ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG CAO ĐẲNG LÝ TỰ TRỌNG THÀNH PHỐ HỒ CHÍ MINH

GIÁO TRÌNH

MÔN HỌC/MÔ ĐUN: LẬP TRÌNH TRÊN THIẾT BỊ DI ĐỘNG

NGÀNH/NGHỀ:LẬP TRÌNH MÁY TÍNH

TRÌNH ĐỘ: CAO ĐẲNG

Ban hành kèm theo Quyết định số: 26A/QĐ-LTT-ĐT *ngày 08 tháng 01 năm 2020 của Hiệu trưởng Trường Cao đẳng Lý Tự Trọng Thành phố Hồ Chí Minh*

LƯU HÀNH NỘI BỘ

LỜI GIỚI THIỆU

Giáo trình môn lập trình lập trình trên thiết bị di động được biên soạn theo chương trình đào tạo chuyên ngành lập trình máy tính của Bộ Lao động Thương binh & Xã hội. Giáo trình này trình bày những vấn đề cơ bản và cốt lõi nhất của môn lập trình trên thiết bị di động. Các bài học trong giáo trình được trình bày ngắn gọn, có nhiều ví dụ minh hoạ. Cuối mỗi chương đều có bài tập để sinh viên luyện tập. Giáo trình này có thể giúp các sinh viên trong việc học môn lập trình trên thiết bị di động ở bậc cao đẳng. Chúng tôi mong rằng các sinh viên tự tìm hiểu trước mỗi vấn đề và kết hợp với bài giảng trên lớp của giáo viên để học môn này đạt hiệu quả hơn. Trong quá trình giảng dạy và biên soạn giáo trình này, chúng tôi đã nhận được sự động viên của các thầy trong Ban Giám Hiệu nhà trường cũng như những ý kiến của các đồng nghiệp trong Khoa Công nghệ thông tin. Chúng tôi xin chân thành cảm ơn và hy vọng rằng giáo trình này sẽ giúp cho việc dạy và học môn lập trình hướng đối tượng của nhà trường ngày càng tốt hơn.

Thành phố Hồ Chí Minh, ngày 07 tháng 01 năm 2020

Tham gia biên soạn

1. ThS Nguyễn Văn Danh
2. ThS Mai Chiếm Tuấn

MỤC LỤC

CHƯƠNG 1. TỔNG QUAN

1.TỔNG QUAN VỀ LẬP TRÌNH DI ĐỘNG

Android là một gói phần mềm và hệ điều hành dựa trên linux cho các thiết bị di động như máy tính bảng và điện thoại thông minh. Android được phát triển bởi Google và sau đó là OHA (Open Handset Alliance). Ngôn ngữ chủ yếu được sử dụng để viết mã Android là Java mặc dù các ngôn ngữ khác có thể được sử dụng để viết trên Android. Mục tiêu của android là tạo ra một sản phẩm trong thế giới thực để cải thiện trải nghiệm di động cho người dùng.

Có rất nhiều phiên bản android như Lollipop, Kitkat, Jelly Bean, Ice Cream Sandwich, Froyo, Ecliar, Donut v.v…

1.1 Lịch sử ra đời Android

Năm 2003, Android Inc. được thành lập bởi Andy Rubin, Rich Miner, Nick Sears và Chris White tại California.

Năm 2005, Google sở hữu Android cùng với các vị trí quản lý.

Năm 2007, OHA (Open Handset Alliance) được thành lập bởi Google cùng với nhiều nhà sản xuất thiết bị phần cứng, thiết bị không dây và vi xử lý. Công bố nền tảng phát triển Android.

Năm 2008, thiết bị HTC Dream là phiên bản thế hệ đầu tiên hoạt động với hệ điều hành Android 1.0.

Năm 2010, Google khởi đầu dòng thiết bị Nexus với thiết bị đầu tiên của HTC là Nexus One.

Năm 2013, ra mắt loạt thiết bị phiên bản GPE.

Năm 2014, Google công báo Android Wear, hệ điều hành dành cho các thiết bị đeo được.

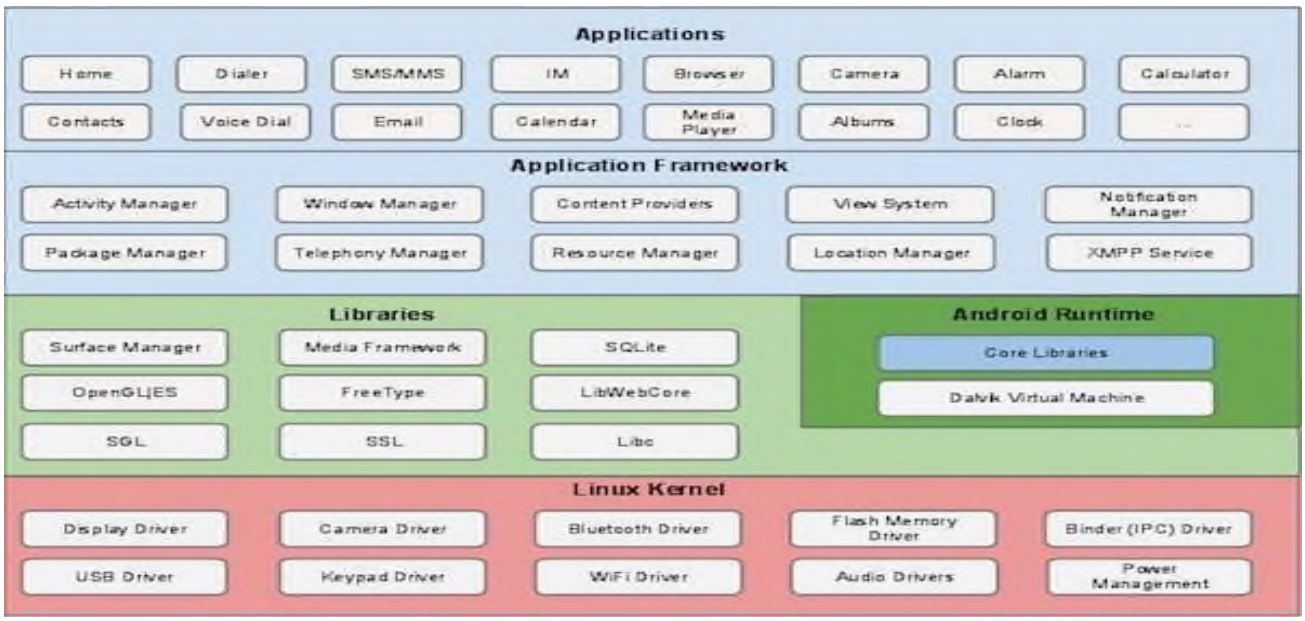
1.2 Các phiên bản của hệ điều hành Android:

Tên mã	Số phiên bản	Ngày công bố
	1.0	September 23, 2008
	1.1	February 9, 2009
Cupcake	1.5	April 27, 2009
Donut	1.6	September 15, 2009
Eclair	2.0–2.1	October 26, 2009
Froyo	2.2–2.2.3	May 20, 2010

Gingerbread	2.3–2.3.7	December 6, 2010
Honeycomb	3.0–3.2.6	February 22, 2011
Ice Cream Sandwich	4.0–4.0.4	October 18, 2011
Jelly Bean	4.1–4.3.1	July 9, 2012
KitKat	4.4–4.4.4, 4.4W–4.4W.2	October 31, 2013
Lollipop	5.0–5.1.1	November 12, 2014
Marshmallow	6.0–6.0.1	October 5, 2015
Nougat	**7.0 – 7.1.1**	August 22, 2016
Oreo	8.0 - 8.1	August 22, 2017
Pie	9.0	August 22, 2018
Android Q	10	September 3,2019
Android 11	11	May, 2020

1.3 Kiến trúc hệ điều hành Android

Hệ điều hành Android là 1 ngăn xếp các thành phần phần mềm, được chia thành 5 phần và 4 lớp chính như trong hình bên dưới.



Linux Kernel

Dưới cùng là lớp Linux - Linux 3.6 cùng với khoảng 115 bản vá. Lớp này cung cấp 1 cấp độ trừu tượng giữa phần cứng của thiết bị và các thành trình điều khiển phần cứng thiết yếu như máy ảnh, bàn phím, màn hình hiển thị... Đồng thời, hạt nhân (kernel) còn xử lý tất cả các thứ mà Linux có thể làm tốt như mạng kết nối và 1 chuỗi các trình điều khiển thiết bị, giúp cho giao tiếp với các thiết bị ngoại vi dễ dàng hơn.

Các thư viện

Ở trên lớp nhân Linux là tập các thư viện bao gồm WebKit - trình duyệt Web mã nguồn mở, được biết đến như thư viện libc, cơ sở dữ liệu SQLite - hữu dụng cho việc lưu trữ và chia sẻ dữ liệu ứng dụng, các thư viênj chơi và ghi âm audio, video, hay các thư viện SSL chịu trách nhiệm bảo mật Internet...

Các thư viện Android

Đây là các thư viện dựa trên Java phục vụ cho việc phát triển Android. Ví dụ của các thư viện này bao gồm các thư viện ứng dụng dùng để xây dựng giao diện người dùng, vẽ đồ họa hay truy cập cơ sở dữ liệu. 1 số thư viện chính của Android:

- android.app - Cung cấp quyền truy cập vào ứng dụng và là nền tảng của tất cả ứng dụng Android.
- android.content - Cung cấp quyền truy cập nội dung (content), truyền tải thông điệp giữa các ứng dụng hay các thành phần của ứng dụng.
- android.database - Đươc sử dụng để truy cập dữ liệu của content provider và cơ sở dữ liệu SQLite
- android.opengl - giao diện các phương thức Java để sử dụng OpenGL ES
- android.os - Cung cấp các ứng dụng với quyền truy cập vào các dịch vụ của hệ điều hành bao gồm thông điệp, các dịch vụ hệ thống và các giao tiếp nội tại
- android.text - Đươc sử dụng để hiển thị và điều chỉnh chữ trên màn hình thiết bị
- android.view - Các thành phần cơ bản trong việc xây dựng giao diện người dùng của ứng dụng.
- android.widget - Tập các thành phần giao diện người dùng đã được xây dựng sẵn như các nút, các nhãn (label), list view,....

- android.webkit - Tập các lớp cho phép xây dựng khả năng duyệt web.

Android Runtime

Đây là phần thứ 3 của kiến trúc và nằm ở lớp thứ 2 từ dưới lên. Phần này cung cấp 1 bộ phận quan trọng là Dalvik Vỉtual Machine - là 1 loại Java Virtual Machine được thiết kế đặc biệt để tối ưu cho Android.

Dalvik VM sử dụng các đặc trưng của nhân Linux như quản lý bộ nhớ và đa luồng, những thứ mà đã có sẵn trong Java. Dalvik VM giúp mọ ứng dụng Android chạy trong tiến trình riêng của nó, với các thể hiện (instance) riêng của Dalvik virtual Machine.

Android Runtime cũng cung cấp 1 tập các thư viện chính giúp các nhà phát triển ứng dụng Android có thể viết ứng dụng Android bằng Java

Application Framework

Lớp Android Framework cung cấp các dịch vụ cấp độ cao hơn cho các ứng dụng dưới dạng các lớp Java. Các nhà phát triển ứng dụng được phép sử dụng các dịch vụ này trong ứng dụng của họ.

Android Framework bao gồm các dịch vụ chính sau:

- Activitty Manager - Kiểm soát tất cả khía cạnh của vòng đời ứng dụng và ngăn xếp các Activity.
- Content Providers - Cho phép các ứng dụng chia sẽ dữ liệu với các ứng dụng khác.
- Resource Manager - Cung cấp quyền truy cập vào các tài nguyên như các chuỗi, màu sắc, các layout giao diện người dùng...
- Notifications Manager - Cho phép các ứng dụng hiển thị cảnh báo và các thông báo cho người dùng.
- View System - Tập các thành phần giao diện (view) được sử dụng để tạo giao diện người dùng.

Application

Lớp trên cùng của kiến trúc là Application. Các ứng dụng bạn tạo ra sẽ được cài đặt trên lớp này. Ví dụ như: Danh bạ, nhắn tin, trò chơi...

2.MÔI TRƯỜNG PHÁT TRIỂN ỨNG DỤNG DI ĐỘNG

Có nhiều công cụ để phát triển Android nhưng đến nay công cụ chính thức và mạnh mẽ nhất là Android Studio. Đây là IDE (Môi trường phát triển tích hợp) chính thức cho nền tảng Android, được phát triển bởi Google và được sử dụng để tạo phần lớn các ứng dụng mà bạn có thể sử dụng hàng ngày.

Android Studio lần đầu tiên được công bố tại hội nghị Google I/O vào năm 2013 và được phát hành cho công chúng vào năm 2014 sau nhiều phiên bản beta khác nhau. Trước khi được phát hành, các nhà phát triển Android thường sử dụng các công cụ như Eclipse IDE, một IDE Java chung cũng hỗ trợ nhiều ngôn ngữ lập trình khác.

Để lập trình ứng dụng trên nền tảng hệ điều hành Android chúng ta cần 3 công cụ gồm: JDK – Java Development Kit, Android Studio, máy ảo Genymotion hoặc Android Emulator được tích hợp trong Android Studio. Trong chương này chúng ta tìm hiểu về cài đặt Android Studio và thiết lập máy ảo để chạy Android Studio.

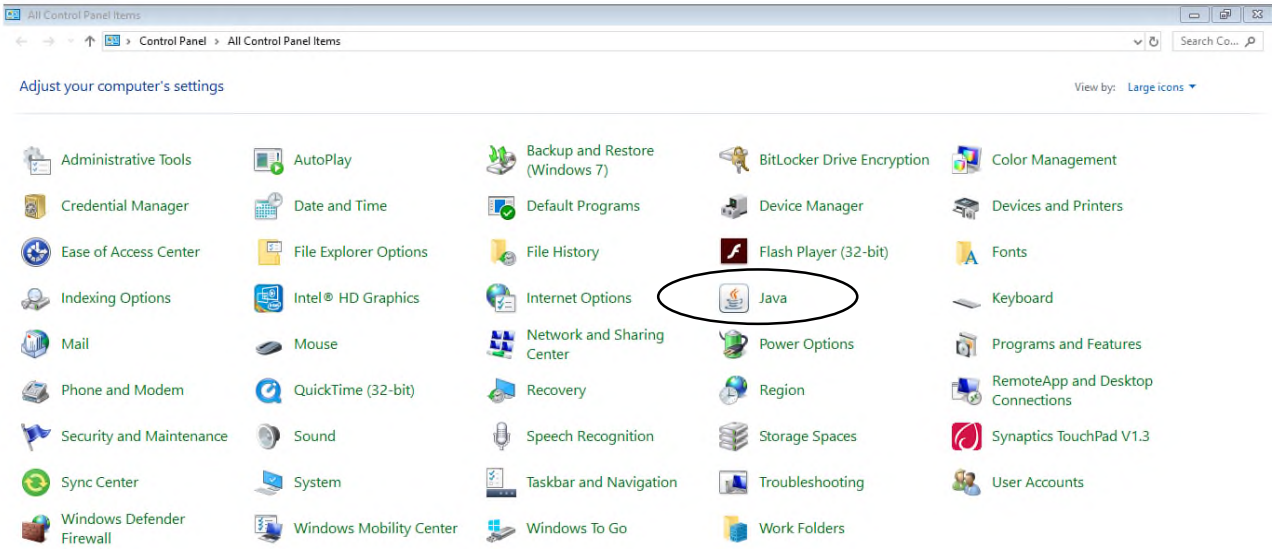
2.1 Hướng dẫn cài đặt JDK

JDK(Java Development Kit) là một tập hợp những công cụ phần mềm được phát triển bởi Sun Microsystems dành cho các nhà phát triển phần mềm. JDK dùng để viết những applet Java hay những ứng dụng Java hoặc ứng dụng Android. Bộ công cụ này được phát hành miễn phí gồm có trình biên dịch, trình thông dịch, trình giúp sửa lỗi (debugger, trình chạy applet) và tài liệu nghiên cứu. Để biên dịch được các source code Java, máy tính của chúng ta phải thiết lập môi trường để các ứng dụng Java có thể chạy.

Trước khi chúng ta cài đặt, chúng ta phải kiểm tra phiên bản của hệ điều hành hiện thời, sau đó chung ta vào liên kết để chọn JDK phù hợp.

https://www.oracle.com/java/technologies/javase-downloads.html

Sau khi tải xong phiên bản đúng với hệ điều hành của chung ta(chú ý là 32 bit hay 64 bit nhé). Tiến hành cài đặt bình thường và nếu thành công thì chúng ta kiểm tra lại bằng cách vào Control Panel kiểm tra. Nếu có biểu tượng Java là đã thành công.



2.2 Hướng dẫn cài đặt Android Studio

Android Studio là công cụ lập trình chuẩn riêng của Google thay thế cho phiên bản Eclipse cũ. Android Studio bao gồm các thành phần: *Android Studio IDE, Android SDK tools, Android 5.0 (Lollipop) Platform, Android 5.0 emulator system image with Google APIs*.

Android Studio là một công cụ lập trình thông minh có khả năng chỉnh sửa code tiên tiến với nhiều thay đổi như: công cụ thiết kế giao diện người dùng mới và trực quan, phân tích hiệu suất, vv cho phép lập trình viên có thể tạo các ứng dụng, thực hiện thay đổi cũng như xem trước sản phẩm trong thời gian thực.

Để tải Android Studio chúng ta có thể tìm kiếm google, hay vào link trực tiếp sau, phiên bản hiện nay là 4.1:

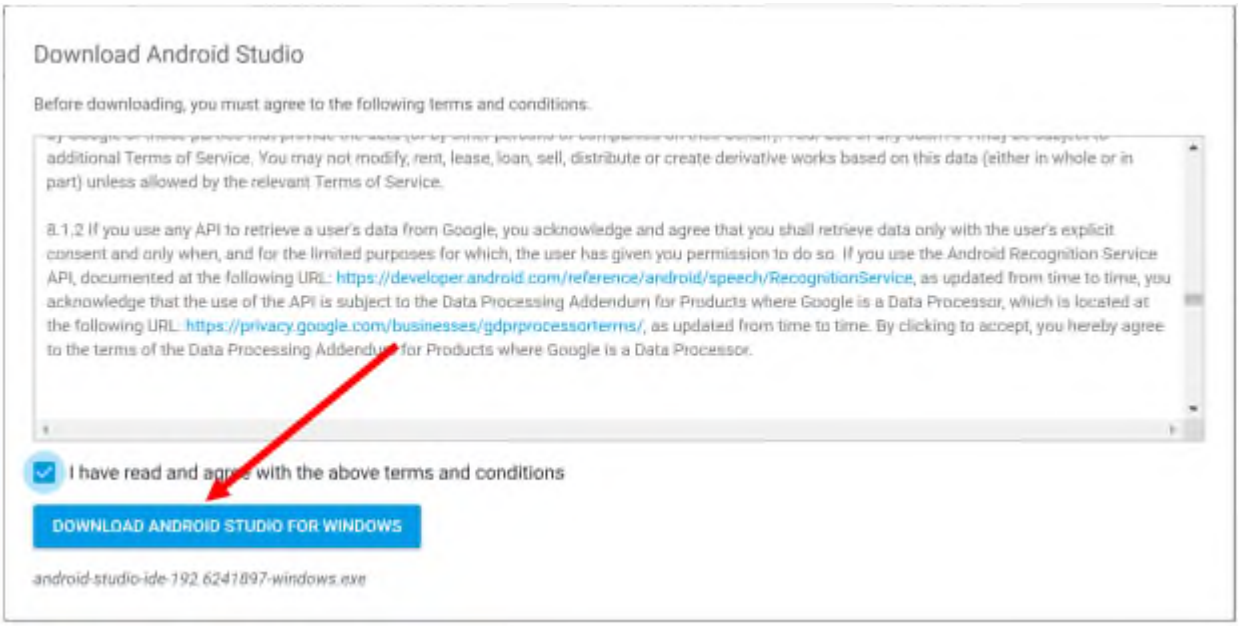
Lập trình được Android, laptop của bạn phải tối thiểu:

- RAM tối thiểu 8GB
- ổ cứng nhiều, ưu tiên SSD
- hỗ trợ ảo hóa
- win 10, 64

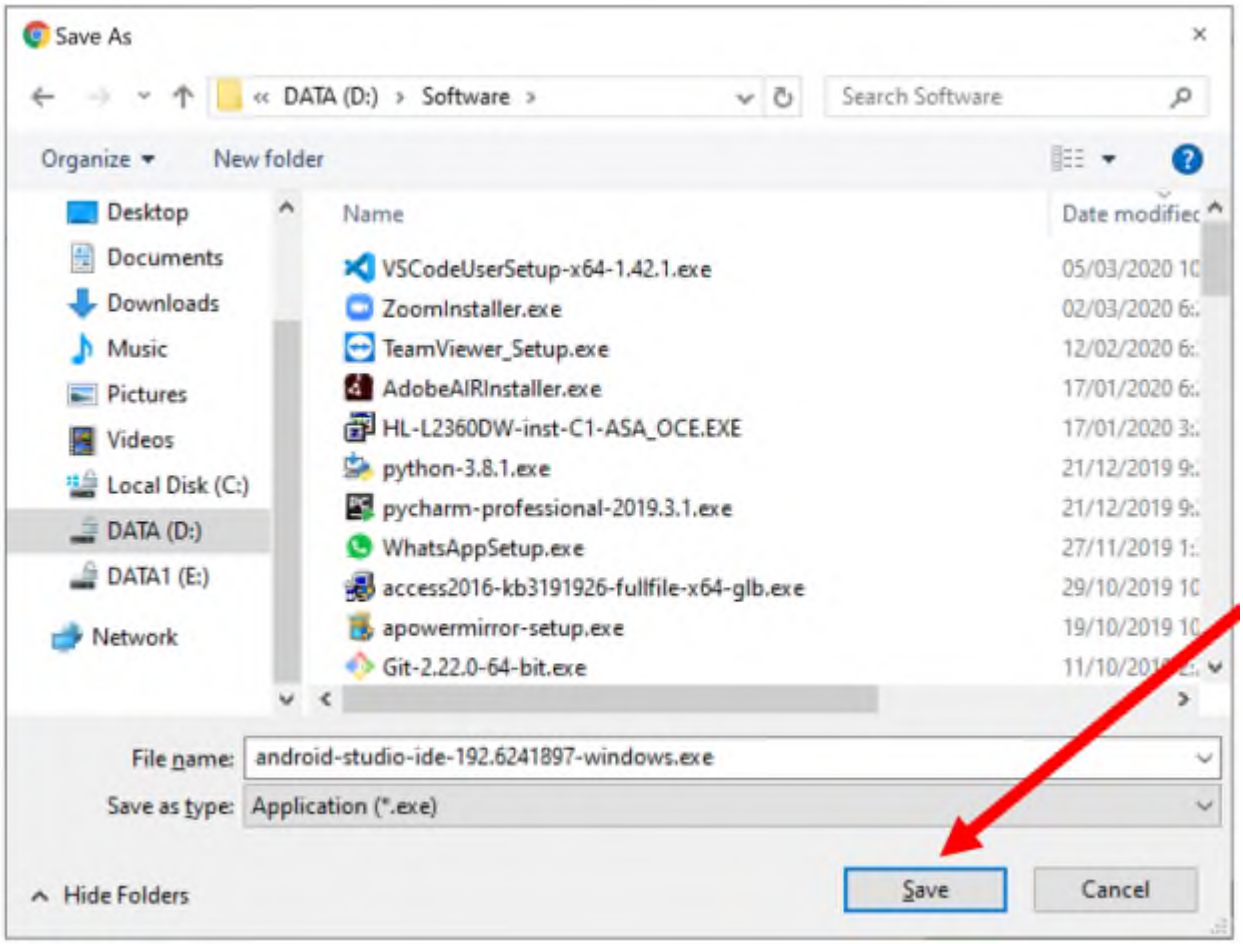
Bước trước tiên, ta cần tải Android Studio phiên bản mới nhất tại đây:

https://developer.android.com/studio

Nhấn vào "DOWNLOAD ANDROID STUDIO", ở trên thấy là 3.6.2 (và không quan trọng, có thể tải phiên bản mới nhất):



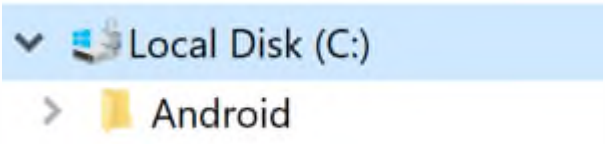
Màn hình trên hiện ra, check vào “I have read and agree with the above terms and conditions” Sau đó bấm “DOWNLOAD ANDROID STUDIO FOR WINDOWS”. Máy bạn là MAC , LINUX thì cũng tương tự.



Bấm save để tải về, tùy tốc độ mà lâu hay mau, đợi chỗ này nó tải cho xong.

Sau khi tải xong thì bắt đầu cài đặt.

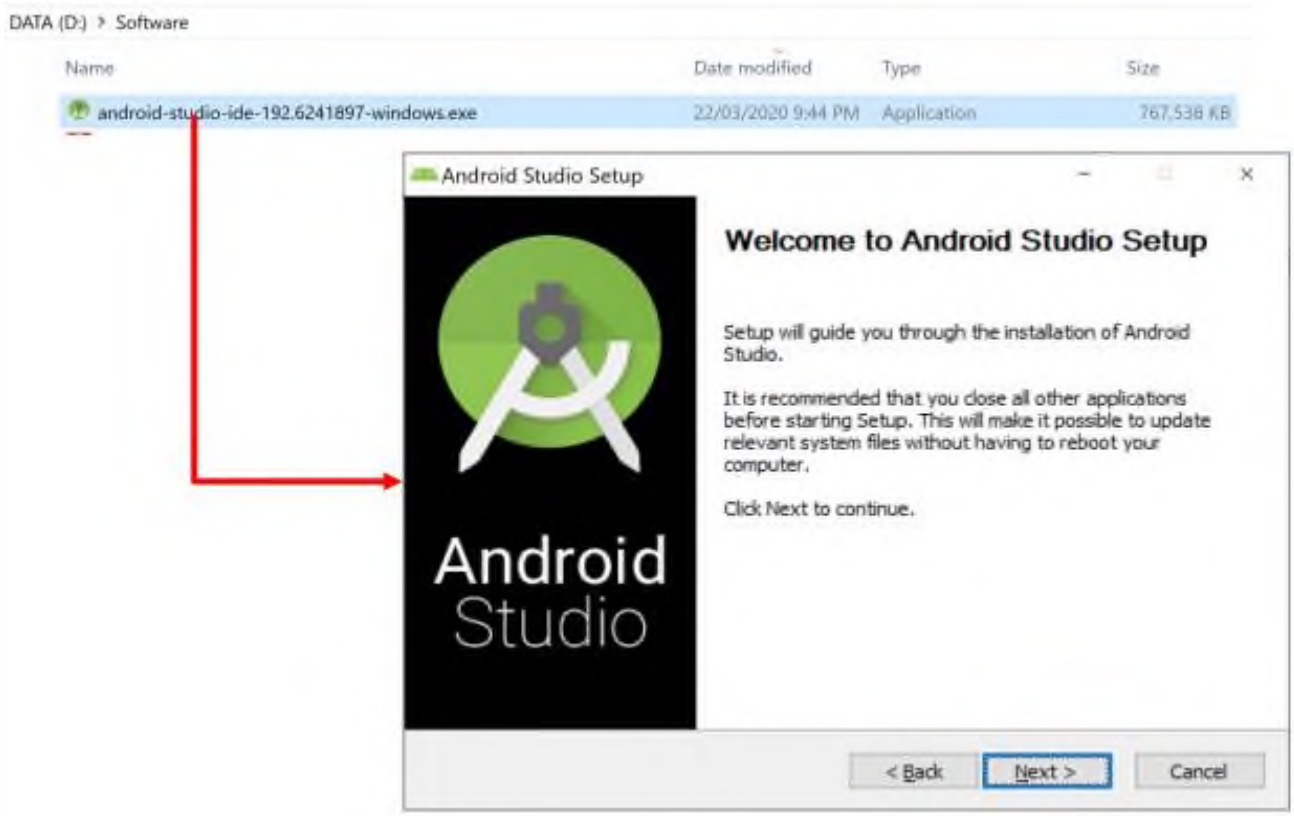
Trước khi cài đặt thì nên tạo 1 thư mục Android trong ổ C, ví dụ:



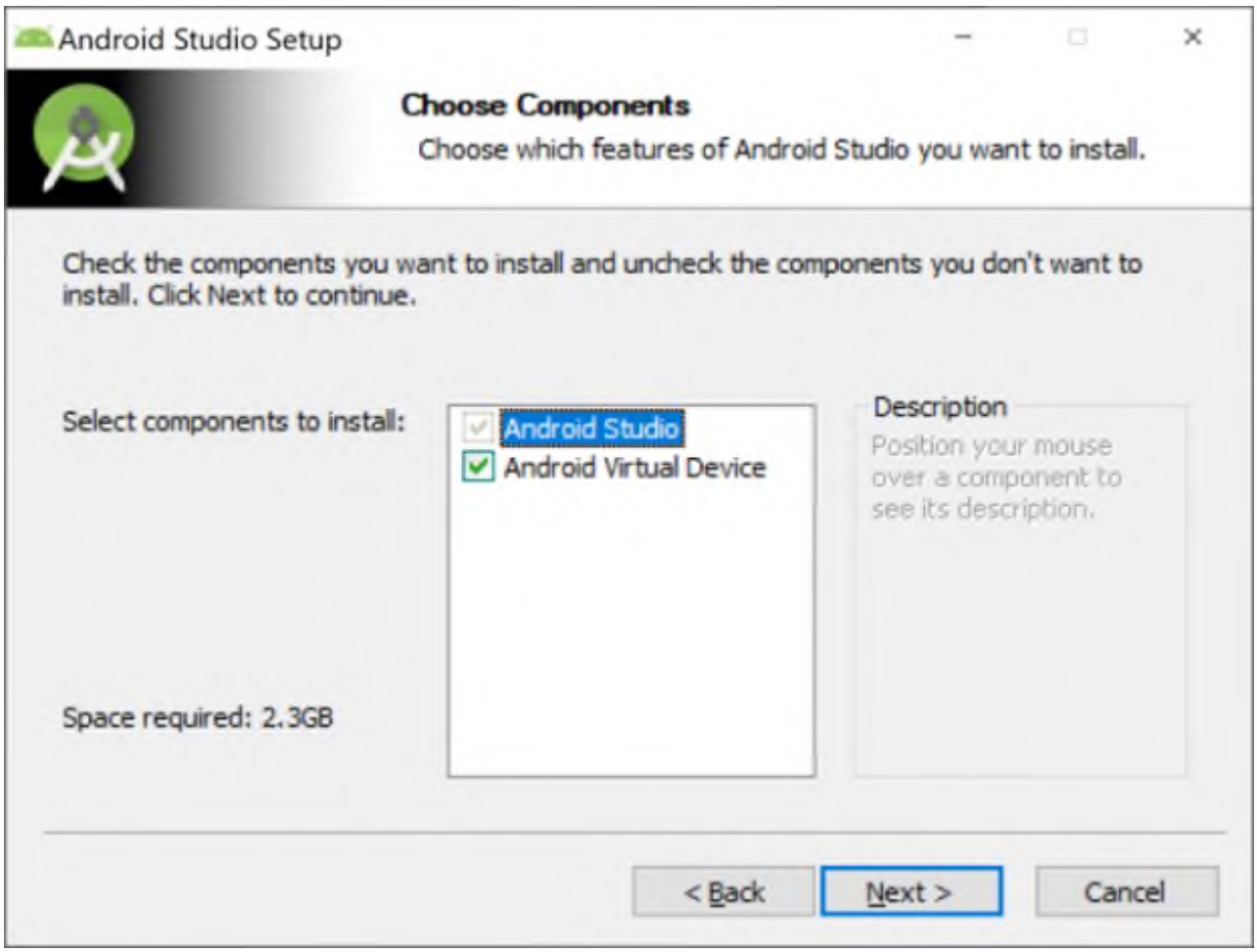
Khi cài đặt sẽ có 2 thành phần mà ta nên cài vào bên trong thư mục Android, đó là:

- android-studio->công cụ lập trình
- sdk->các thư viện để hỗ trợ lập trình

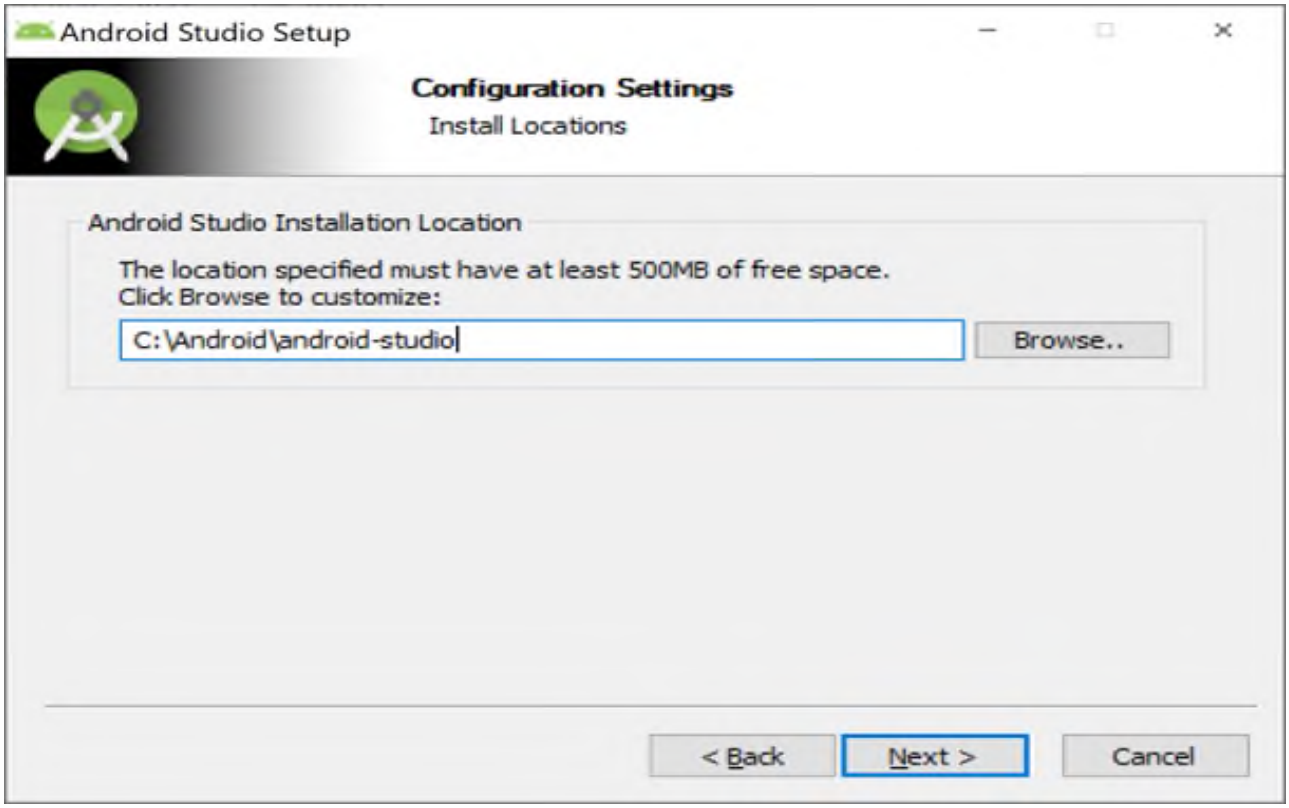
Bây giờ double click vào file cài mới tải ở trên về để cài đặt:



Nhấn Next để tiếp tục:

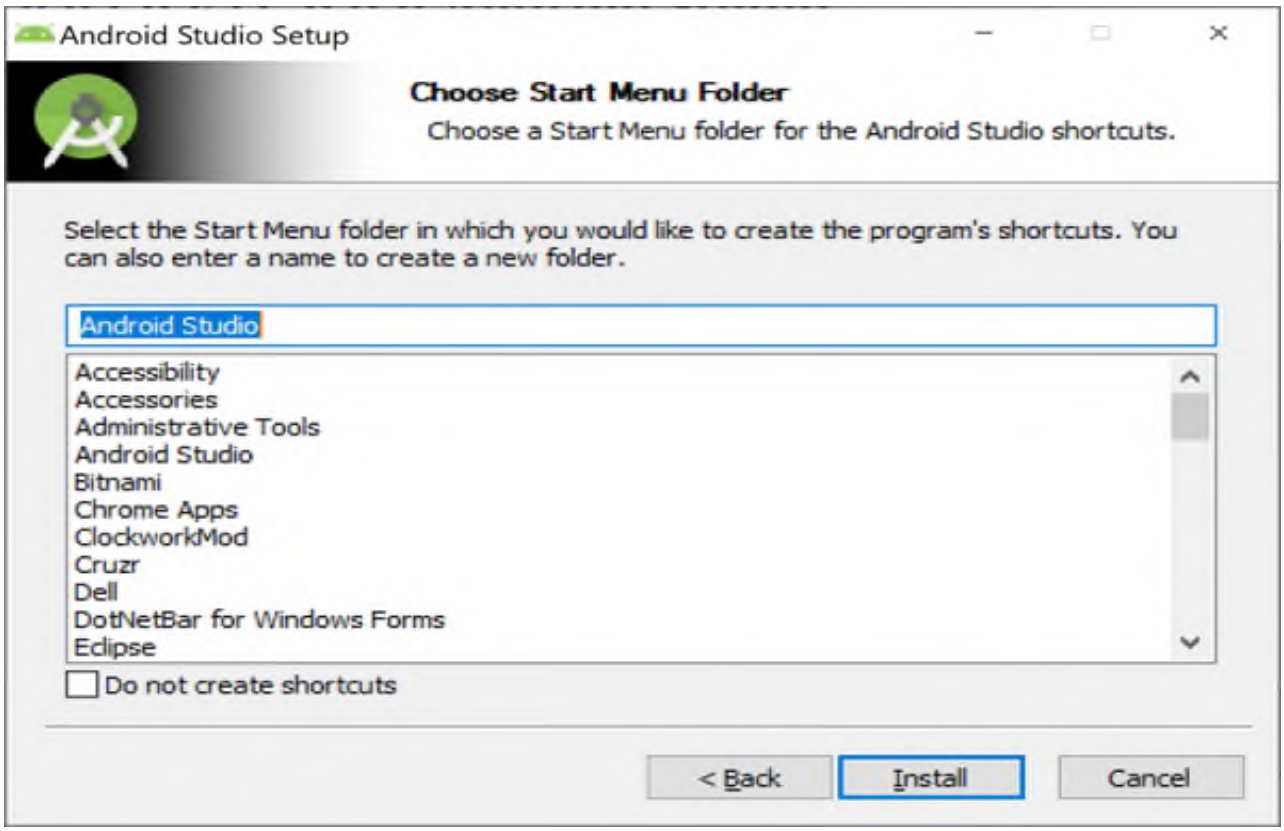


Để mặc định như trên rồi tiếp tục nhấn Next:

