pies[position],
 Toast.LENGTH_SHORT).show();
 }
 });
 return mGallery;
}
}

```

**B17.-**Hiệu chỉnh nội dung file *src\com.example.app\_12\MainActivity.java* để có như sau:

```

package com.example.app_12;
import android.app.ActionBar;
import android.app.Activity;
import android.app.FragmentTransaction;
import android.app.ActionBar.Tab;
import android.os.Bundle;
import android.view.Menu;
public class MainActivity extends Activity
{ Tab mTab1, mTab2, mTab3, mTab4, mTab5;
 @Override
 protected void onCreate(Bundle savedInstanceState)
 { super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 ActionBar actionBar = getActionBar();
 actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
 mTab1= actionBar.newTab().setText("List").setTabListener(new NavTabListener());
 mTab2= actionBar.newTab().setText("Custom").setTabListener(new NavTabListener());
 mTab3= actionBar.newTab().setText("Grid").setTabListener(new NavTabListener());
 mTab4= actionBar.newTab().setText("Gallery").setTabListener(new NavTabListener());
 mTab5= actionBar.newTab().setText("Flipper").setTabListener(new NavTabListener());
 actionBar.addTab(mTab1);
 actionBar.addTab(mTab2);
 actionBar.addTab(mTab3);
 actionBar.addTab(mTab4);
 actionBar.addTab(mTab5);
 showList();
 }
 public void showList()
 { SimpleListFragment fragmentA = new SimpleListFragment();
 FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
 /* thay thế linearLayout có tên layout_container đã khai báo trong
 * file res\layout\activity_main.xml */
 ft.replace(R.id.layout_container, fragmentA);
 ft.addToBackStack("fragment a");
 ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
 ft.commit();
 }
 public void showCustomList()
 { CustomListFragment fragmentE = new CustomListFragment();
 FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
 ft.replace(R.id.layout_container, fragmentE);
 ft.addToBackStack("custom");
 ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
 ft.commit();
 }
 public void showGrid()
 { SimpleGridFragment fragmentB = new SimpleGridFragment();
 FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
 ft.replace(R.id.layout_container, fragmentB);
 ft.addToBackStack("fragment b");
 ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
 ft.commit();
 }
}

```

```

 }
 public void showGallery()
 { SimpleGalleryFragment fragmentC = new SimpleGalleryFragment();
 FragmentTransaction ft = getFragmentManager().beginTransaction();
 ft.replace(R.id.layout_container, fragmentC);
 ft.addToBackStack("gallery");
 ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
 ft.commit();
 }
 @Override
 public boolean onCreateOptionsMenu(Menu menu)
 { getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }
 private class NavTabListener implements ActionBar.TabListener
 {
 public NavTabListener()
 {
 }
 @Override
 public void onTabReselected(Tab tab, FragmentTransaction ft)
 {
 }
 @Override
 public void onTabSelected(Tab tab, FragmentTransaction ft)
 {
 if (tab.equals(mTab1))
 { showList();
 }
 else if (tab.equals (mTab2))
 { showCustomList();
 }
 else if (tab.equals (mTab3))
 { showGrid();
 }
 else if (tab.equals (mTab4))
 { showGallery();
 }
 }
 @Override
 public void onTabUnselected(Tab tab, FragmentTransaction ft)
 {
 }
 }
}

```

**B18.-**Chạy chương trình để xem kết quả.

### 2.6.3. Sử dụng AdapterViewFlipper

*AdapterViewFlipper* (*android.widget.AdapterViewFlipper*) cho hiển thị mỗi lần 1 mục chọn và cung cấp khả năng “lật” (di chuyển) giữa các mục chọn. Bạn có thể sử dụng AdapterViewFlipper trong fragment bằng cách thực hiện tương tự như đã làm với GridView và Gallery.

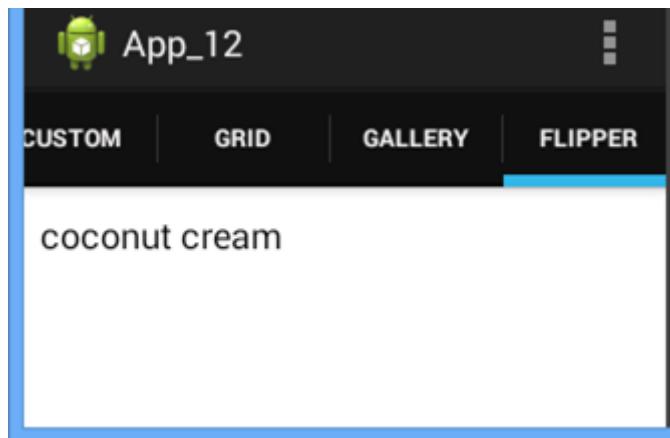
Để bật tính năng “lật” tự động cho AdapterViewFlipper, cần thiết lập cho AutoStart giá trị true và gán thời gian cho mỗi lần “lật” với đơn vị tính là milli giây.

Ngược lại, khi tắt tính năng “lật” tự động, để tác động đến việc “lật” giữa các mục chọn, ta cần sử dụng đến các phương thức *showNext()*, *showPrevious()*, *stopFlipping()*, và *startFlipping()*.

## BÀI THỰC HÀNH App\_12 (tiếp theo)

### ☞ Yêu cầu 5:

- Tạo 1 danh sách các mục chọn bằng AdapterViewFlipper với nội dung tương tự như trong các tab trước đây.



Hình 2-77 Giao diện của tab “AdapterViewFlipper”

### ☞ Thực hiện

**B19.-** Tạo mới file layout với tên *view\_flipper\_fragment.xml* với nội dung như sau:

```
<AdapterViewFlipper xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@+id/flipper"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:layout_centerInParent="true">
</AdapterViewFlipper>
```

**B20.-** Tạo mới file layout với tên *SimpleFlipperFragment.java*, với nội dung:

```
package com.example.app_12;
import android.app.Fragment;
import android.content.res.Resources;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterViewFlipper;
import android.widget.ArrayAdapter;
public class SimpleFlipperFragment extends Fragment {
 AdapterViewFlipper mFlipper;
 String[] mPies;

 @Override
 public void onActivityCreated(Bundle savedInstanceState) {
 super.onActivityCreated(savedInstanceState);
 Resources resources = getResources();
 mPies = resources.getStringArray(R.array.pie_array);
 mFlipper.setAdapter(new ArrayAdapter<String>(getActivity(),
 android.R.layout.simple_list_item_1, mPies));
 }
 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container,
 Bundle savedInstanceState)
 {
 mFlipper = (AdapterViewFlipper) inflater.inflate(R.layout.view_flipper_fragment,
 container, false);
 // Thiết lập chế độ chạy tự động
 mFlipper.setAutoStart(true);
 // Thiết lập thời gian chuyển giữa các mục chọn là 2 giây
 mFlipper.setFlipInterval(2000);
 return mFlipper;
 }
}
```

**B21.-** Hiệu chỉnh nội dung file *src\com.example.app\_12\MainActivity.java* để có như sau:

```
package com.example.app_12;
import android.app.ActionBar;
import android.app.Activity;
import android.app.FragmentTransaction;
import android.app.ActionBar.Tab;
import android.os.Bundle;
import android.view.Menu;

public class MainActivity extends Activity
{ Tab mTab1, mTab2, mTab3, mTab4, mTab5;
 @Override
 protected void onCreate(Bundle savedInstanceState)
 { super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 ActionBar actionBar = getActionBar();
 actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
 mTab1= actionBar.newTab().setText("List").setTabListener(new NavTabListener());
 mTab2= actionBar.newTab().setText("Custom").setTabListener(new NavTabListener());
 mTab3= actionBar.newTab().setText("Grid").setTabListener(new NavTabListener());
 mTab4= actionBar.newTab().setText("Gallery").setTabListener(new NavTabListener());
 mTab5= actionBar.newTab().setText("Flipper").setTabListener(new NavTabListener());
 actionBar.addTab(mTab1);
 actionBar.addTab(mTab2);
 actionBar.addTab(mTab3);
 actionBar.addTab(mTab4);
 actionBar.addTab(mTab5);
 showList();
 }
 public void showList()
 { SimpleListFragment fragmentA = new SimpleListFragment();
 FragmentTransaction ft = getFragmentManager().beginTransaction();
 /* thay thế linearLayout có tên layout_container đã khai báo trong
 * file res\layout\activity_main.xml */
 ft.replace(R.id.layout_container, fragmentA);
 ft.addToBackStack("fragment a");
 ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
 ft.commit();
 }
 public void showCustomList()
 { CustomListFragment fragmentE = new CustomListFragment();
 FragmentTransaction ft = getFragmentManager().beginTransaction();
 ft.replace(R.id.layout_container, fragmentE);
 ft.addToBackStack("custom");
 ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
 ft.commit();
 }
 public void showGrid()
 { SimpleGridFragment fragmentB = new SimpleGridFragment();
 FragmentTransaction ft = getFragmentManager().beginTransaction();
 ft.replace(R.id.layout_container, fragmentB);
 ft.addToBackStack("fragment b");
 ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
 ft.commit();
 }
 public void showGallery()
 { SimpleGalleryFragment fragmentC = new SimpleGalleryFragment();
 FragmentTransaction ft = getFragmentManager().beginTransaction();
 ft.replace(R.id.layout_container, fragmentC);
 ft.addToBackStack("gallery");
 ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
 ft.commit();
 }
 public void showFlipper()
 { SimpleFlipperFragment fragmentD = new SimpleFlipperFragment();
 FragmentTransaction ft = getFragmentManager().beginTransaction();
 ft.replace(R.id.layout_container, fragmentD);
 ft.addToBackStack("flipper");
 ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
```

```
 ft.commit();
 }
 @Override
 public boolean onCreateOptionsMenu(Menu menu)
 {
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }
 private class NavTabListener implements ActionBar.TabListener
 {
 public NavTabListener()
 {
 }
 @Override
 public void onTabReselected(Tab tab, FragmentTransaction ft)
 {
 }
 @Override
 public void onTabSelected(Tab tab, FragmentTransaction ft)
 {
 if (tab.equals(mTab1))
 {
 showList();
 }
 else if (tab.equals (mTab2))
 {
 showCustomList();
 }
 else if (tab.equals (mTab3))
 {
 showGrid();
 }
 else if (tab.equals (mTab4))
 {
 showGallery();
 }
 else if (tab.equals (mTab5))
 {
 showFlipper();
 }
 }
 @Override
 public void onTabUnselected(Tab tab, FragmentTransaction ft)
 {
 }
 }
}
```

#### 2.6.4. Một số control dùng trong việc phân trang dang cuộn ngang

<i>View</i>	<i>Mô tả</i>	<i>API Level</i>
Gallery	Sử dụng cuộn ngang cho các view. Mở rộng từ <i>AdapterView</i>	Từ API Level 1, deprecated từ API level 16
AdapterViewFlipper	“Lật” giữa các view. Mở rộng từ <i>AdapterView</i>	Từ API Level 11
ViewFlipper	“Lật” giữa các view	Từ API Level 1
ViewPager ( <i>android.support.v4. view.ViewPager</i> )	“Lật” giữa các view. Sử dụng <i>PagerAdapter</i> để phát sinh các trang khi hiển thị Là 1 phần của thư viện hỗ trợ. Nhờ vậy có thể làm việc với tất cả các version của Android	Gói hỗ trợ
HorizontalScrollView	Chứa layout dùng cho việc cuộn ngang	Từ API Level 3

## **2.7. BÀI TẬP TỔNG HỢP PHẦN II**

### 2.7.1. Thiết kế giao diện

### **2.7.1.1.**

Sử dụng FrameLayout để tạo giao diện với mỗi button nằm tại 1 góc của màn hình (HD: sử dụng các thuộc tính top/ bottom/left/right Margin).

## **2.7.1.2.**

Thực hiện tương tự với RalativeLayout.

**2.7.1.3.**

Tạo 1 layout dạng LinearLayout với thuộc tính orientation là vertical. Thêm 1 FrameLayout và 1 button vào trong LinearLayout. Sau đó đưa 1 ImageView và 1 textView vào trong FrameLayout.

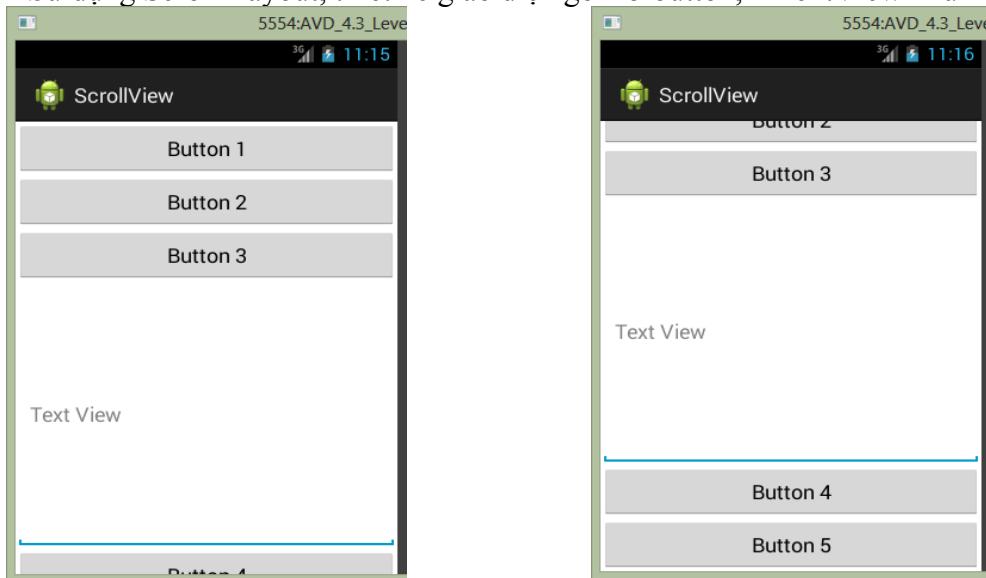
**2.7.1.4.**

Tạo 1 layout dạng RelativeLayout trong đó chứa 1 ImageView và 1 TextView. Lần lượt thay đổi vị trí của textView sao cho có 4 dạng thể hiện của button:

- TextView nằm trên ImageView.
- TextView nằm dưới ImageView.
- TextView nằm bên trái ImageView.
- TextView nằm bên phải ImageView.

**2.7.1.5.**

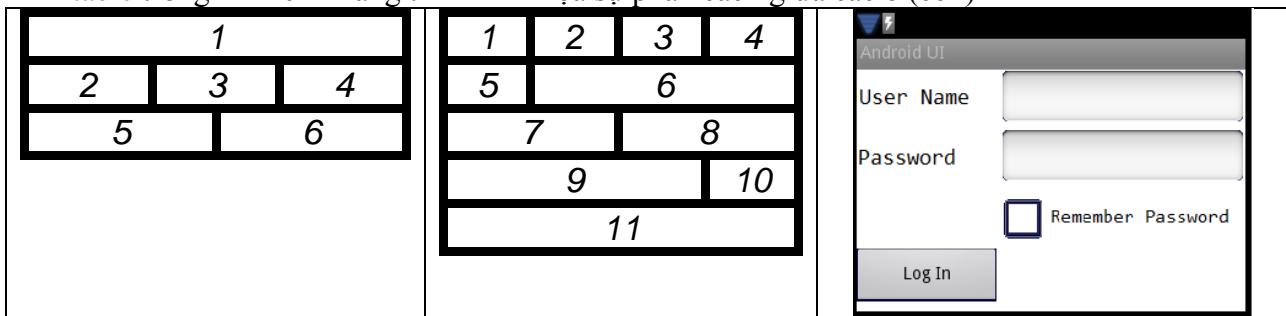
Sử dụng Scroll Layout, thiết kế giao diện gồm 5 button, 1 TextView như hình 2-77.



Hình 2-78 Màn hình kết quả trước (hình bên trái) và sau (hình bên phải) khi cuộn màn hình

**2.7.1.6.**

Sử dụng Table Layout, lần lượt thiết kế giao diện từng giao diện các hình thuộc nhóm 2-78. Lưu ý: phải tạo 3 file layout khác nhau cho mỗi hình có trong yêu cầu. Các đường kẻ của table trong hình chỉ mang tính minh họa sự phân cách giữa các ô (cell)

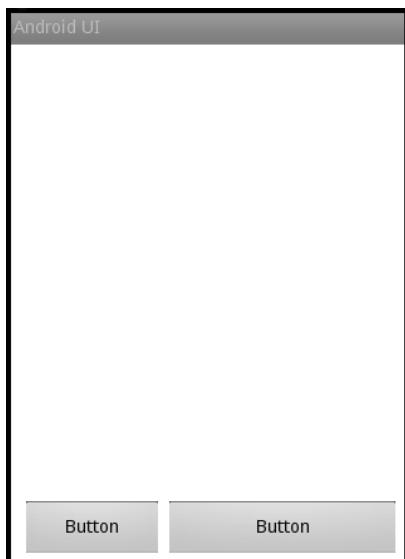


Hình 2-79 Tạo 3 file layout với giao diện có dạng như hình mô tả

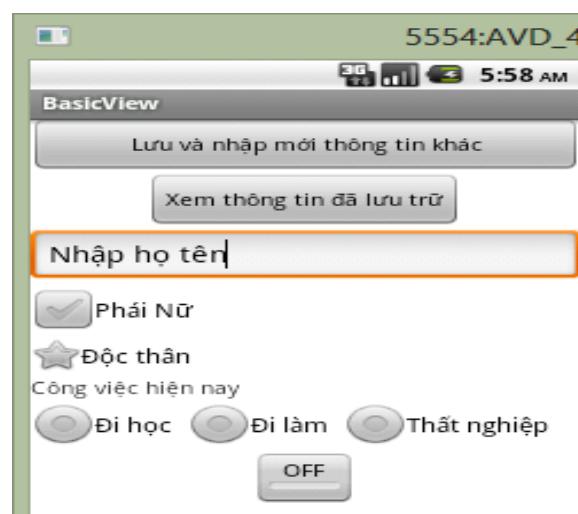
**2.7.1.7.**

Chọn layout phù hợp để lần lượt thiết kế giao diện từng giao diện các hình 2-79 và 2-80 Trong đó hình 2-80:

- *Yêu cầu:* sử dụng ToggleButton cho button “OFF”
- *Hướng dẫn:* Sử dụng thuộc tính style để checkBox dạng hình ngôi sao  
VD: style="?android:attr/starStyle"



Hình 2-80



Hình 2-81

## 2.7.2. Bài tập về fragment

### 2.7.2.1.

Tạo 1 activity với 2 layout. Một trong số đó thực thi bởi 1 ListFragment. Khi người dùng click chọn 1 item trong danh sách có trong ListFragment sẽ hiển thị fragment thứ 2. Fragment thứ 2 có thể là bất kỳ dạng Fragment nào mà bạn đã biết.

### 2.7.2.2.

Trở lại activity *InteractionFragmentActivity*, hiệu chỉnh lại để khi Fragment được định nghĩa, 1 bundle sẽ chuyển đến activity với thông tin về 1 câu hỏi với nội dung bất kỳ (ví dụ: bạn có thực sự muốn kết thúc ứng dụng hay Bạn có muốn ứng dụng giải dùm game đang chơi hay không? ...)

### 2.7.2.3.

Cũng trong activity *InteractionFragmentActivity*, hiệu chỉnh lại để FragmentYesNo thực hiện việc di chuyển trong phương thức callback *onAnswerSelected()* để đến 2 fragment đơn giản được tạo thêm (FragmentC và FragmentD). Cụ thể là sẽ chuyển đến FragmentC khi người dùng chọn Yes và chuyển đến FragmentD khi người dùng chọn No.

### 2.7.2.4.

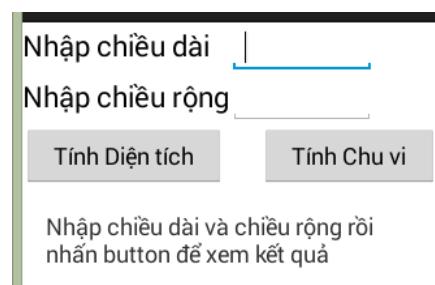
Sử dụng đoạn mã lệnh có dạng như sau cho cả activity và fragment. Trong đó class Log được dùng để đăng nhập vào phương thức có tên được gọi. Ví dụ đối với fragment, thêm phát biểu đăng nhập đối với các phương thức *onAttach()*, *onCreate()*, *onCreateView()*, và *onActivityCreated()*:

```
import android.util.Log;
public class MainActivity extends Activity
{
 private static final String TAG = MainActivity.class.getName();
 @Override
 protected void onCreate(Bundle savedInstanceState)
 {
 Log.d(TAG, "onCreate");
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 }
}
```

### 2.7.3. Bài tập về các xử lý cơ bản

#### 2.7.3.1.

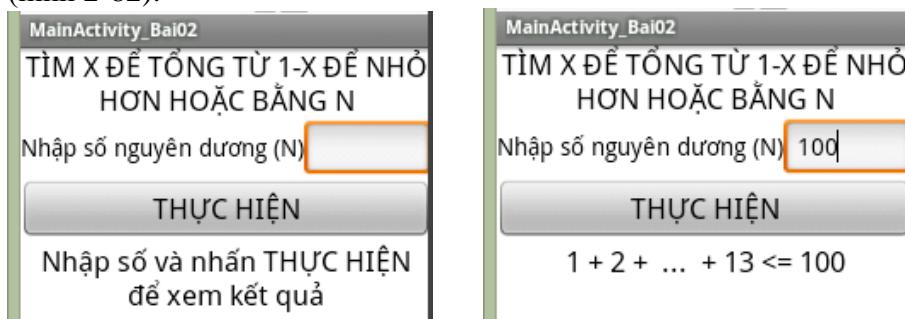
Viết chương trình cho nhập chiều rộng và chiều dài, tính chu vi và diện tích hình chữ nhật theo mẫu sau (hình 2-81). Kết quả sẽ được hiển thị vào TextView “Nhập chiều dài và chiều rộng rồi nhấn button để xem kết quả”:



Hình 2-82 Giao diện của bài tập 2.7.3.1

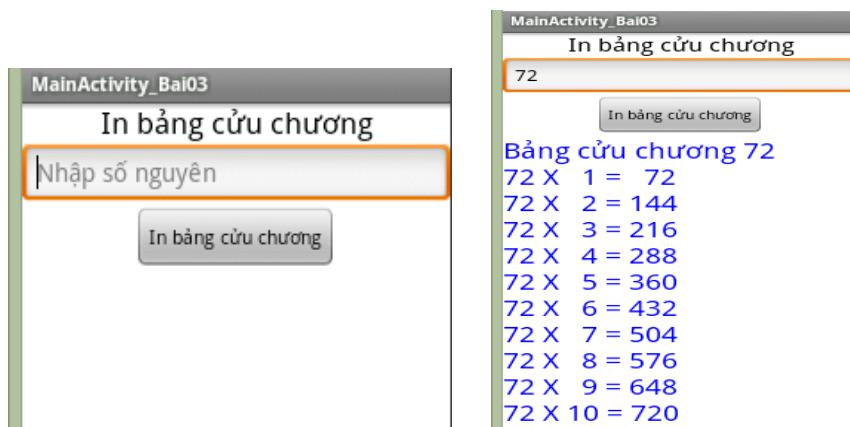
#### 2.7.3.2.

Viết chương trình nhập số nguyên dương N. Tìm X tổng các số nguyên từ 1 đến X  $\leq N$ . Kết quả sẽ được in ra thay thế cho nội dung đang chứa chuỗi “Nhập số và nhấn THỰC HIỆN để xem kết quả”(hình 2-82).



Hình 2-83 Giao diện ban đầu (trái) và giao diện sau khi nhấn button (phải) của bài tập 2.7.3.2

#### 2.7.3.3.



Hình 2-84 Minh họa giao diện của bài tập 2.7.3.3

Viết chương trình nhập bảng cửu chương n (với  $n \geq 2$ ) (hình 2-40). Yêu cầu:

- Đầu chương trình:

  - Trong EditText có sẵn chuỗi “Nhập số nguyên”.
  - Khi EditText được chọn, bàn phím xuất hiện chỉ là các phím số.

- Kết quả: bảng cửu chương có:
  - Màu chữ : màu xanh (Color.BLUE)
  - Kích thước chữ : 18.

#### 2.7.3.4.

Biết thông tin về tình trạng sức khỏe và lời khuyên

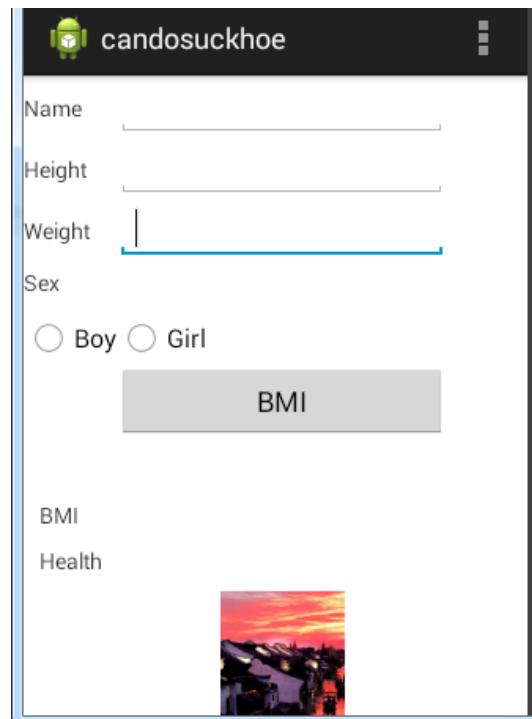
Giá trị của BMI	<19	Từ 19 đến 25	>25
Tình trạng sức khỏe	Gầy ốm	Thể trạng cân đối	Béo phì
Lời khuyên	Cố gắng bồi dưỡng	Cố gắng giữ gìn sức khỏe	Cần giảm cân

Kiểm tra khi người dùng không nhập các giá trị, hoặc nhập không đúng (số âm, nhập chữ, ...)

Kiểm tra tình trạng sức khỏe

- Thông tin người dùng cung cấp:
  - *Name*: Nhập tên người dùng.
  - *Height*: Nhập Chiều cao người dùng, ví dụ: 1m72 nhập vào 1.72.
  - *Weight*: Cân nặng người dùng, ví dụ: 60 kg nhập vào 60.
  - *Sex*: Lựa chọn giới tính, Boy = Nam, Girl = Nữ.
- Nhấn button BMI sau khi nhập đủ thông tin để chương trình tính toán.
- Cách tính BMI:  
 $BMI = \text{Cân Nặng} / (\text{Chiều cao} * \text{Chiều Cao})$
- Thông tin kết quả nhận được:
  - Tên người dùng, giới tính.
  - BMI: Chỉ số sức khỏe người dùng.
  - Health: thông tin về tình trạng sức khỏe và lời khuyên.

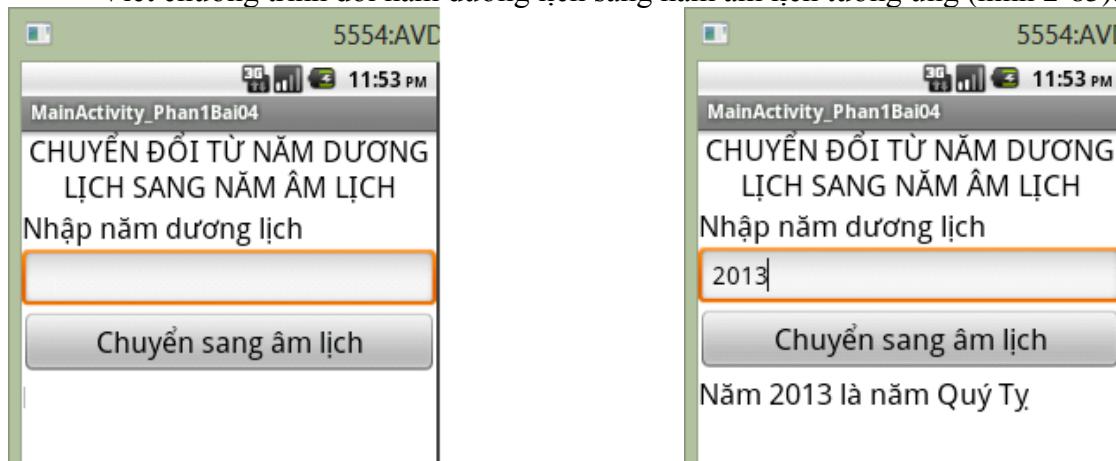
Thực hiện kiểm tra khi người dùng không nhập các giá trị, hoặc nhập không đúng (số âm, nhập chữ thay vì phải nhập số, ...).



Hình 2-85 Minh họa giao diện của bài tập 2.7.3.4

### 2.7.3.5.

Viết chương trình đổi năm dương lịch sang năm âm lịch tương ứng (hình 2-85).



Hình 2-86 Minh họa giao diện của bài tập 2.7.3.5

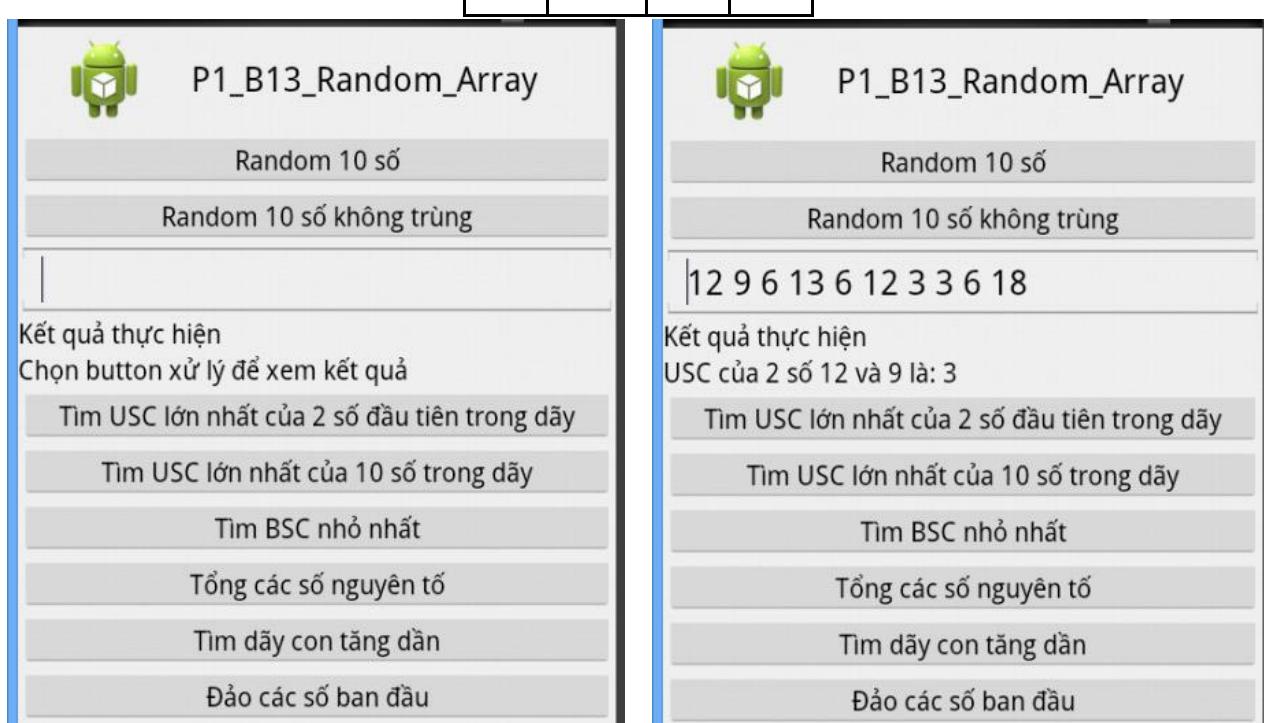
### 2.7.3.6.

Thiết kế giao diện gồm 2 TextView (để hiển thị kết quả), 1 EditText để chứa 10 số được phát sinh, 8 Button.

- Dãy số phát sinh ngẫu nhiên đưa vào EditText được cách nhau bởi khoảng trắng, có giá trị ngẫu nhiên từ 2 đến 20.
- Chức năng thực hiện của mỗi button được ghi trên button đó. (hình 2-86)
- Hướng dẫn:
  - ArrayList:
    - **Khai báo** ArrayList chứa số nguyên
 

```
ArrayList<Integer> AL=new ArrayList<Integer>();
```
    - **Khai báo**

- Gán giá trị cho phần tử trong ArrayList:
    - `AL.add(Giá trị);`
    - `AL.set(chỉ số, giá trị);`
  - Lấy giá trị của phần tử trong ArrayList: `tên biến = A.get(i);`
- (ii). Tạo số ngẫu nhiên từ Min->Max:
- B1:** Khai báo biến Min, Max, x. VD: int Min=0, Max=100, x;
  - B2:** Tạo số ngẫu nhiên x
    - Cách 1: sử dụng phương thức tạo số ngẫu nhiên của đối tượng Math  
`Min+(int)(Math.random()*(Max-Min)+1))`  
VD: `x=Min+(int)(Math.random()*(Max-Min)+1))`;
    - Cách 2: sử dụng phương thức tạo số ngẫu nhiên của đối tượng Random  
`Random rd=new Random();`  
VD: `Random rd=new Random();  
x=rd.nextInt((Max-Min+1)+Min);`
- (iii). Cắt các từ cách nhau bởi cùng 1 dạng ký tự (như khoảng trắng, dấu phẩy, ...) trong chuỗi lớn đưa vào mảng
- VD: `String s="7 14 3 5";  
String[] B = s.split(" ");`  
Sau lệnh này mảng B có dạng
- |     |      |     |     |
|-----|------|-----|-----|
| “7” | “14” | “3” | “5” |
|-----|------|-----|-----|

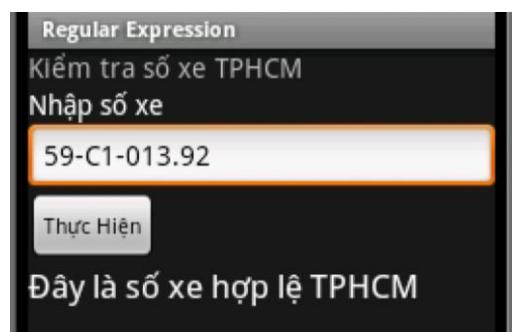


Hình 2-87 Minh họa giao diện của bài tập 2.7.3.6

### 2.7.3.7.

Viết chương trình kiểm tra số xe nhập vào có phải là số xe được đăng ký tại TP.HCM hay không? Quy ước về số xe được xem là hợp lệ: dãy số gồm 3 phần

- Nhóm 1: đầu tiên là chữ số 5, tiếp sau là một ký số từ 0-9
- Nhóm 2 : một ký tự từ A-Z, tiếp theo là một ký số từ 0-9
- Nhóm 3: Ba ký tự số từ 0-9 ([0-9]{3}) tiếp theo là dấu chấm và cuối cùng là 2 ký số từ 0-9



Hình 2-88 Minh họa giao diện của bài tập 2.7.3.7

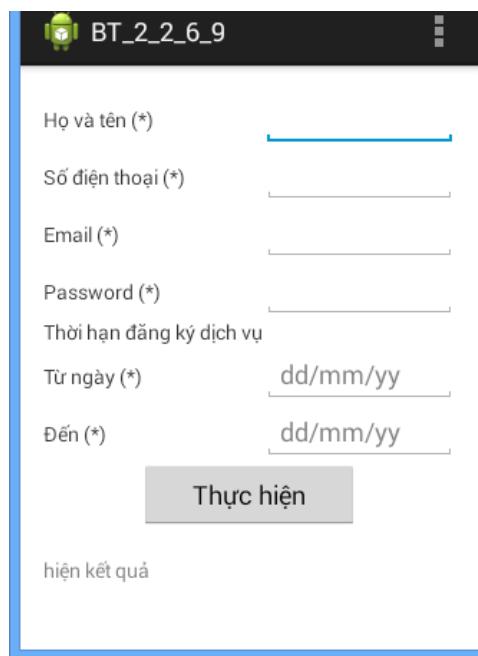
- HD: Ví dụ số xe hợp lệ: 59-C1-013.92

```
String dinhdang_soxe_TPHCM = "5[0-9]-[A-Z][0-9]-[0-9]{3}\.\[0-9]{2}";
```

### 2.7.3.8. Viết chương trình cho nhập 2 ngày đến và đi. Tính và in ra số ngày chênh lệch giữa 2 ngày đó

- Yêu cầu:

- Xác định kiểu bàn phím sẽ xuất hiện khi tương ứng với các mục nhập:
  - Họ và tên: chỉ nhập ký tự hoa.
  - Số điện thoại: chỉ cho nhập ký số.
  - Email: có thể nhập địa chỉ email (có dùng ký tự @).
- Khi nhấn button thực hiện sẽ tổng hợp các thông tin trên và in vào TextView “Kết quả”  
VD: TextView “Kết quả” chứa nội dung: Khách hàng “Trần Văn Tý” có số điện thoại là 1234576890, email [travanty@gmail.com](mailto:travanty@gmail.com) đăng ký dịch vụ trong 30 ngày (từ ngày 1 tháng 9 năm 2013 đến ngày 30/ tháng 9 năm 2013)



Hình 2-89 Minh họa giao diện của bài tập 2.7.3.8

- **HD:** Tính toán số ngày giữa 2 mốc thời gian

- Các control khác, sinh viên vẫn xử lý bình thường.
- Bổ sung phương thức sau vào class MainActivity

```
public static long daysBetween2DateString(String sFromDate, String sToDate)
{ // Định dạng thời gian
 SimpleDateFormat dateFormat = new SimpleDateFormat("dd/mm/yy");
 // Định nghĩa 2 mốc thời gian ban đầu
 long dFrom = 0, dTo = 0;
 try
 { dFrom = dateFormat.parse(sFromDate).getTime();
 dTo = dateFormat.parse(sToDate).getTime();
 }
 catch (ParseException e)
 { e.printStackTrace();
 }
 // Công thức tính số ngày giữa 2 mốc thời gian:
 return Math.abs((dFrom-dTo)/ (24 * 60 * 60 * 1000));
}
```

- Bổ sung khai báo phương thức sau vào class MainActivity

```
static EditText ngaydi, ngayden;
TextView ketquathuchien, thongtin;
ngayden=(EditText)findViewById(R.id.edit_date1);
ngaydi=(EditText)findViewById(R.id.edit_date2);
ketquathuchien=(TextView)findViewById(R.id.tv_thuchien);
thongtin=(TextView)findViewById(R.id.tv_thongtin);
```

- Bổ sung đoạn mã lệnh của sự kiện onClick trên button

```
SimpleDateFormat date_format=new SimpleDateFormat();
date_format.applyPattern("dd/mm/yy");
```

```
String den=ngaydi.getText().toString();
if(den=="")
{ ngayden.requestFocus();
 ngayden.selectAll();
 Toast.makeText(this, "Bạn chưa nhập ngày đến", Toast.LENGTH_LONG).show();
 return;
}
```

```

String tu=ngayden.getText().toString().trim();
if(tu=="")
{
 ngaydi.requestFocus();
 ngaydi.selectAll();
 Toast.makeText(this, "Bạn chưa nhập ngày đi", Toast.LENGTH_LONG).show();
 return;
}
thongtin.setText(hoten1+"\n"+sodienthoai1+"\n"+email1);
String ketqua="Đăng ký dịch vụ trong "+ String.valueOf(daysBetween2DateString
(tu, den))+ " ngày, ";
ketqua+="\n từ ngày " + String.valueOf(tu) + " đến ngày" + String.valueOf(den) ;
ketquathuchien.setText(ketqua);

```

### 2.7.3.9. Viết chương trình tính diện tích và chu vi hình tròn

#### Yêu cầu

- Xuất hiện lời nhắc “Nhập bán kính” ngay trong EditText
- Kết quả tính chu vi và diện tích được xuất vào 1 TextView (mỗi kết quả trên 1 dòng) nằm ngay dưới button “TÍNH”.

**HD:** Định dạng số lẻ

VD:

```

double k=123.45678;
DecimalFormat df = new DecimalFormat();
df.setMinimumFractionDigits(2);
df.setMaximumFractionDigits(2);
textView1.setText("Kết quả sau làm tròn: "+df.format(k));

```

### 2.7.3.10. Viết chương trình cho nhập 3 số, tìm số lớn nhất trong 3 số (hình 2-89)



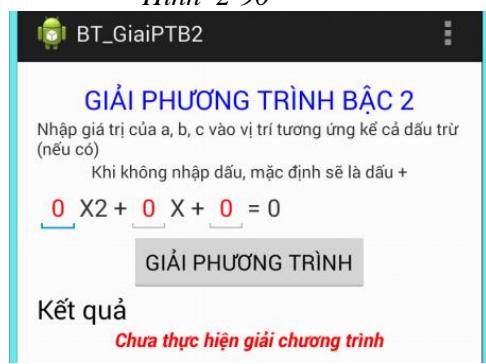
Hình 2-90



Hình 2-91



Hình 2-92



Hình 2-93



Hình 2-94



Hình 2-95

**2.7.3.11. Viết chương trình cho nhập kích thước 3 cạnh. Xác định xem 3 cạnh đó có tạo thành tam giác hay không?** Nếu có, cho biết tam giác là tam giác gì? (đều, cân, vuông cân, vuông, thường) (hình 2-90)

**2.7.3.12. Viết chương trình giải phương trình bậc 1  $ax+b=0$ , với hệ số a và b được nhập vào (hình 2-91)**

**2.7.3.13. Viết chương trình giải phương trình bậc 2  $ax^2+bx+c=0$ , với hệ số a, b, c được nhập vào kể cả dấu trừ nếu có(hình 2-92).** Nếu không nhập dấu, mặc định sẽ là dấu cộng (+). Kết quả xuất hiện sẽ thay thế nội dung của chuỗi “Chưa thực hiện giải chương trình.

**2.7.3.14. Viết chương trình cho nhập 2 số nguyên dương. In ra ước số chung lớn nhất và bội số chung nhỏ nhất của 2 số (hình 2-93).**

**2.7.3.15. Viết chương trình cho nhập số nguyên dương gồm 3 ký số. In ra cách đọc tương ứng (hình 2-94).**

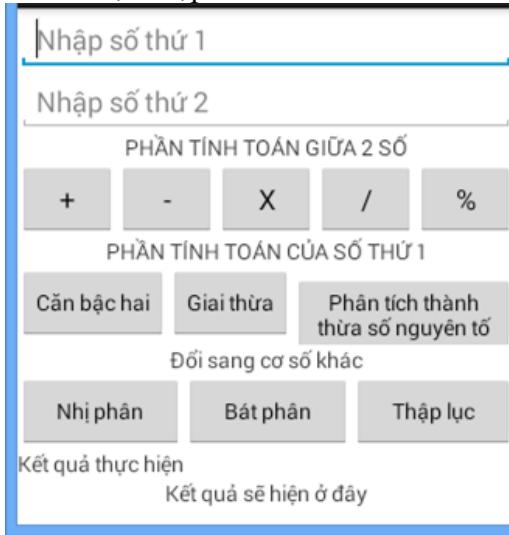
Ví dụ: nhập 135  $\Rightarrow$  in ra một ba năm.

**Mở rộng:** Tương tự, nhập số nguyên tùy ý từ -2.000.000.000 đến 2.000.000.000. In ra cách đọc tương ứng.

**2.7.3.16. Viết chương trình tạo máy tính đơn giản (hình 2-95).**

**2.7.3.17. Cho nhập 1 số nguyên n, phân tích n thành các thừa số nguyên tố (hình 2-96).**

Ví dụ: nhập 135  $\Rightarrow$  in ra  $600 = 2 \times 2 \times 2 \times 3 \times 5 \times 5$



Hình 2-96



Hình 2-97

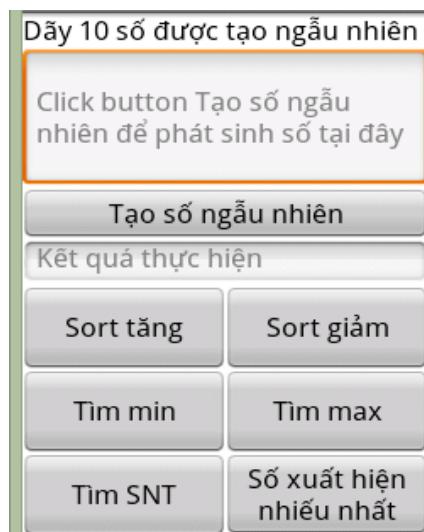
**2.7.3.18. Tạo giao diện gồm 1 TextView, 2 EditText, 7 Button.**

- Dãy số được tạo trong EditText được cách nhau bởi khoảng trắng.
- Khi nhấn chọn button nào thì thực hiện chức năng của button đó. Kết quả sẽ được hiện trong EditText kết quả (hình 2-97)

**2.7.3.19. Viết chương trình chọn ngẫu nhiên 3 lá bài (hình 2-98).**

**Yêu cầu:** Đầu chương trình là hình 3 lá bài “úp”. Khi người dùng nhấn button CHỌN 3 LÁ BÀI KHÁC sẽ:

- Chọn ngẫu nhiên (không trùng) 3 lá bài khác để thay thế.
- Hiện ra tên 3 lá bài với số điểm cộng dồn (số nút) của 3 lá bài.



Hình 2-98



Hình 2-99

### 2.7.3.20. *Tương tự bài tập 2.7.3.19 nhưng dành cho 4 người chơi A, B, C, D.*

Đầu chương trình, màn hình có dạng như hình 2-99. Khi button “CHIA BÀI MỚI” được nhấn, mỗi người chơi sẽ nhận được 3 lá bài ngẫu nhiên (và không trùng nhau). Chương trình cho biết số nút của mỗi người, danh sách người thắng, danh sách người thua (có thể cùng có nhiều người có số nút bằng nhau nên cùng thắng – hình 2-100).



Hình 2-100



Hình 2-101

### 2.7.4. *ListView kết hợp với các cấu trúc dữ liệu*

#### 2.7.4.1. *Kết hợp với AutoCompleteTextView*

Xây dựng ứng dụng gồm các control EditText, Button, AutoCompleteTextView và 1 ListView. Tính năng của từng view như sau:

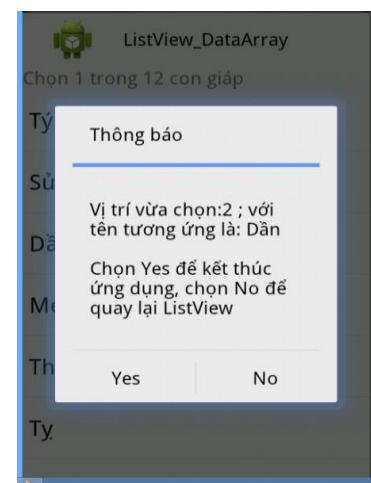
- Sử dụng EditText và Button để đưa 1 dữ liệu kiểu String cung cấp dữ liệu cho AutoCompleteView. Kiểm tra việc nhập dữ liệu bị trùng.
- ListView hiển thị tất cả các dữ liệu cung cấp cho AutoCompleteView. Khi dữ liệu trên AutoCompleteView khớp với mục nào trong ListView thì mục đó được chọn

### 2.7.4.2. Kết hợp với ArrayAdapter

Viết chương trình xử lý khi người dùng nhấn trên item nào sẽ sử dụng Dialog để hiển thị thông tin về vị trí và giá trị của mục vừa được chọn. Khi dialog xuất hiện, chọn “Yes” để kết thúc ứng dụng, chọn “No” để quay lại màn hình chứa ListView.

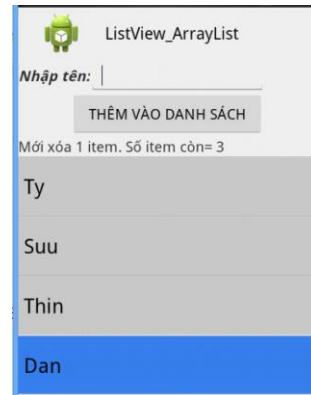
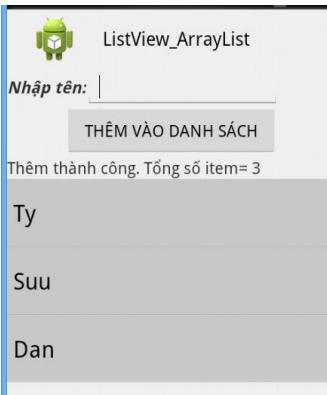
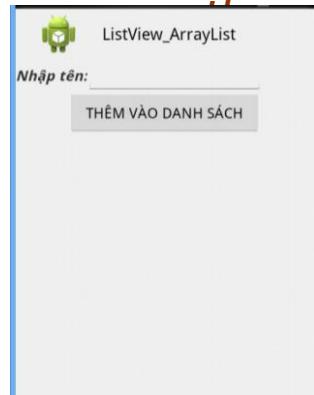


Hình 2-102



Hình 2-103

### 2.7.4.3. Kết hợp với ArrayList

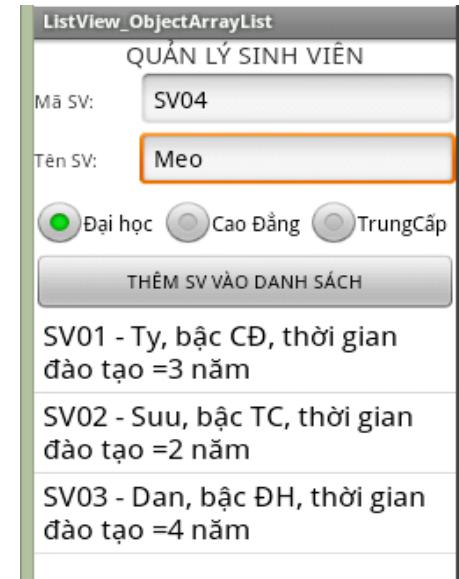
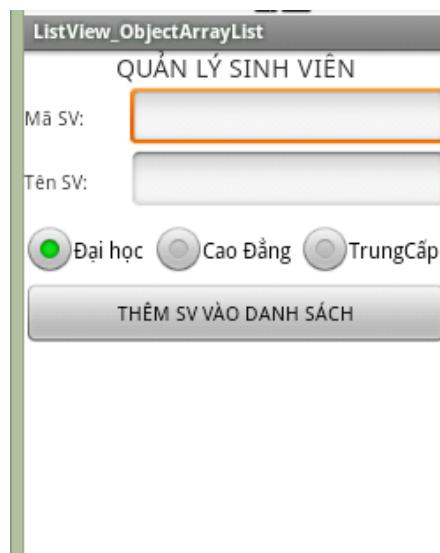


Hình 2-104 Các trạng thái của ListView khi ứng dụng hoạt động

- Nhập dữ liệu và nhấn nút “Nhập” thì sẽ đưa vào ArrayList và hiển thị lên ListView
- Nhấn vào phần tử nào thì hiển thị vị trí và giá trị của phần tử đó lên TextView
- Nhấn thật lâu (long click) vào phần tử nào đó trên ListView thì sẽ xóa phần tử đó.

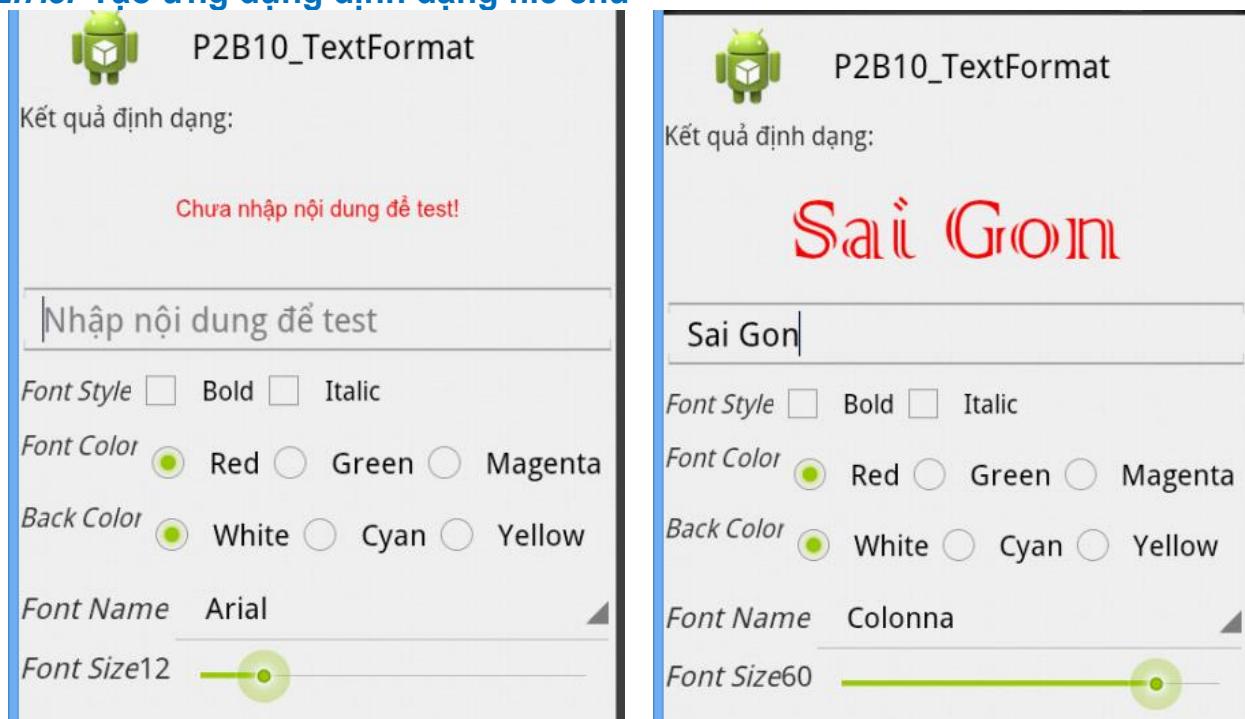
### 2.7.4.4. Kết hợp với Object ArrayList

Ban đầu ListView không có phần tử nào (rỗng). Sau khi thêm sẽ đưa vào ListView.



Hình 2-105 Các trạng thái của ListView khi ứng dụng Quản lý sinh viên hoạt động

### 2.7.5. Tạo ứng dụng định dạng file chữ



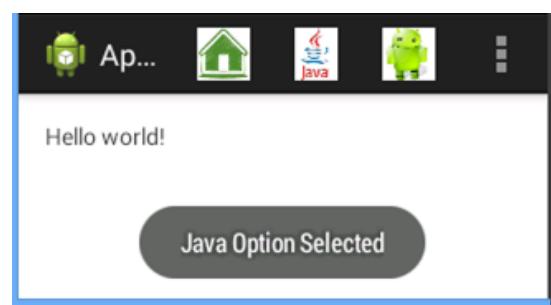
Hình 2-106 Các trạng thái của ứng dụng

#### Mục đích:

- Xử lý sự kiện khi người dùng nhập nội dung trên TextView ( thông qua sự kiện setOnKeyListener )
- Thao tác với Spinner, SeekBar
- Xử lý font chữ (FontName, Font Style, Font Size) thông qua đối tượng Typeface.

### 2.7.6. ActionBar

Tạo project có ActionBar, trên đó gồm 4 item: Home, Java, Android và Setting. 3 item đầu sử dụng icon. Khi click trên item nào sẽ thông báo cho biết tên item đã chọn



Hình 2-107 Giao diện của bài tập 2.7.6

### 2.7.7. TimePickerDialog kết hợp với dialog fragment

Sử dụng *TimePickerDialog* và tạo ra 1 dialog fragment. Trong dialog fragment này, tạo 1 listener interface để chuyển thời gian mà người dùng chọn về cho activity đã kích hoạt *TimePickerDialog*.

## Phần III: LƯU TRỮ DỮ LIỆU

- (i). Quản lý các tùy chọn (Preferences)
- (ii). SQLite
- (iii). Thao tác với file của ứng dụng được tạo ra trên AVD
- (iv). Sử dụng SQLite trong Android
- (v). Content Provider
- (vi). Lưu trữ dữ liệu dưới dạng file

### 3.1. QUẢN LÝ CÁC TÙY CHỌN (Preferences)

`SharedPreferences`(`android.content.SharedPreferences`) là 1 class cung cấp cơ chế cho việc lưu trữ và truy xuất dữ liệu dưới dạng các cặp key-value (khóa-giá trị). Dữ liệu này có thể được thiết lập và truy xuất mọi loại kiểu dữ liệu như: boolean, float, int, long và string từ bất kỳ activity nào. Android cung cấp các tùy chọn API rất mạnh để người dùng có thể sử dụng `SharedPreferences` như là kho dữ liệu cơ bản.

Preferences trên Android sẽ được lưu trữ trên file

```
/data/data/[PACKAGE_NAME]/shared_prefs/[PREFERENCE_FILE_NAME].xml
```

`key` (trong cặp `key-value`) sẽ được đặt tên bằng cách đặt tên của `key` liền sau với tên của package và toàn bộ nội dung đó được đặt trong cặp dấu nháy đôi (""), ví dụ

```
public static final String SETTINGS = "com.example.app_13.settings";
```

#### 3.1.1. Sử dụng SharedPreferences

##### 3.1.1.1. Các phương thức hỗ trợ lấy đối tượng SharedPreferences

Có thể sử dụng một trong hai phương thức sau:

###### 3.1.1.1.1. `public abstract SharedPreferences getSharedPreferences (String fileName, int mode):`

- Phương thức này sẽ lấy về đối tượng `SharedPreferences` để đọc ghi dữ liệu lên file xml.
- Tham số:
  - `String fileName`: tên file xml được dùng để đọc ghi dữ liệu. Chỉ ghi tên file mà không cần ghi phần mở rộng, vì phần mở rộng mặc nhiên sẽ là `.xml`
  - `int mode`: thiết lập quyền truy xuất đến file xml mà đối tượng `SharedPreferences` tham chiếu đến. Có ba loại:
    - `MODE_PRIVATE`: chỉ có thể được truy xuất bên trong ứng dụng tạo ra nó.
    - `MODE_WORLD_READABLE`: có thể được đọc bởi các ứng dụng khác. Được đề nghị hủy bỏ từ API Level 17.
    - `MODE_WORLD_WRITEABLE`: có thể được đọc và ghi bởi các ứng dụng khác. Được đề nghị hủy bỏ từ API Level 17.
- Ở lần đầu tiên tạo ra đối tượng `SharedPreferences` nếu file xml dùng để lưu trữ dữ liệu chưa tồn tại thì hệ thống sẽ tự tạo file xml với tên chỉ định ở tham số trên.
- Ví dụ:

```
SharedPreferences sharePref = getSharedPreferences(SETTINGS,
 MODE_PRIVATE);
```

###### 3.1.1.1.2. `public SharedPreferences getPreferences (int mode):`

- Phương thức này sẽ lấy về đối tượng `SharedPreferences` để đọc ghi dữ liệu lên file xml dùng lưu trữ Preferences của Activity hiện tại.
- Phương thức này sẽ gọi lại phương thức `getSharedPreferences(...)` kể trên với tham số `String name` là tên của Activity hiện tại. Tham số `int mode` của hai phương

- thúc `getPreferences` và `getSharedPreferences` là tương tự nhau.
- Ở lần đầu tiên tạo ra đối tượng `SharedPreference` nếu file xml dùng để lưu trữ dữ liệu chưa tồn tại thì hệ thống sẽ tự tạo file xml với tên là tên của Activity.

### 3.1.1.2. Ghi dữ liệu vào `SharedPreferences`

Các bước thực hiện:

- B1. Tạo đối tượng `SharedPreferences` bằng cách gọi hàm `getSharedPreferences`.
- B2. Gọi phương thức `SharedPreferences.edit()` để lấy về một đối tượng `SharedPreferences.Editor`. Ta sẽ sử dụng đối tượng này để ghi dữ liệu xuống file xml.
- B3. Thêm dữ liệu vào file xml bằng cách gọi các phương thức `SharedPreferences.Editor.putBoolean()`, `SharedPreferences.Editor.putString()`, ...
- B4. Gọi lệnh `SharedPreferences.commit()` để hoàn tất việc thay đổi nội dung và ghi xuống file xml.

### 3.1.1.3. Đọc dữ liệu từ `SharedPreferences`

Để đọc dữ liệu ta sử dụng một trong các phương thức sau:

- `SharedPreferences.getBoolean()`
- `SharedPreferences.getFloat`
- `SharedPreferences.getInt()`
- `SharedPreferences.getLong()`
- `SharedPreferences.getString()`, ...

Các phương thức này đều nhận 2 tham số, tham số thứ 1 là tên của `key`, tham số thứ 2 là giá trị mặc định. Giá trị mặc định sẽ được dùng nếu trước đó giá trị của `key` không được thiết lập.

Đoạn mã sau minh họa cách sử dụng

```
public static final String SETTINGS = "com.example.app_13.settings";
public static final String FIRST_USE = "com.example.app_13.firstUse";
SharedPreferences preferences = getSharedPreferences(SETTINGS, MODE_PRIVATE);
boolean firstUse = preferences.getBoolean(FIRST_USE, true);
if (firstUse)
{
 Editor editor = preferences.edit();
 editor.putBoolean(FIRST_USE, false);
 editor.commit();
}
```

`SharedPreferences` cung cấp cơ chế lưu trữ dưới dạng **key-value** và không cho phép người dùng trực tiếp thay đổi giá trị của `key`. Nhờ vậy, bạn có thể sử dụng `SharedPreferences` trong việc quản lý nhãn thời gian (timestamps) hoặc các theo dõi dạng đơn giản ứng dụng của mình như đếm số lần người dùng sử dụng ứng dụng rồi lưu trữ vào trong `SharedPreferences`. Dựa vào số đếm này để thông báo hoặc nhắc nhở người dùng thực hiện 1 công việc nào đó ...

Một ví dụ khác là bạn sử dụng thời gian trong hệ thống để tính khoảng thời gian mà người dùng đã và đang sử dụng ứng dụng. Lưu ý rằng nếu việc tính toán thời gian là quan trọng trong ứng dụng của bạn, khi đó cần xem xét kỹ việc sử dụng giờ hệ thống của thiết bị vì người dùng có thể thay đổi được thời gian trong 1 số ứng dụng khác của thiết bị.

Ví dụ sau lưu trữ giờ hiện tại với giá trị tính theo milli giây. Thời gian này có thể được đọc sau đó và so sánh với thời gian hiện tại như là 1 cách để tính toán lượng thời gian đã trôi qua.