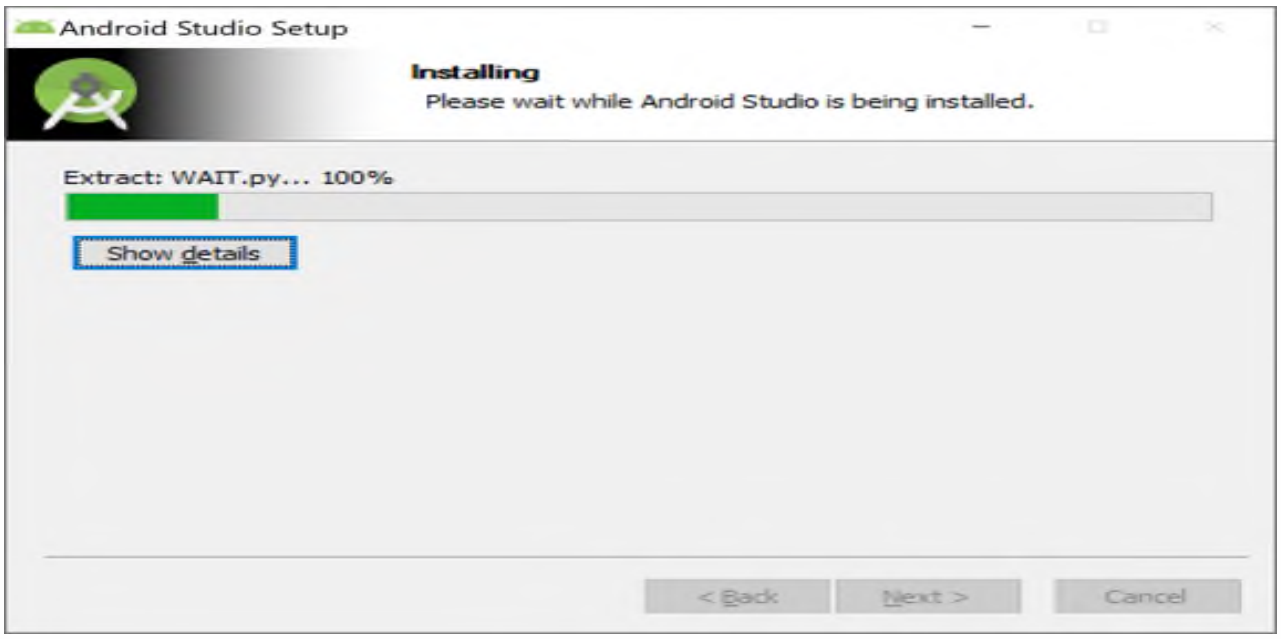


Ở màn hình trên lưu ý cài vào thư mục C:\Android\android-studio

Sau đó nhấn Next để tiếp tục:

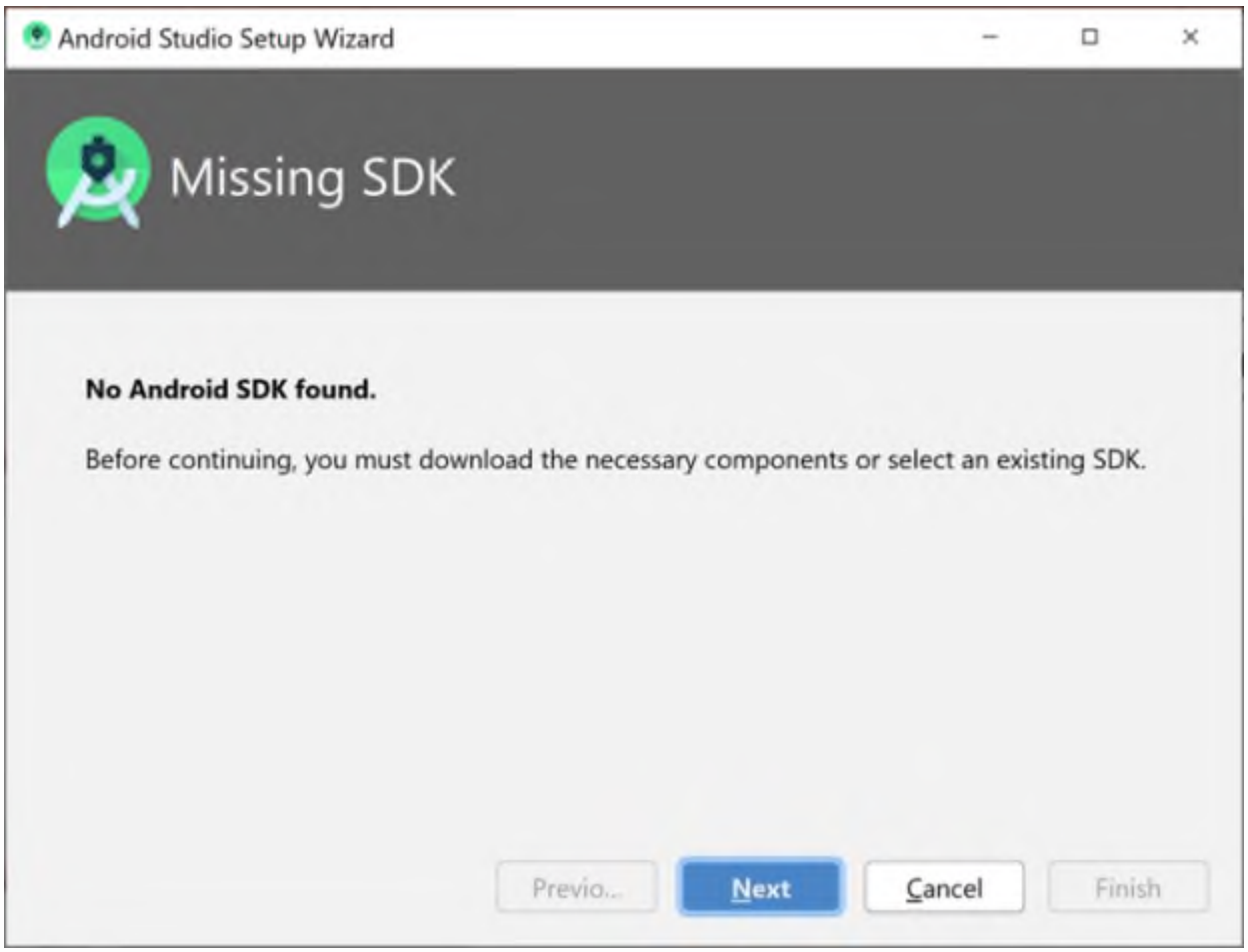


Sau đó nhấn "Install" để bắt đầu cài đặt

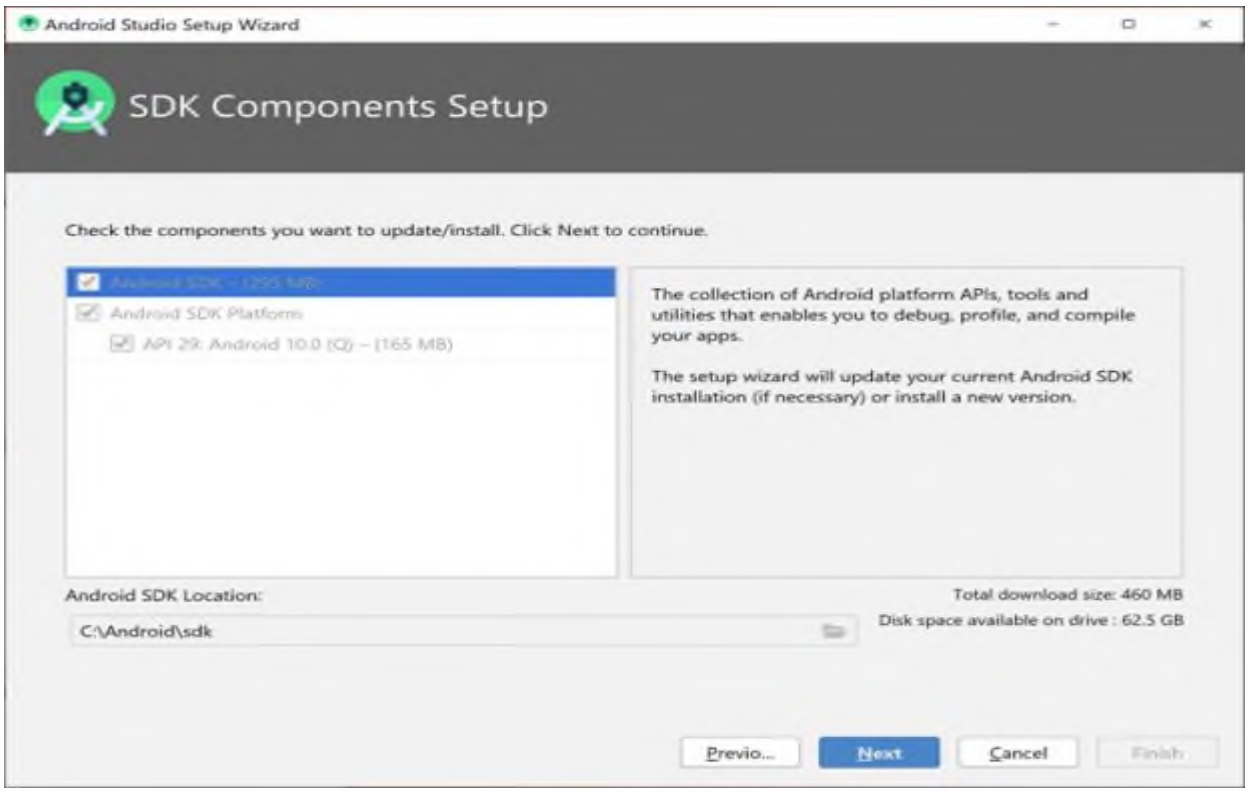


Ngồi chờ xíu cho nó cài đặt, khi cài đặt xong sẽ có thông báo như dưới đây:

Nếu như trước đó đã làm làm Android thì dĩ nhiên có SDK sẵn và phần mềm không yêu cầu gì thêm cả, nếu chưa bao giờ cài thì sẽ tiếp tục được yêu cầu cài SDK:



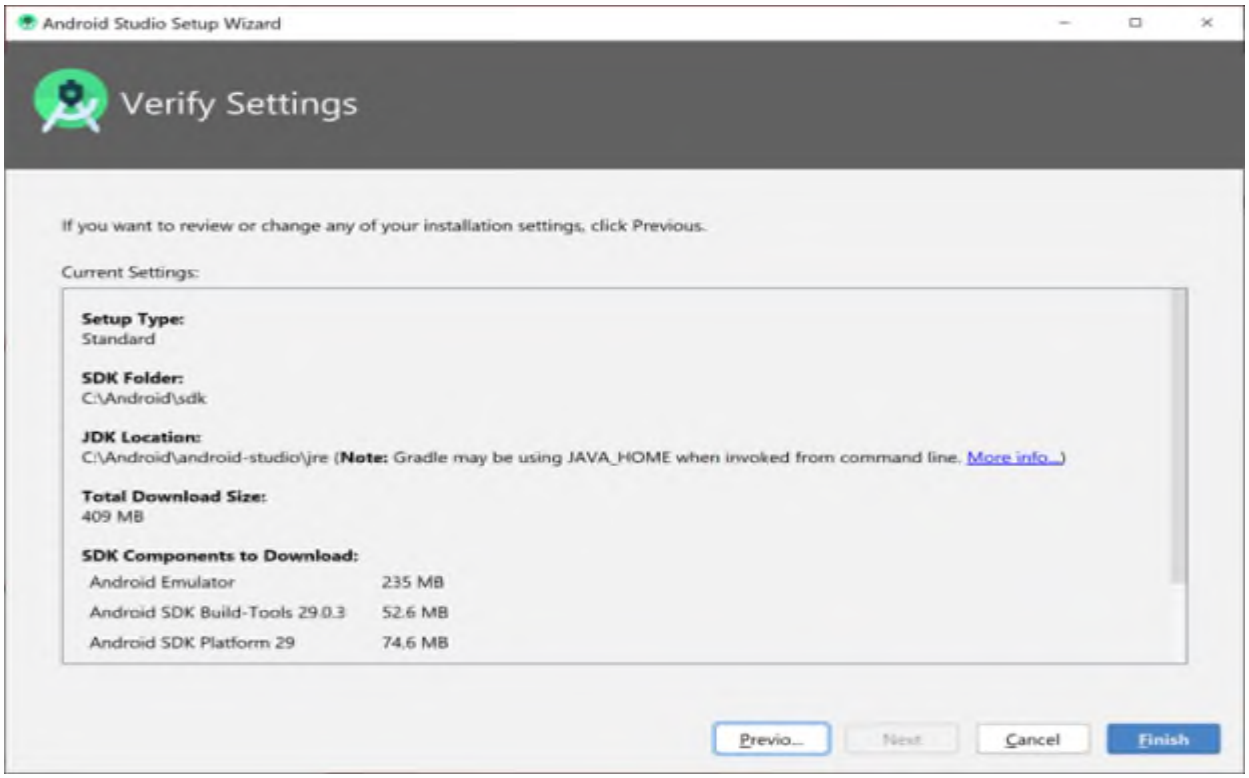
Chương trình sẽ báo Missing SDK, ta tiếp tục nhấn Next



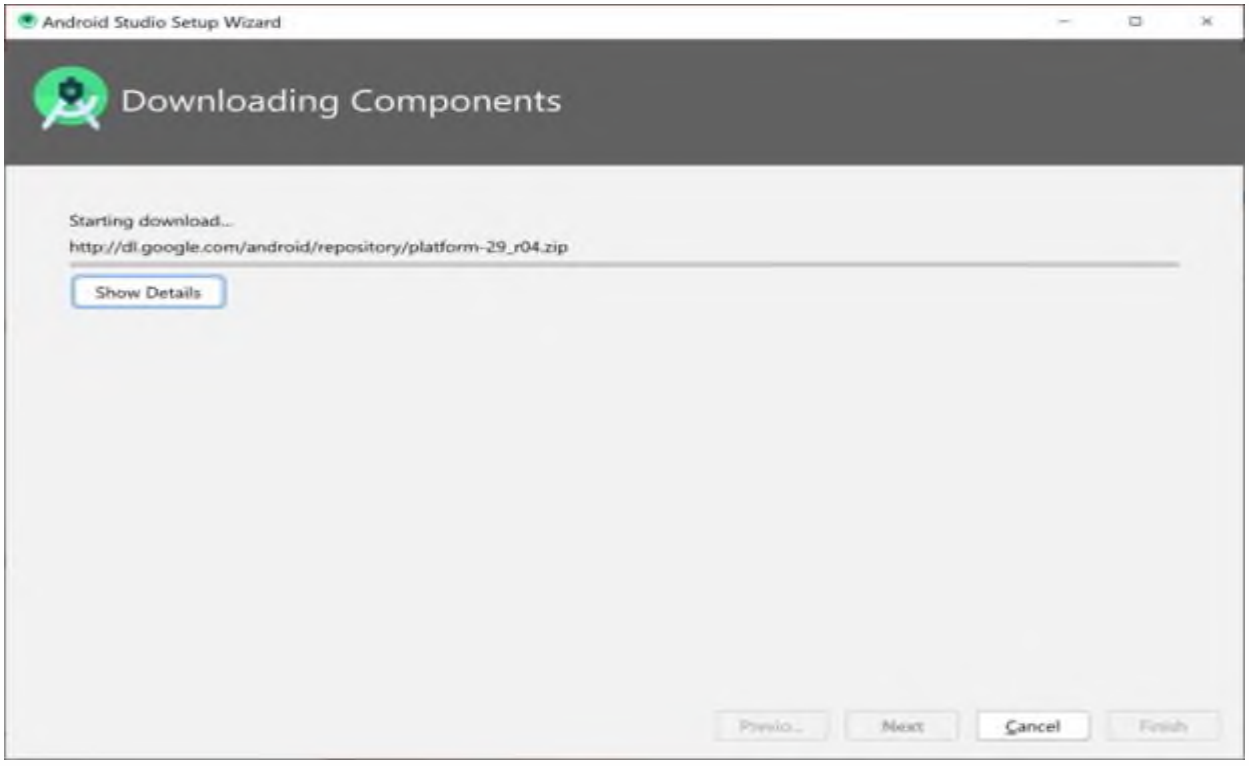
Lưu ý Android SDK Location ta chọn đúng nơi mà ta đã tạo thư mục trước đó:

C:\Android\sdk

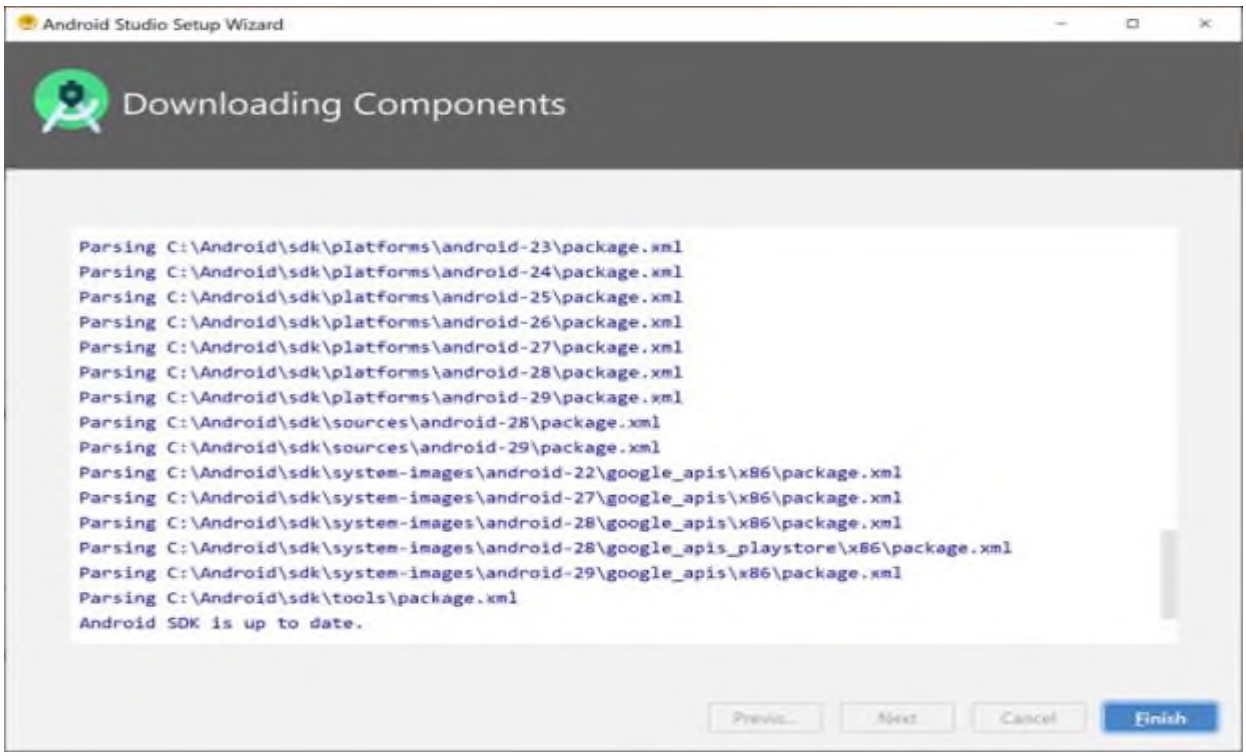
Sau đó nhấn Next để tiếp tục. Màn hình Verify Settings sẽ xuất hiện như dưới đây:



Nhấn FINISH để cài, màn hình Downloading Components sẽ hiển thị như dưới đây, chờ:

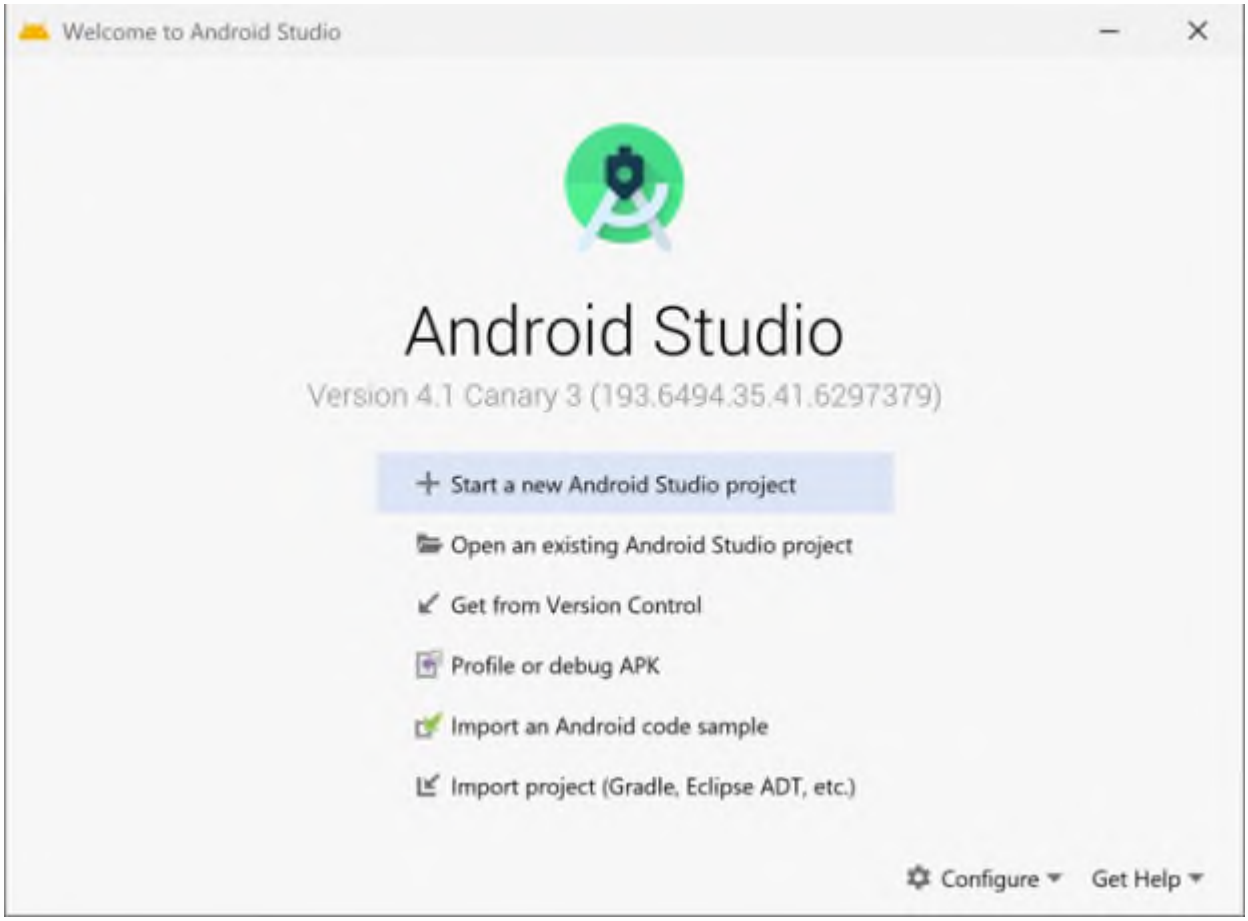


Chờ cho tới khi nó báo hoàn tất:



Nhấn Finish để hoàn tất quá trình cài đặt SDK

lúc này phần mềm Android Studio sẽ xuất hiện như dưới đây:



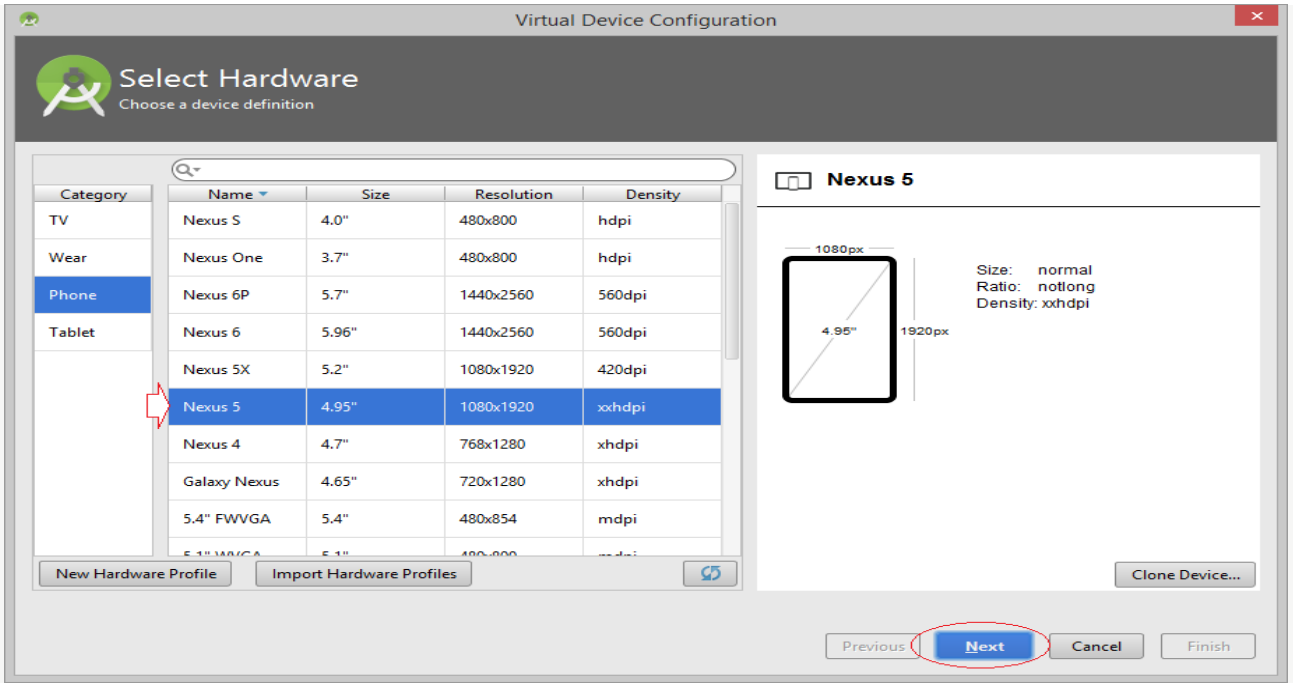
Nếu Android Studio yêu cầu chọn một số Setup Wizard, ví dụ như xuất hiện các màn hình dưới đây:

2.3 Cấu hình máy ảo

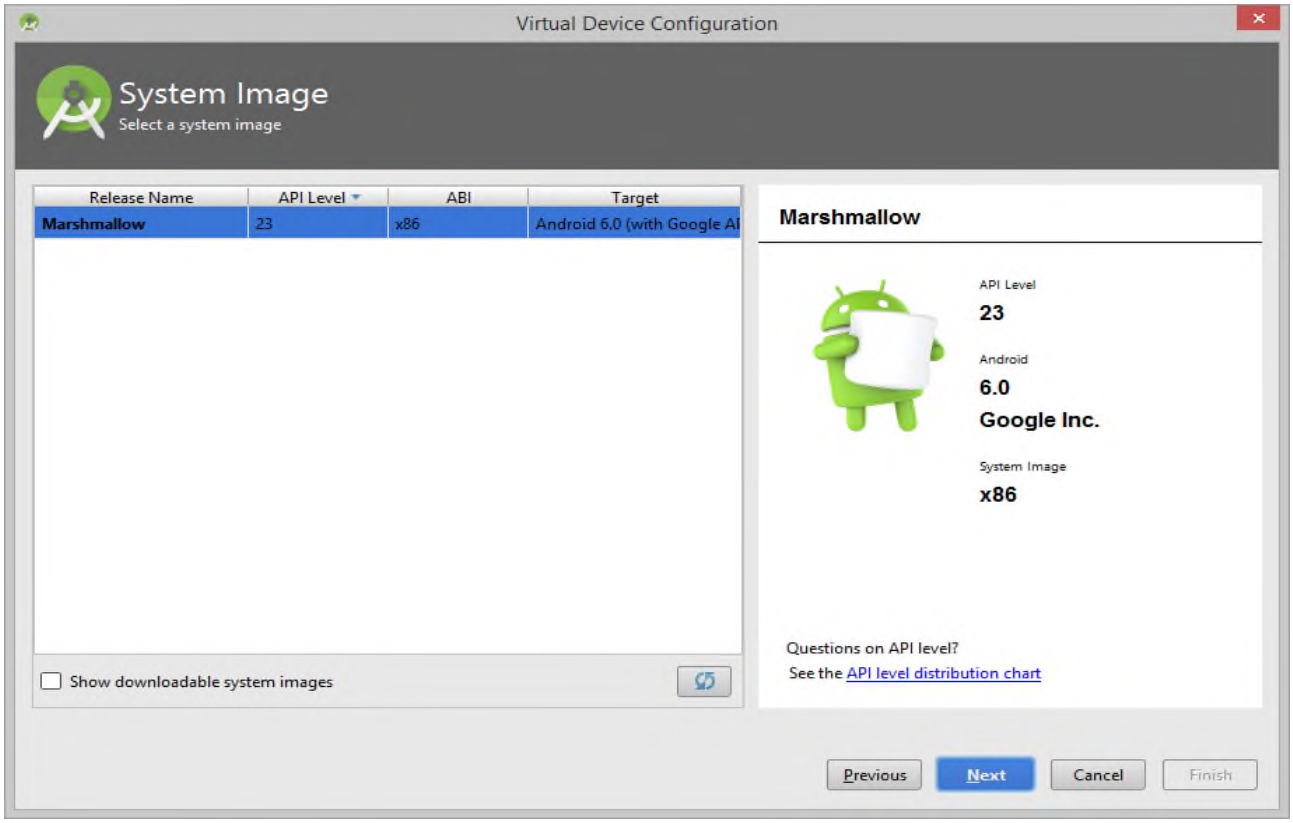
Máy áo để hiện thực chương trình Android Studio có Genymotion và Android Emulator được tích hợp trong Android Studio, trong giáo trình này chúng ta chọn Android Emulator.

Trên Android Studio, chọn Tool → Android → AVD Manager

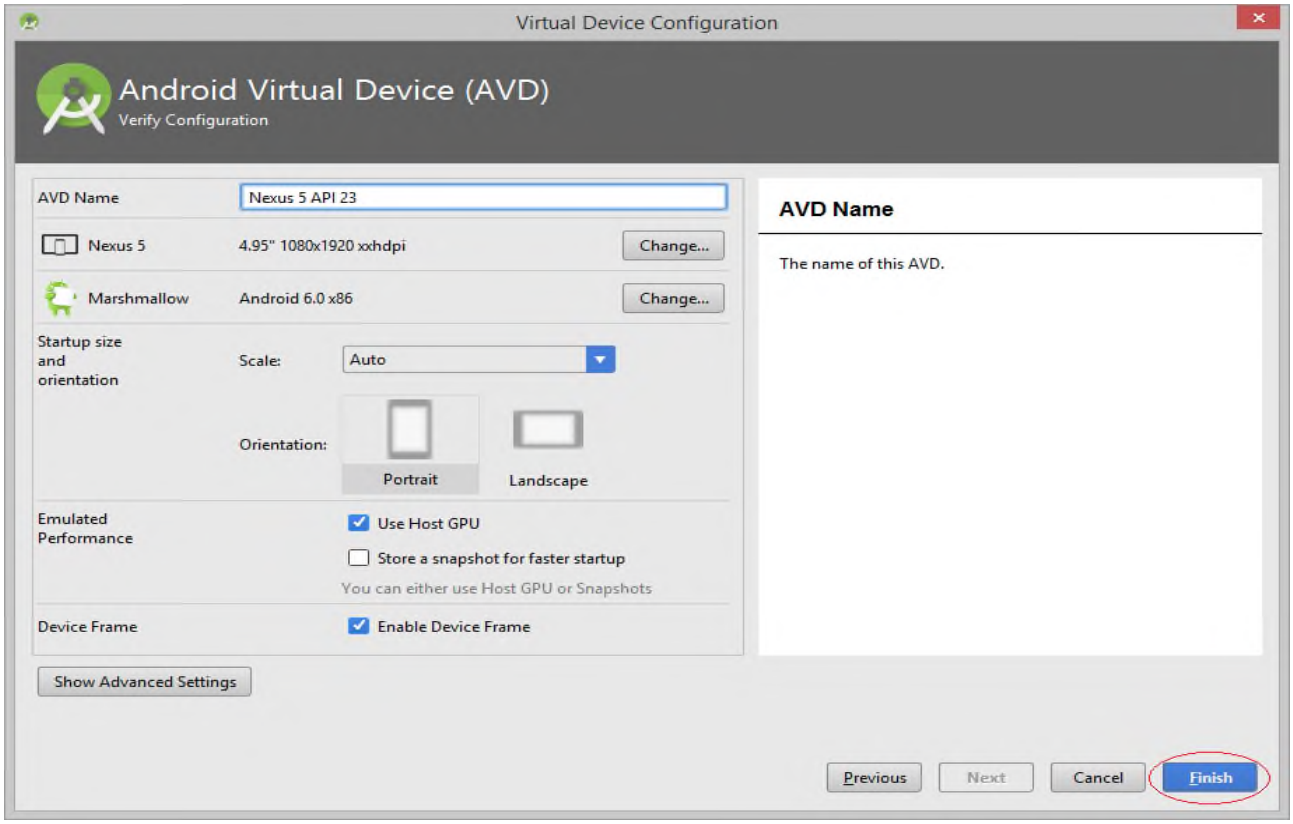
Chọn Create Virtual Device.. tạo một thiết bị mô phỏng Nexus 5 (4.95") - Đây là một thiết bị điện thoại thông dụng.



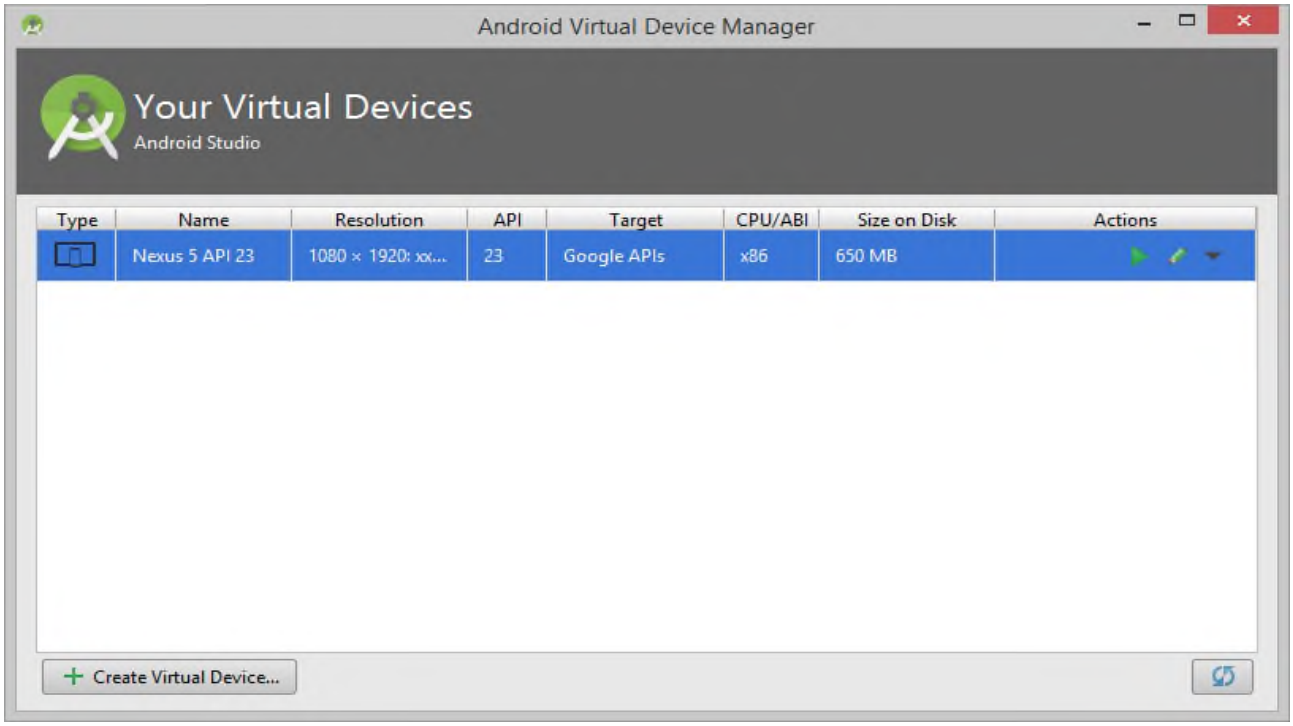
Chọn Nexus 5, tiếp tục chọn Next.