```
public static final String TIMESTAMP = "com.example.app_13.timeStamp";
long timeStamp = System.currentTimeMillis();
SharedPreferences.Editor editor = preferences.edit();
editor.putLong(TIMESTAMP, timeStamp);
editor.commit();
```

### 3.1.1.4. Kiểu dữ liệu và phương thức trong `SharedPreferences`

Các kiểu dữ liệu thường dùng đều có thể được lưu trữ và truy xuất dễ dàng từ SharedPreferences thông qua các phương thức `getBoolean()`, `getFloat()`, `getInt()`, `getLong()`, `getString()`, và `getStringSet()`.

Ngoài những phương thức trên, Android còn cung cấp 2 phương thức nữa là:

- `getAll()`: lấy tất cả các cặp key-value như là 1 *Map*, là nơi các **key** sẽ được lưu trữ dưới dạng chuỗi và **value** là bất kỳ kiểu dữ liệu gì bạn đã đưa vào *SharedPreferences*.
- `contains()`: trả về trị true nếu tìm thấy key trong *SharedPreferences*.

## Lắng nghe các thay đổi trong *SharedPreferences*

*SharedPreferences* cung cấp phương thức `registerOnSharedPreferenceChangeListener()` cho phép bạn thực hiện lắng nghe các thay đổi tùy chọn và phản ứng nếu cần thiết. Tương ứng với phương thức này là phương thức `unregisterOnSharedPreferenceChangeListener()`.

### 3.1.2. Thiết lập các tùy chọn của người dùng (User Preferences)

Android sử dụng dữ liệu lưu trữ trong *SharedPreferences* cùng với tập các tùy chọn APIs để cung cấp mạnh mẽ các thiết lập của người dùng.

Bạn có thể sử dụng `PreferenceActivity(android.preference.PreferenceActivity)` hoặc `PreferenceFragment(android.preference.PreferenceFragment)` như là 1 giao diện người dùng cho các thiết lập của mình. Class `Preference` mở rộng cung cấp các class con để xử lý các kiểu dữ liệu cụ thể khi thiết lập. Các class con của `Preference` bao gồm các thuộc tính cho việc thiết lập và 1 giao diện người dùng cho việc hiển thị các tùy chọn như `CheckBoxPreference` (`android.preference.CheckBoxPreference`) giúp trình bày 1 checkbox và 1 thông báo đi kèm. Người dùng có thể chọn hoặc bỏ chọn. Giá trị chọn trên `CheckBoxPreference` có kiểu là `boolean` với `true` cho biết checkbox đã được chọn.

#### 3.1.2.1. Tạo ra *PreferencesFragment*

Khi tạo ra 1 `PreferenceFragment` hoặc `PreferenceActivity`, bạn sử dụng 1 file XML để định nghĩa các tùy chọn. File XML này không phải là layout của ứng dụng vì file layout giúp định nghĩa các view có trong ứng dụng, còn file XML này được dùng để định nghĩa các tùy chọn. Bạn cần tạo folder `res\XML` (nếu chưa có) để lưu trữ file XML này. File XML phải được đặt tên là `preferences.xml` để tạo ra 1 `PreferenceFragment`.

Nội dung của file `preferences.xml` xác định hình dáng của `SettingsFragment` khi hiển thị đến người dùng.

Đoạn mã lệnh sau minh họa toàn bộ class `SettingsFragment`. `SettingsFragment` được mở rộng từ `PreferenceFragment` và thực thi `onSharedPreferenceChangeListener`. Trong đó, **The `addPreferencesFromResource( )` method reads the settings definition from XML and inflates it into views in the current activity. All the heavy lifting takes place in the PreferenceActivity class.**

```
package com.example.app_13;
import android.content.SharedPreferences;
import android.content.SharedPreferences.OnSharedPreferenceChangeListener;
import android.os.Bundle;
import android.preference.PreferenceFragment;
import android.preference.PreferenceManager;
import android.util.Log;
public class SettingsFragment extends PreferenceFragment implements
 OnSharedPreferenceChangeListener
{
 @Override
 public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 addPreferencesFromResource(R.xml.preferences);
 }
 @Override
```

```

public void onResume()
{
 super.onResume();
 getPreferenceScreen().getSharedPreferences()
 .registerOnSharedPreferenceChangeListener(this);
}
@Override
public void onPause()
{
 super.onPause();
 getPreferenceScreen().getSharedPreferences()
 .unregisterOnSharedPreferenceChangeListener(this);
}
@Override
public void onSharedPreferenceChanged(SharedPreferences sharedPreferences,
 String key)
{
 Log.d("Settings", key);
}

```

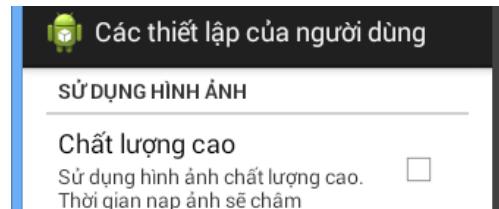
#### Các loại Preference được cho trong bảng sau:

Loại Preference	Kiểu của giá trị lưu trữ	API Level	Mô tả
CheckBoxPreference	Boolean	1	Hiển thị 1 CheckBox
EditTextPreference	String	1	Nhập dữ liệu vào EditText
ListPreference	String	1	Cho phép chọn 1 trong các mục có trong danh sách
MultiSelectListPreference	Tập các String	11	Cho phép chọn 1 hoặc nhiều trong các mục có trong danh sách
SwitchPreference	Boolean	14	Chuyển đổi giữa 2 dạng on và off

##### 3.1.2.1.1. CheckBoxPreference

CheckBoxPreference hiển thị CheckBox cho phép người dùng check chọn.

Minh họa sau chưa nội dung của file *preferences.xml* với cặp tag PreferenceScreen chứa 1 CheckBoxPreference với các thuộc tính được mô tả như key, tiêu đề, nội dung hiển thị giải thích ý nghĩa của mục chọn, giá trị mặc định khi khởi chạy lần đầu (vì có thể được thay đổi trong quá trình sử dụng bởi người dùng).



Hình 3-1 CheckBoxPreference

```

<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
 <CheckBoxPreference android:key="hires"
 android:title="Chất Lượng cao"
 android:summary="Sử dụng hình ảnh chất Lượng cao. Thời gian nạp ảnh sẽ chậm"
 android:defaultValue="False"/>
</PreferenceScreen>

```

Trong ứng dụng, SettingsFragment được sử dụng 1 activity khác với MainActivity (tạm gọi là SettingsActivity). SettingsActivity phải sử dụng phương thức setContentView() để hiển thị layout chưa SettingsFragment. Layout của SettingsActivity được trình bày thông qua file res\layout\activity\_settings.xml có dạng như sau:

```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 <fragment android:name="com.example.app_13.SettingsFragment"
 android:id="@+id/settings_fragment"
 android:layout_width="match_parent"
 android:layout_height="match_parent"/>

```

&lt;/RelativeLayout&gt;

Các phương thức `onCreateOptionsMenu()` và `onOptionsItemSelected()` được cài đặt trong MainActivity để hiển thị chuỗi Settings trong menu ở trên.

### 3.1.2.1.2. ListPreference

`ListPreference(android.preference.ListPreference)` giúp hiển thị 1 danh sách các mục chọn đến người dùng. Với danh sách các mục chọn phải được khai báo trước trong file res/values/strings.xml.

Tag `ListPreference` có một số thuộc tính như sau:

Thuộc tính	Mô tả
<code>android:key</code>	Thuộc tính khóa của mục tùy chọn. Thuộc tính này sẽ dùng để truy xuất giá trị của tùy chọn sau khi giá trị của nó được lưu trữ vào file xml.
<code>android:title</code>	Tiêu đề của mục tùy chọn.
<code>android:summary</code>	Mô tả tóm tắt của mục tùy chọn.
<code>android:entries</code>	Tập nhãn của các mục con có thể được gán cho mục tùy chọn.
<code>android:entryValues</code>	Tập giá trị của các mục con có thể được gán cho mục tùy chọn.
<code>android:dialogTitle</code>	Tiêu đề của hộp thoại hiển thị danh sách các giá trị của mục tùy chọn để người dùng lựa chọn.
<code>android.defaultValue</code>	Giá trị mặc định của mục tùy chọn.

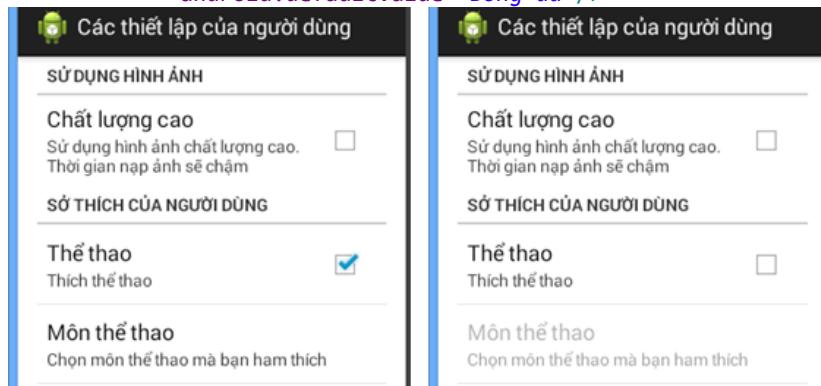
Ví dụ:

```
<string-array name="mon_the_thao">
 <item>Bóng đá</item>
 <item>Bóng chuyền</item>
 <item>Bóng rổ</item>
 <item>Bóng bàn</item>
 <item>Cầu lông</item>
</string-array>
```

Để điều khiển việc cho phép người dùng có được chọn danh sách hay không, ta sử dụng thuộc tính dependency, tại đó cần chỉ ra key của Preference mà ListPreference sẽ phụ thuộc vào (thường là 1 CheckBox)

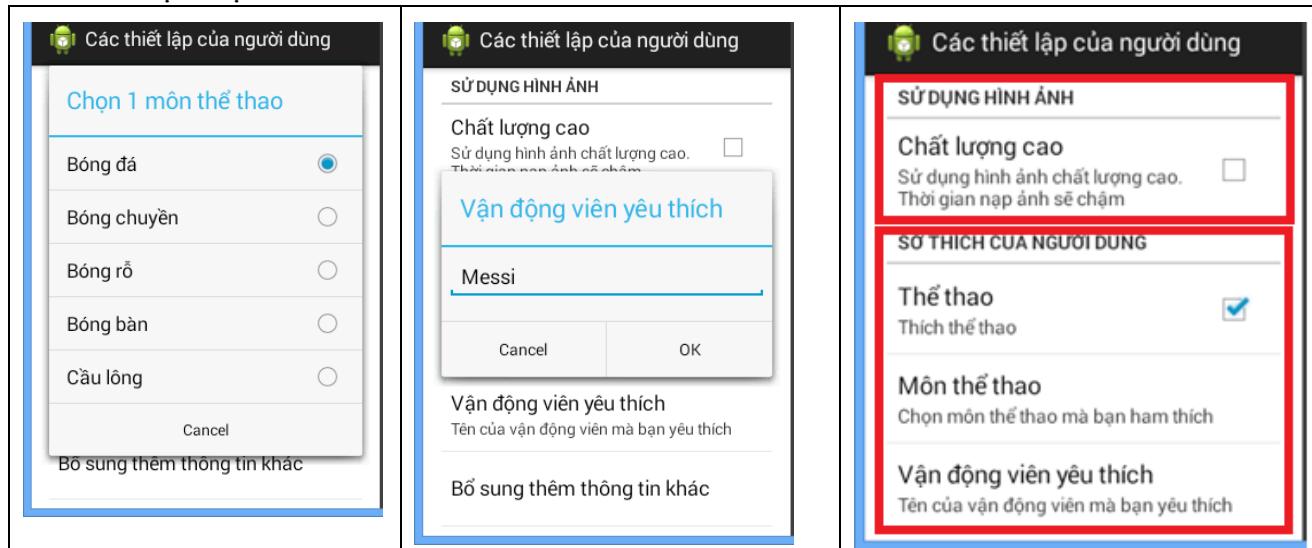
Minh họa sau mô phỏng thuộc tính của ListPreference vào 1 CheckBoxPreference

```
<CheckBoxPreference android:key="pie"
 android:title="Thể thao"
 android:summary="Thích thể thao"
 android:defaultValue="true"/>
<ListPreference android:dependency="pie"
 android:key="pie_type"
 android:title="Môn thể thao"
 android:summary="Chọn môn thể thao mà bạn ham thích"
 android:dialogTitle="Chọn 1 môn thể thao"
 android:entries="@array/mon_the_thao"
 android:entryValues="@array/mon_the_thao"
 android:defaultValue="Bóng đá"/>
```



Hình 3-2 ListPreference phụ thuộc trạng thái của CheckBoxPreference

Khi người dùng click chọn vào ListPreferences “Môn thể thao” sẽ xuất hiện danh sách các mục chọn như sau:



Hình 3-3 ListPreference

Hình 3-4 EditTextPreference

Hình 3-5 PreferenceCategory

### 3.1.2.1.3. EditTextPreference

*EditTextPreference* (`android.preference.EditTextPreference`) sử dụng để hiển thị một hộp thoại cho phép người dùng nhập vào một đoạn văn bản.

Mình họa sau, cho phép người dùng bổ sung thông tin thông qua 1 form khi click chọn vào *EditTextPreference* “Vận động viên yêu thích”.

```
<EditTextPreference
 android:key="more_info"
 android:title="Vận động viên yêu thích"
 android:summary="Tên của vận động viên mà bạn yêu thích"
 android:defaultValue="" />
```

### 3.1.2.1.4. PreferenceCategory

Việc thêm tiêu đề giúp nội dung của các thiết lập được rõ ràng hơn. Để nhóm các preference có liên quan vào cùng 1 tập hợp, bạn đặt các preference vào trong cặp tag *PreferenceCategory*(`android.preference.PreferenceCategory`) rồi sử dụng thuộc tính title của tag này. Ví dụ:

```
<PreferenceCategory xmlns:android="http://schemas.android.com/apk/res/android"
 android:title="Sở thích của người dùng">
<CheckBoxPreference android:key="pie"
 android:title="Thể thao"
 android:summary="Thích thể thao"
 android:defaultValue="true" />
<ListPreference android:dependency="pie"
 android:key="pie_type"
 android:title="Môn thể thao"
 android:summary="Chọn môn thể thao mà bạn ham thích"
 android:dialogTitle="Chọn 1 môn thể thao"
 android:entries="@array/mon_the_thao"
 android:entryValues="@array/mon_the_thao"
 android:defaultValue="Bóng đá" />
<EditTextPreference android:key="more_info"
 android:title="Vận động viên yêu thích"
 android:summary="Tên của vận động viên mà bạn yêu thích"
 android:defaultValue="" />
```

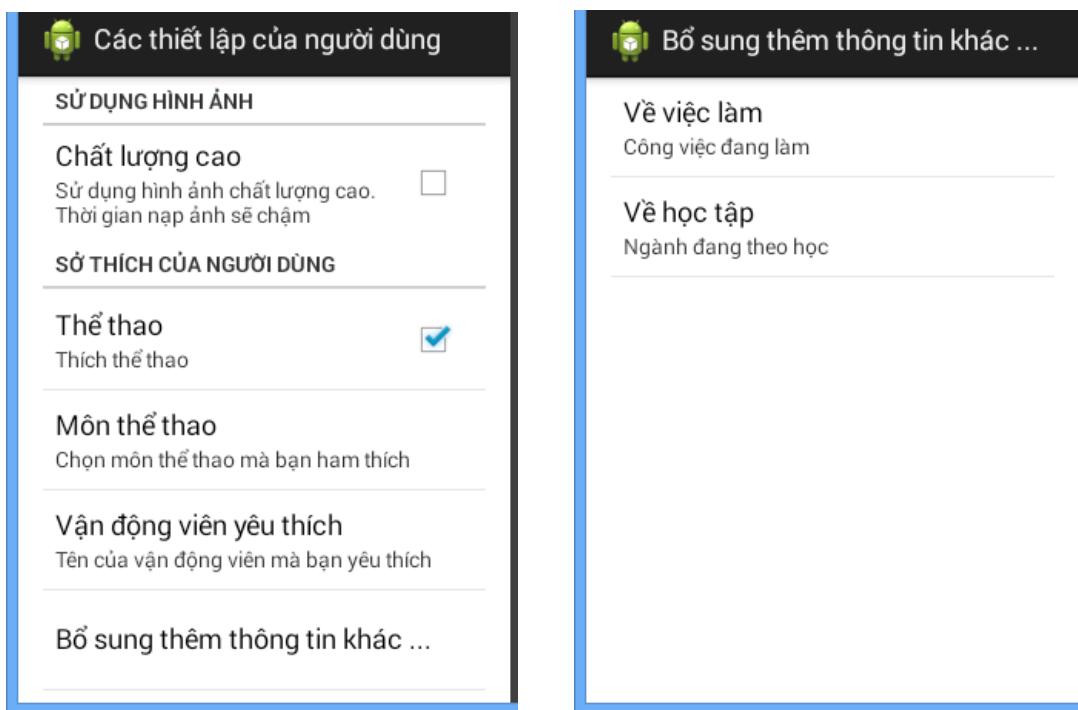
```
</PreferenceCategory>
```

#### 3.1.2.1.5. PreferenceScreen

Dùng để thêm màn hình con (sub Screen) của SharedPreferences, nhờ vậy giúp tổ chức lại nội dung màn hình chính thành 1 số màn hình con. Màn hình con có hình dạng tương tự như màn hình chính của Preferences nhưng được mở thông qua việc chọn một chức năng ở màn hình liền trước đó (thường là màn hình chính của Preferences).

Để định nghĩa 1 màn hình con lồng trong màn hình của Preferences, bạn sử dụng thêm phần tử (element) *PreferenceScreen*(*android.preference.PreferenceScreen*) trong XML và đặt phần tử này nằm gọn trong phần tử *PreferenceScreen* sẽ chứa nó. Ví dụ:

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen>
 <PreferenceCategory xmlns:android="http://schemas.android.com/apk/res/android"
 android:title="Sở thích của người dùng">
 <CheckBoxPreference android:key="pie"
 android:title="Thể thao"
 android:summary="Thích thể thao"
 android:defaultValue="true"/>
 <ListPreference android:dependency="pie"
 android:key="pie_type"
 android:title="Môn thể thao"
 android:summary="Chọn môn thể thao mà bạn ham thích"
 android:dialogTitle="Chọn 1 môn thể thao"
 android:entries="@array/mon_the_thao"
 android:entryValues="@array/mon_the_thao"
 android:defaultValue="Bóng đá"/>
 </PreferenceCategory>
 <PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android"
 android:key="second_preferencescreen"
 android:title="Bổ sung thêm thông tin khác ...">
 <EditTextPreference android:key="extraA"
 android:title="Về việc làm"
 android:summary="Công việc đang làm"
 android:defaultValue="" />
 <EditTextPreference android:key="ExtraB"
 android:title="Về học tập"
 android:summary="Ngành đang theo học"
 android:defaultValue="" />
 </PreferenceScreen>
</PreferenceScreen>
```



Hình 3-6 Sử dụng màn hình con giúp trình bày được nhiều nội dung hơn

### 3.1.2.1.6. RingtonePreference

RingtonePreference sử dụng để hiển thị một hộp lựa chọn nhạc chuông. Đường dẫn chỉ đến file nhạc chuông sẽ được lưu trữ xuống file xml.

### 3.1.2.1.7. Đọc thông tin từ Preferences

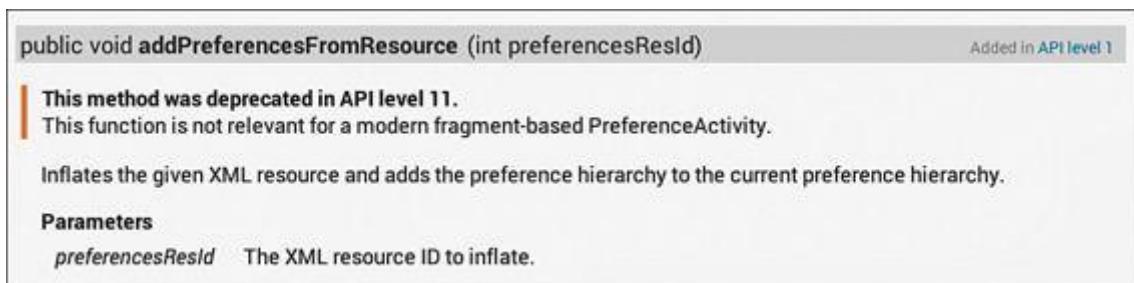
- Cách đọc các preferences tương tự như đối với SharedPreferences, nhưng bạn cần sử dụng PreferenceManager để lấy, ví dụ:  

```
SharedPreferences sharedPref =
 PreferenceManager.getDefaultSharedPreferences(getActivity());
```
- Sau đó, bạn có thể lấy dữ liệu bằng phương thức get tương ứng từ SharedPreferences, ví dụ:  

```
sharedPref.getString("pie_type", "");
```

### 3.1.2.2. Tạo một Preference Activity

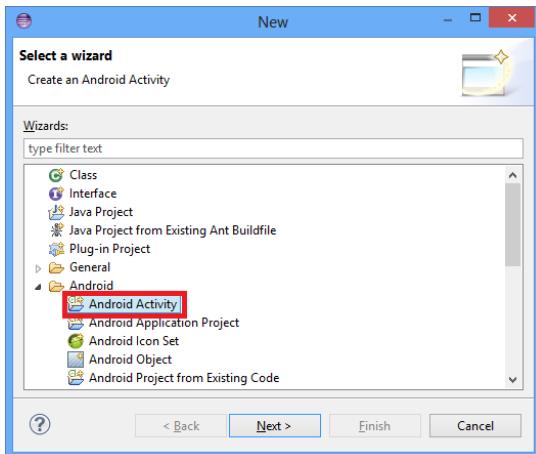
Một yếu tố trong việc sử dụng PreferenceFragment là có 1 số phương thức trong đó đã được đề nghị loại bỏ, Hình sau minh họa 1 thông báo về vấn đề này



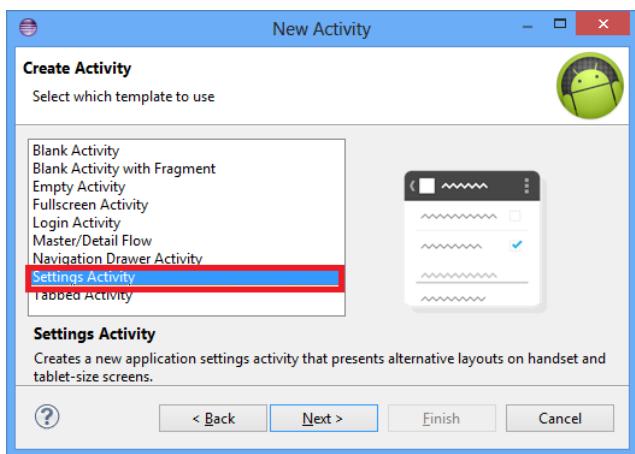
Hình 3-7 PreferenceActivity được đề nghị loại bỏ từ API Level 11

Mặt khác, Eclipse cung cấp chức năng giúp người dùng thuận tiện trong việc tạo SettingsActivity. Để tạo 1 project và thêm 1 SettingsActivity, bạn cần thực hiện theo các bước sau:

- Tạo mới 1 project với MainActivity (New\Android Application Project)
- Trong project mới tạo, chọn New\Other ... \Android Activity (hình 3-8).
- Trong hộp thoại New Activity, chọn Settings Activity, click Next rồi Finish để hoàn tất việc tạo Activity (hình 3-9).



Hình 3-8 Chọn AndroidActivity



Hình 3-9 Chọn Settings Activity trong New Activity

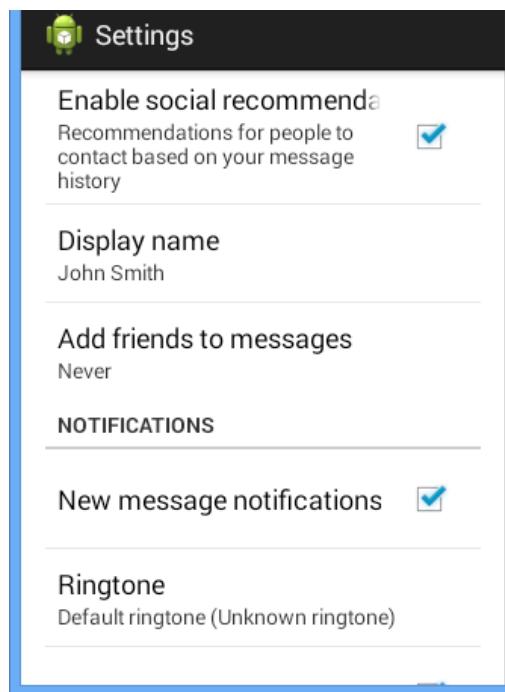
- Sau khi hoàn tất, Android đã thực hiện 1 số công việc như sau:
  - Tạo mới foleder res\xml
  - Thêm 1 số file mô tả giao diện của preferences.
- Bạn cần bổ sung lệnh sau trước tên phương thức bị cảnh báo về deprecated. Ví dụ:

```
@SuppressWarnings("deprecation")
private void setupSimplePreferencesScreen() {
 ...
}
```

- (iv). Bổ sung mã lệnh trong phương thức `onOptionsItemSelected()` của file `MainActivity.java` để gọi `SettingsActivity` khi người dùng click chọn vào menu Setting trên thiết bị.

```
@Override
public boolean onOptionsItemSelected(MenuItem item)
{
 switch (item.getItemId())
 {
 case R.id.menu_settings:
 Intent intent = new Intent(this, SettingsActivity.class);
 startActivity(intent);
 return true;
 default:
 return super.onOptionsItemSelected(item);
 }
}
```

- }
- (v). Với mã lệnh sẵn có, màn hình của Preferences khi người dùng click chọn menu Settings sẽ có dạng như hình 2-111:
- (vi). Bổ sung mã lệnh cho Activity (nếu cần thiết).

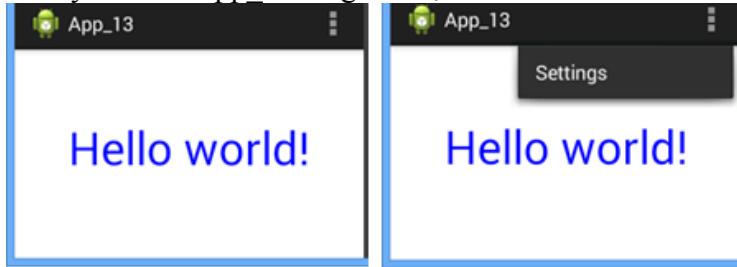


Hình 3-10 Màn hình Preferences sau khi click chọn menu Settings

## BÀI THỰC HÀNH App\_13

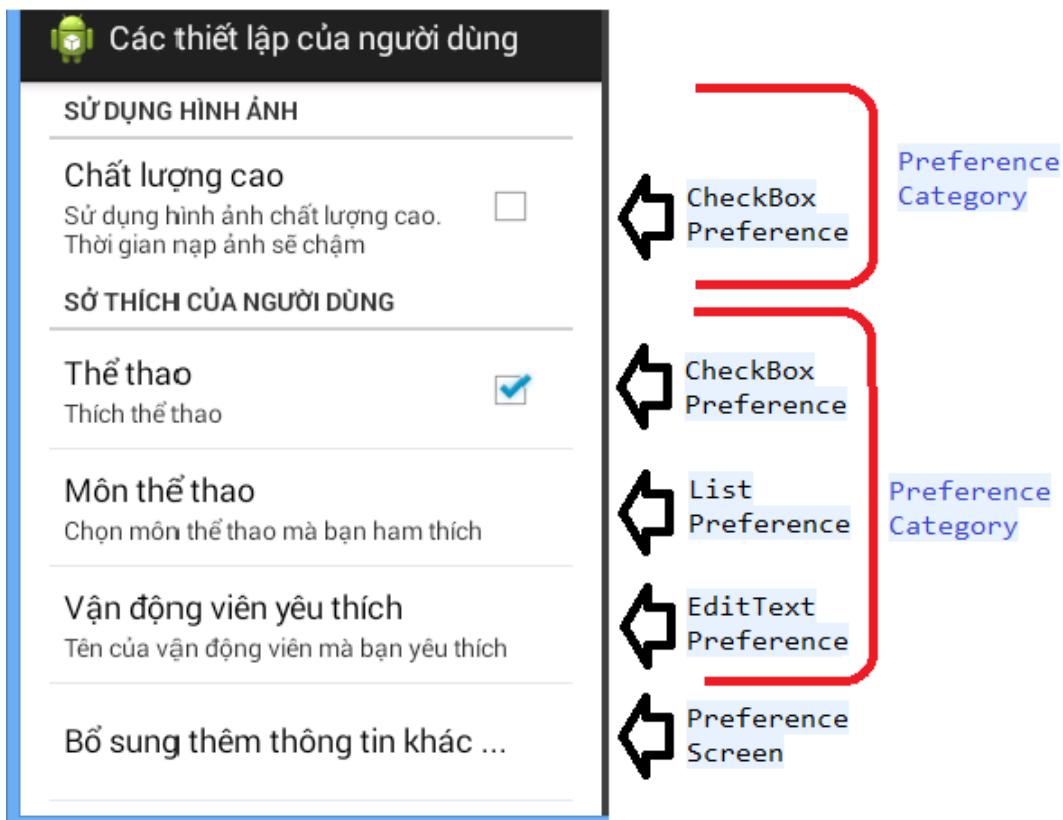
### ☞ Yêu cầu:

- Tạo mới activity với tên App\_13 có giao diện như hình 2-112.

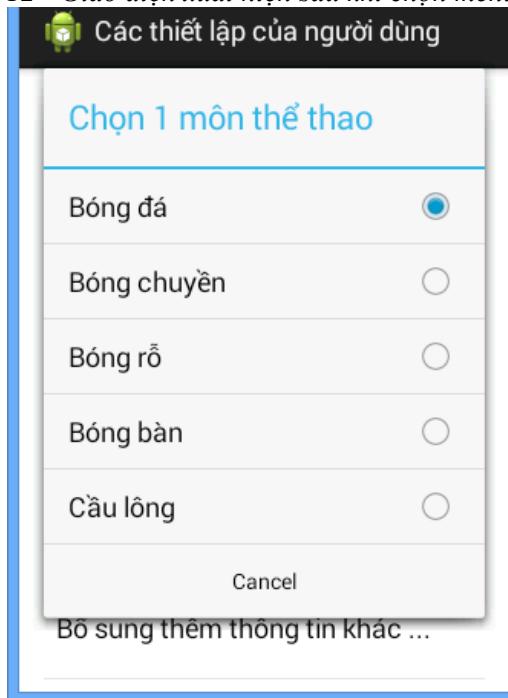


Hình 3-11 Giao diện của App\_13

- Khi ứng dụng được chạy lần đầu tiên, sẽ xuất hiện câu thông báo "Chúc mừng bạn là người đầu tiên khởi chạy ứng dụng". Các lần chạy sau sẽ không xuất hiện nữa.
- Khi click chọn menu Settings sẽ xuất hiện màn hình sau. Các control sử dụng theo yêu cầu như trong hình minh họa (hình 3-12)



Hình 3-12 Giao diện xuất hiện sau khi chọn menu Settings



Hình 3-13 Danh sách các mục chọn khi chọn "Môn thể thao"

## ☛ Thực hiện

**B1.-** Tạo mới project App\_13.

**B2.-** Bổ sung các chuỗi cần dùng trong chương trình vào file res\values\strings.xml để có nội dung như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
 <string name="app_name">App_13</string>
 <string name="hello_world">Hello world!</string>
 <string name="action_settings">Settings</string>
 <string name="title_activity_settings">Các thiết lập của người dùng</string>
```

```

<string-array name="mon_the_thao">
 <item>Bóng đá</item>
 <item>Bóng chuyền</item>
 <item>Bóng rổ</item>
 <item>Bóng bàn</item>
 <item>Cầu lông</item>
</string-array>
</resources>

```

- B3.-** Tao mới folder res\xml. Trong folder này, tạo mới 1 file XML có tên preferences.xml với nội dung như sau:

```

<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen>
 <PreferenceCategory xmlns:android="http://schemas.android.com/apk/res/android"
 android:title="Sử dụng hình ảnh">
 <CheckBoxPreference android:key="hires"
 android:title="Chất Lượng cao"
 android:summary="Sử dụng hình ảnh chất lượng cao."
 android:defaultValue="False"/>
 </PreferenceCategory>
 <PreferenceCategory xmlns:android="http://schemas.android.com/apk/res/android"
 android:title="Sở thích của người dùng">
 <CheckBoxPreference android:key="pie"
 android:title="Thể thao"
 android:summary="Thích thể thao"
 android:defaultValue="true"/>
 <ListPreference android:dependency="pie"
 android:key="pie_type"
 android:title="Môn thể thao"
 android:summary="Chọn môn thể thao mà bạn ham thích"
 android:dialogTitle="Chọn 1 môn thể thao"
 android:entries="@array/mon_the_thao"
 android:entryValues="@array/mon_the_thao"
 android:defaultValue="Bóng đá"/>
 <EditTextPreference android:key="more_info"
 android:title="Vận động viên yêu thích"
 android:summary="Tên của vận động viên mà bạn yêu thích"
 android:defaultValue=""/>
 </PreferenceCategory>
 <PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android"
 android:key="second_preferencescreen"
 android:title="Bổ sung thêm thông tin khác ...">
 <EditTextPreference android:key="extraA"
 android:title="Về việc Làm"
 android:summary="Công việc đang làm"
 android:defaultValue=""/>
 <EditTextPreference android:key="ExtraB"
 android:title="Về học tập"
 android:summary="Ngành đang theo học"
 android:defaultValue=""/>
 </PreferenceScreen>
</PreferenceScreen>

```

- B4.-** Tạo mới file layout với tên res\layout\activity\_settings.xml có nội dung như sau, trong đó chú ý đến dòng đặt tên cho fragment. Về sau, thông qua biến này sẽ giúp lưu trữ các thiết lập của người dùng:

```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 <fragment android:name="com.example.app_13.SettingsFragment"
 android:id="@+id/settings_fragment"

```

```

 android:layout_width="match_parent"
 android:layout_height="match_parent"/>
 </RelativeLayout>

```

**B5.-** Chính sửa file activity\_main.xml để có nội dung:

```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent" >
 <TextView android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_centerHorizontal="true"
 android:layout_centerVertical="true"
 android:text="@string/hello_world"
 android:layout_centerInParent="@string/hello_world"
 android:textSize="42sp"
 android:textColor="#0000FF"/>
</RelativeLayout>

```

**B6.-** Tạo mới file src\com.example.app\_13\SettingsFragment.java

```

package com.example.app_13;
import android.content.SharedPreferences;
import android.content.SharedPreferences.OnSharedPreferenceChangeListener;
import android.os.Bundle;
import android.preference.PreferenceFragment;
import android.preference.PreferenceManager;
import android.util.Log;
public class SettingsFragment extends PreferenceFragment implements
 OnSharedPreferenceChangeListener
{
 @Override
 public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 // gọi giao diện được mô tả trong file preferences.xml
 addPreferencesFromResource(R.xml.preferences);
 }
 @Override
 public void onResume()
 {
 super.onResume();
 getPreferenceScreen().getSharedPreferences()
 .registerOnSharedPreferenceChangeListener(this);
 }
 @Override
 public void onPause()
 {
 super.onPause();
 getPreferenceScreen().getSharedPreferences()
 .unregisterOnSharedPreferenceChangeListener(this);
 }
 // phương thức sau được gọi khi có sự thay đổi trên Preferences
 // Các thay đổi này có thể nhìn thấy được trên của sổ LogCat.
 @Override
 public void onSharedPreferenceChanged(SharedPreferences sharedPreferences, String
key)
 {
 Log.d("Settings", key);
 SharedPreferences sharedPref =
 PreferenceManager.getDefaultSharedPreferences(getActivity());
 if (key.equalsIgnoreCase("pie_type"))
 {
 Log.d("Settings", sharedPref.getString(key, ""));
 }
 }
}

```

**B7.-** Tạo mới file src\com.example.app\_13\SettingsActivity.java (trong phần này sẽ không sử dụng wizard như đã mô tả trong phần lý thuyết) với nội dung:

```

package com.example.app_13;
import android.os.Bundle;
import android.app.Activity;

public class SettingsActivity extends Activity
{
 public static final String SETTINGS = "com.example.app_13.settings";
 public static final String FIRST_USE = "com.example.app_13.firstUse";
 @Override
 protected void onCreate(Bundle savedInstanceState)
 {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_settings);
 }
}

```

**B8.-** Chính sửa nội dung file MainActivity.java

```

package com.example.app_13;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

public class MainActivity extends Activity
{
 public static final String SETTINGS = "com.example.app_13.settings";
 public static final String FIRST_USE = "com.example.app_13.firstUse";

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 SharedPreferences preferences = getSharedPreferences(SETTINGS, MODE_PRIVATE);
 /* do key có tên FIRST_USE chưa được gán giá trị nên ở lần chạy đầu tiên
 * của ứng dụng sẽ nhận giá trị là true */
 boolean firstUse = preferences.getBoolean(FIRST_USE, true);
 if (firstUse)
 {
 Toast helloMessage = Toast.makeText(getApplicationContext(),
 "Chúc mừng bạn là người đầu tiên khởi chạy ứng dụng", Toast.LENGTH_LONG);
 helloMessage.show();
 Editor editor = preferences.edit();
 editor.putBoolean(FIRST_USE, false);
 editor.commit();
 }
 }
 @Override
 public boolean onCreateOptionsMenu(Menu menu)
 {
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }
 @Override
 public boolean onOptionsItemSelected(MenuItem item) {
 switch (item.getItemId()) {
 case R.id.menu_settings:
 // Khởi chạy SettingsActivity
 Intent intent = new Intent(this, SettingsActivity.class);
 startActivity(intent);
 }
 }
}

```

```
 return true;
 default:
 return super.onOptionsItemSelected(item);
 }
}
```

- B9.-** Do SettingsActivity.java tự tạo thủ công mà không dùng wizard của Android nên cần khai báo activity này trong file AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.app_13"
 android:versionCode="1"
 android:versionName="1.0" >
 <uses-sdk
 android:minSdkVersion="11"
 android:targetSdkVersion="21" />
 <application
 android:allowBackup="true"
 android:icon="@drawable/ic_launcher"
 android:label="@string/app_name"
 android:theme="@style/AppTheme" >
 <activity
 android:name=".MainActivity"
 android:label="@string/app_name" >
 <intent-filter>
 <action android:name="android.intent.action.MAIN" />
 <category android:name="android.intent.category.LAUNCHER" />
 </intent-filter>
 </activity>
 <activity android:name=".SettingsActivity"
 android:label="@string/title_activity_settings">
 </activity>
 </application>
</manifest>
```

- B10.-** Chạy ứng dụng để xem kết quả.

## 3.2. *SQLite*

### 3.2.1. Giới thiệu

**SQLite** là 1 hệ quản trị cơ sở dữ liệu thu nhỏ, không có server, thao tác trên file và có thể sử dụng trong hầu hết các hệ điều hành. SQLite được Richard Hipp viết dưới dạng thư viện bằng ngôn ngữ lập trình C. Phiên bản mới nhất là 3.8.6 (15/8/2014).



Hình 3-14 Biểu trưng SQLite

### **3.2.1.1. Nhận định chung về SQLite**

### 3.2.1.1.1. Ưu điểm

- Tin cậy: các *transaction* trong cơ sở dữ liệu được thực hiện trọn vẹn, không gây lỗi khi xảy ra sự cố phần cứng.
  - Tuân theo chuẩn SQL92 (chỉ có một vài đặc điểm không hỗ trợ)
  - Không cần cài đặt, cấu hình
  - Kích thước chương trình gọn nhẹ (khoảng 300 kB)
  - Thực hiện các thao tác đơn giản nhanh hơn các hệ thống cơ sở dữ liệu khách/chủ khác.
  - Không cần phần mềm phụ trợ
  - Phần mềm tự do với mã nguồn mở, được chú thích rõ ràng.
  - Trong Android, chúng ta không cần cài đặt nhiều, chỉ cần cung cấp các hàm để thao tác và chương trình sẽ quản lý phần còn lại.

### 3.2.1.1.2. Một số hạn chế

- Không hỗ trợ 1 số tính năng của SQL 92

Tính năng	Điễn giải
RIGHT OUTER JOIN	Không hỗ trợ. Chỉ dùng được đối với LEFT OUTER JOIN.
FULL OUTER JOIN	Không hỗ trợ. Chỉ dùng được đối với LEFT OUTER JOIN.
ALTER TABLE	Chỉ hỗ trợ các biến thể về RENAME TABLE và ADD COLUMN của lệnh ALTER TABLE. Không hỗ trợ về DROP COLUMN, ALTER COLUMN, ADD CONSTRAINT.
Trigger support	Chỉ hỗ trợ phát biểu FOR EACH ROW nhưng hỗ trợ phát biểu FOR EACH STATEMENT.
VIEWS	VIEWS trong SQLite có dạng là read-only. Bạn không thể thực thi các phát biểu như DELETE, INSERT hoặc UPDATE trong view.

- Cơ sở dữ liệu do SQLite tạo ra sẽ private, tức là chỉ sử dụng cho bản thân ứng dụng.
- SQLite không hỗ trợ lệnh ALTER TABLE, do đó, bạn không thể thực hiện chỉnh sửa hoặc xóa cột trong bảng.
- SQLite không hỗ trợ ràng buộc khóa ngoại, các transactions lồng nhau, phép kết RIGHT OUTER JOIN.
- Vài tiến trình hoặc luồng có thể truy cập tới cùng một cơ sở dữ liệu. Việc đọc dữ liệu có thể chạy song song, còn việc ghi dữ liệu thì không được phép chạy đồng thời.

### 3.2.1.1.3. SQLite phù hợp trong các tình huống sau

- Ứng dụng sử dụng dạng flat file để lưu trữ dữ liệu: như từ điển, các ứng dụng nhỏ, lưu cấu hình ứng dụng... Nó mạnh mẽ hơn nhiều kỹ thuật lưu trữ file thông thường nhờ kỹ thuật hiện, dễ dùng và tin cậy hơn.
- Sử dụng trong các thiết bị nhúng: smart phone, PDA và các thiết bị di động.
- Các website có khoảng 100.000 lượt truy cập/ngày (về mặt lý thuyết thì SQLite có thể đáp ứng cao hơn nhiều).
- Sử dụng làm database tạm để lưu trữ dữ liệu lấy về từ các database trên SQL server, Oracle...
- Làm database demo cho các ứng dụng lớn, dùng để làm mô hình khái niệm cho ứng dụng.
- Sử dụng trong giảng dạy cho những người mới làm quen với ngôn ngữ truy vấn SQL. Ngoài các trường hợp trên, tốt hơn hết là bạn sử dụng SQL Server, Oracle, DB2, ...

## 3.2.1.2. Phân loại các lệnh tương tác với CSDL trong SQLite

### 3.2.1.2.1. DDL - Data Definition Language

Lệnh	Điễn giải
CREATE	Tạo mới table, view, hoặc các đối tượng khác trong cơ sở dữ liệu (CSDL)
ALTER	Hiệu chỉnh các đối tượng đã tồn tại trong CSDL (như table)
DROP	Xóa table, view hoặc các đối tượng khác đã tồn tại trong (CSDL)

### 3.2.1.2.2. DML - Data Manipulation Language

Lệnh	Điễn giải
INSERT	Thêm mới dòng (record) vào table
UPDATE	Hiệu chỉnh dữ liệu của các records
DELETE	Xóa records

### 3.2.1.2.3. DQL - Data Query Language

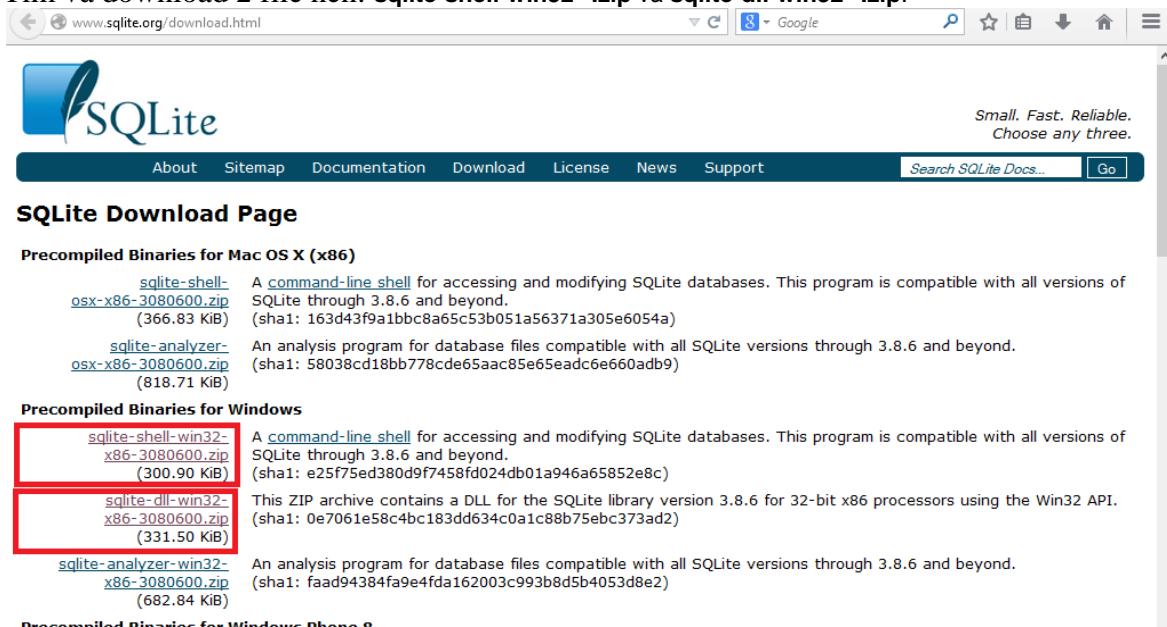
Lệnh	Điễn giải
SELECT	Lấy dữ liệu từ một hoặc nhiều table

### 3.2.2. Cài đặt SQLite trên Windows

#### 3.2.2.1. Cài đặt SQLite

B1. Truy cập trang <http://www.sqlite.org/download.html>

B2. Tìm và download 2 file nén: **sqlite-shell-win32-\*.zip** và **sqlite-dll-win32-\*.zip**.

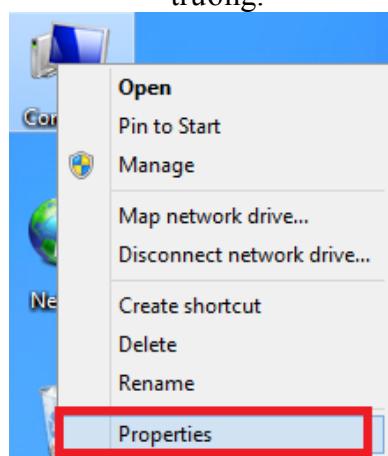


Hình 3-15 website của SQLite

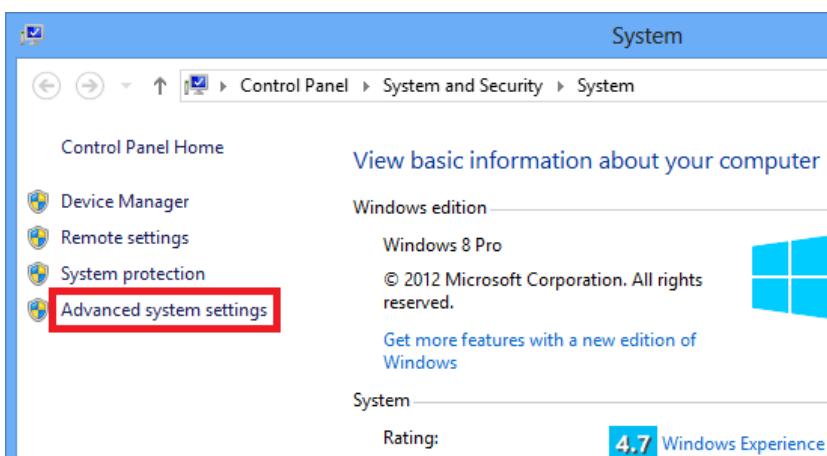
B3. Giải nén 2 file vừa download được: Để dễ quản lý (không bắt buộc), bạn nên tạo folder với tên sqlite trong cùng folder với eclipse và SDK, rồi giải nén 2 file vào đó. Kết quả sẽ thu được 3 file sqlite3.def, sqlite3.dll và sqlite3.exe. Copy đường dẫn để sử dụng cho bước kế tiếp

B4. Thêm đường dẫn vào biến môi trường PATH của Windows:

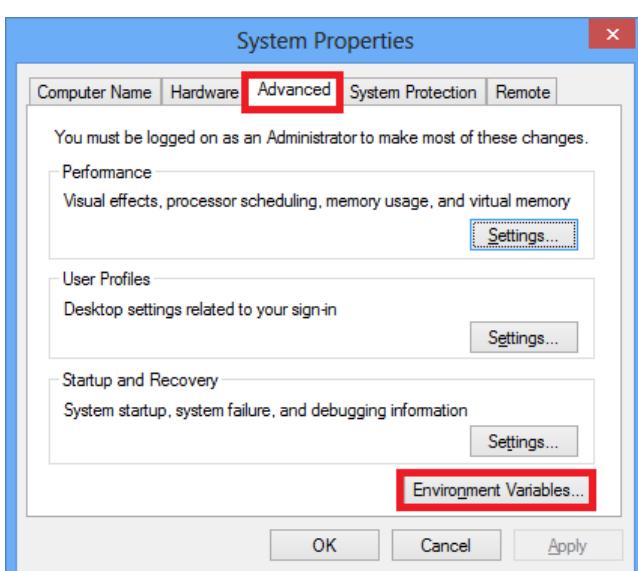
- **B4.1:** Right Click vào icon của Computer trên desktop, chọn Properties.
- **B4.2:** Trong hộp thoại System, chọn Advanced system settings.
- **B4.3:** Trong hộp thoại System Properties, chọn tab Advanced, chọn button Environment Variables ... để mở tiếp hộp thoại Environment Variables.
- **B4.4:** Trong hộp thoại Environment Variables, trong vùng System variables, tìm và chọn biến Path, xong click button Edit để mở hộp thoại Edit Environment Variables.
- **B4.5:** Click mouse vào textbox. Di chuyển về cuối, bổ sung dấu chấm phẩy (;) rồi paste đường dẫn đã copy ở B3 vào đây.
- **B4.6:** Lần lượt click chọn OK trong từng hộp thoại để hoàn tất việc bổ sung biến môi trường.



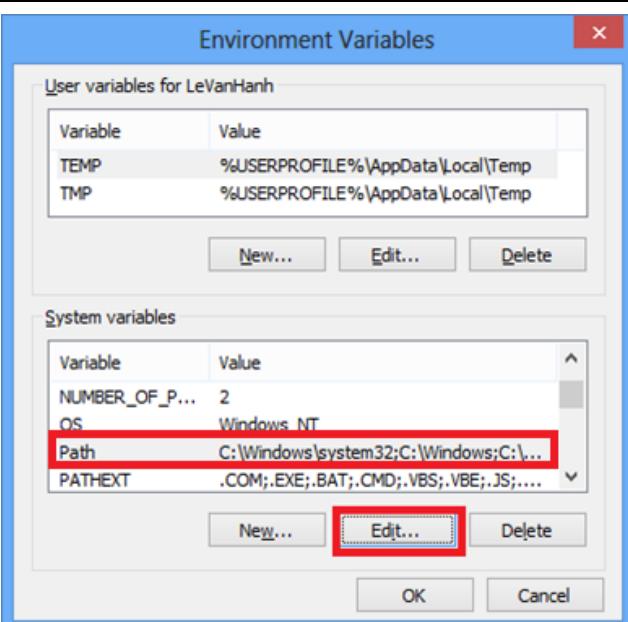
Hình 3-16 Bước 4.1



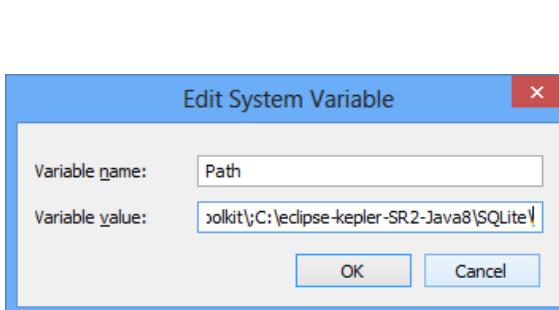
Hình 3-17 Bước 4.2



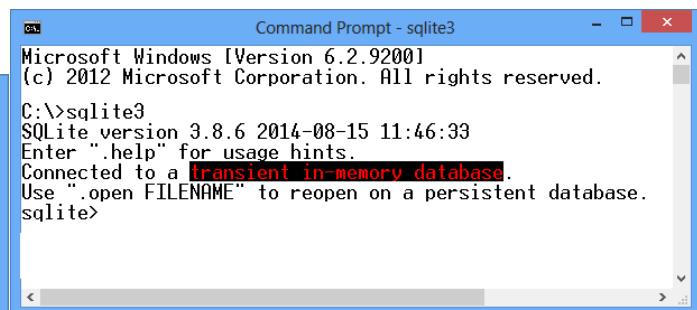
Hình 3-18 Bước 4.3



Hình 3-19 Bước 4.4



Hình 3-20 Bước 4.4 sau khi chọn button Edit

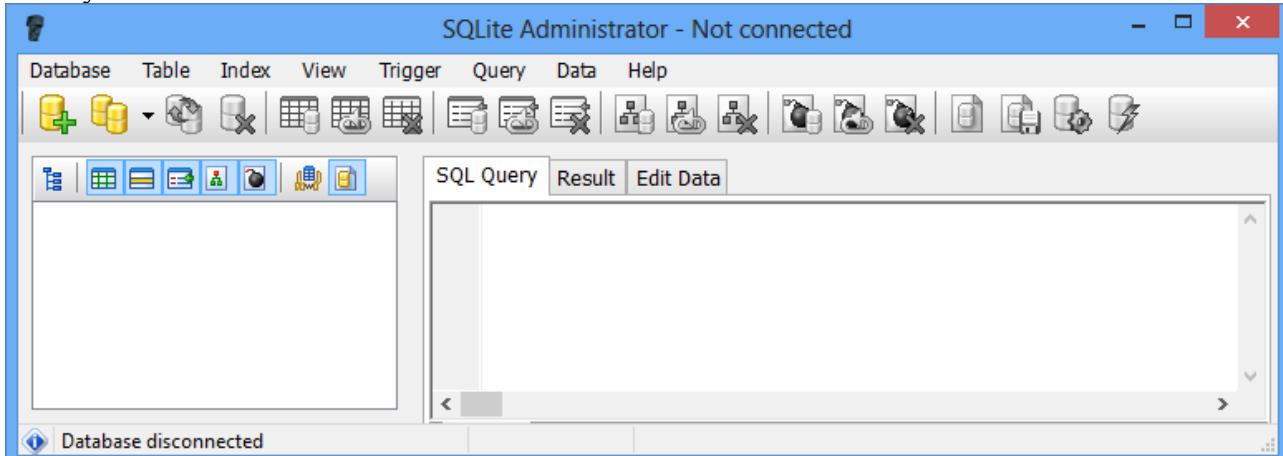


Hình 3-21 Bước 5

**B5.** Kiểm tra kết quả thực hiện: mở cửa sổ cmd của Windows, gõ lệnh sqlite3 sẽ xuất hiện kết quả như hình minh họa

### 3.2.2.2. Sử dụng SQLite Administrator

Có thể sử dụng lệnh SQLite trong cửa sổ command của Windows. Tuy nhiên, bạn nên dùng công cụ SQLite Administrator (download <http://download.orbm2k.de/files/sqliteadmin.zip>). Sau khi download hoàn tất, giải nén vào 1 folder nào đó. Tìm và chạy (double click) file sqliteadmin.exe để thấy cửa sổ sau:



Hình 3-22 Giao diện của SQLite Administrator

Tuy rất tiện ích trong việc quản lý CSDL, nhưng trong tài liệu này không hướng dẫn cách sử dụng ứng dụng *SQLite Administrator*. Bạn có thể tự tìm hiểu vì cách sử dụng ứng dụng này tương đối giống như trong SQL Server của Microsoft.

### 3.2.3. Kiểu dữ liệu trong SQLite

Mỗi cột, biến và biểu thức có liên quan đến dữ liệu trong SQLite đều phải có thuộc tính để xác định kiểu dữ liệu.

#### 3.2.3.1. Các Classes lưu trữ kiểu dữ liệu

- Mỗi giá trị được lưu trữ trong CSDL của SQLite đều phải thuộc 1 trong những class lưu trữ sau:

Class lưu trữ	Điều giải
NULL	Chứa giá trị NULL
INTEGER	Là giá trị nguyên, có dấu được lưu trữ trong 1, 2, 3, 4, 6, hoặc 8 bytes tùy thuộc vào độ lớn của dữ liệu.
REAL	Là giá trị số thực, được lưu trữ trong 8-byte.
TEXT	Là chuỗi ký tự, lưu trữ bằng cách mã hóa CSDL (UTF-8, UTF-16BE hoặc UTF-16LE)
BLOB	Giá trị là 1 cả 1 khối (blob) của dữ liệu, sẽ được lưu đầy đủ theo cả khối.

- Tên của các kiểu dữ liệu trong SQLite:

Class lưu trữ	Các kiểu dữ liệu
INTEGER	INT, INTEGER, TINYINT, SMALLINT, MEDIUMINT, BIGINT, UNSIGNED BIG INT, INT2, INT8
REAL	REAL, DOUBLE, DOUBLE PRECISION, FLOAT
TEXT	CHARACTER(20), VARCHAR(255), VARYING CHARACTER(255), NCHAR(55), NATIVE CHARACTER(70), NVARCHAR(100), TEXT CLOB
BLOB	(không chỉ ra kiểu dữ liệu cụ thể)
NUMERIC	NUMERIC, DECIMAL(10,5), BOOLEAN, DATE, DATETIME

#### 3.2.3.2. Kiểu luận lý (Boolean Datatype)

SQLite không có class lưu trữ riêng cho kiểu dữ liệu luận lý. Thay vào đó, giá trị luận lý được lưu trữ dưới dạng số nguyên với 0 là *false* và 1 là *true*.