Tiếp tục chọn Next để thiết lập tiếp, sẽ xuất hiện cửa sổ sau



Tại cửa sổ này, ta chọn các thông số như màn hình mặc định là dọc hay ngang, thiết lập các thông số còn lại như hình và tiếp tục nhấn Next.

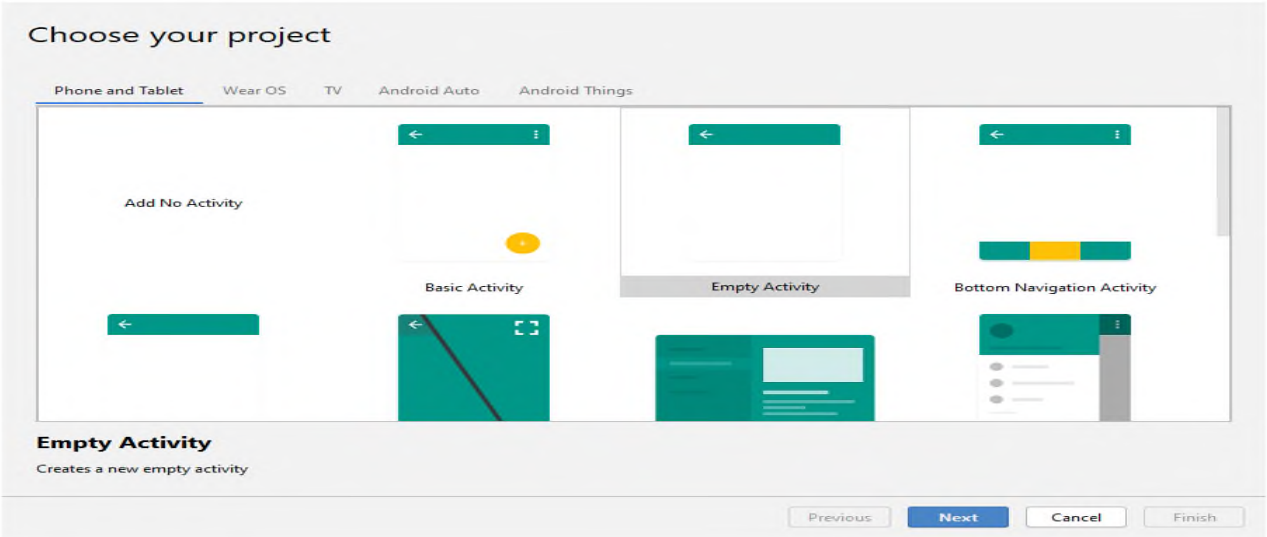


Như vậy chúng ta đã thêm thiết bị xong, bây giờ chúng ta có thể chọn Start tại cột Actions để thiết lập chạy thử.

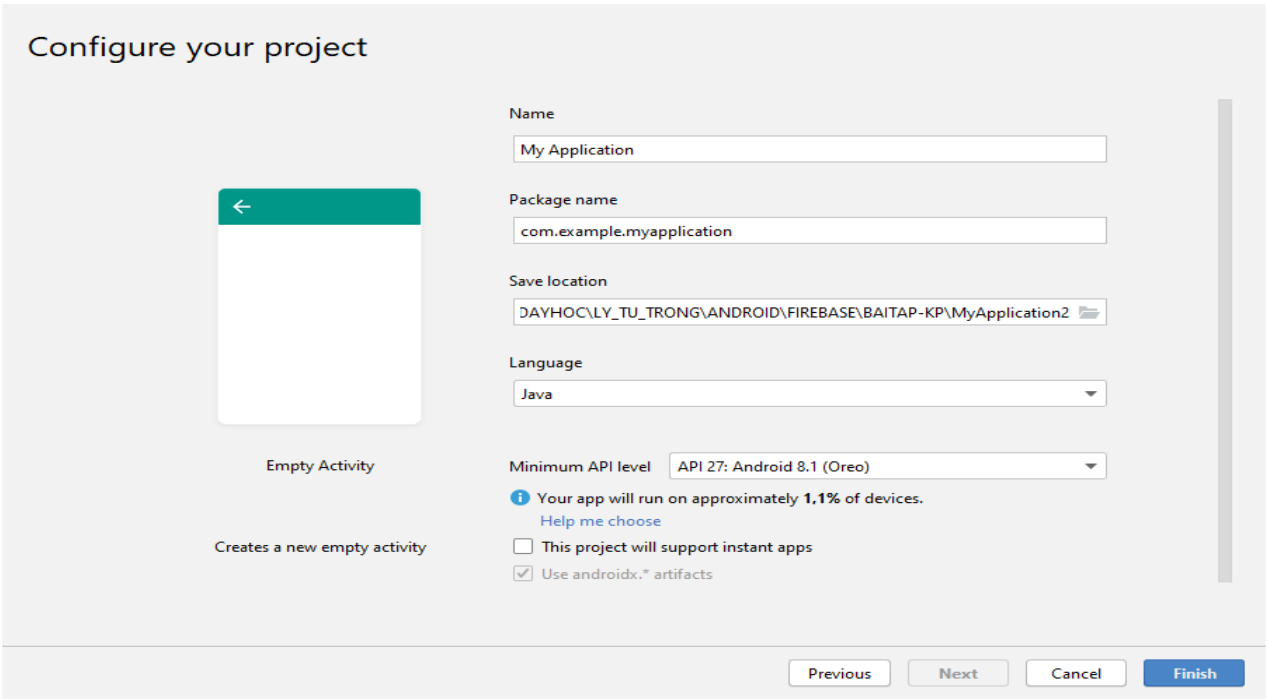
Chương 2. ỨNG DỤNG VÀ CÁC THÀNH PHẦN CỦA ỨNG DỤNG

2.1 Giao diện tương tác

Lần đầu khi khởi động Android Studio sẽ có một số cập nhật, chúng ta cứ để cho nó cập nhật, Sau đó chọn Start a new Android Studio project.



Chọn mục Empty Activity, chọn Next



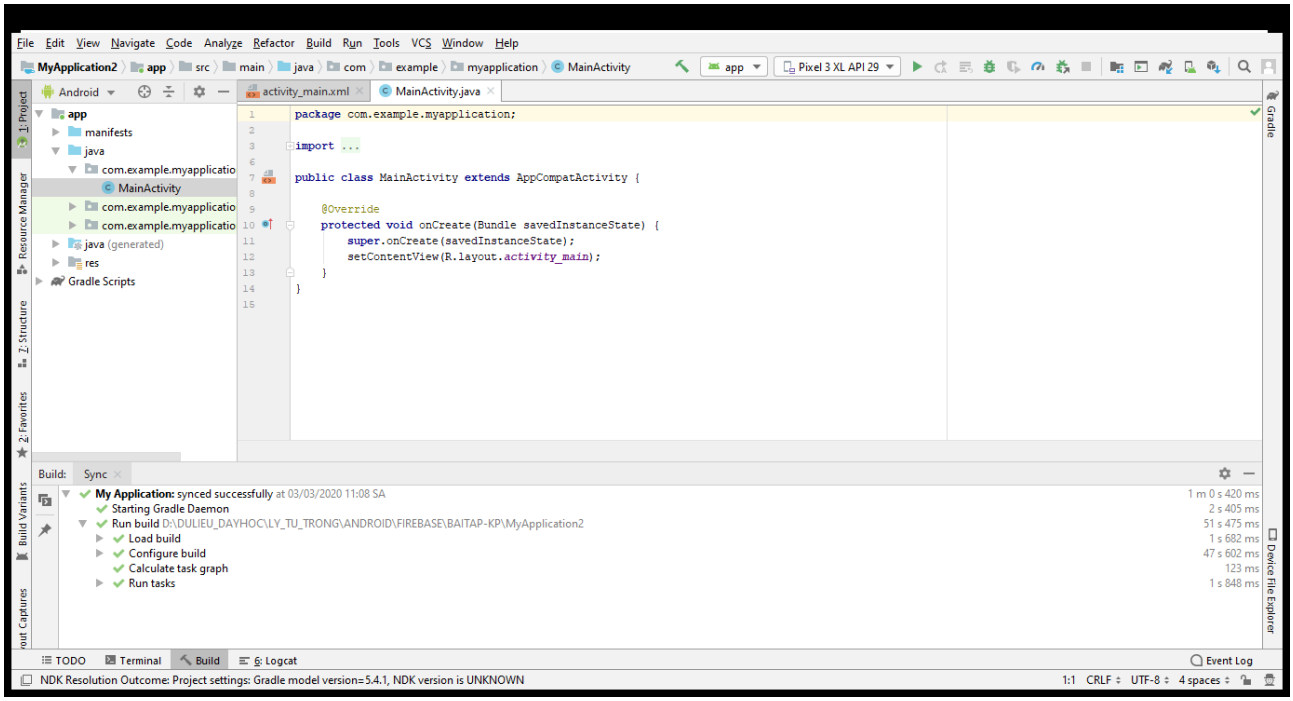
Tại của sổ này ta chọn tên ứng dụng(Ở đây tôi để mặc định là My Application)

Package name: chọn domain của ứng dụng, ví dụ ở đây chúng ta chọn com.example.myapplication

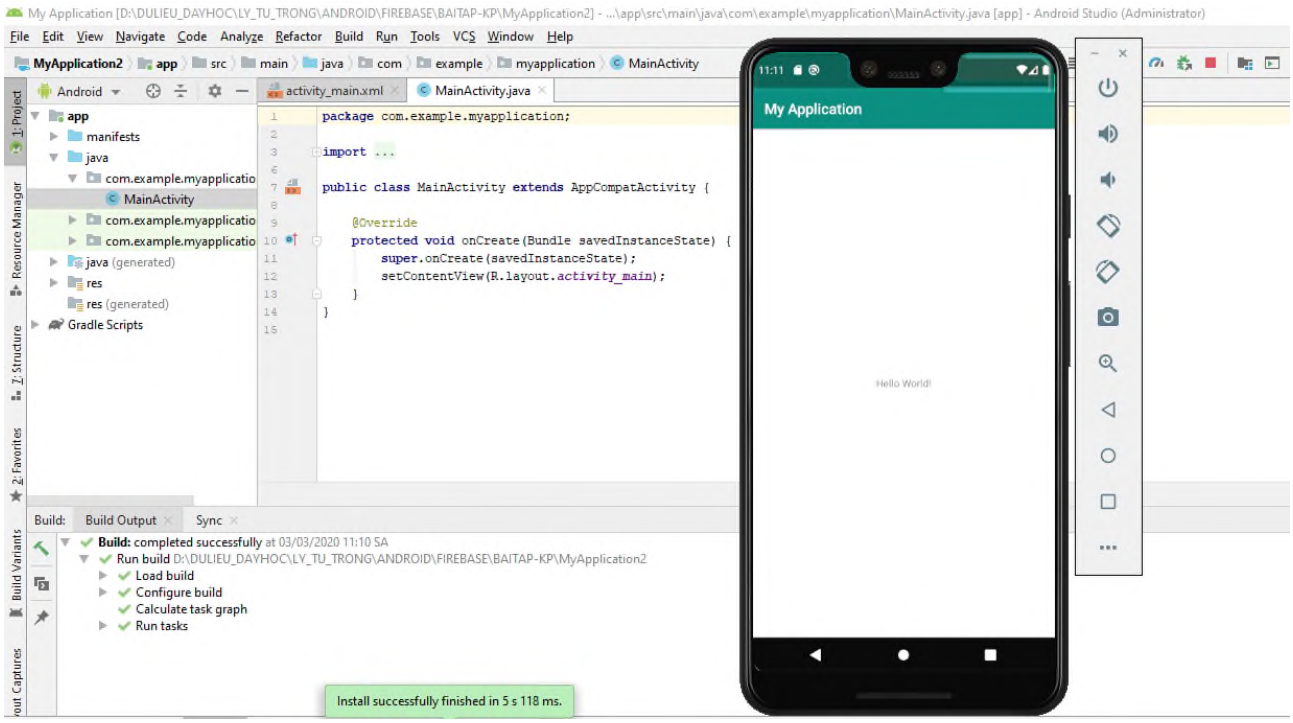
Save location: chọn vị trí cần lưu.

Language: ngôn ngữ lập trình

Minimum API level: chọn phiên bản Android thấp nhất cài được ứng dụng. Sau đó chọn Finish. Màn hình Android Studio sẽ hiển thị như sau:

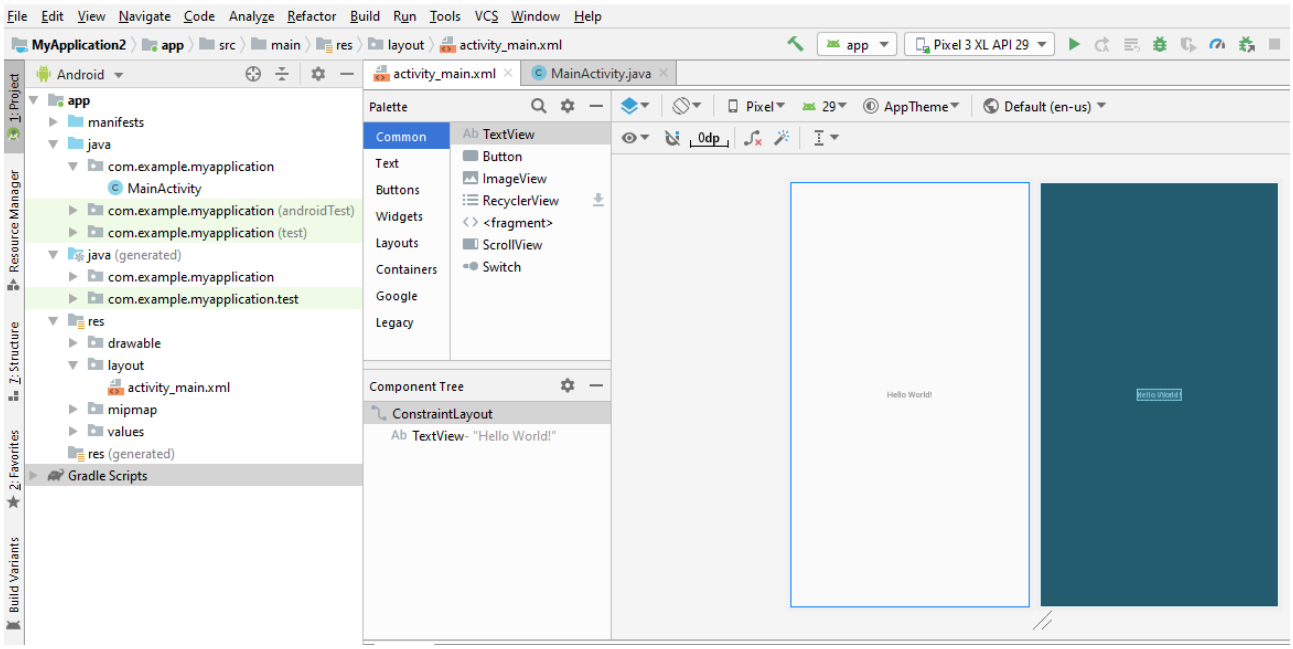


Sau đó chúng ta có thể chạy thử bằng cách nhấn vào chức năng Start trên thanh công cụ. Kết quả nhận được như sau:



Như vậy chúng ta đã thành công với 1 dự án mẫu.

Cấu trúc tổng thể cua một Project như sau:



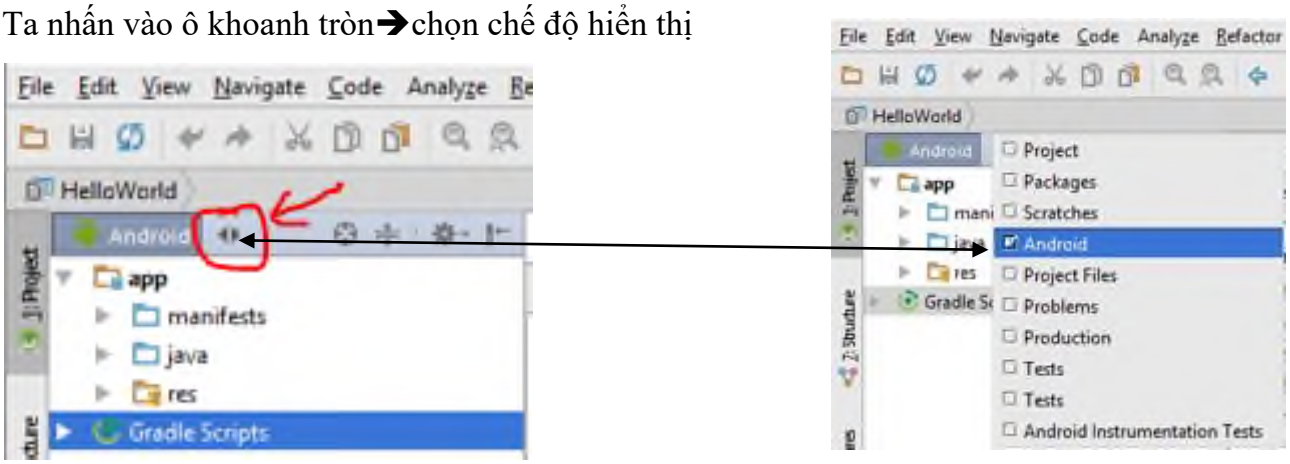
- Manifests : tham chiếu đến file AndroidManifest.xml, file này để bạn mô tả cấu trúc chính của App, nhằm dúng Android OS biết được các thiết lập cơ bản của App để khởi chạy được nó, như Activiy nào sẽ được chạy, ứng dụng xin những quyền gì trong thiết bị ...
- Java : thư mục lưu trữ các file code java của ứng dụng, lớp FirstActivity được định nghĩa với file FirstActivity.java lưu trong cấu trúc thự mục này
- res : lưu trữ các file tài nguyên mà ứng dụng sẽ sử dụng đến, nó tổ chức thành các thư mục con như:
    - drawable : ở đây cơ bản lưu các đối tượng đồ họa như các ảnh dạng png, các ảnh dạng xml ...
    - layout : lưu trữ các file xml biểu diễn về thành phần, bố cục của các thành phần hiện thị được trên màn hình
    - mipmap : cũng để lưu các đối tượng hình ảnh, ví dụ icon ứng dụng ic_launcher đặt ở đây
    - values: chứa các file như colors.xml, dimens.xml, strings.xml,styles.xml, đây là các file xml định nghĩa các giá trị có thể sử dụng trong ứng dụng như màu sắc, kích thước, các chuỗi, các theme ...
- Gradle Scripts: chứa nhiều nhánh con như build.gradle, local.properties ... là nơi bạn thiết lập các thông số để Gradle build ứng dụng. Bạn lưu ý Gradle là một công cụ

tích hợp vào Android Studio, chức năng của nó build mã nguồn, kết hợp tài nguyên, phân tích xml ... rồi kết hợp chúng lại với nhau tạo ra ứng dụng chạy trên JVM.

2.2 Các màn hình quan trọng

2.2.1 Màn hình Project và cách thay đổi layout quan sát

Ta nhấn vào ô khoanh tròn➔chọn chế độ hiển thị

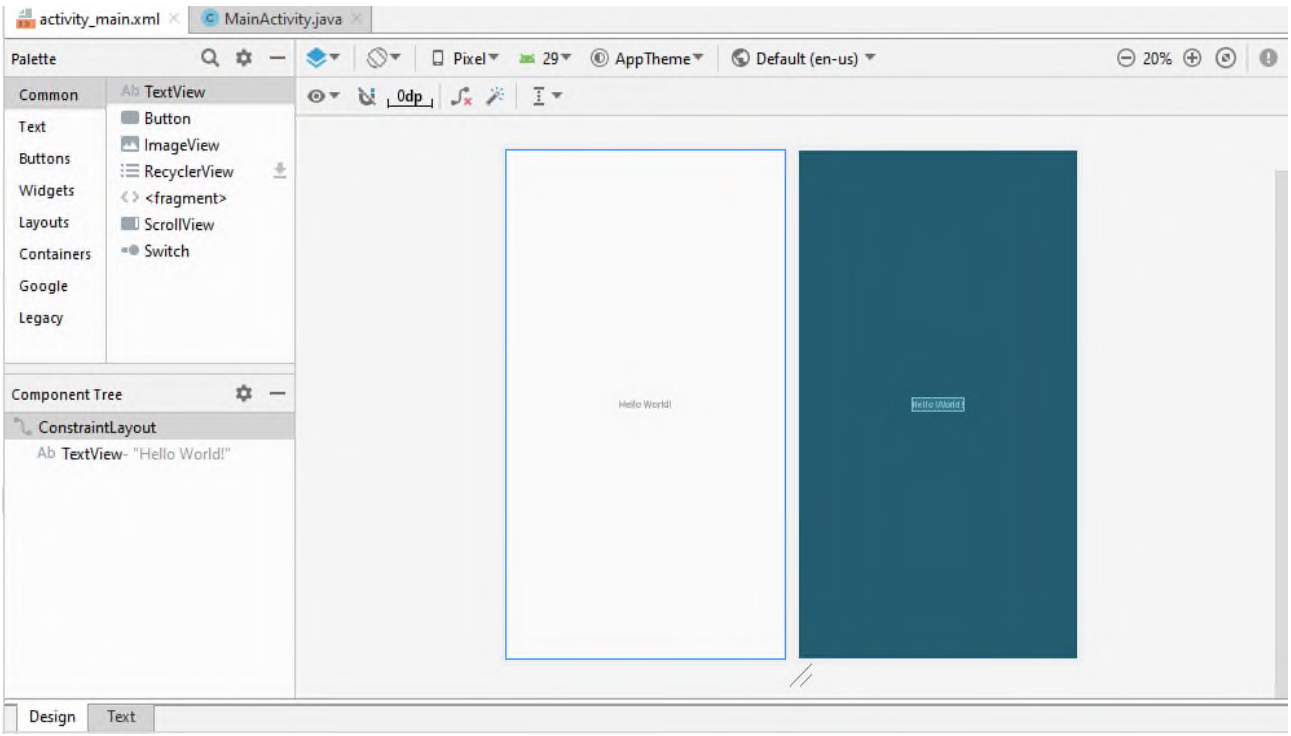


Rất nhiều chế độ để xem cấu trúc của 1 Project. Tùy vào mục đích sử dụng mà ta hiển thị khác nhau, tuy nhiên dùng nhiều nhất là Android.

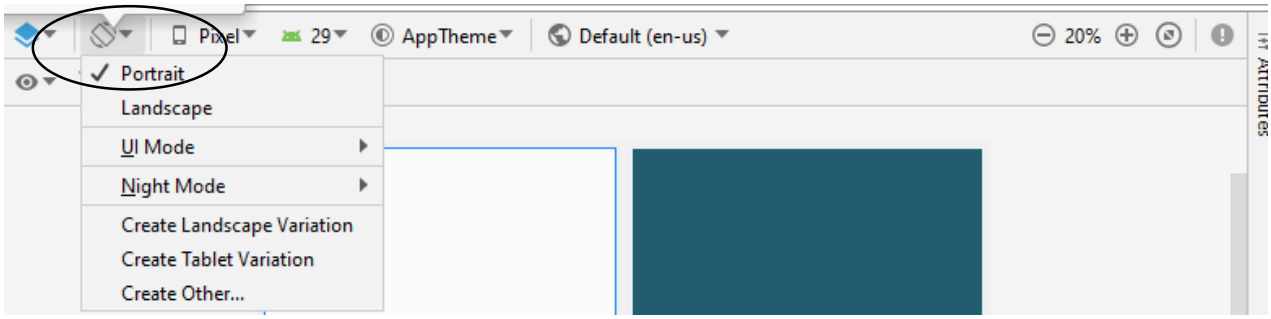
2.2.2 Màn hình thiết kế giao diện:

Design, XML, chế độ hiển thị điện thoại, đổi Theme, API version, phóng to, thu nhỏ

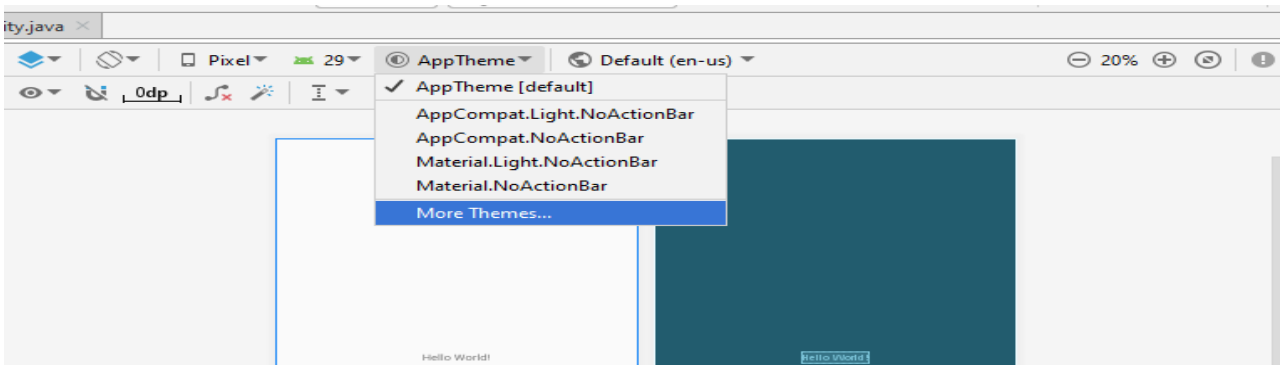
– Design, XML



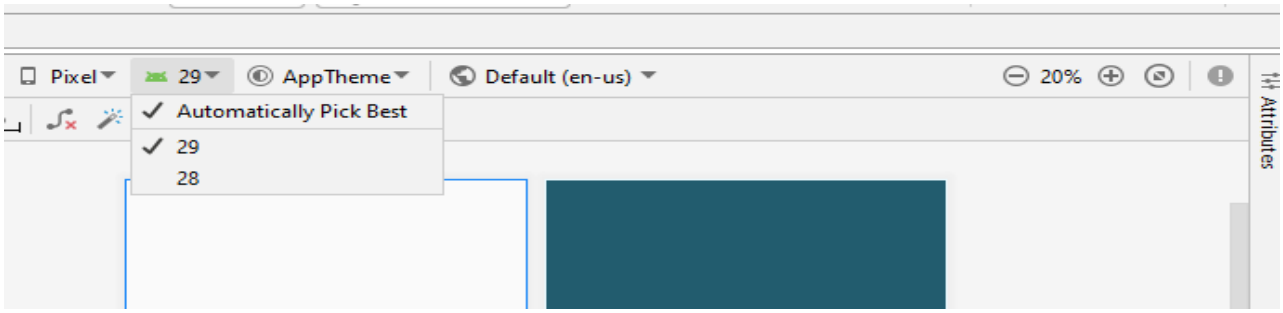
– Chế độ hiển thị điện thoại



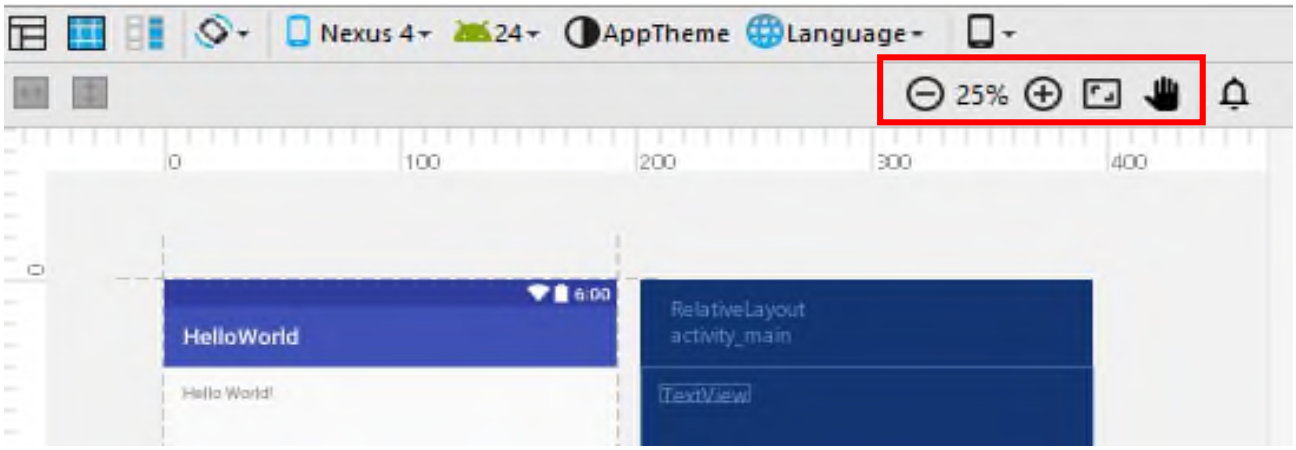
– Đổi Theme



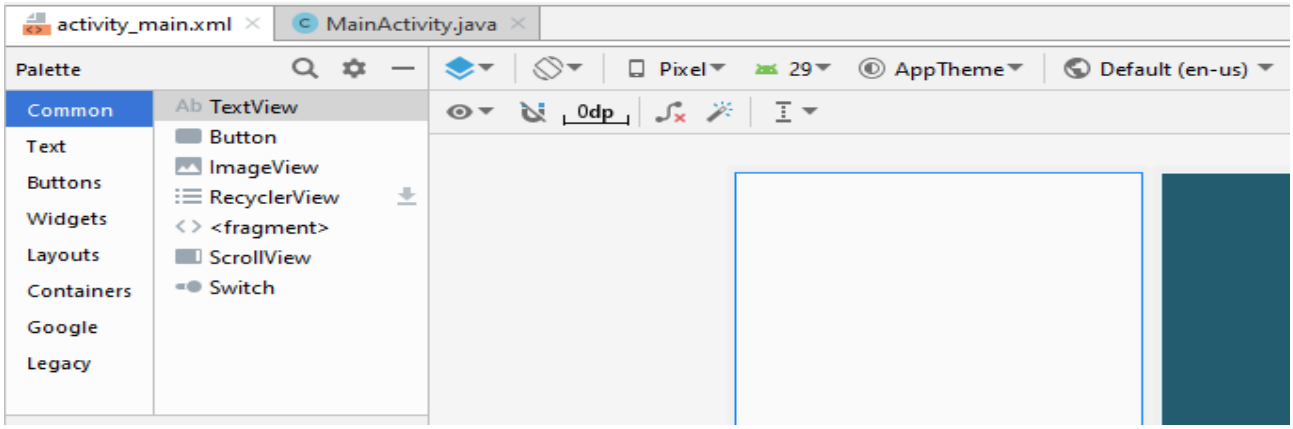
– API version



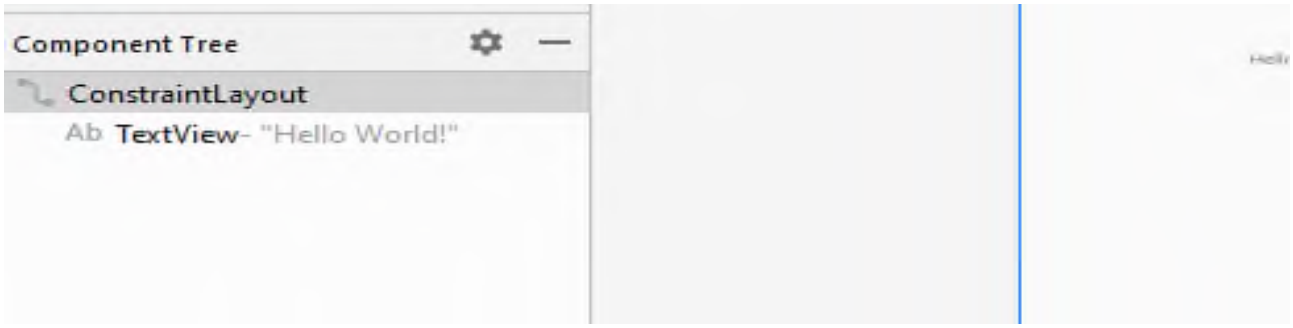
– Phóng to, thu nhỏ



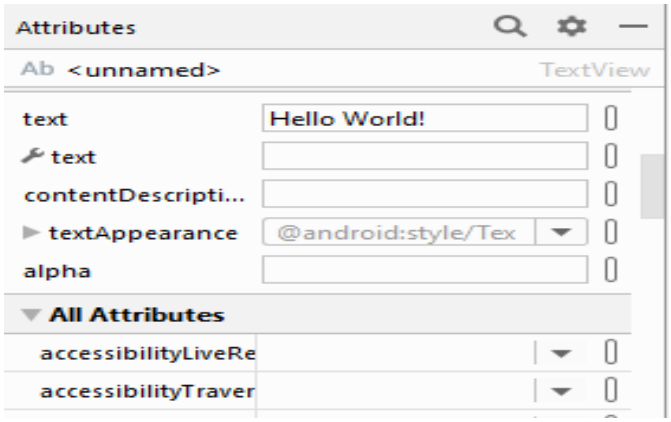
– Màn hình thanh công cụ Palette: Widget, layout, container, Image & Media...



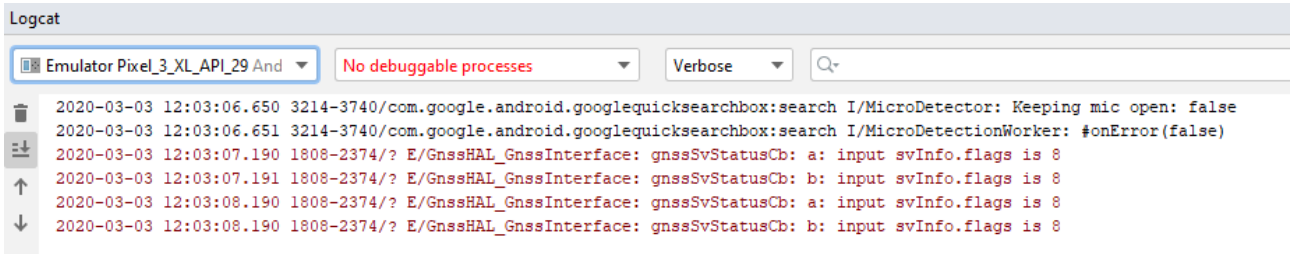
– Màn hình Component Tree



– Màn hình Properties



– Màn hình Logcat



Màn hình rất quan trọng, để kiểm tra lỗi xảy ra trong quá trình lập trình phần mềm.

Chương 3. GIAO DIỆN VÀ CÁC VIEW CƠ BẢN

Để thiết kế màn hình tương tác người dùng phải biết cách bố trí các đối tượng trên giao diện một cách phù hợp. Trong chương này, trình

bày các Layout thường dùng như: ConstrainLayout, RelativeLayout, LinearLayout, FrameLayout, TableLayout, GridLayout,... Các Layout này giúp chúng ta bố trí các đối tượng một cách dễ dàng theo nhiều cách khác nhau, phù hợp với từng ứng dụng cụ thể. Các đối tượng người dùng tương tác trên giao diện còn được gọi là View hay Control. Trong chương này sẽ trình bày thêm một số Control thông dụng như: TextView, EditText, ListView, ImageView, ImageButton, CheckBox, RadionButton, TimePicker, ProcessBar, v.v…

3.1 Các khái niệm

Các thành phần cơ bản để xây dựng giao diện người dùng (UI) trong Android là View, ViewGroup và Layout. Từ các thành phần cơ sở đó kết hợp chúng với nhau tạo ra các loại UI phức tạp cho ứng dụng Android. Để tạo ra một giao diện người dùng trong ứng dụng Android, chúng ta xây dựng từ sự kết hợp các thành phần có tên là View, ViewGroup, Layout.

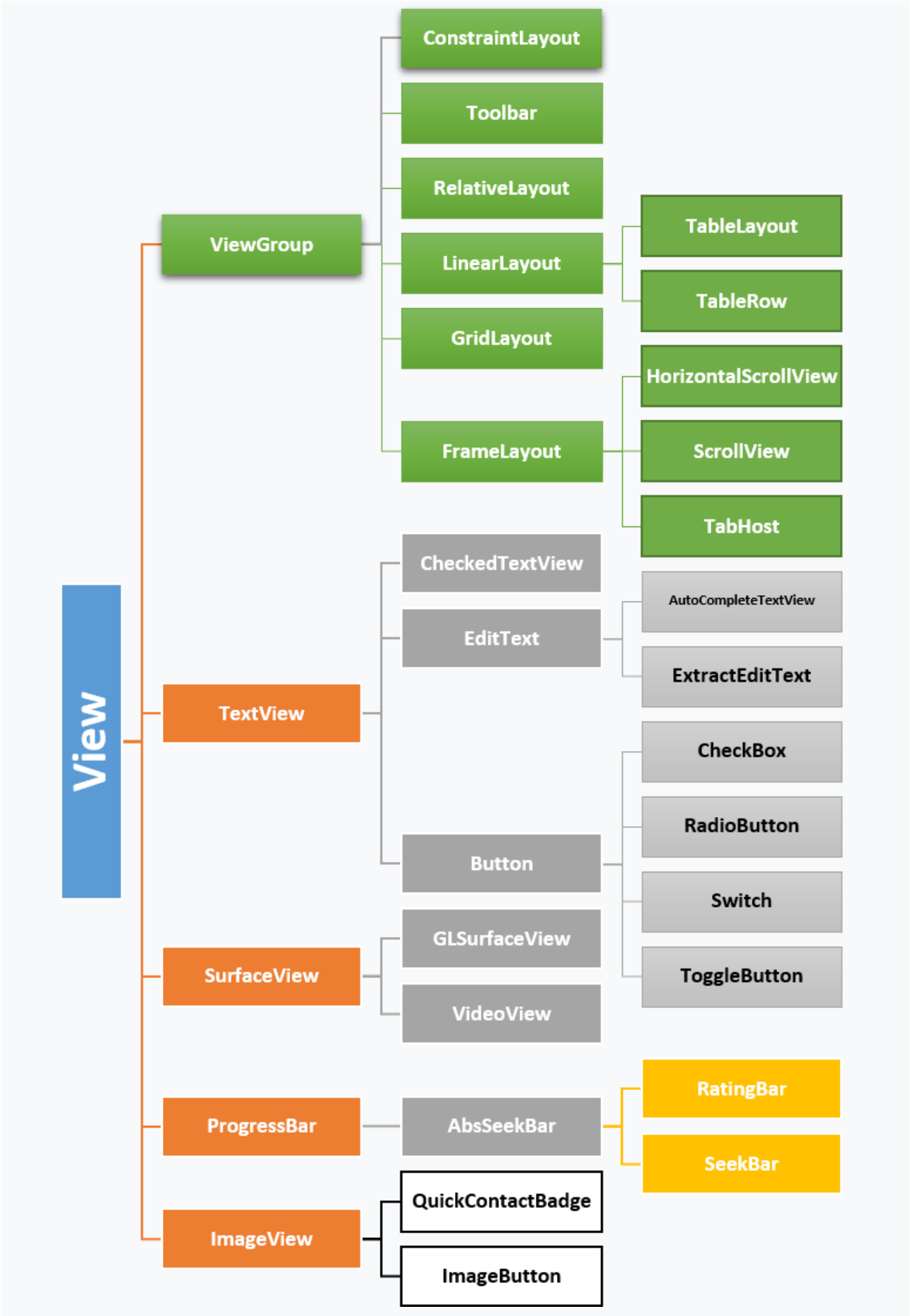
3.1.1 View

Các thành phần giao diện xây dựng từ lớp cơ sở View (android.view.View) của Android, các thành phần này cung cấp sẵn khá đa dạng như Button, TextView, CheckBox ... tất cả chúng ta gọi nó là View. Sơ đồ các View được mô tả theo sơ đồ như hình dưới.

View biểu diễn một hình chữ nhật, trong đó nó hiện thị thông tin nào đó cho người dùng, và người dùng có thể tương tác với View. Nhưng loại View cơ bản cần tìm hiểu trước tiên đó là: TextView, ImageView, Button, ImageButton, EditText

3.1.2 ViewGroup

Trong sơ đồ dưới, có một lớp abstract kế thừa từ View là ViewGroup, nó cũng chính là một View nhưng có khả năng chứa các View khác bên trong (kể cả ViewGroup khác). Nó là lớp cơ sở để xây dựng nên các layout như trên sơ đồ ta thấy có: ConstraintLayout, RelativeLayout, LinearLayout, GridLayout, FrameLayout


View
ViewGroup
ConstraintLayout
Toolbar
RelativeLayout
LinearLayout
TableLayout
TableRow
GridLayout
FrameLayout
HorizontalScrollView
ScrollView
TabHost
TextView
CheckedTextView
EditText
AutoCompleteTextView
ExtractEditText
Button
CheckBox
RadioButton
Switch
ToggleButton
SurfaceView
GLSurfaceView
VideoView
ProgressBar
AbsSeekBar
RatingBar
SeekBar
ImageView
QuickContactBadge
ImageButton

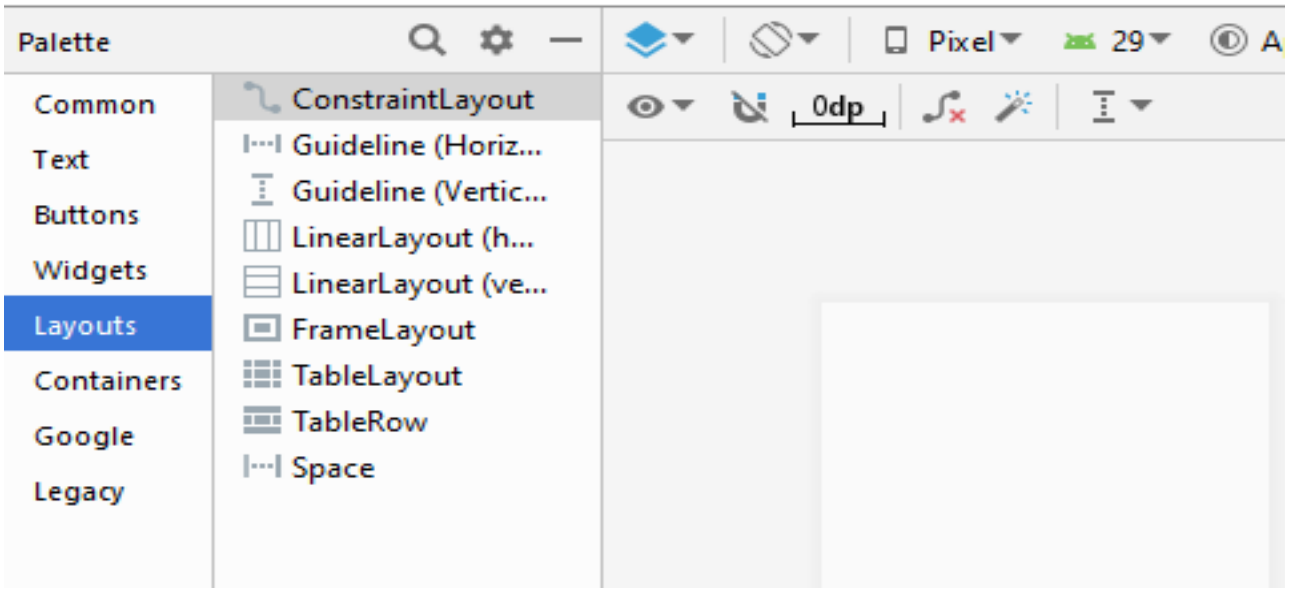
3.1.3 Các Layout

Các layout chính là các View (cụ thể nó kế thừa thừa ViewGroup) được thiết kế với mục đích chứa các View con và điều khiển, sắp xếp vị trí các View con đó trên màn hình, mỗi layout có cơ chế điều khiển vị trí View con riêng của mình. Có một số layout mà bạn tham khảo trước tiên như:

- **FrameLayout** đơn giản chỉ cung cấp một vùng màn hình, thường dùng nó để hiện thị một View con duy nhất. Nếu đặt vào nó nhiều view con, thì mặc định các view con này sẽ xếp chồng lên nhau. Tuy vậy, vị trí các view con trong nó cũng có thể điều chỉnh thông qua giá trị tham số gravity, ví dụ trong view con thiết lập android:layout_gravity="bottom|right" thì view con nằm về phía trái, dưới của FrameLayout
- **ConstraintLayout** (giới thiệu trong Android 7), sử dụng layout này được khuyến khích cho hầu hết trường hợp. ConstraintLayout cho phép điều khiển vị trí và ứng sử của các view con trong layout bằng cách gán dàng buộc đơn giản vảo mỗi view con. Từ đó mà một bố cục phức tạp có thể dễ dàng được tạo ra mà sử dụng ít nhất sự lồng nhau trong layout (layout này nằm trong layout khác) giúp cho cải thiện tốc độ. ConstraintLayout cũng tích hợp sẵn vào Android Studio Layout Editor nên bạn có thể điều chỉnh một cách trực quan các View con trong layout này.
- **LinearLayout** với layout này thì các view con được xếp nối tiếp nhau (linear) thành một hàng hay một cột (tùy vào lúc thiết kế thiết lập hướng xếp). Có một giá trị weight có thể gán vào mỗi View con để cho biết View con đó chiếm bao nhiêu không gian trong một tỷ lệ tương quan với các View con khác.
- **RelativeLayout** chô phép các view con định vị căn vào liên hệ với các view con khác đồng thời liên hệ với view cha thông qua các tham số align và margin. Ví dụ một View con thiết lập nằm ở giữa RelateiveLayout (android:layout_centerInParent="true"), một View con khác có thể thiết lập nằm căn trùng lề phải với View này (android:layout_alignRight="@+id/other")
- **GridLayout** chia ra thành lưới gồm một số hàng và một số cột để chứa các view con.
- **TableLayout** cung cấp khả năng bố trí các view con thành một lưới dạng bảng (gồm có hàng và cột). Một dòng của bảng biểu diễn bàng đối tượng view con TableRow, trong nó chứa có phần tử View con hiểu như các ô bảng.

- **CoordinatorLayout** nó được thiết kế nhằm mục đích có sự tương tác của các View con trong nó, đặc biệt sử dụng với ActionBar, FloatingActionButton, Snackbar ...

Các Layout hiển thị trong của sổ Palette như hình sau:



3.1.4 Các thuộc tính cơ bản trong View, GroupView và Layout

Thuộc tính	Diễn tả
android:alpha	Nhận giá trị từ 0 - 1 là kênh alpha (độ trọng suốt) của View. Nhận 0 trong suốt hoàn toàn.
android:background	Thiết lập nền bằng màu sắc, drawable ... cho View
android:backgroundTint	Thiết lập màu sẽ nhuộm vào nền
android:clickable	Thiết lập true thì View có khả năng nhận sự kiện click
android:foreground	Gán một ảnh Drawable để vẽ vào phần nội dung View
android:foregroundGravity	Định vị foreground trong View
android:id	Gán một ID vào View, ví dụ android:id="@+id/imagebutton"
android:minWidth	Gán kích thước chiều rộng nhỏ nhất
android:padding	Thiết lập kích thước Padding cho cả 4 cạnh
android:paddingLeft	Thiết lập Padding cạnh trái (Tương tự có android:paddingRight, android:paddingStart, android:paddingTop)
android:rotation	Thiết lập góc xoay của View (độ)
android:scaleX	Tỷ lệ thu/phong View theo chiều X
android:scaleY	Tỷ lệ thu/phong View theo chiều Y
android:tag	Gán một giá trị vào làm tag của View, sau này có thể đọc lại bằng getTag() và có thể tìm lại View bằng findViewWithTag()
android:theme	Thiết lập theme cho View
android:visibility	Gán các giá trị: gone (ẩn hoàn toàn), invisible (không hiện thị - nhưng chỗ nó chiếm vẫn còn), visible (hiện thị)
android:translationX	Dịch chuyển view theo chiều X
android:translationY	Dịch chuyển view theo chiều Y

3.2 Sử dụng các Layout trong Android

3.2.1 LinearLayout

LinearLayout là một layout rất tiện lợi trong thiết kế giao diện, nó sắp xếp các View con một cách liên tục theo tùy chọn xếp theo chiều ngang (một hàng các view) hay chiều đứng (một cột các view). Chúng ta có thể định hướng bố cục theo thuộc tính: android:orientation

Layout có 2 chiều bố cục là:

- Vertical Orientation: các view bên trong được sắp xếp theo chiều dọc.
- Horizontal Orientation: các view bên trong được sắp xếp theo chiều ngang.

Cấu trúc XML của LinearLayout được mô tả như sau:

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

</LinearLayout>
```

Một số thuộc tính quan trọng của LinearLayout:

Thuộc tính	Mô tả
android:orientation	ể thiết lập cách sắp xếp phần tử. android:orientation="horizontal" xếp theo chiều ngang và android:orientation="vertical" xếp theo chiều dọc.
Android:layout_width	Chiều rộng của View/ViewGroup. Có giá trị match_parent là chiều rộng sẽ full màn hình. Nếu có giá trị wrap_content thì chiều rộng sẽ tự co giãn theo đúng kích thước các View khác.
Android:layout_height	Chiều cao của View/ViewGroup. Có giá trị match_parent là chiều cao sẽ full màn hình. Nếu có giá trị wrap_content thì chiều cao sẽ tự co giãn theo đúng kích thước các View khác.
android:gravity	Xác định cách một đối tượng nên đặt nội dung của nó, trên cả hai tọa độ X và Y. Giá trị có thể là top, bottom, left, right, center, center_vertical, center_horizontal …
Android: layout_gravity	Xác định một đối tượng đặt ở vị trí nào trong LinearLayout. Giá trị có thể là top, bottom, left, right, center, center_vertical, center_horizontal …
android:weightSum	Tính tổng độ rộng các view con

Các View con trong LinearLayout có thể gán cho nó một giá trị trọng số bằng thuộc tính android:layout_weight ví dụ như: android:layout_weight="2"; android:layout_weight ="1.5" .... Nếu View không gán giá trị này coi như nó có trọng số bằng không.

Giá trị trọng số này sẽ được LinearLayout sử dụng để điều chỉnh kích thước View con có trọng số (điều chỉnh chiều cao nếu là loại LinearLayout đứng và điều chỉnh chiều rộng nếu là loại ngang)

Xét sự ảnh hưởng của trọng số với hai trường hợp sau:

Trường hợp 1

Trường hợp LinearLayout không sử dụng đến thuộc tính android:weightSum

- Đầu tiên các View không gán trọng số android:layout_weight sẽ có kích thước (rộng hay là cao tùy vào LinearLayout là ngang hay đứng)sẽ theo đúng như thuộc tính android:layout_width, android:layout_height của nó.
- Không gian (kích thước) còn lại của LinearLayout sẽ chia cho các View có trọng số. Cái nào trọng số lớn hơn sẽ có kích thước lớn hơn. Nếu gọi kích thước còn lại là Size, gọi tổng các trọng số của các View là SUM, thì một view có trọng số là weight sẽ có kích thước là: weight*(Size/SUM).

Trường hợp 2: Trường hợp này trong LinearLayout thiết lập thuộc tính weightSum ví dụ như android:weightSum="10", thì lúc này giá trị SUM trong công thức phân bổ kích thước ở trên không phải là lấy từ tổng weight các View con mà lấy bằng đúng android:weightSum của LinearLayout, sau đó vẫn dùng công thức phân bổ kích thước như trên: weight*(Size/SUM).

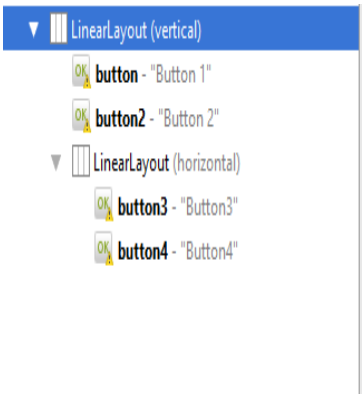
Lưu ý: các kích thước mà gán match_parent không bị ảnh hưởng của trọng số.

Bài tập

1. Dùng Linear layout thiết kế giao diện sau:

Phân tích layout :Phần layout được khoanh vùng đỏ linear layout ở dạng vertical, chứa 2 Button Phần layout được khoanh vùng xanh chứa linear layout ở dạng horizontal, chứa 2 Button có cấu trúc như hình bên

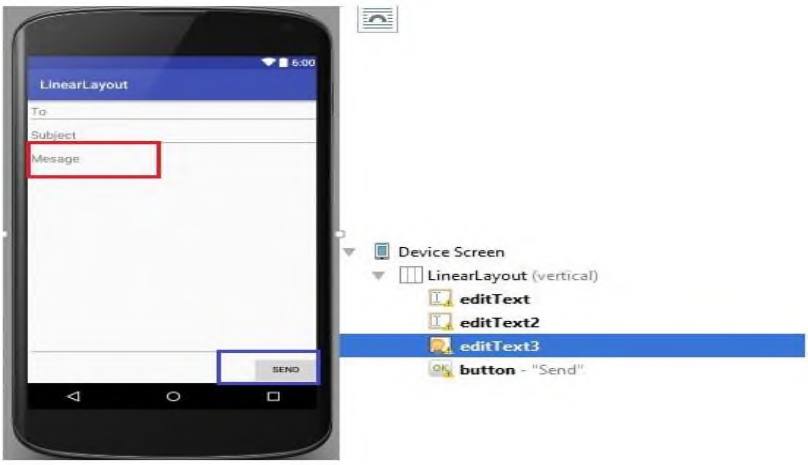
Hướng dẫn: Thiết kế layout dạng layout_weight với



Button 3 và 4: Được chứa trong Linear horizontal ,kết hợp sử dụng thuộc tính các giá trị là 3 và 1.

Bài 2: Kết hợp lồng nhiều layout trong linearlayout:

Phân tích layout:



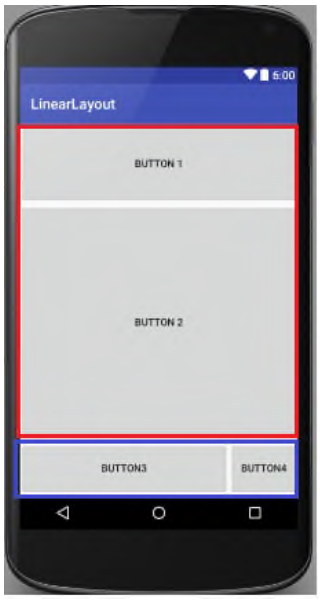
- Phần View được khoanh đỏ để nội dung trong View hiển thị lên trên ta sử dụng thuộc tính gravity
- Phần View được khoanh xanh để View nằm bên phải ta sử dụng thuộc tính layout_gravity

*Lưu ý:

- Gravity: Định nghĩa nội dung bên trong View
- Layout_gravity: Định nghĩa vị trí của View

Bắt đầu thiết kế:

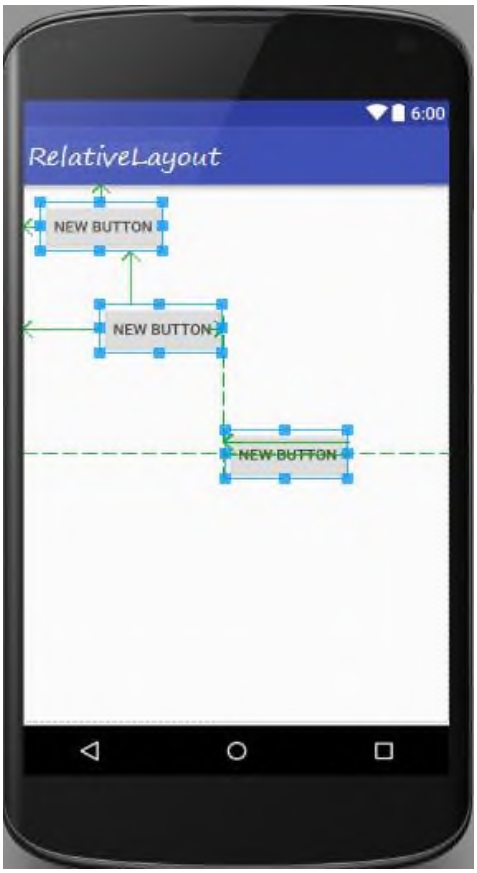
- Với View chứa Message để hiển thị lên top sử dụng thuộc tính : android:gravity="top" tùy theo nhu cầu có thể sử dụng các giá trị khác nhau: center,bottom…..



- Với View Button Send để nằm bên trái theo yêu cầu sử dụng thuộc tính: android:layout_gravity="right" với giá trị bằng right.

Ngoài ra thuộc tính này cũng có nhiều giá trị khác tương tự như gravity.

3.2.2 RelativeLayout



RelativeLayout là layout mà nó hiện thị các view con nó chứa ở các vị trí trong mối liên hệ của chúng với nhau (như View con này nằm dưới một View con khác, View con này nằm bên trái một View con khác ...), kể cả mối liên hệ của chúng với chính phần tử cha RelativeLayout (như căn thẳng theo cạnh đấy của phần tử cha, nằm giữa phần tử cha, nằm bên trái phần tử cha ...).

RelativeLayout là một tiện ích rất mạnh mẽ cho thiết kế một giao diện người sử dụng vì nó có thể loại bỏ các nhóm **View** lồng nhau và giữ cho hệ thống phân cấp bố trí của bạn bằng phẳng, đồng thời cải thiện hiệu suất. Nếu bạn sử dụng một vài nhóm **LinearLayout** lồng nhau, bạn có thể thay thế chúng bằng một **RelativeLayout** duy nhất.

Định vị mặc định

Khi các View con đưa vào RelativeLayout nếu chưa có thiết lập mối liên hệ qua lại nào với phần tử cha hay với phần tử View con khác thì nó sẽ được định vị ở góc trên - trái của RelativeLayout. Như trường hợp dưới đây cả 3 View con không có thiết lập mối liên hệ nào, nên nó đều định vị ở góc trên / trái và vẽ chồng nên nhau, View con nào xếp sẽ ở lớp trên của màn hình.

Một số thuộc tính quan trọng của RelativeLayout:

Thuộc tính	Mô tả
Android:layout_width	Chiều rộng của View/ViewGroup. Có giá trị match_parent là chiều rộng sẽ full màn hình. Nếu có giá trị wrap_content thì chiều rộng sẽ tự co giãn theo đúng kích thước các View khác.
Android:layout_height	Chiều cao của View/ViewGroup. Có giá trị match_parent là chiều cao sẽ full màn hình. Nếu có giá trị wrap_content thì chiều cao sẽ tự co giãn theo đúng kích thước các View khác.

Thuộc tính	Mô tả
android:gravity	Xác định cách một đối tượng nên đặt nội dung của nó, trên cả hai tọa độ X và Y. Giá trị có thể là top, bottom, left, right, center, center_vertical, center_horizontal …
Android:ignoreGravity	RelativeLayout có hỗ trợ chỉ ra một View con tách khỏi khối biên chữ nhật chứa các View con để phần tử đó không bị ảnh hưởng bởi **gravity** bằng thuộc tính **android:ignoreGravity="id-view-con"**

Định vị View con bằng liên hệ với View cha RelativeLayout

Vị trí của View con trong RelativeLayout có thể thiết lập bằng cách chỉ ra mối liên hệ vị trí với view cha như căn thẳng cạnh trái View cha với View con, căn thẳng cạnh phải View cha với View con ... Các thuộc tính thực hiện chức năng này như sau:

Thuộc tính	Ý nghĩa
android:layout_alignParentBottom	**true** căn thẳng cạnh dưới view con với cạnh dưới View cha
android:layout_alignParentLeft	**true** căn thẳng cạnh trái view con với cạnh trái View cha
android:layout_alignParentRight	**true** căn thẳng cạnh phải view con với cạnh phải View cha
android:layout_alignParentTop	**true** căn thẳng cạnh trên view con với cạnh trên View cha
android:layout_centerInParent	**true** căn view con vào giữa View cha
android:layout_centerHorizontal	**true** căn view con vào giữa View cha theo chiều ngang
android:layout_centerVertical	**true** căn view con vào giữa View cha theo chiều đứng

Định vị View con bằng liên hệ giữa chúng với nhau

Thuộc tính	Ý nghĩa
android:layout_below	Nằm phía dưới View có ID được chỉ ra
android:layout_above	Nằm phía trên View có ID được chỉ ra
android:layout_toLeftOf	Nằm phía trái View có ID được chỉ ra
android:layout_toRightOf	Nằm phía phải View có ID được chỉ ra
android:layout_alignBottom	Căn thẳng cạnh dưới với cạnh dưới của View có ID được chỉ ra
android:layout_alignLeft	Căn thẳng cạnh trái với cạnh trái của View có ID được chỉ ra

android:layout_alignRight	Căn thẳng cạnh phải với cạnh phải của View có ID được chỉ ra
android:layout_alignTop	Căn thẳng cạnh trên với cạnh trên của View có ID được chỉ ra

View con trong RelativeLayout ngoài mối liên hệ với View cha như trên, chúng có có thể thiết lập liên hệ với nhau ví dụ như View con này nằm phía trên một View con khác, nằm phía dưới một view con khác ...

Các android:layout_margin ... của View con: Về phía các cạnh của View con (left, top, right, bottom) nếu có mối liên hệ với View cha hoặc View con thì theo phía đó có thể thiết lập thêm thuộc tính về margin như: android:layout_marginLeft, android:layout_marginTop, android:layout_marginRight, android:layout_marginBottom để thiết lập khoảng cách của mối liên hệ đó.

Bài tập

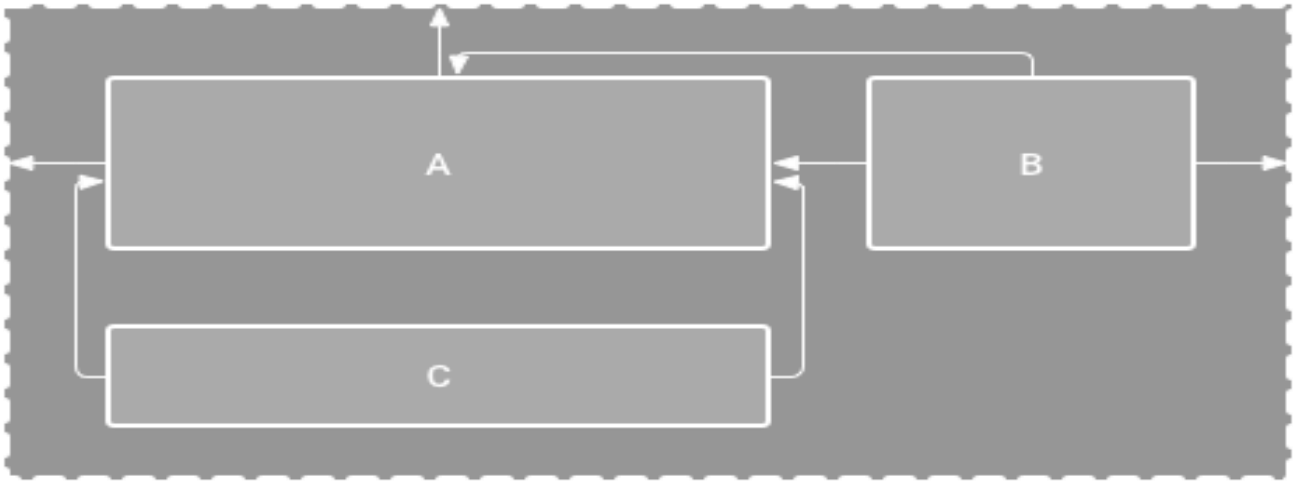
1. Dùng RelationLayout thiết kế giao diện sau

3.2.3 ConstraintLayout

ConstraintLayout là một layout mạnh, khuyến khích bạn dùng nếu có thể vì nó giúp tạo ra các giao diện phức tạp, mềm dẻo (hạn chế tối đa sử dụng các layout lồng nhau). Nó giúp định vị, sắp xếp các View con dựa trên sự ràng buộc liên hệ của các View con với View cha và sự liên hệ ràng buộc giữa các View con với nhau, với cơ chế tạo xích các View, gán trọng số hay sử dụng trợ giúp giao diện với Guideline.

ConstraintLayout thuộc Libaray Support nên để tích hợp vào dự án hãy thêm vào Gradle phiên bản muốn dùng, ví dụ:

```
implementation 'com.android.support:appcompat-v7:27.1.0'
implementation 'com.android.support.constraint:constraint-layout:1.0.2'
```



Sự ràng buộc: Mỗi view trong ConstraintLayout để định vị được chính xác cần tối thiểu 2 ràng buộc, một theo phương ngang (X) và một theo phương đứng (Y)

Ta có thể xác định một Constraint cho một hay nhiều mặt của một View bằng các chế độ kết nối bất kỳ sau đây:

- Điểm neo nằm trên một View khác.
- Một cạnh của Layout
- Một guideline

Khái niệm ràng buộc giữa các phần tử ở đây ám chỉ sự liên kết với nhau của các phần tử với nhau (kể cả với phần tử cha ConstraintLayout), sự căn chỉnh phần tử theo phần tử khác, hoặc với những đường thẳng ẩn thêm vào. Mỗi ràng buộc của phần tử View sẽ hoặc hưởng đến vị trí của nó theo trục X hoặc trục Y. Các View không có ràng buộc sẽ định vị ở góc trái - trên (tọa độ 0,0).

Trước tiên tham khảo bảng các thuộc tính về ràng buộc layout_constraint ... , các thuộc tính ràng buộc sử dụng với namespace:app, giá trị nó gán vào là một ID của phần tử khác để kết nối ràng buộc hoặc là phần tử ch bằng hằng số "parent", ví dụ:

```
app:layout_constraintBottom_toBottomOf="parent"
```

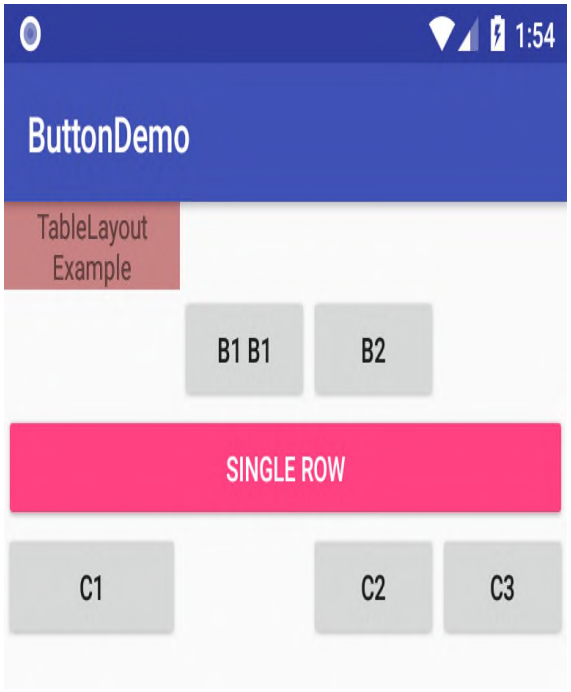
Các thuộc tính về ràng buộc ConstraintLayout

Ràng buộc	Ý nghĩa ràng buộc
app:layout_constraintLeft_toLeftOf	Rang buộc cạnh trái của phần tử tới phần tử chỉ ra trong giá trị (gán ID)
app:layout_constraintLeft_toRightOf	Bên trái với bên phải của phần tử chỉ ra
app:layout_constraintRight_toLeftOf	Bên phải với bên trái
app:layout_constraintRight_toRightOf	Phải với phải
app:layout_constraintTop_toTopOf	Cạnh trên với cạnh trên
app:layout_constraintTop_toBottomOf	Cạnh trên nối với cạnh dưới
app:layout_constraintBottom_toTopOf	Dưới với trên
app:layout_constraintBottom_toBottomOf	Dưới với dưới
app:layout_constraintBaseline_toBaselineOf	Trùng Baseline
app:layout_constraintStart_toEndOf	Bắt đầu - Kết thúc
app:layout_constraintStart_toStartOf	Bắt đầu - Bắt đầu
app:layout_constraintEnd_toStartOf	Cuối với bắt đầu
app:layout_constraintEnd_toEndOf	Cuối với cuối

3.2.4 TableLayout

Với TableLayout nó sẽ sắp xếp các View con bên trong thành dạng bảng. Mỗi hàng là một đối tượng view TableRow bên trong TableRow chứa các View con, mỗi View con này nằm ở vị trí một ô bảng (cell). Cột / hàng trong bảng bắt đầu từ số 0.

Số dòng của giao diện chính là số TableRow, số cột cột của giao diện là số view nhiều nhất trên 1 TableRow.



- Độ rộng của cột rộng bằng phần tử lớn nhất trong cột
- Các View trong TableRow không cần thiết lập chiều cao, chiều rộng vì nó sẽ tự động thiết lập rộng là match_parent, cao là wrap_content.
- Mặc dù phần tử con trực tiếp của TableLayout thường dùng chỉ là các TableRow, tuy nhiên bạn có thể đặt View con bất kỳ trực tiếp chứ không bên trong TableRow, lúc này View đó coi như một hàng và độ rộng mở rộng hết tất cả các cột.

Các thuộc tính thường dùng của TableLayout

Thuộc tính	Ý nghĩa
android:layout_width	Độ rộng
android:layout_height	Độ cao
android:layout_column	Thiết lập cột cho View
android:stretchColumns	Thiết lập độ rộng của cột được dãn tối đa. Ký hiệu * cho tất cả các cột hoặc nhập số để chỉ định cột được dãn. Ví dụ: android:stretchColumns = "1" //Cột 1 được dãn ra.
android:shrinkColumns	Thiết lập độ rộng của cột được co lại nhỏ nhất trong giao diện. Ký hiệu * cho tất cả các cột hoặc nhập số để chỉ định cột được co lại. Ví dụ: android:shrinkColumns = "2" //Cột 2 được co lại.

Bài tập:

Dùng TableLayout thiết kế giao diện sau:

3.2.5 Frame Layout

FrameLayout là loại View cơ sở, nó là loại Layout đơn giản nhất. Mặc dù nó có thể chứa nhiều View con bên trong, nhưng mục đích chính thiết kế ra nó để chứa một View, từ đó nó trở thành cơ sở để tạo ra các View khác phức tạp hơn. Khi thiết kế Layout chứa nhiều View thì không nên sử dụng layout này, vì nó quá đơn giản việc bố cục các View con trong nó rất khó khăn (nó không có các tính năng điều khiển vị trí View con sao cho việc độc lập về màn hình được đảm bảo).

Nếu bạn vẫn sử dụng FrameLayout để thiết kế layout, thì cần lưu ý: Các View con đặt vào FrameLayout nằm chồng nên nhau theo thứ tự cái nào đưa vào sau thì hiện thị ở lớp trước, mỗi View con chỉ có thể điều chỉnh vị trí nó thông qua thuộc tính android:layout_gravity gán cho View con. Và có thể tinh chỉnh khoảng cách theo hướng cách cánh bằng các thuộc tính liên quan đến **margin** như: android:layout_margin, android:layout_marginTop, android:layout_marginBottom, android:layout_marginLeft, android:layout_marginRight, android:layout_marginStart, android:layout_marginEnd, android:layout_marginHorizontal, android:layout_marginVertical.

Thuộc tính android:layout_gravity trong các View con

Khi các View nằm trong FrameLayout thì khi gán thuộc tính **android:layout_gravity** gán các giá trị ở bảng sau vị trí của nó thay đổi tương ứng:

Các giá trị có thể kết hợp bằng ký hiệu |

giá trị	Vị trí của View con
bottom	Nằm dưới FrameLayout
center	Nằm giữa FrameLayout
center_horizontal	Giữa theo chiều ngang
center_vertical	Giữa theo chiều đứng
end	Cuối FrameLayout
left	Bên trái
right	Bên phải
start	Bắt đầu FrameLayout
top	Trên FrameLayout

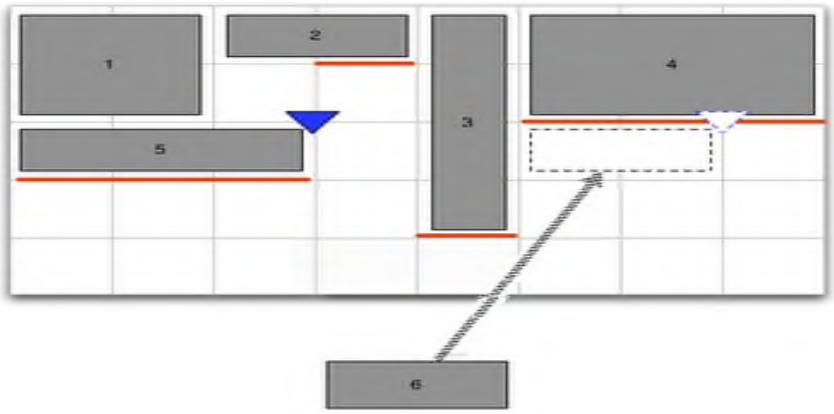
Ví dụ minh hoạ:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">


    <ImageView
        android:id="@+id/imageView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="centerCrop"
        app:srcCompat="@mipmap/nhatrang" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center|left"
        android:layout_marginLeft="20dp"
        android:text="Button"
        android:textColor="@color/colorAccent"
        android:textSize="18sp" />

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="149dp"
        android:layout_height="163dp"
        android:adjustViewBounds="true"
        android:layout_gravity="center|bottom"
        app:srcCompat="@mipmap/haisan" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center|top"
        android:text="Test Frame Layout"
        android:textColor="@color/colorAccent"
        android:textSize="30sp"
        android:textStyle="bold" />

</FrameLayout>
```

3.2.6 GridLayout

GridLayout sử dụng một mạng lưới các dòng mỏng và vô hạn để tách khu vực bản vẽ của nó thành: các hàng, các cột, và các ô (cell). Nó hỗ trợ cả việc bắc qua (span) các hàng và các cột, nghĩa là cho phép hợp nhất ô gần nhau thành một ô lớn (hình chữ nhật) để chứa một View.



Minh hoạ cách bố trí các View trên GridLayout

Trong GridLayout, việc chỉ định kích thước và căn lề làm giống với LinearLayout. Căn chỉnh/trọng lượng (Alignment/gravity) cũng làm việc giống như trọng lực (gravity) trong LinearLayout và sử dụng chung các hằng số: left, top, right, bottom, center_horizontal, center_vertical, center, fill_horizontal, fill_vertical và fill.

Bảng thuộc tính thường dùng của GridLayout

Thuộc tính	Ý nghĩa
android:layout_width	Độ rộng
android:layout_height	Độ cao
android:layout_columnWeight	Khối lượng theo cột (column) của vật thể trong ô, nó có ảnh hưởng tới việc chiếm chỗ theo cột, mặc định giá trị này là 0.
android:layout_rowWeight	Khối lượng theo hàng (row) của vật thể trong ô, nó có ảnh hưởng tới không gian chiếm chỗ theo hàng, mặc định giá trị này là 0.
android:layout_column	Xác định các vị trí của View trên lưới theo cột
android:layout_row	Xác định các vị trí của View trên lưới theo dòng
android:layout_columnSpan	Cho phép hợp nhất các ô trên các cột liền nhau thành 1 ô.
android:layout_rowSpan	Cho phép hợp nhất các ô trên các dòng liền nhau thành 1 ô.

3.3 Hàm findViewById

Khi lập trình Android, để tương tác với các View/Control trong giao diện chúng ta thường thông qua thuộc tính Id của các view/control để truy suất thay đổi dữ liệu. Vì Android chia màn hình (Activity) thành hai phần: Phần thiết giao diện, phần xử lý nghiệp vụ. Do đó để truy suất được tới các View trong phần giao diện Android cung cấp hàm findViewById.