#### 3.2.3.3. Kiểu ngày và giờ (Date and Time Datatype)

SQLite cũng không có class lưu trữ riêng cho kiểu dữ liệu ngày/giờ, nhưng SQLite có khả năng lưu trữ ngày/giờ như là TEXT, REAL hoặc INTEGER.

Class lưu trữ	Định dạng
TEXT	Ngày được định dạng theo dạng thức "YYYY-MM-DD HH:MM:SS.SSS".
REAL	Số ngày tính từ 24 November 24 năm 4714 trước Công nguyên
INTEGER	Số lượng thời gian tính từ 1970-01-01 00:00:00 UTC.

Bạn có thể chọn kiểu ngày/giờ bất kỳ trong các định dạng trên. Khi sử dụng SQLite sẽ cung cấp 1 số hàm giúp chuyển đổi giữa các định dạng.

### 3.2.4. Lệnh SQLite

#### 3.2.4.1. Một số lưu ý

- Case Sensitivity: SQLite có phân biệt chữ hoa/thường (case insensitive). Mặt khác, SQLite có trường hợp cùng 1 tên nhưng khi viết hoa và khi viết thường là 2 lệnh có ý nghĩa khác nhau như lệnh GLOB và lệnh glob.

- Sử dụng ghi chú trong SQLite:
  - Giúp giải thích hoặc ghi chú 1 vấn đề nào đó trong mã lệnh.
  - Nội dung trong ghi chú có thể chứa khoảng trắng hoặc các ký tự khác nhưng không thể đặt các ghi chú lồng vào nhau.
  - Sử dụng:
    - Ghi chú theo dòng: trong SQLite bắt đầu bằng hai liên tiếp "--" ký tự (ASCII 0x2D).
    - Ghi chú theo đoạn (theo phong cách của ngôn ngữ C): các ghi chú được đặt trong cặp ký hiệu /\* và \*/.

```
sqlite>.help -- This is a single line comment
```
- SQLite Statements (phát biểu)
  - Tất cả các phát biểu trong SQLite đều bắt đầu với một từ khóa như SELECT, INSERT, CREATE, UPDATE, DELETE, ALTER, DROP, ... và đều phải kết thúc bằng dấu chấm phẩy (,).
  - Mỗi phát biểu có thể được trình bày trên nhiều dòng.

### 3.2.4.2. Các lệnh SQLite thường dùng

Những lệnh này được gọi là dấu chấm lệnh SQLite và ngoại lệ với các lệnh này là chúng không được kết thúc bằng một dấu chấm phẩy (;).

Để tiện theo dõi, bạn hãy khởi động cmd trong Windows. Sau đó gõ lệnh sqlite3 tại dấu nhắc để có kết quả:

```
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.

C:\>sqlite3
SQLite version 3.8.6 2014-08-15 11:46:33
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> -
```

Hình 3-23 Dòng cuối cùng trong hình là dấu nhắc lệnh của SQLite

Các lệnh thường dùng:

<b>Lệnh</b>	<b>Diễn giải</b>
.backup ?DB? FILE	Backup DB (default "main") thành FILE
.bail ON OFF	Dừng lại sau khi chạm một lỗi. Mặc định là OFF
.databases	Liệt kê tên và các file CSDL đính kèm (attached)
.dump ?TABLE?	Dump the database in an SQL text format. If TABLE specified, only dump tables matching LIKE pattern TABLE. “Đỗ cơ sở dữ liệu sang định dạng văn bản SQL. Nếu tên TABLE được chỉ ra, lệnh chỉ thực hiện với bảng đó
.echo ON OFF	Bật/tắt lệnh echo
.exit	Thoát khỏi dấu nhắc lệnh của SQLite
.explain ON OFF	Bật/tắt EXPLAIN mode. Nếu không có đối số, mặc định sẽ là ON
.header(s) ON OFF	Bật/tắt việc hiển thị tiêu đề cột của table
.help	Liệt kê các lệnh trong SQLite
.import FILE TABLE	Import dữ liệu từ FILE vào TABLE
.indices ?TABLE?	Hiển thị tên của tất cả các chỉ mục. Nếu tên TABLE được chỉ ra, chỉ liệt kê chỉ mục có trong TABLE
.load FILE ?ENTRY?	Nạp thư viện mở rộng
.log FILE off	Bật/tắt log.

.mode MODE	Thiết lập output mode, trong đó MODE thuộc 1 trong những giá trị sau: <ul style="list-style-type: none"> <li>▫ <b>csv</b> các giá trị ngăn cách nhau bởi dấu phẩy</li> <li>▫ <b>column</b> canh lề trái cho các cột</li> <li>▫ <b>html</b> HTML &lt;table&gt; code</li> <li>▫ <b>insert</b> SQL insert statements for TABLE</li> <li>▫ <b>line</b> One value per line</li> <li>▫ <b>list</b> Values delimited by .separator string</li> <li>▫ <b>tabs</b> Tab-separated values</li> <li>▫ <b>tcl</b> TCL list elements</li> </ul>
.nullvalue STRING	Print STRING in place of NULL values
.output FILENAME	Send output to FILENAME
.output stdout	Send output to the screen
.print STRING...	Print literal STRING
.prompt MAIN	
CONTINUE	Thay thế dấu nháy lệnh của sqlite
.quit	thoát khỏi dấu nháy lệnh của sqlite
.read FILENAME	Execute SQL in FILENAME
.schema ?TABLE?	Hiển thị nội dung của phát biểu CREATE. Nếu tên TABLE được chỉ ra, sẽ hiển thị phát biểu CREATE của TABLE có dạng tương tự như TABLE có tên được chỉ ra.
.separator STRING	Thay đổi dấu ngăn cách sẽ sử dụng cho output mode và .import
.show	Hiển thị giá trị hiện tại của các thiết lập
.stats ON OFF	Bật/tắt thiết lập stats
.tables ?PATTERN?	Liệt kê tên những table có dạng tương tự như mẫu PATTERN
.timeout MS	Try opening locked tables for MS milliseconds
.width NUM NUM	Thiết lập độ rộng cột cho mode "column"
.timer ON OFF	Bật/tắt thiết lập timer của CPU

Sử dụng lệnh `.show` để xem xem các thiết lập mặc định:

```
sqlite>.show
echo: off
explain: off
headers: off
mode: column
nullvalue: ""
output: stdout
separator: "|"
width:
sqlite>
```

Không được có khoảng trắng giữa dấu nháy lệnh `sqlite>` và lệnh cần thực hiện.

### 3.2.4.3. Định dạng dữ liệu xuất

Có thể sử dụng các lệnh của SQLite để định dạng dữ liệu cần xuất. Ví dụ:

```
sqlite>.header on
sqlite>.mode column
sqlite>.timer on
sqlite>
```

Với định dạng vừa có, kết quả hiển thị sẽ có dạng như sau:

ID	NAME	AGE	ADDRESS	SALARY
-----	-----	-----	-----	-----

1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Jim	22	South-Hall	45000.0
7	Jimmy	24	Houston	10000.0
CPU Time: user 0.000000 sys 0.000000				

### 3.2.4.4. Table sqlite\_master

Table chính lưu giữ tất cả các thông tin về CSDL của bạn có tên là sqlite\_master. Bạn có thể xem các thành phần của table này bằng lệnh .schema như sau:

```
sqlite>.schema sqlite_master
```

Kết quả hiển thị có dạng:

```
CREATE TABLE sqlite_master (type text,
 name text,
 tbl_name text,
 rootpage integer,
 sql text
);
```

## 3.2.5. Toán tử trong SQLite(SQLite Operators)

### 3.2.5.1. Toán tử số học (Arithmetic Operators)

Toán tử	Điễn giải
+	Cộng
-	Trừ
*	Nhân
/	Chia
%	Lấy phần dư trong phép chia giữa 2 số nguyên

#### Ví dụ

```
sqlite> .mode line
sqlite> select 10 + 20;
10 + 20 = 30

sqlite> select 10 - 20;
10 - 20 = -10

sqlite> select 10 * 20;
10 * 20 = 200

sqlite> select 10 / 5;
10 / 5 = 2

sqlite> select 12 % 5;
12 % 5 = 2
```

### 3.2.5.2. Toán tử so sánh (Comparison Operators)

Toán tử	Điễn giải
== Or =	So sánh bằng
!= Or <>	So sánh khác
>	So sánh lớn hơn
<	So sánh nhỏ hơn
>=	So sánh lớn hơn hay bằng
<=	So sánh nhỏ hơn hay bằng
!<	So sánh nếu không nhỏ hơn
!>	So sánh nếu không lớn hơn

Ví dụ: với dữ liệu trong table COMPANY như sau:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Jim	22	South-Hall	45000.0
7	Jimmy	24	Houston	10000.0

Giả sử cần tìm những nhân viên có lương (SALARY) lớn hơn 50.000. Kết quả thu được:

sqlite> SELECT * FROM COMPANY WHERE SALARY > 50000;				
ID	NAME	AGE	ADDRESS	SALARY
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0

### 3.2.5.3. Toán tử luận lý (Logical Operators)

Toán tử	Điễn giải
AND	Và
BETWEEN min AND max	Chọn những giá trị nằm trong khoảng từ min đến max
EXISTS	Chọn những dòng có giá trị hiện diện trong một table được chỉ ra với các tiêu chí nhất định.
IN	Chọn những dòng có giá trị xuất hiện trong danh sách được liệt kê sau toán tử IN
NOT IN	Chọn những dòng không có giá trị xuất hiện trong danh sách được liệt kê sau toán tử IN
LIKE	So sánh giá trị theo 1 dạng thức cho trước. LIKE không phân biệt chữ hoa/thường (case sensitive)
GLOB	So sánh giá trị theo 1 dạng thức cho trước. GLOB có phân biệt chữ hoa/thường.
NOT	Là phép phủ định. Thường dùng đi liền trước các toán tử luận lý khác như NOT EXISTS, NOT BETWEEN, NOT IN, ...
OR	Hoặc
IS NULL	So sánh 1 giá trị với giá trị NULL
IS	Thực hiện tương tự như toán tử bằng (=)
IS NOT	Thực hiện tương tự như toán tử khác (!=)
	Tạo 1 chuỗi mới bằng cách cộng 2 chuỗi tham gia vào toán tử.
UNIQUE	Tìm những dòng chỉ xuất hiện duy nhất 1 lần trong table

Ví dụ: tương tự, với dữ liệu trong table COMPANY như sau:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Jim	22	South-Hall	45000.0
7	Jimmy	24	Houston	10000.0

– Tìm những dòng có Name bắt đầu bằng 2 ký tự ‘Ji’, ký tự thứ 3 là bất kỳ.

sqlite> SELECT * FROM COMPANY WHERE NAME LIKE 'Ki%';				
ID	NAME	AGE	ADDRESS	SALARY
6	Jim	22	South-Hall	45000.0

– Tìm những dòng có Name bắt đầu bằng 2 ký tự ‘Ji’, các ký tự đi sau là bất kỳ.

sqlite> SELECT * FROM COMPANY WHERE NAME LIKE 'Ki%';				
------------------------------------------------------	--	--	--	--

ID	NAME	AGE	ADDRESS	SALARY
6	Jim	22	South-Hall	45000.0
7	Jimmy	24	Houston	10000.0

- Tìm những dòng có AGE là 1 trong 2 giá trị 25 hoặc 27:

sqlite> SELECT * FROM COMPANY WHERE AGE IN (25, 27);				
ID	NAME	AGE	ADDRESS	SALARY
2	Allen	25	Texas	15000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0

- Tìm những dòng có AGE không chứa cả 2 giá trị 25 và 27:

sqlite> SELECT * FROM COMPANY WHERE AGE NOT IN (25, 27);				
ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
3	Teddy	23	Norway	20000.0
6	Jim	22	South-Hall	45000.0
7	Jimmy	24	Houston	10000.0

- Tìm những dòng có AGE nằm trong khoảng từ 25 đến 27:

sqlite> SELECT * FROM COMPANY WHERE AGE BETWEEN 25 AND 27;				
ID	NAME	AGE	ADDRESS	SALARY
2	Allen	25	Texas	15000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0

### 3.2.5.4. Toán tử trên bit (Bitwise Operators)

Toán tử hoạt động trên bit và thực hiện so sánh từng cặp bit với nhau.

P	Q	p & q	p   q
0	0	0	0
0	1	0	1
1	1	1	1
1	0	0	1

SQLite hỗ trợ các toán tử bit như sau:

Toán tử	Điễn giải
&	Và giữa 2 bit giá trị
	Hoặc giữa 2 bit giá trị
~	Phủ định các bit có trong giá trị đi sau toán tử
<<	Dịch trái các bit theo số lần được chỉ ra ngay sau toán tử. Mỗi lần dịch trái sẽ tương đương với việc nhân đôi giá trị của số đặt bên trái của toán tử
>>	Dịch phải các bit theo số lần được chỉ ra ngay sau toán tử. Mỗi lần dịch phải sẽ tương đương với việc chia lấy phần nguyên của số đặt bên trái của toán tử

Giả sử cho A = 60 và B = 13. Thực hiện các toán tử trên bit, ta được kết quả:

A=60 $\Rightarrow$	0	0	1	1	1	1	0	0	=60
B=13 $\Rightarrow$	0	0	0	0	1	1	0	1	=13
A&B $\Rightarrow$	0	0	0	0	1	1	0	0	=12
A B $\Rightarrow$	0	0	1	1	1	1	0	1	=61
~A $\Rightarrow$	1	1	0	0	0	0	1	1	=195
A<<2 $\Rightarrow$	1	1	1	1	0	0	0	0	=240
A>>2 $\Rightarrow$	0	0	0	0	1	1	1	1	=15

### 3.2.6. Biểu thức trong SQLite (SQLite Expressions)

Một biểu thức là sự kết hợp của một hoặc nhiều giá trị, toán tử hoặc các hàm trong SQLite để đánh giá một giá trị là đúng hay sai.

#### 3.2.6.1. Cú pháp

```
SELECT column1, column2, columnN
FROM table_name
WHERE [CONDITION | EXPRESSION];
```

#### 3.2.6.2. Phân loại biểu thức

##### 3.2.6.2.1. Biểu thức luận lý (Boolean Expressions)

Tìm những dòng dữ liệu phù hợp với điều kiện đưa ra. Cú pháp có dạng:

```
SELECT column1, column2, columnN
FROM table_name
WHERE SINGLE VALUE MATCHTING EXPRESSION;
```

**Ví dụ:** tương tự, với dữ liệu trong table COMPANY như sau:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Jim	22	South-Hall	45000.0
7	Jimmy	24	Houston	10000.0

Cho biết thông tin về những nhân viên có SALARY=10.000:

```
sqlite> SELECT * FROM COMPANY WHERE SALARY = 10000;
ID NAME AGE ADDRESS SALARY
----- ----- ----- ----- -----
7 Jimmy 24 Houston 10000.0
```

##### 3.2.6.2.2. Biểu thức số học

- Là những biểu thức thực hiện bất kỳ những phép tính toán có trong bất kỳ vị trí nào của câu truy vấn:

```
SELECT numerical_expression as OPERATION_NAME
[FROM table_name WHERE CONDITION] ;
```

- Ví dụ:

- Tính tổng 2 số 15 và 6

```
sqlite> SELECT (15 + 6) AS ADDITION ADDITION = 21
```

- Đếm số record có trong table COMPANY

```
sqlite> SELECT COUNT(*) AS "RECORDS" FROM COMPANY; RECORDS = 7
```

##### 3.2.6.2.3. Biểu thức ngày

- Là những biểu thức trả về ngày/giờ của hệ thống và kết quả đó có thể tham gia vào các thao tác về dữ liệu khác.

- Ví dụ:

```
sqlite> SELECT CURRENT_TIMESTAMP;
CURRENT_TIMESTAMP = 2014-10-03 10:43:35
```

### 3.2.7. Các lệnh liên quan đến CSDL

#### 3.2.7.1. Tạo mới CSDL (Database)

Mặc định, CSDL được tạo ra sẽ được lưu tại:

DATA/data/APP\_NAME/databases/DATABASE\_NAME

Trong đó: □ DATA: là đường dẫn *Environment.getDataDirectory()*

- APP\_NAME: tên của ứng dụng
- DATABASE\_NAME: tên của CSDL được tạo ra

### 3.2.7.1.1. Sử dụng tại dấu nhắc lệnh của SQLite

- Người dùng không cần phải có bất kỳ đặc quyền đặc biệt để tạo ra một CSDL.

#### Cú pháp

```
sqlite3 DatabaseName.db
```

Tên cơ sở dữ liệu luôn phải là duy nhất trong RDBMS (không được trùng tên giữa các CSDL).

- **Ví dụ:** giả sử folder hiện hành là C:\sqlite. Cân tạo mới CSDL có tên là testDB  
C:\sqlite>sqlite3 testDB.db

- Lệnh trên sẽ tạo ra một file testDB.db trong thư mục hiện hành (C:\sqlite).
- Nếu cần tạo tiếp 1 CSDL khác, bạn cần thoát khỏi dấu nhắc lệnh của sqlite bằng lệnh .quit, chuyển đến folder khác để tạo vì 1 folder chỉ có thể chứa 1 CSDL

- **Lệnh .database:** giúp liệt kê các CSDL hiện có

```
sqlite>.databases
seq name file
--- -----
0 main C:\sqlite\testDB.db
```

- **Lệnh .dump:** giúp chuyển đổi 1 CSDL hoàn chỉnh trong 1 file dạng văn bản. Lệnh phải thực hiện trên dấu nhắc lệnh của hệ điều hành chứ không thể thực hiện trong dấu nhắc lệnh của sqlite

```
C:\sqlite>sqlite3 testDB.db .dump > testDB.sql
```

Lệnh trên sẽ chuyển đổi toàn bộ nội dung của cơ sở dữ liệu SQLite testDB.db vào file văn bản ASCII testDB.sql. Nhờ vậy, bạn có thể làm phục hồi CSDL testDB.db từ file testDB.sql đã được tạo ra như sau:

```
C:\sqlite>sqlite3 testDB.db < testDB.sql
```

### 3.2.7.1.2. Sử dụng bằng mã lệnh

Để tạo ra một CSDL, bạn chỉ cần gọi phương thức *openOrCreateDatabase* này với tham số là tên CSDL của bạn và mode tạo lập. phương thức này trả về một thê hiện của CSDL SQLite. Cú pháp sử dụng như sau:

```
SQLiteDatabase mydatabase = openOrCreateDatabase("your database name",
 MODE_PRIVATE, null);
```

Ngoài ra, có các chức năng khác trong gói databseae cùng làm công việc này, gồm:

- **openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags, DatabaseErrorHandler errorHandler)**  
Phương thức này chỉ mở ra CSDL hiện có với chế độ cờ thích hợp. Chế độ cờ phổ biến có thể là OPEN\_READWRITE OPEN\_READONLY
- **openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags)**  
Tương tự như phương thức trên vì cũng mở ra CSDL hiện có, chỉ khác là phương thức này không xác định bất kỳ xử lý để xử lý các lỗi CSDL.
- **openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory)**  
Phương thức này không chỉ mở mà còn có khả năng tạo ra CSDL nếu nó không tồn tại.
- **openOrCreateDatabase(File file, SQLiteDatabase.CursorFactory factory)**  
Cũng tương tự như phương thức trên nhưng thay vì dùng đối số có kiểu là String, phương thức này sử dụng File. Có thể dùng file.getPath() thay cho đối số File.

### 3.2.7.2. Attach Database

- Gắn kèm 1 CSDL vào SQLite.

#### - Cú pháp

```
ATTACH DATABASE 'DatabaseName' AS 'Alias-Name';
```

- Lệnh trên gắn CSDL *DatabaseName* vào SQLite với tên là Alias-Name.
- Nếu không tìm thấy CSDL *DatabaseName*, SQLite sẽ tạo ra 1 CSDL mới.

#### - Ví dụ: đính kèm 1 CSDL hiện có testDB.db vào SQLite với bí danh là TEST:

```
sqlite> ATTACH DATABASE 'testDB.db' AS 'TEST';
```

Sau đó sử dụng lệnh `.database` để hiển thị các CSDL hiện có (bao gồm cả những CSDL được attached).

```
sqlite> .database
seq name file
--- -----
0 main C:\sqlite\testDB.db
2 test C:\sqlite\testDB.db
```

Trong SQLite, 2 tên *main* và *TEMP* được dành riêng cho các cơ sở dữ liệu chính và cơ sở dữ liệu để giữ các bảng tạm thời. Vì vậy không được đặt bí danh cho các file gắn kèm với 1 trong 2 tên này.

```
sqlite> ATTACH DATABASE 'testDB.db' AS 'TEMP';
Error: database TEMP is already in use
sqlite> ATTACH DATABASE 'testDB.db' AS 'main';
Error: database TEMP is already in use
```

### 3.2.7.3. Detach Database

- Sử dụng để tách 1 CSDL đã được gắn kèm trước đó bằng lệnh `ATTACH DATABASE`.
- Nếu cùng 1 CSDL nhưng được gắn kèm với nhiều bí danh, lệnh `DETACH` sẽ thực hiện ngắt kết nối với tên bí danh được chỉ ra, các bí danh còn lại vẫn được sử dụng bình thường
- Không thể tách các cơ sở dữ liệu *main* hoặc *TEMP*.
- Nếu thực hiện `DETACH` với CSDL được lưu trong bộ nhớ hoặc CSDL tạm, CSDL sẽ bị phá hủy và các nội dung sẽ bị mất.

#### - Cú pháp

```
DETACH DATABASE 'Alias-Name';
```

Trong đó *Alias-Name* là tên bí danh mà bạn đã sử dụng khi gắn CSDL bằng phát biểu `ATTACH`.

#### - Ví dụ: giả sử khi sử dụng lệnh `.database`, ta được

```
sqlite>.databases
seq name file
--- -----
0 main /home/sqlite/testDB.db
2 test /home/sqlite/testDB.db
3 currentDB /home/sqlite/testDB.db
```

Sử dụng lệnh `DETACH` để tách '*currentDB*' từ testDB.db như sau:

```
sqlite> DETACH DATABASE 'currentDB';
```

Sau đó sử dụng lại lệnh `.database`, ta được kết quả:

```
sqlite>.databases
seq name file
--- -----
0 main /home/sqlite/testDB.db
2 test /home/sqlite/testDB.db
```

## 3.2.8. Các lệnh liên quan đến cấu trúc của TABLE

### 3.2.8.1. Create Table

### 3.2.8.1.1. Công dụng

Tạo mới 1 table với tên và kiểu dữ liệu của các cột được chỉ ra.

### 3.2.8.1.2. Cú pháp

```
CREATE TABLE database_name.table_name(column1 datatype
 PRIMARY KEY(one or more columns),
 column2 datatype,
 column3 datatype,

 columnN datatype,
) ;
```

Trong đó *database\_name* được dùng khi tồn tại nhiều CSDL .

- **Ví dụ:** tạo ra table COMPANY với cột ID có kiểu dữ liệu là số nguyên, được dùng làm khóa chính, do đó cần ràng buộc thêm NOT NULL.

```
sqlite> CREATE TABLE COMPANY(ID INT PRIMARY KEY NOT NULL,
 NAME TEXT NOT NULL,
 AGE INT NOT NULL,
 ADDRESS CHAR(50),
 SALARY REAL);
```

Tạo thêm table DEPARTMENT:

```
sqlite> CREATE TABLE DEPARTMENT(ID INT PRIMARY KEY NOT NULL,
 DEPT CHAR(50) NOT NULL,
 EMP_ID INT NOT NULL);
```

### 3.2.8.1.3. AutoIncrement

AUTOINCREMENT là từ khóa được sử dụng để tự động tăng một giá trị của một thuộc tính (field) trong table. AUTOINCREMENT chỉ được dùng với thuộc tính có kiểu là số nguyên và thường sử dụng kèm với lệnh tạo lập table.

- **Cú pháp**

```
CREATE TABLE table_name(column1 INTEGER AUTOINCREMENT,
 column2 datatype,

 columnN datatype);
```

- **Ví dụ**

Tạo table với AUTOINCREMENT:

```
sqlite> CREATE TABLE COMPANY(ID INTEGER PRIMARY KEY AUTOINCREMENT,
 NAME TEXT NOT NULL,
 AGE INT NOT NULL,
 ADDRESS CHAR(50),
 SALARY REAL);
```

Insert dữ liệu vào table COMPANY:

```
INSERT INTO COMPANY (NAME,AGE,ADDRESS,SALARY)
VALUES ('Paul', 32, 'California', 20000.00);
INSERT INTO COMPANY (NAME,AGE,ADDRESS,SALARY)
VALUES ('Allen', 25, 'Texas', 15000.00);
INSERT INTO COMPANY (NAME,AGE,ADDRESS,SALARY)
VALUES ('Teddy', 23, 'Norway', 20000.00);
INSERT INTO COMPANY (NAME,AGE,ADDRESS,SALARY)
VALUES ('Mark', 25, 'Rich-Mond ', 65000.00);
INSERT INTO COMPANY (NAME,AGE,ADDRESS,SALARY)
VALUES ('David', 27, 'Texas', 85000.00);
INSERT INTO COMPANY (NAME,AGE,ADDRESS,SALARY)
VALUES ('Kim', 22, 'South-Hall', 45000.00);
INSERT INTO COMPANY (NAME,AGE,ADDRESS,SALARY)
VALUES ('James', 24, 'Houston', 10000.00);
```

Dữ liệu của table COMPANY sau khi insert:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Kim	22	South-Hall	45000.0
7	James	24	Houston	10000.0

### 3.2.8.1.4. Các lệnh khác liên quan đến table:

- **.tables**: Xem danh sách các table đang có

```
sqlite>.tables
COMPANY DEPARTMENT
```

- **.schema**: xem lại nội dung lệnh đã tạo ra cấu trúc của table đang có

```
sqlite>.schema COMPANY
CREATE TABLE COMPANY(ID INT PRIMARY KEY NOT NULL,
 NAME TEXT NOT NULL,
 AGE INT NOT NULL,
 ADDRESS CHAR(50),
 SALARY REAL);
```

### 3.2.8.2. Drop Table

- Xóa table cùng tất cả các kết hợp đã có (data, indexes, triggers, constraints và quyền hạn được cấp trên table). Vì vậy bạn cần cân nhắc trước khi xóa 1 table.
- Không thể sử dụng DROP TABLE để xóa các ràng buộc (constraints).

- **Cú pháp**

```
DROP TABLE database_name.table_name;
```

- **Ví dụ**

Liệt kê danh sách các table đang có (giả sử đang có 1 table duy nhất là COMPANY)

```
sqlite>.tables
COMPANY
sqlite>
```

Xóa table COMPANY

```
sqlite>DROP TABLE COMPANY;
sqlite>
```

Xem lại danh sách các table

```
sqlite>.tables
sqlite>
```

Kết quả không hiển thị gì cả vì danh sách table hiện là rỗng (trống).

### 3.2.8.3. Alter Table

- Dùng để đổi tên table, thêm mới cột vào table hoặc có thể dùng để thay đổi tên cột.
- Không thể sử dụng ALTER TABLE để đổi tên các ràng buộc (constraints).
- Cú pháp:

- Thêm mới cột vào table

```
ALTER TABLE table_name ADD COLUMN column_def...;
```

- Đổi tên cột

```
ALTER TABLE table_name RENAME TO new_table_name;
```

### 3.2.8.4. Ràng buộc dữ liệu trong SQLite (Constraints)

Constraint là các quy tắc thực thi trên dữ liệu của table, giúp đảm bảo tính chính xác và độ tin cậy của dữ liệu trong CSDL.

#### 3.2.8.4.1. PRIMARY KEY Constraint

- Ràng buộc PRIMARY KEY xác định duy nhất mỗi record trong một table của CSDL.
- Khi cài đặt, một table bắt buộc phải có duy nhất 1 khóa chính.
- Khóa chính có thể là sự kết hợp của 1 hay nhiều cột.
- Trong SQLite khóa chính có thể được chứa giá trị NULL.

**Ví dụ:** Tạo table COMPANY gồm 5 cột ID, NAME, AGE, ADDRESS và SALARY, trong đó các cột ID được chọn làm khóa chính

```
CREATE TABLE COMPANY (ID INT PRIMARY KEY NOT NULL,
 NAME TEXT NOT NULL,
 AGE INT NOT NULL,
 ADDRESS CHAR(50),
 SALARY REAL);
```

#### 3.2.8.4.2. NOT NULL Constraint

Theo mặc định, 1 cột có thể chứa giá trị NULL. Để ràng buộc 1 cột không được phép sử dụng giá trị NULL, bạn cần sử dụng ràng buộc loại này cho cột đó.

NULL không được xem là không có dữ liệu, thay vào đó nó diễn tả dữ liệu chưa được biết.

**Ví dụ:** bổ sung lệnh tạo table COMPANY ở trên sao cho các cột ID, NAME và AGE không được chứa giá trị NULL:

```
CREATE TABLE COMPANY (ID INT PRIMARY KEY NOT NULL,
 NAME TEXT NOT NULL,
 AGE INT NOT NULL,
 ADDRESS CHAR(50),
 SALARY REAL);
```

#### 3.2.8.4.3. DEFAULT Constraint

Ràng buộc DEFAULT cung cấp giá trị mặc định cho cột khi lệnh INSERT INTO không cung cấp giá trị cho cột đó.

**Ví dụ:** bổ sung vào lệnh tạo table COMPANY ở trên và thiết lập giá trị mặc định cho cột SALARY là 5.000:

```
CREATE TABLE COMPANY (ID INT PRIMARY KEY NOT NULL,
 NAME TEXT NOT NULL,
 AGE INT NOT NULL,
 ADDRESS CHAR(50),
 SALARY REAL DEFAULT 50000.00);
```

#### 3.2.8.4.4. UNIQUE Constraint

- Dùng để ngăn cản 2 record giá trị giống hệt nhau trong một cột cụ thể.
- Có thể xuất hiện nhiều ràng buộc UNIQUE trên cùng 1 table (mỗi ràng buộc được gắn cho 1 cột).
- UNIQUE được chứa giá trị NULL.

**Ví dụ:** bổ sung vào lệnh tạo table COMPANY ở trên và thiết lập cột ADDRESS không được trùng nhau

```
CREATE TABLE COMPANY (ID INT PRIMARY KEY NOT NULL,
 NAME TEXT NOT NULL,
 AGE INT NOT NULL,
 ADDRESS CHAR(50) UNIQUE,
 SALARY REAL DEFAULT 50000.00);
```

#### 3.2.8.4.5. CHECK Constraint

Ràng buộc CHECK cho phép kiểm tra giá trị được nhập vào record phải tuân thủ theo một điều kiện nào đó. Nếu dữ liệu nhập vào vi phạm ràng buộc CHECK, record đó sẽ không được nhập vào table.

**Ví dụ:** bổ sung lệnh tạo table COMPANY để cột AGE phải nhận giá trị  $\geq 18$

```
CREATE TABLE COMPANY (ID INT PRIMARY KEY NOT NULL,
```

```

 NAME TEXT NOT NULL,
 AGE INT NOT NULL CHECK (AGE>=18),
 ADDRESS CHAR(50) UNIQUE,
 SALARY REAL DEFAULT 50000.00);

```

### 3.2.9. Lệnh Insert Into

- Thêm 1 dòng dữ liệu mới vào table.

#### - Cú pháp

- Dạng 1:

```

INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)
VALUES (value1, value2, value3,...valueN);

```

Trong đó, column1, column2,...columnN là tên các cột có trong table.

- Dạng 2:

```

INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);

```

Sử dụng khi thứ tự dữ liệu đưa vào đúng với thứ tự khi tạo lập table.

#### - Ví dụ

- Tạo table COMPANY trong CSDL testDB.db:

```

sqlite> CREATE TABLE COMPANY(ID INT PRIMARY KEY NOT NULL,
 NAME TEXT NOT NULL,
 AGE INT NOT NULL,
 ADDRESS CHAR(50) ,
 SALARY REAL);

```

- Sử dụng dạng 1 để thêm mới 6 record vào table COMPANY:

```

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Paul', 32, 'California', 20000.00);
INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (2, 'Allen', 25, 'Texas', 15000.00);
INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (3, 'Teddy', 23, 'Norway', 20000.00);
INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00);
INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (5, 'David', 27, 'Texas', 85000.00);
INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (6, 'Jim', 22, 'South-Hall', 45000.00);

```

- Sử dụng dạng 2 để thêm 1 record vào table COMPANY:

```

INSERT INTO COMPANY VALUES (7, 'Jimmy', 24, 'Houston', 10000.00);

```

#### - Insert với nguồn dữ liệu được lấy từ table khác:

- Cú pháp:

```

INSERT INTO destination_table_name [(column1, column2, ... columnN)]
SELECT column1, column2, ...columnN
FROM source_table_name
[WHERE condition];

```

### 3.2.10. Lệnh truy vấn dữ liệu

Lệnh SELECT được sử dụng để lấy dữ liệu từ 1 table trong SQLite và trả về dữ liệu dưới dạng bảng kết quả. Các bảng kết quả còn được gọi là tập kết quả (result-sets).

#### 3.2.10.1. Cú pháp đơn giản

```

SELECT [DISTINCT] column1, column2, columnN FROM table_name;

```

Trong đó, column1, column2, ... là tên những cột cần lấy và những tên cột này phải có trong table. Khi bạn cần lấy tất cả các cột, sử dụng cú pháp sau:

```

SELECT * FROM table_name;

```

**Ví dụ**

Giả sử trong table COMPANY đã có dữ liệu như sau:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Jim	22	South-Hall	45000.0
7	Jimmy	24	Houston	10000.0

- Lấy tất cả các cột có trong table với yêu cầu xuất dữ liệu theo dạng bảng và bao gồm cả tên các cột:

```
sqlite>.header on
sqlite>.mode column
sqlite> SELECT * FROM COMPANY;
```

Kết quả thu được:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Jim	22	South-Hall	45000.0
7	Jimmy	24	Houston	10000.0

- Sau đó, nếu chỉ cần lấy thông tin về 1 số cột có trong table:

```
sqlite> SELECT ID, NAME, SALARY FROM COMPANY;
```

Kết quả thu được:

ID	NAME	SALARY
1	Paul	20000.0
2	Allen	15000.0
3	Teddy	20000.0
4	Mark	65000.0
5	David	85000.0
6	Jim	45000.0
7	Jimmy	10000.0

- DISTINCT**

- Giúp loại bỏ những dòng trùng nhau (trên tất cả những cột trong SELECT).
- Ví dụ:

```
sqlite> SELECT DISTINCT AGE FROM COMPANY;
```

Kết quả thu được:

AGE
32
25
23
27
22
24

### 3.2.10.2. Mệnh đề LIMIT

- Dùng để giới hạn số lượng dòng được trả về trong tập kết quả.
- OFFSET *n***: nếu có sử dụng sẽ lấy từ record có số thứ tự thứ *n* (dòng đầu tính từ 0)
- Cú pháp:

```
SELECT column1, column2, columnN
FROM table_name
LIMIT [no of rows] OFFSET [row num]
```

**Ví dụ**

Giả sử trong table COMPANY đã có dữ liệu như sau:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Jim	22	South-Hall	45000.0
7	Jimmy	24	Houston	10000.0

- Lấy 6 record đầu tiên:

```
sqlite> SELECT * FROM COMPANY LIMIT 6;
```

Kết quả thu được:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Jim	22	South-Hall	45000.0

- Lấy 3 record đầu tiên tính từ dòng thứ 2 (dòng đầu tính từ 0):

```
sqlite> SELECT * FROM COMPANY LIMIT 3 OFFSET 2;
```

Kết quả thu được:

ID	NAME	AGE	ADDRESS	SALARY
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0

### 3.2.10.3. Mệnh đề FROM

Mệnh đề FROM giúp chỉ ra tên các table cần lấy dữ liệu phục vụ cho câu truy vấn.

Khi có nhiều table cùng tham gia trong câu truy vấn, các table này cần thực hiện phép kết (JOIN). Trong SQLite hỗ trợ các phép kết sau:

#### 3.2.10.3.1. CROSS JOIN

- CROSS JOIN thực hiện kết trên mọi record của table thứ 1 với tất cả các record của bảng thứ hai.
- Bảng kết quả sẽ có:
  - Số lượng cột = tổng số lượng các cột của 2 table thành phần.
  - Số lượng dòng = tích giữa số lượng các dòng của 2 table thành phần.
- Do CROSS JOIN có khả năng tạo ra các bảng rất lớn, vì vậy chỉ nên sử dụng khi cần thiết.
- Cú pháp:

```
SELECT ... FROM table1 CROSS JOIN table2 ...
```

- Ví dụ: Giả sử trong table COMPANY đã có dữ liệu như sau:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Jim	22	South-Hall	45000.0
7	James	24	Houston	10000.0

Và table DEPARTMENT được định nghĩa như sau:

```
CREATE TABLE DEPARTMENT(ID INT PRIMARY KEY NOT NULL,
 DEPT CHAR(50) NOT NULL,
 EMP_ID INT NOT NULL);
```

Với dữ liệu được thêm vào thông qua lệnh INSERT như sau:

```
INSERT INTO DEPARTMENT (ID, DEPT, EMP_ID)
VALUES (1, 'IT Billing', 1);
INSERT INTO DEPARTMENT (ID, DEPT, EMP_ID)
VALUES (2, 'Engineering', 2);
INSERT INTO DEPARTMENT (ID, DEPT, EMP_ID)
VALUES (3, 'Finance', 7);
```

Vậy dữ liệu có trong table DEPARTMENT hiện tại là:

ID	DEPT	EMP_ID
1	IT Billing	1
2	Engineering	2
3	Finance	7

Với lệnh sau:

```
sqlite> SELECT EMP_ID, NAME, DEPT FROM COMPANY CROSS JOIN DEPARTMENT;
```

Kết quả thu được

EMP_ID	NAME	DEPT
1	Paul	IT Billing
2	Paul	Engineering
7	Paul	Finance
1	Allen	IT Billing
2	Allen	Engineering
7	Allen	Finance
1	Teddy	IT Billing
2	Teddy	Engineering
7	Teddy	Finance
1	Mark	IT Billing
2	Mark	Engineering
7	Mark	Finance
1	David	IT Billing
2	David	Engineering
7	David	Finance
1	Kim	IT Billing
2	Kim	Engineering
7	Kim	Finance
1	James	IT Billing
2	James	Engineering
7	James	Finance

### 3.2.10.3.2. INNER JOIN

- INNER JOIN tạo ra một bảng kết quả mới bằng cách dò tìm rồi kết hợp các giá trị cột của hai bảng (table1 và table2) dựa trên điều kiện kết hợp.
- INNER JOIN được dùng khá phổ biến nhờ không gây ra hiện tượng “bùng nổ” kết hợp như các loại JOIN khác.
- Cú pháp:  

```
SELECT ... FROM table1 [INNER] JOIN table2 ON conditional_expression ...
```
- Ví dụ:

```
sqlite> SELECT EMP_ID, NAME, DEPT FROM COMPANY INNER JOIN DEPARTMENT
ON COMPANY.ID = DEPARTMENT.EMP_ID;
```

Kết quả thu được

EMP_ID	NAME	DEPT

1	Paul	IT Billing
2	Allen	Engineerin
7	James	Finance

### 3.2.10.3.3. OUTER JOIN

- OUTER JOIN là một phần mở rộng của INNER JOIN. Mặc dù tiêu chuẩn SQL định nghĩa ba loại OUTER JOIN là: LEFT, RIGHT và FULL nhưng SQLite chỉ hỗ trợ LEFT OUTER JOIN.
- Tương tự như INNER JOIN, OUTER JOIN sử dụng từ khóa ON để chỉ ra điều kiện kết.
- Cú pháp:  

```
SELECT ... FROM table1 LEFT OUTER JOIN table2 ON conditional_expression
...
```
- Ví dụ:  

```
sqlite> SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN DEPARTMENT
ON COMPANY.ID = DEPARTMENT.EMP_ID;
```

Kết quả thu được

EMP_ID	NAME	DEPT
1	Paul	IT Billing
2	Allen	Engineerin
	Teddy	
	Mark	
	David	
	Kim	
7	James	Finance

### 3.2.10.4. Mệnh đề WHERE

Mệnh đề WHERE được dùng để lọc và lấy các record cần thiết theo một điều kiện nào đó từ dữ liệu có trong một hoặc nhiều bảng. Nếu điều kiện được đáp ứng (thỏa điều kiện), thì kết quả sẽ trả về giá trị cụ thể từ table.

Mệnh đề WHERE không chỉ được sử dụng trong câu lệnh SELECT mà còn được sử dụng trong các lệnh UPDATE, DELETE, ...

#### 3.2.10.4.1. Cú pháp

```
SELECT column1, column2, columnN
FROM table_name
WHERE [condition] ;
```

#### 3.2.10.4.2. Ví dụ

Giả sử trong table COMPANY đã có dữ liệu như sau:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Jim	22	South-Hall	45000.0
7	Jimmy	24	Houston	10000.0

- Tìm những record có AGE>=25 và SALARY >=65.000:

sqlite> SELECT * FROM COMPANY WHERE AGE >= 25 AND SALARY >= 65000;				
ID	NAME	AGE	ADDRESS	SALARY
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0

- Tìm những record có AGE>=25 hoặc SALARY >=65.000:

sqlite> SELECT * FROM COMPANY WHERE AGE >= 25 OR SALARY >= 65000;				
ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0

### 3.2.10.4.3. Toán tử Like

- Sử dụng để chọn những record có giá trị của thuộc tính phù hợp với mẫu được đưa ra.
- Hai ký tự đại diện được dùng kèm với LIKE là:
  - Ký tự phần trăm (%): đại diện cho không, một, hoặc nhiều số (hay ký tự).
  - Ký tự gạch dưới (\_): đại diện cho một số (hay ký tự).
- Ví dụ:

Điều kiện	Điễn giải
WHERE SALARY LIKE '200%'	Tìm bất kỳ giá trị nào bắt đầu bởi 3 ký số là 200
WHERE SALARY LIKE '%200%'	Tìm bất kỳ giá trị nào có chứa 3 ký số là 200 (tại bất kỳ vị trí nào)
WHERE SALARY LIKE '_00%'	Tìm bất kỳ giá trị nào có ký tự thứ 2 và 3 đều là 0
WHERE SALARY LIKE '2_%_%'	Tìm bất kỳ giá trị nào bắt đầu là số 2 và có chiều dài tối thiểu là 3
WHERE SALARY LIKE '%2'	Tìm bất kỳ giá trị nào có ký tự cuối là số 2
WHERE SALARY LIKE '_5%7'	Tìm bất kỳ giá trị nào có ký tự thứ hai là số 5, ký tự cuối là số 7 và chiều dài chuỗi tối thiểu là 3
WHERE SALARY LIKE '2__3'	Tìm bất kỳ giá trị nào có ký tự đầu là số 2, ký tự cuối là số 3 và chiều dài chuỗi (chính xác) là 5

### 3.2.10.4.4. Toán tử GLOB

- Tương tự như toán tử LIKE, được sử dụng để chọn những record có giá trị của thuộc tính phù hợp với mẫu được đưa ra.
- Khác biệt
  - Có phân biệt chữ hoa/thường (LIKE không phân biệt).
  - Hai ký tự đại diện được dùng kèm với GLOB là:
    - Ký tự hoa thị (\*): đại diện cho không, một, hoặc nhiều số (hay ký tự).
    - Ký tự chấm hỏi (?): đại diện cho một số (hay ký tự).
- Ví dụ:

Điều kiện	Điễn giải
WHERE SALARY GLOB '200*'	Tìm bất kỳ giá trị nào bắt đầu bởi 3 ký số là 200
WHERE SALARY GLOB '*200*'	Tìm bất kỳ giá trị nào có chứa 3 ký số là 200 (tại bất kỳ vị trí nào)
WHERE SALARY GLOB '?00*'	Tìm bất kỳ giá trị nào có ký tự thứ 2 và 3 đều là 0
WHERE SALARY GLOB '2??'	Tìm bất kỳ giá trị nào bắt đầu là số 2 và có chiều dài tối thiểu là 3
WHERE SALARY GLOB '*2'	Tìm bất kỳ giá trị nào có ký tự cuối là số 2
WHERE SALARY GLOB '?5*7'	Tìm bất kỳ giá trị nào có ký tự thứ hai là số 5, ký tự cuối là số 7 và chiều dài chuỗi tối thiểu là 3
WHERE SALARY GLOB '2????3'	Tìm bất kỳ giá trị nào có ký tự đầu là số 2, ký tự cuối là số 3 và chiều dài chuỗi (chính xác) là 5

### 3.2.10.4.5. Sử dụng NULL trong mệnh đề WHERE

- NULL là một thuật ngữ được sử dụng để đại diện cho một giá trị thiếu và khác với giá trị số không (zero) hay khoảng trống ("").
- **Minh họa cú pháp sử dụng NULL**
  - **Khi tạo lập table:**

```
SQLite> CREATE TABLE COMPANY(ID INT PRIMARY KEY NOT NULL,
 NAME TEXT NOT NULL,
 AGE INT NOT NULL,
 ADDRESS CHAR(50),
 SALARY REAL);
```

Ở đây, NOT NULL có nghĩa rằng cột nên luôn luôn chấp nhận một giá trị rõ ràng của kiểu dữ liệu nhất định. Có hai cột mà chúng ta không sử dụng NOT NULL có nghĩa là các cột này có thể là NULL.

- **Khi tạo lập table:**

```
sqlite> UPDATE COMPANY
 SET ADDRESS = NULL, SALARY = NULL
 WHERE ID IN(6,7);
```

- **Sử dụng trong mệnh đề WHERE:**

```
sqlite> SELECT ID, NAME, AGE, ADDRESS, SALARY
 FROM COMPANY
 WHERE SALARY IS NOT NULL;
```

Hay

```
sqlite> SELECT ID, NAME, AGE, ADDRESS, SALARY
 FROM COMPANY
 WHERE SALARY IS NULL;
```

### 3.2.10.5. GROUP BY

- Sử dụng khi câu truy vấn có sử dụng các hàm thuộc nhóm Aggregate (MIN, MAX, ...). Mục đích để sắp xếp dữ liệu thành các nhóm.
- Trong câu lệnh SELECT, mệnh đề GROUP BY đi sau mệnh đề WHERE và đi trước mệnh đề ORDER BY.
- Cú pháp:

```
SELECT column-list
 FROM table_name
 WHERE [conditions]
 GROUP BY column1, column2....columnN
 ORDER BY column1, column2....columnN
```

### 3.2.10.6. HAVING

- Sử dụng khi câu truy vấn có sử dụng các hàm thuộc nhóm Aggregate (MIN, MAX, ...) trong biểu thức điều kiện. HAVING luôn đi chung và đi sau mệnh đề GROUP BY.
- Cú pháp:

```
SELECT column1, column2
 FROM table1, table2
 WHERE [conditions]
 GROUP BY column1, column2 HAVING [conditions]
 ORDER BY column1, column2
```

### 3.2.10.7. ORDER BY

- Sử dụng khi câu truy vấn có yêu cầu sắp xếp dựa trên giá trị của 1 hoặc nhiều cột.
- Cú pháp:

```
SELECT column1, column2....columnN
 FROM table_name
 WHERE CONDITION
 ORDER BY column_name {ASC|DESC};
```

### 3.2.10.8. Sử dụng ALIAS trong câu lệnh SELECT

- Alias được dùng để đổi tên một table hoặc một cột tạm thời (trong câu lệnh đang thực hiện) bằng cách đưa ra một cái tên khác, được gọi là bí danh.
- Cú pháp:
  - Sử dụng đổi với table
 

```
SELECT column1, column2...
 FROM table_name AS alias_name WHERE [condition];
```
  - Sử dụng đổi với cột
 

```
SELECT column_name AS alias_name
 FROM table_name WHERE [condition];
```

- Ví dụ:
  - Sử dụng đổi với table
 

```
sqlite> SELECT C.ID, C.NAME, C.AGE, D.DEPT
 FROM COMPANY AS C, DEPARTMENT AS D
 WHERE C.ID = D.EMP_ID;
```
  - Sử dụng đổi với cột
 

```
sqlite> SELECT C.ID AS COMPANY_ID, NAME AS COMPANY_NAME, AGE, DEPT
 FROM COMPANY AS C INNER JOIN DEPARTMENT AS D
 ON C.ID = D.EMP_ID;
```

### 3.2.10.9. Thiết lập độ rộng cột cho kết quả

Sử dụng lệnh `.width num, num, ...` như sau:

```
sqlite>.width 10, 20, 10
sqlite>SELECT * FROM COMPANY;
```

Kết quả: độ rộng các cột 1, 2 và 3 lần lượt được thiết lập là 10, 20, 10. Các cột sau đó có độ rộng bằng với số đi cuối (là số 10):

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Kim	22	South-Hall	45000.0
7	James	24	Houston	10000.0

### 3.2.10.10. Thông tin về lược đồ (Schema)

Do các lệnh dẫn trước bởi dấu chấm chỉ có tác dụng trong dấu nhắc lệnh của SQLite, vì vậy khi lập trình với SQLite để hiển thị danh sách các table có trong CSDL của mình, bạn cần sử dụng lệnh sau:

```
sqlite> SELECT tbl_name FROM sqlite_master WHERE type = 'table';
```

### 3.2.10.11. Mệnh đề/toán tử UNIONS trong SQLite

Mệnh đề/toán tử UNION được dùng để kết nối kết quả của 2 hay nhiều câu lệnh SELECT lại với nhau mà kết quả này sẽ không chứa những dòng trùng nhau.

Để sử dụng UNION, mỗi lệnh SELECT phải có cùng số lượng cột, cùng kiểu dữ liệu (có thể không có cùng kích thước) và cùng được sắp xếp thứ tự trong lệnh SELECT.

#### 3.2.10.11.1. Cú pháp:

```
SELECT column1 [, column2]
FROM table1 [, table2]
[WHERE condition]
UNION
SELECT column1 [, column2]
FROM table1 [, table2] [WHERE condition]
```

Trong đó, điều kiện (condition) có thể khác nhau tùy vào nhu cầu sử dụng của bạn.

### 3.2.10.11.2. Ví dụ:

Giả sử trong table COMPANY đã có dữ liệu như sau (chữ trong ngoặc là kiểu dữ liệu):

ID(INT)	NAME (TEXT)	AGE(INT)	ADDRESS(CHAR(50))	SALARY(REAL)
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Kim	22	South-Hall	45000.0
7	James	24	Houston	10000.0

Và dữ liệu trong table DEPARTMENT là (chữ trong ngoặc là kiểu dữ liệu):

ID(INT)	DEPT(CHAR(50))	EMP_ID(INT)
1	IT Billing	1
2	Engineering	2
3	Finance	7
4	Engineering	3
5	Finance	4
6	Engineering	5
7	Finance	6

Thực hiện kết 2 table bằng cách sử dụng lệnh SELECT cùng với mệnh đề UNION như sau:

```
sqlite> SELECT EMP_ID, NAME, DEPT
 FROM COMPANY INNER JOIN DEPARTMENT
 ON COMPANY.ID = DEPARTMENT.EMP_ID
UNION
SELECT EMP_ID, NAME, DEPT
 FROM COMPANY LEFT OUTER JOIN DEPARTMENT
 ON COMPANY.ID = DEPARTMENT.EMP_ID
```

This would produce the following result:

EMP_ID	NAME	DEPT
1	Paul	IT Billing
2	Allen	Engineering
3	Teddy	Engineering
4	Mark	Finance
5	David	Engineering
6	Kim	Finance
7	James	Finance

### 3.2.10.11.3. The UNION ALL Clause:

Mệnh đề/toán tử UNION được dùng để kết nối kết quả của 2 hay nhiều câu lệnh SELECT lại với nhau mà kết quả này sẽ có chứa những dòng trùng nhau.

Các quy tắc sử dụng cho UNION và UNION ALL là tương tự nhau.

- **Cú pháp:**

```
SELECT column1 [, column2]
 FROM table1 [, table2]
 [WHERE condition]
 UNION ALL
SELECT column1 [, column2]
 FROM table1 [, table2] [WHERE condition]
```

- **Ví dụ:** thực hiện tương tự như ví dụ trước, chỉ thêm từ khóa ALL

```
sqlite> SELECT EMP_ID, NAME, DEPT
 FROM COMPANY INNER JOIN DEPARTMENT
 ON COMPANY.ID = DEPARTMENT.EMP_ID
UNION
SELECT EMP_ID, NAME, DEPT
```

```
FROM COMPANY LEFT OUTER JOIN DEPARTMENT
ON COMPANY.ID = DEPARTMENT.EMP_ID
```

Kết quả thu được:

EMP_ID	NAME	DEPT
1	Paul	IT Billing
2	Allen	Engineerin
3	Teddy	Engineerin
4	Mark	Finance
5	David	Engineerin
6	Kim	Finance
7	James	Finance
1	Paul	IT Billing
2	Allen	Engineerin
3	Teddy	Engineerin
4	Mark	Finance
5	David	Engineerin
6	Kim	Finance
7	James	Finance

### 3.2.11. Update

- Dùng để hiệu chỉnh dữ liệu của các record trong table.
- Sử dụng mệnh đề WHERE:
  - Không sử dụng: việc hiệu chỉnh sẽ thực hiện trên tất cả các record.
  - Có sử dụng: chỉ những record nào thỏa điều kiện mới được hiệu chỉnh.

#### 3.2.11.1. Cú pháp

```
UPDATE table_name
SET column1 = value1, column2 = value2..., columnN = valueN
WHERE [condition];
```

#### 3.2.11.2. Ví dụ

Giả sử trong table COMPANY đã có dữ liệu như sau:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Jim	22	South-Hall	45000.0
7	Jimmy	24	Houston	10000.0

- Cập nhật lại thuộc tính ADDRESS là 'Texas' cho record có ID=6:

```
sqlite> UPDATE COMPANY SET ADDRESS = 'Texas' WHERE ID = 6;
```

Kết quả thực hiện:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Jim	22	Texas	45000.0
7	Jimmy	24	Houston	10000.0

### 3.2.12. Delete

- Dùng để xóa các record trong table.
- Sử dụng mệnh đề WHERE:
  - Không sử dụng: xóa tất cả các record.
  - Có sử dụng: chỉ xóa những record nào thỏa điều kiện.

- Trong SQLite, sau khi sử dụng lệnh DELETE, bạn nên sử dụng thêm lệnh VACUUM để xóa đi không gian không còn sử dụng

### 3.2.12.1. Cú pháp

```
DELETE FROM table_name WHERE [condition];
VACUUM
```

### 3.2.12.2. Ví dụ

Giả sử trong table COMPANY đã có dữ liệu như sau:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Jim	22	South-Hall	45000.0
7	Jimmy	24	Houston	10000.0

- Xóa record có thuộc tính ID=7:

```
sqlite>DELETE FROM COMPANY WHERE ID = 7;
sqlite>VACUUM
```

Kết quả thực hiện:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Jim	22	Texas	45000.0

## 3.2.13. Lệnh VACUUM

Lệnh VACUUM làm sạch CSDL chính bằng cách sao chép nội dung của nó vào một file cơ sở dữ liệu tạm thời và tải lại các file cơ sở dữ liệu ban đầu từ các bản sao. Điều này giúp sắp xếp các dữ liệu table sát nhau. Cũng có thể sử dụng lệnh VACUUM để sửa đổi các thông số cấu hình CSDL cụ thể.

### 3.2.13.1. Sử dụng lệnh VACUUM thủ công

- Sử dụng đơn giản:

```
sqlite> VACUUM;
```

- Sử dụng cho database có tên được chỉ ra:

```
$sqlite3 database_name "VACUUM;"
```

- Sử dụng cho riêng table có tên được chỉ ra:

```
sqlite> VACUUM table_name;
```

### 3.2.13.2. Sử dụng lệnh VACUUM tự động

- Lệnh *auto\_vacuum* không thực hiện tương tự như lệnh VACUUM mà có chức năng giúp CSDL chống lại việc phân mảnh, nhờ vậy giúp CSDL nhỏ gọn.
- Bạn có thể kích hoạt (Enable)/vô hiệu hóa (Disable) lệnh *auto\_vacuum* tại dấu nhắc lệnh của SQLite như sau:

```
sqlite> PRAGMA auto_vacuum = NONE; -- hay =0 ⇒ disable auto vacuum
sqlite> PRAGMA auto_vacuum = INCREMENTAL; hay =1 ⇒ enable incremental vacuum
sqlite> PRAGMA auto_vacuum = FULL; hay =2 ⇒ enable full auto vacuum
```

- Sử dụng lệnh sau để kiểm tra thiết lập về *auto\_vacuum*:

```
$sqlite3 database_name "PRAGMA auto_vacuum;"
```

## 3.2.14. Sub Queries

- Một Subquery hay Inner query hoặc Nested query là 1 lệnh truy vấn dữ liệu. lệnh này được nhúng vào bên trong mệnh đề WHERE của một lệnh SQLite khác.

- Mục đích của subquery là trả về dữ liệu và dữ liệu này sẽ được dùng để hạn chế số record thu được (hoặc bị ảnh hưởng) bởi lệnh SQLite chính chứa subquery.
- Subqueries có thể được nhúng vào trong mệnh đề WHERE của các lệnh SELECT, INSERT, UPDATE, DELETE và đi sau các toán tử như =, <, >, >=, <=, IN, BETWEEN, ....
- Một số quy định khi dùng subqueries:
  - Subqueries phải được đặt trong cặp ngoặc đơn (parentheses).
  - subquery chỉ được có duy nhất 1 cột trong mệnh đề SELECT của mình.
  - mệnh đề ORDER BY: không được sử dụng trong subquery, mặc dù lệnh truy vấn chính vẫn có thể được dùng
  - Mệnh đề GROUP BY: có thể được dùng trong cả subquery và mainquery.
  - Khi kết quả trả về từ subqueries gồm nhiều dòng, bạn chỉ được dùng những toán tử có khả năng so sánh với nhiều giá trị như toán tử IN.
  - Toán tử BETWEEN không được dùng với subquery; tuy nhiên toán tử BETWEEN vẫn có thể được dùng bên trong subquery.

### 3.2.14.1. Sử dụng Subqueries với lệnh SELECT

- Cú pháp**

```
SELECT column_name [, column_name]
 FROM table1 [, table2]
 WHERE column_name OPERATOR (SELECT column_name [, column_name]
 FROM table1 [, table2]
 [WHERE])
```

- Ví dụ**

Giả sử trong table COMPANY đã có dữ liệu như sau:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Jim	22	South-Hall	45000.0
7	Jimmy	24	Houston	10000.0

Sử dụng subquery để tìm những record có SALARY>45.000:

```
sqlite> SELECT *
 FROM COMPANY
 WHERE ID IN (SELECT ID
 FROM COMPANY
 WHERE SALARY > 45000);
```

Kết quả:

ID	NAME	AGE	ADDRESS	SALARY
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0

### 3.2.14.2. Sử dụng Subqueries với lệnh INSERT

- Lệnh INSERT sử dụng dữ liệu trả về của subquery để thêm vào table khác. Có thể sử dụng các hàm về số, chuỗi, ngày/giờ trong subquery để tạo dữ liệu phù hợp với table được thêm dữ liệu vào.