Mỗi View trong giao diện tương tác với người dùng, khi chúc ta đặt Id thì Android sẽ tự động phát sinh trong lớp R. Vì vậy, khi truy xuất đến một View bất kỳ, chúng ta truy xuất qua R.id.Viewid bằng hàm findViewById.

Ví dụ: truy xuất tới một View Button trên layout

Phần giao diện trên XML như sau:

```
<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:id="@+id/btnTest"
   android:text="Click Me"
 />
```

Phần xử lý nghiệp vụ(Code Java) như sau:

```
Button btnButton = (Button) findViewById(R.id.btnTest);
```

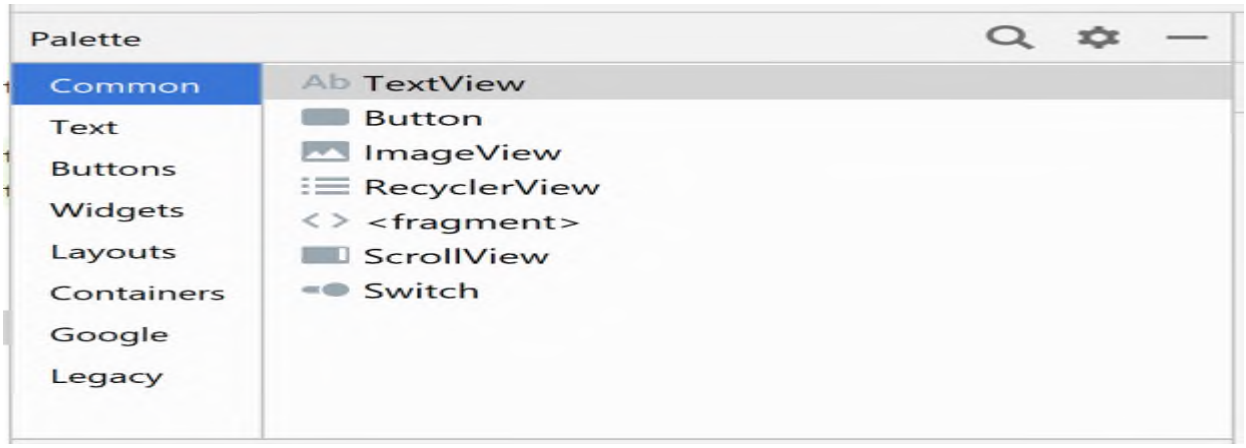
3.4 Các View trong Android

Android cung cấp rất nhiều View/Control để ta có thể tùy chọn thiết kế giao diện tương tác người dùng. TextView dùng để hiển thị dữ liệu, EditText để hiển thị và cho phép thay đổi thông tin, Button để ra lệnh, CheckBox để chọn nhiều lựa chọn, RadioButton để chọn một lựa chọn, ImagView để hiển thị hình ảnh,…

3.4.1 TextView

TextView là một View cho phép hiện thị các dòng chữ (text) trên màn hình, nó có nhiều thuộc tính tùy mục đích sử dụng mà áp dụng, như thiết lập cỡ chữ, font chữ, màu chữ, ...

TextView thuộc nhóm Common và Text trong công cụ Palette:



Ta có thể kéo thả các View vào giao diện thiết kế một cách dễ dàng.

Các thuộc tính/sự kiện quan trọng của TextView

Các thuộc tính/sự kiện	Mô tả
android:id	Id của TextView
android:layout_width	Độ rộng
android:layout_height	Độ cao
android:text	Chữ hiển thị lên giao diện
android:textColor	Màu chữ
android:textSize	Cỡ chữ
android:background	Màu nền
android:hint	Chữ gợi ý khi android:text rỗng

Để truy suất TextView ta cần đặt Id cho nó trong thanh công cụ Properties hoặc trong XML.

TextView txtMessage=findViewById(R.id.txtMessage);

Để thiết lập nội dung hiển thị ta có hai cách:

- Trong Java code (Activity):

  txtMessage.setText(“ Xin Chào Các Bạn! ”);

- Trong XML:

  android:text=“ Xin Chào Các Bạn! ”

```
<TextView
   android:id="@+id/txtMessage"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:background="@android:color/holo_red_light"
   android:textColor="@android:color/holo_blue_light"
   android:text="Khoa Hệ Thống Thông Tin"
   android:textSize="30sp"
/>
```

Ngoài id, text thì TextView còn có các thuộc tính quan trọng khác như: layout_width(thiết lập độ rộng), layout_height(thiết lập chiều cao), background(màu nền), textColor(màu chữ), textSize(cỡ chữ). Các thuộc tính cho TextView rất nhiều và được cung cấp sẵn trong công cụ Properties ta có thể vào đây để thao tác.

Các phương thức của TextView thường dùng:

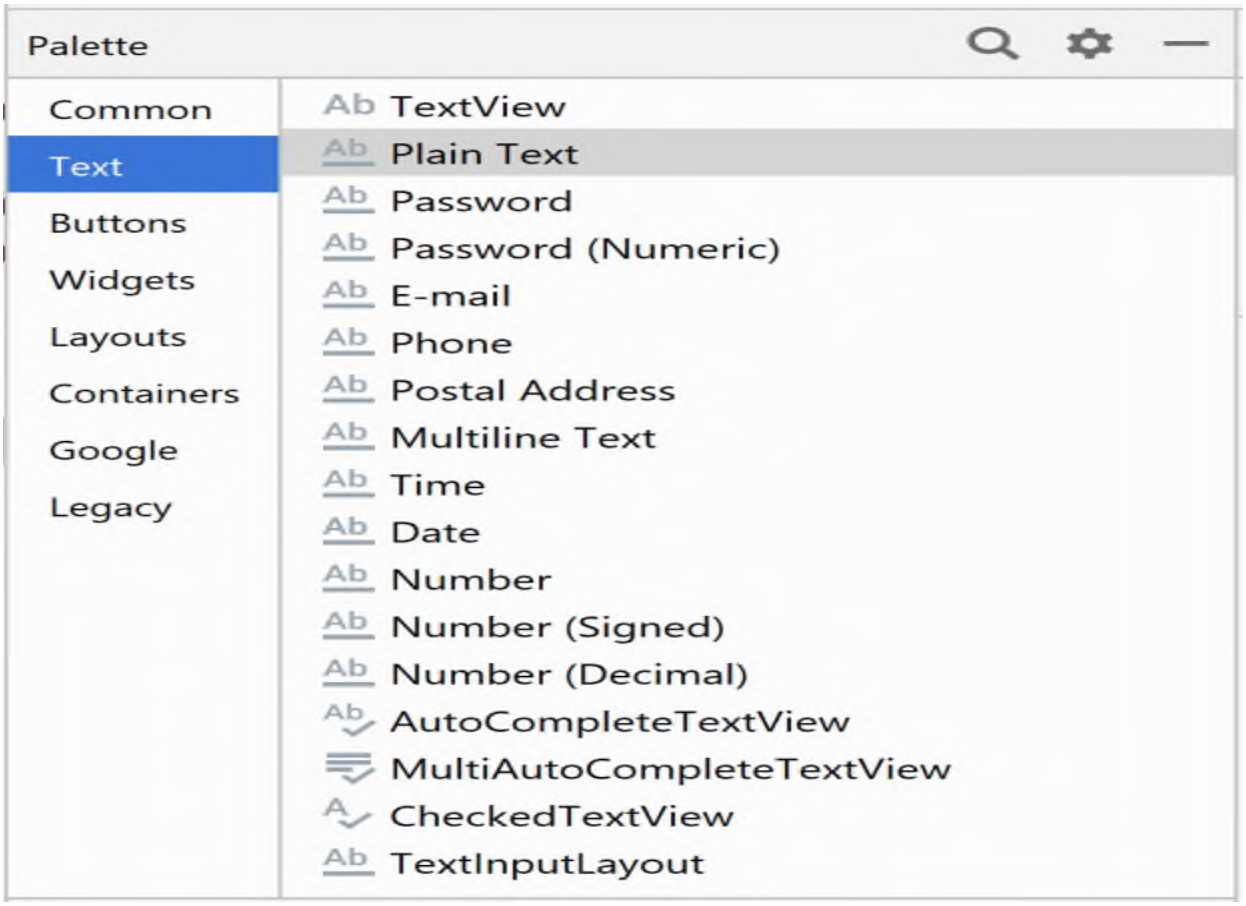
Phương thức	Mô tả
setText()	Hiển thị thông tin lên Control TextView Ví dụ: TxtView.setText(“ Xin chào!”);
getText()	Lấy thông tin bên trong Control TextView Ví dụ: String Msg = TxtView.getText().toString();
setTextSize(float)	Thiết lập cỡ chữ trong Control TextView Ví dụ: TxtView.setTextSize(16); // 16dp

setTypeface()	Thiết lập đinh dạng chữ
	Ví dụ: Thiết lập định dạng chữ đậm nghiêng
	mytextview.setTypeface(null, Typeface.BOLD_ITALIC);
	//Font chữ "serif-monospace" - Đậm
	mytextview.setTypeface(Typeface.create("serif-monospace", Typeface.BOLD));

3.4.2 EditText

EditText là lớp kế thừa từ TextView, được dùng thay đổi nội dung text, chứa tất cả thuộc tính của TextView nên TextView hoạt động như thế nào thì EditText hoạt động như vậy, nằm trong nhóm Text của Palette. Đặt Id nên bắt đầu là là edt hoặc txt

Android đã tự động phân EditText thành nhiều loại khác nhau: Nhập chuỗi, số, điện thoại, email, mật khẩu … dựa vào thuộc tính inputType. Nhờ thuộc tính inputType mà trải nghiệm người dùng được tốt hơn, điện thoại sẽ hiển thị bàn phím tương ứng với inputType mà ta cấu hình giúp người dùng thao tác một cách dễ dàng và nhanh chóng:



EditText có nhiều thuộc tính quan trọng ta cần quan tâm:

Các thuộc tính/sự kiện quan trọng của EditText

Các thuộc tính/sự kiện	Mô tả
android:id	Id của EditText
android:layout_width	Độ rộng

android:layout_height	Độ cao
android:text	Chữ hiển thị lên giao diện
android:textColor	Màu chữ
android:textSize	Cỡ chữ
android:background	Màu nền
android:hint	Chữ gợi ý khi android:text rỗng
android:inputType	Thiết lập loại dữ liệu nhập để có bàn phím phù hợp

Ta thấy trong nhóm Text rất nhiều View (EditText), tùy vào nhu cầu sử dụng mà ta kéo thả các EditText phù hợp ra.

Cấu trúc XML của Edittext:

```
< EditText
        android:id="@+id/edtPassword"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="1"
        android:textColor="@android:color/holo_blue_light"
        android:inputType="textPassword"
        android:text="hoilamgi"
 />
```

Thuộc tính inPutType= “textPassword” giúp ta khi nhập dữ liệu vào thì phần mềm tự động mã hóa thành các ký tự * để bảo mật. Edittext ngoài ra còn sử dụng Hint để nhắc nhớ người dùng, ta có thể bổ sung android:hint = “nhập mật khẩu ở đây”, người dùng sẽ biết ô này dùng để làm gì.

Để truy suất ta cũng dùng hàm findViewById

EditText edtPassword=findViewById(R.id.edtPassword);

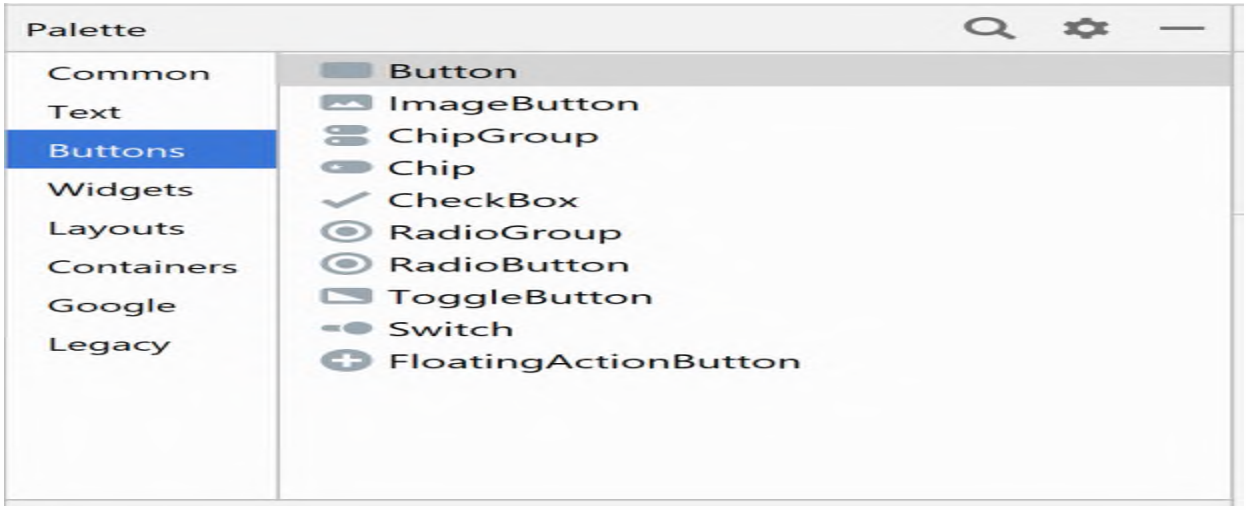
Để thay đổi nội dung hiển thị ta dùng: edtPassword.setText(“obama”)

Để lấy nội dung người dùng nhập liệu: edtPassword.getText().toString()

3.4.3 Button

Đối tượng Button được xây dựng từ TextView, cho phép thể hiện các nội dung văn bản, hình ảnh – nhận và phản hồi tương tác nhấn từ người dùng. Nên đặt Id cho Button bắt đầu bằng btn

Button thuộc nhóm Buttons hoặc Common trong công cụ Palette:



Các thuộc tính trong Button tương tự như trong TextView và EditText.

Các thuộc tính/sự kiện quan trọng của Button.

Các thuộc tính/sự kiện	Mô tả
android:id	Id của EditText
android:layout_width	Độ rộng
android:layout_height	Độ cao
android:text	Chữ hiển thị lên giao diện
android:textColor	Màu chữ
android:textSize	Cỡ chữ
android:background	Màu nền
android:onClick	Gán sự kiện cho Button

Cấu trúc XML của Button:

```
<Button
    android:id="@+id/btnDangNhap"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Đăng Nhập"
/>
```

Button thường dùng để cho phép người dùng ra lệnh thực thi một nghiệp vụ nào đó, sự ra lệnh này gọi là các sự kiện (event). Button ngoài hiển thị dữ liệu còn có nhiệm vụ lắng nghe các sự kiện người dùng, hai cách nhanh nhất để lắng nghe sự kiện:

- Lắng nghe trong XML: android:onClick=“tenPhuongThuc”
- Lắng nghe trong java code:

btnDangNhap.setOnCLickListener(new OnClickLisntener() {

@Override

public void onClick(View v) {

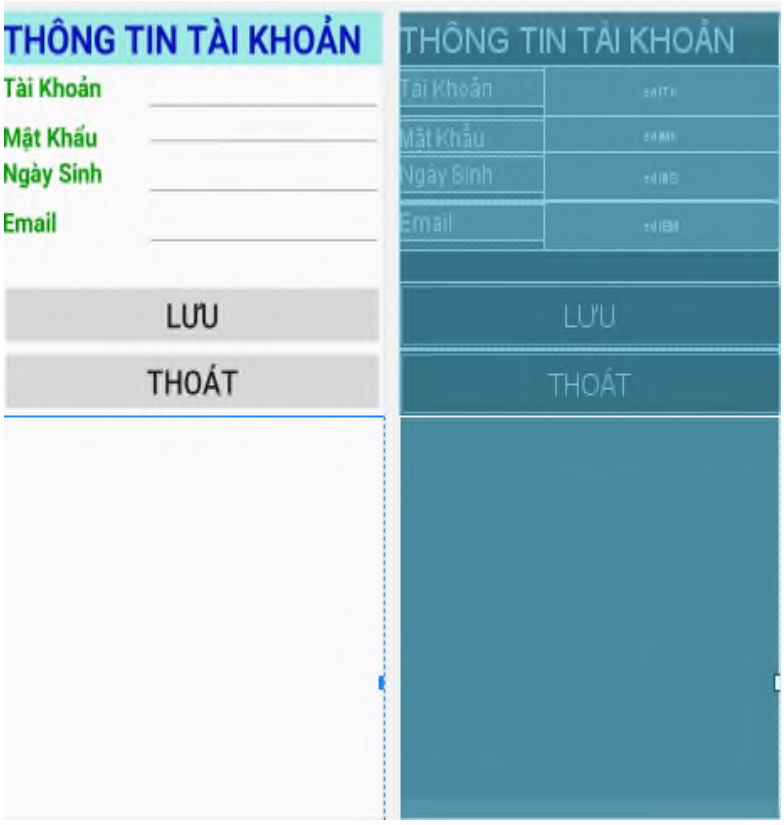
xuLyDangNhap();

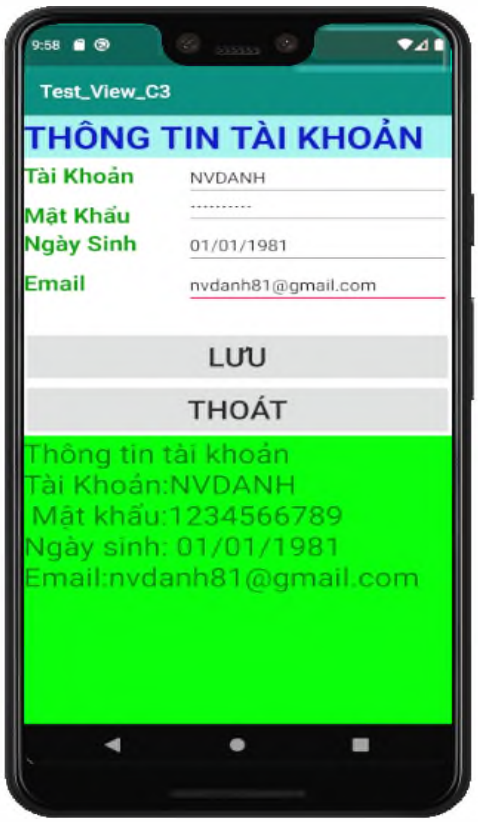
}

});

Ví dụ:

Thiết kế giao diện và thực hiện chức năng sau:

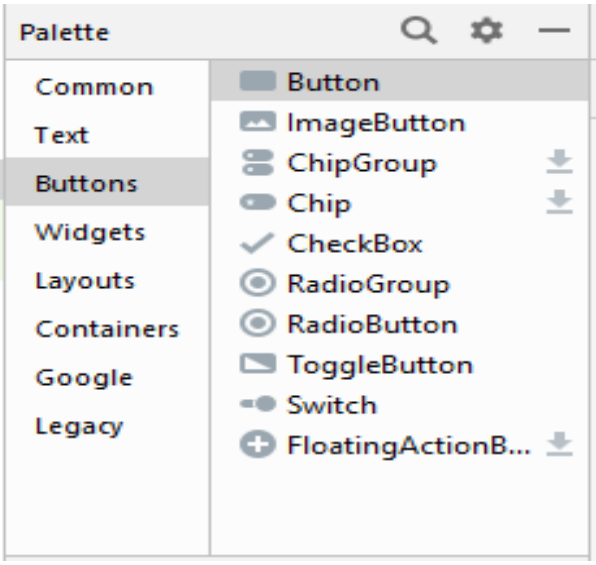




MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    Button btnThem,btnThoat;
    EditText edtTK, edtMK, edtNS, edtEM;
    TextView txtKQ;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        addControl();
        addEvents();
    }

    private void addEvents() {
        btnThoat.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
            finish();
        }
    });

    btnThem.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String str="Thông tin tài khoản \n";
            str+="Tài Khoản:"+edtTK.getText().toString()+"\n Mật
khẩu:"+edtMK.getText().toString();
            str+="\nNgày sinh:
"+edtNS.getText().toString()+"\nEmail:"+edtEM.getText().toString();
            txtKQ.setText(str);

txtKQ.setBackgroundColor(Color.GREEN);
        }
    });
}


  private void addControl() {
    btnThem =
this.<Button>findViewById(R.id.btnThem);
    edtEM =
this.<EditText>findViewById(R.id.edtEM);
    edtMK =
this.<EditText>findViewById(R.id.edtMK);
    edtNS = this.<EditText>findViewById(R.id.edtNS);
    edtTK = this.<EditText>findViewById(R.id.edtTK);
    txtKQ = this.<TextView>findViewById(R.id.txtKQ);
    btnThoat = this.<Button>findViewById(R.id.btnThoat);
  }

}
```



3.4.4 CheckBox

CheckBox mở rộng từ CompoundButton là một loại Button cho phép hiển thị hộp kiểm tra, hiển thị thông tin trạng thái checked hay unchecked, mà để tùy biến CheckBox.

CompoundButton là một lớp cơ abstract mở rộng từ TextView (Button), từ lớp này nó được mở rộng để xây dựng các View là: **CheckBox, RadioButton, Switch, ToggleButton**

CheckBox thuộc nhóm Buttons trong công cụ Palette.

Cú pháp khai báo CheckBox trong XML có dạng:

```
<CheckBox
```

```
   android:id="@+id/checkbox_id"
   android:text="Java - Android"
   android:checked="true"  <!--Hoặc "false" -->
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
/>
```

CheckBox kế thừa từ TextView, Button nên các thuộc tính thiết lập màu nền, màu chữ, Drawable, … cho CheckBox tương tự như TextView và Button. Ngoài các thuộc tính được kế thừa đó, còn một số thuộc tính cần lưu ý:

Các thuộc tính/sự kiện	Mô tả
android:id	Id của CheckBox
android:layout_width	Độ rộng
android:layout_height	Độ cao
android:text	Chữ hiển thị lên giao diện CheckBox
android:checked	True/False. Dùng để gán giá trị cho CheckBox android:checked = “True” // Chọn android:checked = “False” // Không chọn
android:unchecked	True/False. Dùng để gán giá trị cho CheckBox android:unchecked = “False” // Chọn android:unchecked = “True” // Không chọn

Các phương thức thường dùng của CheckBox

Phương thức	Mô tả

ischecked()	Kết quả trả về là True/False. Dùng để kiểm tra trạng thái là checked (True) hay uncheck (False)
	`CheckBox chk=(CheckBox) findViewById(R.id.checkBox1);`
	`if(chk.isChecked())`
	`{`
	`  //xử lý checked`
	`}`
	`else`
	`{`
	`  //xử lý unchecked`
	`}`
	`//Muốn thiết lập checked:`
	`chk.setChecked(true);`
	`//Muốn clear checked:`
	`chk.setChecked(false);`

setChecked()	Gán giá trị cho CheckBox.
	chk.setChecked(true); // Chọn
	chk.setChecked(false); // Bỏ chọn

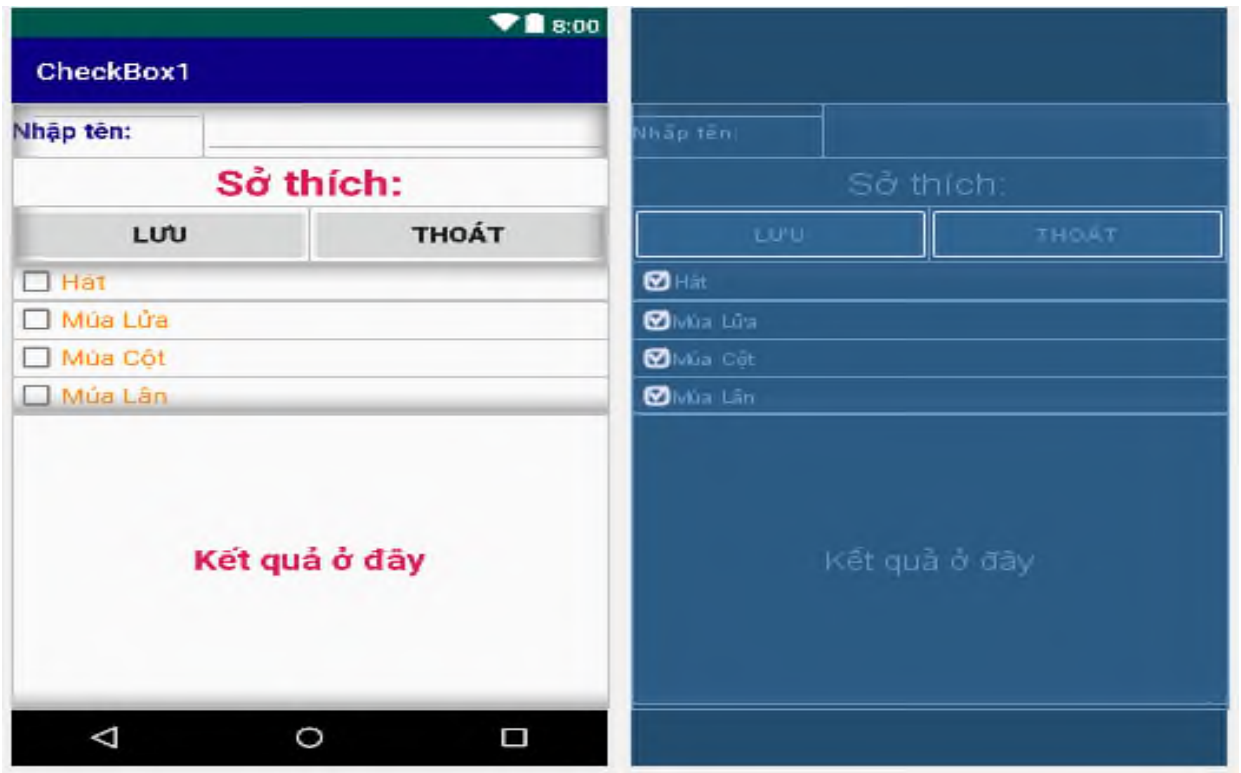
Sự kiện của CheckBox thường dùng

Sự kiện	Mô tả

OnCheckedChanged	Bắt sự kiện khi trạng thái thay đổi.
	Code trong Java
	CheckBox chk = (CheckBox) findViewById(R.id.chk1);
	
	chk.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
	@Override
	public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
	if(isChecked) {
	// Xử lý khi chọn
	} else {
	// Xử lý khi bỏ chọn
	}
	}
	});

Ví dụ:

Thiết kế giao diện như sau



Bước 2: Khai báo các thành phần

```
EditText edtTen;
CheckBox chkHat,chkMuaLua,chkMuaCot,chkMuaLan;
Button btnLuu,btnThoat;
TextView txtKQ;

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

Bước 3: Viết phương thức addControls()

```
private void addControls() {
    edtTen = this.<EditText>findViewById(R.id.edtTen);
    chkHat = this.<CheckBox>findViewById(R.id.chkHat);
    chkMuaCot  = this.<CheckBox>findViewById(R.id.chkMuaCot);
    chkMuaLan = this.<CheckBox>findViewById(R.id.chkMuaLan);
    chkMuaLua = this.<CheckBox>findViewById(R.id.chkMuaLua);
    btnLuu = this.<Button>findViewById(R.id.btnLuu);
    btnThoat = this.<Button>findViewById(R.id.btnThoat);
    txtKQ = this.<TextView>findViewById(R.id.txtKQ);
}
```

Bước 4: Viết phương thức addEvents()

```
private void addEvents() {
    btnLuu.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            xuLyLuu();
        }
    });

    btnThoat.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            finish();
        }
    });
}
```

Bước 5: Viết phương thức xuLyLuu()

```
private void xuLyLuu() {
    String kq = "";
    kq+=edtTen.getText();
    kq = kq+" "+"Có các sở thích sau: ";
    if(chkHa
        kq=k
    if (chkMuaLua.isChecked())
        kq=kq+"\n"+chkMuaLua.getText();
    if(chkMuaLan.isChecked())
        kq=kq+"\n"+chkMuaLua.getText();
    if(chkMuaCot.isChecked())
        kq = kq+"\n"+chkMuaCot.getText();
    txtKQ.setText(kq);
}
```

Typo: In word 'thích' more... (Ctrl+F1)

Hoàn thiện bài và thực thi kết quả

Ví dụ 2: Cải thiện cách viết thứ 2:

```
private void xuLyLuu() {

    kq+=edtTen.getText();
    kq = kq+" "+"Có các sở thích sau: ";
    if(chkHat.isChecked())
        kq=kq+"\n"+chkHat.getText();
    if (chkMuaLua.isChecked())
        kq=kq+"\n"+chkMuaLua.getText();
    if(chkMuaLan.isChecked())
        kq=kq+"\n"+chkMuaLua.getText();

    chkMuaCot.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
            if(b==true)
                kq = kq +"\n"+chkMuaCot.getText();
        }
    });
        kq = kq+"\n"+chkMuaCot.getText();
    txtKQ.setText(kq);
}
```

3.4.5 RadioGroup

RadioGroup là một bộ chứa (container) nó có thể chứa các RadioButton, khi bạn check chọn một radio button trong một nhóm, tất cả các radio button khác trong nhóm sẽ bị mất lựa chọn (deselect).

RadioGroup thuộc nhóm Buttons trong công cụ Palette.

Cú pháp khai báo RadioGroup trong XML có dạng:

```
<RadioGroup
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:orientation = "vertical"
</RadioGroup>
```

RadioGroup kế thừa từ TextView, Button nên các thuộc tính thiết lập màu nền, màu chữ, Drawable, … cho RadioGroup tương tự như TextView và Button. Ngoài các thuộc tính được kế thừa đó, còn một số thuộc tính cần lưu ý:

Các thuộc tính/sự kiện	Mô tả
android:orientation	Bố trí các layout của RadioGroup theo chiều ngang(horizontal) hay chiều dọc(vertical).
android:checkedButton	Thiết lập các giá trị checked cho RadioButton. Trong một nhóm các RadioButton(bên trong RadioGroup) chúng ta thiết lập một RadioButton ở trạng thái checked mặc định bằng cách dùng thuộc tính của RadioGroup android:checkButton = "@id/radioid"

3.4.6 RadioButton

Radiobutton thường được đưa ra 2 hoặc nhiều hơn hai phần tử trong đó người dùng chỉ được chọn một phần tử, để làm được như vậy chúng ta cần nhóm chúng vào một nhóm đó chính là RadioGroup để khi người dùng chọn thì chỉ chọn được duy nhất.

Dưới đây là hình ảnh các RadioButton được nhóm lại trong các nhóm khác nhau.

RadioGroup có thể sắp xếp các RadioButton theo chiều dọc hoặc chiều ngang.

RadioButton thuộc nhóm Buttons trong công cụ Palette.

Cú pháp khai báo RadioButton trong XML có dạng:

```
<RadioButton
 android:id="@+id/radioButton_nam"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Nam"
/>
```

RadioButton kế thừa từ TextView, Button nên các thuộc tính thiết lập màu nền, màu chữ, Drawable, … cho RadioButton tương tự như TextView và Button. Ngoài các thuộc tính được kế thừa đó, còn một số thuộc tính cần lưu ý:

Các thuộc tính/sự kiện	Mô tả
android:id	Id của RadioButton
android:layout_width	Độ rộng
android:layout_height	Độ cao
android:text	Chữ hiển thị lên giao diện RadioButton
android:checked	True/False. Dùng để gán giá trị cho RadioButton android:checked = “True” // Chọn android:checked = “False” // Không chọn
android:unchecked	True/False. Dùng để gán giá trị cho RadioButton android:unchecked = “False” // Chọn android:unchecked = “True” // Không chọn
android:buttonTint	Thay đổi màu sắc icon trạng thái của RadioButton

Các phương thức thường dùng của RadioButton

Phương thức	Mô tả

ischecked()	Kết quả trả về là True/False. Dùng để kiểm tra trạng thái là checked (True) hay uncheck (False)
	`Final RadioButton radiobtn = findViewById(R.id.radiobtn_id);`
	`//Kiểm tra checked`
	`if (radiobtn.isChecked())`
	`{`
	`  // Checked`
	`}`
	`else`
	`{`
	`  //UnChecked`
	`}`

setChecked()	Gán giá trị cho CheckBox.
	radiobtn.setChecked(true); // Chọn
	radiobtn.setChecked(false); // Bỏ chọn

Sự kiện của CheckBox thường dùng

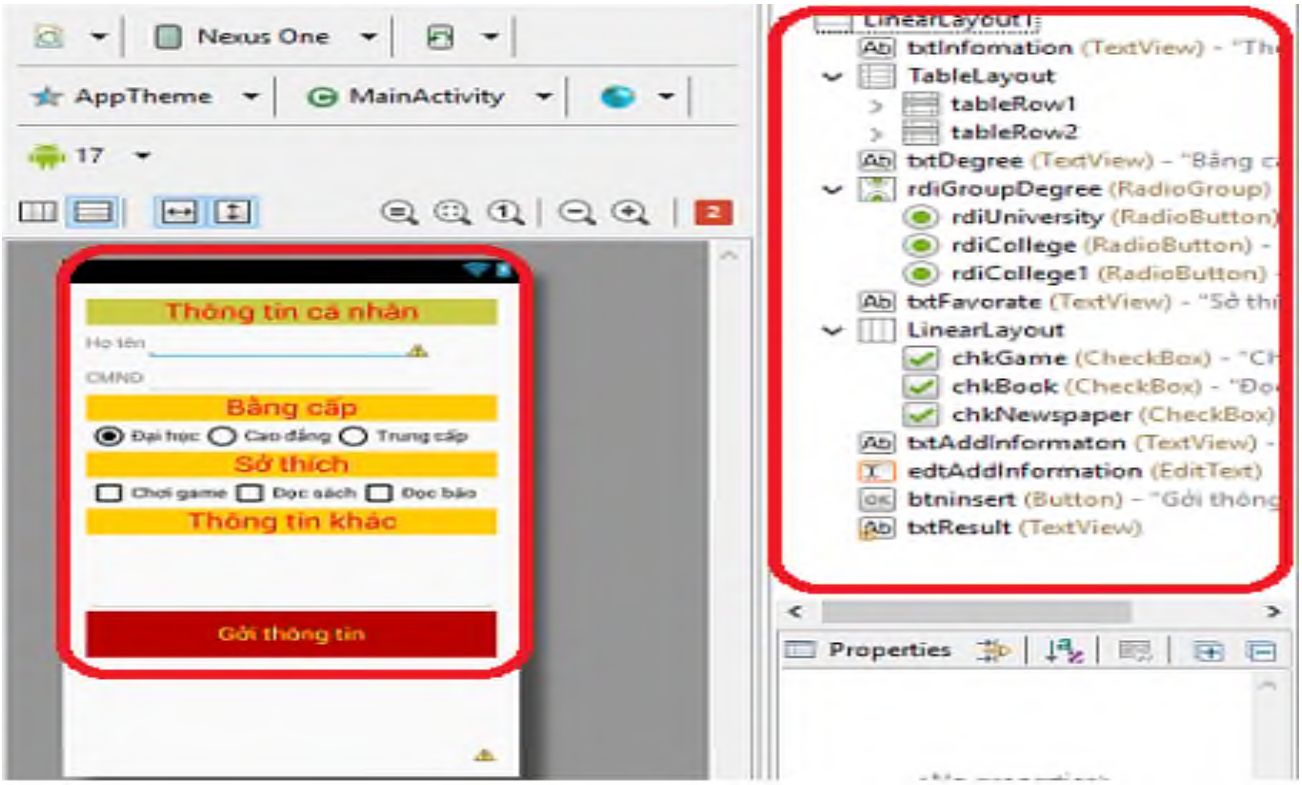
Sự kiện	Mô tả
OnCheckedChanged	Bắt sự kiện khi trạng thái thay đổi. Code trong Java RadioButton radiobtn = (RadioButton) findViewById(R.id. radiobtn_id);

	radiobtn.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
	  @Override
	  public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
	  if(isChecked) {
	  // Xử lý khi chọn
	  } else {
	  // Xử lý khi bỏ chọn
	  }
	  }
	});

Ví dụ 1: Yêu cầu ứng dụng

- Tên người không được để trống và phải có ít nhất 3 ký tự
- Chứng minh nhân dân chỉ được nhập kiểu số và phải có đúng 9 chữ số
- Bằng cấp mặc định sẽ chọn là Đại học
- Sở thích phải chọn ít nhất 1 chọn lựa
- Thông tin bổ sung có thể để trống
- Khi bấm gửi thông tin, chương trình sẽ hiển thị toàn bộ thông tin cá nhân cho người sử dụng biết thông qua TextView:

Thiết kế:



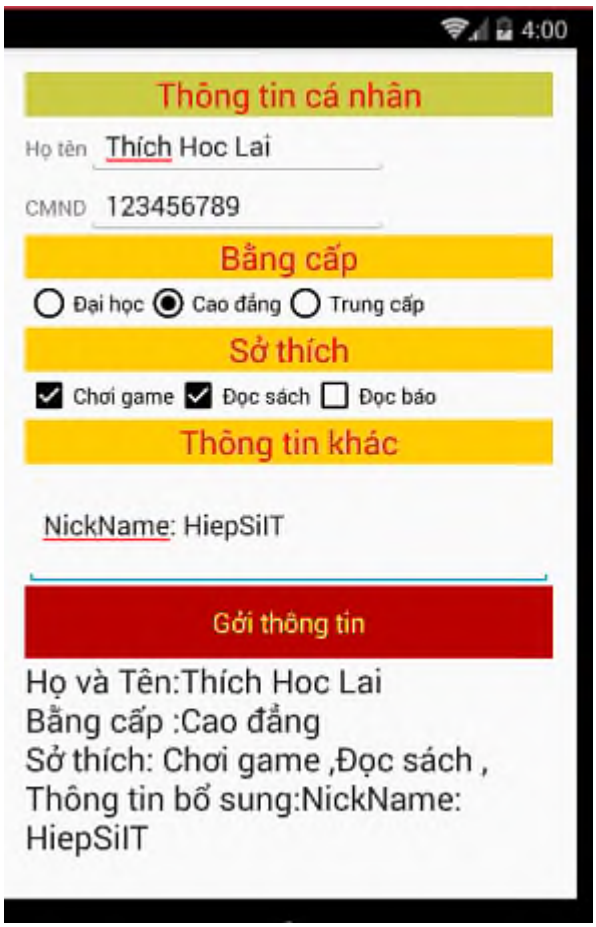
Đặt tên

Điều khiển	android:id	android:text
TextView1	txtInformation	@string/information
TextView2	txtDegree	@string/degree
TextView3	txtFavorate	@string/favorate
TextView4	txtAddInformation	@string/addinformation
TextView5	txtResult	
EditText1	edtFirstName	@string/firstname
EditText2	edtID	@string/id
RadioButton1	rdiUniversity	@string/university
RadioButton2	rdiCollege	@string/college
RadioButton3	rdiCollege	@string/college1
CheckBox1	chkGame	@string/game
CheckBox2	chkBook	@string/book
CheckBox3	chkNewspaper	@string/newspaper
Button1	btnInsert	@string/btninsert

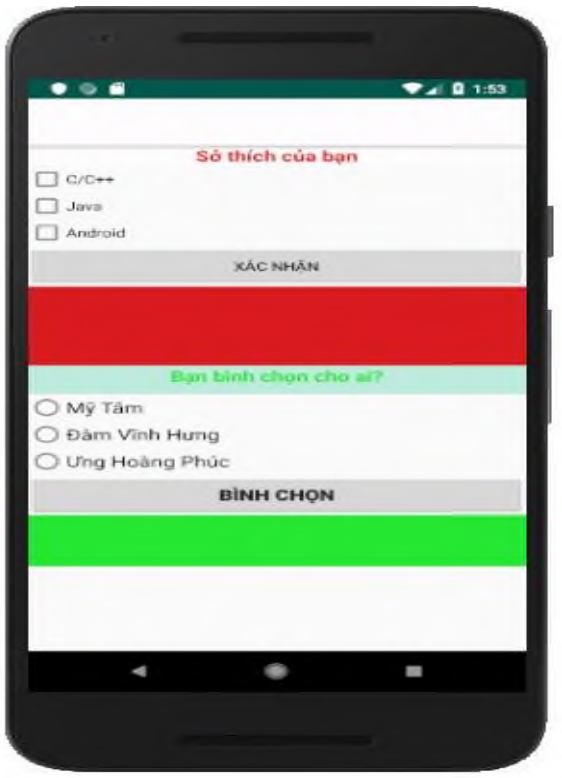
MaiActivity.java

```
public class MainActivity extends Activity {
	EditText edtFirstName, edtID, edtInformation;
	CheckBox chkGame, chkBook, chkNewspaper;
	TextView txtResult;
	Button btnInsert;
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);
		edtFirstName=(EditText)  findViewById(R.id.edtFirstName);
		edtID  =(EditText)  findViewById(R.id.edtID);
		edtInformation=(EditText) findViewById(R.id.edtAddInformation);
		chkGame =(CheckBox)  findViewById(R.id.chkGame);
		chkBook=(CheckBox)  findViewById(R.id.chkBook);
		chkNewspaper  =(CheckBox)findViewById(R.id.chkNewspaper);
		btnInsert  =(Button)  findViewById(R.id.btninsert);
		txtResult  =(TextView) findViewById(R.id.txtResult);
		btnInsert.setOnClickListener(new OnClickListener() {

			@Override
			public void onClick(View v) {
				// TODO Auto-generated method stub
				showInformation();
			}
		});


	}
	// Lấy thông tin người sử dụng nhập vào
	private void showInformation(){
		try{
			String message ="";
			//Kiểm tra họ và tên
			String firstname  =  edtFirstName.getText().toString().trim();
			if(firstname.length()<=3){
				edtFirstName.selectAll();
				edtFirstName.requestFocus();
				Toast.makeText(getApplication(), "Họ và tên không nhỏ hơn 3
ký tự", Toast.LENGTH_LONG).show();
				return;
			}
			message +="Họ và Tên:" +firstname +"\n";
			String strID  =  edtID.getText().toString().trim();
			if(strID.length()!=9){
				edtID.selectAll();
				edtID.requestFocus();
```

```
                    Toast.makeText(getApplication(), "Số chứng minh nhân dân phải 9 số", Toast.LENGTH_LONG).show();
                    return;
                }
                String degree;
                RadioGroup rdigroupDegree=(RadioGroup) findViewById(R.id.rdiGroupDegree);
                int id  =rdigroupDegree.getCheckedRadioButtonId();

                if(id==-1){
                    Toast.makeText(getApplication(), "Xin vui lòng chọn bằng cấp", Toast.LENGTH_LONG).show();
                    return;
                }
                RadioButton  radDegree=  (RadioButton) findViewById(id);
                degree=       radDegree.getText()+"";
                message += "Bằng cấp :"+ degree +"\n";
                String favorate="";
                if(chkGame.isChecked())
                    favorate+= chkGame.getText()+ " ,";
                if(chkBook.isChecked())
                    favorate+= chkBook.getText()+ " ,";
                if(chkNewspaper.isChecked())
                    favorate+=chkNewspaper.getText()+ " ";

                if(favorate.trim().length()==0){
                    Toast.makeText(getApplication(), "Xin vui một sở thích", Toast.LENGTH_LONG).show();
                    return;
                }
                else
                    message+="Sở thích: "+ favorate +"\n";
                if(edtInformation.getText().toString().trim().length()!=0)
                    message +=   "Thông tin bổ sung:" +edtInformation.getText().toString().trim();
                txtResult.setText(message);
            }
            catch(Exception e){
                Log.d("1", e.getMessage());
            }
        }

}
```



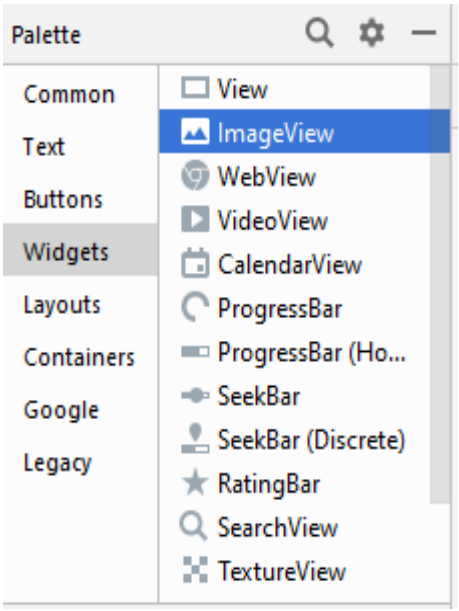
Ví dụ 2: Phân tích bố cục và thiết kế ứng dụng sau:



3.4.7 ImageView và ImageButton

ImageView là loại View dùng để hiện thị tài nguyên hình ảnh như các ảnh Bitmap, các ảnh Drawable. Nó cũng cung cấp các chức năng tùy biến khác nhau như đổ màu nhuộm (tint) vào ảnh, co kéo/cắt ảnh khi hiện thị trên View.

ImageView thuộc nhóm Common hoặc Widget trong công cụ Palette.



Cú pháp khai báo ImageView trong XML có dạng:

```
<ImageView
    android:id="@+id/image"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@mipmap/ic_launcher"
    />
```

Thuộc tính cơ bản của ImageView thường dùng:

Các thuộc tính/sự kiện	Mô tả
android:id	Id của ImageView

android:layout_width	Độ rộng
android:layout_height	Độ cao
android:src	Là thuộc tính chứa hình ảnh cần hiển thị.
android:background	Thuộc tính này xác định màu nền cho **ImageView**.
android:padding	Thuộc tính này xác định khoảng cách từ đường viền của **ImageView** với nội dung nó chứa: **left, right, top or bottom.** ▪ paddingRight: thiết khoảng cách bên phải của ImageView ▪ paddingLeft: thiết khoảng cách bên trái của ImageView. ▪ paddingTop: thiết khoảng cách phía trên của ImageView. ▪ paddingBottom: thiết khoảng cách phía bên dưới của ImageView. ▪ padding: thiết khoảng cách tất cả 4 phía của ImageView.

Thay đổi tỷ lệ căn chỉnh ảnh trong ImageView

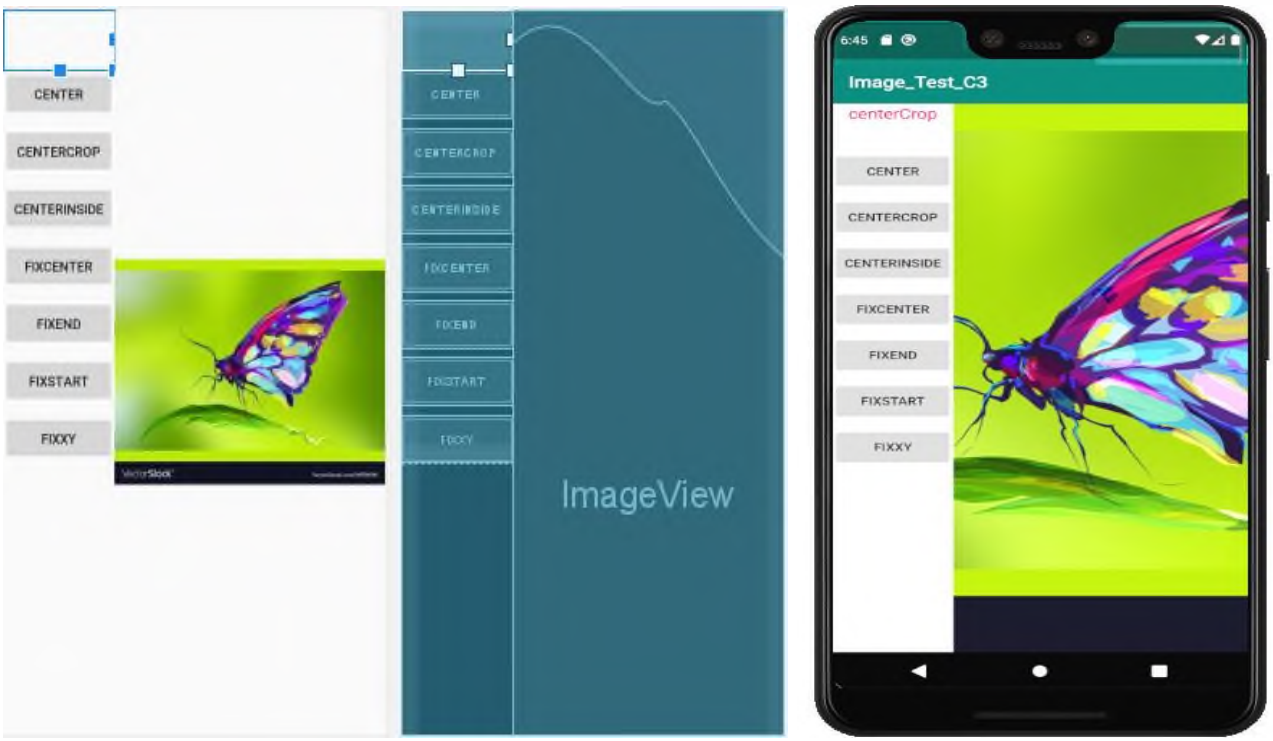
Thuộc tính **android:scaleType** dùng để thiết lập thu phóng ảnh, nhận các giá trị như: *fitXY, center, fitXY* ..., ví dụ như **android:scaleType="center"**

hoặc code Java: imageView.setScaleType(ImageView.ScaleType.CENTER);

Các hằng thuộc tính scaleType:

Giá trị	Sử dụng
center	ImageView.ScaleType.CENTER : Đặt ảnh vào giữa ImageView, không có thay đổi tỷ lệ ảnh.
centerCrop	ImageView.ScaleType.CENTER_CROP: Đặt ảnh vào giữa ImageView, có thu phóng ảnh (nhưng giữ nguyên tỉ lệ cao / rộng) sao cho ảnh phủ kín hết cả ImageView (phần thừa bị cắt)
centerInside	ImageView.ScaleType.CENTER_INSIDE: Đặt ảnh vào giữa ImageView, có thu phóng ảnh (nhưng giữ nguyên tỉ lệ cao / rộng) sao cho toàn bộ các phần của ảnh hiện thị trên ImageView.
fitCenter	ImageView.ScaleType.FIT_CENTER: Đặt ảnh vào giữa ImageView, có thu phóng ảnh (nhưng giữ nguyên tỉ lệ cao / rộng) sao cho toàn bộ các phần của ảnh hiện thị trên ImageView.
fitEnd fitStart	ImageView.ScaleType.FIT_END, ImageView.ScaleType.FIT_START: Co ảnh vừa View, vị trí ảnh ở cuối (ở đầu) ImageView
fitXY	ImageView.ScaleType.FIT_XY: Co ảnh vừa khít cả chiều rộng và cao.

Ví dụ 1: Thiết kế ứng dụng sau:



Thiết kế trong XML:

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

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <TextView
            android:id="@+id/txtKQ"
            android:layout_width="match_parent"
            android:layout_height="61dp"
            android:textAlignment="center"
            android:textColor="@color/colorAccent"
            android:textSize="18sp" />

        <Button
            android:id="@+id/btncenter"
            android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:text="center" />

    <Button
        android:id="@+id/btncenterCrop"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:text="centerCrop" />

    <Button
        android:id="@+id/btncenterInside"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:text="centerInside" />

    <Button
        android:id="@+id/btnfixCenter"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:text="fixCenter" />

    <Button
        android:id="@+id/btnfixEnd"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:text="fixEnd" />

    <Button
        android:id="@+id/btnfixStart"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:text="fixStart" />

    <Button
        android:id="@+id/btnfixXY"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:text="fixXY" />
</LinearLayout>

<ImageView
```

```
        android:id="@+id/imgHinh"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@drawable/hinh" />

</LinearLayout>
```

Sử dụng Activity As Listener

MainActivity.java

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener
{


    TextView txtKQ;
    Button
btncenter,btncenterCrop,btncenterInside,btnfixCenter,btnfixEnd,btnfixStart,btnfixX
Y;
    ImageView imgHinh;

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

    private void addEvents() {
        btnfixXY.setOnClickListener(this);
        btnfixStart.setOnClickListener(this);
        btnfixEnd.setOnClickListener(this);
        btncenterInside.setOnClickListener(this);
        btncenterCrop.setOnClickListener(this);
        btnfixCenter.setOnClickListener(this);
        btncenter.setOnClickListener(this);
    }


    private void addControls() {
        btncenter = this.<Button>findViewById(R.id.btncenter);
        btncenterCrop = this.<Button>findViewById(R.id.btncenterCrop);
        btncenterInside = this.<Button>findViewById(R.id.btncenterInside);
        btnfixCenter = this.<Button>findViewById(R.id.btnfixCenter);
        btnfixEnd = this.<Button>findViewById(R.id.btnfixEnd);
        btnfixStart = this.<Button>findViewById(R.id.btnfixStart);
        btnfixXY = this.<Button>findViewById(R.id.btnfixXY);
```

```
        txtKQ = this.<TextView>findViewById(R.id.txtKQ);
        imgHinh = this.<ImageView>findViewById(R.id.imgHinh);



    }

    @Override
    public void onClick(View v) {
        ImageView.ScaleType scaletype  = ImageView.ScaleType.CENTER;
        switch (v.getId())
        {
            case R.id.btncenter:
                scaletype = ImageView.ScaleType.CENTER;
                break;
            case R.id.btncenterCrop:
                scaletype = ImageView.ScaleType.CENTER_CROP;
                break;
            case R.id.btncenterInside:
                scaletype = ImageView.ScaleType.CENTER_INSIDE;
                break;
            case R.id.btnfixCenter:
                scaletype = ImageView.ScaleType.FIT_CENTER;
                break;
            case R.id.btnfixEnd:
                scaletype = ImageView.ScaleType.FIT_END;
                break;
            case R.id.btnfixStart:
                scaletype = ImageView.ScaleType.FIT_START;
                break;
            case R.id.btnfixXY:
                scaletype = ImageView.ScaleType.FIT_XY;
                break;

        }

        imgHinh.setScaleType(scaletype);
        txtKQ.setText(((Button)v).getText());
    }
}
```

Chương 4. KỸ THUẬT LẬP TRÌNH SỰ KIỆN TRÊN VIEW

4.1. XMLListener

Khi bạn tạo một View trong layout, bất kỳ layout nào cũng sẽ có thuộc tính như sau: **android:onClick="EventName"** , EventName ở đây chính là tên sự kiện, khi người dùng tương tác với view và nó sẽ bắt sự kiện tương ứng với view đó dựa vào ID.
Đây là một view định nghĩa trong layout:

```
<Button
      android:id="@+id/btnLogin"
      android:layout_gravity="center"
      android:onClick="onClick"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Login" />
```

Sau đó ở MainActivity sẽ bắt sự kiện như sau:

```
package com.cheng.handeleventandroid;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

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

    public void onClick(View view) {
                Toast.makeText(MainActivity.this, "Blog ThangCoder.com xin c
        }
    }
```

Ở file layout mình đã khai báo một sự kiện **onClick** có tên là "*onClick*" luôn, và trong class MainActivity mình khai báo một phương thức có tên **public void onClick(View view)**, ở đây các bạn muốn nó bắt sự kiện cho nút button thì bắt buộc phương thức khai báo phải trùng tên với tên onClick trong file layout.

Ở đây mình đã khi báo giống nhau rồi nhé, nhớ tham số truyền vào phương thức là View nhé chứ không có là lỗi ngay.Ngoài ra trong trường hợp có nhiều nút View gán cùng tên sự kiện thì bạn sẽ bắt theo ID như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<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.cheng.handeleventandroid.MainActivity">

    <Button
        android:id="@+id/btnLogin"
        android:layout_gravity="center"
        android:onClick="onClick"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Login" />
    <Button
        android:id="@+id/btnLogout"
        android:layout_gravity="center"
        android:onClick="onClick"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Logout" />
</LinearLayout>
```

Và đây là các bắt sự kiện cho 2 nút button có chung tên sự kiện onClick:

```
package com.cheng.handeleventandroid;


import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.view.View;

import android.widget.Toast;

public class MainActivity extends AppCompatActivity {


    @Override
```

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

    public void onClick(View view) {
        switch (view.getId()) {
            case R.id.btnLogin:
                Toast.makeText(MainActivity.this, "Blog ThangCoder.com xin chào!",
Toast.LENGTH_SHORT).show();
                break;
            case R.id.btnLogout:
                Toast.makeText(MainActivity.this, "Cheng xin chào!",
Toast.LENGTH_SHORT).show();
                break;
        }
    }
}
```

4.2. Anonymous Listener

Đây là một cách bắt sự kiện gọi là kinh điển của những người lập trình viên Android vì nó được sử dụng rất là nhiều do đơn giản và tiện dụng.Đây là cách bắt sự kiện dựa vào ID của view.

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

```
    android:layout_gravity="center"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Login" />
</LinearLayout>
```

Và đây là cách bắt sự kiện trong MainAcitivty:

```
package com.cheng.handeleventandroid;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
/**
 * Created by Welcome on 8/17/2016.
 */
public class Cach2Activity extends AppCompatActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.cach2);
        Button btnLogin = (Button)
findViewById(R.id.btnLogin);
        btnLogin.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Toast.makeText(Cach2Activity.this, "Blog
ThangCoder.com xin chào!", Toast.LENGTH_SHORT).show();
            }
```

```
            });
        }
}
```

Với kiểu xử lý sự kiện này thì bạn sẽ thấy thông qua id của view đúng không nào?Sau khi khởi tạo xong Button thì bạn gọi phương thức setOnClickListener và truyền vào param là Inner Anonymous (lớp nặc danh), lớp này có công dụng bắt sự kiện cho view gọi nó.

4.3. Variable Listerner

Cách gán thứ ba là Variables Event Listerner, cách này ta sẽ dung một biến trung gian để lưu sự kiện của control và cách này ta có thể viết xử lý sự kiện một cách tập trong nhất..

Code:

```
package com.tuandc.project1.project1;


import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;


public class MainActivity extends AppCompatActivity {
    TextView tv_Hello;
    Button btn_press;// Khởi tạo hai biến TextView và Button để chuẩn bị ánh xạ các control
    View.OnClickListener ClickEvent = null;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tv_Hello = (TextView) findViewById(R.id.tv_hello); // sử dụng findViewById để tìm và ánh xạ control tù File X
ml vào biến tv_Hello
        btn_press = (Button) findViewById(R.id.btn_press);
        // Khởi tao gia trị cho biến ClickEvent
```

```
        ClickEvent = new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                switch (v.getId()) {
                    case R.id.btn_press:
                        tv_Hello.setText("Xin chào mọi người");
                        break;
                }
            }
        };

        btn_press.setOnClickListener(ClickEvent);// Gán biến sự kiện cho control
    }
}
```

Như bạn thấy tham số truyền vào phương thức setOnClickListener() là biến ClickEvent mà ta đã tạo ở trên

Và đây là kết quả khi chạy của hai cách gán sự kiện trên.

4.4. Activity Listerner

– Trong cách viết sự kiện này thì Activity sẽ implements interface có kiểu sự kiện (rất nhiều loại interface). Ở đây Tôi chỉ ví dụ trường hợp cho Button các trường hợp khác các bạn tự tìm hiểu và suy luận ra.

– Tôi sẽ có một bài ví dụ nhỏ sau: Hãy xây dựng ứng dụng tính Chỉ số khối cơ thể -chữ viết tắt BMI (Body Mass Index)- được dùng để đánh giá mức độ gầy hay béo của một người. Chỉ số này có thể giúp xác định một người bị bệnh béo phì hay bị bệnh suy dinh dưỡng.

+Cách tính như sau:

Gọi W là khối lượng của một người (tính bằng kg) và H là chiều cao của người đó (tính bằng m), chỉ số khối cơ thể được tính theo công thức:

$$BMI = \frac{W}{(H)^2}$$

Phân loại để đánh giá như sau:

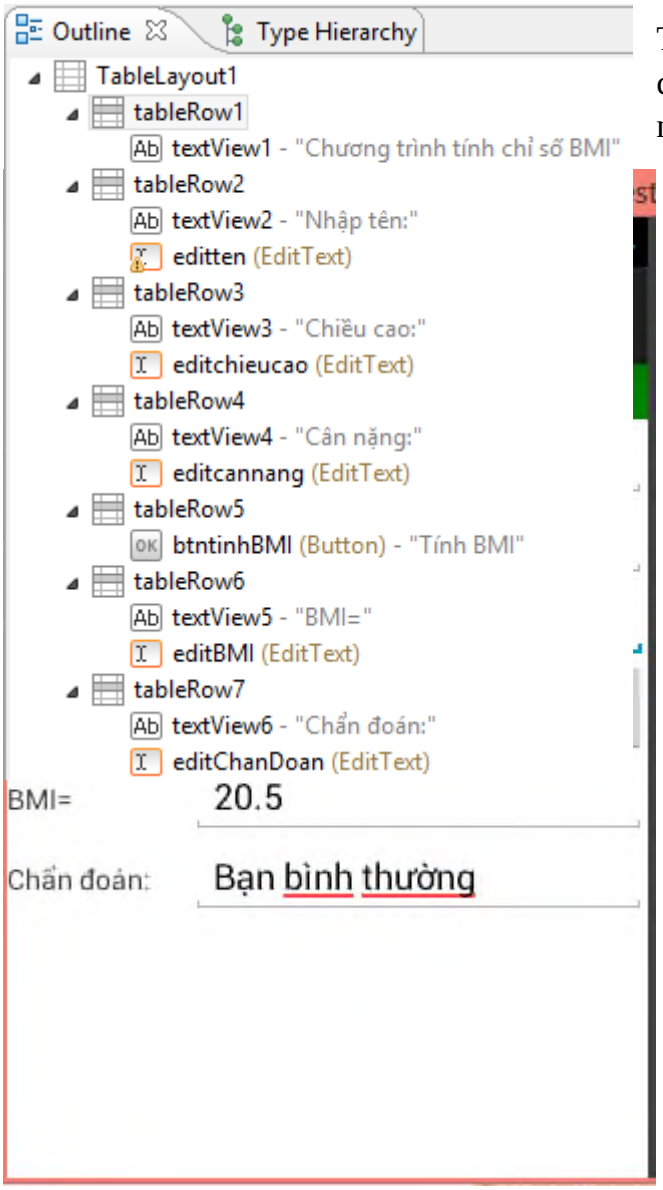
BMI < 18: người gầy

BMI = 18 – 24,9: người bình thường

BMI = 25 – 29,9: người béo phì độ I

BMI = 30 – 34,9: người béo phì độ II

BMI > 35: người béo phì độ III



Tôi sẽ thiết kế giao diện như hình bên dưới và cung cấp Outline, các bạn hãy thiết kế lại để nâng cao kinh nghiệm:

-Bạn thấy đó: Thông số bên trên chính là của chính Tôi, Tên Tôi là Thanh, chiều cao 1.68 mét, cân nặng 58 kg. Khi Tôi chọn "Tính BMI" thì chương trình sẽ tính ra được BMI của Tôi là 20.5 và chẩn đoán là "Bình thường", tức là Tôi có sức khỏe tốt, dáng người chuẩn (có thể là niềm ước ao của một số các bạn trong lớp).

– Để các bạn đỡ căng thẳng thì Tôi xin nói lý do vì sao Tôi lại viết phần mềm này làm ví dụ minh họa. Bởi vì bạn gái của Tôi luôn luôn chê Tôi gầy (hay gọi là Xí Quách gì đó). Tôi đã chứng minh là Tôi có dáng người chuẩn không cần chỉnh bằng cách viết phần mềm này (công thức theo chuẩn Quốc Tế nên nó là một bằng chứng không thể chối cãi). Tôi nghĩ các bạn cũng có thể dùng nó để chứng minh rằng mình không bị Béo Phì Cấp độ 3.

– Quay trở lại ví dụ ,Bạn xem Outline của giao diện này dưới đây (Tôi cố tình không cung cấp source XML nhằm mục đích kích thích các bạn tự tìm tòi):

– Đây là nội dung Coding trong Activity:

```
import java.text.DecimalFormat;
import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
public class MainActivity extends Activity
implements OnClickListener{
```

```
Button btnChandoan;
EditText editTen,editChieucao,
editCannang,editBMI,editChandoan;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
btnChandoan=(Button) findViewById(R.id.btntinhBMI);
btnChandoan.setOnClickListener(this);
editTen=(EditText) findViewById(R.id.editten);
editChieucao=(EditText) findViewById(R.id.editchieucao);
editCannang=(EditText) findViewById(R.id.editcannang);
editBMI=(EditText) findViewById(R.id.editBMI);
editChandoan=(EditText) findViewById(R.id.editChanDoan);
}
@Override
public void onClick(View arg0) {
double H=Double.parseDouble(editChieucao.getText()+"");
double W=Double.parseDouble(editCannang.getText()+"");
double BMI=W/Math.pow(H, 2);
String chandoan="";
if(BMI<18)
{
chandoan="Bạn gầy";
}
else if(BMI<=24.9)
{
chandoan="Bạn bình thường";
}
```

```
else if(BMI<=29.9)
{
chandoan=”Bạn béo phì độ 1″;
}
else if(BMI<=34.9)
{
chandoan=”Bạn béo phì độ 2″;
}
else
{
chandoan=”Bạn béo phì độ 3″;
}
DecimalFormat dcf=new DecimalFormat(“#.0”);
editBMI.setText(dcf.format(BMI));
editChandoan.setText(chandoan);
}
}
```

4.5. Explicit Listerner

– Trường hợp này ta tách riêng một class đóng vai trò là class sự kiện riêng.

– Khi nào lượng coding trong ứng dụng khổng lồ và phức tạp thì ta nên tách class sự kiện riêng để dễ quản lý.

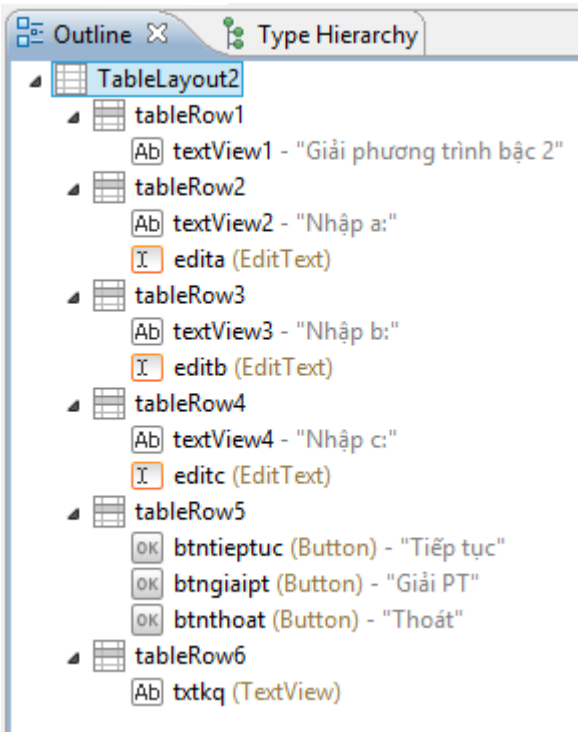
Tôi ví dụ giải phương trình bậc 2, bạn xem giao diện bên dưới:

– Khi chọn “Tiếp tục”, chương trình sẽ xóa trắng toàn bộ dữ liệu trên màn hình đồng thời focus tới ô Nhập a.

– Khi chọn “Giải PT”, chương trình sẽ tiến hành lấy thông số a,b,c và tiến hành giải phương trình bậc 2 và cho ra kết quả như hình trên.

– Khi chọn “Thoát”, chương trình sẽ được đóng lại.

– Bạn xem Outline dưới đây:



– Tiến hành coding, bạn mở Activity class và coding như bên dưới:

```
import java.text.DecimalFormat;

import android.os.Bundle;

import android.app.Activity;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

public class MainActivity extends Activity {

Button btnTieptuc,btnGiai,btnThoat;

EditText edita,editb,editc;

TextView txtkq;

@Override

protected void onCreate(Bundle savedInstanceState) {
```

```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
btnTieptuc=(Button) findViewById(R.id.btntieptuc);
btnGiai=(Button) findViewById(R.id.btngiaipt);
btnThoat=(Button) findViewById(R.id.btnthoat);
btnTieptuc.setOnClickListener(new MyEvent());
btnGiai.setOnClickListener(new MyEvent());
btnThoat.setOnClickListener(new MyEvent());
edita=(EditText) findViewById(R.id.edita);
editb=(EditText) findViewById(R.id.editb);
editc=(EditText) findViewById(R.id.editc);
txtkq=(TextView) findViewById(R.id.txtkq);
}
public void giaiPtb2()
{
String sa=edita.getText()+"";
String sb=editb.getText()+"";
String sc=editc.getText()+"";
int a=Integer.parseInt(sa);
int b=Integer.parseInt(sb);
```

```
int c=Integer.parseInt(sc);
String kq="";
DecimalFormat dcf=new DecimalFormat("#.00");
if(a==0)
{
if(b==0)
{
if(c==0)
kq="PT vô số nghiệm";
else
kq="PT vô nghiệm";
}
else
{
kq="Pt có 1 No, x="+dcf.format(-c/b);
}
}
else
{
double delta=b*b-4*a*c;
```

```
if(delta<0)
{
kq=”PT vô nghiệm”;
}
else if(delta==0)
{
kq=”Pt có No kép x1=x2=”+dcf.format(-b/(2*a));
}
else
{
kq=”Pt có 2 No: x1=”+dcf.format((-b-Math.sqrt(delta))/(2*a))+
“; x2=”+dcf.format((-b-Math.sqrt(delta))/(2*a));
}
}
txtkq.setText(kq);
}
private class MyEvent implements OnClickListener
{
@Override
public void onClick(View arg0) {
```

```
if(arg0==btnTieptuc)
{
edita.setText("");
editb.setText("");
editc.setText("");
edita.requestFocus();
}
else if(arg0.getId()==R.id.btngiaipt)
{
giaiPtb2();
}
else if(arg0.getId()==R.id.btnthoat)
{
finish();
}
}
}
}
```

– Bạn quan sát coding ở bên trên. Tôi tạo một lớp sự kiện tên là **MyEvent**, control nào muốn được gán sự kiện chỉ cần gọi lệnh giống như Tôi gán cho Button Giải phương trình:

btnGiai.setOnClickListener(**new** MyEvent());

– Trong hàm **public void** onClick(View arg0) . Bạn để ý là lúc thì Tôi so sánh theo Object, lúc thì Tôi lại lấy Id ra để so sánh. Đây là cố ý của Tôi, Tôi muốn nói rằng các bạn có thể kiểm tra xem Button nào sẽ được chọn trên giao diện bằng cách so sánh đối tượng hoặc lấy Id ra để so sánh (tùy ý đồ lập trình của mỗi người)

– Ở đây Tôi có 1 lời khuyên cho các bạn là khi phải xử lý quá nhiều dòng lệnh (lệnh phức tạp) thì bạn nên viết thành từng hàm riêng, và trong hàm xử lý sự kiện bạn chỉ cần gọi tên hàm mà thôi. Cụ thể là đối với Button "**Giải PT**" thì Tôi lại viết và gọi riêng hàm giaiPtb2(), còn đối với Button Tiếp tục và Thoát thì Tôi lại không cần thiết viết hàm vì xử lý quá đơn giản.

4.6. SubClassing Listerner

Kỹ thuật này không được phổ biến cho lắm. Bạn chỉ sài khi thêm Control động (lúc runtime) vào màn hình. Ta có thể dùng bất kỳ kỹ thuật nào (6 cách Tôi vừa nêu) để thêm sự kiện động cho một Button động.

-Ở cách cuối cùng này thì bạn phải override phương trình performClick của chính Button control:

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

        Button But9= new Button(this) {
          @Override
          public boolean performClick() {
            //your code here
            return false;
          }
        };
        But9.setText("But9");
        LinearLayout layout = (LinearLayout)findViewById(R.id.lll);
        layout.addView(But9);
  }

}
```

Như vậy các bạn đã được thực hành về các kiểu lập trình sự kiện trong Android, biết được cách lấy dữ liệu từ EditText, biết xử lý định dạng dữ liệu, củng cổ thêm được Layout.

– Trong các bài tập sắp tới các bạn sẽ được thực hành về Toast & Alert Dialog, và rất nhiều các control cơ bản cũng như nâng cao trong Android .

– Bạn phải hiểu rõ các kỹ thuật lập trình này để tùy vào từng trường hợp cụ thể mà bạn nên quyết định kỹ thuật nào cho phù hợp.

CHƯƠNG 5. VIEW NÂNG CAO TRONG ANDROID

5.1 Adapter trong Android

Adapter View ở khắp mọi nơi và bạn sẽ khó mà tìm thấy một ứng dụng Android phổ biến mà không sử dụng chúng. Tên nghe có vẻ lạ, nhưng nếu bạn nghĩ rằng bạn chưa bao giờ nhìn thấy một Adapter View, thì bạn có thể đã sai lầm. Mỗi khi bạn nhìn thấy một ứng dụng Android hiển thị các phần tử của giao diện người dùng dưới dạng một danh sách, một lưới hoặc ngăn xếp, tức là bạn đã nhìn thấy một Adapter View trong thực tế.

Một **Adapter View**, như tên gọi của nó, là một đối tượng `View`. Điều này có nghĩa là, bạn có thể thêm nó vào các Activity của bạn theo cùng một cách bạn thêm bất cứ thành phần giao diện người dùng khác. Tuy nhiên, nó không có khả năng hiển thị bất kỳ dữ liệu nào của riêng mình. Nội dung của nó luôn luôn được xác định bởi một đối tượng khác, một **Adapter**. Trong bài này, tôi sẽ hướng dẫn cho bạn cách làm thế nào để tạo ra các Adapter và sử dụng chúng để tạo ra các loại Adapter View khác nhau chẳng hạn như `ListView` và `GridView`.

Một Adapter là một đối tượng của một lớp cài đặt giao diện Adapter. Nó đóng vai trò như là một liên kết giữa một tập hợp dữ liệu và một Adapter View, một đối tượng của một lớp thừa kế lớp trừu tượng AdapterView. Tập hợp dữ liệu có thể là bất cứ điều gì mà trình bày dữ liệu một cách có cấu trúc. Mảng, các đối tượng List và các đối tượng Cursor thường sử dụng bộ dữ liệu.

Một Adapter có trách nhiệm lấy dữ liệu từ bộ dữ liệu và tạo ra các đối tượng View dựa trên dữ liệu đó. Các đối tượng View được tạo ra sau đó được sử dụng để gắn lên bất kỳ Adapter View mà ràng buộc với Adapter

Bạn có thể tạo các lớp Adapter riêng của bạn từ đầu, nhưng hầu hết các nhà phát triển muốn sử dụng hoặc thừa kế các lớp Adapter được cung cấp bởi Android SDK, chẳng hạn như ArrayAdapter và SimpleCursorAdapter. Trong hướng dẫn này, chúng ta sẽ tập trung vào lớp ArrayAdapter.

5.2 Listview

1. Tạo một ArrayAdapter

Để tạo một Adapter, bạn cần những thứ sau đây:

một tập hợp dữ liệu một tập tin có chứa Layout của các đối tượng View được tạo ra

Ngoài ra, vì lớp ArrayAdapter chỉ có thể làm việc với chuỗi, nên bạn cần phải chắc chắn rằng Layout của các đối tượng View được tạo ra có chứa ít nhất một TextView.

Bước 1: Tạo bộ dữ liệu

Lớp ArrayAdapter có thể sử dụng cả mảng và các đối tượng List như là bộ dữ liệu. Bây giờ, hãy sử dụng một mảng làm tập hợp dữ liệu.