- Cú pháp**

```
INSERT INTO table_name [(column1 [, column2])]
 SELECT [*|column1 [, column2]]
 FROM table1 [, table2]
 [WHERE VALUE OPERATOR]
```

- **Ví dụ:** Giả sử đã có 1 table tên COMPANY\_BACKUP với cấu trúc tương tự như table hiện có. Để copy các record trong table COMPANY\_BACKUP vào table COMPANY ta dùng lệnh sau:

```
sqlite> INSERT INTO COMPANY_BACKUP
 SELECT * FROM COMPANY
 WHERE ID IN (SELECT ID
 FROM COMPANY) ;
```

### 3.2.14.3. Sử dụng Subqueries với lệnh UPDATE

- Giúp việc cập nhật dữ liệu được thực hiện theo điều kiện cho trước, hay nói cách khác chỉ những record thỏa điều kiện có trong WHERE mới thực hiện việc cập nhật lại dữ liệu.
- **Cú pháp**

```
UPDATE table
SET column_name = new_value
[WHERE OPERATOR [VALUE] (SELECT COLUMN_NAME
 FROM TABLE_NAME
 [WHERE])]
```

- **Ví dụ:** tăng SALARY lên 10% cho những record có ký tự đầu của ADDRESS là 'T' trong table STATE:

```
sqlite> UPDATE COMPANY
 SET SALARY = SALARY * 1.10
 WHERE ADDRESS IN (SELECT ADDRESS FROM STATE
 WHERE ADDRESS LIKE 'T%');
```

### 3.2.14.4. Sử dụng Subqueries với lệnh DELETE

- Giúp xóa những record thỏa điều kiện cho trước.

```
DELETE FROM TABLE_NAME
[WHERE OPERATOR [VALUE] (SELECT COLUMN_NAME
 FROM TABLE_NAME
 [WHERE])]
```

- **Ví dụ:** xóa những record mà ID không có trong table COMPANY\_BACKUP:

```
sqlite> DELETE FROM COMPANY
 WHERE ID NOT IN (SELECT ID
 FROM COMPANY_BACKUP);
```

## 3.2.15. Views

- View được cấu thành từ một lệnh truy vấn dữ liệu trong SQLite, do đó view sẽ chứa các thành phần của một (hay nhiều) table trong CSDL.
- Một View có thể được xem là 1 table ảo và có những đặc điểm sau:
  - Cấu trúc dữ liệu theo cách người sử dụng (hoặc class của người sử dụng) mong muốn.
  - Có thể là nguồn dữ liệu cung cấp cho lệnh SELECT khác.
  - Thay vì hiển thị đầy đủ thông tin của table, View giúp hạn chế truy cập vào các dữ liệu mà người dùng không được phép xem hoặc chỉnh sửa.
  - Do dữ liệu của View là read only nên không thể thực hiện các thao tác DELETE, INSERT hoặc UPDATE trên một View.

### 3.2.15.1. Tạo Views

- **Cú pháp**

```
CREATE [TEMP | TEMPORARY] VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE [condition];
```

Trong đó: Nếu 1 trong 2 tùy chọn TEMP hoặc TEMPORARY, View sẽ được tạo ra trong CSDL tạm thời.

- **Ví dụ** dữ liệu có trong table COMPANY như sau:

Giả sử trong table COMPANY đã có dữ liệu như sau:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Jim	22	South-Hall	45000.0
7	Jimmy	24	Houston	10000.0

Tạo 1 view chứa các thuộc tính ID, NAME, AGE của table COMPANY:

```
sqlite> CREATE VIEW COMPANY_VIEW
 AS
 SELECT ID, NAME, AGE FROM COMPANY;
```

Chọn những record có AGE >=27 trong COMPANY\_VIEW:

```
sqlite> SELECT * FROM COMPANY_VIEW;
```

Kết quả thu được

ID	NAME	AGE
1	Paul	32
5	David	27

### 3.2.15.2. Dropping Views

- **Cú pháp**

```
sqlite> DROP VIEW view_name;
```

- **Ví dụ** dữ liệu có trong table COMPANY như sau:

```
sqlite> DROP VIEW COMPANY_VIEW;
```

## 3.2.16. Trigger

Triggers là chức năng do SQLite cung cấp cho người lập trình tự cài đặt. Sau khi được cài đặt xong, Triggers sẽ tự động thực hiện khi có 1 sự kiện xảy ra trên CSDL.

Một số điểm quan trọng về Trigger:

- Khi cài đặt Trigger cần chỉ định rõ:
    - Trigger áp dụng cho table nào?
    - Trigger được gắn với những thao tác nào trong các thao tác DELETE, INSERT hoặc UPDATE.
  - Hiện tại, SQLite chỉ hỗ trợ trigger cho FOR EACH ROW, chứ chưa hỗ trợ cho FOR EACH STATEMENT. Do đó khi tạo lập Trigger, việc ghi FOR EACH ROW trong lệnh là tùy chọn.
  - Cả mệnh đề WHEN và trigger actions có thể truy cập các phần tử của record được insert, delete hoặc update bằng cách sử dụng tham chiếu của mẫu *NEW.column-name* và *OLD.column-name*, trong đó *column-name* là tên của một cột trong table mà trigger được liên kết tới.
  - Nếu mệnh đề WHEN được chỉ ra, các câu lệnh SQL chỉ được thực hiện trên các record mà điều kiện trong WHEN là đúng. Nếu mệnh đề WHEN không được chỉ ra, các câu lệnh SQL được thực hiện cho tất cả các record.
  - Từ khóa BEFORE hoặc AFTER cho biết trigger sẽ được thực hiện trước (BEFORE) hay sau (AFTER) khi thực hiện các lệnh về insert, delete, update trên các record.
- Triggers sẽ được tự động xóa khi table có liên quan đến nó bị xóa.
- Cả table hoặc view được đính kèm trigger và table được sửa đổi phải tồn tại trong CSDL, do đó chỉ được dùng dạng *table\_name* mà không *database\_name.table\_name*.
  - Hàm RAISE() có thể được sử dụng khi lập trình với trigger để nâng cao khả năng xử lý các ngoại lệ (exception).

### 3.2.16.1. Cú pháp

- Tạo trigger trên table

```
CREATE TRIGGER trigger_name [BEFORE|AFTER] event_name
```

```

ON table_name [FOR EACH ROW]
BEGIN
 -- Trigger logic goes here....
END;

```

Trong đó:

- **event\_name** phải là các giá trị *INSERT*, *DELETE*, *UPDATE*.
- **table\_name**: chỉ ra tên table mà trigger được gắn vào
- Tạo trigger đối với thao tác *UPDATE* trên 1 (hoặc nhiều) cột của table

```

CREATE TRIGGER trigger_name [BEFORE|AFTER]
UPDATE OF column_name ON table_name
BEGIN
 -- Trigger logic goes here....
END;

```

### 3.2.16.2. Ví dụ

Giả sử có nhu cầu lưu lại ID của tất cả các record đã được thêm vào table COMPANY trong 1 table khác có tên là AUDIT. Với table AUDIT chỉ gồm 2 thuộc tính là *EMP\_ID* và *ENTRY\_DATE*:

- Tạo mới table COMPANY:

```

sqlite> CREATE TABLE COMPANY(ID INT PRIMARY KEY NOT NULL,
 NAME TEXT NOT NULL,
 AGE INT NOT NULL,
 ADDRESS CHAR(50),
 SALARY REAL);

```

- Tạo mới table AUDIT:

```

sqlite> CREATE TABLE AUDIT(EMP_ID INT NOT NULL,
 ENTRY_DATE TEXT NOT NULL);

```

Trong đó, ID của table COMPANY sẽ được lưu vào thuộc tính *EMP\_ID* của table AUDIT và thuộc tính *ENTRY\_DATE* sẽ lưu lại thời gian mà record được thêm vào table COMPANY.

- Tạo trigger trên table COMPANY:

```

sqlite> CREATE TRIGGER audit_log AFTER INSERT ON COMPANY
 BEGIN
 INSERT INTO AUDIT(EMP_ID, ENTRY_DATE)
 VALUES (new.ID, datetime('now'));
 END;

```

- Thêm mới 1 dòng vào table COMPANY:

```

sqlite> INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
 VALUES (1, 'Paul', 32, 'California', 20000.00);

```

Lúc này dữ liệu trong table COMPANY là:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0

Đồng thời, dữ liệu trong table AUDIT sẽ là:

EMP_ID	ENTRY_DATE
1	2014-10-05 06:26:00

### 3.2.16.3. Liệt kê các TRIGGERS

- Liệt kê tên các trigger có trong CSDL bằng cách truy vấn dữ liệu trong table *sqlite\_master* như sau:
 

```
sqlite> SELECT name FROM sqlite_master WHERE type = 'trigger';
```
- Liệt kê tên các trigger có trong một table:
 

```
sqlite> SELECT name FROM sqlite_master
WHERE type = 'trigger' AND tbl_name = 'COMPANY';
```

### 3.2.16.4. Xóa TRIGGER

Sử dụng lệnh *DROP TRIGGER* và tên trigger cần xóa theo cú pháp sau:

```
sqlite> DROP TRIGGER trigger_name;
```

### 3.2.17. Transactions

Transaction là 1 (hoặc các chuỗi) công việc (lệnh) thực hiện theo một thứ tự hợp lý.

#### 3.2.17.1. Các đặc tính tiêu chuẩn của Transactions

Các transaction có bốn đặc tính tiêu chuẩn sau đây, thường được gọi tắt là ACID (viết tắt từ Atomicity- Consistency- Isolation- Durability):

- **Atomicity (tính nguyên tử)**: đảm bảo rằng tất cả các hoạt động trong đơn vị công việc được hoàn thành; nếu không, các transaction bị hủy bỏ tại thời điểm thất bại và các hoạt động trước đó được cuộn trở lại (*rolled back*) tình trạng ban đầu.
- **Consistency (Tính nhất quán)**: đảm bảo rằng CSDL thay đổi trạng thái theo đúng những gì đã thực hiện trong transaction.
- **Isolation (Tính cách ly)**: cho phép các transaction hoạt động độc lập và minh bạch với nhau.
- **Durability (Độ bền)**: đảm bảo rằng các kết quả hoặc hiệu quả của một transaction vẫn tồn tại trong trường hợp lỗi hệ thống.

#### 3.2.17.2. Các lệnh sử dụng để điều khiển Transaction

Các lệnh điều khiển về Transaction chỉ được dùng với các lệnh INSERT, UPDATE và DELETE mà không thể dùng với các lệnh CREATE TABLE hoặc DROP TABLE vì các lệnh này được thực hiện tự động trong CSDL.

##### 3.2.17.2.1. BEGIN TRANSACTION

- Transactions có thể được khởi tạo bằng cách dùng lệnh BEGIN TRANSACTION hoặc đơn giản hơn là BEGIN. Có thể xem các lệnh sử dụng trong transactions được xử lý trong bộ nhớ tạm cho đến khi gặp lệnh COMMIT hoặc ROLLBACK.
- Cú pháp:  
BEGIN;  
or  
BEGIN TRANSACTION;

##### 3.2.17.2.2. COMMIT hoặc END TRANSACTION

- Lệnh COMMIT dùng để lưu lại tất cả những thay đổi trên CSDL kể từ lệnh COMMIT hay ROLLBACK gần nhất đến lệnh COMMIT đang xét.

- Cú pháp:  
COMMIT;  
or  
END TRANSACTION;

##### 3.2.17.2.3. ROLLBACK

- Công dụng:
  - Lệnh ROLLBACK được dùng để bỏ qua các giao tác vừa thực hiện trên CSDL kể từ lệnh COMMIT hay ROLLBACK gần nhất đến lệnh ROLLBACK đang xét.
  - Một transaction cũng sẽ tự động ROLLBACK khi CSDL đang làm việc bị đóng hay khi có lỗi xảy ra.
- Cú pháp:  
ROLLBACK;

##### 3.2.17.2.4. Ví dụ

- Giả sử dữ liệu trong table COMPANY đang có như sau:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Kim	22	South-Hall	45000.0
7	James	24	Houston	10000.0

- Thực hiện lệnh xóa những record có AGE=25, ngay sau đó sử dụng lệnh ROLLBACK. Như vậy sau khi lệnh xóa thực hiện xong, lệnh ROLLBACK đã undo kết quả của lệnh delete do đó dữ liệu trong table COMPANY không có gì thay đổi

```
sqlite> BEGIN;
sqlite> DELETE FROM COMPANY WHERE AGE = 25;
sqlite> ROLLBACK;
sqlite> SELECT * FROM COMPANY;
```

Kết quả kiểm tra:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Kim	22	South-Hall	45000.0
7	James	24	Houston	10000.0

- Thực hiện lại lệnh xóa những record có AGE=25, nhưng sau đó thay vì dùng lệnh ROLLBACK, ta dùng lệnh COMMIT để lưu các thay đổi trên table COMPANY.

Now, let's start another transaction and delete records from the table having age = 25 and finally we use COMMIT command to commit all the changes.

```
sqlite> BEGIN;
sqlite> DELETE FROM COMPANY WHERE AGE = 25;
sqlite> COMMIT;
sqlite> SELECT * FROM COMPANY;
```

Kết quả kiểm tra:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
3	Teddy	23	Norway	20000.0
5	David	27	Texas	85000.0
6	Kim	22	South-Hall	45000.0
7	James	24	Houston	10000.0

### 3.2.18. Indexes

Thường trong những cuốn sách về kỹ thuật, phần cuối của cuốn sách sẽ liệt kê các từ khóa xuất hiện trong cuốn sách theo thứ tự alphabet và số trang nơi từ khóa đó xuất hiện. Khi cần tra cứu từ khóa nào, bạn tìm từ trong phần đó, dựa vào số trang đi kèm từ đó bạn sẽ tra nội dung của từ một cách dễ dàng.

Trong SQLite Indexes được tổ chức tương tự như vậy. Khi đó Indexes được tổ chức thành các table giúp tăng tốc độ tìm kiếm, truy xuất dữ liệu.

Chỉ số cũng có thể là duy nhất (tương tự như UNIQUE), trong đó chỉ số ngăn cản mục trùng lặp trong cột hoặc kết hợp các cột mà trên đó có một chỉ số.

#### 3.2.18.1. Tạo INDEX

- Index dựa trên duy nhất 1 cột  
CREATE INDEX index\_name ON table\_name (column\_name);
- Index dựa trên nhiều cột (Composite Indexes)  
CREATE INDEX index\_name ON table\_name (column1, column2);

Thông thường, người ta chỉ tạo index dựa trên khóa chính của table (khóa chính có thể gồm 1 hay nhiều thuộc tính). Vì vậy, khi tạo index mà các thuộc tính tham gia tạo index không phải là khóa chính có thể làm ảnh hưởng đến hiệu suất của index.

- Implicit Indexes

Index tiềm ẩn (Implicit indexes) là các indexes có thể được tự động tạo ra bởi các database server. Khi đó, các indexes được tự động tạo ra dựa trên các ràng buộc khóa chính và ràng buộc unique.

- Ví dụ: tạo index cho table COMPANY dựa trên thuộc tính ID  

```
sqlite> CREATE INDEX salary_index ON COMPANY (ID);
```

### 3.2.18.2. Liệt kê các index

- Sử dụng lệnh .indices đi kèm tên table để xem index của table:  

```
sqlite> .indices COMPANY
```

Kết quả hiển thị có 2 index, trong đó `sqlite_autoindex_COMPANY_1` là 1 implicit index, index này được tạo ra ngay sau khi lệnh tạo table được thực hiện.

```
id_index sqlite_autoindex_COMPANY_1
```

- Liệt kê tất cả các indexes có trong CSDL:  

```
sqlite> SELECT * FROM sqlite_master WHERE type = 'index';
```

### 3.2.18.3. Xóa INDEX

- Sử dụng lệnh DROP với cú pháp như sau:  

```
DROP INDEX index_name;
```
- Ví dụ:  

```
sqlite> DROP INDEX id_index;
```

### 3.2.18.4. Một số trường hợp nên tránh dùng indexes

Mặc dù indexes nhằm nâng cao hiệu suất của một CSDL, tuy nhiên có những lúc bạn cần cân nhắc khi sử dụng indexes trong các trường hợp sau:

- Kích thước table nhỏ.
- Table thường xuyên thực hiện hàng loạt thao tác cập nhật hoặc chèn dữ liệu.
- Cột dự định sử dụng làm indexes có chứa một số lượng lớn các giá trị NULL.
- Lập chỉ mục trên những cột thường xuyên phải thao tác (thêm, xóa, sửa).

### 3.2.18.5. Indexed By

Mệnh đề "*INDEXED BY index-name*" đi kèm trong câu lệnh cho biết việc dò tìm dữ liệu phải được ưu tiên thực hiện trên index trước.

Mệnh đề "*NOT INDEXED*" cho biết không có index được sử dụng khi truy cập vào table. Tuy nhiên, INTEGER PRIMARY KEY vẫn có thể được sử dụng để tìm ngay cả khi đã chỉ rõ mệnh đề "*NOT INDEXED*" trong câu lệnh.

#### 3.2.18.5.1. Cú pháp

Sử dụng được cho các lệnh DELETE, UPDATE hoặc SELECT:

```
SELECT|DELETE|UPDATE column1, column2...
INDEXED BY (index_name) table_name
WHERE (CONDITION);
```

#### 3.2.18.5.2. Ví dụ

Cho table COMPANY với dữ liệu như sau:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Jim	22	South-Hall	45000.0
7	Jimmy	24	Houston	10000.0

Tạo index cho table COMPANY dựa trên thuộc tính SALARY:

```
sqlite> CREATE INDEX salary_index ON COMPANY(salary);
sqlite>
```

Thực hiện truy vấn dữ liệu dựa trên index:

```
sqlite> SELECT * FROM COMPANY INDEXED BY salary_index WHERE salary > 5000;
```

### 3.2.19. Các hàm thường dùng trong SQLite

### **3.2.19.1. Hàm về Date & Time**

### 3.2.19.1.1. Hàm về date và time:

<b>Function</b>	<b>Ví dụ</b>
date(timestring, modifiers...)	Trả về ngày theo định dạng: YYYY-MM-DD
time(timestring, modifiers...)	Trả về thời gian theo định dạng HH:MM:SS
datetime(timestring, modifiers...)	Trả về ngày giờ theo định dạng YYYY-MM-DD HH:MM:SS
julianday(timestring, modifiers...)	Trả về số ngày tính từ November 24, 4714 B.C.
strftime(timestring, modifiers...)	Trả về ngày được định dạng theo chuỗi định dạng được chỉ ra trong đối số đầu tiên. Tham khảo cách sử dụng chuỗi cho đối số đầu tiên trong các phần tiếp sau.

Cả 5 hàm về ngày và thời gian ở trên đều có tham số *timestring*. *Timestring* được theo sau bằng số không (zero) hoặc các định dạng bổ sung khác (modifiers).

### 3.2.19.1.2. Time Strings

Timestring gồm các kiểu định dạng như sau:

<b>Time String</b>	<b>Ví dụ</b>
YYYY-MM-DD	2010-12-30
YYYY-MM-DD HH:MM	2010-12-30 12:10
YYYY-MM-DD HH:MM:SS.SSS	2010-12-30 12:10:04.100
MM-DD-YYYY HH:MM	30-12-2010 12:10
HH:MM	12:10
YYYY-MM-DD <b>T</b> HH:MM	2010-12-30 12:10
HH:MM:SS	12:10:01
YYYYMMDD HHMMSS	20101230 121001
Now	2013-05-07

Bạn có thể sử dụng "**T**" làm ký tự ngăn cách giữa date và time.

### 3.2.19.1.3. Modifiers

*Timestring* được sau bằng số không (zero) hoặc các định dạng bổ sung khác (modifiers), những định dạng này sẽ thay thế date và/hoặc time trả về bởi bất kỳ hàm nào trong 5 hàm ở trên.

- NNN days
  - NNN hours
  - start of month
  - weekday N
  - NNN months
  - NNN minutes
  - start of year
  - unixepoch
  - NNN years
  - NNN.NNNN seconds
  - start of day
  - localtime
  - utc

#### *3.2.19.1.4. Formatters*

SQLite cung cấp hàm `strftime()` rất tiện dụng để định dạng về date và time. Bạn có thể sử dụng các định dạng thay thế sau đây để định dạng lại date và time của mình:

<b>Định dạng thay thế</b>	<b>Điển giải</b>
%d	Day of month, 01-31
%f	Fractional seconds, SS.SSS
%H	Hour, 00-23
%j	Day of year, 001-366
%J	Julian day number, DDDD.DDDD

%m	Month, 00-12
%M	Minute, 00-59
%S	Seconds since 1970-01-01
%S	Seconds, 00-59
%w	Day of week, 0-6 (0 is Sunday)
%W	Week of year, 01-53
%Y	Year, YYYY
%%	% symbol

### 3.2.19.1.5. Ví dụ

- Lấy ngày hiện tại:

```
sqlite> SELECT date('now'); -- => 2014-10-09
```

- Lấy ngày cuối cùng của tháng hiện tại:

```
sqlite> SELECT date('now', 'start of month', '+1 month', '-1 day'); -- => 2014-10-30
```

- Tính toán date và time theo UNIX timestamp với giá trị là 1092941466:

```
sqlite> SELECT datetime(1092941466, 'unixepoch'); -- => 2004-08-19 18:51:06
```

- Tính toán date và time theo UNIX timestamp với giá trị là 1092941466 theo múi giờ địa phương:

```
sqlite> SELECT datetime(1092941466, 'unixepoch', 'localtime'); --=> 2004-08-19 11:51:06
```

- Tính toán số ngày kể từ khi Tuyên bố Độc lập của Hoa Kỳ:

```
sqlite> SELECT julianday('now') - julianday('1776-07-04'); -- => 86504.4775830326
```

- Tính toán ngày Thứ Ba đầu tiên của tháng và năm hiện tại:

```
sqlite> SELECT date('now', 'start of year', '+9 months', 'weekday 2'); --=> 2014-10-06
```

- Tính toán số giây kể từ một thời điểm cụ thể trong năm 2004:

```
sqlite> SELECT strftime('%s', 'now')-strftime('%s', '2004-01-01 02:34:56');--=> 295001572
```

- Tính tương tự như hàm strftime ('% s', 'now'), ngoại trừ bao gồm phần thập phân:

```
sqlite> SELECT (julianday('now') - 2440587.5)*86400.0; -- => 1367926077.12598
```

- Chuyển đổi time giữa UTC và giá trị thời gian địa phương:

```
sqlite> SELECT time('12:00', 'localtime'); -- => 05:00:00
```

```
sqlite> SELECT time('12:00', 'utc'); -- => 9:00:00
```

### 3.2.19.2. Hàm về số

- Có thể sử dụng tên hàm dưới dạng chữ thường, chữ hoa hay vừa thường vừa hoa đều được.
- Cho dữ liệu mẫu của table COMPANY như sau:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Kim	22	South-Hall	45000.0
7	James	24	Houston	10000.0

### 3.2.19.2.1. COUNT

- Đếm số dòng có trong table.
- Ví dụ:

```
sqlite> SELECT count(*) FROM COMPANY;
```

Kết quả

count(*)
-----
7

### 3.2.19.2.2. MAX

- Chọn giá trị lớn nhất trên cột có tên được chỉ ra.
- Ví dụ:

```
sqlite> SELECT max(salary) FROM COMPANY;
```

Kết quả

```
max(salary)

85000.0
```

### 3.2.19.2.3. MIN

- Chọn giá trị nhỏ nhất trên cột có tên được chỉ ra.
- Ví dụ:

```
sqlite> SELECT min(salary) FROM COMPANY;
```

Kết quả

```
min(salary)

10000.0
```

### 3.2.19.2.4. AVG

- Lấy giá trị trung bình trên cột có tên được chỉ ra.
- Ví dụ:

```
sqlite> SELECT avg(salary) FROM COMPANY;
```

Kết quả

```
avg(salary)

37142.8571428572
```

### 3.2.19.2.5. SQLite SUM Function

- Lấy tổng giá trị trên cột có tên được chỉ ra.
- Ví dụ:

```
sqlite> SELECT sum(salary) FROM COMPANY;
```

Kết quả

```
sum(salary)

260000.0
```

### 3.2.19.2.6. RANDOM

- Chọn giá trị nguyên ngẫu nhiên trong khoảng từ -9223372036854775808 đến +9223372036854775807.
- Ví dụ:

```
sqlite> SELECT random() AS Random;
```

Kết quả

```
Random

5876796417670984050
```

### 3.2.19.2.7. ABS

- Trả về trị tuyệt đối của đối số.
- Ví dụ:

```
sqlite> SELECT abs(5), abs(-15), abs(NULL), abs(0), abs("ABC");
```

Kết quả

abs(5)	abs(-15)	abs(NULL)	abs(0)	abs("ABC")
5	15		0	0.0

## 3.2.19.3. Hàm xử lý chuỗi

### 3.2.19.3.1. UPPER

- Chuyển chuỗi tham số của hàm thành chữ hoa.
- Ví dụ:

```
sqlite> SELECT upper(name) FROM COMPANY;
```

Kết quả

```
upper(name)

PAUL
ALLEN
TEDDY MARK
DAVID KIM
JAMES
```

### 3.2.19.3.2. LOWER

- Chuyển chuỗi tham số của hàm thành chữ thường.
- Ví dụ:

```
sqlite> SELECT lower(name) FROM COMPANY;
```

Kết quả

```
lower(name)

paul
allen
teddy
mark
david
kim
james
```

### 3.2.19.3.3. LENGTH

- Lấy chiều dài của tham số chuỗi.
- Ví dụ:

```
sqlite> SELECT name, length(name) FROM COMPANY;
```

Kết quả

NAME	length(name)
Paul	4
Allen	5
Teddy	5
Mark	4
David	5
Kim	3
James	5

## 3.2.19.4. Hàm khác

### 3.2.19.4.1. sqlite\_version

- trả về version của SQLite library đang dùng.
- Ví dụ:

```
sqlite> SELECT sqlite_version() AS 'SQLite Version';
```

Kết quả

```
SQLite Version

3.6.20
```

## 3.2.20. SQLite PRAGMA

Lệnh pragma là một lệnh đặc biệt được sử dụng để kiểm soát các biến môi trường và các cờ (flag) trạng thái khác nhau trong môi trường SQLite. Một giá trị pragma có thể được đọc và nó cũng có thể được thiết lập dựa trên yêu cầu.

### 3.2.20.1. Cú pháp:

- Truy vấn giá trị hiện tại của PRAGMA

```
PRAGMA pragma_name;
```

Giá trị trả về luôn là 1 số nguyên dương.

- Thiết lập giá trị cho PRAGMA

```
PRAGMA pragma_name = value;
```

Trong đó, value có thể là tên hoặc 1 số nguyên dương tương đương với tên đó (được SQLite quy định trước).

### 3.2.20.2. Các loại biến môi trường hoặc cờ trạng thái

#### 3.2.20.2.1. auto\_vacuum

- Các pragma auto\_vacuum được lấy hoặc thiết lập mode (chế độ) của auto-vacuum.
- Cú pháp:

```
PRAGMA [database.]auto_vacuum;
PRAGMA [database.]auto_vacuum = mode;
```

- Các mode có thể được dùng:

Giá trị	Điễn giải
0 hoặc NONE	(mặc định)Auto-vacuum bị vô hiệu hóa, nghĩa là một file CSDL sẽ không bao giờ thu gọn kích thước trừ khi lệnh VACUUM được thực hiện thủ công.
1 hoặc FULL	Kích hoạt Auto-vacuum, cho phép tự động thu gọn dữ liệu.
2 hoặc INCREMENTAL	Kích hoạt Auto-vacuum, nhưng phải được kích hoạt bằng tay. Trong chế độ này, dữ liệu tham khảo được duy trì, nhưng trang trống (free pages) chỉ đơn giản là được đưa vào danh sách trống (free list). Những trang này có thể được phục hồi bằng cách sử dụng <i>incremental_vacuum pragma</i> .

#### 3.2.20.2.2. cache\_size

- Giúp nhận được hoặc tạm thời thiết lập kích thước tối đa của bộ nhớ trong (cache)
- Cú pháp:

```
PRAGMA [database.]cache_size;
PRAGMA [database.]cache_size = pages;
```

Trong đó, *pages* đại diện cho số lượng các trang trong bộ nhớ cache. Mặc định, *pages* có giá trị là 2.000, tối thiểu là 10.

#### 3.2.20.2.3. case\_sensitive\_like

- *case\_sensitive\_like* pragma kiểm soát các trường hợp khi so sánh chữ thường/chữ hoa có toán tử LIKE. Mặc định pragma này có giá trị là false, nghĩa là toán tử LIKE bỏ qua chữ hoa.
- Cú pháp:

```
PRAGMA case_sensitive_like = [true|false];
```

- SQLite không cung cấp truy vấn giá trị hiện tại của pragma này.

#### 3.2.20.2.4. count\_changes

- *count\_changes* giúp nhận được hoặc thiết lập giá trị trả về của các thao tác INSERT, UPDATE và DELETE.
- Cú pháp:

```
PRAGMA count_changes;
PRAGMA count_changes = [true|false];
```

- Mặc định giá trị của *count\_changes* pragma là false và các thao tác trên sẽ không trả về bất kỳ giá trị gì. Nếu được thiết lập là true, giá trị trả về của các lệnh trên sẽ trả về 1 table gồm 1 dòng và 1 cột, trong đó chứa 1 số nguyên cho biết số dòng bị tác động bởi lệnh.

#### 3.2.20.2.5. database\_list

- *database\_list* pragma được sử dụng để liệt kê tất cả các cơ sở dữ liệu đính kèm (attached).

- Cú pháp:

```
PRAGMA database_list;
```

- Pragma này sẽ trả về một table gồm ba cột và 1 dòng cho mỗi CSDL đính kèm gồm các cột số thứ tự, tên và file liên quan.

#### 3.2.20.2.6. encoding

- **encoding** pragma điều khiển việc chuỗi được mã hóa như thế nào và lưu trữ trong một file CSDL.
- Cú pháp:

```
PRAGMA encoding;
PRAGMA encoding = format;
```

Trong đó format có thể là 1 trong những giá trị sau: UTF-8, UTF-16le, hoặc UTF-16be.

#### 3.2.20.2.7. freelist\_count

- **freelist\_count** pragma trả về một số nguyên duy nhất cho biết có bao nhiêu trang (page) CSDL hiện đang được đánh dấu là trống (free).
- Cú pháp:

```
PRAGMA [database.]freelist_count;
```

Trong đó format có thể là 1 trong những giá trị sau: UTF-8, UTF-16le, hoặc UTF-16be.

#### 3.2.20.2.8. index\_info

- **index\_info** pragma trả về thông tin về chỉ mục (index) của CSDL.
- Cú pháp:

```
PRAGMA [database.]index_info(index_name);
```

- Tập kết quả sẽ gồm các cột chỉ số cho trình tự cột, chỉ số cột trong table và tên cột.

#### 3.2.20.2.9. index\_list

- **index\_list** pragma liệt kê tất cả các indexes được kết hợp với table.
- Cú pháp:

```
PRAGMA [database.]index_list(table_name);
```

- Tập kết quả sẽ có một hàng cho mỗi chỉ số cho dãy số, tên chỉ mục và cờ hiệu cho biết chỉ số là duy nhất hay không.

#### 3.2.20.2.10. journal\_mode

- **journal\_mode** pragma cho phép lấy hoặc thiết lập journal mode, giúp kiểm soát cách các file nhật ký được lưu trữ và xử lý.
- Cú pháp:

```
PRAGMA journal_mode;
PRAGMA journal_mode = mode;
PRAGMA database.journal_mode;
PRAGMA database.journal_mode = mode;
```

- SQLite hỗ trợ 5 journal modes:

Giá trị	Điễn giải
DELETE	(mặc định) lưu lại kết luận của một giao dịch, các file nhật ký sẽ bị xóa.
TRUNCATE	File nhật ký được cắt ngắn để có chiều dài là 0 (zero) byte.
PERSIST	File nhật ký được để lại tại chỗ, nhưng phần đầu (header) được ghi đè để cho biết journal là không còn giá trị.
MEMORY	File nhật ký được tổ chức trong bộ nhớ, chứ không phải là trên đĩa.
OFF	Không thực hiện việc ghi nhật ký.

#### 3.2.20.2.11. max\_page\_count

- **max\_page\_count** pragma cho phép lấy hoặc đặt mức tối đa số trang (pages) cho 1 CSDL.
- Cú pháp:

```
PRAGMA [database.]max_page_count;
PRAGMA [database.]max_page_count = max_page;
```

- Giá trị mặc định là 1.073.741.823 đó là một trong giga-page có nghĩa là nếu mặc định kích thước 1 KB trang, điều này cho phép cơ sở dữ liệu lớn lên thêm 1 terabyte.

### 3.2.20.2.12. *page\_count*

- *page\_count* pragma trả về số lượng trang (pages) hiện tại trong CSDL.
- Cú pháp:

```
PRAGMA [database.]page_count;
```

- Kích thước của file CSDL phải là *page\_count \* page\_size*.

### 3.2.20.2.13. *page\_size*

- *page\_size* pragma cho phép trả về hoặc lấy kích thước các trang (page) CSDL.
- Cú pháp:

```
PRAGMA [database.]page_size;
PRAGMA [database.]page_size = bytes;
```

- Mặc định, cho phép các kích thước 512, 1024, 2048, 4096, 8192, 16384, and 32768 bytes. Cách duy nhất để thay đổi kích thước trang trên 1 CSDL hiện có là thiết lập kích thước trang và sau đó ngay lập tức thu gọn CSDL.

### 3.2.20.2.14. *parser\_trace*

- *parser\_trace* pragma điều khiển in trạng thái gỡ lỗi (debugging state) như là phân tích cú pháp câu lệnh SQL.
- Cú pháp:

```
PRAGMA parser_trace = [true|false];
```

- Mặc định, pragma này được thiết lập là false.

### 3.2.20.2.15. *recursive\_triggers*

- *recursive\_triggers* pragma cho phép lấy hoặc thiết lập chức năng đệ quy của trigger. Nếu =false sẽ không cho phép trigger kích hoạt trigger khác.
- Cú pháp:

```
PRAGMA recursive_triggers;
PRAGMA recursive_triggers = [true|false];
```

### 3.2.20.2.16. *schema\_version*

- *schema\_version* pragma cho phép lấy hoặc thiết lập giá trị phiên bản của lược đồ, giá trị này được lưu trong púa CSDL
- Cú pháp:

```
PRAGMA [database.]schema_version;
PRAGMA [database.]schema_version = number;
```

- Đây là một giá trị số nguyên có dấu 32-bit giúp theo dõi những thay đổi sơ đồ. Bất cứ khi nào một lệnh làm thay đổi lược đồ được thực hiện (như, CREATE ... hoặc DROP ...), giá trị này được tăng lên.

### 3.2.20.2.17. *secure\_delete*

- *secure\_delete* pragma được sử dụng để kiểm soát nội dung sẽ bị xóa khỏi CSDL.
- Cú pháp:

```
PRAGMA secure_delete;
PRAGMA secure_delete = [true|false];
PRAGMA database.secure_delete;
PRAGMA database.secure_delete = [true|false];
```

- Thay đổi giá trị cho pragma này thông qua tùy chọn SQLITE\_SECURE\_DELETE.

### 3.2.20.2.18. *sql\_trace*

- *sql\_trace* pragma được sử dụng để đổ kết quả thực hiện SQL ra màn hình.
- Cú pháp:

```
PRAGMA sql_trace;
PRAGMA sql_trace = [true|false];
```

- SQLite phải được biên dịch bao gồm với các chỉ thị SQLITE\_DEBUG cho pragma này.

### 3.2.20.2.19. *synchronous*

- *synchronous* pragma cho phép lấy hoặc thiết lập chế độ đồng bộ hóa ô đĩa, giúp kiểm soát việc ghi dữ liệu ra để lưu trữ vật lý.

- Cú pháp:

```
PRAGMA [database.]synchronous;
PRAGMA [database.]synchronous = mode;
```

- SQLite hỗ trợ các chế độ đồng bộ sau (synchronisation modes):

Giá trị	Điễn giải
0 hoặc OFF	Không thực hiện đồng bộ
1 hoặc NORMAL	Thực hiện đồng bộ sau một chuỗi các hoạt động đĩa quan trọng.
2 hoặc FULL	Thực hiện đồng bộ sau mỗi hoạt động đĩa quan trọng.

### 3.2.20.2.20. temp\_store

- temp\_store** pragma cho phép lấy hoặc thiết lập chế độ lưu trữ được sử dụng bởi các file CSDL tạm thời (temporary database files).
- Cú pháp:

```
PRAGMA temp_store;
PRAGMA temp_store = mode;
```

- SQLite hỗ trợ các chế độ lưu trữ sau:

Giá trị	Điễn giải
0 hoặc DEFAULT	Sử dụng theo mặc định (thông thường FILE)
1 hoặc FILE	Lưu trữ trên cơ sở file
2 hoặc MEMORY	Lưu trữ trên cơ sở bộ nhớ.

### 3.2.20.2.21. temp\_store\_directory

- temp\_store\_directory** pragma cho phép lấy hoặc thiết lập vị trí sử dụng cho các file CSDL tạm thời.
- Cú pháp:

```
PRAGMA temp_store_directory;
PRAGMA temp_store_directory = 'directory_path';
```

### 3.2.20.2.22. user\_version

- user\_version** pragma cho phép lấy hoặc thiết lập giá trị phiên bản người dùng định nghĩa được lưu trữ trong tiêu đề (header) CSDL.
- Cú pháp:

```
PRAGMA [database.]user_version;
PRAGMA [database.]user_version = number;
```

- Đây là giá trị nguyên có dấu 32 bits. Có thể được thiết lập bởi các nhà phát triển cho mục đích theo dõi phiên bản.

### 3.2.20.2.23. writable\_schema

- writable\_schema** pragma cho phép lấy hoặc thiết lập khả năng sửa đổi các table hệ thống.
- Cú pháp:

```
PRAGMA writable_schema;
PRAGMA writable_schema = [true|false];
```

- Nếu pragma này được thiết lập, tên table bắt đầu bằng *sqlite\_* có thể được tạo ra và sửa đổi, bao gồm cả bảng *sqlite\_master*. Cần thận khi sử dụng pragma này bởi vì nó có thể dẫn hỏng CSDL.

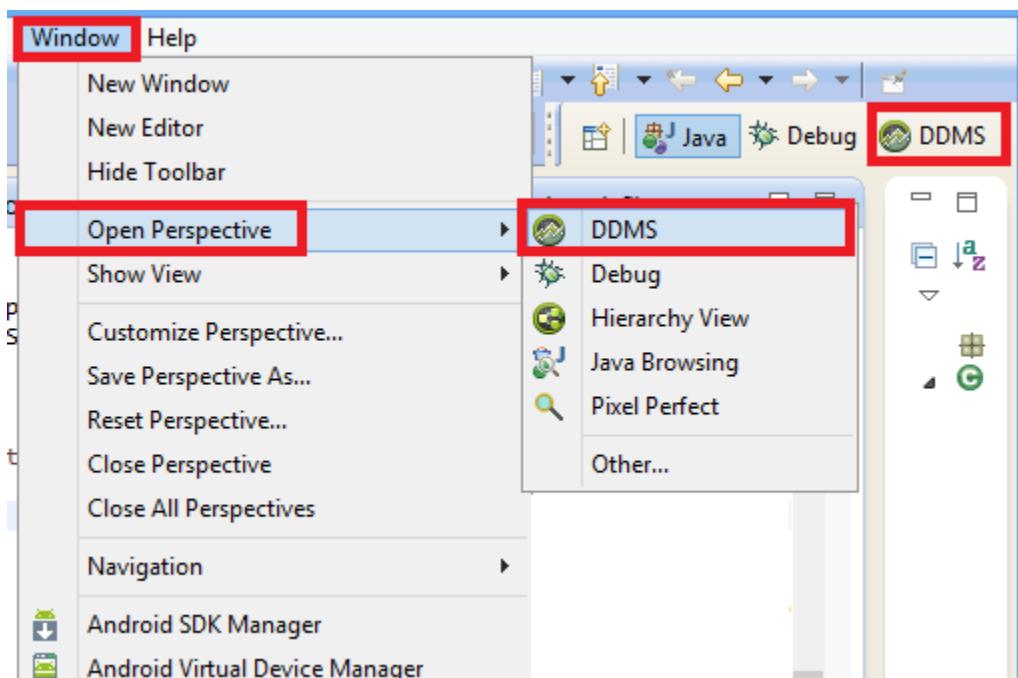
## 3.3. THAO TÁC VỚI FILE CỦA ỨNG DỤNG ĐƯỢC TẠO RA TRÊN AVD

Thao tác sau có thể sử dụng cho các loại file như Preferences, SQLite, ContentProvider và các dạng file khác

**B1.** Khởi chạy AVD

**B2.** Mở cửa sổ DDMS. Có thể thực hiện bằng 1 trong 2 cách

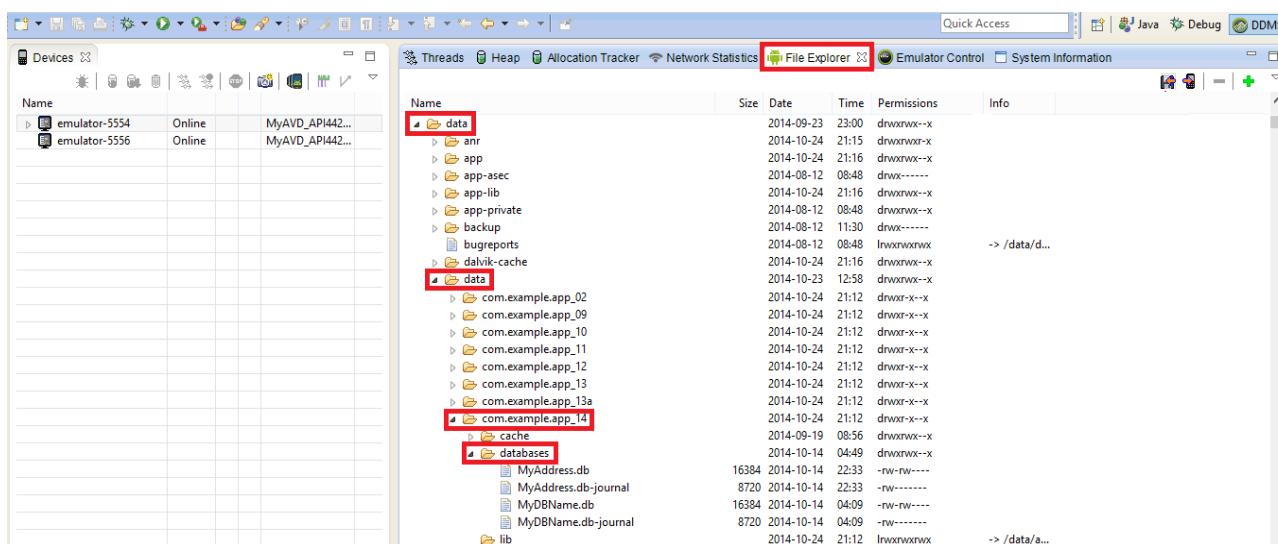
- Chọn button DDMS trên thanh công cụ
- Chọn menu Window\Open Perspective\DDMS



Hình 3-24 Minh họa mở cửa sổ DDMS

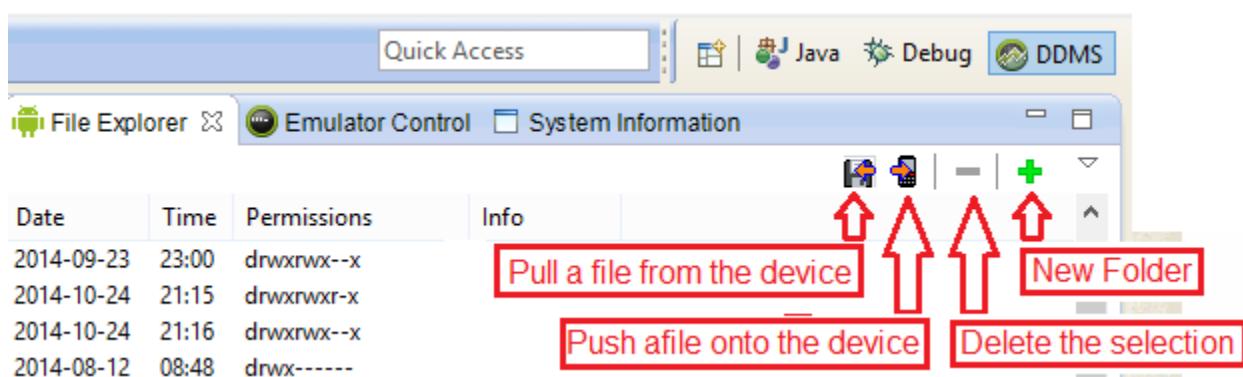
- B3.** Click chọn tab *File Explorer*. Giả sử cần tìm CSDL *MyDBName* được tạo ra trong bài thực hành *App\_14*. Ta chọn theo đường dẫn

data\data\com.example.app\_14\databases\MyDBName



Hình 3-25 Xác định vị trí file được tạo trong device

- B4.** Thanh công cụ của tab này gồm:



Hình 3-26 Các chức năng trong tab File Explorer

- *Pull a file from the device*: copy file đang chọn của device ra máy tính.
- *Push a file onto the device*: copy file từ máy tính vào device.
- *Delete the selection*: xóa file đang chọn (của device).
- *New Folder*: tạo mới folder trên device.

### 3.4. SỬ DỤNG CSDL SQLite TRONG Android

SQLite là một CSDL SQL mã nguồn mở giúp lưu trữ dữ liệu vào một file văn bản trên thiết bị. SQLite hỗ trợ tất cả các tính năng của CSDL quan hệ. Để truy cập vào CSDL loại này, bạn không cần phải thiết lập bất kỳ loại kết nối giống như JDBC, ODBC, ...

#### 3.4.1. Package

*android.database* chứa các interface/class làm việc với CSDL.

*android.database.sqlite* chứa các interface/class làm việc với SQLite.

#### 3.4.2. Class

Để thao tác với cơ sở dữ liệu lưu trữ trên SQLite trên Android ta sẽ làm việc với các loại đối tượng:

- *SQLiteOpenHelper*: đối tượng dùng để tạo, nâng cấp, đóng mở kết nối trên một CSDL
- *SQLiteDatabase*: Đối tượng dùng để thực thi các câu lệnh SQL trên một CSDL.
- *SQLiteQueryBuilder* tạo câu truy vấn SQL.
- *Cursor* chứa kết quả câu truy vấn.

#### 3.4.3. Cursor

##### 3.4.3.1. Giới thiệu

Kết quả trả về từ câu lệnh truy vấn là 1 đối tượng của class Cursor. Cursor giúp Android tiết kiệm bộ nhớ vì không cần phải load tất cả dữ liệu lên bộ nhớ.

Có thể lấy bất cứ thứ gì từ CSDL bằng cách sử dụng 1 đối tượng của class Cursor. Sử dụng phương thức *rawQuery* của class này để nhận về một tập kết quả với con trỏ (cursor) chỉ vào table chứa tập kết quả. Từ đó, bạn có thể di chuyển con trỏ về phía trước và lấy dữ liệu.

```
Cursor resultSet = mydatabase.rawQuery("Select * from User",null);
resultSet.moveToFirst();
String username = resultSet.getString(1);
String password = resultSet.getString(2);
```

Một số chức năng trong class Cursor:

- *close()*: close Cursor.
- *getColumnCount()*: trả về số lượng cột có trong table.
- *getColumnIndex(String columnName)*: trả về số thứ tự của cột trong table có tên được chỉ ra.
- *getColumnName(int columnIndex)*: trả về tên của cột dựa vào số thứ tự của cột.
- *getColumnNames()*: trả về tên của tất cả các cột có trong table.
- *getCount()*: trả về số lượng dòng có trong cursor.
- *getLong(Col\_index)* hoặc *getString(Col\_index)*: lấy giá trị tại vị trí giao nhau giữa cột thứ Col\_index và dòng hiện tại.
- *getPosition()*: trả về vị trí hiện tại của cursor trong table.
- *isAfterLast()*: kiểm tra xem đã di chuyển đến cuối Cursor hay chưa.
- *isClosed()*: trả về true nếu cursor đang đóng, ngược lại sẽ trả về false.

##### 3.4.3.2. Minh họa 1 số hàm sử dụng cursor

(<http://www.programcreek.com/java-api-examples/index.php?api=android.database.Cursor>)

#### 3.4.3.2.1. Example 1

```
/* Trả về 1 List liệt kê tất cả các filters*/
public List<Filter> getAllFilters()
{
 List<Filter> filters=new ArrayList<Filter>();
 Cursor cursor=mDb.query(DATABASE_TABLE, new
 String[]{KEY_ROWID,KEY_FILTERNAME,KEY_FILTER_KEYWORDS},
 null,null,null,null,null);
 cursor.moveToFirst();
 while (!cursor.isAfterLast())
 {
 String name=cursor.getString(1);
 String[] words=cursor.getString(2).split(Filter.KEYWORD_DELIMITER);
 Set<String> set=new HashSet<String>(Arrays.asList(words));
 filters.add(new Filter(name,(KEY_USED == "1"),set));
 cursor.moveToNext();
 }
 cursor.close();
 return filters;
}
```

#### 3.4.3.2.2. Example 2

```
/* Trả về 1 List liệt kê tất cả các filters được sử dụng*/
public List<Filter> getUsedFilters()
{
 List<Filter> filters=new ArrayList<Filter>();
 Cursor cursor=mDb.query(DATABASE_TABLE,new String[]{KEY_ROWID,KEY_FILTERNAME,
 KEY_FILTER_KEYWORDS, KEY_USED}, KEY_USED + "=\\'1\\'",null,null,null,null);
 cursor.moveToFirst();
 while (!cursor.isAfterLast())
 {
 String name=cursor.getString(1);
 String[] words=cursor.getString(2).split(Filter.KEYWORD_DELIMITER);
 Set<String> set=new HashSet<String>(Arrays.asList(words));
 filters.add(new Filter(name,(KEY_USED == "1"),set));
 cursor.moveToNext();
 }
 cursor.close();
 return filters;
}
```

#### 3.4.3.2.3. Example 3

```
/* Trả về 1 Filter với tên filter là tham số NameFilter */
public Filter getFilter(String NameFilter)
{
 Cursor cursor=mDb.query(DATABASE_TABLE,new String[]{KEY_ROWID,
 KEY_FILTERNAME, KEY_FILTER_KEYWORDS}, KEY_FILTERNAME + "=\\'" + NameFilter +
 "\\'",null,null,null,null,null);
 if (cursor.getCount() > 0)
 {
 cursor.moveToFirst();
 String[] words=cursor.getString(2).split(Filter.KEYWORD_DELIMITER);
 Set<String> set=new HashSet<String>(Arrays.asList(words));
 return new Filter(NameFilter, (KEY_USED == "1"),set);
 }
 else
 {
 return null;
 }
}
```

**3.4.3.2.4. Example 4**

```
/* Xóa một ảnh từ CSDL của ứng dụng. Thao tác này cũng xóa các file hình ảnh có
liên quan. Với idPhoto là ID của ảnh cần xóa. Hàm trả về kết quả của việc xóa có
thành công hay không (true or false) */
public boolean deletePhotoEntry(long idPhoto)
{
 SQLiteDatabase db=mDbHelper.getWritableDatabase();
 Cursor c=db.query(DatabaseHelper.TABLE_NAME,null,KEY_ID + "=" +
 idPhoto,null,null,null,null);
 File photoPath;
 boolean deletedFile=false;
 if (c.moveToFirst())
 {
 photoPath=new File(c.getString(c.getColumnIndexOrThrow(KEY_FILENAME)));
 deletedFile=photoPath.delete();
 }
 int deletedEntry=db.delete(DatabaseHelper.TABLE_NAME,KEY_ID + "=?",
 new String[]{String.valueOf(idPhoto)});
 c.close();
 db.close();
 return (deletedEntry > 0) && deletedFile;
}
```

**3.4.3.2.5. Example 5**

```
/* Thực hiện tương tự như Example 4 (deletePhotoEntry()), với khác biệt là chỉ
xóa thông tin trong CSDL mà không xóa file hình ảnh có liên quan */
public boolean deletePhotoDBEntry(long idPhoto)
{
 SQLiteDatabase db=mDbHelper.getWritableDatabase();
 Cursor c=db.query(DatabaseHelper.TABLE_NAME,null,KEY_ID + "=" +
 idPhoto, null, null, null, null);
 int deletedEntry=db.delete(DatabaseHelper.TABLE_NAME,KEY_ID + "=?",
 new String[]{String.valueOf(idPhoto)});
 c.close();
 db.close();
 return (deletedEntry > 0);
}
```

**3.4.3.2.6. Example 6**

```
/* Xóa tất cả các ảnh có trong CSDL */
public void deleteAllPhotoEntries()
{
 SQLiteDatabase db = mDbHelper.getWritableDatabase();
 Cursor c = db.query(DatabaseHelper.TABLE_NAME,
 new String[]{KEY_FILENAME}, null, null, null, null, null);
 File f;
 while (c.moveToNext())
 {
 f=new File(c.getString(c.getColumnIndexOrThrow(KEY_FILENAME)));
 f.delete();
 }
 db.delete(DatabaseHelper.TABLE_NAME, null, null);
 c.close();
 db.close();
}
```

**3.4.3.2.7. Example 7**

```
/* Trả về photo entry thông qua tham số idPhoto truyền cho hàm*/
public PhotoEntry getPhotoEntry(long idPhoto)
{
 SQLiteDatabase db=mDbHelper.getReadableDatabase();
 Cursor cs = db.query(DatabaseHelper.TABLE_NAME, null, KEY_ID + "=" +
```

```

 idPhoto, null, null, null, null);
PhotoEntry pe = new PhotoEntry();
if (cs.moveToFirst())
{
 pe.setId(cs.getLong(cs.getColumnIndexOrThrow(KEY_ID)));
 pe.setTimeStamp(cs.getString(cs.getColumnIndexOrThrow(KEY_TIMESTAMP)));
 pe.setTag(cs.getString(cs.getColumnIndexOrThrow(KEY_TAG)));
 pe.setFilePath(cs.getString(cs.getColumnIndexOrThrow(KEY_FILENAME)));
}
cs.close();
db.close();
return pe;
}

```

#### 3.4.3.2.8. Example 8

```

/* Trả về 1 mảng các chuỗi của tất cả các tags có trong CSDL */
public String[] getAllTags()
{
 SQLiteDatabase db=mDbHelper.getReadableDatabase();
 Cursor c=db.query(true,DatabaseHelper.TABLE_NAME,
 new String[]{KEY_TAG}, null, null, null, null, null);
 ArrayList<String> allTags=new ArrayList<String>();
 while (c.moveToNext())
 {
 allTags.add(c.getString(c.getColumnIndexOrThrow(KEY_TAG)));
 }
 c.close();
 db.close();
 return allTags.toArray(new String[c.getCount()]);
}

```

#### 3.4.3.2.9. Example 9

```

/* Trả về 1 mảng các chuỗi của tất cả các thẻ có chứa các truy vấn nhất định.
Khác so với Example 8 (getAllTags) là có nhận 1 tham số là chuỗi query */
public String[] getMatchingTags(String query)
{
 SQLiteDatabase db=mDbHelper.getReadableDatabase();
 Cursor c=db.query(true,DatabaseHelper.TABLE_NAME,new
 String[]{KEY_TAG},null,null,null,null,null);
 ArrayList<String> tags=new ArrayList<String>();
 while (c.moveToNext())
 {
 if (c.getString(c.getColumnIndexOrThrow(KEY_TAG)).contains(query))
 {
 tags.add(c.getString(c.getColumnIndexOrThrow(KEY_TAG)));
 }
 }
 c.close();
 db.close();
 return tags.toArray(new String[tags.size()]);
}

```

#### 3.4.3.2.10. Example 10

```

/* Xóa thẻ quy định và tất cả các mục hình ảnh kết hợp với thẻ. Hữu ích khi
người dùng muốn ngừng theo dõi các điều kiện quy định. Nhận tham số là tag cần
xóa. Trả về giá trị boolean cho biết thao tác thành công hay không? */
public boolean deleteTagAndPhotoEntries(String tag)
{
 SQLiteDatabase db=mDbHelper.getWritableDatabase();
 Cursor c=db.query(DatabaseHelper.TABLE_NAME,new String[]{KEY_FILENAME},
 KEY_TAG + "=?", new String[]{tag}, null, null, null);
 deleteFilesFromCursor(c);
 int rowCount=0;

```

```

 int deletedEntries=db.delete(DatabaseHelper.TABLE_NAME,KEY_TAG + "=?",
 new String[]{tag});
 c.close();
 db.close();
 return (deletedEntries == rowCount);
 }
}

```

#### 3.4.3.2.11. Example 11

```

/* đếm số lượng photo có trong CSDL */
public int getPhotoEntryCount()
{
 SQLiteDatabase db=mDbHelper.getReadableDatabase();
 Cursor c=db.query(DatabaseHelper.TABLE_NAME,
 new String[]{KEY_ID}, null, null, null, null, null);
 int entryCount=c.getCount();
 c.close();
 db.close();
 return entryCount;
}

```

#### 3.4.3.2.12. Example 12

```

/* Nhận tham số là 1 tag. Trả về 1 ArrayList chứa tất cả các hình ảnh liên quan
đến tham số tag. Dữ liệu này nên được sắp xếp với chỉ số (index) tính từ 0 */
public ArrayList<PhotoEntry> getAllPhotoEntriesWithTag(String tag)
{
 SQLiteDatabase db=mDbHelper.getReadableDatabase();
 Cursor c=db.query(DatabaseHelper.TABLE_NAME, null,KEY_TAG + "=?",
 new String[]{tag}, null, null, null);
 ArrayList<PhotoEntry> entries=new ArrayList<PhotoEntry>();
 while (c.moveToNext())
 {
 PhotoEntry e=new PhotoEntry();
 e.setId(c.getInt(c.getColumnIndexOrThrow(KEY_ID)));
 e.setTimeStamp(c.getString(c.getColumnIndexOrThrow(KEY_TIMESTAMP)));
 e.setTag(c.getString(c.getColumnIndexOrThrow(KEY_TAG)));
 e.setFilePath(c.getString(c.getColumnIndexOrThrow(KEY_FILENAME)));
 entries.add(e);
 }
 c.close();
 db.close();
 return entries;
}

```

### 3.4.4. SQLiteOpenHelper class

Trên mỗi CSDL của ứng dụng sẽ có một đối tượng *SQLiteOpenHelper* dùng để dùng để tạo mới, nâng cấp cũng như đóng mở CSDL. Thông qua việc sử dụng đối tượng *SQLiteOpenHelper*, bạn sẽ lấy về một đối tượng kiểu *SQLiteDatabase*, đối tượng kiểu *SQLiteDatabase* chính là thể hiện của CSDL ta cần thao tác.

- *SQLiteOpenHelper* hỗ trợ hai phương thức để lấy về một đối tượng *SQLiteDatabase* là:
  - *getReadableDatabase()*: Lấy về một đối tượng *SQLiteDatabase* ở dạng “chỉ đọc”.
  - *getWritableDatabase()*: Lấy về một đối tượng *SQLiteDatabase* ở dạng “đọc và ghi”.
- Các phương thức chính:
  - *Phương thức khởi tạo (constructor)*: cần gọi *super()* với tham số thứ 2 là tên CSDL và tham số cuối là phiên bản hiện tại.
  - *Override phương thức onCreate*: Phương thức này được dùng để khởi tạo khi CSDL, table chưa tồn tại và có thể dùng để nhập dữ liệu ban đầu cho table. Tham số là đối tượng *SQLiteDatabase* của CSDL hiện tại.

- **Override phương thức onUpgrade:** Phương thức này được gọi khi phiên bản CSDL được tăng lên. Tham số lần lượt là đối tượng *SQLiteDatabase* của CSDL hiện tại, phiên bản cũ, phiên bản mới.

Ví dụ về phiên bản của CSDL:

- *Phiên bản 01:* ứng dụng chỉ mới thực hiện các tính năng quản lý về dữ liệu của
  - ✓ Học kỳ.
  - ✓ Môn học.
  - ✓ Class học.
- *Phiên bản 02:* bổ sung thêm dữ liệu về:
  - ✓ Sinh viên
  - ✓ Kết quả học tập

– Cú pháp:

```
public class DBHelper extends SQLiteOpenHelper
{
 public DBHelper()
 {
 super(context, DATABASE_NAME, null, 1);
 }

 public void onCreate(SQLiteDatabase db) {}

 public void onUpgrade(SQLiteDatabase database, int oldVersion,
 int newVersion) {}
}
```

### 3.4.5. SQLiteDatabase class

*SQLiteDatabase* class là base class, cho phép bạn làm việc với CSDL. Các phương thức chủ yếu cần xây dựng cho class này

#### 3.4.5.1. getDatabase():

- Mô tả: Dùng để lấy về một đối tượng kiểu *SQLiteDatabase*, đây chính là đối tượng CSDL của ứng dụng.