```
1    String[] cheeses = {
2            "Parmesan",
3            "Ricotta",
4            "Fontina",
5            "Mozzarella",
6            "Cheddar"
7           };
```

Bước 2: Tạo tập tin nguồn

Tạo một tập tin Layout XML mới mà phần tử gốc là một LinearLayout và đặt tên nó là item.xml. Kéo và thả một Large text widget vào trong đó và thiết lập giá trị thuộc tính id của nó thành cheese_name. Tập tin Layout XML sẽ trông như thế này:

```
01    <?xml version="1.0" encoding="utf-8"?>
02    <LinearLayout xmlns:android="https://schemas.android.com/apk/res/android"
03       android:orientation="vertical" android:layout_width="match_parent"
04       android:layout_height="match_parent"
05       android:padding="@dimen/activity_horizontal_margin">
06       <TextView
07          android:layout_width="wrap_content"
08          android:layout_height="wrap_content"
09          android:textAppearance="?android:attr/textAppearanceLarge"
10          android:text="Large Text"
11          android:id="@+id/cheese_name" />
12    </LinearLayout>
```

Bước 3: Tạo Adapter

Trong Activity của bạn, tạo ra một đối tượng mới của lớp ArrayAdapter bằng cách sử dụng hàm xây dựng của nó. Đối với các đối số của nó, truyền vào tên của tập tin tài nguyên, định danh của TextView, và một tham chiếu đến mảng. Adapter bây giờ đã sẵn sàng.

```
1   ArrayAdapter<String> cheeseAdapter =
2       new ArrayAdapter<String>(this,
3           R.layout.item,
4           R.id.cheese_name,
5           cheeses
6       );
```

2. Tạo một List

Để hiển thị một danh sách cuộn theo chiều dọc của các phần tử, bạn có thể sử dụng ListView. Để thêm nó vào Activity của bạn, bạn có thể kéo và thả nó vào trong tập tin layout XML của Activity hoặc tạo ra nó bằng cách sử dụng hàm xây dựng của nó trong code Java của bạn. Bây giờ, chúng ta thực hiện cái thứ hai.

```
1   ListView cheeseList = new ListView(this);
```

Thông thường, không có các thành phần giao diện người dùng khác được đặt bên trong một Layout có chứa một ListView. Vì vậy, truyền ListView vào phương thức setContentView() của Activity để nó chiếm toàn bộ màn hình.

```
1   setContentView(cheeseList);
```

Để liên kết ListView với Adapter mà chúng ta đã tạo ra ở bước trước đó, hãy gọi phương thức setAdapter() như hình dưới đây.

```
1   cheeseList.setAdapter(cheeseAdapter);
```

Nếu bạn chạy ứng dụng của bạn ngay bây giờ, bạn có thể xem nội dung của các mảng ở dạng một danh sách.

3. Tạo một Grid

Để hiển thị một danh sách lưới hai chiều cuộn theo chiều dọc của các phần tử, bạn có thể sử dụng GridView. Cả ListView và GridView là lớp con của lớp trừu tượng AbsListView và chúng chia sẻ nhiều điểm tương đồng. Vì vậy, nếu bạn biết cách sử dụng một cái, thì bạn cũng sẽ biết cách sử dụng cái còn lại.

Sử dụng hàm xây dựng của lớp GridView để tạo ra một đối tượng mới và truyền nó vào phương thức setContentView() của Activity.

```
1  GridView cheeseGrid = new GridView(this);
2  setContentView(cheeseGrid);
```

Để thiết lập số cột trong Grid, hãy gọi phương thức setNumColumns() của nó. Tôi sẽ cho nó hai cột.

```
1  cheeseGrid.setNumColumns(2);
```

Thông thường, bạn sẽ cần điều chỉnh độ rộng của các cột và khoảng cách giữa chúng bằng cách sử dụng các phương thức setColumnWidth(), setVerticalSpacing() và setHorizontalSpacing(). Lưu ý rằng những phương thức này sử dụng pixel làm đơn vị của chúng.

```
1  cheeseGrid.setColumnWidth(60);
2  cheeseGrid.setVerticalSpacing(20);
3  cheeseGrid.setHorizontalSpacing(20);
```

Bạn bây giờ có thể buộc GridView vào adapter mà chúng ta đã tạo ra trước đó bằng cách sử dụng phương thức setAdapter().

```
1  cheeseGrid.setAdapter(cheeseAdapter);
```

Chạy lại ứng dụng của bạn một lần nữa để xem GridView trông như thế nào.

4. Thêm Event Listener

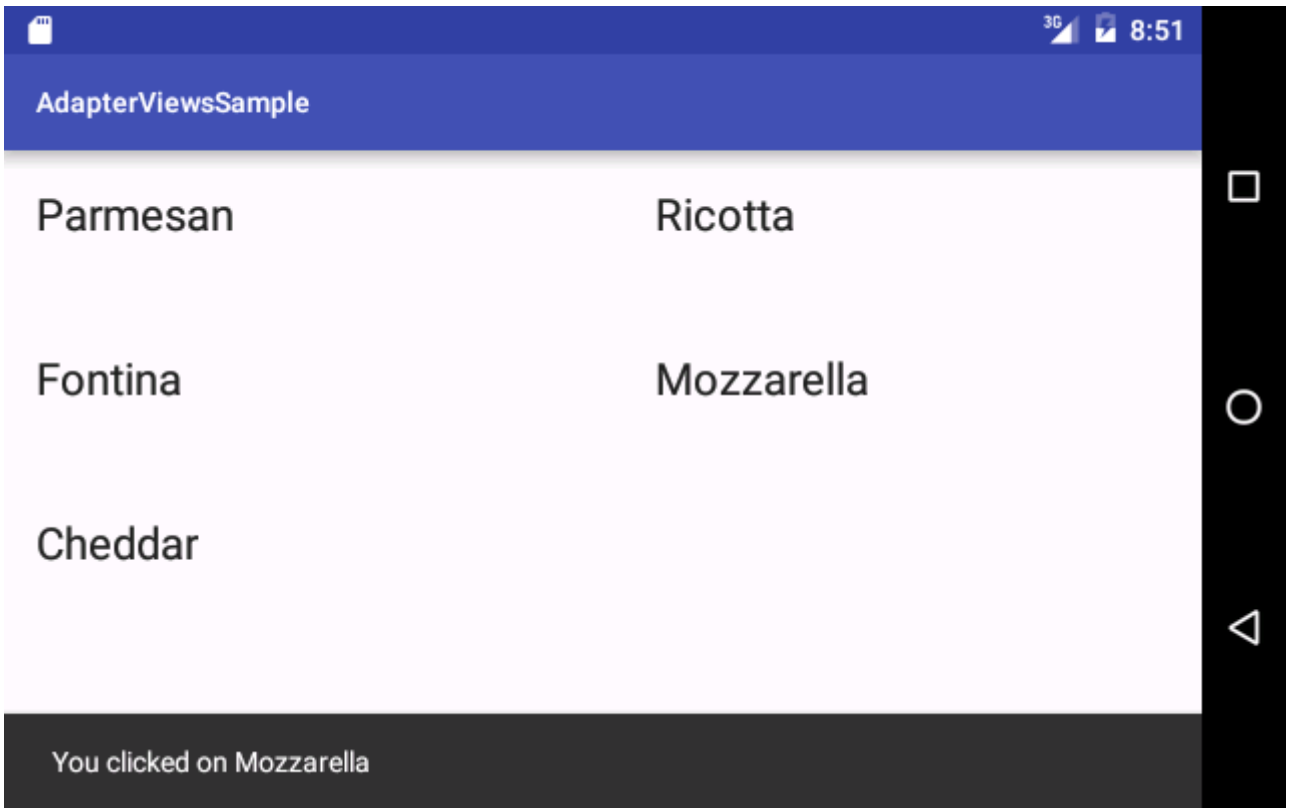
Chúng ta có thể lắng nghe các sự kiện chạm và chạm lâu vào các phần tử bên trong một Adapter View. Ví dụ, hãy thêm một listener sự kiện bấm vào GridView.

Tạo ra một đối tượng mới của một lớp nặc danh cài đặt giao diện AdapterView.OnItemClickListener và truyền nó vào phương thức setOnItemClickListener() của đối tượng GridView. Android Studio tự động tạo ra ngẫu nhiên phương thức onItemClick() của giao diện. Bạn sẽ thấy rằng các tham số của phương thức bao gồm một số nguyên xác định vị trí của phần tử. Bạn có thể sử dụng số nguyên này để tìm thấy phần tử trong bộ dữ liệu mà người dùng đã nhấp vào.

Các code sau đây minh họa cách làm thế nào để hiển thị một thông báo đơn giản mỗi khi một phần tử trong GridView được nhấp.

```
01  cheeseGrid.setOnItemClickListener(new AdapterView.OnItemClickListener() {
02      @Override
03      public void onItemClick(AdapterView<?> adapterView,
04                              View view, int position, long rowId) {
05
06          // Generate a message based on the position
07          String message = "You clicked on " + cheeses[position];
08
09          // Use the message to create a Snackbar
10          Snackbar.make(adapterView, message, Snackbar.LENGTH_LONG)
11                  .show(); // Show the Snackbar
12      }
13  });
```

Nếu bạn chạy ứng dụng và nhấp vào bất kỳ phần tử nào trong Grid, một thông báo sẽ xuất hiện ở dưới cùng của màn hình. Lưu ý rằng bạn có thể sử dụng code tương tự để lắng nghe các sự kiện nhấp lên các phần tử bên trong một ListView.



5. Thừa kế ArrayAdapter

Một ArrayAdapter chỉ có thể xử lý một TextView bên trong layout của các đối tượng View mà nó tạo ra. Để mở rộng khả năng của nó, bạn phải thừa kế nó. Tuy nhiên, trước khi chúng ta làm điều đó, chúng ta hãy tạo ra một tập hợp dữ liệu phức tạp hơn một chút.

Thay vì chuỗi, giả sử rằng tập hợp dữ liệu của chúng ta có chứa các đối tượng của lớp sau đây:

```
1   static class Cheese {
2       String name;
3       String description;
4
5       public Cheese(String name, String description) {
6           this.name = name;
7           this.description = description;
8       }
9   }
```

Đây là tập hợp dữ liệu mà chúng ta sẽ sử dụng:

```
1   Cheese[] cheeses = {
```

```
2           new Cheese("Parmesan", "Hard, granular cheese"),
3           new Cheese("Ricotta", "Italian whey cheese"),
4           new Cheese("Fontina", "Italian cow's milk cheese"),
5           new Cheese("Mozzarella", "Southern Italian buffalo milk cheese"),
6           new Cheese("Cheddar", "Firm, cow's milk cheese"),
7   };
```

Như bạn thấy, lớp Cheese có chứa hai trường, name và description. Để hiển thị cả hai trường trong một List hoặc Grid, thì layout của các phần tử phải chứa hai TextView.

Tạo một tập tin layout XML mới và đặt tên là custom_item.xml. Thêm Large text và một Small text vào nó. Thiết lập thuộc tính id của cái đầu tiên thành cheese_name và của cái thứ hai là cheese_description. Nội dung của tập tin layout XML bây giờ sẽ giống như thế này:

```
01  <?xml version="1.0" encoding="utf-8"?>
02  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
03      android:orientation="vertical" android:layout_width="match_parent"
04      android:layout_height="match_parent"
05      android:padding="@dimen/activity_horizontal_margin">
06      <TextView
07          android:layout_width="wrap_content"
08          android:layout_height="wrap_content"
09          android:textAppearance="?android:attr/textAppearanceLarge"
10          android:text="Large Text"
11          android:id="@+id/cheese_name" />
12      <TextView
13          android:layout_width="wrap_content"
14          android:layout_height="wrap_content"
15          android:textAppearance="?android:attr/textAppearanceSmall"
16          android:text="Small Text"
17          android:id="@+id/cheese_description" />
18  </LinearLayout>
```

ArrayAdapter cũng phải có khả năng xử lý hai TextView. Xem lại Activity của bạn, tạo ra một lớp nặc danh mới thừa kế lớp ArrayAdapter và override phương thức getView() của nó. Hãy chắc chắn rằng bạn truyền mảng như là đối số vào hàm xây dựng của nó.

```
1   ArrayAdapter<Cheese> cheeseAdapter =
2       new ArrayAdapter<Cheese>(this, 0, cheeses) {
3           @Override
4           public View getView(int position,
5                               View convertView,
6                               ViewGroup parent) {
7
8           }
9       };
```

Bên trong phương thức getView(), bạn phải sử dụng tham số position như là một chỉ số của mảng và lấy phần tử ở chỉ số đó.

```
1   Cheese currentCheese = cheeses[position];
```

Tham số thứ hai của phương thức getView() là những gì cho phép chúng ta tái sử dụng các đối tượng View. Nếu bạn bỏ qua nó, hiệu suất của các adapter view của bạn sẽ thấp. Khi phương thức getView() được gọi lần đầu tiên, thì convertView là null. Bạn phải khởi tạo nó bằng cách inflate tập tin tài nguyên xác định layout của các phần tử. Để làm như vậy, hãy lấy một tham chiếu đến một LayoutInflater bằng cách sử dụng phương thức getLayoutInflater() và gọi phương thức inflate() của nó.

```
1   // Inflate only once
2   if(convertView == null) {
3       convertView = getLayoutInflater()
4               .inflate(R.layout.custom_item, null, false);
5   }
```

Tại thời điểm này, bạn có thể sử dụng findViewById() để có được một tham chiếu đến các TextView bên trong layout và gọi phương thức setText() để khởi tạo chúng bằng cách sử dụng dữ liệu từ mảng.

```
1   TextView cheeseName =
2       (TextView)convertView.findViewById(R.id.cheese_name);
3   TextView cheeseDescription =
```

```
4       (TextView)convertView.findViewById(R.id.cheese_description);
5
6   cheeseName.setText(currentCheese.name);
7   cheeseDescription.setText(currentCheese.description);
```

Cuối cùng, trả về convertView vì vậy mà nó có thể được sử dụng để đưa bất kỳ adapter view kết hợp với adapter.

```
1   return convertView;
```

8. Sử dụng một View Holder

Phương thức getView() được gọi liên tục bởi Adapter View để phân phối chính nó. Vì vậy, bạn phải cố gắng giảm thiểu số lượng các hoạt động mà bạn thực hiện bên trong nó.

Ở bước trước, bạn có thể thấy rằng, mặc dù chúng ta đã chắc chắn rằng layout của các phần tử trong danh sách inflate chỉ một lần, nhưng phương thức findViewById(), chiếm nhiều CPU, được gọi mỗi khi phương thức getView() được gọi.

Để tránh điều này và cải thiện hiệu suất của Adapter View, chúng ta cần lưu trữ các kết quả của phương thức findViewById() bên trong đối tượng convertView. Để làm như vậy, chúng ta có thể sử dụng một đối tượng View Holder, nó không có gì khác hơn là một đối tượng của một lớp có thể lưu trữ các thành phần hiện diện trong layout.

Bởi vì layout có hai TextView, nên View Holder cũng phải có hai TextView. Tôi đã đặt tên lớp là ViewHolder.

```
1   static class ViewHolder{
2       TextView cheeseName;
3       TextView cheeseDescription;
4   }
```

Trong phương thức getView(), sau khi bạn inflate layout, bây giờ bạn có thể khởi tạo đối tượng View Holder bằng cách sử dụng phương thức findViewById().

```
1   ViewHolder viewHolder = new ViewHolder();
2   viewHolder.cheeseName =
3           (TextView)convertView.findViewById(R.id.cheese_name);
4   viewHolder.cheeseDescription =
5           (TextView)convertView.findViewById(R.id.cheese_description);
```

Để lưu trữ đối tượng View Holder trong convertView, sử dụng phương thức setTag() của nó.

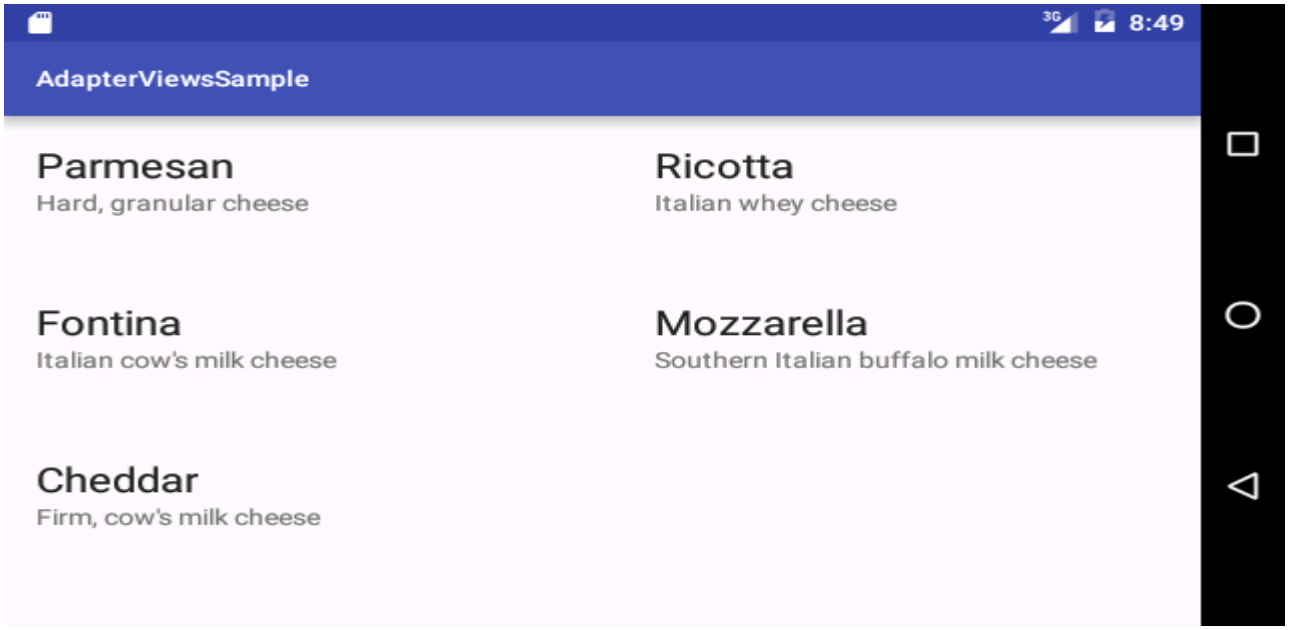
```
1   // Store results of findViewById
```

```
2     convertView.setTag(viewHolder);
```

Và bây giờ, mỗi khi getView() được gọi, bạn có thể lấy đối tượng View Holder từ convertView bằng cách sử dụng phương thức getTag() và cập nhật các TextView bên trong nó bằng cách sử dụng phương thức setText() của chúng.

```
1     TextView cheeseName =
2         ((ViewHolder)convertView.getTag()).cheeseName;
3     TextView cheeseDescription =
4         ((ViewHolder)convertView.getTag()).cheeseDescription;
5
6     cheeseName.setText(currentCheese.name);
7     cheeseDescription.setText(currentCheese.description);
```

Nếu bạn chạy ứng dụng của bạn ngay bây giờ, bạn có thể thấy GridView hiển thị hai dòng văn bản trong mỗi ô.



5.3 Custom Listview

Bây giờ chúng ra sẽ đi custom lại cái listview mà mình đã giới thiệu ở bài 16 nhé, cái này cũng là ở mức dễ chứ không có gì khó, sau bài này mình sẽ có bài thực hành khó hơn tí nữa để mọi người làm quen làm việc với listview vì khi đi làm đụng đến nó rất là nhiều.

Chúng ta nhìn lại hình ảnh bên dưới trước khi bắt tay vào làm nhé, mình sẽ phân tích cụ thể các bước chúng ta cần làm là gì:

Bạn thấy ở trên có rất nhiều dòng đúng không nhưng bạn chỉ cần thiết kế layout một dòng và sau đó add vào ListView, bạn add bao nhiêu cái thì nó có bấy nhiêu dòng thôi.

Custom listview android
Công việc chúng ta bao gồm:

1. Tạo một file layout cho các dòng (item) trong listView.
2. Tạo 1 class tượng trưng cho Contact
3. Tạo class CustomAdapter để custom listview cho layout đã tạo phía trên.
4. Gọi CustomAdapter để lấy dữ liệu.

1. File layout ở trên sẽ bao gồm 2 TextView và dưới đây là file layout đó:
row_listview.xml

```
<?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">
<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">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/tvAvatar"
            android:layout_width="70dp"
            android:layout_height="70dp"
            android:background="#18ade4"
            android:gravity="center"
            android:text="1"
            android:textColor="#ffffff"
            android:textSize="28sp"
            android:textStyle="bold" />
```

```
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginLeft="50dp"
        android:orientation="vertical">

        <TextView
            android:id="@+id/tvName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="Truong Dinh Chien"
            android:textSize="16sp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/tvPhoneNumber"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="0988933xxx"
            android:textSize="16sp"
            android:textStyle="bold" />
    </LinearLayout>
</LinearLayout>
```

```
    </LinearLayout>
</LinearLayout>
```

2. Đây là class Contact bạn sẽ thiết kế để tạo ra một đối tượng lưu vào các item của listview, ở đây là các row.Cách tạo **Constructer** và **getter(), setter()** chắc mọi người đã biết rồi chứ đúng không nào, nó học ở Java căn bản đó.
Contact này sẽ có các thuộc tính như:

- **Name**: hiển thị tên ngườ dùng.
- **PhoneNumber**: hiển thị số điện thoại.
- **Color**: màu sắc avatar

Contact.java

```
package com.cheng.bai18customlistview.model;


/**
 * Created by Welcome on 8/27/2016.
 */
public class Contact {
    private String name;
    private String phoneNumber;
    private int color;


    public Contact(String name, String phoneNumber, int color) {
        this.name = name;
        this.phoneNumber = phoneNumber;
        this.color = color;
    }


    public String getName() {
        return name;
```

```
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

    public int getColor() {
        return color;
    }

    public void setColor(int color) {
        this.color = color;
    }
}
```

3.Như bài học về listview trước chúng ta không custom ArrdayAdapter mà là sử dụng luôn thư viện có sẵn mà Android đã có, thế nên listview chỉ hiển thị dữ liệu dạng 1 đoạn text ở trên 1 row, còn bây giờ trên một row chúng ta có tới tận 3 image view thì không dùng nó được mà chúng ta phải custom lại để dữ liệu có thể hiển thị được.

CustomAdapter.java

```
package com.cheng.bai18customlistview.adapter;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import com.cheng.bai18customlistview.R;
import com.cheng.bai18customlistview.model.Contact;

import java.util.ArrayList;
import java.util.List;

/**
 * Created by Welcome on 8/27/2016.
 */
public class CustomAdapter extends ArrayAdapter<Contact> {

    private Context context;
    private int resource;
    private List<Contact> arrContact;

    public CustomAdaper(Context context, int resource, ArrayList<Contact> arrContact) {
        super(context, resource, arrContact);
```

```
        this.context = context;
        this.resource = resource;
        this.arrContact = arrContact;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ViewHolder viewHolder;
        if (convertView == null) {
            convertView = LayoutInflater.from(context).inflate(R.layout.row_listview, parent,
false);
            viewHolder = new ViewHolder();
            viewHolder.tvName = (TextView) convertView.findViewById(R.id.tvName);
            viewHolder.tvNumberPhone = (TextView)
convertView.findViewById(R.id.tvPhoneNumber);
            viewHolder.tvAvatar = (TextView) convertView.findViewById(R.id.tvAvatar);

            convertView.setTag(viewHolder);
        } else {
            viewHolder = (ViewHolder) convertView.getTag();
        }
        Contact contact = arrContact.get(position);
        viewHolder.tvAvatar.setBackgroundColor(contact.getColor());
        viewHolder.tvAvatar.setText(String.valueOf(position+1));
        viewHolder.tvName.setText(contact.getName());
        viewHolder.tvNumberPhone.setText(contact.getPhoneNumber());
        return convertView;
```

```
    }


    public class ViewHolder {
        TextView tvName, tvNumberPhone, tvAvatar;


    }
}
```

Ở thằng này sẽ có rất nhiều cái mới nên mình sẽ giải thích cặn kẽ, không hiểu bạn có thể xem thêm ở video bên dưới nhé:

Ở đây class CustomAdaper kế thừa lại thằng ArrayAdapter<> có kiểu Generic là Contact, nghĩa là nó chỉ nhận vào kiểu dữ liệu là Contact thôi.Khi bạn kế thừa thì bắt buộc bạn phải ghi đè **@Override** constructer là:

```
    public CustomAdaper(Context context, int resource, ArrayList<Contact> arrContact) {
        super(context, resource, arrContact);
        this.context = context;
        this.resource = resource;
        this.arrContact = arrContact;
    }
```

Ở đây bạn sẽ truyền vào các params như trên, nếu bạn có xem bài 18 thì sẽ còn nhớ dòng lênh này:

```
 ArrayAdapter arrayAdapter = new ArrayAdapter(this,android.R.layout.simple_list_item_1,number);
```

Thì bạn custom lại Adapter thì khi dùng bạn cũng sẽ gọi Adapter đó và truyền các tham số vào, ở đây bạn sẽ thay đổi thằng:

android.R.layout.simple_list_item thành layout bạn đã thiết kế ở trên **row_listview.xml,** và list number trên sẽ thay bằng 1 list Contact khác.

Tiếp đến là phương thức getView(), đây là phương thức để khởi tạo view để add nó vào Listview, ở đây nó sẽ tạo ra các row trên listview và set dữ liệu là các Contact lên các row đó, thực ra bạn cũng chẳng cần hiểu nó là cái gì bởi vì đây là cú pháp cứng chúng ta chỉ cần học thuộc rồi dùng, dùng nhiều lần dẫn đến quen và hiểu.

```
@Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ViewHolder viewHolder;
        if (convertView == null) {
            convertView = LayoutInflater.from(context).inflate(R.layout.row_listview, parent, false);
            viewHolder = new ViewHolder();
            viewHolder.tvName = (TextView) convertView.findViewById(R.id.tvName);
            viewHolder.tvNumberPhone = (TextView) convertView.findViewById(R.id.tvPhoneNumber);
            viewHolder.tvAvatar = (TextView) convertView.findViewById(R.id.tvAvatar);


            convertView.setTag(viewHolder);
        } else {
            viewHolder = (ViewHolder) convertView.getTag();
        }
        Contact contact = arrContact.get(position);
        viewHolder.tvAvatar.setBackgroundColor(contact.getColor());
        viewHolder.tvAvatar.setText(String.valueOf(position+1));
        viewHolder.tvName.setText(contact.getName());
        viewHolder.tvNumberPhone.setText(contact.getPhoneNumber());
        return convertView;
    }
```

Ở đây mình sử dụng thằng ViewHolder để **setTag()** và **getTag()** khi khởi tạo view, các bạn biết tại sao lại sử dụng nó không? Thực ra listview nếu bạn chỉ có vài row thì khi vuốt lên xuống rất là mượt mà nhưng nếu bạn có đến 1000 row thì lại khác, nó sẽ rất lag đó(sau này mình sẽ giới thiệu Recyclerview dùng mượt hơn listview).
Chính vì thế thằng ViewHolder này ra đời để khắc phục, khi các bạn trượt listview lên xuống thì nếu các row này đã khởi tạo rồi thì nó sẽ lưu ở Tag và sẽ không khởi tạo lại bàng việc getTag ra, trong video mình giải thích rất rõ cái này nên bạn xem sẽ hiểu ngay.

```
Contact contact = arrContact.get(position);
```

Vì khi gọi class CustomAdaper ở MainActivity bạn sẽ truyền cho nó 1 list Contact nên dòng code trên sẽ có tác dụng là lấy ra từng Contact một tương tứng với từng view, khi lấy được contact nó sẽ có được dữ liệu của 1 Contact như: name, phonenumber, color… và dữ liệu này sẽ set lên trên row của listview.

Và cuối cùng là return lại **converView** chính là các row mà nãy giờ chúng ta đã tạo và set dữ liệu lên.
4.Như bài trước mình đã nói là Adapter có nhiệm vụ là trung gian để chứa data hiển thị lên listview, ở đây bạn cũng làm tương tự là gọi CustomAdapter rồi truyền mảng Contact vào cho nó xử lý đưa vào row thôi.
MainActivity.java

```
package com.cheng.bai18customlistview;


import android.graphics.Color;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.widget.ListView;


import com.cheng.bai18customlistview.adapter.CustomAdaper;

import com.cheng.bai18customlistview.model.Contact;


import java.util.ArrayList;


public class MainActivity extends AppCompatActivity {
```

```
    private ListView lvContact;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        lvContact = (ListView) findViewById(R.id.lv_contact);
        ArrayList<Contact> arrContact = new ArrayList<>();

        Contact contact1 = new Contact("Trương Đình Chiến","0988 933 xxx", Color.RED);
        Contact contact2 = new Contact("Võ Văn Tá","01667 585 545", Color.GREEN);
        Contact contact3 = new Contact("Lê Tấn Dũng","0918 033 033", Color.GRAY);
        Contact contact4 = new Contact("Trương Quang Lâm","0978 102 102",
Color.YELLOW);
        Contact contact5 = new Contact("Võ Duy Tính","01667 333 000", Color.BLACK);
        Contact contact6 = new Contact("Trần Văn Toàn","08 999 321", Color.BLUE);
        Contact contact7 = new Contact("Lại Thế Quang","01222 331 331", Color.CYAN);

        arrContact.add(contact1);
        arrContact.add(contact2);
        arrContact.add(contact3);
        arrContact.add(contact4);
        arrContact.add(contact5);
        arrContact.add(contact6);
        arrContact.add(contact7);

        CustomAdapter customAdaper = new
CustomAdapter(this,R.layout.row_listview,arrContact);
```

```
        lvContact.setAdapter(customAdaper);


    }
}
```

Còn đây là file layout của MainActivity:

activity_main.xml

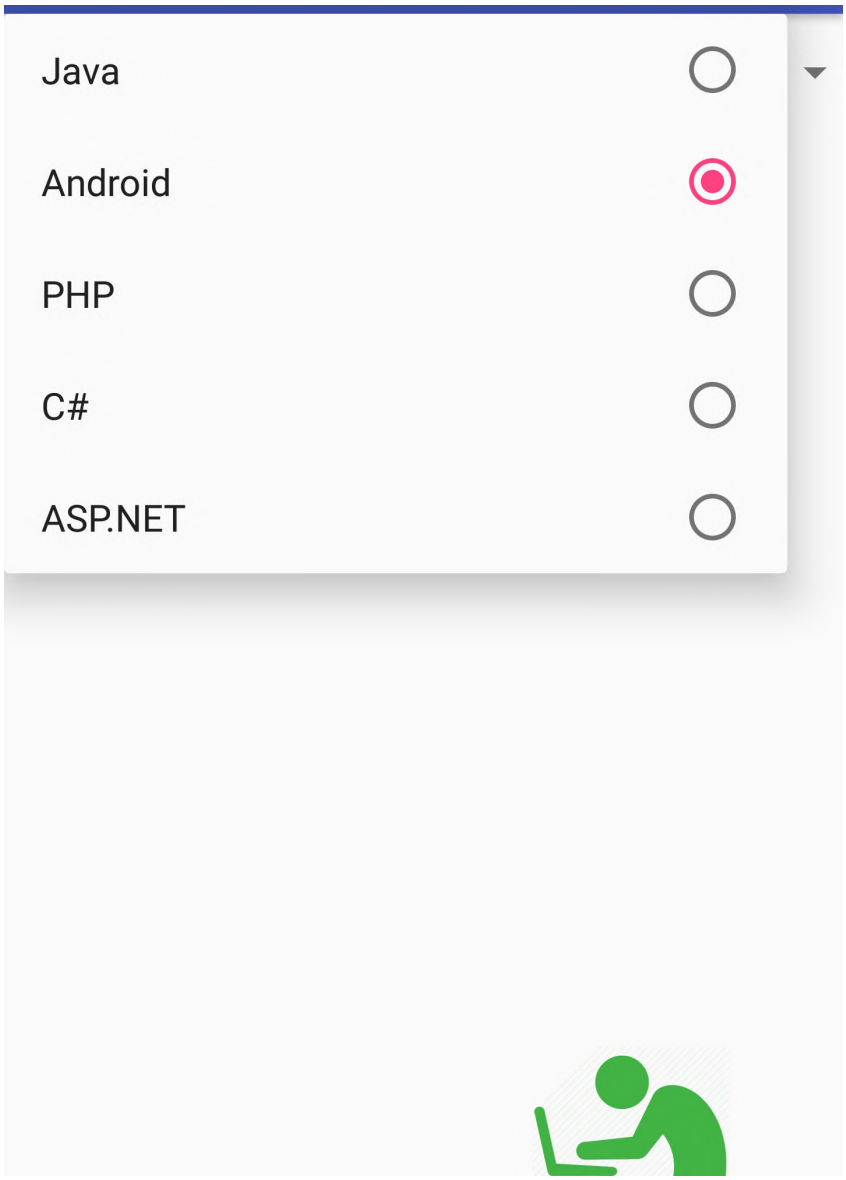
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">


    <ListView
        android:id="@+id/lv_Contact"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</RelativeLayout>
```

Ở MainActivity mình tạo ra 7 đối tượng contact rồi add nó vào Arraylist<Contact> là **arrContact** để bỏ vào CustomAdapter , sau đó setAdapter đó lên listView để nó hiển thị là xong.Như ban đầu mình nói custom Listview khó nhưng tuy nhiên là đối với những người mới học thôi, vì nó hơi khó hiểu nhưng biết rồi thì thấy nó đơn giản thật.
Ngoài cách custom Adapter bằng việc kế thừa ArrayAdapter thì bạn còn cách kế thừa **BaseAdapter** và mình sẽ hướng dẫn trong chuyên mục android nâng cao các bạn có thể tìm xem.Và dưới đây là video chi tiết để cho bạn nào không hiểu có thể xem qua video nhé.

5.4 Spinner

Giới thiệu Spinner

Spinner là view sử dụng để hiển thị lên một list các danh sách giống như listview mà chúng ta sẽ học ở phần sau, tuy nhiên bạn thường chỉ chọn 1 lựa chọn trong 1 danh sách đó thôi.



Spinner trong lập trình Android

Vậy nó áp dụng trong những trường hợp nào đây? Nhìn vào hình là bạn có thể thấy ngay liền đúng không nào? Những trường hợp cần sổ ra một đống danh sách cho người ta chọn một cái như: lựa chọn ngôn ngữ, nơi sinh, thành phố… thì đa số điều sử dụng Spinner.

Các bạn lưu ý là Spinner nếu custom lại thì có thể chọn được nhiều lựa chọn (multiselect) chứ không phải chỉ chọn được 1 cái thôi đâu nhé.

Cách dùng Spinner

Bây giờ chúng ta sẽ đi code layout và sử lí sự kiện giống như hình ở phía trên, khi người dùng chọn vào một item nào đó thì ta Toast thông báo lên cái tên của Item đó thôi, rất là đơn giản đúng không nào?

Layout bạn sẽ làm như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout 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.cheng.bai14spinner.MainActivity">

    <Spinner
        android:id="@+id/spnCategory"
        android:layout_width="match_parent"
        android:layout_height="50dp" />

</FrameLayout>
```

Ở đoạn code xml trên chẳng có gì để chúng ta không hiểu cả, toàn những cái cơ bản mà ta đã học ở những bài trước khi thiết kế layout rồi, bây giờ là đoạn code xử lí bằng Java ở MainActivity nhé:

```
package com.cheng.bai14spinner;
```

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;

import java.lang.reflect.Array;
import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {

    private Spinner spnCategory;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        spnCategory = (Spinner) findViewById(R.id.spnCategory);
```

```
List<String> list = new ArrayList<>();
list.add("Java");
list.add("Android");
list.add("PHP");
list.add("C#");
list.add("ASP.NET");

ArrayAdapter<String> adapter = new ArrayAdapter(this, android.R.layout.simple_spinner_item,list);

adapter.setDropDownViewResource(android.R.layout.simple_list_item_single_choice);
spnCategory.setAdapter(adapter);
spnCategory.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
        Toast.makeText(MainActivity.this, spnCategory.getSelectedItem().toString(), Toast.LENGTH_SHORT).show();
    }

    @Override
```

```
        public void onNothingSelected(AdapterView<?> adapterView) {

        }

    });

  }

}
```

Bạn học qua Java thì biết về ArrayList rồi đúng không nhỉ, ở trên mình tạo một ArrayList để lưu lại các item sẽ show lên khi click vào Spinner.Các bạn nhớ là ở đây là một mảng String nhé.

```
List<String> list = new ArrayList<>();

    list.add("Java");

    list.add("Android");

    list.add("PHP");

    list.add("C#");

    list.add("ASP.NET");
```

Ta có được đối tượng **" list "** , nó chứa các phần tử như: Java, Android, PHP, C#, ASP.NET. Tiếp theo bạn sẽ thấy dòng này:

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

Đây là kiến thức mới và ở bài 17 trở đi mình sẽ nói nhiều về thằng này nên bạn cứ iên tâm, sau này cứ làm gì mà sử dụng đến danh sách bạn đều phải sử dụng thằng **ArrayAdapter** này cả.

Ở đây **ArrayAdapter** chính là cầu nối trung gian giữa View và dữ liệu, như trên dữ liệu ở đây chính là thằng **list** , dữ liệu không thể đưa trực tiếp lên View mà phải thông qua thằng này mới được, bản chất nó như vậy thôi nên mọi người đừng tìm hiểu gì nhiều cho mệt, kệ đi cứ làm rồi sẽ hiểu.
Ở đây có 3 tham số truyền vào là :

- **this**: chính là lấy ra Activity hiện tại
- **android.R.layout.simple_spinner_item** : đây là một mẫu hiển thị danh sách mà android đã đinh nghĩa sẵn, ở trên bạn có thể thấy các item sắp xếp theo chiều dọc và từng hàng, có màu chữ là màu đen.Nếu bạn muốn màu chữ là màu xanh chẳng hạn thì bạn có thể custom lại và thay thế thằng *android.R.layout.simple_spinner_item* bằng một file layout của bạn.
- **list**: là danh sách một **Objects** nào đó, String cũng là một Object nhé các bạn.