#### 3.4.5.2. closeAllDatabase():

- Mô tả: Dùng để đóng toàn bộ các CSDL đã được mở. Phương thức này đơn giản sẽ gọi phương thức *close()* của class *SQLiteOpenHelper*.

#### 3.4.5.3. Phương thức execSQL

- Mô tả: Dùng để thực thi một câu lệnh truy vấn.
- Các dạng sử dụng phương thức *execSQL*:

- (i). Phương thức *mySQLiteDatabase.execSQL*: thường dùng để tạo table hoặc chèn dữ liệu vào table:

```
mySQLiteDatabase.execSQL("CREATE TABLE IF NOT EXISTS User (Username
 VARCHAR, Password VARCHAR);");
mySQLiteDatabase.execSQL("INSERT INTO User
 VALUES ('admin', 'admin');");

```

- (ii). Phương thức *execSQL*

```
execSQL(String sql, Object[] bindArgs)
```

- Nhờ đối số là lệnh SQL nên phương thức này không chỉ dùng trong việc chèn dữ liệu, mà còn được sử dụng để cập nhật hoặc sửa đổi dữ liệu đã tồn tại trong CSDL nhờ vào đối số ràng buộc là 1 mảng Object.
- Khi insert/update dữ liệu, bạn có thể dùng *ContentValues* class để định nghĩa dữ liệu theo dạng key/value. Trong đó “key” là tên cột, “value” là giá trị của cột đó.

- (iii). Phương thức *query*

```
mySQLiteDatabase.query(table, columns, selection, selectionArgs,
 groupBy, having, orderBy)
```

Trong đó:

- *String table:* Tên bảng cần truy vấn dữ liệu.
- *String[] columns:* Mảng chứa tên các cột cần lấy dữ liệu.

- *String selection*: Mệnh đề điều kiện của câu truy vấn (mệnh đề WHERE). Mệnh đề này có dạng: “id=? AND name=?”. Trong đó giá trị của id và name sẽ được chỉ định ở tham số *String[] selectionArgs*.
- *String[] selectionArgs*: Tham số của mệnh đề điều kiện ở trên.
- *String groupBy*: Điều kiện GROUP BY.
- *String having*: Điều kiện HAVING.
- *String orderBy*: Điều kiện ORDER BY.

**Output:** Con trỏ Cursor chỉ đến dòng dữ liệu đầu tiên. Ta sẽ sử dụng con trỏ này để đọc các dòng dữ liệu trên tập dữ liệu được trả về.

#### 3.4.5.4. Phương thức insert():

- Mô tả:
  - Đây là phương thức dùng để thêm một dòng vào CSDL.
  - Phương thức này sẽ tạo một đối tượng database kiểu SQLiteDatabase. Đây là đối tượng được trả về bởi phương thức *getWritableDatabase()* được định nghĩa trong class SQLiteOpenHelper. Phương thức này cho phép lấy về đối tượng SQLiteDatabase ở dạng “đọc và ghi”. Ta sẽ sử dụng đối tượng database này để thực thi câu truy vấn.
- Cú pháp

```
mySQLiteDatabase.insert(tableName, null, ContentValues values)
```

Trong đó:

- *String tableName*: Tên bảng cần thêm dữ liệu.
- *ContentValues values*: Đối tượng chứa dữ liệu của các cột cần thêm vào. Đối tượng này gần giống với kiểu dữ liệu từ điển. Trong đó key sẽ là tên cột cần thêm dữ liệu và value là dữ liệu cần thêm.

**Output:** Số dòng dữ liệu thêm thành công.

#### 3.4.5.5. Phương thức update():

- Mô tả:
  - Đây là phương thức dùng để cập nhật một dòng trong CSDL.
- Cú pháp

```
mySQLiteDatabase.update(tableName, values, whereClause, params)
```

Trong đó:

- *String tableName*: Tên bảng cần cập nhật.
- *ContentValues values*: Đối tượng chứa dữ liệu của các cột cần cập nhật. Đối tượng này gần giống với kiểu dữ liệu từ điển. Trong đó key sẽ là tên cột và value là dữ liệu cần cập nhật.
- *String whereClause*: Mệnh đề điều kiện của dòng cần cập nhật dữ liệu (mệnh đề WHERE). Mệnh đề này có dạng: “id=? AND name=?”. Trong đó giá trị của id và name sẽ được chỉ định ở tham số *String[] parms*.
- *String[] parms*: Tham số của mệnh đề điều kiện ở trên.

**Output:** Số dòng cập nhật thành công.

#### 3.4.5.6. Phương thức delete():

- Mô tả: Đây là phương thức dùng xóa một dòng trong CSDL.
- Cú pháp

```
mySQLiteDatabase.delete(tableName, whereClause, params)
```

Trong đó:

- *String tableName*: Tên bảng cần xóa dữ liệu.
- *String whereClause*: Mệnh đề điều kiện của dòng cần xóa (mệnh đề WHERE). Mệnh đề này có dạng: “id=? AND name=?”. Trong đó giá trị của id và name sẽ được chỉ định ở tham số *String[] parms*.
- *String[] parms*: Tham số của mệnh đề điều kiện ở trên.

**Output:** Số dòng xóa thành công.

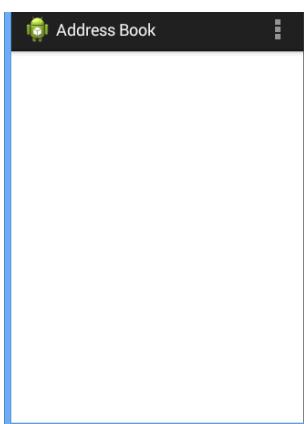
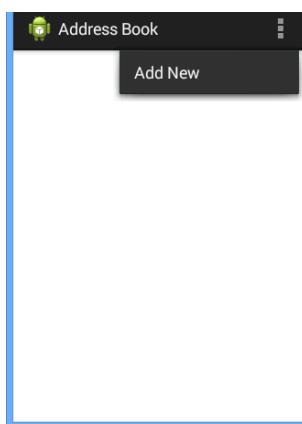
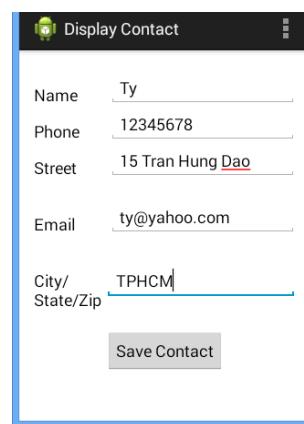
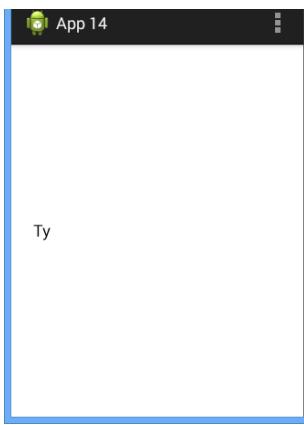
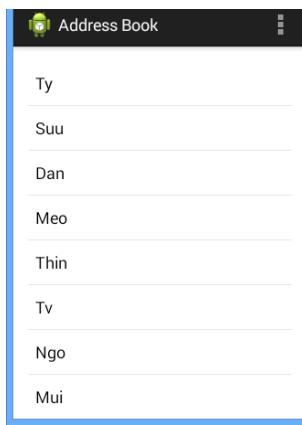
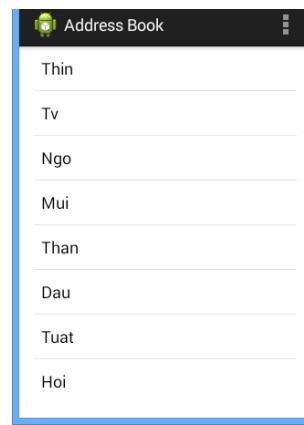
**BÀI THỰC HÀNH App\_14****☞ Yêu cầu:**

Tạo mới project App14. Tạo lặp để *MainActivity.java* (activity thứ 1) có giao diện như sau:

*Hình 3-27:* Ứng dụng Address Book khi khởi chạy lần đầu.

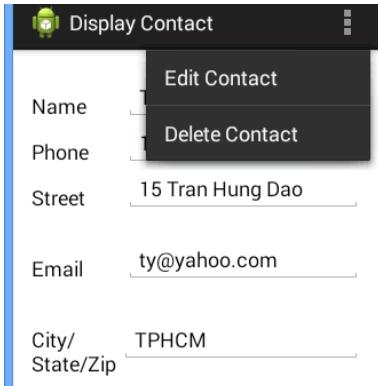
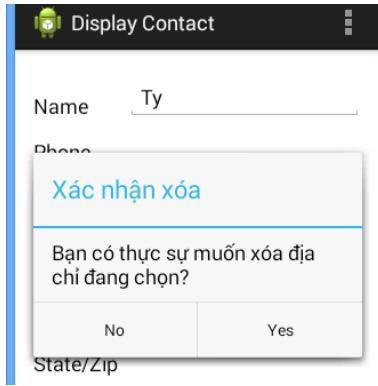
*Hình 3-28:* nhấn chọn menu *Add New*.

*Hình 3-29:* Sau khi nhấn chọn menu *Add New*, xuất hiện activity *Display Contact* cho phép nhập thông tin.

*Hình 3-27**Hình 3-28**Hình 3-29**Hình 3-30**Hình 3-31**Hình 3-32*

*Hình 3-30:* Trở lại màn hình chính sau khi nhấn chọn button *Save Contact* (hình 3-16).

*Hình 3-31 và Hình 3-32:* Màn hình chính sau khi đã nhập thông tin về nhiều người. Chú ý màn hình có thể cuộn lên xuống.

*Hình 3-33**Hình 3-34*

*Hình 3-33 và 3-34:* Khi nhấn chọn 1 tên trong màn hình chính sẽ đưa đến activity Display Contact. Có thể chọn 1 trong 2 chức năng cần thực hiện trên menu của activity này.

☛ **Thực hiện:**

Để thực hiện ứng dụng cần thực hiện các công việc chính sau:

- (i). Tạo 2 giao diện (layout) cho ứng dụng.
- (ii). Xây dựng class xử lý dữ liệu.
- (iii). Tạo menu cho từng activity (2 menu cho 2 activity).
- (iv). Viết các class xử lý cho các activity.

**B1.** Tạo mới project App\_14. Lưu ý chọn version mới nhất cho *Target SDK* và chọn level API cao đối với *Compile With*.

**B2.** Chỉnh sửa và tạo mới các layout cần sử dụng

**B2.1.** Chỉnh sửa nội dung file layout chính của ứng dụng (*activity\_main.xml*) để hiển thị tên của những người có địa chỉ được lưu

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:paddingBottom="@dimen/activity_vertical_margin"
 android:paddingLeft="@dimen/activity_horizontal_margin"
 android:paddingRight="@dimen/activity_horizontal_margin"
 android:paddingTop="@dimen/activity_vertical_margin"
 tools:context=".MainActivity" >
 <ListView android:id="@+id/listView1"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_centerHorizontal="true"
 android:layout_centerVertical="true" >
 </ListView>
</RelativeLayout>
```

**B2.2.** Tạo thêm 1 file layout với tên *display\_contact.xml* để hiển thị thông tin chi tiết về địa chỉ của từng người.

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:id="@+id/scrollView1"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 tools:context=".DisplayContact" >
 <TableLayout android:layout_width="fill_parent"
 android:layout_height="wrap_content">
 <TableRow>
 <TextView android:id="@+id/textView1"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="Name"
 android:textAppearance="?android:attr/textAppearanceMedium" />
 <EditText android:id="@+id/editTextName"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:ems="10"
 android:inputType="text" >
 <requestFocus />
 </EditText>
 </TableRow>
 <TableRow>
 <TextView android:id="@+id/textView5"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Phone"
 android:textAppearance="?android:attr/textAppearanceMedium" />
 <EditText android:id="@+id/editTextPhone" />
 </TableRow>
 </TableLayout>
</ScrollView>
```

```

 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:inputType="phone/text" />
 </TableRow>
 <TableRow>
 <EditText android:id="@+id/editTextStreet"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Street"
 android:textAppearance="?android:attr/textAppearanceMedium" />
 <EditText android:id="@+id/editTextEmail"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:inputType="text" />
 </TableRow>
 <TableRow>
 <EditText android:id="@+id/editTextCity"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="City/State/Zip"
 android:textAppearance="?android:attr/textAppearanceMedium" />
 <EditText android:id="@+id/button1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:onClick="run"
 android:layout_span="2"
 android:text="Save Contact" />
 </TableRow>
</TableLayout>
</ScrollView>

```

### B3. Tạo menu cho các activity

**B3.1.** Chỉnh sửa file menu tên *res\menu\main.xml* cung cấp menu cho activity thứ 2 (activity AddressBook) với nội dung:

```

<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
 <item android:id="@+id/item1"
 android:title="@string/Add_New"
 android:showAsAction="always/collapseActionView">
 </item>
</menu>

```

**B3.2.** Tạo mới thêm 1 file menu tên *res\menu\display\_contact\_menu.xml* cung cấp menu cho activity thứ 3 (activity DisplayContact) với nội dung:

```
<?xml version="1.0" encoding="utf-8"?>
```

```

<menu xmlns:android="http://schemas.android.com/apk/res/android" >
 <item android:id="@+id/Edit_Contact"
 android:orderInCategory="100"
 android:title="@string/edit"/>
 <item android:id="@+id/Delete_Contact"
 android:orderInCategory="100"
 android:title="@string/delete"/>
</menu>

```

- B4. Tạo mới 1 class có chức năng xử lý dữ liệu và đặt tên là src\com\example\app\_14\DBHelper.java với nội dung.

```

package com.example.app_14;
import java.util.ArrayList;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.DatabaseUtils;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteDatabase;

public class DBHelper extends SQLiteOpenHelper
{
 public static final String DATABASE_NAME = "MyAddress.db";
 public static final String CONTACTS_TABLE_NAME = "contacts";
 public static final String CONTACTS_COLUMN_ID = "id";
 public static final String CONTACTS_COLUMN_NAME = "name";
 public static final String CONTACTS_COLUMN_EMAIL = "email";
 public static final String CONTACTS_COLUMN_STREET = "street";
 public static final String CONTACTS_COLUMN_CITY = "place";
 public static final String CONTACTS_COLUMN_PHONE = "phone";

 public DBHelper(Context context)
 { super(context, DATABASE_NAME , null, 1);
 }
 @Override
 public void onCreate(SQLiteDatabase db)
 {
 // TODO Auto-generated method stub
 db.execSQL("create table contacts (id integer primary key, name text,
 phone text,email text, street text, place text)");
 }
 @Override
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
 // TODO Auto-generated method stub
 db.execSQL("DROP TABLE IF EXISTS contacts");
 onCreate(db);
 }
 public boolean insertContact (String name, String phone, String email,
 String street, String place)
 {
 SQLiteDatabase db = this.getWritableDatabase();
 ContentValues contentValues = new ContentValues();

 contentValues.put("name", name);
 contentValues.put("phone", phone);
 contentValues.put("email", email);
 contentValues.put("street", street);
 contentValues.put("place", place);

 db.insert("contacts", null, contentValues);
 return true;
 }
}

```

```

 }
 public Cursor getData(int id){
 SQLiteDatabase db = this.getReadableDatabase();
 Cursor res = db.rawQuery("select * from contacts where id='"+id+"'", null);
 return res;
 }
 public int numberOfRows(){
 SQLiteDatabase db = this.getReadableDatabase();
 int numRows = (int) DatabaseUtils.queryNumEntries(db, CONTACTS_TABLE_NAME);
 return numRows;
 }
 public boolean updateContact (Integer id, String name, String phone, String email,
 String street, String place)
 {
 SQLiteDatabase db = this.getWritableDatabase();
 ContentValues contentValues = new ContentValues();
 contentValues.put("name", name);
 contentValues.put("phone", phone);
 contentValues.put("email", email);
 contentValues.put("street", street);
 contentValues.put("place", place);
 db.update("contacts",contentValues,"id = ?", new String[]{Integer.toString(id)});
 return true;
 }
 public Integer deleteContact (Integer id)
 {
 SQLiteDatabase db = this.getWritableDatabase();
 return db.delete("contacts", "id = ? ", new String[] { Integer.toString(id) });
 }
 public ArrayList getAllCotacts()
 {
 ArrayList array_list = new ArrayList();
 SQLiteDatabase db = this.getReadableDatabase();
 Cursor res = db.rawQuery("select * from contacts", null);
 res.moveToFirst();
 while(res.isAfterLast() == false)
 {
 array_list.add(res.getString(res.getColumnIndex(CONTACTS_COLUMN_NAME)));
 res.moveToNext();
 }
 return array_list;
 }
}

```

- B5. Tạo thêm 1 activity tên *DisplayContact.java*. activity này sẽ sử dụng file layout của file *display\_contact.xml* với nội dung như sau:

```

package com.example.app_14;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class DisplayContact extends Activity {
 private DBHelper mydb ;
 TextView name ;

```

```
TextView phone;
TextView email;
TextView street;
TextView place;
int id_To_Update = 0;

@Override
protected void onCreate(Bundle savedInstanceState)
{
 super.onCreate(savedInstanceState);
 setContentView(R.layout.display_contact);
 name = (TextView) findViewById(R.id.editTextName);
 phone = (TextView) findViewById(R.id.editTextPhone);
 email = (TextView) findViewById(R.id.editTextStreet);
 street = (TextView) findViewById(R.id.editTextEmail);
 place = (TextView) findViewById(R.id.editTextCity);

 mydb = new DBHelper(this);

 Bundle extras = getIntent().getExtras();
 if(extras !=null)
 {
 int Value = extras.getInt("id");
 if(Value>0){
 //means this is the view part not the add contact part.
 Cursor rs = mydb.getData(Value);
 id_To_Update = Value;
 rs.moveToFirst();
 String nam =rs.getString(rs.getColumnIndex(DBHelper.CONACTS_COLUMN_NAME));
 String phon = rs.getString(rs.getColumnIndex
 (DBHelper.CONACTS_COLUMN_PHONE));
 String emai = rs.getString(rs.getColumnIndex
 (DBHelper.CONACTS_COLUMN_EMAIL));
 String stree = rs.getString(rs.getColumnIndex
 (DBHelper.CONACTS_COLUMN_STREET));
 String plac = rs.getString(rs.getColumnIndex
 (DBHelper.CONACTS_COLUMN_CITY));
 if (!rs.isClosed())
 {
 rs.close();
 }
 Button b = (Button)findViewById(R.id.button1);
 b.setVisibility(View.INVISIBLE);

 name.setText((CharSequence)nam);
 name.setFocusable(false);
 name.setClickable(false);

 phone.setText((CharSequence)phon);
 phone.setFocusable(false);
 phone.setClickable(false);

 email.setText((CharSequence)emai);
 email.setFocusable(false);
 email.setClickable(false);

 street.setText((CharSequence)stree);
 street.setFocusable(false);
 street.setClickable(false);

 place.setText((CharSequence)plac);
 place.setFocusable(false);
 place.setClickable(false);
 }
 }
}
```

```
 }
 @Override
 public boolean onCreateOptionsMenu(Menu menu)
 {
 getMenuInflater().inflate(R.menu.display_contact_menu, menu);
 return true;
 }
 public boolean onOptionsItemSelected(MenuItem item)
 {
 super.onOptionsItemSelected(item);
 switch(item.getItemId())
 {
 case R.id.Edit_Contact:
 Button b = (Button)findViewById(R.id.button1);
 b.setVisibility(View.VISIBLE);
 name.setEnabled(true);
 name.setFocusableInTouchMode(true);
 name.setClickable(true);

 phone.setEnabled(true);
 phone.setFocusableInTouchMode(true);
 phone.setClickable(true);

 email.setEnabled(true);
 email.setFocusableInTouchMode(true);
 email.setClickable(true);

 street.setEnabled(true);
 street.setFocusableInTouchMode(true);
 street.setClickable(true);

 place.setEnabled(true);
 place.setFocusableInTouchMode(true);
 place.setClickable(true);

 return true;
 case R.id.Delete_Contact:
 AlertDialog.Builder builder = new AlertDialog.Builder(this);
 builder.setMessage("Bạn có thực sự muốn xóa địa chỉ đang chọn?")
 .setPositiveButton("Yes", new DialogInterface.OnClickListener()
 {
 public void onClick(DialogInterface dialog, int id)
 {
 mydb.deleteContact(id_To_Update);
 Toast.makeText(getApplicationContext(), "Xóa thành công",
 Toast.LENGTH_SHORT).show();
 Intent intent = new Intent(getApplicationContext(),
 com.example.app_14.MainActivity.class);
 startActivity(intent);
 }
 })
 .setNegativeButton("No", new DialogInterface.OnClickListener()
 {
 public void onClick(DialogInterface dialog, int id)
 {
 // User cancelled the dialog
 }
 });
 AlertDialog d = builder.create();
 d.setTitle("Xác nhận xóa");
 d.show();
 return true;
 default:
 return super.onOptionsItemSelected(item);
 }
 }
 // Hàm xử lý sự kiện khi người dùng nhấn trên button Save Contact
 /* Thuộc tính android:onClick="run" của button Save Contact trong file layout
display_contact.xml sẽ gọi hàm run thực hiện */

```

```
public void run(View view)
{
 Bundle extras = getIntent().getExtras();
 if(extras !=null)
 {
 int Value = extras.getInt("id");
 if(Value>0)
 { if(mydb.updateContact(id_To_Update,name.getText().toString(),
 phone.getText().toString(), email.getText().toString(),
 street.getText().toString(), place.getText().toString()))
 {
 Toast.makeText(getApplicationContext(), "Cập nhật thành công",
 Toast.LENGTH_SHORT).show();
 Intent intent = new Intent(getApplicationContext(),
 com.example.app_14.MainActivity.class);
 startActivity(intent);
 }
 else
 { Toast.makeText(getApplicationContext(), "Cập nhật KHÔNG thành công",
 Toast.LENGTH_SHORT).show();
 }
 }
 else
 { if(mydb.insertContact(name.getText().toString(),
 phone.getText().toString(), email.getText().toString(),
 street.getText().toString(), place.getText().toString()))
 { Toast.makeText(getApplicationContext(), "Thêm mới thành công",
 Toast.LENGTH_SHORT).show();
 }
 else
 { Toast.makeText(getApplicationContext(), "Thêm mới KHÔNG thành công",
 Toast.LENGTH_SHORT).show();
 }
 Intent intent = new Intent(getApplicationContext(),
 com.example.app_14.MainActivity.class);
 startActivity(intent);
}
}
```

**B6.** Bổ sung activity *DisplayContact.java* vào file *Manifest.xml* để có nội dung như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.app_14"
 android:versionCode="1"
 android:versionName="1.0" >
 <uses-sdk
 android:minSdkVersion="17"
 android:targetSdkVersion="21" />
 <application
 android:allowBackup="true"
 android:icon="@drawable/ic_launcher"
 android:label="@string/app_name"
 android:theme="@style/AppTheme" >
 <activity
 android:name=".MainActivity"
 android:label="@string/app_name" >
 <intent-filter>
 <action android:name="android.intent.action.MAIN" />
 <category android:name="android.intent.category.LAUNCHER" />
 </intent-filter>
 </activity>
 <activity
 android:name=".DisplayContact"
 android:label="Display Contact" >
```

```

</activity>
</application>
</manifest>
```

- B7. Chính sửa nội dung của file *src/com/example/MainActivity.java*, activity này sẽ sử dụng file layout của file *activity\_main.xml* với nội dung như sau:

```

package com.example.app_14;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import java.util.ArrayList;
import android.content.Intent;
import android.view.KeyEvent;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
public class MainActivity extends ActionBarActivity
{
 private ListView obj;
 DBHelper mydb;

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

 mydb = new DBHelper(this);
 ArrayList array_list = mydb.getAllCotacts();

 ArrayAdapter arrayAdapter = new ArrayAdapter(this,
 android.R.layout.simple_list_item_1, array_list);
 //adding it to the list view.
 obj = (ListView)findViewById(R.id.listView1);
 obj.setAdapter(arrayAdapter);
 obj.setOnItemClickListener(new OnItemClickListener()
 {
 @Override
 public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3)
 {
 // TODO Auto-generated method stub
 int id_To_Search = arg2 + 1;
 Bundle dataBundle = new Bundle();
 dataBundle.putInt("id", id_To_Search);
 Intent intent = new Intent(getApplicationContext(),
 com.example.app_14.DisplayContact.class);
 intent.putExtras(dataBundle);
 startActivity(intent);
 }
 });
 }

 @Override
 public boolean onCreateOptionsMenu(Menu menu)
 {
 // Inflate the menu; this adds items to the action bar if it is present.
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }

 @Override
 public boolean onOptionsItemSelected(MenuItem item)
 {
 super.onOptionsItemSelected(item);
 switch(item.getItemId())
 }}
```

```

 {
 case R.id.item1:
 Bundle dataBundle = new Bundle();
 dataBundle.putInt("id", 0);
 Intent intent = new Intent(getApplicationContext(),
 com.example.app_14.DisplayContact.class);
 intent.putExtras(dataBundle);
 startActivity(intent);
 return true;
 default:
 return super.onOptionsItemSelected(item);
 }
}

public boolean onKeyDown(int keycode, KeyEvent event) {
 if (keycode == KeyEvent.KEYCODE_BACK) {
 moveTaskToBack(true);
 }
 return super.onKeyDown(keycode, event);
}
}

```

B8. Chạy ứng dụng để xem kết quả thực hiện

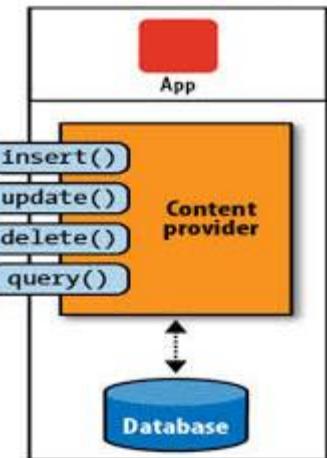
### 3.5. Content Provider

#### 3.5.1. Giới thiệu

CSDL do SQLite tạo ra chỉ dùng trong nội bộ ứng dụng. Để chia sẻ dữ liệu với các ứng dụng khác, bạn cần dùng *content provider*.

Một *content provider* cho phép các ứng dụng chia sẻ một tập dữ liệu ứng dụng chung nào đó. Dữ liệu có thể lưu trữ dữ liệu trên file, trên SQLite database, trên web hoặc trên bất kỳ nơi lưu trữ nào mà ứng dụng có thể truy xuất.

Trong Android có sẵn một số ContentProvider mặc định như: Browser, CallLog, Contacts, MediaStore, Settings...



Hình 3-35 Content Provider với 4 phương thức chính

#### 3.5.2. Truy cập Content Provider

Việc truy cập vào một *content provider* được thực hiện thông qua một URI. Cơ sở cho URI được định nghĩa trong phần khai báo provider trong file *AndroidManifest.xml* qua thuộc tính **Android: authorities**.

#### 3.5.3. Custom content provider

- Để tạo *content provider* cho mình, bạn phải khai báo 1 class mở rộng từ `android.content.ContentProvider` (và đương nhiên phải khai báo trong file manifest Android). Trong đó cần ghi rõ thuộc tính **Android: authorities** cho phép xác định *content provider* để làm cơ sở cho URI để truy cập dữ liệu và phải là duy nhất.

```

<provider
 android:authorities="de.vogella.android.todos.contentprovider"
 android:name=".contentprovider.MyTodoContentProvider" >
</provider>

```

- Trong class Content Provider có 6 phương thức cần được hiện thực. Trong trường hợp bạn không hỗ trợ một số phương thức, bạn cần “ném” (throw) ra 1 ngoại lệ dạng *UnsupportedOperationException()*.

Tên phương thức	Giá trị trả về	Diễn giải
-----------------	----------------	-----------

onCreate	boolean	<ul style="list-style-type: none"> <li>Được gọi khi khởi động provider</li> <li>Nếu khởi tạo thành công trả về true, ngược lại trả về false</li> </ul>
getType	String	<ul style="list-style-type: none"> <li>Trả về MIME của URL được cung cấp</li> </ul>
query	Cursor	<ul style="list-style-type: none"> <li>Truy vấn dữ liệu</li> <li>Kết quả trả về 1 Cursor</li> </ul>
insert	int	<ul style="list-style-type: none"> <li>Thêm dữ liệu</li> <li>Trả về số lượng record được thêm vào</li> </ul>
update	int	<ul style="list-style-type: none"> <li>Cập nhật dữ liệu</li> <li>Trả về số lượng record được cập nhật</li> </ul>
delete	int	<ul style="list-style-type: none"> <li>Xóa dữ liệu</li> <li>Trả về số lượng record bị xóa</li> </ul>

### 3.5.4. UriMatcher

- Class tiện ích hỗ trợ trong việc kết hợp các URI trong Content Provider.
- UriMatcher giúp bạn cách thức xử lý tốt đối với 1 mẫu URI, API cung cấp một số dịch vụ thuận tiện tạo class UriMatcher, với 1 cây mẫu dưới dạng các số nguyên. Bạn có thể sử dụng các số nguyên này trong một câu lệnh mà bạn muốn xử lý với hành động là chuỗi URI được định danh
- Các ký tự đại diện sử dụng trong UriMatcher:
  - \*: đại diện cho một chuỗi ký tự hợp lệ với độ dài bất kỳ.
  - #: đại diện cho một chuỗi ký số với độ dài bất kỳ.

Ví dụ:

- Content://com.example.app.provider/\*: đại diện cho nhiều chuỗi URI trong provider
- Content://com.example.app.provider/tablename/\*: đại diện cho các URI có chứa trong tablename
- Content://com.example.app.provider/tablename/#: đại diện cho các dòng có trong tablename

### 3.5.5. Thread Safety

Nếu bạn làm việc trực tiếp với CSDL và những tiêu trình (threads) do nhiều thành viên trong nhóm lập trình của bạn viết nên có thể chạy đồng thời.

Một content provider có thể được truy cập từ nhiều chương trình cùng một lúc, do đó bạn phải thực hiện truy cập các tiêu trình một cách an toàn. Cách đơn giản nhất là sử dụng các từ khóa đồng bộ ở phía trước của tất cả các phương thức của provider, do đó, chỉ có một thread có thể truy cập các phương thức này cùng một lúc.

Nếu bạn không yêu cầu Android đồng bộ hóa việc truy cập dữ liệu đến provider, bạn cần thiết lập thuộc tính `Android:multiprocess=true` trong phần định nghĩa về provider trong file `AndroidManifest.xml`. Điều này cho phép một thể hiện của provider được tạo ra cho mỗi client.

## BÀI THỰC HÀNH App\_15A

### ☞ Yêu cầu:

Sử dụng 1 Content Provider sẵn có trong Android là People. Tạo 1 project có chức năng liệt kê các tên đã được lưu trong People.

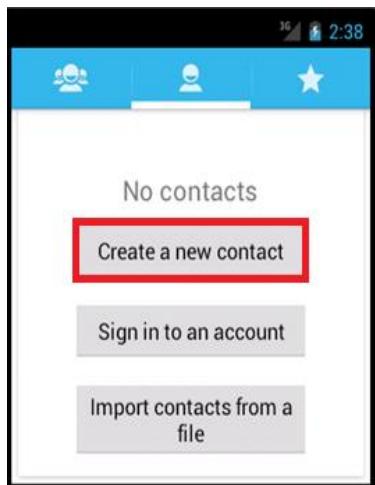
### ☞ Thực hiện:

- B1. Do khi AVD mới tạo lập danh sách các địa chỉ sẵn có trong ứng dụng People là trống. Vì vậy, bạn cần tạo 1 số địa chỉ trước để xem bằng cách nhấn chọn button Home () của AVD, chọn ứng dụng People ()

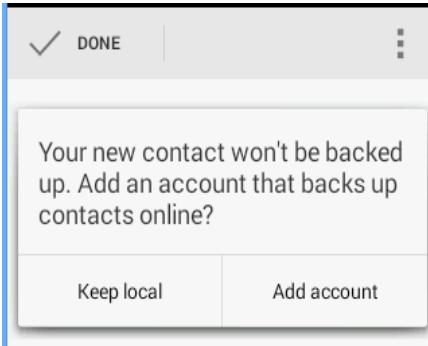


Hình 3-36 Ứng dụng People xuất hiện tùy thuộc giao diện của AVD

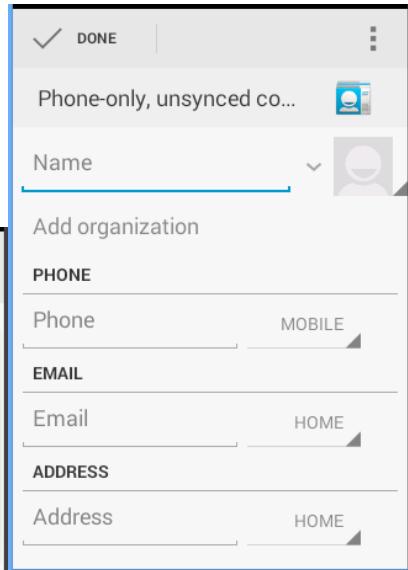
Xuất hiện màn hình sau:



Hình 3-37



Hình 3-38



Hình 3-39

Hình 3-37 Ứng dụng People khi mới sử dụng lần đầu

Hình 3-38 Ứng dụng People khi chọn "Create a new contact"

Hình 3-39 Nhập thông tin về người cần lưu

Do chỉ cần tạo danh sách địa chỉ trên AVD nên sau khi nhấn chọn button "Create a new contact", bạn chọn button *Keep local* trong thông báo vừa xuất hiện.

Nhập thông tin về người cần lưu thông tin (giả sử bạn sẽ nhập thông tin về 3 người có tên lần lượt là Tý, Sửu, Dần).

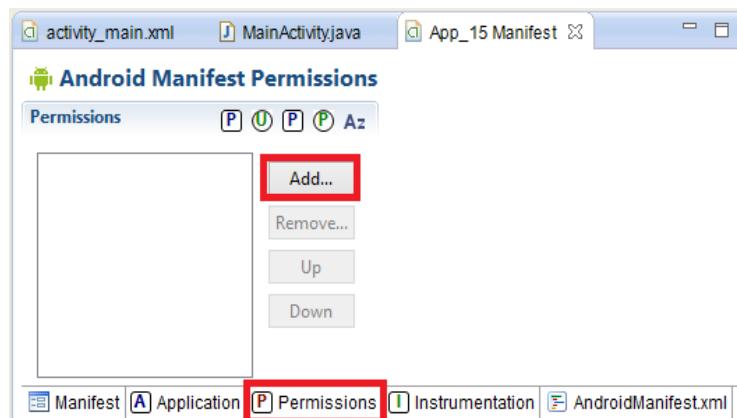
**B2.** Tạo mới project lấy tên là App\_15A.

**B3.** Chính sửa nội dung của file layout để có như sau:

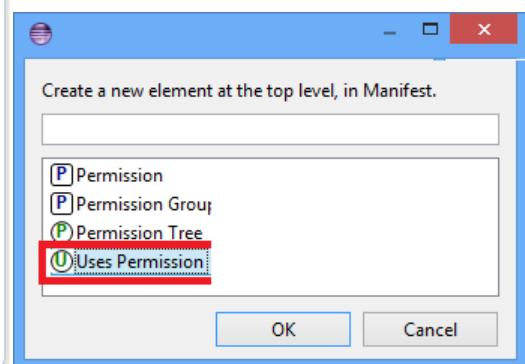
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical" >
 <ListView android:id="@+id/ListView1"
 android:layout_width="match_parent"
 android:layout_height="wrap_content" >
 </ListView>
</LinearLayout>
```

**B4.** Cấp quyền chỉ đọc cho người dùng đối với Contact ContentProvider:

- Mở file `AndroidManifest.xml`, chọn tab *Permissions*.
- Trong tab *Permissions*, click button *Add*. Trong hộp thoại vừa xuất hiện, chọn tiếp *Uses Permission*. Xong click *OK* để trở về tab *Permissions* của file `AndroidManifest.xml`.

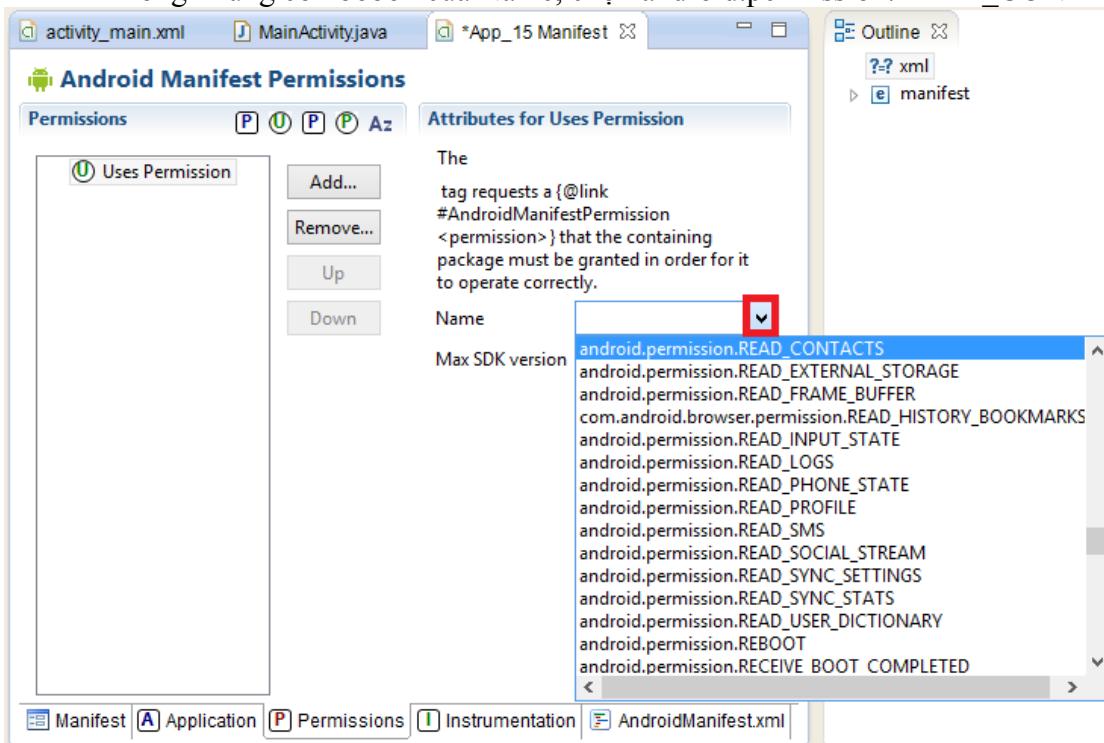


Hình 3-40 Chọn button Add trong tab Permissions

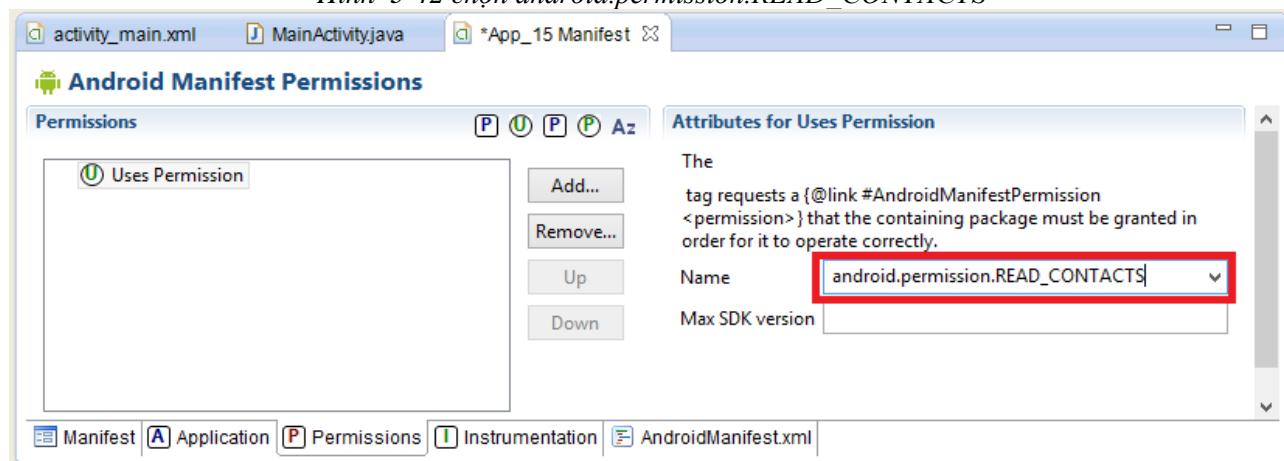


Hình 3-41 Chọn Uses Permission

- Trong khung combobox của Name, chọn android.permission.READ\_CONTACTS.



Hình 3-42 chọn android.permission.READ\_CONTACTS



Hình 3-43 Kết quả sau khi chọn android.permission.READ\_CONTACTS

B5. Chính sửa nội dung file MainActivity.java để có nội dung như sau:

```
package com.example.app_15a;
import android.app.Activity;
```

```
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.database.Cursor;
import android.net.Uri;
import android.provider.ContactsContract;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import java.util.ArrayList;
public class MainActivity extends Activity {
 @Override
 public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 ListView lv = (ListView) findViewById(R.id.listView1);
 ArrayList<String> al=new ArrayList<String>();
 Cursor cursor = getContacts();
 // Đọc tên từng người có trong Contact để đưa vào ArrayList al
 while (cursor.moveToNext())
 {
 String displayName =
 cursor.getString(cursor.getColumnIndex(ContactsContract.Data.DISPLAY_NAME));
 al.add(displayName);
 }
 /*Gán Data source từ al vào ArrayAdapter với android.R.layout.simple_list_item_1
chính là layout Listview được Android xây dựng sẵn*/
 ArrayAdapter<String>adapter=new ArrayAdapter<String>(this,
 android.R.layout.simple_list_item_1, al);
 // Đưa Data source vào ListView
 lv.setAdapter(adapter);
 }
 @SuppressWarnings("deprecation")
 private Cursor getContacts()
 { // Run query
 Uri uri = ContactsContract.Contacts.CONTENT_URI;
 String[] projection = new String[] { ContactsContract.Contacts._ID,
 ContactsContract.Contacts.DISPLAY_NAME };
 String selection = ContactsContract.Contacts.IN_VISIBLE_GROUP + " = '" + ("1") +
 "' ";
 String[] selectionArgs = null;
 String sortOrder = ContactsContract.Contacts.DISPLAY_NAME
 + " COLLATE LOCALIZED ASC";
 return managedQuery(uri, projection, selection, selectionArgs, sortOrder);
 }
 @Override
 public boolean onCreateOptionsMenu(Menu menu)
 { // Inflate the menu; this adds items to the action bar if it is present.
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }
 @Override
 public boolean onOptionsItemSelected(MenuItem item)
 { /* Handle action bar item clicks here. The action bar will automatically handle
 clicks on the Home/Up button, so long as you specify a parent activity in
 AndroidManifest.xml. */

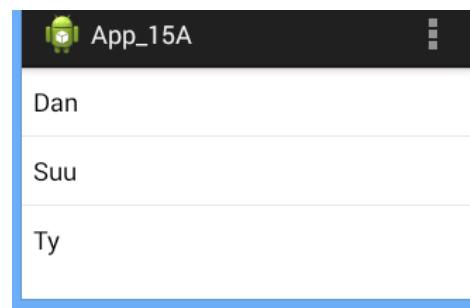
```

```

int id = item.getItemId();
if (id == R.id.action_settings)
{
 return true;
}
return super.onOptionsItemSelected(item);
}

```

- B6. Chạy chương trình để xem kết quả.

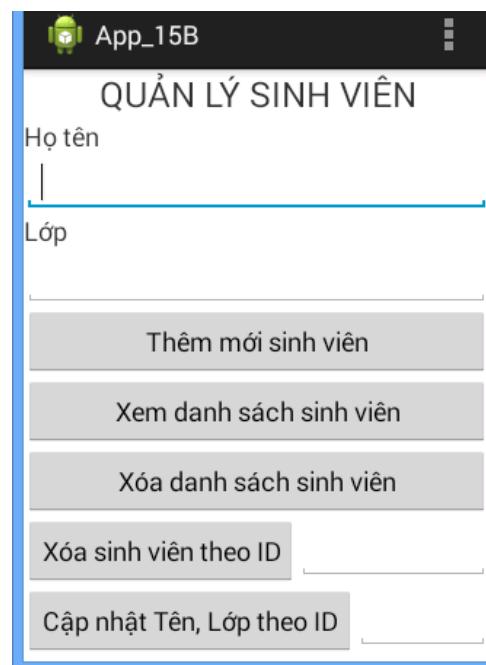


Hình 3-44 Kết quả sau khi chạy ứng dụng

## BÀI THỰC HÀNH App\_15B

### ☞ Yêu cầu:

- Tạo 1 Content Provider quản lý danh sách Students gồm các thông tin ID, Họ tên, Class có giao diện như hình sau:
- Chức năng của từng button:
  - **Thêm mới sinh viên:** ID được cấp tự động, tăng dần; Họ tên và class được lấy từ 2 editText tương ứng. Nếu 2 editText Họ tên hoặc Class để trống sẽ báo lỗi; ngược lại sẽ cho nhập vào table Students.
  - **Xem danh sách sinh viên:** Sử dụng Toast để lần lượt hiển thị thông tin (ID, họ tên và class) của từng sinh viên có trong danh sách. Nếu danh sách trống, xuất hiện thông báo để người dùng biết.
  - **Xóa danh sách sinh viên:** xóa toàn bộ danh sách sinh viên
  - **Xóa sinh viên theo ID:** xóa sinh viên có ID chưa trong editText bên cạnh. Thông báo kết quả (xóa thành công hay không?)
  - **Cập nhật Tên, Class theo ID:** cập nhật sinh viên có ID chưa trong editText bên cạnh. Thông báo kết quả (cập nhật thành công hay không?). Biết rằng Họ tên và Class mới lấy từ 2 editText Họ tên và class ở trên, do đó nếu 2 editText này để trống hoặc editText chưa ID còn bỏ trống sẽ báo lỗi.



Hình 3-45 Giao diện của ứng dụng App\_15B

- Các bước cần thực hiện:
  - (i). Tạo mới project với tên App\_15B
  - (ii). Hiệu chỉnh nội dung file `res/layout/activity_main.xml` để có giao diện như yêu cầu. Trong đó, sử dụng thuộc tính `android:onClick=` “Tên phương thức sẽ được gọi trong class `MainActivity`”
  - (iii). Tạo mới file với tên `src\com\example\StudentsProvider.java` giúp thực hiện các xử lý của Content Provider.
  - (iv). Khai báo Content Provider trong file `AndroidManifest.xml`
  - (v). Hiệu chỉnh file `src\com\example>MainActivity.java` và cài đặt các chức năng thực hiện của từng button.

### ☞ Thực hiện:

- B1. Tạo mới project với tên App\_15B  
 B2. Hiệu chỉnh nội dung file `res/layout/activity_main.xml`

```

<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:gravity="center"
 android:textSize="24sp"
 android:text="QUẢN LÝ SINH VIÊN" />
 <TextView android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:textSize="18sp"
 android:text="Họ tên" />
 <EditText android:id="@+id/txtName"
 android:layout_height="wrap_content"
 android:layout_width="fill_parent"
 android:inputType="text"/>
 <TextView android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:textSize="18sp"
 android:text="Class" />
 <EditText android:id="@+id/txtGrade"
 android:layout_height="wrap_content"
 android:layout_width="fill_parent"
 android:inputType="text"/>
 <Button android:text="Thêm mới sinh viên"
 android:id="@+id/btnThem"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:onClick="onClickThem" />
 <Button android:text="Xem danh sách sinh viên"
 android:id="@+id/btnXem"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:onClick="onClickXem" />
 <Button android:text="Xóa danh sách sinh viên"
 android:id="@+id/btnXoaAll"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:onClick="onClickXoaAll" />
 <LinearLayout android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:orientation="horizontal" >
 <Button android:text="Xóa sinh viên theo ID"
 android:id="@+id/btnXoaID"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:onClick="onClickXoaID" />
 <EditText android:id="@+id/txtDeleteID"
 android:layout_height="wrap_content"
 android:layout_width="fill_parent"
 android:inputType="text"/>
 </LinearLayout>
 <LinearLayout android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:orientation="horizontal" >
 <Button android:text="Cập nhật Tên, Class theo ID"
 android:id="@+id/btnCapNhatID"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:onClick="onClickCapNhatID" />
 <EditText android:id="@+id/txtCapNhatID"
 android:layout_height="wrap_content"
 android:layout_width="fill_parent"
 android:inputType="text"/>
```

```
</LinearLayout>
</LinearLayout>
```

- B3. Tạo mới file với tên `src\com\example\StudentsProvider.java` giúp thực hiện các xử lý của Content Provider.

```
package com.example.app_15b;
import java.util.HashMap;
import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.Context;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteQueryBuilder;
import android.net.Uri;
import android.text.TextUtils;

public class StudentsProvider extends ContentProvider
{
 /** Khai báo các hằng số về Database*/
 private SQLiteDatabase db;
 static final String DATABASE_NAME = "QLSV";
 static final String STUDENTS_TABLE_NAME = "students";
 static final int DATABASE_VERSION = 1;
 static final String CREATE_DB_TABLE = " CREATE TABLE " + STUDENTS_TABLE_NAME +
 " (_id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT NOT NULL, " +
 " grade TEXT NOT NULL);";
 /** Khai báo các hằng số về Content Provider sẽ sử dụng */
 static final String PROVIDER_NAME = "com.example.app_15b"; //authority
 static final String URL = "content://" + PROVIDER_NAME + "/" + STUDENTS_TABLE_NAME;
 static final Uri CONTENT_URI = Uri.parse(URL);
 /** Table students sẽ gồm 3 thuộc tính */
 static final String _ID = "_id";
 static final String NAME = "name";
 static final String GRADE = "grade";

 private static HashMap<String, String> STUDENTS_PROJECTION_MAP;
 // Định nghĩa các chỉ số sẽ sử dụng cho UriMatcher
 static final int STUDENTS = 1;
 static final int STUDENT_ID = 2;
 static final UriMatcher uriMatcher;
 static { uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
 uriMatcher.addURI(PROVIDER_NAME, "students", STUDENTS);
 uriMatcher.addURI(PROVIDER_NAME, "students/#", STUDENT_ID);
 }
 /**== Helper class that actually creates and manages the provider's underlying data
repository ==*/
 private static class DatabaseHelper extends SQLiteOpenHelper
 {
 DatabaseHelper(Context context)
 {
 super(context, DATABASE_NAME, null, DATABASE_VERSION);
 }
 @Override
 public void onCreate(SQLiteDatabase db)
 {
 db.execSQL(CREATE_DB_TABLE);
 }
 @Override
```

```

 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
 {
 db.execSQL("DROP TABLE IF EXISTS " + STUDENTS_TABLE_NAME);
 onCreate(db);
 }
}
/** ===== End Helper class ===== */
@Override
public boolean onCreate()
{
 Context context = getContext();
 DatabaseHelper dbHelper = new DatabaseHelper(context);
 db = dbHelper.getWritableDatabase();
 /** Nếu tạo CSDL KHÔNG thành công sẽ return false, thành công return true */
 return (db == null)? false:true;
}

@Override
public Uri insert(Uri uri, ContentValues values)
{
 /** Thêm mới 1 sinh viên */
 long rowID = db.insert(STUDENTS_TABLE_NAME, "", values);
 /** Nếu thêm thành công */
 if (rowID > 0)
 {
 Uri _uri = ContentUris.withAppendedId(CONTENT_URI, rowID);
 getContext().getContentResolver().notifyChange(_uri, null);
 return _uri;
 }
 throw new SQLException("Thêm SV KHÔNG thành công khi thêm vào " + uri);
}

@Override
public Cursor query(Uri uri, String[] projection, String selection,
 String[] selectionArgs, String sortOrder)
{
 SQLiteQueryBuilder qb = new SQLiteQueryBuilder();
 qb.setTables(STUDENTS_TABLE_NAME);
 switch (uriMatcher.match(uri))
 {
 case STUDENTS:
 qb.setProjectionMap(STUDENTS_PROJECTION_MAP);
 break;
 case STUDENT_ID:
 qb.appendWhere(_ID + "=" + uri.getPathSegments().get(1));
 break;
 default:
 throw new IllegalArgumentException("Unknown URI " + uri);
 }
 if (sortOrder == null || sortOrder == "")
 { /** Mặc định sẽ sort danh sách theo họ tên*/
 sortOrder = NAME;
 }
 Cursor c = qb.query(db, projection, selection, selectionArgs, null, null,
 sortOrder);
 /** register to watch a content URI for changes */
 c.setNotificationUri(getContext().getContentResolver(), uri);
 return c;
}

@Override
public int delete(Uri uri, String selection, String[] selectionArgs)
{
}

```

```

int count = 0;
switch (uriMatcher.match(uri))
{
 case STUDENTS:
 count = db.delete(STUDENTS_TABLE_NAME, selection, selectionArgs);
 break;
 case STUDENT_ID:
 String id = uri.getPathSegments().get(1);
 count = db.delete(STUDENTS_TABLE_NAME, _ID + " = " + id +
 (!TextUtils.isEmpty(selection) ? " AND (" + selection + ')'
 : ""), selectionArgs);
 break;
 default:
 throw new IllegalArgumentException("Unknown URI " + uri);
}
getContext().getContentResolver().notifyChange(uri, null);
return count;
}

@Override
public int update(Uri uri, ContentValues values, String selection,
 String[] selectionArgs)
{
 int count = 0;
 switch (uriMatcher.match(uri))
 {
 case STUDENTS:
 count = db.update(STUDENTS_TABLE_NAME, values, selection,
 selectionArgs);
 break;
 case STUDENT_ID:
 count = db.update(STUDENTS_TABLE_NAME, values, _ID + " = " +
 uri.getPathSegments().get(1) +
 (!TextUtils.isEmpty(selection) ? " AND (" + selection + ')'
 : ""), selectionArgs);
 break;
 default:
 throw new IllegalArgumentException("KHÔNG tìm thấy ID " + STUDENT_ID);
 }
 getContext().getContentResolver().notifyChange(uri, null);
 return count;
}

@Override
public String getType(Uri uri)
{
 switch (uriMatcher.match(uri))
 { /** Get all student records*/
 case STUDENTS:
 return "vnd.android.cursor.dir/vnd.example.students";
 /** Get a particular student */
 case STUDENT_ID:
 return "vnd.android.cursor.item/vnd.example.students";
 default:
 throw new IllegalArgumentException("Unsupported URI: " + uri);
 }
}

```

**B4.** Khai báo Content Provider trong file *AndroidManifest.xml*

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.app_15b"
 android:versionCode="1"

```

```

 android:versionName="1.0" >
 <uses-sdk
 android:minSdkVersion="12"
 android:targetSdkVersion="21" />
 <application
 android:allowBackup="true"
 android:icon="@drawable/ic_launcher"
 android:label="@string/app_name"
 android:theme="@style/AppTheme" >
 <activity
 android:name=".MainActivity"
 android:label="@string/app_name" >
 <intent-filter>
 <action android:name="android.intent.action.MAIN" />
 <category android:name="android.intent.category.LAUNCHER" />
 </intent-filter>
 </activity>
 <provider android:name=".StudentsProvider"
 android:authorities="com.example.app_15b">
 </provider>
 </application>
</manifest>

```

- B5. Hiệu chỉnh file `src\com\example\MainActivity.java` và cài đặt các chức năng thực hiện của từng button.

```

package com.example.app_15b;
import android.net.Uri;
import android.os.Bundle;
import android.app.Activity;
import android.content.ContentValues;
import android.database.Cursor;
import android.view.Menu;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends Activity
{
 EditText etName, etGrade, etDeleteID, etUpdateID;
 @Override
 protected void onCreate(Bundle savedInstanceState)
 {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 etName=(EditText)findViewById(R.id.txtName);
 etGrade=(EditText)findViewById(R.id.txtGrade);
 etDeleteID=(EditText)findViewById(R.id.txtDeleteID);
 etUpdateID=(EditText)findViewById(R.id.txtCapNhatID);
 }
 @Override
 public boolean onCreateOptionsMenu(Menu menu)
 {
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }
 public void onClickThem(View view)
 {
 if ((etName.getText().toString().length()!=0) &&
 (etGrade.getText().toString().length()!=0))
 {
 // Add a new student record
 ContentValues values = new ContentValues();
 values.put(StudentsProvider.NAME, etName.getText().toString());
 values.put(StudentsProvider.GRADE, etGrade.getText().toString());
 Uri uri = getContentResolver().insert(StudentsProvider.CONTENT_URI, values);
 }
 }
}

```

```

 Toast.makeText(getApplicationContext(), uri.toString(), Toast.LENGTH_LONG).show();

 etName.setText("");
 etGrade.setText("");
 }
 else
 Toast.makeText(getApplicationContext(), "Chưa nhập đủ thông tin của SV",
 Toast.LENGTH_LONG).show();
}
@SuppressLint("deprecation")
public void onClickXem(View view)
{
// Retrieve student records
String URL = "content://com.example.app_15b/students";
Uri students = Uri.parse(URL);
Cursor c = managedQuery(students, null, null, null, "name");
if (c.moveToFirst())
{
 do
 {
 String sID = c.getString(c.getColumnIndex(StudentsProvider._ID));
 String sName = c.getString(c.getColumnIndex(StudentsProvider.NAME));
 String sGrade = c.getString(c.getColumnIndex(StudentsProvider.GRADE));
 Toast.makeText(this, "Mã số: " + sID + "; Tên: " + sName + "; Class: " +
sGrade,
 Toast.LENGTH_SHORT).show();
 } while (c.moveToNext());
}
else
 Toast.makeText(this, "KHÔNG có SV nào trong danh sách",
 Toast.LENGTH_SHORT).show();
}
public void onClickXoaAll (View view)
{
// delete all the records and the table of the database provider
String URL = "content://com.example.app_15b/students";
Uri uriSV = Uri.parse(URL);
int count = getContentResolver().delete(uriSV, null, null);
String msg = "Đã xóa tất cả các record trong table STUDENTS (gồm "+ count +" records)";
Toast.makeText(getApplicationContext(), msg, Toast.LENGTH_LONG).show();
}
public void onClickXoaID (View view)
{
// delete records with ID
String URL = "content://com.example.app_15b/students/" +
etDeleteID.getText().toString();
Uri uriSV = Uri.parse(URL);
int count = getContentResolver().delete(uriSV, null, null);
String msg ;
if (count>0)
 msg = "Đã xóa SV có ID = " + etDeleteID.getText().toString();
else
 msg = "KHÔNG tìm thấy SV có ID = " + etDeleteID.getText().toString();
Toast.makeText(getApplicationContext(), msg, Toast.LENGTH_LONG).show();
}
public void onClickCapNhatID (View view)
{
//nếu muốn cập nhật thì phải bổ sung đầy đủ thông tin về tên, class, ID cần sửa
if ((etName.getText().toString().length()!=0)
&& (etGrade.getText().toString().length()!=0)
&& (etUpdateID.getText().toString().length()!=0))
}

```

```

 {
 ContentValues values = new ContentValues();
 values.put(StudentsProvider._ID, etUpdateID.getText().toString());
 values.put(StudentsProvider.NAME, etName.getText().toString());
 values.put(StudentsProvider.GRADE, etGrade.getText().toString());
 String URL = "content://com.example.app_15b/students/"
 + etUpdateID.getText().toString();

 Uri uriSV = Uri.parse(URL);
 int count = getContentResolver().update(uriSV, values, null, null);
 String msg="";
 if(count>0)
 {
 msg = "Đã cập nhật SV có ID = " + etUpdateID.getText().toString();
 etName.setText("");
 etGrade.setText("");
 }
 else
 msg = "KHÔNG tìm thấy SV có ID = " + etUpdateID.getText().toString();
 Toast.makeText(getApplicationContext(),msg,Toast.LENGTH_LONG).show();
 }
 else
 Toast.makeText(getApplicationContext(),"Chưa nhập đủ thông tin của SV",
 Toast.LENGTH_LONG).show();
}
}

```

B6. Chạy ứng dụng và kiểm tra các yêu cầu đã đưa ra ban đầu.

## 3.6. LƯU TRỮ DỮ LIỆU DƯỚI DẠNG FILE

Cách đơn giản nhất để lưu trữ dữ liệu của ứng dụng là ghi xuống file. Tuy nhiên cách này hạn chế rất nhiều do các phương thức đọc, ghi dữ liệu rất thô sơ, đơn giản. Phần dưới đây thể hiện cách đọc ghi file đơn giản.

### 3.6.1. Sử dụng bộ nhớ trong

Bạn có thể lưu file trực tiếp bộ nhớ trong của ứng dụng. Mặc định thì file này sẽ thuộc về ứng dụng tạo ra nó và các ứng dụng khác không thể truy xuất đến nó.

#### 3.6.1.1. Ghi dữ liệu xuống file

Để ghi dữ liệu xuống file ta thực hiện các bước:

B1. Gọi phương thức *FileOutputStream openFileOutput (String name, int mode)* để tạo một đối tượng *FileOutputStream* dùng để ghi dữ liệu lên file.

Phương thức này nhận vào hai tham số:

- String name: tên file.
- int mode: chế độ ghi. Có ba chế độ:
  - *MODE\_PRIVATE*: file chỉ có thể được truy xuất bên trong ứng dụng tạo ra nó.
  - *MODE\_WORLD\_READABLE*: file có thể được đọc bởi các ứng dụng khác.
  - *MODE\_WORLD\_WRITEABLE*: file có thể được đọc và ghi bởi các ứng dụng khác.

B2. Gọi phương thức *FileOutputStream.write(...)* để ghi dữ liệu xuống file.

B3. Gọi phương thức *FileOutputStream.close()* để đóng luồng ghi dữ liệu xuống file.

VD minh họa:

```

public void CreateFileOnInternal()
{
 FileOutputStream fos=null;
 String eol = System.getProperty("line.separator");
 try
 {
 String FILENAME = "Test";

```

```

 String string = etContentCreate.getText().toString() + eol;
 fos = openFileOutput(FILENAME, Context.MODE_PRIVATE);
 fos.write(string.getBytes());
 }
 catch (Exception e)
 {
 e.printStackTrace();
 }
 finally
 {
 if (fos != null)
 {
 try
 {
 fos.close();
 Toast.makeText(this, "Đã lưu file thành công", Toast.LENGTH_LONG).show();
 }
 catch (IOException e)
 {
 e.printStackTrace();
 Toast.makeText(this, "Lưu file KHÔNG thành công", Toast.LENGTH_LONG).show();
 }
 }
 }
}

```

### 3.6.1.2. Đọc dữ liệu từ file

Để đọc dữ liệu từ file ta thực hiện các bước:

- B1. Gọi phương thức *public abstract FileInputStream openFileInput(String name)* tạo luồng đọc dữ liệu từ file. Phương thức này nhận vào một tham số là tên file cần đọc.
- B2. Gọi phương thức *InputStream.read(...)* để đọc dữ liệu từ file.
- B3. Gọi phương thức *InputStream.close()* để đóng luồng đọc dữ liệu từ file.

VD minh họa:

```

private String ReadFileFromInternal()
{
 String FILENAME == "Test";
 String CONTENT = "";
 FileInputStream fis=null;
 try
 {
 fis = openFileInput(FILENAME);
 byte[] buffer = new byte[1024];
 int count = fis.read(buffer);
 if (count>0)
 CONTENT = new String(buffer);
 else
 CONTENT = "File không có nội dung";
 }
 catch (IOException e)
 {
 e.printStackTrace();
 CONTENT = e.getMessage();
 }
 finally
 {
 try
 {
 fis.close();
 }
 catch (IOException e)
 {
 e.printStackTrace();
 CONTENT = e.getMessage();
 }
 }
 return CONTENT;
}

```

### 3.6.2. Sử dụng cache file

Đôi lúc bạn cũng muốn lưu trữ file vào thư mục cache thay vì lưu trữ vĩnh viễn. Tuy nhiên nếu lưu trữ file trong thư mục cache thì khi thiết bị thiếu bộ nhớ trong thì hệ thống sẽ tự xóa các file này đi.

Thực hiện lưu trữ file cache cũng tương tự như thực hiện đối với file ở bộ nhớ trong, chỉ khác là ta sẽ sử dụng phương thức `getCacheDir()` để lấy về thư mục cache lưu trữ dữ liệu của ứng dụng. Thư mục này thường là: `data/data/[packageName]/cache`.

Ví dụ:

```
private String ReadFileFromCache()
{
 String FILENAME = "Test";
 String CONTENT = "";
 // biến f là đối tượng trả đến thư mục cache của ứng dụng.
 File f = getCacheDir();
 // lấy về chuỗi đường dẫn tuyệt đối trả đến thư mục cache
 String cacheDir = f.getPath();

 // tạo đối tượng File trả đến file myMedia.dat trong thư mục của cache
 File cacheFile = new File(cacheDir,FILENAME);
 FileInputStream fos = null;
 try
 {
 // đọc file ra biến str
 fos = new FileInputStream(cacheFile);
 byte[] buffer = new byte[1024];
 int count = fos.read(buffer);
 if (count>0)
 CONTENT = new String(buffer);
 else
 CONTENT="File không có nội dung";
 }
 catch (Exception e)
 {
 e.printStackTrace();
 CONTENT="ERROR";
 }
 finally
 {
 try
 {
 fos.close();
 }
 catch (Exception e)
 {
 e.printStackTrace();
 CONTENT="ERROR";
 }
 }
 //CONTENT trả về 1 trong 3 trường hợp: "", "ERROR" hoặc kết quả đọc được từ file
 return CONTENT;
}
```

### 3.6.3. Sử dụng bộ nhớ ngoài

- Ngoài việc lưu trữ file xuống bộ nhớ trong của thiết bị ta còn có thể lưu trữ xuống bộ nhớ ngoài (thẻ SD).
- Bộ nhớ ngoài này là thiết bị lưu trữ có thể tháo rời nên ta cần thực hiện 2 việc sau:
  - Cấp quyền cho ứng dụng được ghi trên thẻ SD trong file `AndroidManifest.xml` như sau:
 

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

Lưu ý: tag `<uses-permission ...>` phải được đặt ngang cấp với các tag `<uses-sdk .../>` và `<application ...> ...</application>`

- Tiến hành kiểm tra trạng thái của bộ nhớ ngoài (đang cắm vào máy tính, hay đã tháo ra, ...) trước khi đọc và ghi dữ liệu lên thẻ như sau:

```

import android.os.Environment;
boolean mExternalStorageAvailable = false;
boolean mExternalStorageWriteable = false;

private void TestExternalStorageAvailable()
{
 String state = Environment.getExternalStorageState();
 if (Environment.MEDIA_MOUNTED.equals(state))
 { // Có thẻ đọc và ghi dữ liệu
 mExternalStorageAvailable = mExternalStorageWriteable = true;
 }
 else
 if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(state))
 { // Chỉ có thể đọc
 mExternalStorageAvailable = true;
 mExternalStorageWriteable = false;
 }
 else
 { // KHÔNG thẻ đọc và ghi dữ liệu
 mExternalStorageAvailable = mExternalStorageWriteable = false;
 }
}

```

- Đọc dữ liệu từ bộ nhớ ngoài:

```

import android.util.Log;
private void readFileFromSDCard()
{
 File directory = Environment.getExternalStorageDirectory();
 // assumes that a file article.rss is available on the SD card
 File file = new File(directory + "/article.rss");
 if (!file.exists())
 { throw new RuntimeException("File not found");
 }
 Log.e("Testing", "Starting to read");
 BufferedReader reader = null;
 try
 { reader = new BufferedReader(new FileReader(file));
 StringBuilder builder = new StringBuilder();
 String line;
 while ((line = reader.readLine()) != null)
 { builder.append(line);
 }
 }
 catch (Exception e)
 { e.printStackTrace();
 }
 finally
 { if (reader != null)
 {
 try
 { reader.close();
 }
 catch (IOException e)
 { e.printStackTrace();
 }
 }
 }
}

```

- Nếu sử dụng từ API lever 8 trở lên thì bạn có thể gọi phương thức *public abstract File getExternalFilesDir (String type)* để lấy về đối tượng trả đến thư mục lưu trữ trên bộ nhớ ngoài (Thẻ nhớ SD) của thiết bị. Thông số *String type* là kiểu thư mục muốn truy xuất:
  - *DIRECTORY\_ALARMS*: Thư mục chứa các file âm thanh dùng làm báo thức.
  - *DIRECTORY\_DCIM*: Thư mục chứa các file hình ảnh và video được ghi lại bởi thiết bị.
  - *DIRECTORY\_DOWNLOADS*: Thư mục chứa các file được tải về.
  - *DIRECTORY\_MOVIES*: Thư mục chứa các file phim.
  - *DIRECTORY\_MUSIC*: Thư mục chứa các file âm nhạc.
  - *DIRECTORY\_NOTIFICATIONS*: Thư mục chứa các file âm thanh dùng làm âm báo notification.
  - *DIRECTORY\_PICTURES*: Thư mục chứa các file hình ảnh.
  - *DIRECTORY\_PODCASTS*: Thư mục chứa các file podcast.
  - *DIRECTORY\_RINGTONES*: Thư mục chứa các file dùng làm nhạc chuông.
- Nếu sử dụng từ API 7 trở xuống, bạn có thể sử dụng phương thức: *public static File getExternalStorageDirectory()* để tạo đối tượng File trả đến thư mục gốc của bộ nhớ ngoài. Ta sẽ cộng thêm vào đường dẫn này để chỉ đến thư mục cần lưu trữ dữ liệu:
  - *Alarms*: Thư mục chứa các file âm thanh dùng làm báo thức.
  - *Download*: Thư mục chứa các file được tải về.
  - *Movies*: Thư mục chứa các file phim (Bao gồm cả các video quay bằng camera).
  - *Music*: Thư mục chứa các file âm nhạc.
  - *Notifications*: Thư mục chứa các file âm thanh dùng làm âm báo notification.
  - *Pictures*: Thư mục chứa các file hình ảnh (Bao gồm cả hình ảnh chụp từ camera).
  - *Podcasts*: Thư mục chứa các file podcast.
  - *Ringtones*: Thư mục chứa các file dùng làm nhạc chuông.

Sau khi có được đối tượng File ta có thể tiếp tục làm việc như ở các ví dụ trên.

#### 3.6.4. Một số phương thức hữu dụng đối với file:

- **public abstract File getFilesDir ()**: Lấy đường dẫn tuyệt đối đến thư mục hệ thống nơi file được lưu trữ.
- **public abstract File getDir (String name, int mode)**: Tạo mới (hoặc mở nếu nó đã tồn tại) một thư mục bên trong bộ nhớ trong của ứng dụng.
- **public abstract boolean deleteFile (String name)**: Xóa một file trong bộ nhớ trong.
- **public abstract String[] fileList ()**: Trả về danh sách các Tập tên đã được ghi xuống bộ nhớ trong của ứng dụng.

## BÀI THỰC HÀNH App\_15C

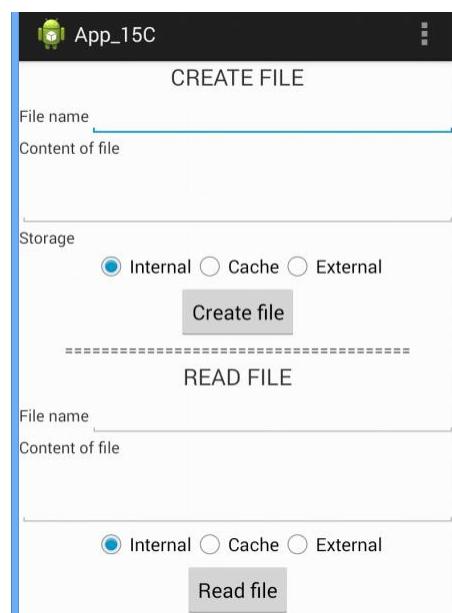
### Yêu cầu:

- Tạo mới 1 project có giao diện như hình sau. Có thể tạm chia giao diện gồm 2 phần: ghi và đọc file.
- Khi tạo file: dùng Toast để thông báo kết quả thực hiện.
- Khi đọc file: kết quả đọc (được hay không hoặc có lỗi) thì kết quả đều xuất hiện trong editText “Content of file”.
- Các button thực hiện đúng tính năng ghi trên button. Khi đó cần kiểm tra “File name” không được để trống.

**Thực hiện:**

- B1. Tạo mới project. Chính sửa file layout *activity\_main.xml* để có nội dung như sau, trong đó cả 2 button đều dùng thuộc tính *android:onClick* để gọi phương thức trong *MainActivity.java* thực hiện:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:id="@+id/LinearLayout1"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 tools:context="com.example.app_15c.MainActivity" >
```



Hình 3-46 Giao diện của ứng dụng App\_15C

```
<TextView android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_gravity="center"
 android:textSize="20sp"
 android:text="CREATE FILE" />
<LinearLayout android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:orientation="horizontal" >
 <TextView android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="File name" />
 <EditText android:id="@+id/etFileNameCreate"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:ems="10"
 android:inputType="text" >
 </EditText>
 </LinearLayout>
 <TextView android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Content of file" />
 <EditText android:id="@+id/etContentCreate"
 android:layout_width="match_parent"
 android:layout_height="60dp"
 android:ems="10"
 android:inputType="text"/>
 <TextView android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Storage" />
 <RadioGroup android:id="@+id/radioGroup1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:orientation="horizontal"
 android:layout_gravity="center">
 <RadioButton android:id="@+id/radCreateInternal"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:checked="true"
 android:text="Internal" />
 <RadioButton android:id="@+id/radCreateCache"
 android:layout_width="wrap_content"
```

```
 android:layout_height="wrap_content"
 android:text="Cache" />
 <RadioButton android:id="@+id/radCreateExternal"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_gravity="center"
 android:text="External" />
</RadioGroup>
<Button android:id="@+id/btnCreate"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:onClick="onClickCreateFile"
 android:layout_gravity="center"
 android:text="Create file" />
<TextView android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_gravity="center"
 android:text="===== " />
<TextView android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_gravity="center"
 android:textSize="20sp"
 android:text="READ FILE" />
<LinearLayout android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:orientation="horizontal" >
 <TextView android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="File name" />
 <EditText android:id="@+id/etFileNameRead"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:ems="10"
 android:inputType="text" >
 </EditText>
</LinearLayout>
<TextView android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Content of file" />
<EditText android:id="@+id/etContentRead"
 android:layout_width="match_parent"
 android:layout_height="60dp"
 android:ems="10"
 android:inputType="text"/>
<RadioGroup android:id="@+id/radioGroup2"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:orientation="horizontal"
 android:layout_gravity="center">
 <RadioButton android:id="@+id/radReadInternal"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:checked="true"
 android:text="Internal" />
 <RadioButton android:id="@+id/radReadCache"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Cache" />
 <RadioButton android:id="@+id/radReadExternal"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="External" />
</RadioGroup>
```

```

<Button android:id="@+id/btnRead"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_gravity="center"
 android:text="Read file"
 android:onClick="onClickReadFile"/>
</LinearLayout>

```

B2. Chính sửa nội dung file MainActivity.java để có nội dung như sau:

```

package com.example.app_15c;
import android.app.Activity;
import android.view.Menu;
import android.view.MenuItem;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.Toast;
import android.os.Environment;
import android.util.Log;

public class MainActivity extends Activity
{
 //Khai báo biến quản lý việc Create File
 EditText etFileNameCreate, etContentCreate;
 Button btnCreate;
 RadioButton radCreateInternal, radCreateCache, radCreateExternal;

 //Khai báo biến quản lý việc Read File
 EditText etFileNameRead, etContentRead;
 Button btnRead;
 RadioButton radReadInternal, radReadCache, radReadExternal;

 // biến giúp kiểm tra khả năng đọc/ghi của bộ nhớ ngoài
 boolean mExternalStorageAvailable = false;
 boolean mExternalStorageWriteable = false;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 etFileNameCreate=(EditText)findViewById(R.id.etFileNameCreate);
 etContentCreate=(EditText)findViewById(R.id.etContentCreate);
 btnCreate=(Button)findViewById(R.id.btnCreate);
 radCreateInternal=(RadioButton) findViewById(R.id.radCreateInternal);
 radCreateCache=(RadioButton) findViewById(R.id.radCreateCache);
 radCreateExternal=(RadioButton) findViewById(R.id.radCreateExternal);

 etFileNameRead=(EditText)findViewById(R.id.etFileNameRead);
 etContentRead=(EditText)findViewById(R.id.etContentRead);
 btnRead=(Button)findViewById(R.id.btnRead);
 }
}

```

```
 radReadInternal=(RadioButton) findViewById(R.id.radReadInternal);
 radReadCache=(RadioButton) findViewById(R.id.radReadCache);
 radReadExternal=(RadioButton) findViewById(R.id.radReadExternal);
}
private void TestExternalStorageAvailable()
{
 String state = Environment.getExternalStorageState();
 if (Environment.MEDIA_MOUNTED.equals(state))
 { // Có thể đọc và ghi dữ liệu
 mExternalStorageAvailable = mExternalStorageWriteable = true;
 }
 else
 if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(state))
 { // Chỉ có thể đọc
 mExternalStorageAvailable = true;
 mExternalStorageWriteable = false;
 }
 else
 { // KHÔNG thể đọc và ghi dữ liệu
 mExternalStorageAvailable = mExternalStorageWriteable = false;
 }
}
//===== CÁC PHƯƠNG THỨC GHI FILE =====
public void onClickCreateFile(View v)
{
 if(etFileCreate.getText().length()==0)
 Toast.makeText(this, "Chưa nhập tên file", Toast.LENGTH_LONG).show();
 else
 if(etContentCreate.getText().length()==0)
 Toast.makeText(this, "Chưa nhập nội dung file", Toast.LENGTH_LONG).show();
 else
 if (radCreateInternal.isChecked())
 CreateFileOnInternal();
 else
 if (radCreateCache.isChecked())
 CreateFileOnCache();
 else
 {
 TestExternalStorageAvailable();
 if (mExternalStorageWriteable)
 CreateFileOnExternal();
 }
}
public void CreateFileOnInternal()
{
 FileOutputStream fos=null;
 String eol = System.getProperty("line.separator");
 try
 {
 String FILENAME = etFileCreate.getText().toString();
 String CONTENT = etContentCreate.getText().toString()+eol;
 fos = openFileOutput(FILENAME, Context.MODE_PRIVATE);
 fos.write(CONTENT.getBytes());
 }
 catch (Exception e)
 { e.printStackTrace();
 }
 finally
 {
 if (fos != null)
 {
 try
```

```
 { fos.close();
 Toast.makeText(this, "Đã lưu file thành công", Toast.LENGTH_LONG).show();
 }
 catch (IOException e)
 { e.printStackTrace();
 Toast.makeText(this, "Lưu file KHÔNG thành công",
 Toast.LENGTH_LONG).show();
 }
 }
 }

public void CreateFileOnCache()
{
 FileOutputStream fos=null;
 String eol = System.getProperty("line.separator");
 try
 {
 String FILENAME = etFileCreate.getText().toString();
 String CONTENT = etContentCreate.getText().toString()+eol;
 // biến f là đối tượng trả đến thư mục cache của ứng dụng.
 File cacheDir = getCacheDir();
 // tạo đối tượng File trả đến file trong thư mục của cache
 File cacheFile = new File(cacheDir,FILENAME);
 // ghi CONTENT vào file
 fos = new FileOutputStream(cacheFile);
 fos.write(CONTENT.getBytes());
 }
 catch (Exception e)
 { e.printStackTrace();
 }
 finally
 {
 if (fos != null)
 {
 try
 { fos.close();
 Toast.makeText(this, "Đã lưu file thành công", Toast.LENGTH_LONG).show();
 }
 catch (IOException e)
 { e.printStackTrace();
 Toast.makeText(this, "Lưu file KHÔNG thành công",
 Toast.LENGTH_LONG).show();
 }
 }
 }
}

public void CreateFileOnExternal()
{
 FileOutputStream fos=null;
 OutputStreamWriter OutStreWriter=null;
 String FILENAME = etFileCreate.getText().toString();
 String CONTENT = etContentCreate.getText().toString();
 String DIRECTORY = Environment.getExternalStorageDirectory().getAbsolutePath();
 File myFile = new File(DIRECTORY + File.separator + FILENAME);
 try
 {
 myFile.createNewFile();
 fos = new FileOutputStream(myFile);
 OutStreWriter = new OutputStreamWriter(fos);
 OutStreWriter.append(CONTENT);
 }
}
```

```

 catch (Exception e)
 {
 e.printStackTrace();
 Toast.makeText(this, "Lỗi ghi file External", Toast.LENGTH_LONG).show();
 }
 finally
 {
 if (writer != null)
 {
 try
 { writer.close();
 Toast.makeText(this, "Đã lưu file External thành công",
 Toast.LENGTH_LONG).show();
 }
 catch (IOException e)
 { e.printStackTrace();
 Toast.makeText(this, "Lưu file External KHÔNG thành công",
 Toast.LENGTH_LONG).show();
 }
 }
 }
}
//===== CÁC PHƯƠNG THỨC ĐỌC FILE =====
public void onClickReadFile(View v)
{
 if(etFileRead.getText().length()==0)
 Toast.makeText(this, "Chưa nhập tên file", Toast.LENGTH_LONG).show();
 else
 if (radReadInternal.isChecked())
 etContentRead.setText(ReadFileFromInternal());
 else
 if (radReadCache.isChecked())
 etContentRead.setText(ReadFileFromCache());
 else
 {
 TestExternalStorageAvailable();
 if (mExternalStorageAvailable)
 { StringBuilder sb=new StringBuilder();
 String CONTENT="";
 sb=ReadFileFromExternal();
 CONTENT=sb.substring(0);
 etContentRead.setText(CONTENT);
 }
 else
 etContentRead.setText("KHÔNG thể đọc từ thiết bị ngoài");
 }
 }
private String ReadFileFromInternal()
{
 String FILENAME = etFileRead.getText().toString();
 String CONTENT = "";
 FileInputStream fis=null;
 try
 {
 fis = openFileInput(FILENAME);
 byte[] buffer = new byte[1024];
 int count = fis.read(buffer);
 if (count>0)
 CONTENT = new String(buffer);
 else
 CONTENT="File không có nội dung";
 }
}

```

```

 catch (IOException e)
 { e.printStackTrace();
 CONTENT=e.getMessage();
 }
 finally
 {
 try
 { fis.close();
 }
 catch (IOException e)
 { e.printStackTrace();
 CONTENT=e.getMessage();
 }
 }
 return CONTENT;
 }

private String ReadFileFromCache()
{
 String FILENAME = etFileRead.getText().toString();
 String CONTENT = "";
 // biến f là đối tượng trả đến thư mục cache của ứng dụng.
 File f = getCacheDir();
 // lấy về chuỗi đường dẫn tuyệt đối trả đến thư mục cache
 String cacheDir = f.getPath();
 // tạo đối tượng File trả đến file myMedia.dat trong thư mục của cache
 File cacheFile = new File(cacheDir,FILENAME);
 FileInputStream fos = null;
 try
 { // đọc file ra biến str
 fos = new FileInputStream(cacheFile);
 byte[] buffer = new byte[1024];
 int count = fos.read(buffer);
 if (count>0)
 CONTENT = new String(buffer);
 else
 CONTENT="File không có nội dung";
 }
 catch (Exception e)
 { e.printStackTrace();
 }
 finally
 {
 try
 {
 fos.close();
 }
 catch (Exception e)
 { e.printStackTrace();
 CONTENT="ERROR";
 }
 }
 //CONTENT trả về 1 trong 3 trường hợp: "", "ERROR" hoặc kết quả đọc được từ file
 return CONTENT;
}

private StringBuilder ReadFileFromExternal()
{
 String FILENAME = etFileRead.getText().toString();
 StringBuilder builder = new StringBuilder();
 File directory = Environment.getExternalStorageDirectory();
 // assumes that a file article.rss is available on the SD card
}

```

```

File file = new File(directory + "/" + FILENAME);
if (!file.exists())
{
 builder.append("File not found");
 return builder;
}
Log.e("Testing", "Starting to read");
BufferedReader reader = null;
try
{
 reader = new BufferedReader(new FileReader(file));
 String line;
 while ((line = reader.readLine()) != null)
 builder.append(line);
}
catch (Exception e)
{
 e.printStackTrace();
 builder.append(e.getMessage());
}
finally
{
 if (reader != null)
 {
 try
 {
 reader.close();
 }
 catch (IOException e)
 {
 e.printStackTrace();
 builder.append(e.getMessage());
 }
 }
}
return builder;
}

```

```

//=====các phương thức khác sẵn có trong class MainActivity =====
@Override
public boolean onCreateOptionsMenu(Menu menu)
{ // Inflate the menu; this adds items to the action bar if it is present.
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item)
{ // Handle action bar item clicks here. The action bar will
 // automatically handle clicks on the Home/Up button, so long
 // as you specify a parent activity in AndroidManifest.xml.
 int id = item.getItemId();
 if (id == R.id.action_settings) {
 return true;
 }
 return super.onOptionsItemSelected(item);
}
}

```

- B3. Cấp quyền cho ứng dụng được ghi trên thiết bị ngoài trong nội dung file *AndroidManifest.xml* để có như sau:

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.app_15c"
 android:versionCode="1"
 android:versionName="1.0" >

```

```

<uses-sdk android:minSdkVersion="13"
 android:targetSdkVersion="21" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<application android:allowBackup="true"
 android:icon="@drawable/ic_launcher"
 android:label="@string/app_name"
 android:theme="@style/AppTheme" >
 <activity android:name=".MainActivity"
 android:label="@string/app_name" >
 <intent-filter>
 <action android:name="android.intent.action.MAIN" />
 <category android:name="android.intent.category.LAUNCHER" />
 </intent-filter>
 </activity>
</application>
</manifest>

```

B4. Chạy ứng dụng để kiểm tra kết quả thực hiện.

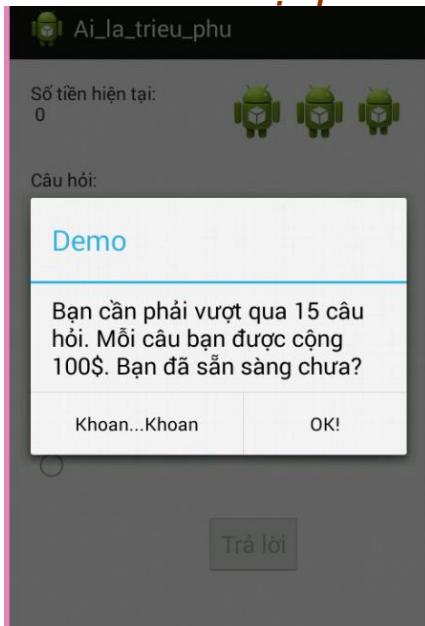
### 3.7. BÀI TẬP TỔNG HỢP CHƯƠNG 3

#### 3.7.1. Game

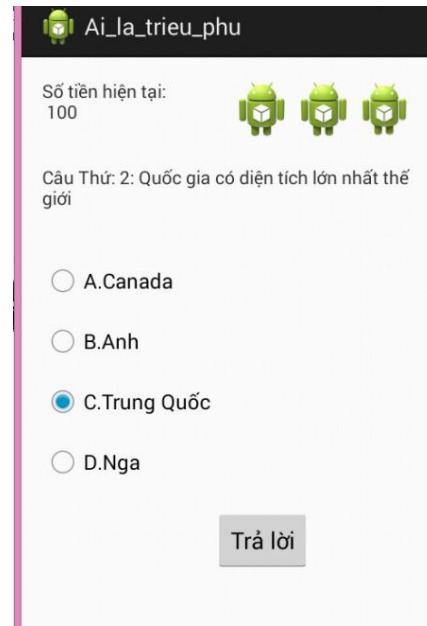
Yêu cầu chung: Sử dụng các loại dữ liệu đã biết để thực hiện các chức năng:

- Lưu lại game đang chơi. Sau đó có thể mở lại chơi tiếp
- Ghi nhận các kỷ lục về thời gian/điểm/... của 5 người chơi cao nhất
- Phân cấp độ khó
- Khuyến khích tổ chức game thành các “màn” (level)

##### 3.7.1.1. Game “Ai là triệu phú”

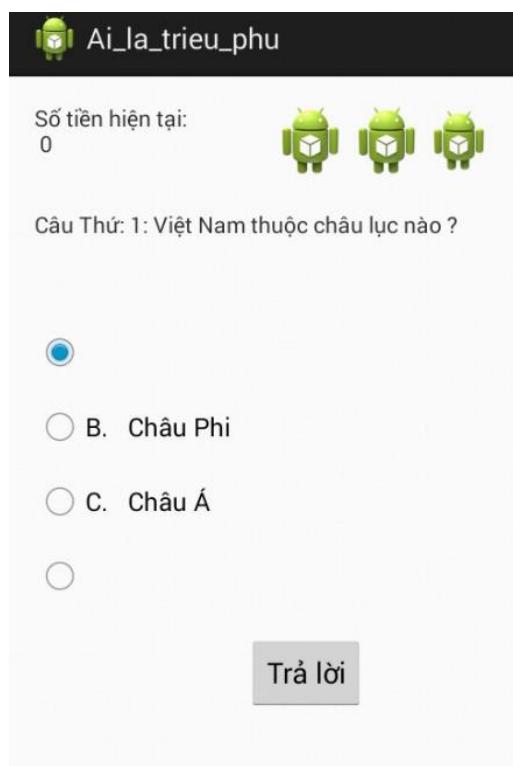


Hình 3-47



Hình 3-48

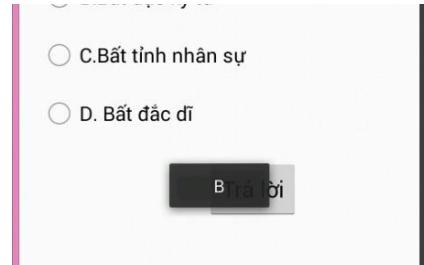
- Đầu chương trình hiện Dialog giới thiệu (hình 3-47)
- Sau khi người dùng nhấn button OK, xuất hiện màn hình có dạng như hình 3-48
- Để là người chiến thắng trong trò chơi này cần phải vượt qua 15 câu hỏi . Mỗi câu trả lời đúng thì ta được cộng 100\$ . Sai thì bị trừ điểm và chơi lại từ đầu (hình 3-49).
- Người chơi có 3 quyền trợ giúp cho mỗi lần chơi (dùng chung cho cả 15 câu hỏi) là:
  - 50/50: bỏ bớt 2 đáp án sai (hình 3-50)
  - Đổi câu hỏi.
  - Gợi ý câu trả lời: chương trình sẽ tạo ra giá trị ngẫu nhiên 1 trong 3 giá trị (-1, 0, 1). Với 0 là câu trả lời đúng, -1 là câu trả lời liền trước, và 1 là câu trả lời liền sau. (hình 3-51)



Hình 3-50



Hình 3-49



Hình 3-51



Hình 3-52



Hình 3-53

- Khởi đầu, màn hình ứng dụng tương tự như hình 26A.
- Trò chơi sẽ mặc định gồm có 6 màu: đỏ , vàng , cam , xanh lá, xanh dương và nâu .
- Máy sẽ chọn ngẫu nhiên 4 màu theo thứ tự (ví dụ xanh lá ở vị trí 1, vàng ở vị trí 2, xanh dương ở vị trí 3, cam ở vị trí 4). Người chơi có 6 lượt lựa chọn. Nếu chọn đúng 4 màu và 4 vị trí thì chiến thắng. Khi sai, chương trình sẽ báo cho người chơi biết đã đúng bao nhiêu màu và vị trí (hình 26B)

Nếu qua 6 lượt mà không chọn đúng 4 màu và 4 vị trí thì người chơi thua. Ngược lại sẽ dành được chiến thắng.

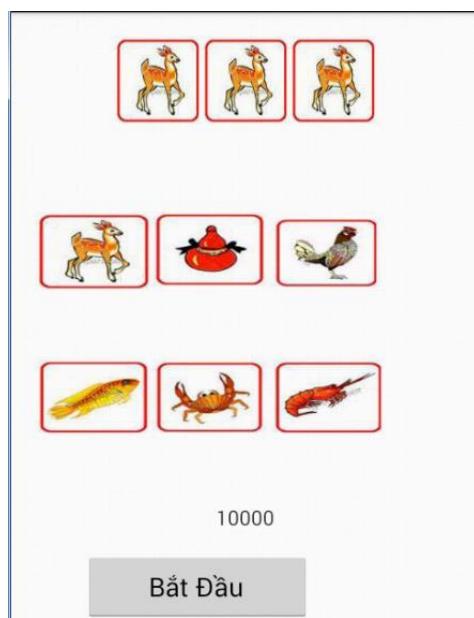
### 3.7.1.3. GAME XÌ ZÁCH

- Yêu cầu của trò chơi là cỗ găng đạt được tổng số điểm từ các lá bài trên tay bạn càng gần 21 càng tốt, nhưng không được vượt quá 21. Số lá bài tối đa được mở là 5.
- Điểm của người chơi là tổng các giá trị của các lá bài đã được mở. Giá trị của các lá bài từ 2 đến 10 là chính con số trên lá bài. Các lá bài Bồi (Jack), Đầm (Queen), Già (King) là 10. Giá trị của bất kỳ con xì nào trong bài có thể được tính là 1, 10 hay 11 (chương trình sẽ chọn bất kỳ 1 trong 3 giá trị đó để giá trị bài của người chơi càng cao càng tốt).
- Mọi người chơi phải có số điểm ít nhất 16 trong tay với 1 ngoại lệ là Ngũ Linh (có nghĩa là có 5 lá bài trong tay). Nếu không làm đúng như vậy thì người chơi sẽ bị thua ván đó.
- Đầu tiên, mỗi người chơi sẽ được mở 2 lá bài.
- Button “Rút bài”: người dùng tự nhấn khi cần mở thêm lá bài cho mình.
- Button “Chia bài”: chơi lại ván mới.
- Button “Dừng”: người chơi sẽ nhấn khi không còn nhu cầu rút bài. Khi đó, chương trình sẽ tự động rút bài cho “Computer”. Quá trình rút bài cho “Computer” sẽ tiếp tục khi điểm còn nhỏ hơn 16 hoặc nhỏ hơn người chơi và số lá bài còn ít hơn hoặc bằng 5 lá (theo quy định).

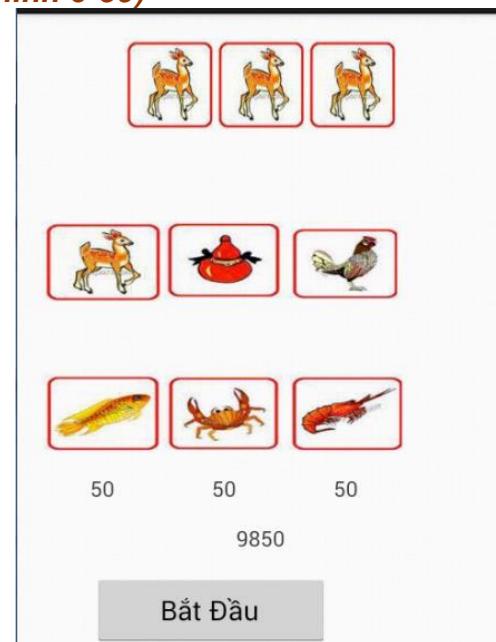


Hình 3-54

### 3.7.1.4. Game “Bầu cua” (từ hình 3-55 đến hình 3-59)



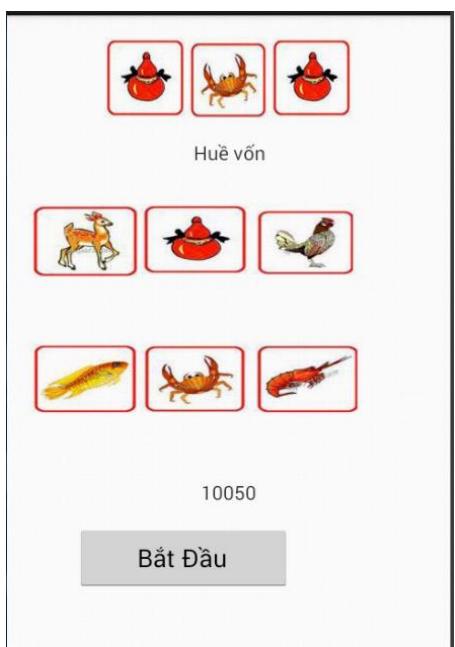
Hình 3-55



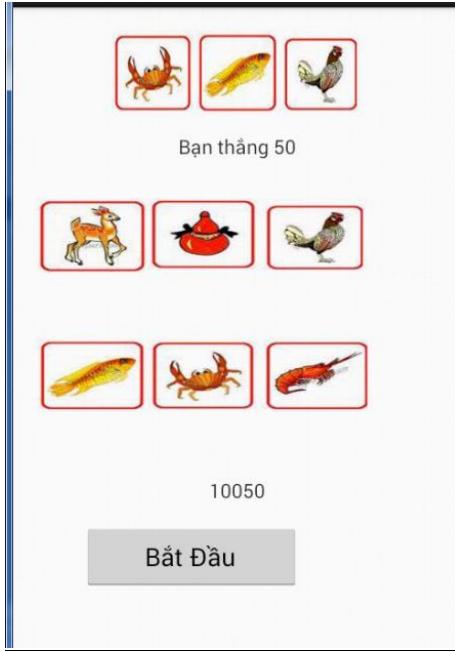
Hình 3-56

- Đầu tiên, người dùng được cấp 10000 điểm.
- Đặt cược: click vào mỗi con mèo chọn bằng cách click trái vào hình. Mỗi lần click tốn 50 điểm
- Button Bắt đầu: sau khi hoàn tất đặt cược, click vào này để chơi
- Trò chơi kết thúc khi điểm của người chơi=0.

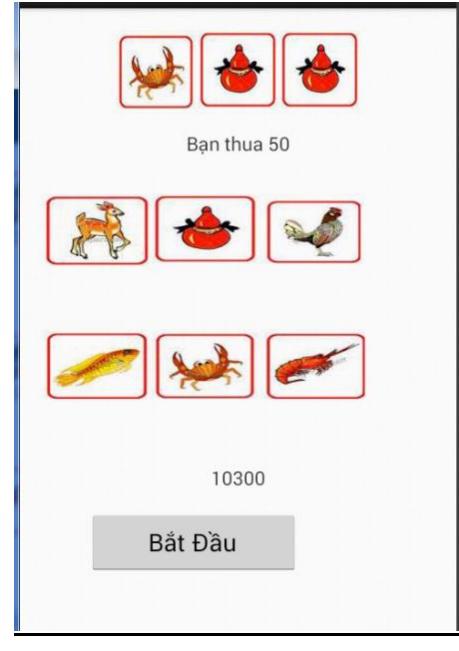
- Khi button “Bắt đầu” được click, 3 image trên cùng sẽ được đổi ngẫu nhiên nhiều lần (trong khoảng thời gian 30 giây), sau đó sẽ dừng hẳn để người dùng biết kết quả.
- Một số kết quả của trò chơi



Bắt Đầu



Bắt Đầu



Bắt Đầu

Hình 3-57

Hình 3-58

Hình 3-59

### 3.7.1.5. Game TÀI - XỈU

- Màn hình bắt đầu có dạng như hình 3-60, trong đó:
  - Số tiền mặc định cược cho người chơi là 200 \$
  - 3 hình của con xúc sắc chưa được lật
- Trò chơi gồm 2 kiểu cược
  - Xỉu: 4 đến 10 điểm
  - Tài: 11 đến 17 điểm

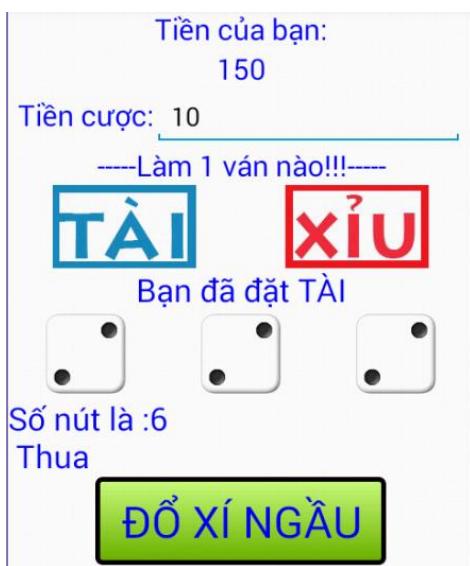


Hình 3-60



Hình 3-61

Cược Tài và Xỉu sẽ đều thua nếu kết quả là Bộ ba đồng nhất (ba viên súc sắc hiện cùng 1 số)



Hình 3-62

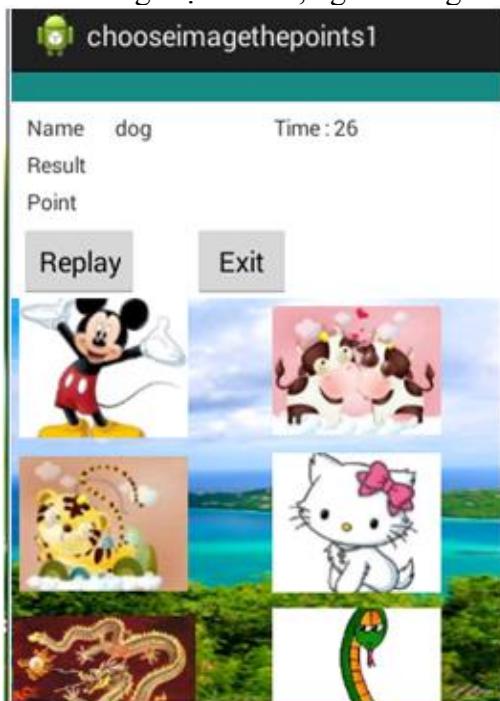


Hình 3-63

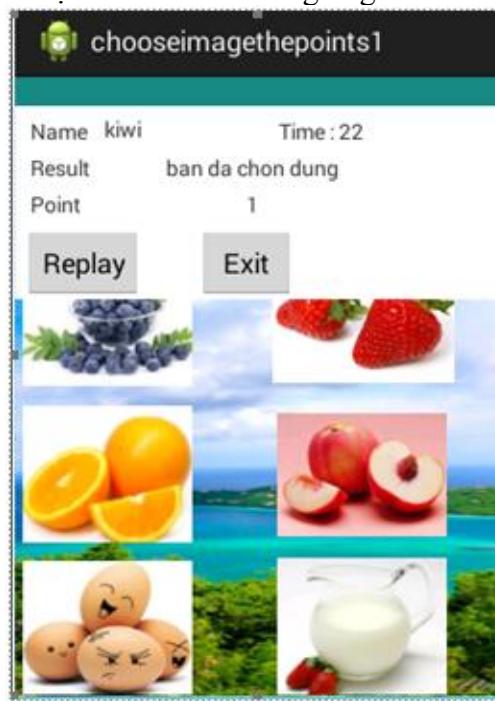
- Hình 3-61: Giao diện của trò chơi khi người chơi đặt số tiền lớn hơn số tiền hiện tại mà họ có:
  - Số tiền mà họ cược sẽ bị xóa
  - Hiển thị thông báo (Toast): “Không đủ tiền cược”.
- Hình 3-62 và hình 3-63: Giao diện của trò chơi khi người chơi làm đúng luật chơi:
  - Số tiền trong tiền cược sẽ được cộng thêm vào số tiền sẵn có của họ, đồng thời giữ nguyên số tiền họ vừa đặt ở lần chơi trước.
  - Hiển thị sự lựa chọn Tài/Xiu
  - Hiển thị số nút, và kết quả thắng thua
- Khi button “Đổ xí ngầu” được click, 3 image thẻ hiện kết quả sẽ được đổi ngẫu nhiên nhiều lần (trong khoảng thời gian 30 giây), sau đó sẽ dừng hẳn để người dùng biết kết quả

### 3.7.1.6. Game Luyện từ vựng tiếng Anh

- Mỗi lượt chơi có thời gian là 30 giây và sẽ được đếm ngược. chương trình sẽ xuất hiện từ tiếng Anh trong mục Name, người dùng sẽ tìm và nhấn chọn hình có tên tương ứng



Hình 3-64



Hình 3-65

- Result: thông báo kết quả chọn (sai hay đúng)
- Point: số điểm đạt được. Nếu trả lời đúng được 1 điểm, ngược lại, khi trả lời sai sẽ bị trừ 1 điểm.
- Số từ xuất hiện (số câu hỏi) tùy thuộc thời gian chọn, nếu người chơi chọn nhanh sẽ có nhiều câu hỏi, do đó khả năng đạt điểm sẽ cao.
- Replay: dùng để chơi lại.
- Exit: thoát chương trình.

### 3.7.1.7. Game Thủ Tài Trí Nhớ 1



Hình 3-66

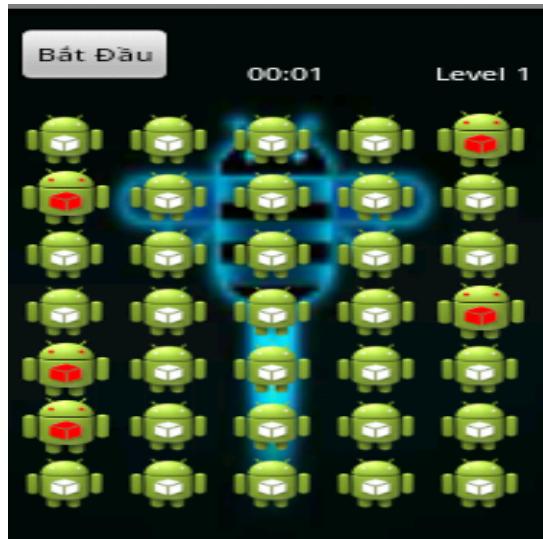


Hình 3-67

- Khi nhấn button “Bắt Đầu” màn hình sẽ hiển thị ngẫu nhiên 20 lá bài (hình 36A). Người chơi cần nhanh mắt ghi nhớ vị trí của các lá bài.
- Sau 10 giây các lá bài đóng lại và yêu cầu bạn chọn lá bài theo yêu cầu (hình 36B).
- 

### 3.7.1.8. Game Thủ Tài Trí Nhớ 2

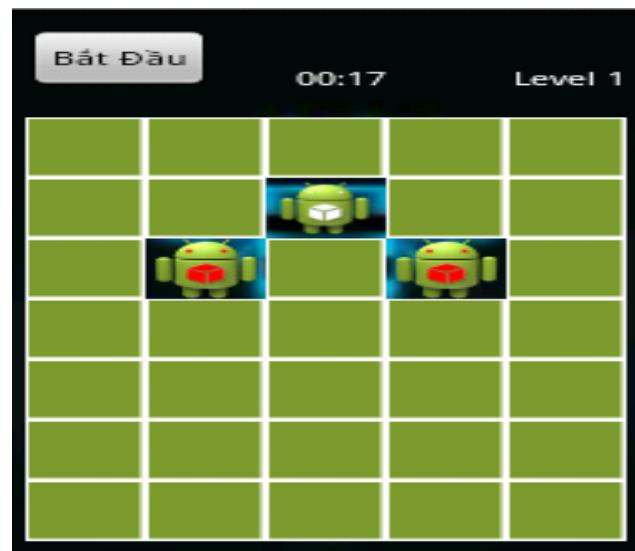
- Khi nhấn button “Bắt Đầu” màn hình sẽ hiển thị những Android màu đỏ xen lẫn với Android trắng. Người chơi cần nhớ vị trí các image Android màu đỏ.
- Sau 5s các hình Android sẽ bị che mắt. Người chơi cần chọn lại đúng các vị trí Android màu đỏ.
- Nếu người chơi tìm đúng lại các vị trí Android màu đỏ thì sẽ thắng được màn (level) đó.
- Game có 10 level (đánh số thứ tự từ 1-5). Số lần được chọn sai của mỗi level là số thứ tự level.
- Từ level 2 đến level 7, mỗi dòng chỉ xuất hiện tối đa 1 Android màu đỏ
- Từ level trở đi, mỗi dòng chỉ xuất hiện tối đa 2 Android màu đỏ.



Hình 3-68



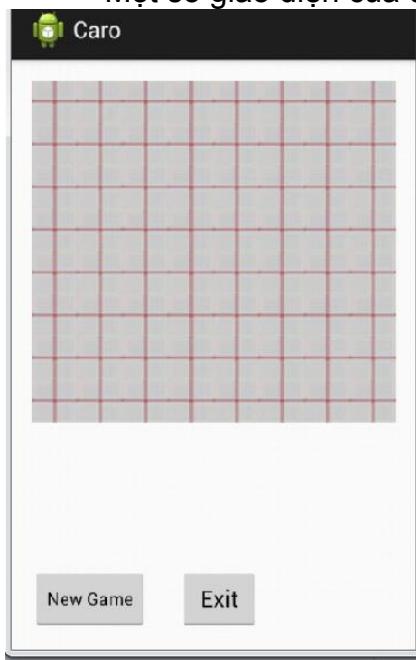
Hình 3-69



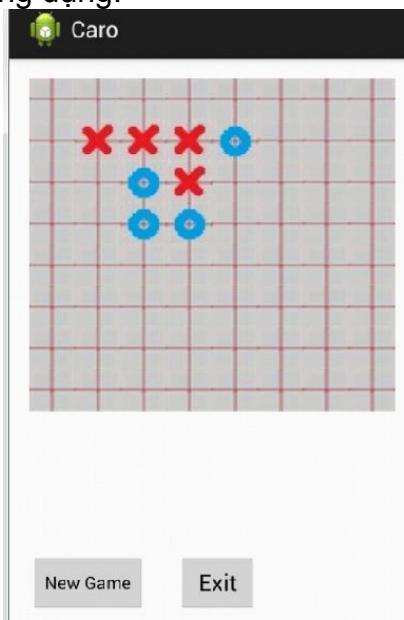
Hình 3-70

### 3.7.1.9. Game Caro

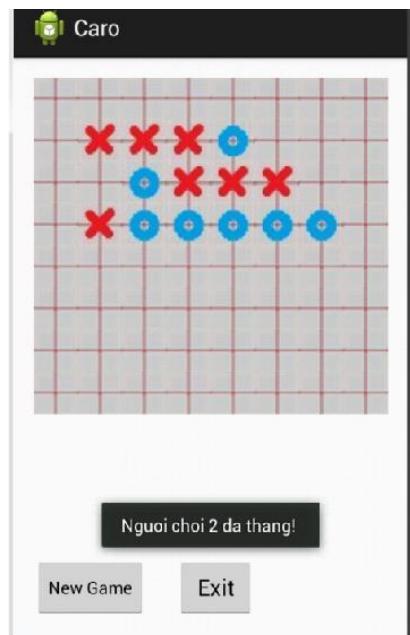
- Giao diện game bao gồm:
  - 1 GridView
  - 2 Button
  - 3 Image: , ,
- Một số giao diện của ứng dụng:



Hình 3-71



Hình 3-72

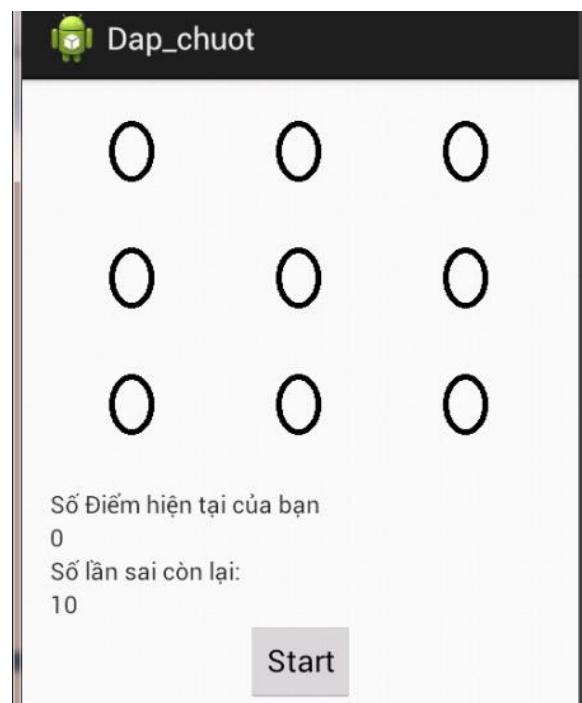


Hình 3-73

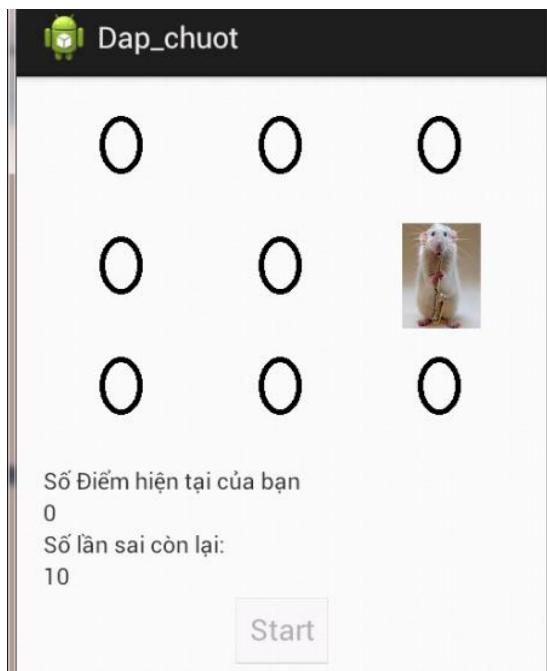
- Quy trình hoạt động:
  - Khi người dùng click vào GridView, hệ thống sẽ kiểm tra tại vị trí vừa được click đã có giá trị chưa thông qua ma trận vuông 8x8. Nếu chưa sẽ cập nhật giá trị và kiểm tra độ dài của chuỗi quân cờ vừa được đánh. Nếu =5 thì người chơi vừa đánh sẽ thắng
  - Khi click vào nút New Game hệ thống sẽ reset ma trận và adapter của GirdView lại.
  - Khi click nút exit sẽ kết thúc chương trình.

### 3.7.1.10. Game Đập Chuột

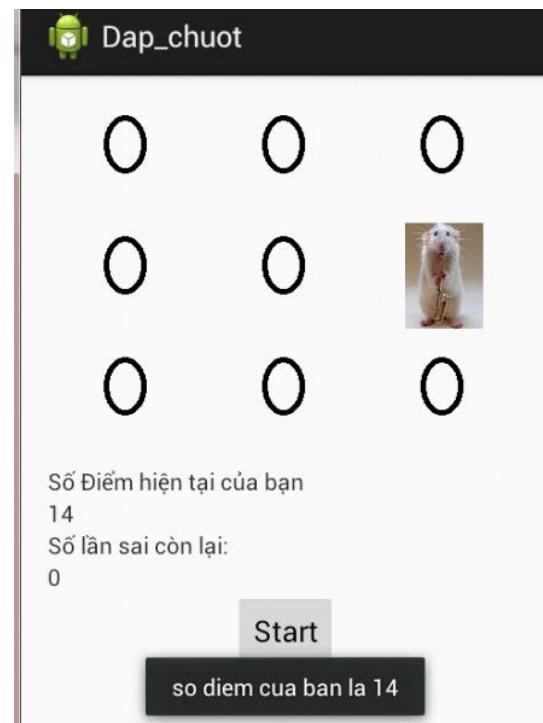
- Thiết kế màn hình gồm: 9 ImageView, 4 TextView, 1 Button.
- Khi người dùng nhấn button Start, chương trình sẽ lần lượt random ra hình con chuột, mỗi hình cách nhau 2 giây.
- Người chơi khi chọn đúng sẽ được cộng lên 1 điểm.
- Khi chọn sai, số lần sai sẽ giảm đi 1 lần.
- Nếu Sai 10 lần cho phép thì sẽ kết thúc trò chơi và thông báo lên số điểm người chơi.



Hình 3-74



Hình 3-75



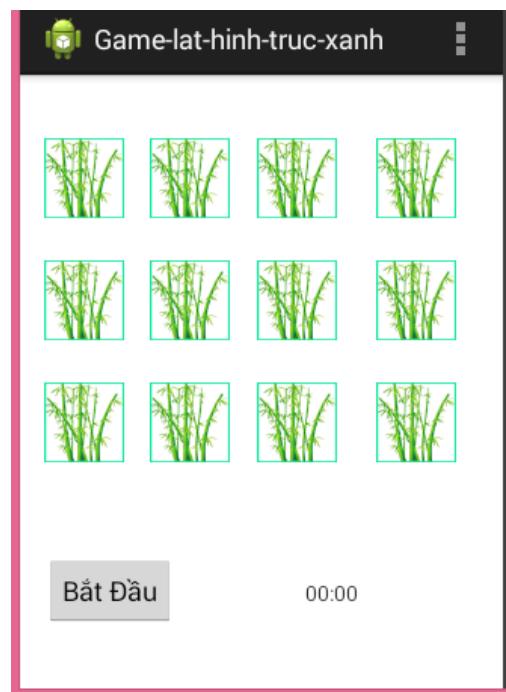
Hình 3-76

### 3.7.1.11. Game Trúc xanh

- Game gồm 5 level, với quy định về thời gian và số hình như sau:

Level	Thời gian(s)	Số hình	Số cột	Số dòng
1	30	12	3	4
2	60	16	4	4
3	90	20	4	5
4	120	25	5	5
5	150	30	5	6

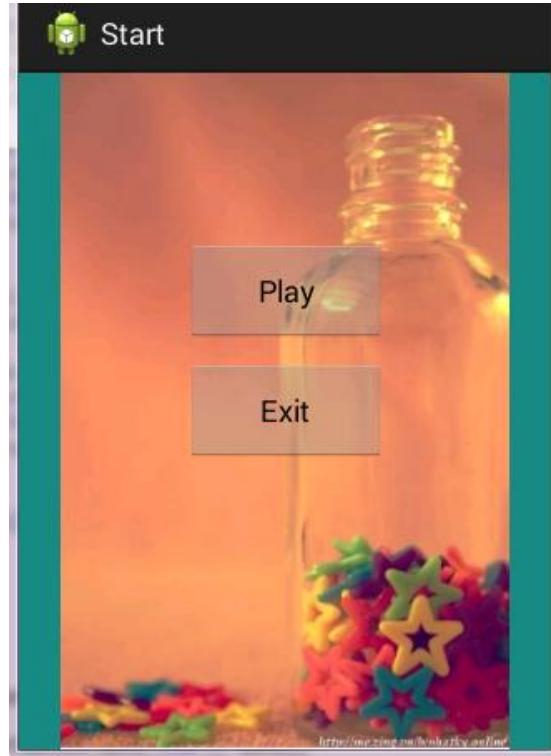
- Khi nhấn button “Bắt đầu” thời gian sẽ bắt đầu đếm ngược.
- Khi có 2 hình chọn liên tiếp giống nhau, 2 hình đó sẽ mất đi, ngược lại 2 hình sẽ đóng lại như ban đầu.



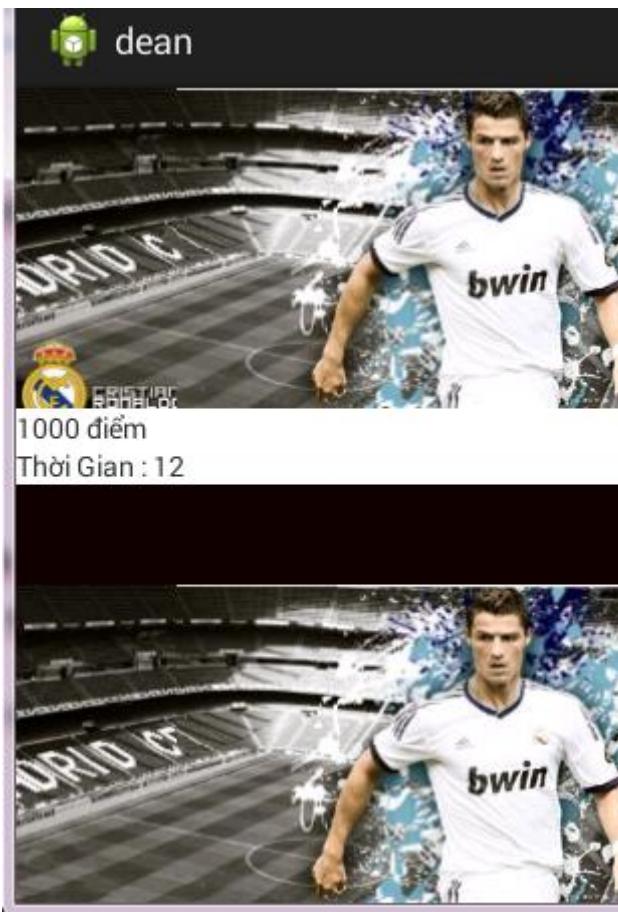
Hình 3-77

### 3.7.1.12. Game Tìm điểm khác nhau

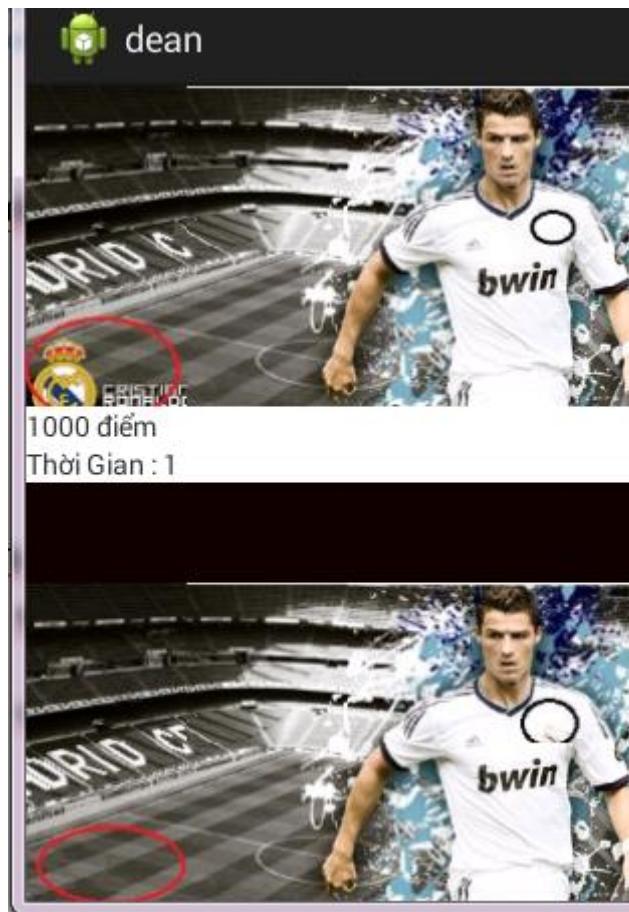
- Tạo màn hình chính (Hình 3-78). Khi người dùng nhấn Play sẽ chuyển sang màn hình 3-79.
- Màn hình 3-79 hiển thị 2 hình trong 2 hình đó sẽ có những điểm khác nhau, người chơi chỉ cần nhấn đúng những điểm khác nhau.
- Khi nhấn đúng điểm sai, điểm sai sẽ được khoanh lại (hình 3-80). Mỗi màn có 3 điểm khác nhau, sau khi chọn đúng 3 điểm, người chơi sẽ được chuyển level. Thời gian của mỗi level là 15s.
- Qua mỗi màn, người chơi được 1000đ.
- Tổ chức game gồm 5 level với mức độ khó tăng dần.



Hình 3-78



Hình 3-79



Hình 3-80

### 3.7.1.13. GAME ĐUỒI HÌNH BẮT CHỮ

- Game minh họa lại gameshow truyền hình "Đuối Hình Bắt Chữ".
- Màn hình chính (hình 3-81)
  - Button “*Hướng dẫn*” sẽ mở màn hình 3-82. Khi nhấn button “*Back*” sẽ trở về màn hình chính (hình 3-81).
  - Button “*Bắt đầu chơi*”: sẽ mở màn hình 3-83
  - Button “*Exit*”: kết thúc ứng dụng.
- Màn hình 3-83: giới thiệu thể lệ trước khi chơi. Nhấn Play để bắt đầu
- Màn hình 3-84: chọn mức chơi (level).



Hình 3-81

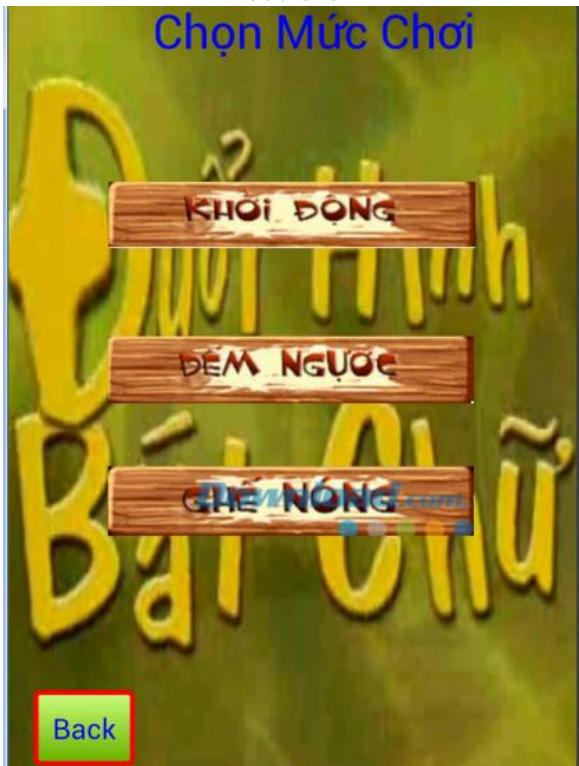
Dựa vào các hình ảnh và gợi ý của chương trình, người chơi cần tìm ra các thành ngữ, tục ngữ, cụm từ hoặc từ có liên quan đến hình đó

**Back**

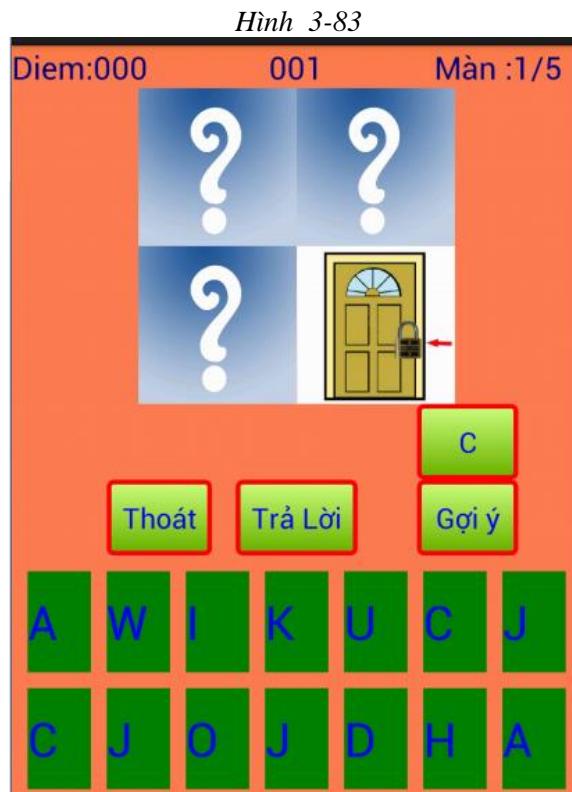
Hình 3-82

Vòng chơi này bao gồm 5 màn nhỏ. Mỗi màn sẽ có 4 miếng ghép chồng lên 1 miếng hình to trả lời đúng bạn sẽ có cơ hội trả lời miếng to. Mỗi câu trả lời đúng càng nhanh thì sẽ được càng nhiều điểm. Chúc bạn may mắn!!!

**Play**



Hình 3-84



Hình 3-83

- Màn hình 3-85:

- Gồm 4 tấm hình nhỏ nằm chồng lên 1 tấm hình to. Sau khi giải đáp được tấm hình nhỏ nào, tấm hình nhỏ đó sẽ mất đi để lộ dần hình to ra.
- Khi tìm đáp án, nếu người chơi có khó khăn khi tìm đáp án, có thể nhấn button “Gợi Ý” để chương trình nêu gợi ý về đáp án
- Thời gian trả lời cho các tấm hình nhỏ là 25s và hình to là 50s. Đáp án không có khoảng trắng và chỉ có thể nhấn chọn các ký tự có trong GridView.

### 3.7.1.14. Số ký tự có trong Gridview được lấy từ đáp án cộng thêm 1 số ký tự ngẫu nhiên để luôn có đủ 14 ký tự. Game Đoán nhạc

- Trò chơi sẽ phát một bài hát ngẫu nhiên có thời lượng là 30s, cùng với 4 lựa chọn.
- Người chơi sẽ nghe và chọn tên bài hát đúng. Người chơi được phép sai 3 lần. Mỗi lần đoán đúng, sẽ qua bài hát khác.
- Nếu chọn sai sẽ trừ dần số lần sai. Nếu số lần sai lớn hơn 3 trò chơi sẽ kết thúc.
- Mỗi lần dành cho 1 người chơi. Khi có nhiều người chơi, người thắng cuộc là người trả lời đúng nhiều bài hát nhất.
- Giao diện:
  - Đầu chương trình sẽ xuất màn hình giới thiệu (Splash Screen Activity) trong 5 giây (hình 3-86), sau đó chuyển sang màn hình chính (hình 3-87).

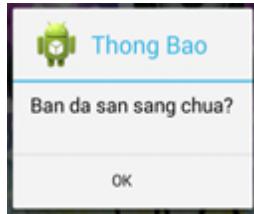


Hình 3-86

- Màn hình chính (hình 3-87) gồm 3 button:
- ☞ Start: mở màn hình có dạng như hình 3-88. Trong màn hình 3-88, khi nhấn button là bắt đầu nghe nhạc và chọn đáp án (hình 3-89).
- ☞ About: xuất hiện màn hình giới thiệu về chương trình, cách chơi và về người viết ứng dụng (SV tự xây dựng màn hình theo ý riêng).
- ☞ Exit: mở màn hình kết thúc ứng dụng (Hình 3-90).



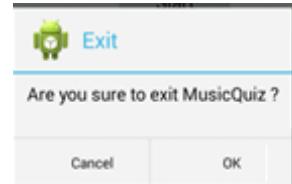
Hình 3-87



Hình 3-88



Hình 3-89



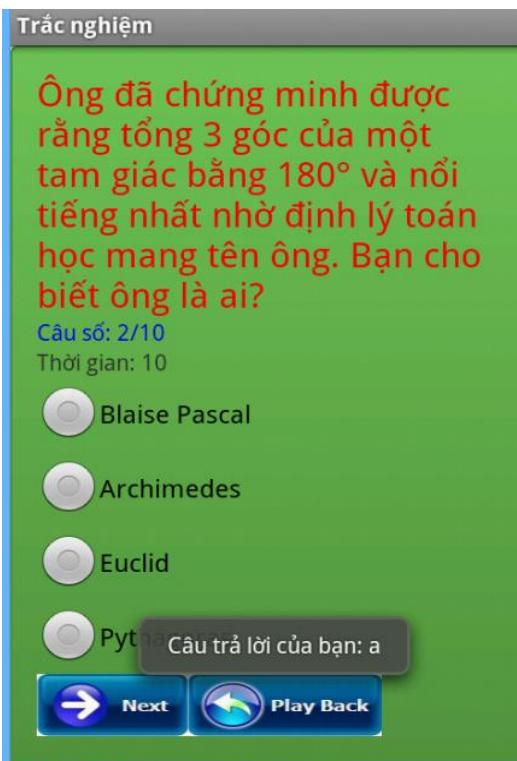
Hình 3-90

### 3.7.1.15. Game trắc nghiệm

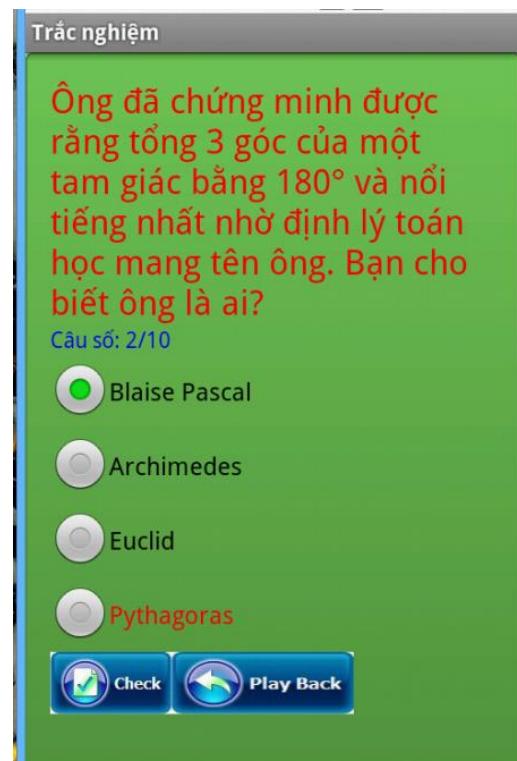
- Tổ chức CSDL trên SQLite.
- Bao gồm các câu hỏi về chuyên ngành công nghệ thông tin, tự nhiên, xã hội, lịch sử, địa lý.
- Mỗi câu đều có 4 đáp án để lựa chọn, lưu ý chỉ được chọn 1 đáp án duy nhất.
- Mỗi lượt chơi sẽ có 10 câu hỏi. Và người chơi có thể chơi lại.
- Ở mỗi câu sẽ có 10 giây để người chơi chọn đáp án.
- Đầu chương trình sẽ xuất màn hình giới thiệu (Splash Screen Activity) trong 5 giây (hình 49A), sau đó chuyển sang màn hình trắc nghiệm (hình 48C).
- Sau khi trả lời hết 10 câu sẽ có xuất hiện nút kiểm tra lại kết quả.
- Khi kiểm tra lại, đáp án đúng của câu hỏi sẽ hiện lên chữ màu đỏ và toast lên thông báo (hình 49C).



Hình 3-91



Hình 3-92



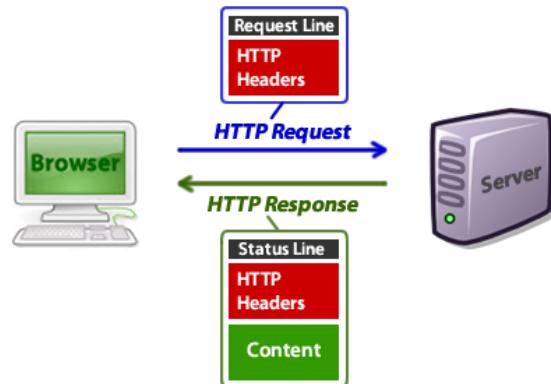
Hình 3-93

## Phần IV: NETWORK & TELEPHONY

### 4.1. NETWORK

#### 4.1.1. Giao thức HTTP

Chức năng của HTTP như là một giao thức kiểu gửi yêu cầu-nhận hồi đáp trong mô hình Client-Server. Trên HTTP, một trình duyệt web (Firefox, Internet Explorer, Chrome...) sẽ đóng vai trò như một người khách, trong khi ứng dụng chạy trên máy server (máy trung tâm) sẽ đóng vai trò như là một máy chủ. Người khách ở đây sẽ gửi một yêu cầu HTTP (HTTP request) lên máy chủ. Máy chủ, nơi lưu trữ một nguồn tài nguyên nào đó (hình ảnh, âm thanh...) sẽ thay mặt máy khách thực thi một chức năng nào đó là trả về một hồi đáp cho máy khách. Hồi đáp này sẽ bao gồm thông tin trạng thái về yêu cầu được gửi đi cùng với nội dung được yêu cầu bởi máy khách trong nó. Giao thức HTTP được thể hiện qua hình 4.1:



Hình 4-1 Hoạt động của giao thức HTTP

Trong mô hình trên tồn tại hai đối tượng chính: Client và Server. Đối tượng Client sẽ gửi một HTTP Request đến máy Server để yêu cầu thực thi một tác vụ hoặc lấy một dữ liệu nào đó. Đối tượng Request chứa các tham số xác định địa chỉ nó được gửi đến cùng với các tham số khác để xác định tác vụ cần thực thi cũng như loại dữ liệu cần lấy. Các tham số này có thể thuộc hai loại: POST và GET. Tham số thuộc loại GET sẽ nằm trong đường dẫn địa chỉ mà đối tượng Request đang nắm giữ, tham số loại POST sẽ không nằm trong đường dẫn địa chỉ.

VD: Gởi đi request có địa chỉ sau: <https://www.google.com.vn/search?q=android>. Đây là một request dạng GET. Ta có thể thấy trên địa chỉ trên có một tham số `q=android` xác định từ khóa yêu cầu máy chủ Google tìm kiếm từ khóa: `android`.

Máy Server sẽ phân tích đối tượng Request để tìm ra yêu cầu của máy Client xử lý và trả về dữ liệu trong đối tượng Response. Máy Client sẽ phân tích đối tượng Response được trả về để lấy các thông tin cần thiết. Đây là cách mà giao thức HTTP hoạt động.

#### 4.1.2. Sử dụng kết nối mạng trong Android

Android cho phép ứng dụng của bạn có thể kết nối với internet hoặc bất kỳ thiết bị nào trong mạng cục bộ để thực hiện các hoạt động mạng.

Khi sử dụng kết nối mạng trong Android, bạn cần thực hiện 1 số công việc:

- Bổ sung các quyền của ứng dụng về kết nối mạng vào file `AndroidManifest.xml`. Các quyền thường dùng gồm:

- Để cho phép ứng dụng kết nối Internet:  
`<uses-permission android:name="android.permission.INTERNET" />`

- Để được quyền kiểm tra trạng thái của kết nối mạng:

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

- Kiểm tra trạng thái của kết nối mạng.
- Thực hiện các yêu cầu của ứng dụng

#### 4.1.2.1. Kiểm tra trạng thái của kết nối mạng

Trước khi bạn thực hiện bất kỳ hoạt động mạng, đầu tiên bạn phải kiểm tra là bạn kết nối với mạng hoặc internet, ... Android cung cấp class *ConnectivityManager*. Bạn cần tạo 1 đối tượng của class này bằng cách gọi phương thức *getSystemService()*. Cú pháp của phương thức này có dạng như dưới đây:

```
ConnectivityManager check = (ConnectivityManager)
 this.context.getSystemService(Context.CONNECTIVITY_SERVICE);
```

Sau khi đã có đối tượng thuộc class *ConnectivityManager*, bạn có thể sử dụng phương thức *getAllNetworkInfo* để có được các thông tin của tất cả các mạng. Phương thức này trả về một mảng của *NetworkInfo*. Vì vậy, bạn sẽ sử dụng mã lệnh có dạng như sau:

```
NetworkInfo[] info = check.getAllNetworkInfo();
```

Cuối cùng bạn kiểm tra trạng thái kết nối (Connected State) của mạng theo cú pháp đưa ra dưới đây:

```
for (int i = 0; i<info.length; i++)
{
 if (info[i].getState() == NetworkInfo.State.CONNECTED)
 {
 Toast.makeText(context, "Internet is connected",
 Toast.LENGTH_SHORT).show();
 }
}
```

Các trạng thái của mạng gồm: CONNECTED, CONNECTING, DISCONNECTED, DISCONNECTING, SUSPENDED, UNKNOWN

#### 4.1.2.2. Performing Network Operations

Android cung cấp 2 class *HttpURLConnection* và *URL* để xử lý các hoạt động này.

- Tạo 1 đối tượng của class URL bằng cách cung cấp địa chỉ của website với cú pháp minh họa như sau:

```
String link = "http://www.google.com";
URL url = new URL(link);
```

- Gọi phương thức *openConnection* của class URL để nhận được một đối tượng *HttpURLConnection*. Sau đó bạn cần phải gọi phương thức kết nối của class *HttpURLConnection*.

```
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
```

- Gọi phương thức kết nối của class *HttpURLConnection*.

```
conn.connect();
```

- Sử dụng các lệnh để thực hiện công việc cần làm.

VD: lấy mã HTML từ trang web bằng cách sử dụng class *InputStream* và *BufferedReader*:

```
InputStream is = conn.getInputStream();
BufferedReader reader =new BufferedReader(new
 InputStreamReader(is, "UTF-8"));
String webPage = "",data="";
while ((data = reader.readLine()) != null)
{
 webPage += data + "\n";
}
```

Các phương thức thường dùng trong class *HttpURLConnection*:

<i>Method</i>	<i>Description</i>
<i>connect()</i>	Mở kết nối

disconnect()	This method releases this connection so that its resources may be either reused or closed
getRequestMethod()	trả về phương thức được yêu cầu sẽ được sử dụng để thực hiện các yêu cầu đến máy chủ HTTP từ xa
getResponseCode()	trả về mã phản hồi bởi các máy chủ HTTP từ xa
setRequestMethod(String method)	Thiết lập lệnh yêu cầu sẽ được gửi đến máy chủ HTTP từ xa
usingProxy()	Cho biết kết nối có sử dụng 1 máy chủ proxy (server proxy) hay không?

#### 4.1.2.3. Sử dụng HTTP

Việc sử dụng giao thức HTTP trên Android hiển nhiên cũng tuân theo cách trên với các bước như sau:

- i. Khởi tạo một đối tượng HttpClient.
- ii. Khởi tạo một phương thức HTTP: PostMethod hoặc GetMethod.
- iii. Thiết lập tham số cho phương thức HTTP.
- iv. Thực thi HTTP request bằng cách sử dụng HttpClient.
- v. Thao tác trên đối tượng Response được trả về.

##### 4.1.2.3.1. Khai báo phương thức HTTP với GET Request

Định nghĩa một class java như sau:

```

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URI;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import java.util.ArrayList;
import java.util.List;

public class NetworkManager
{
 public String executeHttpGet(String uri) throws Exception
 {
 BufferedReader inBR = null;
 String result = "";
 try
 {
 HttpClient client = new DefaultHttpClient();
 HttpGet request = new HttpGet();
 // gán địa chỉ sẽ gửi đối tượng request đi
 request.setURI(new URI(uri));
 // thực thi một HTTP request và trả về một đối tượng Response
 HttpResponse response = client.execute(request);

 inBR = new BufferedReader(new
 InputStreamReader(response.getEntity().getContent()));
 StringBuffer sb = new StringBuffer("");
 String line = "";
 String eol = System.getProperty("line.separator");
 while ((line = inBR.readLine())!=null)

```

```

 {
 sb.append(line + eol);
 }
 inBR.close();
 result = sb.toString();
 }
 finally
 {
 if (in != null)
 {
 try
 {
 in.close();
 }
 catch (IOException e)
 {
 e.printStackTrace();
 }
 }
 }
 return result;
}
String result = "";
try
{
 String link = urlField.getText().toString();
 URL url = new URL(link);
 HttpURLConnection conn = (HttpURLConnection) url.openConnection();
 conn.setReadTimeout(10000);
 conn.setConnectTimeout(15000);
 conn.setRequestMethod("GET");
 conn.setDoInput(true);
 conn.connect();
 InputStream is = conn.getInputStream();
 BufferedReader reader = new BufferedReader(new InputStreamReader(is, "UTF-8"))
);
 String data = null;
 while ((data = reader.readLine()) != null)
 {
 result += data + "\n";
 }
}
catch(Exception e)
{
 result="Exception: " + e.getMessage();
}
 data.setText(result);
}

}

```

Trong đó:

- **HttpClient**: đại diện cho máy Client trong mô hình trên. Một đối tượng *HttpClient* đóng gói tất cả các đối tượng cần thiết để thực thi một HTTP request (địa chỉ gửi đi, các tham số đi kèm). Đây cũng là đối tượng xử lý cookie, chứng thực tài khoản, quản lý kết nối cũng như các tính năng khác của giao thức HTTP.
- **response.getEntity()**: trả về một đối tượng *HttpEntity*. Đây là đối tượng chứa các thẻ HTML (HTML entity) được trả về từ server.
- **response.getEntity().getContent()**: trả về một đối tượng kiểu *InputStream*. Đây là một đối tượng đại diện cho một luồng (stream) nối đến khối dữ

- liệu được trả về chưa trong response. Dữ liệu sẽ được đọc từ đối tượng này.
- **`new InputStreamReader(...)`**: khởi tạo một đối tượng InputStreamReader để tạo thành một bộ đọc từ luồng mới tạo ở trên.
  - **`new BufferedReader()`**: tạo một BufferedReader để tạo thành một bộ đọc có hỗ trợ bộ đệm (buffer) giúp tăng tốc việc đọc dữ liệu từ luồng kể trên.
  - **`StringBuffer ( sb.append(line + eol))`**: tiến hành dựng một đối tượng String từ dữ liệu nhận về sau khi thực thi một HTTP request. Thực tế thì đối tượng String này chính là chuỗi chứa nội dung HTML được trả về.

#### 4.1.2.3.2. Khai báo phương thức HTTP với POST Request

Các bước làm việc với giao thức HTTP cùng với một request dạng POST cũng tương tự như với request dạng GET. Bổ sung phương thức `executeHttpPost` vào class `NetworkManager` như sau:

```
public class NetworkManager
{
 public String executeHttpGet(String uri) throws Exception
 {
 // chứa mã lệnh đã có như mình họa ở trên
 }

 public String executeHttpPost(String uri, List<NameValuePair>
 postParameters) throws Exception
 {
 BufferedReader inBR = null;
 String result = "";
 try
 {
 HttpClient client = new DefaultHttpClient();
 HttpPost request = new HttpPost(uri);
 //Tiến hành mã hóa số tham số trên để kèm request
 UrlEncodedFormEntity formEntity = new UrlEncodedFormEntity(postParameters);
 request.setEntity(formEntity);
 HttpResponse response = client.execute(request);
 inBR = new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
 StringBuffer sb = new StringBuffer("");
 String line = "";
 String eol = System.getProperty("line.separator");
 while ((line = in.readLine()) != null)
 {
 sb.append(line + eol);
 }
 inBR.close();
 result = sb.toString();
 }
 finally
 {
 if (inBR != null)
 {
 try
 {
 inBR.close();
 }
 catch (IOException e)
 {
 e.printStackTrace();
 }
 }
 }
 return result;
 }
}
```

}

Trong đó:

- List<NameValuePair> postParameters = new ArrayList<NameValuePair>(): định nghĩa 1 danh sách để chứa các tham số gửi đi với POST Request. Các tham số này là dữ liệu dùng để thực thi các hành động trên máy chủ hoặc các tham số dùng để xác định dữ liệu trả về.
- postParameters.add(new BasicNameValuePair("one", "valueGoesHere")): thêm các tham số vào danh sách tham số đã định nghĩa ở trên.

#### 4.1.2.3.3. Sử dụng class đã xây dựng

Sử dụng class NetworkManager trên như sau:

```
String resultGet="";
String resultPost="";
NetworkManager networkManager = new NetworkManager();
try
{
 resultGet = networkManager.executeHttpGet("http://www.google.com.vn/search?q=android+network");
 System.out.print(result);
}
catch (Exception e)
{
 e.printStackTrace();
}

List<NameValuePair> postParameters = new ArrayList<NameValuePair>();
postParameters.add(new BasicNameValuePair("q", "android+network"));
```

hoặc:

```
try
{
 resultPost = networkManager.executeHttpPost("http://www.abc.com",
 postParameters);
 System.out.print(result);
}
catch (Exception e)
{
 e.printStackTrace();
}
```

### 4.1.3. WebView

- *Giới thiệu:* WebView là 1 view cho phép hiển thị các trang web bên trong ứng dụng của bạn.
- *Sử dụng:*
  - Để thêm WebView vào ứng dụng, bạn phải khai báo phần tử <WebView> vào file layout của ứng dụng. Ví dụ:

```
<WebView xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@+id/webView"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent" />
```

  - Tương tự như đối với các loại view khác, để sử dụng WebView bạn phải khai báo 1 biến kiểu WebView tham chiếu đến view có trong layout. Ví dụ:

```
WebView browser = (WebView) findViewById(R.id.webView);
```

  - Để tải một địa chỉ web vào WebView, bạn cần phải gọi phương thức *loadUrl (String url)* của class WebView với tham số là địa chỉ yêu cầu. Ví dụ:

```
browser.loadUrl("http://www.google.com");
```

- Ngoài chức năng tải 1 url, bạn có thể kiểm soát WebView hiệu quả hơn bằng cách sử dụng các phương thức định nghĩa trong class WebView như sau:
  - `canGoBack()` Phương thức này xác định WebView sẽ tổ chức các địa chỉ đã đi qua để có thể sử dụng chức năng Back (trở lại) sau này
  - `canGoForward()` Phương thức này xác định WebView sẽ tổ chức các địa chỉ đã đi lui (Back) để có thể sử dụng chức năng Forward sau này
  - `clearHistory()` Phương thức này sẽ xóa các địa chỉ đã được lưu trong Forward và Back
  - `getProgress()` Lấy thông tin về tiến trình (progress) của trang hiện tại
  - `getTitle()` Lấy tiêu đề của trang hiện tại
  - `getUrl()` Lấy URL của trang hiện tại
  - `findAllAsync(String findString)` Tìm chuỗi findString và làm nổi bật chúng.
- Nếu bạn muốn khi click vào bất kỳ liên kết bên trong trang web của WebView, trang đó sẽ không được nạp bên trong WebView của bạn. Để làm được điều đó bạn cần mở rộng (extends) class của mình từ WebViewClient và ghi đè lên phương thức của nó như sau:

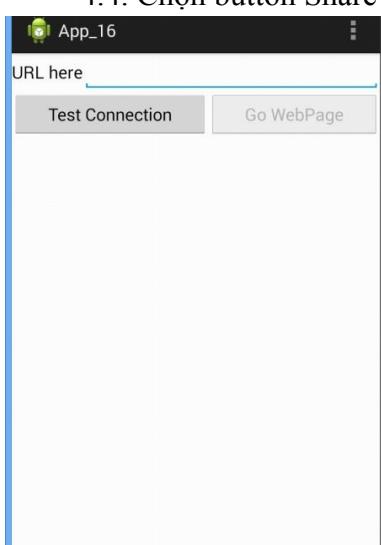
```
private class MyBrowser extends WebViewClient
{
 @Override
 public boolean shouldOverrideUrlLoading(WebView view, String url)
 {
 view.loadUrl(url);
 return true;
 }
}
```

## BÀI THỰC HÀNH App\_16

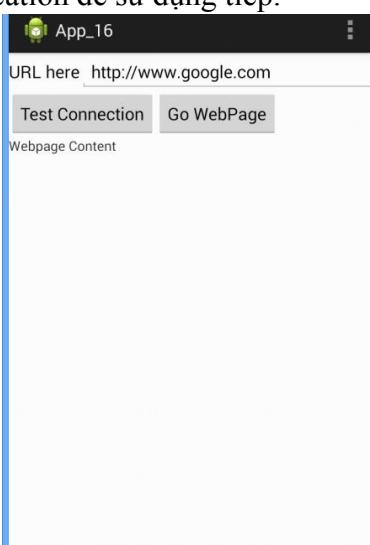
### Yêu cầu:

Xây dựng 1 project, trên giao diện chính gồm 1 TextView, 1 EditText 2 button. Khi bắt đầu ứng dụng, button “Go WebPage” bị mờ (disabled). Sau khi click chọn button “Test Connection” sẽ thông báo về trạng thái mạng hiện có. Nếu có kết nối mạng, button “Go WebPage” sẽ được enable.

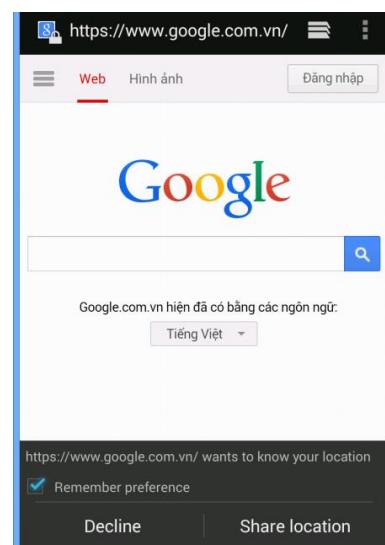
Khi người dùng chưa nhập địa chỉ website thì button Go WebPage bị mờ (hình 4.2). Sau đó, khi đã nhập địa chỉ và chọn button “Go WebPage” (hình 4.3), sẽ tìm và mở trang cần đến trong 1 activity khác. Do AVD mới gọi ứng dụng web lần đầu nên màn hình có dạng như hình 4.4. Chọn button Share location để sử dụng tiếp.



Hình 4-2 Chưa nhập URL



Hình 4-3 Đã nhập URL



Hình 4-4 Khi gọi ứng dụng web lần đầu



 Thực hiện:

- B1. Tạo mới project. Cấp quyền truy cập Internet cho ứng dụng trong file *AndroidManifest.xml*

```
<!-- Network State Permissions -->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<!-- Internet Permissions -->
<uses-permission android:name="android.permission.INTERNET"/>
<!-- Wifi Network State Permissions. Lệnh này có thể không cần dùng -->
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
```

- B2. Chính sửa nội dung file *activity\_main.xml* để tạo giao diện chính với nội dung. Lưu ý lời gọi phương thức của 2 button được thực hiện thông qua thuộc tính

```
 android:onClick:
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:id="@+id/LinearLayout1"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 tools:context=".MainActivity" >
 <LinearLayout android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:orientation="horizontal">
 <TextView android:id="@+id/textView1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="URL here"
 android:textAppearance="?android:attr/textAppearanceMedium"/>
 <EditText android:id="@+id/editText1"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:inputType="textEmailAddress"
 android:ems="10" />
 </LinearLayout>
 <LinearLayout android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:orientation="horizontal">
 <Button android:id="@+id/button1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:onClick="TestConnection"
 android:layout_weight="1"
 android:text="Test Connection" />
 <Button android:id="@+id/button2"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:onClick="GoWebPage"
 android:layout_weight="1"
 android:text="Go WebPage" />
 </LinearLayout>
</LinearLayout>
```

- B3. Tạo thêm mới 1 file layout để hiển thị nội dung website sẽ truy cập với nội dung:

```
<?xml version="1.0" encoding="utf-8"?>
<WebView xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@+id/webView1"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"/>
```

- B4. Chính sửa nội dung file *MainActivity.java* để có:

```
package com.example.app_16;
```

```
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends Activity
{ EditText urlField;
 public TextView data;
 Button btnCheck, btnDownload;
 @Override
 protected void onCreate(Bundle savedInstanceState)
 { super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 urlField = (EditText)findViewById(R.id.editText1);
 btnCheck = (Button)findViewById(R.id.button1);
 btnDownload = (Button)findViewById(R.id.button2);
 btnDownload.setEnabled(false);
 }
 public void TestConnection(View v)
 {
 if (isConnectingToInternet())
 { Toast.makeText(this, "Internet is connected", Toast.LENGTH_SHORT).show();
 btnDownload.setEnabled(true);
 }
 else
 { Toast.makeText(this, "not conencted to internet", Toast.LENGTH_SHORT).show();
 btnDownload.setEnabled(true);
 }
 }
 public void GoWebPage(View v)
 {
 String url=urlField.getText().toString();
 if (url.trim().length()>0)
 { Intent intent = new Intent(this, WebViewActivity.class);
 intent.setType("text/plain");
 intent.putExtra(Intent.EXTRA_TEXT, url);
 intent.setAction("android.intent.action.SEND");
 startActivity(intent);
 }
 }
 //check Internet conenction.
 private boolean isConnectingToInternet()
 {
 ConnectivityManager connectivity = (ConnectivityManager)
 this.getSystemService(Context.CONNECTIVITY_SERVICE);
 if (connectivity != null)
 { NetworkInfo[] info = connectivity.getAllNetworkInfo();
 if (info != null)
 for (int i = 0; i < info.length; i++)
 if (info[i].getState() == NetworkInfo.State.CONNECTED)
 return true;
 }
 return false;
 }
}
```

```
/* Sau dòng này là nội dung cài đặt sẵn có của 2 phương thức onCreateOptionsMenu và
onOptionsItemSelected */
}
```

**B5.** Tạo mới file file *WebViewActivity.java* để tải nội dung website cần truy cập:

```
package com.example.app_16;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.webkit.WebView;
import android.widget.Toast;

@SuppressLint("SetJavaScriptEnabled")
public class WebViewActivity extends Activity
{
 private WebView webView;
 String url="";
 public void onCreate(Bundle savedInstanceState)
 {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.web_view_layout);
 Intent intent = getIntent();
 if (intent!=null)
 {
 String action = intent.getAction();
 String type = intent.getType();
 if ((Intent.ACTION_SEND.equals(action)) && ("text/plain".equals(type)))
 {
 url=intent.getStringExtra(Intent.EXTRA_TEXT);
 webView = (WebView) findViewById(R.id.webView1);
 webView.getSettings().setJavaScriptEnabled(true)
 webView.loadUrl(url);
 }
 else
 Toast.makeText(this, "URL not found", Toast.LENGTH_SHORT).show();
 }
 else
 Toast.makeText(this, "URL not found", Toast.LENGTH_SHORT).show();
 }
}
```

## 4.2. Telephony

### 4.2.1. SMS

#### 4.2.1.1. Gửi tin nhắn

Có 2 cách để gửi 1 SMS trên thiết bị sử dụng Android là:

- Sử dụng SmsManager
- Sử dụng Built-in Intent

##### 4.2.1.1.1. Sử dụng SmsManager để gửi SMS

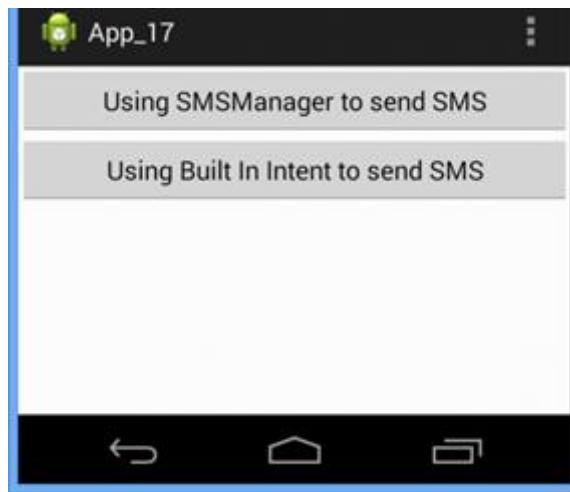
- Các *SmsManager* giúp quản lý hoạt động nhắn tin SMS như gửi dữ liệu đến các thiết bị di động nhất định. Bạn có thể tạo ra đối tượng này bằng cách gọi phương thức tĩnh *SmsManager.getDefault()* như sau:  
`SmsManager smsManager = SmsManager.getDefault();`
- Khi đã có đối tượng *SmsManager*, bạn có thể sử dụng phương thức *sendDataMessage()* để gửi tin nhắn SMS vào số điện thoại di động theo dạng thức quy định như sau:  
`smsManager.sendTextMessage("phoneNo", null, "SMS text", null, null);`

- Một số phương thức khác sẵn có trong class `SmsManager`:
  - `ArrayList<String> divideMessage(String text)`  
Chia 1 tin nhắn văn bản thành nhiều mảnh (fragment) với kích thước tối đa của các fragment không lớn hơn kích thước tối đa của 1 SMS.
  - `static SmsManager getDefault()`  
Được dùng để lấy thể hiện mặc định (default instance) của một `SmsManager`.
  - `void sendDataMessage(String destinationAddress, String scAddress, short destinationPort, byte[] data, PendingIntent sentIntent, PendingIntent deliveryIntent)`  
Gởi dữ liệu dựa trên tin nhắn SMS đến một công ứng dụng cụ thể.
  - `void sendMultipartTextMessage(String destinationAddress, String scAddress, ArrayList<String> parts, ArrayList<PendingIntent> sentIntents, ArrayList<PendingIntent> deliveryIntents)`  
Gởi văn bản gồm nhiều phần dựa trên tin nhắn.
  - `void sendTextMessage(String destinationAddress, String scAddress, String text, PendingIntent sentIntent, PendingIntent deliveryIntent)`  
Gởi văn bản dựa trên tin nhắn SMS.

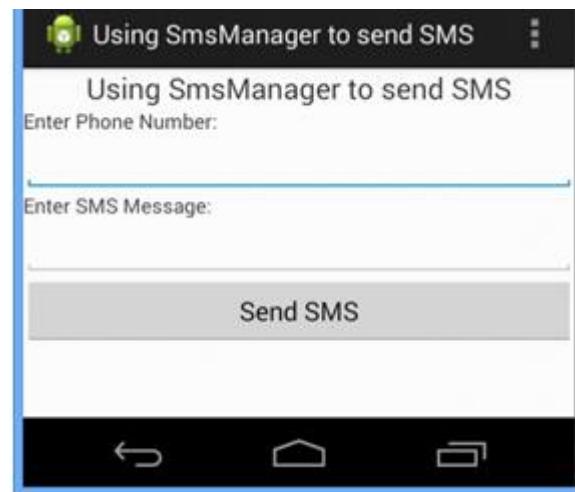
## BÀI THỰC HÀNH App\_17

### ☞ Yêu cầu:

Tạo 1 project sử dụng chung cho các bài tập về telephony có giao diện như hình minh họa. Khi nhấn button “Using SmsManager to send SMS” sẽ xuất hiện màn hình tương ứng. Trên activity thứ 2 này sẽ sử dụng đối tượng `SmsManager` để gởi 1 SMS đến 1 số điện thoại có thực.



Hình 4-5 Màn hình chính của ứng dụng

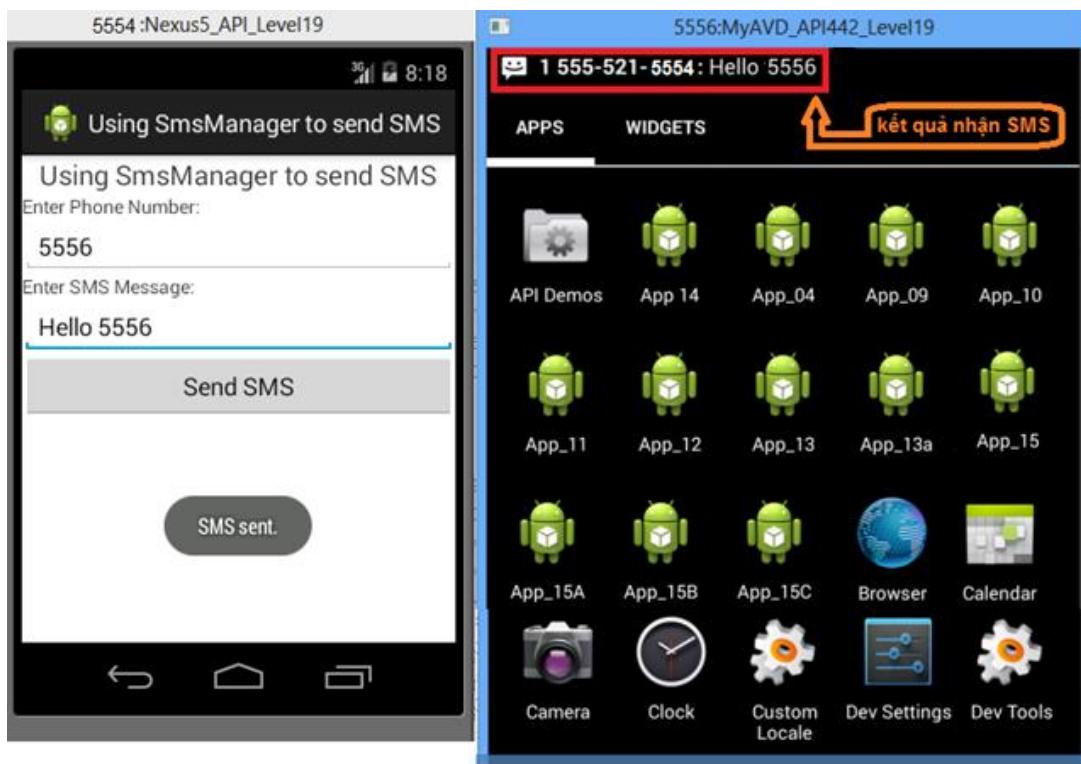


Hình 4-6 Màn hình gởi tin nhắn

Để thử nghiệm với project này, bạn cần có thiết bị di động thực sự, được trang bị hệ điều hành Android và được kết nối trực tiếp với máy tính.

Nếu không có điện thoại thật, bạn có thể dùng 2 điện thoại ảo. Để gởi tin nhắn giữa hai máy ảo với nhau ta thay số điện thoại người nhận bằng cách thay đổi số điện thoại người nhận bằng *Emulator's port* của máy ảo. *Emulator's port* của máy ảo thứ 1 (số điện thoại mặc định là 5554) và của máy ảo thứ 2 (số điện thoại mặc định là 5556) hiển thị ở giữa thanh tiêu đề (title bar) của cửa sổ chứa máy ảo.

Chạy ứng dụng trên điện thoại thứ 1 (số điện thoại là 5554) và gởi tin nhắn qua điện thoại ảo thứ 2 (số điện thoại là 5556) sẽ nhận được kết quả như hình sau:



Hình 4-7 AVD có số máy 5554 gửi tin nhắn có nội dung “Hello 5556” đến AVD có số máy 5556

#### ☛ Thực hiện:

- B1. Tạo 1 mới project. Trong quá trình tạo project, bạn cần đảm bảo đã chọn 2 mục *Target SDK* và *Compile With* với version của Android SDK là mới nhất để có thể sử dụng các APIs ở mức cao nhất có thể.
- B2. Xây dựng giao diện của ứng dụng bằng cách hiệu chỉnh nội dung file *res/layout/activity\_main.xml* để có nội dung như sau:

```
<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/btnSMSManager"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:onClick="UsingSMSManager"
 android:text="Using SMSManager to send SMS"/>
 <Button android:id="@+id/btnBuiltInIntent"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:onClick="UsingBuiltInContent"
 android:text="Using Built In Intent to send SMS"/>
</LinearLayout>
```

- B3. Tạo mới file *res/layout/smsmanager.xml* làm giao diện của activity thứ 2 của ứng dụng để gửi tin nhắn bằng SMSManager với nội dung như sau:

```
<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="wrap_content"
 android:layout_height="wrap_content"
 android:layout_gravity="center"
 android:textSize="20sp"
 android:text="Using SmsManager to send SMS" />
 <TextView android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_gravity="center"
```

```

 android:text="Enter Phone Number:" />
<EditText android:id="@+id/editTextPhoneNo"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:inputType="phone"/>
<TextView android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Enter SMS Message:" />
<EditText android:id="@+id/editTextSMS"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:inputType="textMultiLine"/>
<Button android:id="@+id/btnSendSMS"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="Send SMS"/>
</LinearLayout>

```

- B4. Bổ sung mã lệnh cho file *src/com/example/app\_17/MainActivity.java* để khi người dùng click button “Using SmsManager to send SMS” sẽ mở activity thứ 2

```

package com.example.app_17;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;

public class MainActivity extends Activity
{
 @Override
 protected void onCreate(Bundle savedInstanceState)
 {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 }
 public void UsingSmsManager(View v)
 {
 Intent intent = new Intent(this, UsingSmsManager.class);
 startActivity(intent);
 }
 /* Sau dòng này là nội dung cài đặt sẵn có của 2 phương thức onCreateOptionsMenu và
 onOptionsItemSelected */
}

```

- B5. Tạo mới file *src/com/example/app\_17/UsingSmsManager.java* cho activity thứ 2 để quản lý việc gửi tin nhắn bằng SmsManager:

```

package com.example.app_17;
import android.os.Bundle;
import android.app.Activity;
import android.telephony.SmsManager;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class UsingSmsManager extends Activity
{
 Button sendBtn;
 EditText txtphoneNo;
 EditText txtMessage;

 @Override

```

```

protected void onCreate(Bundle savedInstanceState)
{
 super.onCreate(savedInstanceState);
 setContentView(R.layout.smsmanager);
 sendBtn = (Button) findViewById(R.id.btnSendSMS);
 txtphoneNo = (EditText) findViewById(R.id.editTextPhoneNo);
 txtMessage = (EditText) findViewById(R.id.editTextSMS);
 sendBtn.setOnClickListener(new View.OnClickListener()
 {
 public void onClick(View view)
 {
 sendSMSMessage();
 }
 });
}
protected void sendSMSMessage()
{
 Log.i("Send SMS", "");
 String phoneNo = txtphoneNo.getText().toString();
 String message = txtMessage.getText().toString();
 try
 {
 SmsManager smsManager = SmsManager.getDefault();
 smsManager.sendTextMessage(phoneNo, null, message, null, null);
 Toast.makeText(getApplicationContext(), "SMS sent.",
 Toast.LENGTH_LONG).show();
 }
 catch (Exception e)
 {
 Toast.makeText(getApplicationContext(), "SMS failed, please try again.",
 Toast.LENGTH_LONG).show();
 e.printStackTrace();
 }
}
}

```

- B6. Bổ sung khai báo activity thứ 2 và quyền gửi (và nhận nếu ứng dụng có chức năng đó) trong file *AndroidManifest.xml*

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.app_17"
 android:versionCode="1"
 android:versionName="1.0" >
 <uses-sdk
 android:minSdkVersion="12"
 android:targetSdkVersion="21" />
 <uses-permission android:name="android.permission.SEND_SMS" />
 <!-- nếu ứng dụng cần nhận SMS thì thêm quyền RECEIVE_SMS -->
 <uses-permission android:name="android.permission.RECEIVE_SMS" />

 <application
 android:allowBackup="true"
 android:icon="@drawable/ic_launcher"
 android:label="@string/app_name"
 android:theme="@style/AppTheme" >
 <activity
 android:name=".MainActivity"
 android:label="@string/app_name" >
 <intent-filter>
 <action android:name="android.intent.action.MAIN" />
 <category android:name="android.intent.category.LAUNCHER" />
 </intent-filter>
 </activity>
 <activity
 android:name=".UsingSmsManager"
 android:label="Using SmsManager to send SMS" >
 </activity>
 </application>
</manifest>

```

- B7. Kết nối thiết bị di động Android bạn đang có với máy tính (hoặc khởi động 2 máy ảo) rồi chạy ứng dụng để xem kết quả.

Trong ứng dụng App\_17, có thể gửi cùng 1 SMS cho nhiều số điện thoại bằng cách dùng dấu chấm phẩy (;) để phân cách giữa các số điện thoại. Trong mã lệnh, bạn sử dụng một vòng lặp để gửi tin nhắn đến tất cả các số cho trước.

#### 4.2.1.1.2. Sử dụng Intent sẵn có để gửi SMS

- Khởi tạo 1 intent phục vụ việc gửi SMS

Bạn sẽ sử dụng action ACTION\_VIEW để khởi động một ứng dụng gửi tin nhắn được cài đặt trên thiết bị Android của bạn. Sau đây là cú pháp đơn giản để tạo ra một intent với action ACTION\_VIEW

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

- Thiết lập dữ liệu và kiểu dữ liệu để gửi SMS

- Dữ liệu: bạn cần phải xác định **smsto**: như URI bằng cách sử dụng phương thức SetData()
- Kiểu dữ liệu: sử dụng phương thức setType () để thiết lập kiểu dữ liệu là **vnd.android-dir/mms-sms**:

```
smsIntent.setData(Uri.parse("smsto:"));
smsIntent.setType("vnd.android-dir/mms-sms");
```

- Intent Object - Extra to send SMS

Android cung cấp sẵn phương thức putExtra để người lập trình có thể thêm số điện thoại và tin nhắn văn bản để gửi 1 SMS:

```
smsIntent.putExtra("address", new String("0123456789;3393993300"));
smsIntent.putExtra("sms_send", "Test SMS to somebody");
```

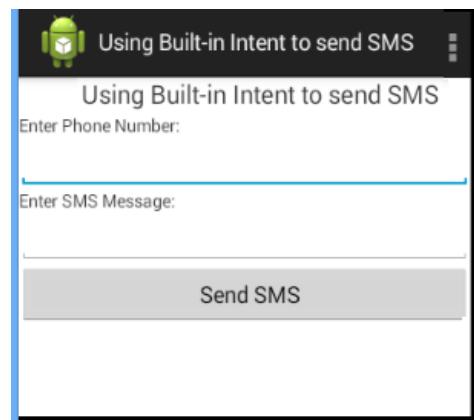
Trong đó **address** và **sms\_send** có phân biệt chữ hoa/thường và được quy định là chỉ sử dụng các ký tự thường. Có thể có nhiều hơn 1 số điện thoại nhưng phải phân cách nhau bởi dấu chấm phẩy (;).

### BÀI THỰC HÀNH App\_17 (bổ sung lần 1)

#### ☞ Yêu cầu:

Trong App\_17, tạo thêm 1 activity để khi nhấn button “Using Built-in Intent to send SMS” sẽ xuất hiện màn hình tương ứng. Trên activity thứ 3 này sẽ sử dụng *Built-in Intent* để gửi 1 SMS đến 1 số điện thoại có thực. Giao diện của activity thứ 3 như hình bên

Để thử nghiệm với project này, bạn cũng cần có thiết bị di động thực sự và được kết nối trực tiếp với máy tính như đã nói ở phần trước.



Hình 4-8 Sử dụng Intent để gửi SMS

#### ☞ Thực hiện:

- B8. Tạo thêm 1 file layout mới *builtin\_intent.xml* phục vụ cho activity thứ 3 với nội dung:

```
<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="wrap_content"
 android:layout_height="wrap_content"
 android:layout_gravity="center"
 android:textSize="20sp"
 android:text="Using Built-in Intent to send SMS" />
```

```

<TextView android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Enter Phone Number:" />
<EditText android:id="@+id/etPhoneNo"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:inputType="phone"/>
<TextView android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Enter SMS Message:" />
<EditText android:id="@+id/etSMS"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:inputType="textMultiLine"/>
<Button android:id="@+id/sendSMS"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:onClick="sendSMS"
 android:text="Send SMS"/>
</LinearLayout>

```

B9. Tạo mới file *UsingBuiltInIntent.java* để thực hiện các xử lý cho activity thứ 3, với nội dung:

```

package com.example.app_17;
import android.os.Bundle;
import android.app.Activity;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import android.net.Uri;
import android.content.Intent;
public class UsingBuiltInIntent extends Activity
{
 EditText etPhoneNumber, etSmsBody;
 @Override
 protected void onCreate(Bundle savedInstanceState)
 {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.builtin_intent);
 etPhoneNumber=(EditText)findViewById(R.id.etPhoneNo);
 etSmsBody=(EditText)findViewById(R.id.etSMS);
 }
 public void sendSMS(View v)
 {
 String sNo=etPhoneNumber.getText().toString().trim();
 String ssMS=etSmsBody.getText().toString().trim();
 if ((sNo.length()==0)|| (ssMS.length()==0))
 Toast.makeText(this,"Chưa nhập đủ thông tin", Toast.LENGTH_SHORT).show();
 else
 {
 Log.i("Send SMS", "");
 Intent smsIntent = new Intent(Intent.ACTION_VIEW);
 smsIntent.setData(Uri.parse("smsto:"));
 smsIntent.setType("vnd.android-dir/mms-sms");
 smsIntent.putExtra("address", sNo);
 smsIntent.putExtra("sms_body", ssMS);
 try
 {
 startActivity(smsIntent);
 finish();
 Log.i("Finished sending SMS...", "");
 }
 }
 }
}

```

```

 }
 catch (android.content.ActivityNotFoundException ex)
 {
 Toast.makeText(this,"SMS faild, please try again later.",
 Toast.LENGTH_SHORT).show();
 }
 }
}

```

**B10.** Bổ sung lệnh cho phương thức trong file *ActivityMain.java* để gọi activity thứ 3

```

package com.example.app_17;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
public class MainActivity extends Activity
{
 @Override
 protected void onCreate(Bundle savedInstanceState)
 {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 }
 public void UsingSMSManager(View v)
 {
 Intent intent = new Intent(this, UsingSmsManager.class);
 startActivity(intent);
 }
 public void UsingBuiltInIntent(View v)
 {
 Intent intent = new Intent(this, UsingBuiltInIntent.class);
 startActivity(intent);
 }
 /* Sau dòng này là nội dung cài đặt sẵn có của 2 phương thức onCreateOptionsMenu và
 onOptionsItemSelected */
}

```

**B11.** Bổ sung khai báo activity thứ 3 vào file *AndroidManifest.xml*. Do sử dụng built-in intent nên nếu chỉ dùng cho phần này bạn có thể bỏ qua khai báo về permission. Nhưng đây là bài tập ghép từ nhiều nội dung khác nhau nên bạn vẫn giữ các khai báo permission và chỉ bổ sung khai báo activity thứ 3:

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.app_17"
 android:versionCode="1"
 android:versionName="1.0" >
 <uses-sdk
 android:minSdkVersion="12"
 android:targetSdkVersion="21" />
 <uses-permission android:name="android.permission.SEND_SMS" />
 <!-- nếu ứng dụng cần nhận SMS thì thêm quyền RECEIVE_SMS -->
 <uses-permission android:name="android.permission.RECEIVE_SMS" />

 <application
 android:allowBackup="true"
 android:icon="@drawable/ic_launcher"
 android:label="@string/app_name"
 android:theme="@style/AppTheme" >
 <activity
 android:name=".MainActivity"
 android:label="@string/app_name" >
 <intent-filter>
 <action android:name="android.intent.action.MAIN" />
 <category android:name="android.intent.category.LAUNCHER" />
 </intent-filter>
 </activity>
 <activity

```

```

 android:name=".UsingSmsManager"
 android:label="Using SmsManager to send SMS" >
 </activity>
 <activity
 android:name=".UsingBuiltInIntent"
 android:label="Using Built-in Intent to send SMS" >
 </activity>
</application>
</manifest>

```

B12. Chạy ứng dụng để kiểm tra kết quả thực hiện.

#### 4.2.1.2. Nhận phản hồi từ việc gửi tin nhắn

Trong phần trước, bạn đã thực hiện việc gửi tin nhắn từ máy thứ 1 sang máy thứ 2 bằng cách sử dụng class *SmsManager*; nhưng nếu không được quan sát máy thứ 2 (máy nhận) khi đó bạn không biết chắc rằng máy thứ 2 đã nhận được tin nhắn hay chưa? Để kiểm tra được việc này, bạn cần tạo ra 2 đối tượng *PendingIntent* để giám sát trạng thái của tiến trình gửi tin nhắn. Hai đối tượng *PendingIntent* vừa tạo ra sẽ được chuyển cho 2 đối số cuối cùng của phương thức *sendTextMessage()*. Việc bổ sung đoạn mã lệnh sau đây vào phương thức *sendSMSMessage()* sẽ giúp bạn giám sát trạng thái của việc gửi tin nhắn:

```

//---sends an SMSmessage to another device---
protected void sendSMSMessage()
{
 Log.i("Send SMS", "");
 String phoneNo = txtphoneNo.getText().toString();
 String message = txtMessage.getText().toString();
 try
 { // == Đoạn mã lệnh bổ sung để giám sát trạng thái của tiến trình gửi tin nhắn ==
 String SENT = "SMS_SENT";
 String DELIVERED = "SMS_DELIVERED";
 PendingIntent sentPI = PendingIntent.getBroadcast(this, 0, new Intent(SENT), 0);
 PendingIntent deliveredPI = PendingIntent.getBroadcast(this, 0,
 new Intent(DELIVERED), 0);
 //---when the SMS has been sent---
 registerReceiver(new BroadcastReceiver()
 { @Override
 public void onReceive(Context arg0, Intent arg1)
 {
 switch (getResultCode())
 {
 case Activity.RESULT_OK:
 Toast.makeText(getApplicationContext(), "SMS sent",
 Toast.LENGTH_SHORT).show();
 break;
 case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
 Toast.makeText(getApplicationContext(), "Generic failure",
 Toast.LENGTH_SHORT).show();
 break;
 case SmsManager.RESULT_ERROR_NO_SERVICE:
 Toast.makeText(getApplicationContext(), "No service",
 Toast.LENGTH_SHORT).show();
 break;
 case SmsManager.RESULT_ERROR_NULL_PDU:
 Toast.makeText(getApplicationContext(), "Null PDU",
 Toast.LENGTH_SHORT).show();
 break;
 case SmsManager.RESULT_ERROR_RADIO_OFF:
 Toast.makeText(getApplicationContext(), "Radio off",
 Toast.LENGTH_SHORT).show();
 break;
 }
 }
 });
 }
}

```

```

 }, new IntentFilter(SENT));
 //---when the SMS has been delivered---
registerReceiver(new BroadcastReceiver()
{ @Override
 public void onReceive(Context arg0, Intent arg1)
 {
 switch (getResultCode())
 {
 case Activity.RESULT_OK:
 Toast.makeText(getApplicationContext(), "SMS delivered",
 Toast.LENGTH_SHORT).show();
 break;
 case Activity.RESULT_CANCELED:
 Toast.makeText(getApplicationContext(), "SMS not delivered",
 Toast.LENGTH_SHORT).show();
 break;
 }
 }
}, new IntentFilter(DELIVERED));
//===== Gởi SMS =====
SmsManager smsManager = SmsManager.getDefault();
smsManager.sendTextMessage(phoneNo, null, message, null, null);
}
catch (Exception e)
{
 Toast.makeText(getApplicationContext(), "SMS failed, please try again.",
 Toast.LENGTH_LONG).show();
 e.printStackTrace();
}
}

```

Ở đây, bạn tạo ra 2 đối tượng *PendingIntent* rồi đăng ký vào 2 *BroadcastReceivers*.

Hai *BroadcastReceivers* lắng nghe các intent “SMS\_SENT” và “SMS\_DELIVERED”, chúng sẽ được hệ điều hành thông báo ngay khi tin nhắn được gửi đi và chuyển giao. Trong mỗi *BroadcastReceiver*, bạn thực hiện override phương thức *onReceive()* và nhận được kết quả thực hiện.

Hai đối tượng *PendingIntent* được truyền vào 2 đối số cuối của phương thức *sendTextMessage()*:

```
smsManager.sendTextMessage(phoneNo, null, message, sentPI, deliveredPI);
```

Nhờ bổ sung đoạn mã lệnh trên giúp bạn nắm được tình trạng của quá trình gửi và nhận tin nhắn 1 cách rõ ràng, chính xác hơn trước khi bổ sung mã lệnh vừa nêu.

Nếu sử dụng phương pháp này để gọi các ứng dụng Messaging, bạn không cần bổ sung việc cho phép (permission) SMS\_SEND trong file *AndroidManifest.xml*.

#### 4.2.1.3. Nhận tin nhắn

Bên cạnh việc gửi tin nhắn SMS từ các ứng dụng Android của bạn, bạn cũng có thể nhận được các tin nhắn SMS từ bên trong ứng dụng của bạn bằng cách sử dụng một đối tượng *BroadcastReceiver*. Điều này rất hữu ích khi bạn muốn ứng dụng của bạn thực hiện một hành động khi nhận được một tin nhắn SMS nhất định nào đó. Ví dụ, bạn có thể muốn theo dõi vị trí của điện thoại của bạn trong trường hợp nó bị mất hoặc bị đánh cắp. Trong trường hợp này, bạn có thể viết một ứng dụng tự động lắng nghe các tin nhắn SMS có chứa một số mã bí mật. Khi có tin nhắn nhận được, sau đó bạn có thể gửi một tin nhắn SMS có chứa tọa độ của vị trí lại cho người gửi.

Một điểm thú vị của *BroadcastReceiver* là bạn có thể tiếp tục lắng nghe các tin nhắn SMS ngay cả khi ứng dụng không hoạt động (miễn là ứng dụng đã được cài đặt trên thiết bị).

#### 4.2.1.3.1. Trích xuất thông tin từ tin nhắn

Để lắng nghe các tin nhắn SMS, bạn cần tạo một class *BroadcastReceiver*. Class *BroadcastReceiver* cho phép ứng dụng của bạn nhận các intent được gửi bởi các ứng dụng khác bằng cách sử dụng phương thức *sendBroadcast()*. Khi nhận được 1 intent, phương thức *onReceive()* sẽ được gọi; do đó, bạn cần phải override phương thức này để thực hiện các yêu cầu của mình.

Tin nhắn SMS được chứa trong các đối tượng Intent (intent; tham số thứ hai trong phương thức *onReceive()*) thông qua một đối tượng Bundle. Các tin nhắn được lưu trữ trong một mảng đối tượng trong định dạng PDU. Để trích xuất mỗi tin nhắn, bạn sử dụng phương thức tĩnh *createFromPdu()* từ class *SmsMessage*. Các tin nhắn SMS sẽ được hiển thị bằng cách sử dụng class *Toast*. Số điện thoại của người gửi thu được thông qua phương thức *getOriginatingAddress()*, vì vậy bạn cần thực hiện autoreply (tự động trả lời) cho người gửi, nhờ vậy sẽ có được số điện thoại của người gửi.

### BÀI THỰC HÀNH App\_17 (bổ sung lần 2)

#### ☞ Yêu cầu:

Vẫn trong project App\_17. Yêu cầu khi nhận được tin nhắn, sử dụng class *Toast* để hiển thị thông tin về số điện thoại và nội dung tin nhắn như hình bên

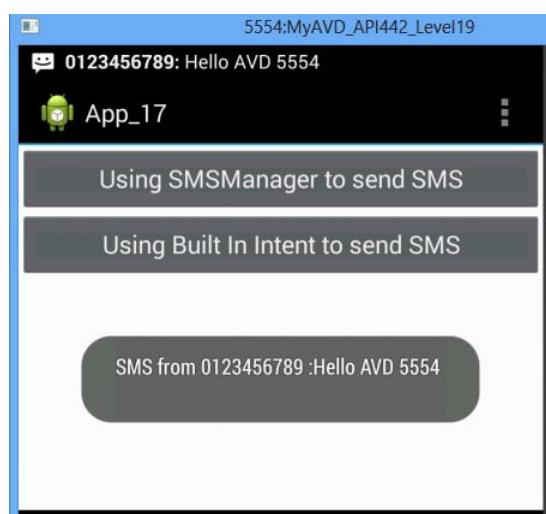
#### ☞ Thực hiện:

- B13.** Trở lại bài thực hành App\_17. Tạo mới file *src\com\example\app\_17\SMSReceiver.java* để tạo 1 class mở rộng từ *BroadcastReceiver* với nội dung như sau:

```
package com.example.app_17;
import android.content.BroadcastReceiver;
import android.content.Context;

import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
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 = "";
 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++)
 {
 /* trích xuất tin nhắn bằng cách sử dụng phương thức tĩnh createFromPdu()
 * từ class SmsMessage */
 msgs[i] = SmsMessage.createFromPdu((byte[]) pdus[i]);
 //Lấy số điện thoại của người gửi bằng phương thức getOriginatingAddress()
 }
 }
 }
}
```



Hình 4-9 Hiển thị số máy đã gửi và nội dung SMS

```

 str += "SMS from " + msgs[i].getOriginatingAddress();
 str += " :";
 str += msgs[i].getMessageBody().toString();
 str += "\n";
 }
 //---display the new SMS message--
 Toast.makeText(context, str, Toast.LENGTH_SHORT).show();
}
}
}
}

```

**B14.** Bổ sung tính năng receiver trong file AndroidManifest.xml để có nội dung như sau:

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.app_17"
 android:versionCode="1"
 android:versionName="1.0" >
 <uses-sdk
 android:minSdkVersion="12"
 android:targetSdkVersion="21" />
 <uses-permission android:name="android.permission.SEND_SMS" />
 <!-- nếu ứng dụng cần nhận SMS thì thêm quyền RECEIVE_SMS -->
 <uses-permission android:name="android.permission.RECEIVE_SMS" />
 <application
 android:allowBackup="true"
 android:icon="@drawable/ic_launcher"
 android:label="@string/app_name"
 android:theme="@style/AppTheme" >
 <activity
 android:name=".MainActivity"
 android:label="@string/app_name" >
 <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>
 <action android:name="android.provider.Telephony.SMS_RECEIVED" />
 </intent-filter>
 </receiver>
 <activity
 android:name=".UsingBuiltInIntent"
 android:label="Using Built-in Intent to send SMS" >
 </activity>
 <activity
 android:name=".UsingSmsManager"
 android:label="Using SmsManager to send SMS" >
 </activity>
 </application>
</manifest>

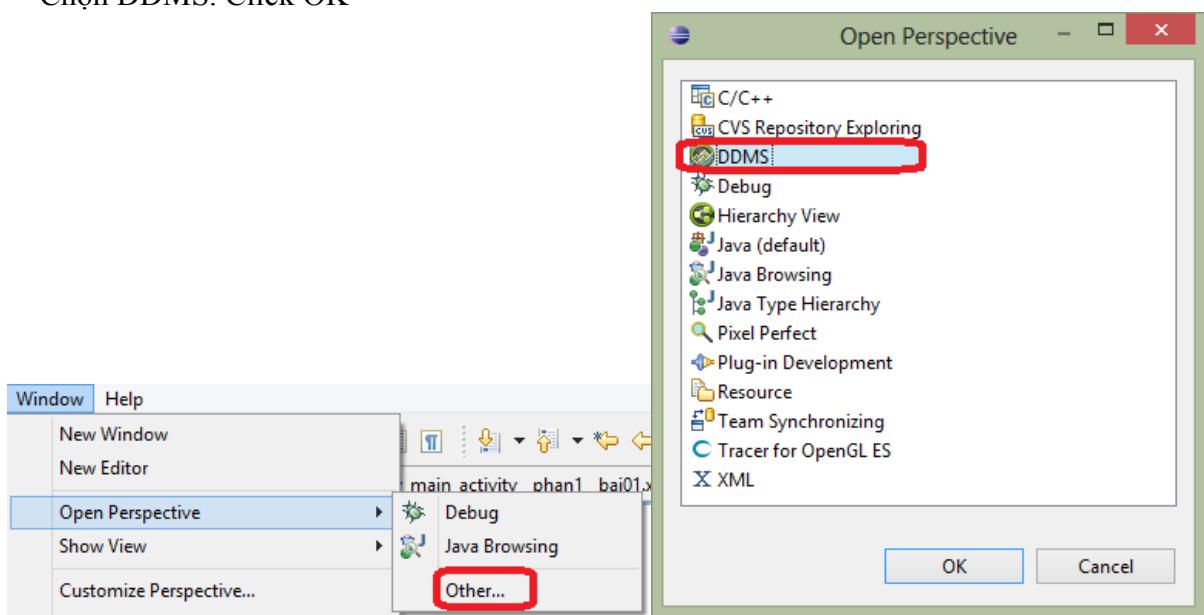
```

**B15.** Khởi chạy ứng dụng.

**B16.** Sử dụng DDMS để gửi tin nhắn đến gửi tin nhắn đến máy ảo để xem kết quả thực hiện.

Cách gửi tin nhắn – gọi điện thoại bằng AVD

- Chọn menu Windows/Open Perspective/ Others...
- Chọn DDMS. Click OK



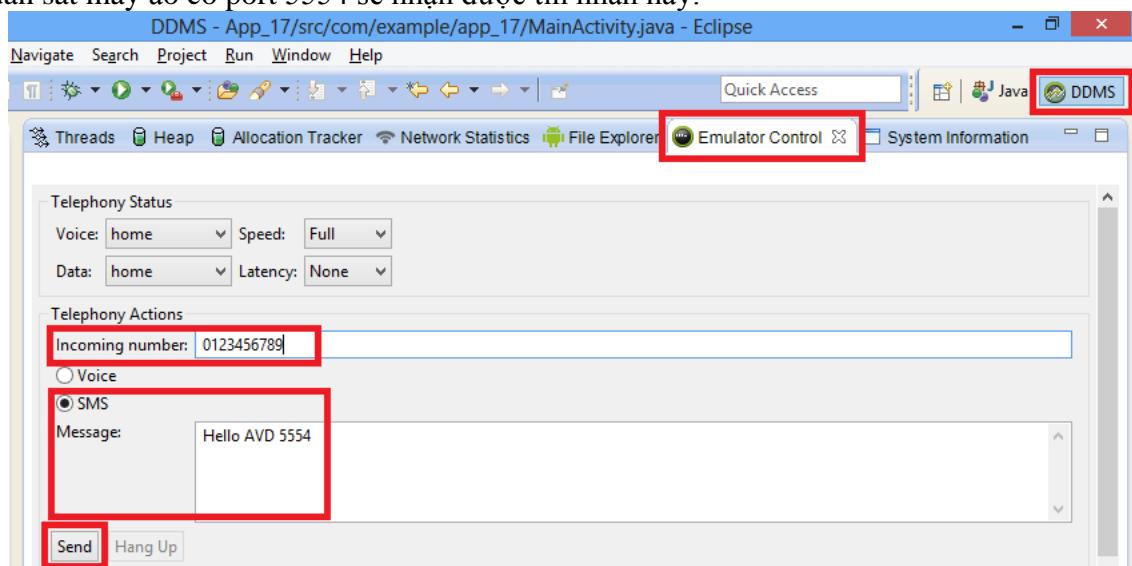
Hình 4-10 Mở cửa sổ DDMS từ menu Window

- Chọn tab Emulator Control

Trong tab *Emulator Control*, điền thông tin vào các mục:

- **Incoming Number**: số điện thoại của máy sẽ gọi (có thể gõ tạm 1 dãy số bất kỳ).
- **SMS**: click chọn để xác nhận đây là tin nhắn.
- **Message**: nhập vào “Hello”
- Click nút **Send**

Quan sát máy ảo có port 5554 sẽ nhận được tin nhắn này.



Hình 4-11 Chọn tab Emulator Control để gửi tin nhắn

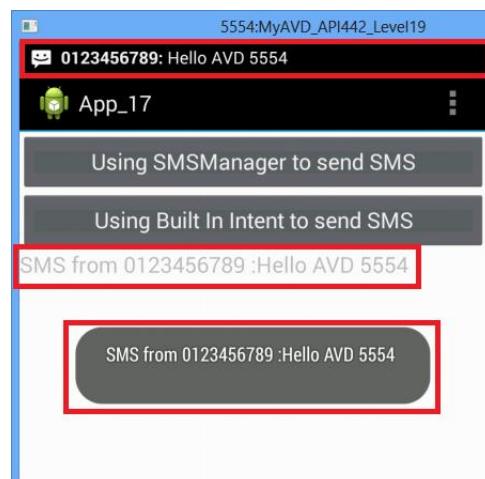
4.2.1.3.2. Cập nhật thông tin từ tin nhắn cho activity

Trong phần 4.2.1.3.1., bạn đã sử dụng một class *BroadcastReceiver* để lắng nghe các tin nhắn SMS và sau đó sử dụng class *Toast* để hiển thị các tin nhắn SMS đã nhận. Thông thường, bạn sẽ muốn gửi tin nhắn SMS trả lại hoạt động chính của ứng dụng của mình để xử lý 1 yêu cầu nào đó ví dụ như hiển thị tin nhắn trong một *TextView*.

## BÀI THỰC HÀNH App\_17 (bổ sung lần 3)

### Yêu cầu

Vẫn trong project App\_17 (đã bổ sung lần 2). Yêu cầu bổ sung 1 TextView vào dưới các button. Khi nhận được tin nhắn, thông tin về số điện thoại và nội dung tin nhắn sẽ được đưa vào TextView vừa thêm.



Hình 4-12 Kết quả thực hiện sau khi bổ sung lần 3

### Thực hiện

- B17.** Mở file *activity\_main.xml*. Bổ sung mã để tạo 1 TextView vào dưới button cuối cùng:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 android:orientation="vertical"
 android:background="#FFFFFF">
 <Button android:id="@+id/btnSMSManager"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:onClick="UsingSMSManager"
 android:text="Using SMSManager to send SMS"/>
 <Button android:id="@+id/btnBuiltInIntent"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:onClick="UsingBuiltInIntent"
 android:text="Using Built In Intent to send SMS"/>
 <TextView android:id="@+id/textView1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text=""
 android:textSize="18sp" />
</LinearLayout>
```

- B18.** Mở file *SMSReceiver.java* để sửa đổi nội dung class *SMSReceiver* với mục đích khi nhận được một tin nhắn SMS, class này sẽ phát ra 1 đối tượng Intent để cho bất kỳ ứng dụng nào đang thực hiện lắng nghe tin nhắn có thể nhận được thông báo. Trong Intent này chứa thông tin về tin nhắn

```
package com.example.app_17;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
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 = "";
 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]);
 str += "SMS from " + msgs[i].getOriginatingAddress();
 str += " :";
 str += msgs[i].getMessageBody().toString();
 str += "\n";
}
//---display the new SMS message---
Toast.makeText(context, str, Toast.LENGTH_SHORT).show();

/*phát ra 1 đối tượng Intent để cho bất kỳ ứng dụng nào đang thực hiện lắng
nghe tin nhắn có thể nhận được thông báo*/
Intent broadcastIntent = new Intent();
broadcastIntent.setAction("SMS_RECEIVED_ACTION");
broadcastIntent.putExtra("sms", str);
context.sendBroadcast(broadcastIntent);
}
}
}

```

- B19. Mở file *MainActivity.java* để bổ sung 1 số yêu cầu:

- Tạo ra một đối tượng BroadcastReceiver trong activity để lắng nghe broadcast intent. Khi một broadcast intent được nhận, bạn cập nhật tin nhắn SMS trong TextView.
- Tạo một đối tượng IntentFilter để bạn có thể lắng nghe cho một mục đích cụ thể. Trong trường hợp này, mục đích là "SMS\_RECEIVED\_ACTION".
- Đăng ký Broadcast Receiver trong 2 sự kiện onResume và onPause () của activity. Nếu không thực hiện đăng ký này nội dung trong TextView sẽ không được cập nhật.

Sau khi hiệu chỉnh hoàn tất, nội dung file *MainActivity.java* sẽ như sau:

```

package com.example.app_17;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.IntentFilter;
import android.widget.TextView;

public class MainActivity extends Activity
{
 IntentFilter intentFilter;
 TextView tvSMS ;
 /*tạo ra một đối tượng BroadcastReceiver trong activity để lắng nghe broadcast
 intents*/
 private BroadcastReceiver intentReceiver = new BroadcastReceiver()
 {
 @Override
 public void onReceive(Context context, Intent intent)
 {
 //Khi một broadcast intent được nhận, ban cập nhật tin nhắn SMS trong TextView
 tvSMS = (TextView) findViewById(R.id.textView1) ;
 tvSMS.setText(intent.getExtras().getString("sms")) ;
 }
 };
 @Override
 protected void onCreate(Bundle savedInstanceState)
 {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 //---intent to filter for SMS messages received---
 }
}

```

```

 intentFilter = new IntentFilter();
 intentFilter.addAction("SMS_RECEIVED_ACTION");
 }
 //đăng ký BroadcastReceiver trong 2 sự kiện onResume và onPause () của activity
 @Override
 protected void onResume()
 {
 //---register the receiver---
 registerReceiver(intentReceiver, intentFilter);
 super.onResume();
 }
 @Override
 protected void onPause()
 {
 //---unregister the receiver---
 unregisterReceiver(intentReceiver);
 super.onPause();
 }
 public void UsingSMSManager(View v)
 {
 Intent intent = new Intent(this, UsingSmsManager.class);
 startActivity(intent);
 }
 public void UsingBuiltInIntent(View v)
 {
 Intent intent = new Intent(this, UsingBuiltInIntent.class);
 startActivity(intent);
 }
 @Override
 public boolean onCreateOptionsMenu(Menu menu)
 {
 // Inflate the menu; this adds items to the action bar if it is present.
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }
 @Override
 public boolean onOptionsItemSelected(MenuItem item)
 {
 int id = item.getItemId();
 if (id == R.id.action_settings) {
 return true;
 }
 return super.onOptionsItemSelected(item);
 }
}

```

**B20.** Chạy ứng dụng. Sử dụng DDMS gửi tin nhắn để xem kết quả thực hiện.

**Lưu ý:** *TextView sẽ hiển thị các tin nhắn SMS chỉ khi tin nhắn gửi đến trong lúc activity có thể nhìn thấy trên màn hình. Nếu các tin nhắn SMS nhận được khi activity không ở trạng thái foreground, TextView sẽ không được cập nhật.*

#### 4.2.1.3.3. Chuyển activity đang hoạt động ở chế độ background sang chế độ foreground từ một BroadcastReceiver

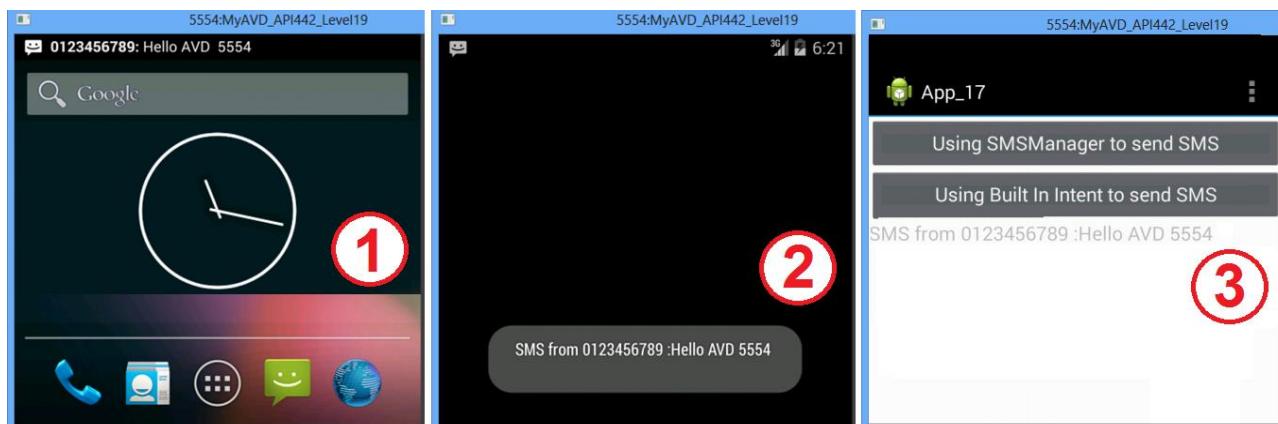
- Sau khi bỏ sung hoàn tất project lần thứ 3, ứng dụng của bạn đã có thể nhận được thông tin từ tin nhắn gửi đến. Tuy nhiên, trong nhiều tình huống activity của bạn có thể nhận được tin nhắn SMS khi đang hoạt động ở chế độ nền (background). Để khắc phục trường hợp này, bạn cần sử dụng đến BroadcastReceiver.
- **Lưu ý:** cần phân biệt rõ 2 tình huống sau: khi activity đang ở chế độ foreground (ứng dụng có thể nhìn thấy được trên màn hình), sau đó bạn nhấn vào nút:
  - *Home* (để hiển thị màn hình chủ nhằm thực hiện 1 công việc khác): khi đó activity này được đưa đến background. TextView có trong activity được cập nhật với các tin nhắn SMS nhận được.

- *Back*: hủy (destroy) activity đang thực hiện. Khi nhận được tin nhắn, activity này được đưa ra nhưng TextView không được cập nhật.
- Để chuyển activity đang hoạt động ở chế độ background sang chế độ foreground, bạn cần bổ sung 1 số việc như sau:
  - Trong class *MainActivity*:
    - Chuyển đăng ký *BroadcastReceiver* trong sự kiện *onCreate()*, thay vì trong sự kiện *onResume()*
    - Chuyển hủy đăng ký của *BroadcastReceiver* trong sự kiện *onPause()* sang hủy đăng ký trong sự kiện *onDestroy()*.
  - Điều này đảm bảo rằng ngay cả khi hoạt động này là ở chế độ nền, nó vẫn sẽ có thể lắng nghe broadcast intent.
- Trong class *SMSReceiver*: bổ sung vào sự kiện *onReceive()* một intent để đưa activity của mình lên foreground. Intent được đưa vào này cần 2 yếu tố:
  - Để chuyển 1 activity đang ở chế độ background sang foreground, bạn cần phải thiết lập các cờ *Intent.FLAG\_ACTIVITY\_NEW\_TASK*.
  - Sử dụng phương thức *startActivity()* để thực thi activity và mang activity ra nền trước (foreground) của màn hình.
- Trong file *AndroidManifest.xml*: bổ sung thiết lập cho thuộc tính *launchMode* của phần tử *<activity>* là *singleTask*. Nếu bạn không thực hiện thiết lập này, có thể có nhiều thẻ hiện của cùng activity sẽ được thực hiện khi nhận được tin nhắn SMS.

## BÀI THỰC HÀNH App\_17 (bổ sung lần 4)

### ⦿ Yêu cầu

Vẫn trong project App\_17 (đã bổ sung lần 3). Yêu cầu bổ sung mã lệnh để khi ứng dụng đang chạy ở chế độ background sẽ được chuyển sang foreground.



Hình 4-13 Các trạng thái của ứng dụng khi nhận tin nhắn

Hình 4-13-① Ứng dụng đang chạy ở background và tin nhắn gửi đến (phía trên cùng của màn hình)

Hình 4-13-② Toast giúp hiển thị thông báo

Hình 4-13-③ Ứng dụng được đưa sang chế độ foreground và nội dung tin nhắn hiển thị trong TextView

### ⦿ Thực hiện

**B21.** Mở file *MainActivity.java*. Bổ sung mã để chuyển các lệnh đăng ký và hủy đăng ký vào các sự kiện cho phù hợp với yêu cầu:

```
package com.example.app_17;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.content.BroadcastReceiver;
import android.content.Context;
```

```

import android.content.IntentFilter;
import android.widget.TextView;
public class MainActivity extends Activity
{ IntentFilter intentFilter;
 TextView tvSMS ;
 //tạo ra một đối tượng BroadcastReceiver trong activity để lắng nghe broadcast intents
 private BroadcastReceiver intentReceiver = new BroadcastReceiver()
 { @Override
 public void onReceive(Context context, Intent intent)
 { //Khi một broadcast intent được nhận, bạn cập nhật tin nhắn SMS trong TextView
 tvSMS = (TextView) findViewById(R.id.textView1) ;
 tvSMS.setText(intent.getStringExtra("sms")) ;
 }
 };
 @Override
 protected void onCreate(Bundle savedInstanceState)
 {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 //---intent to filter for SMS messages received---
 intentFilter = new IntentFilter();
 intentFilter.addAction("SMS_RECEIVED_ACTION");
 /*chuyển đăng ký BroadcastReceiver trong sự kiện onResume() sang onCreate() */
 registerReceiver(intentReceiver, intentFilter) ;
 }
 //đăng ký BroadcastReceiver trong 2 sự kiện onResume và onPause () của activity
 @Override
 protected void onResume()
 { //---register the receiver--
 // Do đã chuyển lệnh đăng ký này sang sự kiện onCreate() nên cần che lại
 //registerReceiver(intentReceiver, intentFilter) ;
 super.onResume();
 }
 @Override
 protected void onPause()
 { //---unregister the receiver--
 // Do đã chuyển lệnh hủy đăng ký này sang sự kiện onDestroy() nên cần che lại
 // unregisterReceiver(intentReceiver) ;
 super.onPause();
 }
 @Override
 protected void onDestroy()
 { //---unregister the receiver-
 /* chuyển hủy đăng ký BroadcastReceiver trong sự kiện onPause()
 * sang sự kiện onDestroy() */
 unregisterReceiver(intentReceiver) ;
 super.onPause();
 }
 public void UsingSMSManager(View v)
 { Intent intent = new Intent(this, UsingSmsManager.class);
 startActivity(intent);
 }
 public void UsingBuiltInIntent(View v)
 { Intent intent = new Intent(this, UsingBuiltInIntent.class);
 startActivity(intent);
 }
 @Override
 public boolean onCreateOptionsMenu(Menu menu)
 { getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }
 @Override
 public boolean onOptionsItemSelected(MenuItem item)
 { int id = item.getItemId();
 if (id == R.id.action_settings)
 { return true;
 }
 }

```

```

 return super.onOptionsItemSelected(item);
 }
}

```

- B22.** Mở file *SMSReceiver.java*. Bổ sung mã lệnh, khai báo một intent để đưa activity của bạn lên foreground:

```

package com.example.app_17;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
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 = "";
 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]);
 str += "SMS from " + msgs[i].getOriginatingAddress();
 str += " :";
 str += msgs[i].getMessageBody().toString();
 str += "\n";
 }
 //---display the new SMS message--
 Toast.makeText(context, str, Toast.LENGTH_SHORT).show();
 }
 }
}

```

```

//---launch the MainActivity---
Intent mainActivityIntent = new Intent(context, MainActivity.class);
/*thiết lập cờ Intent.FLAG_ACTIVITY_NEW_TASK bởi vì gọi startActivity()
 khi activity đang ở background */
mainActivityIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
/* Phương thức startActivity() cho thực thi activity và mang activity ra
 foreground */
context.startActivity(mainActivityIntent);

```

```

/*phát ra 1 đối tượng Intent để cho bất kỳ ứng dụng nào đang thực hiện lắng nghe tin
nhắn có thể nhận được thông báo*/
Intent broadcastIntent = new Intent();
broadcastIntent.setAction("SMS_RECEIVED_ACTION");
broadcastIntent.putExtra("sms", str);
context.sendBroadcast(broadcastIntent);
}
}
}

```

- B23.** Mở file *AndroidManifest.xml*. Bổ sung thiết lập cho thuộc tính *launchMode* của phần tử *<activity>* là *singleTask*:

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.app_17"
 android:versionCode="1"
 android:versionName="1.0" >
 <uses-sdk android:minSdkVersion="12"
 android:targetSdkVersion="21" />
 <uses-permission android:name="android.permission.SEND_SMS" />
 <!-- nếu ứng dụng cần nhận SMS thì thêm quyền RECEIVE_SMS -->
 <uses-permission android:name="android.permission.RECEIVE_SMS" />
 <application
 android:allowBackup="true"

```

```

 android:icon="@drawable/ic_launcher"
 android:label="@string/app_name"
 android:launchMode="singleTask" >
<activity android:name=".MainActivity"
 android:label="@string/app_name" >
 <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>
 <action android:name="android.provider.Telephony.SMS_RECEIVED" />
 </intent-filter>
</receiver>
<activity android:name=".UsingBuiltInIntent"
 android:label="Using Built-in Intent to send SMS" >
</activity>
<activity android:name=".UsingSmsManager"
 android:label="Using SmsManager to send SMS" >
</activity>
</application>
</manifest>

```

**B24.** Chạy ứng dụng. Sử dụng DDMS gửi tin nhắn để xem kết quả thực hiện.

#### 4.2.1.4. Cảnh báo

Khả năng gửi và nhận tin nhắn SMS làm cho Android trở thành một nền tảng rất hấp dẫn để phát triển các ứng dụng phức tạp. Tuy nhiên sự linh hoạt này đi kèm với một mức giá mà bạn cũng cần cân nhắc.

Một ứng dụng dường như vô tội có thể gửi tin nhắn SMS dựa trên Android Trojan SMS (tham khảo <http://forum.vodafone.co.nz/topic/5719-android-sms-trojan-warning/>) tự xung là một ứng dụng nghe nhạc, sau khi cài đặt, các ứng dụng đó sẽ sử dụng số điện thoại của bạn để gửi tin nhắn SMS đến một số dịch vụ nhằm thanh toán tiền phí bảo hiểm, thanh toán hóa đơn điện thoại, ...

Ngoài ra, ứng dụng cũng có thể "đánh hơi" (sniff) cho tin nhắn SMS. Ví dụ, dựa trên các kỹ thuật bạn vừa biết, bạn có thể dễ dàng viết một ứng dụng để kiểm tra các từ khóa nhất định trong tin nhắn SMS. Khi một tin nhắn SMS có chứa các từ khóa mà bạn đang tìm kiếm, sau đó bạn có thể sử dụng ứng dụng Quản lý địa điểm (*Location Manager*) để có được vị trí địa lý của bạn và sau đó gửi các tọa độ lại cho người gửi các tin nhắn SMS. Người gửi sau đó có thể dễ dàng theo dõi vị trí của bạn. Tất cả những công việc này có thể được thực hiện dễ dàng mà người dùng không hề biết! Điều đó nói rằng, người dùng nên cố gắng tránh cài đặt các ứng dụng Android đến từ các nguồn không rõ ràng, chẳng hạn như từ trang web không rõ, người lạ, ...

Về phần mình, khi tạo ra các ứng dụng, bạn cần minh bạch tất cả các yêu cầu của ứng dụng khi người dùng chuẩn bị cài đặt như: ứng dụng có tính phí, ứng dụng có quyền xác định vị trí địa lý của người dùng, ...



Hình 4-14 Cân minh bạch với người dùng về các yêu cầu của ứng dụng

#### 4.2.1.5. Làm việc với các folder chứa SMS

##### 4.2.1.5.1. Danh sách các folder chứa tin nhắn SMS và URI cho mỗi folder

- All : content://sms/all
- Inbox : content://sms/inbox
- Sent : content://sms/sent

- *Draft* : content://sms/draft
- *Outbox* : content://sms/outbox
- *Failed* : content://sms/failed
- *Queued* : content://sms/queued
- *Undelivered* : content://sms/undelivered
- *Conversations* : content://sms/conversations

#### 4.2.1.5.2. Các bước cần thực hiện

- Tùy vào chức năng định xây dựng của ứng dụng để cấp quyền Read/Write SMS trong file *AndroidManifest.xml*.

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

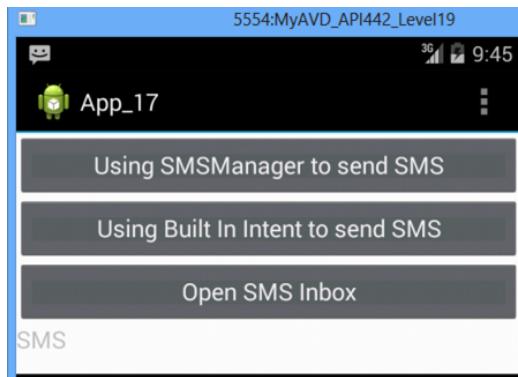
- Để liệt kê tin nhắn: thông thường tin nhắn được dùng kèm với control List, vì vậy bạn cần thực hiện:
  - Đối với file layout của activity, chỉ cần dùng 1 TextView với nhiệm vụ giữ chỗ cho mỗi mục (list item) có trong ListView.
  - Đối với file java: trong sự kiện *onCreate()*, thực hiện lần lượt các lệnh sau:
    - Tạo 1 URI chỉ đến tên folder cần sử dụng (ví dụ content://sms/inbox)
    - Thực thi 1 truy vấn trên URI.
    - Lọc ra nội dung tin nhắn để đưa vào ListView.
- Cách thực hiện trên áp dụng cho tất cả các folder khác (như Sent, Draft, ...) và chỉ cần thay đổi URI tương ứng.
- Android kết hợp MMS và SMS và cho phép bạn truy cập vào content providers để truy cập cả 2 cùng một lúc bằng cách sử dụng Authority là mms-sms. Nhờ vậy, bạn có thể truy cập vào một URI như sau:

```
content://mms-sms/conversations
```

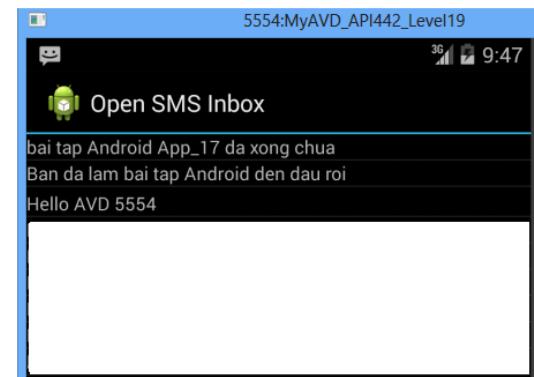
## BÀI THỰC HÀNH App\_17 (bổ sung lần 5)

### ☞ Yêu cầu

Vẫn trong project App\_17 (đã bổ sung lần 4). Yêu cầu bổ sung 1 button trên MainActivity. Khi button này được nhấn chọn sẽ hiển thị layout chứa tất cả các tin nhắn đã nhận được (inbox).



Hình 4-15 Bổ sung button “Open SMS Inbox” trên activity chính



Hình 4-16 AVD với 3 tin nhắn đã nhận

### ☞ Thực hiện

- B25.** Mở file *activity.xml*. Bổ sung mã để có button thứ 3:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 android:orientation="vertical"
 android:background="#FFFFFF">
 <Button
 android:id="@+id/btnSMSManager"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:onClick="UsingSMSManager"
```

```

<Button android:text="Using SMSManager to send SMS"/>
 android:id="@+id/btnBuiltInIntent"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:onClick="UsingBuiltInIntent"
 android:text="Using Built In Intent to send SMS"/>
<Button android:id="@+id/btnOpenSMSInbox"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:onClick="OpenSMSInbox"
 android:text="Open SMS Inbox"/>
<TextView android:id="@+id/textView1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="SMS"
 android:textSize="18sp" />
</LinearLayout>

```

- B26.** Tạo mới file layout *sms\_inbox.xml* để hiển thị nội dung các tin nhắn theo yêu cầu của đề bài.  
Như đã giới thiệu ở trên, TextView có trong layout này có nhiệm vụ giữ chỗ cho mỗi mục (list item) có trong ListActivity:

```

<?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/row"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"/>
</LinearLayout>

```

- B27.** Tạo mới file *SMS\_Inbox.java* và bổ sung mã lệnh để hiển thị nội dung các tin nhắn theo yêu cầu của đề bài.

```

package com.example.app_17;
import android.app.ListActivity;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.widgetListAdapter;
import android.widget.SimpleCursorAdapter;
public class SMS_Inbox extends ListActivity
{
 privateListAdapter adapter;
 private static final Uri SMS_INBOX = Uri.parse("content://sms/inbox");
 @SuppressWarnings("deprecation")
 @Override
 public void onCreate(Bundle bundle)
 {
 super.onCreate(bundle);
 Cursor cs = getContentResolver().query(SMS_INBOX, null, null, null, null);
 startManagingCursor(cs);
 String[] columns = new String[] { "body" };
 int[] names = new int[] { R.id.row };
 adapter = new SimpleCursorAdapter(this, R.layout.sms_inbox, cs, columns,
 names);
 setListAdapter(adapter);
 }
}

```

- B28.** Bổ sung mã lệnh trong file *MainActivity.java* để khi người dùng nhấp chọn button vừa thêm sẽ mở ra danh sách chứa nội dung các tin nhắn:

```

package com.example.app_17;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;

```

```

import android.view.MenuItem;
import android.view.View;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.IntentFilter;
import android.widget.TextView;
public class MainActivity extends Activity
{ IntentFilter intentFilter;
 TextView tvSMS ;
 //tạo ra một đối tượng BroadcastReceiver trong activity để lắng nghe broadcast intents
 private BroadcastReceiver intentReceiver = new BroadcastReceiver()
 { @Override
 public void onReceive(Context context, Intent intent)
 { //---display the SMS received in the TextView---
 //Khi một broadcast intent được nhận, bạn cập nhật tin nhắn SMS trong TextView
 tvSMS = (TextView) findViewById(R.id.textView1) ;
 tvSMS.setText(intent.getStringExtra("sms")) ;
 }
 };
 @Override
 protected void onCreate(Bundle savedInstanceState)
 { super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 //---intent to filter for SMS messages received---
 intentFilter = new IntentFilter();
 intentFilter.addAction("SMS_RECEIVED_ACTION");
 /* chuyển đăng ký BroadcastReceiver trong sự kiện onResume() sang sự kiện onCreate() */
 registerReceiver(intentReceiver, intentFilter) ;
 }
 //đăng ký BroadcastReceiver trong 2 sự kiện onResume và onPause () của activity
 @Override
 protected void onResume()
 { //---register the receiver--
 // Do đã chuyển lệnh đăng ký này sang sự kiện onCreate() nên cần che lại
 //registerReceiver(intentReceiver, intentFilter) ;
 super.onResume();
 }
 @Override
 protected void onPause()
 { //---unregister the receiver--
 // Do đã chuyển lệnh hủy đăng ký này sang sự kiện onDestroy() nên cần che lại
 // unregisterReceiver(intentReceiver) ;
 super.onPause();
 }
 @Override
 protected void onDestroy()
 { //---unregister the receiver-
 /* chuyển hủy đăng ký BroadcastReceiver trong sự kiện onPause() sang sự kiện onDestroy() */
 unregisterReceiver(intentReceiver) ;
 super.onPause();
 }
 public void UsingSMSManager(View v)
 { Intent intent = new Intent(this, UsingSmsManager.class);
 startActivity(intent);
 }
 public void UsingBuiltInIntent(View v)
 { Intent intent = new Intent(this, UsingBuiltInIntent.class);
 startActivity(intent);
 }
 public void OpenSMSInbox(View v)
 {
 Intent intent = new Intent(this, SMS_Inbox.class);
 startActivity(intent);
 }
 @Override
 public boolean onCreateOptionsMenu(Menu menu)
 {
 getMenuInflater().inflate(R.menu.main, menu);
 }
}

```

```

 return true;
 }
 @Override
 public boolean onOptionsItemSelected(MenuItem item) {
 int id = item.getItemId();
 if (id == R.id.action_settings) {
 return true;
 }
 return super.onOptionsItemSelected(item);
 }
}

```

- B29.** Bổ sung quyền đọc (và ghi nếu ứng dụng có cho phép) đối với tin nhắn đã nhận và đang ký activity SMS\_Inbox.

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.app_17"
 android:versionCode="1"
 android:versionName="1.0" >
 <uses-sdk android:minSdkVersion="12"
 android:targetSdkVersion="21" />
 <uses-permission android:name="android.permission.SEND_SMS" />
 <!-- nếu ứng dụng cần nhận SMS thì thêm quyền RECEIVE_SMS -->
 <uses-permission android:name="android.permission.RECEIVE_SMS" />
 <uses-permission android:name="android.permission.READ_SMS" />
 <!-- nếu ứng dụng cần cho xóa, thêm SMS thì thêm quyền WRITE_SMS -->
 <uses-permission android:name="android.permission.WRITE_SMS" />
 <application android:allowBackup="true"
 android:icon="@drawable/ic_launcher"
 android:label="@string/app_name"
 android:launchMode="singleTask" >
 <activity android:name=".MainActivity"
 android:label="@string/app_name" >
 <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>
 <action android:name="android.provider.Telephony.SMS_RECEIVED" />
 </intent-filter>
 </receiver>
 <activity android:name=".UsingBuiltInIntent"
 android:label="Using Built-in Intent to send SMS" >
 </activity>
 <activity android:name=".UsingSmsManager"
 android:label="Using SmsManager to send SMS" >
 </activity>
 <activity android:name=".SMS_Inbox"
 android:label="Open SMS Inbox" >
 </activity>
 </application>
</manifest>

```

- B30.** Trước khi chạy ứng dụng, bạn cần sử dụng DDMS để nháń 1 số tin đến AVD

- B31.** Chạy ứng dụng để xem kết quả

### 4.3. Sending e-mail

Giống như tin nhắn SMS, Android cũng hỗ trợ e-mail. Các ứng dụng Gmail/Email trên Android cho phép bạn cấu hình một tài khoản e-mail sử dụng POP3 hoặc IMAP. Bên cạnh việc gửi và nhận e-mail bằng cách sử dụng ứng dụng Gmail/Email, bạn cũng có thể lập trình từ bên trong ứng dụng Android của bạn để gửi tin e-mail.

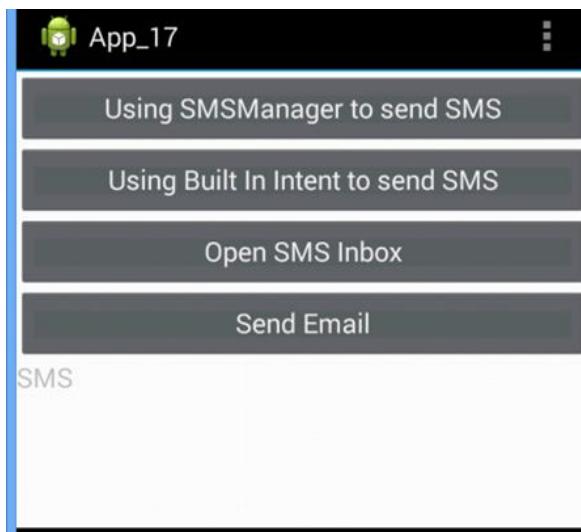
Thực ra, do Android không hỗ trợ gửi email trực tiếp từ ứng dụng nên ta sẽ phải tạo một đối tượng Intent để khởi động cửa sổ Activity dùng gửi mail

## BÀI THỰC HÀNH App\_17 (bổ sung lần 6)

### ☞ Yêu cầu

Vẫn trong project App\_17 (đã bổ sung lần 5). Yêu cầu bổ sung 1 button trên MainActivity. Khi button này được nhấn chọn sẽ mở activity cho phép gửi tin nhắn.

Sau khi người dùng điền đầy đủ thông tin và nhấn button Send Email, ứng dụng sẽ thực hiện việc chuyển nội dung email đến địa chỉ đã có.



Hình 4-17 Bổ sung button “Send Email” trên activity chính



Hình 4-18 Activity Send Email

### ☞ Thực hiện

- B32.** Tạo mới file layout *send\_email.xml* với 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="#FFFFFF">
 <TextView android:id="@+id/textView1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="SEND EMAIL"
 android:layout_gravity="center"
 android:textAppearance="?android:attr/textAppearanceLarge"
 android:textColor="#888888"
 android:textStyle="bold" />
 <LinearLayout android:orientation="horizontal"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content">
 <TextView android:id="@+id/tvTo"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="To (delimit by ';'):"/>
 android:textAppearance="?android:attr/textAppearanceMedium"
 android:textColor="#000000"/>
 <EditText android:id="@+id/etTo"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:ems="10"
 android:textColor="#000000">
 <requestFocus />
 </EditText>
 </LinearLayout>
</LinearLayout>
```

```

<LinearLayout android:orientation="horizontal"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content">
 <TextView android:id="@+id/tvCC"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="CC (delimit by ';'):"
 android:textAppearance="?android:attr/textAppearanceMedium"
 android:textColor="#000000"/>
 <EditText android:id="@+id/etCC"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:ems="10"
 android:textColor="#000000"/>
</LinearLayout>
<LinearLayout android:orientation="horizontal"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content">
 <TextView android:id="@+id/tvSubject"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Subject:"
 android:textAppearance="?android:attr/textAppearanceMedium"
 android:textColor="#000000"/>
 <EditText android:id="@+id/etSubject"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:ems="10"
 android:textColor="#000000"/>
</LinearLayout>
<EditText android:id="@+id/etBody"
 android:layout_width="match_parent"
 android:layout_height="150dp"
 android:ems="10"
 android:textColor="#000000"/>
<Button android:id="@+id/btnSendEmail"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="Send Email"
 android:onClick="Send"/>
</LinearLayout >

```

- B33. Tạo mới file `src\com\example\SendEmail.java` với nội dung như sau:

```

package com.example.app_17;
import android.app.Activity;
import android.os.Bundle;
import android.content.Intent;
import android.net.Uri;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
public class SendEmail extends Activity
{
 Button btnSendEmail;
 EditText etTo, etCC, etSubject, etBody;
 @Override
 public void onCreate(Bundle savedInstanceState)
 {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.send_email) ;
 etTo = (EditText) findViewById(R.id.etTo) ;
 etCC = (EditText) findViewById(R.id.etCC) ;
 }
}

```

```

 etSubject = (EditText) findViewById(R.id.etSubject) ;
 etBody = (EditText) findViewById(R.id.etBody) ;
 btnSendEmail = (Button) findViewById(R.id.btnSendEmail) ;
 }
 public void Send(View v)
 {
 String strTo=etTo.getText().toString();
 String strCC=etTo.getText().toString();

 String[] to = strTo.trim().split(";");
 String[] cc = strCC.trim().split(";");
 String subject = etSubject.getText().toString();
 String body = etBody.getText().toString();
 sendEmail(to, cc, subject, body) ;
 }
 //---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")) ;
 }
}

```

- B34. Bổ sung vào file *src\com\example\activity\_main.xml* mã lệnh tạo button cho button *Send Email*. Trong đó button mới thêm có sử dụng thuộc tính android:onClick để gọi hàm xử lý tên *Send\_Email* :

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 android:orientation="vertical"
 android:background="#FFFFFF">
 <Button android:id="@+id/btnSMSManager"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:onClick="UsingSMSManager"
 android:text="Using SMSManager to send SMS"/>
 <Button android:id="@+id/btnBuiltInIntent"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:onClick="UsingBuiltInIntent"
 android:text="Using Built In Intent to send SMS"/>
 <Button android:id="@+id/btnOpenSMSInbox"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:onClick="OpenSMSInbox"
 android:text="Open SMS Inbox"/>
 <Button android:id="@+id/btnSendEmail"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:onClick="Send_Email"
 android:text="Send Email"/>
 <TextView android:id="@+id/textView1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="SMS"
 android:textSize="18sp" />

```

&lt;/LinearLayout&gt;

B35. Bổ sung hàm xử lý sự kiện cho button vừa tạo trong file layout (*btnSendEmail*):

```

package com.example.app_17;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.IntentFilter;
import android.widget.TextView;

public class MainActivity extends Activity
{ IntentFilter intentFilter;
 TextView tvSMS ;
 //tạo ra một đối tượng BroadcastReceiver trong activity để lắng nghe broadcast intents
 private BroadcastReceiver intentReceiver = new BroadcastReceiver()
 { @Override
 public void onReceive(Context context, Intent intent)
 { //Khi một broadcast intent được nhận, bạn cập nhật tin nhắn SMS trong TextView
 tvSMS = (TextView) findViewById(R.id.textView1) ;
 tvSMS.setText(intent.getExtras().getString("sms")) ;
 }
 };
 @Override
 protected void onCreate(Bundle savedInstanceState)
 { super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 //---intent to filter for SMS messages received---
 intentFilter = new IntentFilter();
 intentFilter.addAction("SMS_RECEIVED_ACTION") ;
 /* chuyển đăng ký BroadcastReceiver trong sự kiện onResume() sang sự kiện onCreate() */
 registerReceiver(intentReceiver, intentFilter) ;
 }
 //đăng ký BroadcastReceiver trong 2 sự kiện onResume và onPause () của activity
 @Override
 protected void onResume()
 { //---register the receiver--
 // Do đã chuyển lệnh đăng ký này sang sự kiện onCreate() nên cần che lại
 //registerReceiver(intentReceiver, intentFilter) ;
 super.onResume();
 }
 @Override
 protected void onPause()
 { //---unregister the receiver--
 // Do đã chuyển lệnh hủy đăng ký này sang sự kiện onDestroy() nên cần che lại
 // unregisterReceiver(intentReceiver) ;
 super.onPause();
 }
 @Override
 protected void onDestroy()
 { //---unregister the receiver-
 /* chuyển hủy đăng ký BroadcastReceiver trong sự kiện onPause()
 * sang sự kiện onDestroy() */
 unregisterReceiver(intentReceiver) ;
 super.onPause();
 }
 public void UsingSMSManager(View v)
 { Intent intent = new Intent(this, UsingSmsManager.class);
 startActivity(intent);
 }
 public void UsingBuiltInIntent(View v)
 { Intent intent = new Intent(this, UsingBuiltInIntent.class);
 startActivity(intent);
 }
 public void OpenSMSInbox(View v)

```

```

 { Intent intent = new Intent(this, SMS_Inbox.class);
 startActivity(intent);
 }
 public void Send_Email(View v)
 {
 Intent intent = new Intent(this, SendEmail.class);
 startActivity(intent);
 }
 @Override
 public boolean onCreateOptionsMenu(Menu menu)
 {
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }
 @Override
 public boolean onOptionsItemSelected(MenuItem item)
 {
 int id = item.getItemId();
 if (id == R.id.action_settings)
 {
 return true;
 }
 return super.onOptionsItemSelected(item);
 }
}

```

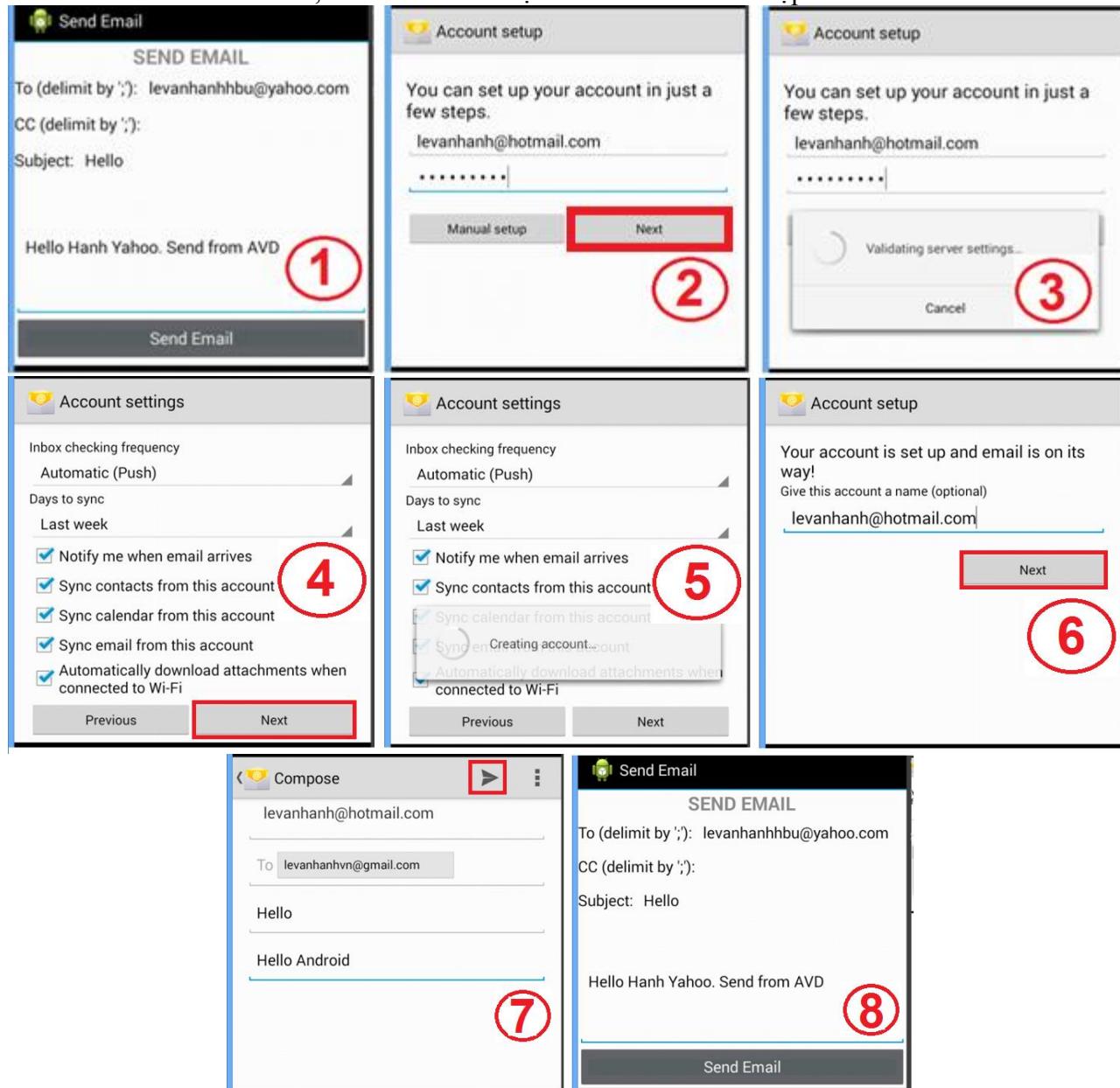
- B36. Bổ sung khai báo activity mới và quyền gửi SMS trong file AndroidManifest.xml. Do các lần bổ sung trước bạn đã bổ sung quyền được gửi email rồi (`<uses-permission android:name="android.permission.SEND_SMS" />`) nên ở đây chỉ thực hiện bổ sung activity mới mà thôi:

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.app_17"
 android:versionCode="1"
 android:versionName="1.0" >
 <uses-sdk android:minSdkVersion="12"
 android:targetSdkVersion="21" />
 <uses-permission android:name="android.permission.SEND_SMS" />
 <!-- nếu ứng dụng cần nhận SMS thì thêm quyền RECEIVE_SMS -->
 <uses-permission android:name="android.permission.RECEIVE_SMS" />
 <uses-permission android:name="android.permission.READ_SMS" />
 <uses-permission android:name="android.permission.WRITE_SMS" />
 <application android:allowBackup="true"
 android:icon="@drawable/ic_launcher"
 android:label="@string/app_name"
 android:launchMode="singleTask" >
 <activity android:name=".MainActivity"
 android:label="@string/app_name" >
 <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>
 <action android:name="android.provider.Telephony.SMS_RECEIVED" />
 </intent-filter>
 </receiver>
 <activity android:name=".UsingBuiltIntent"
 android:label="Using Built-in Intent to send SMS" >
 </activity>
 <activity android:name=".UsingSmsManager"
 android:label="Using SmsManager to send SMS" >
 </activity>
 <activity android:name=".SMS_Inbox"
 android:label="Open SMS Inbox" >
 </activity>
 <activity android:name=".SendEmail"
 android:label="Send Email" >
 </activity>
 </application>
</manifest>

```

B37. Chạy ứng dụng và thực hiện gửi email. Do mới thực hiện gửi email lần đầu nên sau khi nhấn button *Send Email*, Android sẽ đưa bạn đến màn hình thiết lập Account.



Hình 4-19 Các trạng thái của ứng dụng khi gửi email lần đầu

Hình 4-19-① Nhập địa chỉ và nội dung cần gửi rồi nhấn button *Send Email*

Hình 4-19-② Yêu cầu bổ sung thông tin về địa chỉ và password của người gửi để chứng thực

Hình 4-19-③ Quá trình chứng thực

Hình 4-19-④ Chứng thực thành công và yêu cầu 1 số nhiệm ý từ người dùng. Nhấn Next để tiếp tục

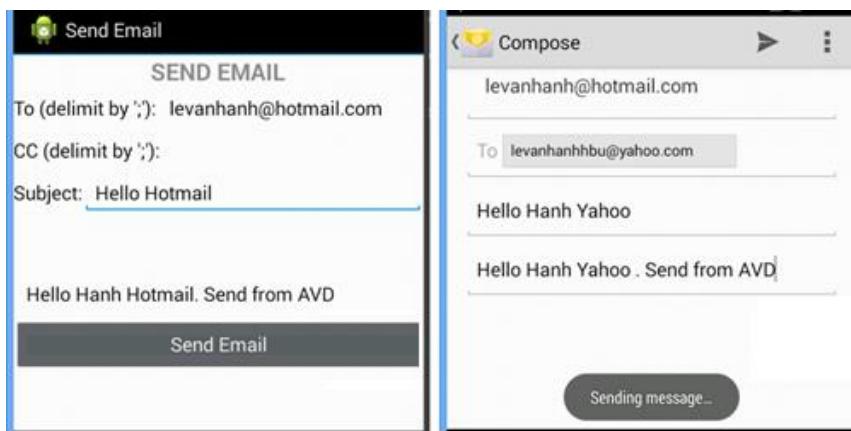
Hình 4-19-⑤ Quá trình ghi nhận lại các nhiệm ý

Hình 4-19-⑥ Xác nhận lại email của người gửi

Hình 4-19-⑦ Màn hình hiển thị toàn bộ thông tin về email sẽ gửi. Nhấn icon send (▶) để gửi

Hình 4-19-⑧ Sau khi gửi trả về màn hình ban đầu của activity *SendEmail* (chính là Hình 4-19-①)

Từ những lần gửi email sau, bạn chỉ còn phải chuyển qua lại giữa 2 màn hình sau:



Hình 4-20 Các trạng thái của ứng dụng khi gửi email những lần sau

## 4.4. BÀI TẬP TỔNG HỢP CHƯƠNG 4

### 4.4.1.

Thực hiện tương tự như yêu cầu của bài thực hành App\_16 nhưng chỉ sử dụng chung 1 activity cho cả ứng dụng.