Tiếp theo:

```
adapter.setDropDownViewResource(android.R.layout.simple_list_item_single_choice);
```

Dòng trên có nghĩa là định dạng cho thằng Spinner này kiểu single choise, khi add phương thức trên vào bạn sẽ thấy nút tròn màu hồng khi bạn chọn một item nào đó, nhìn lại trên ảnh trên sẽ thấy, nếu không có dòng này bạn sẽ không thấy biểu tượng đó đâu.

Tiếp theo:

```
spnCategory.setAdapter(adapter);
```

Dòng này có nghĩa là set Adapter trên vào thằng spinner, nãy giờ bạn đã làm giao diện, đưa dữ liệu vào Adapter rồi thì bây giờ chỉ việc đưa nó lên thằng Spinner thôi, đó là ý nghĩa dòng lệnh trên.

Bắt sự kiện khi bạn chọn vào một item trong Spinner sẽ như sau:

```
spnCategory.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
```

```
        @Override
        public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l)
{
            Toast.makeText(MainActivity.this,       spnCategory.getSelectedItem().toString(),
Toast.LENGTH_SHORT).show();
        }
        @Override
        public void onNothingSelected(AdapterView<?> adapterView) {
        }
    });
```

Ở bài hướng dẫn bắt sự kiện Onlick mình đã hướng dẫn khá kĩ rồi đúng không nào, ở đây bạn sẽ không bắt sự kiện onClick vào Spinner mà bạn sẽ bắt sự kiện OnClick vào Item của Spinner luôn vì thông thường chúng ta cần lấy ra giá trị của item người dùng select mà.
Dù là bắt sự kiện nào đi nữa thì nó cũng như nhau cả thôi nhé các bạn.

Ở đây khi **setOnItemSelectedListener** cho Spinner sẽ bắt bạn Override 2 phương thức là:

- onItemSelected: gọi hàm này khi có một sự kiện chọn item nào đó.
- onNothingSelected: gọi hàm này khi click vào Spinner mà không chọn item nào cả

Dựa vào yêu cầu bài toàn mà bạn xử lý thôi chứ cái này không có gì khó cả, ở đây mình show lên một Toast lấy ra giá trị của item đã được chọn.

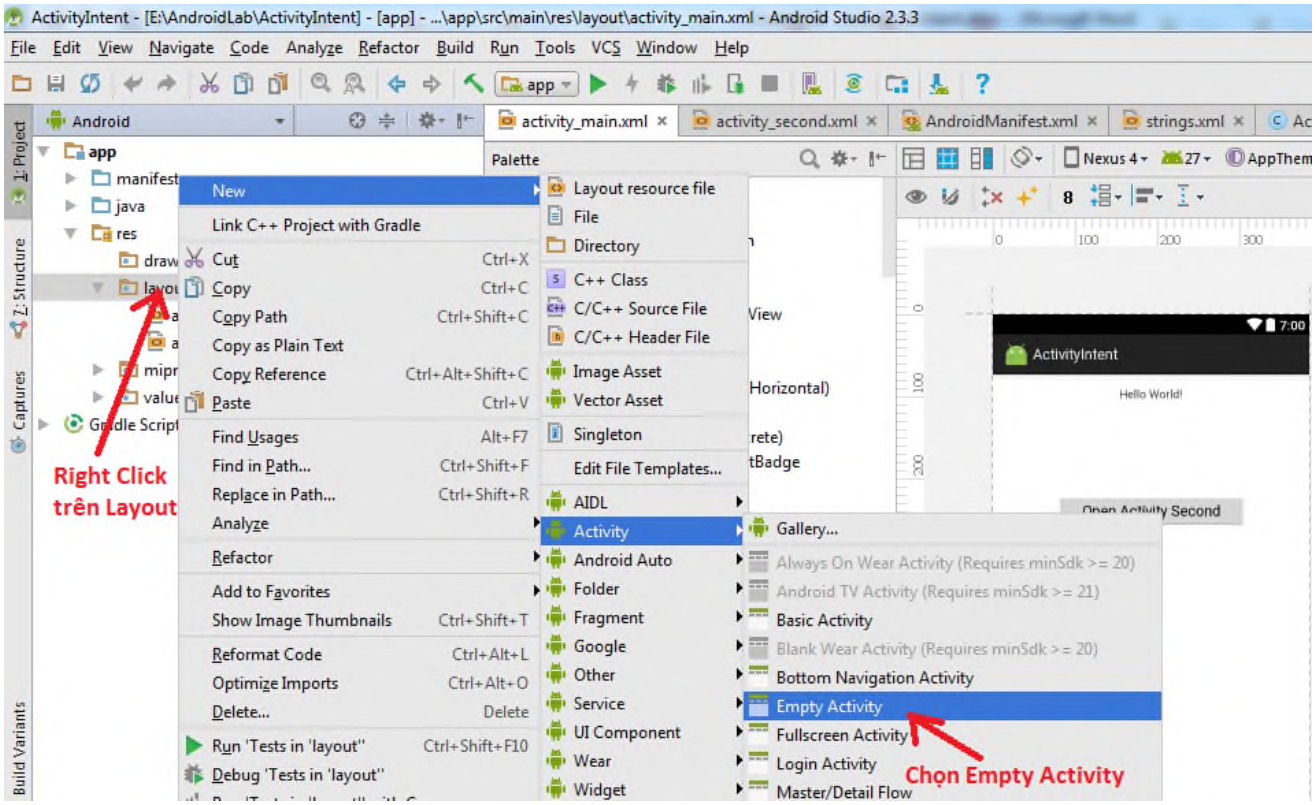
CHƯƠNG 6. ACTIVITY VÀ INTENT

Activity và Intent là 2 thành phần quan trọng trong phát triển ứng dụng trên Android. Activity là màn hình ứng dụng có chức năng tương tác với người dùng. Intent là một đối tượng trừu tượng trong Android, Intent có chức năng lưu trữ dữ liệu để truyền, nhận dữ liệu giữa các Activity hoặc giữa Activity và thiết bị, hay ứng dụng khác trên Android như SMS, gọi điện thoại, trình duyệt web, Email, Camera, ...

1. Activity

Activity có thể là một module, thành phần chức năng độc lập của ứng dụng Android. Activity là màn hình ứng dụng có chức năng tương tác với người dùng do đó lớp Activity đảm nhận việc tạo ra một cửa sổ (window) để người lập trình đặt lên đó một giao diện UI với phương thức setContentView(View). Cửa sổ này thường lấp đầy màn hình, nhưng có thể nhỏ hơn màn hình và nổi bên trên các cửa sổ khác.

Tạo mới một Activity: Trước tiên, click vào app -> res -> layout -> Click chuột phải vào layout. Sau đó chọn New > Activity và chọn Activity mà bạn muốn. Ở đây, chúng ta chọn Empty Activity như hình bên dưới.



Màn hình tạo mới một Activity

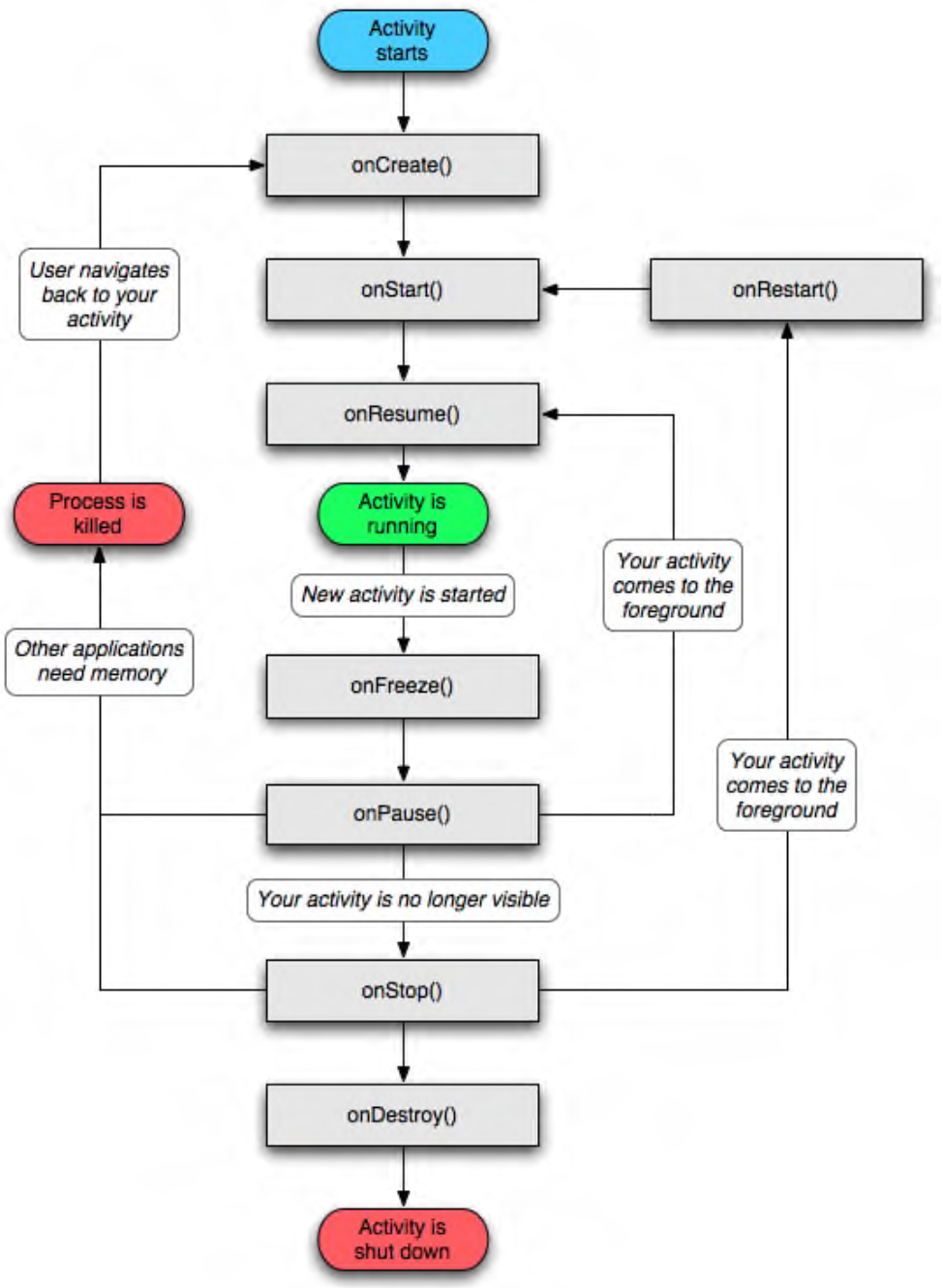
Mỗi Activity tồn tại một file XML lưu trữ giao diện và một file java lưu trữ code xử lý các nghiệp vụ của ứng dụng. Mặc định file giao diện XML có tên activity_main.xml và file java có tên MainActivity.java.

Activity được xây dựng bằng cách kế thừa lớp Activity, một Activity không thể gọi trực tiếp các phương thức hay truy cập vào dữ liệu của một Activity khác. Do đó, để truyền và nhận giữ liệu chúng ta phải dùng tới thành phần gọi là Intent từ Activity này kích hoạt Activity khác.

Chúng ta có thể gọi một activity khác theo một trong hai cách: startActivity() hoặc startActivityForResult(). Phương thức startActivityForResult() sẽ trả về kết quả tại hàm onActivityResult().

Mỗi ứng dụng (Application) thường có một hoặc nhiều Activity và mỗi Activity có vòng đời hoàn toàn độc lập với các Activity khác. Vòng đời của một Activity trong Android gồm có 7 phương thức, mỗi phương thức thể hiện hành vi khác nhau của Activity.

Phương thức	**Ý nghĩa**
onCreate()	Phương thức onCreate() là phương thức đầu tiên được gọi dùng để tạo một activity vào lần đầu tiên activity được gọi.
onStart()	Phương thức onStart() được gọi khi nó hiện hữu với người dùng.
onResume()	Phương thức onResume()được gọi khi người dùng tương tác với các ứng dụng.
onPause()	Phương thức onPause() tạm dừng một activity, không nhận dữ liệu do người dùng nhập vào và không thế thực thi lệnh nào. Phương thức này được gọi khi activity hiện tại đang được tạm dừng, và activity trước đó đang được tiếp tục.
onStop()	Phương thức onStop() được gọi khi một activity đã không được nhìn thấy trong thời gian dài.
onRestart()	Phương thức onRestart() được gọi khi activity cần được dùng trở lại sau khi bị gọi onStop();
onDestroy()	Phương thức onDestroy() được gọi trước khi hệ thống hủy activity.



Vòng đời của Activity

Vòng đời của một activity: Các activity được quản lý dưới dạng các Activity stack - First-In-Last-Out. Khi một Activity mới được khởi tạo, nó sẽ được đưa lên trên cùng stack, các activity khác muốn chạy trên nền (foreground) trở lại thì cần phải chờ tới khi Activity mới này kết thúc.

2. Intent

Intent trong android được dùng để kết nối các thành phần trong ứng dụng Android. Trong quá trình kết nối, nó cũng có thể yêu cầu thành phần đó thực hiện một tác vụ được định trước. Các thành phần cơ bản của một Intent:

Actions: Là những thứ mà Intent cần thực thi, chẳng hạn như quay số điện thoại, mở URL hoặc chỉnh sửa dữ liệu. Một action đơn giản là một String mô tả cho tác vụ cần thực hiện. Ví dụ: ACTION_VIEW, ACTION_EDIT, ACTION_MAIN…

Data: Đây chính là dữ liệu để intent hoạt động. Nó được biểu diễn dưới dạng Uniform Resource Identifier hoặc Uri trong Android – một kiểu định danh cho một tài nguyên cụ thể. Kiểu của Data yêu cầu (nếu có) tùy thuộc vào action. Ví dụ: Bạn sẽ không muốn tạo một dial number Intent mà lại lấy số điện thoại từ một hình ảnh

Khả năng kết hợp các action và data này cho phép Android biết chính xác intent đang yêu cầu gì và những đối tượng nào có thể thực hiện nó.

Intents có thể được sử dụng để:

- Lưu trữ dữ liệu để mở một Activity mới và truyền dữ liệu cho Activity đó bằng phương thức startActivity(). Một Activity được gọi sau khi kết thúc có thể gửi dữ liệu về cho Activity đã gọi nó bằng phương thức startActivityForResult().
- Lưu trữ dữ liệu để mở hoặc dừng một Service
- Hoặc có thể gọi một Activity bằng Broadcast Receiver

Trong Intent chúng ta có 2 loại gồm Explicit Intent (Intent tường minh), Implicit Intent (Intent không tường minh).

Explicit Intent: Là loại Intent xác định đích danh. Ví dụ trong một ứng dụng Android đơn giản, chúng ta có 2 màn hình, màn hình đầu tiên của ứng dụng là cho người dung login vào ứng dụng (Login Activity), màn hình thứ 2 là chương trình chính của ứng dụng (Main Activity). Khi người dùng nhập đầy đủ và đúng thong tin người dùng thì chuyển qua màn hình chương trình chính. Để di chuyển từ Login Activity qua Main Activity trong trường hợp này chúng ta dùng Explicit Intent. Ở đây chúng ta đã xác định được chính xác là màn hình Main Activity chúng ta cần chuyển đến. Hay nói cách khác Explicit Intent là chúng ta dùng để di chuyển qua lại giữa các màn hình trong cùng một ứng dụng.

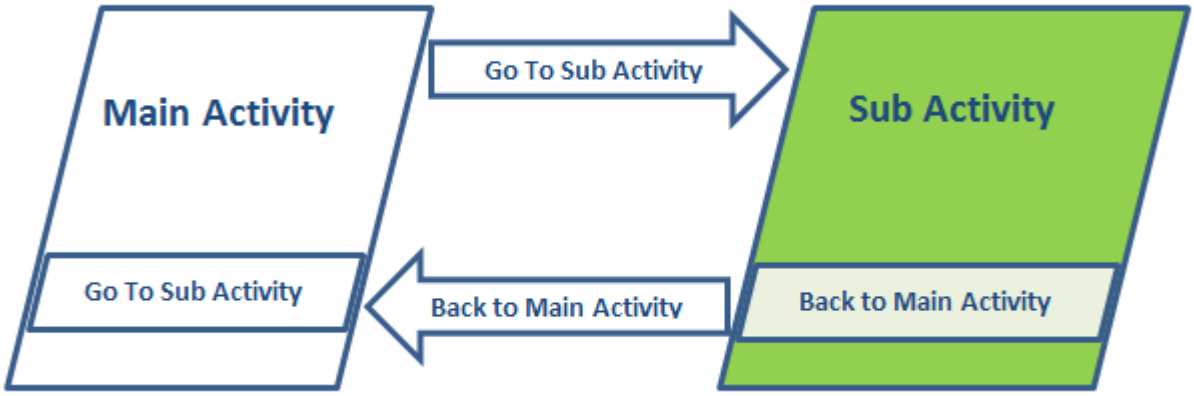
Các phương thức thường dùng của Explicit Intent

Phương thức	**Diễn giải**
.putExtra()	putExtra(Key, value): Chỉ truyền được dữ liệu kiểu cơ bản: Int, Float, Char, Double, Boolean, String, …
.getStringExtra	Nhận giá trị chuỗi từ Activity khác gửi đến
.getIntExtra	Nhận giá trị số nguyên từ Activity khác gửi đến
.get<KDL>Extra	Nhận giá trị <KDL> Activity khác gửi đến. KDL là các dữ liệu kiểu cơ bản: Int, Float, Char, Double, Boolean, String, …

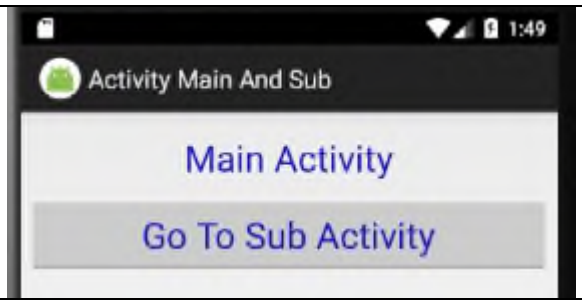
Cách tạo và sử dụng Explicit Intent

```
//Khai báo và truyền tham số cho Intent
Intent intentsub = new Intent(MainActivity.this, SubActivity.class);
// MainActivity.this là Activity hiện hành, SubActivity.class là Activity cần chuyển đến
//Gọi phương thức startActivity với tham số truyền vào là Intent chứa Activity cần chuyển đến
startActivity(intentsub);
```

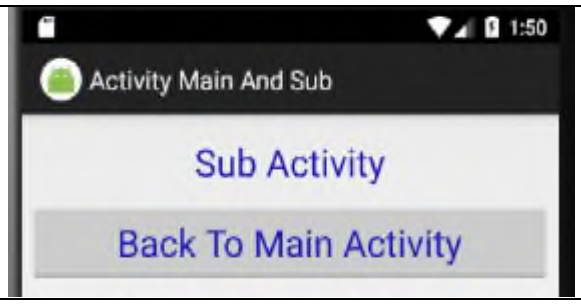
Ví dụ: Trong ứng dụng Android có 2 Activity gồm MainActivity và SubActivity. Trên MainActivity gồm một TextView và một Button có tên "Go To SubActivity". Khi người dùng bấm vào Button "Go To Sub Activity" thì chuyển đến Sub Activity. Tương tự, trên SubActivity, gồm một TextView và một Button có tên "Back To Main Activity". Khi người dùng bấm vào Button "Back To Main Activity" thì chuyển về Main Activity.



Dùng Intent di chuyển qua lại giữa các Activity



Màn hình Main Activity



Màn hình Sub Activity

Code bắt sự kiện Click cho Button "Go To Sub Activity"

```
Button btngotosub=(Button) findViewById(R.id.btnGoToSubId);

btngotosub.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intentmain = new Intent(MainActivity.this, SubActivity.class);
        startActivity(intentmain);
    }
});
```

Code bắt sự kiện Click cho Button "Back To Main Activity"

```
Button btnbacktomain=(Button) findViewById(R.id.btnBackToMainId);

btnbacktomain.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intentsub = new Intent(SubActivity.this, MainActivity.class);
        startActivity(intentsub);
    }
});
```

Implicit Intent: Là loại Intent không xác định đích danh. Implicit Intent được dùng trong các trường hợp chúng ta muốn thực hiện một chức năng nào đó mà sẽ phải chuyển đến một màn hình khác nằm ứng dụng hiện hành. Ví dụ SMS, Call, Camera, Mail, Trình duyệt web, …

Thông thường, chúng ta dùng các intent tường minh để kích hoạt các thành phần trong ứng dụng, còn intent không tường minh để chạy các thành phần của ứng dụng bên thứ 3.

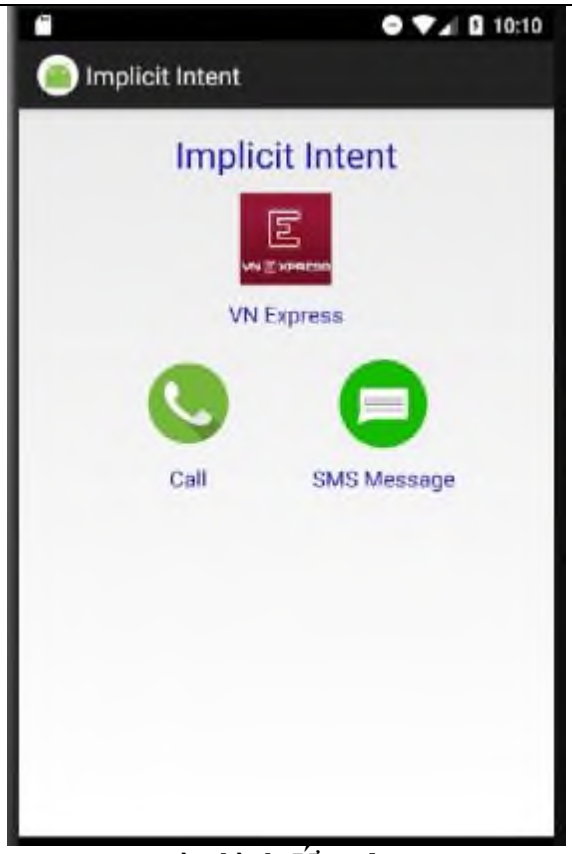
Các Action thường dùng của Implicit Intent

Action	**Diễn giải**
ACTION_VIEW	Hiển thị dữ liệu cho người dùng nhìn thấy.
ACTION_DIAL	Gọi tới một số điện thoại được chỉ định trong dữ liệu.
ACTION_CALL	Thực hiện cuộc gọi tới một người nào đó được chỉ định trong dữ liệu.
ACTION_SENDTO	Gửi dữ liệu cho ai đó được chỉ định trong dữ liệu.
ACTION_SEND	Gửi dữ liệu cho ai đó.
ACTION_WEB_SEARCH	Thực hiện thao tác tìm kiếm trên web.

Cách tạo và sử dụng Implicit Intent

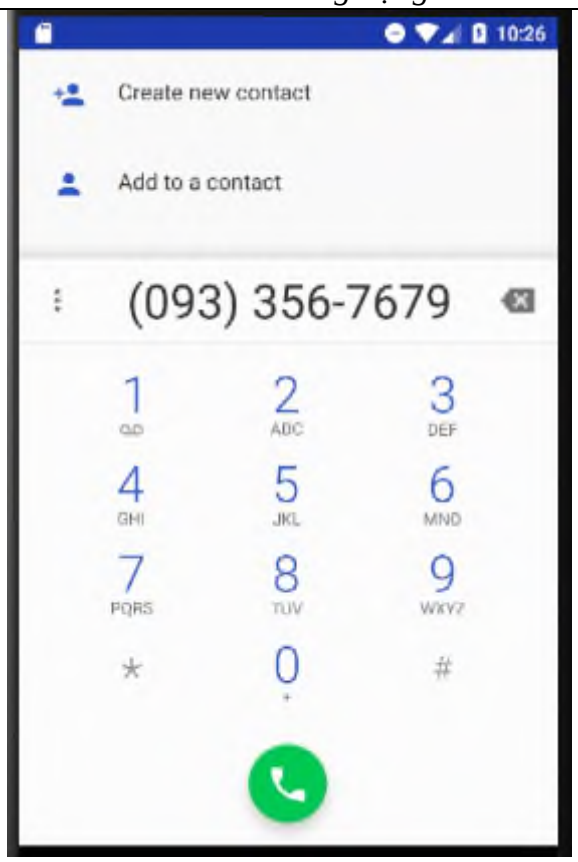
```
//Khai báo Intent
Intent intentVNExpress = new Intent();
//Gán Action cho Intent
intentVNExpress.setAction(Intent.ACTION_VIEW);
//Gán dữ liệu cho Intent
intentVNExpress.setData(Uri.parse("https://vnexpress.net"));
//Gọi phương thức mở cửa sổ trình duyệt web và truy cập vào trang vnexpress.net
startActivity(intentVNExpress);
```

Ví dụ: Trong ứng dụng Android có một Activity có tên MainActivity. Trên MainActivity có 3 ImageView và 3 TextView như hình bên dưới. Khi người dùng Click vào biểu tượng của VN Express thì chương trình mở cửa sổ trình duyệt và truy cập vào trang web *vnexpress.net.* Tương tự cho SMS Message và Call như màn hình bên dưới.



Màn hình Ứng dụng

Màn hình khi Click VN Express



Màn hình Dial

Màn hình SMS message

Code bắt sự kiện Click cho các image như sau:

```
ImageView imageVNExpress=(ImageView) findViewById(R.id.imageviewId);
ImageView imagecall=(ImageView) findViewById(R.id.imgCallId);
ImageView imagesms=(ImageView) findViewById(R.id.imgsmsId);
//Open vnexpress.net
imageVNExpress.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intentVNExpress = new Intent();
        intentVNExpress.setAction(Intent.ACTION_VIEW);
        intentVNExpress.setData(Uri.parse("https://vnexpress.net"));
        startActivity(intentVNExpress);
    }
});
//Dial
imagecall.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intentcall = new Intent();
        intentcall.setAction(Intent.ACTION_DIAL);
        intentcall.setData(Uri.parse("tel:0933567679"));
        startActivity(intentcall);
    }
});
//SMS Message
imagesms.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intentsms = new Intent(Intent.ACTION_SENDTO);
        intentsms.putExtra("sms_body","Chào bạn ...");
        intentsms.setData(Uri.parse("sms:0933567679"));
        startActivity(intentsms);
    }
});
```

3. Truyền dữ liệu giữa các Activity bằng putExtra

Để truyền dữ liệu giữa các Activity, chúng ta sử dụng phương thức putExtra của Intent có 2 tham số truyền vào ở đây là các cặp key và value. Thông qua Intent, bên nhận dựa vào key để nhận dữ liệu bằng phương thức get<KDL>Extra, KDL là kiểu dữ liệu tương ứng khi putExtra gửi đi.

Truyền dữ liệu kiểu String bằng putExtra

```
//Khai báo Intent và gán dữ liệu kiểu String vào Intent bằng phương thức putExtra
Intent intentSendString = new Intent(MainActivity.this, GetStringExtraActivity.class);
 intentSendString.putExtra("dulieuString","Nội dung dữ liệu");
startActivity(intentSendString);
```

Nhận dữ liệu kiểu String bằng getStringExtra

```
//Khai báo và nhận dữ liệu kiểu String từ Intent bằng phương thức getStringExtra
Intent intentString = getIntent();
String noiDungString = intentString. getStringExtra(("dulieuString");
```

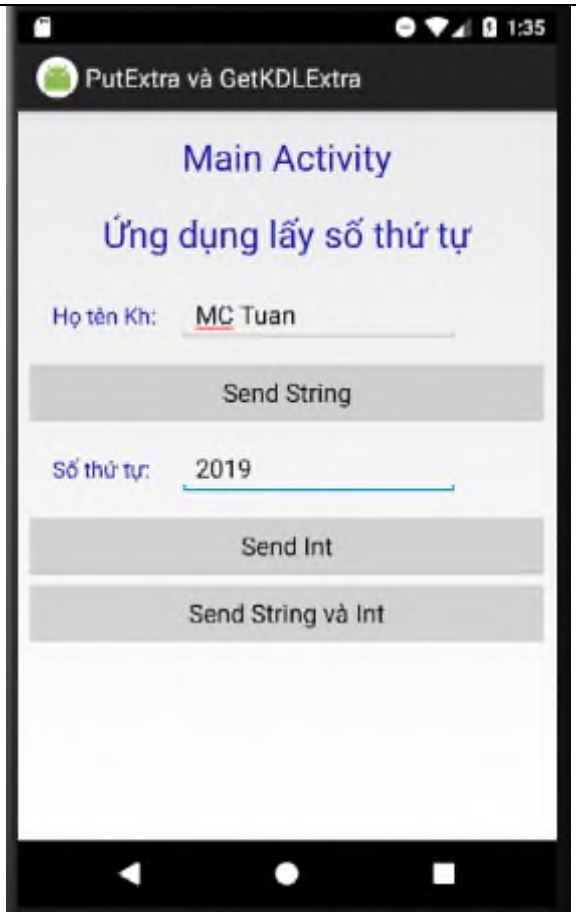
Truyền dữ liệu kiểu Int bằng putExtra

```
//Khai báo Intent và gán dữ liệu kiểu Int vào Intent bằng phương thức putExtra
Intent intentSendInt = new Intent(MainActivity.this, GetStringExtraActivity.class);
 intentSendInt.putExtra("dulieuInt", 2019);
startActivity(intentSendInt);
```

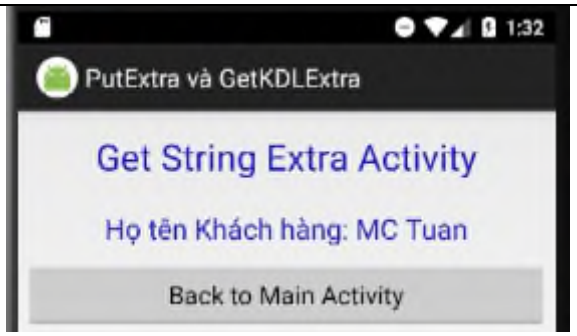
Nhận dữ liệu kiểu Int bằng getIntExtra

```
//Khai báo và nhận dữ liệu kiểu String từ Intent bằng phương thức getStringExtra
Intent intentInt = getIntent();
String noiDungInt = intentInt.getIntExtra(("dulieuInt",0); // 0 là giá trị mặc định
```

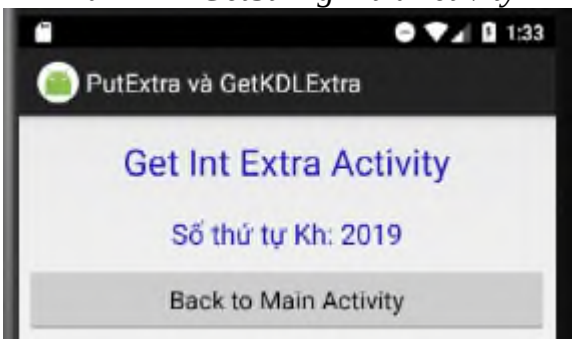
Ví dụ: Tạo ứng dụng có 3 Activity gồm Main Activity, GetStringExtraActivity, GetIntExtraActivity. Màn hình GetStringExtraActivity dùng để nhận dữ liệu kiểu String từ Main Activity gửi đến. Màn hình GetIntExtraActivity dùng để nhận dữ liệu kiểu Int từ Main Activity gửi đến. Màn hình GetStringIntExtraActivity để nhận cùng lúc cả String và Int.



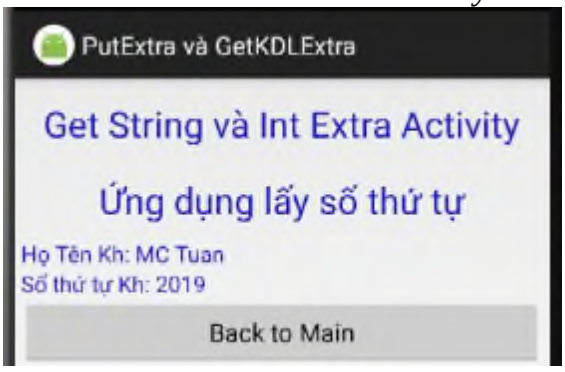
Màn hình Main Activity



Màn hình GetStringExtraActivity



Màn hình GetIntExtraActivity



Màn hình GetStringIntExtraActivity

Code bắt sự kiện Click cho Button Send String như sau:

```
btnSendString.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intentString = new Intent(MainActivity.this, GetStringExtraActivity.class);
        intentString.putExtra("StringHoTen", editHoten.getText().toString());
        startActivity(intentString);
    }
});
```

Code nhận dữ liệu kiểu String từ Main Activity tại GetStringExtraActivity như sau:

```
TextView txthoten=(TextView) findViewById(R.id.TxtViewHTId);
Intent intentGetString = getIntent();
String hoten=intentGetString.getStringExtra("StringHoTen");
txthoten.setText("Họ tên Khách hàng: "+hoten);
```

Code bắt sự kiện Click cho Button Send Int như sau:

```
btnSendInt.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intentInt=new Intent(MainActivity.this, GetIntExtraActivity.class);
        int stt;
        stt=Integer.parseInt(editStt.getText().toString());
        intentInt.putExtra("IntSoTT",stt);
        startActivity(intentInt);
    }
});
```

Code nhận dữ liệu kiểu Int từ Main Activity tại GetIntExtraActivity như sau:

```
TextView textViewSott=(TextView) findViewById(R.id.TxtViewgetSottId);
Intent intentgetint=getIntent();
textViewSott.setText("Số thứ tự Kh: "+intentgetint.getIntExtra("IntSoTT",0));
```

Code bắt sự kiện Click cho Button Send String và Int như sau:

```
btnSendStringInt.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intentStringInt = new Intent(MainActivity.this,
GetStringIntExtraActivity.class);
        intentStringInt.putExtra("StringHoTen", editHoten.getText().toString());
        int stt=Integer.parseInt(editStt.getText().toString());
        intentStringInt.putExtra("IntSoTT",stt);
        startActivity(intentStringInt);
    }
});
```

Code nhận dữ liệu kiểu String và Int từ Main Activity tại GetStringIntExtraActivity như sau:

```
TextView textViewhotenkh=(TextView) findViewById(R.id.TxtViewHoTen_Id);
TextView textViewsott=(TextView) findViewById(R.id.TxtViewSoTT_Id);
//Nhận dữ liệu từ Itent
Intent intentgetStringInt = getIntent();
textViewhotenkh.setText("Họ Tên Kh:
"+intentgetStringInt.getStringExtra("StringHoTen").toString());
textViewsott.setText("Số thứ tự Kh: "+intentgetStringInt.getIntExtra("IntSoTT",0) );
```

Truyền dữ liệu Array bằng putExtra: Kiểu dữ liệu của các phần tử trong mảng là các kiểu dữ liệu cơ bản như Int, Float, Char, Double, Boolean, String, …

```
Intent intentSendStringArray = new
Intent(MainActivity.this,GetStringArrayPutExtra.class);
String[] ArrayKhoaHoc = {"Lập Trình Android Cơ Bản","Lập Trình Android Nâng
Cao","Lập Trình C#","Quảng cáo FaceBook","Quảng Cáo Google Adword" };
intentSendStringArray.putExtra("KhoaHoc",ArrayKhoaHoc);
startActivity(intentSendStringArray);
```

Nhận dữ liệu kiểu Array bằng get<KDL>ArrayExtra

```
//Nhận kết quả String Array từ Main Activity
Intent intentgetStringArray = getIntent();
String[] getStringKhoaHocArray;
getStringKhoaHocArray=intentgetStringArray.getStringArrayExtra("KhoaHoc");
```

Truyền dữ liệu Object bằng putExtra: Để truyền dữ liệu giữ các Activity bằng putExtra thì khi khai báo class bắt buộc phải **implements** Serializable.

```
//Khai báo Class
public class KhoaHoc implements Serializable {

    private String TenKhoaHoc;
    private Long HocPhi;

    public KhoaHoc(String tenKhoaHoc, Long hocPhi) {
        TenKhoaHoc = tenKhoaHoc;
        HocPhi = hocPhi;
    }

    public String getTenKhoaHoc() {
        return TenKhoaHoc;
    }

    public void setTenKhoaHoc(String tenKhoaHoc) {
        TenKhoaHoc = tenKhoaHoc;
    }
```

```
    public Long getHocPhi() {
        return HocPhi;
    }

    public void setHocPhi(Long hocPhi) {
        HocPhi = hocPhi;
    }
}
```

Truyền dữ liệu kiểu Object bằng putExtra thông qua Intent giữa các Activity trong Android.

```
//Khai báo biến Object Khóa học và lấy dữ liệu
KhoaHoc khoaHoc=new KhoaHoc(("Lập Trình Android", 6000000);
//Khai báo biến Intent
Intent intentSendObject = new Intent(MainActivity.this,GetExtraObject.class);
intentSendObject.putExtra("khoahocObject",khoaHoc);
startActivity(intentSendObject);
```

Nhận dữ liệu Object gửi bằng putExtra: Để nhận dữ liệu Object từ các Activity gửi đến bằng putExtra thông qua Intent thì khi khai báo class bắt buộc phải **implements** Serializable và khi nhận bằng **getSerializableExtra**.

```
//Khai báo Intent
Intent intentgetKhoaHocObject = getIntent();
//Nhận dữ liệu Object từ Intent
KhoaHoc khoaHoc =
(KhoaHoc)intentgetKhoaHocObject.getSerializableExtra("khoahocObject");
```

4. Truyền dữ liệu giữa các Activity bằng đóng gói dữ liệu Bundle

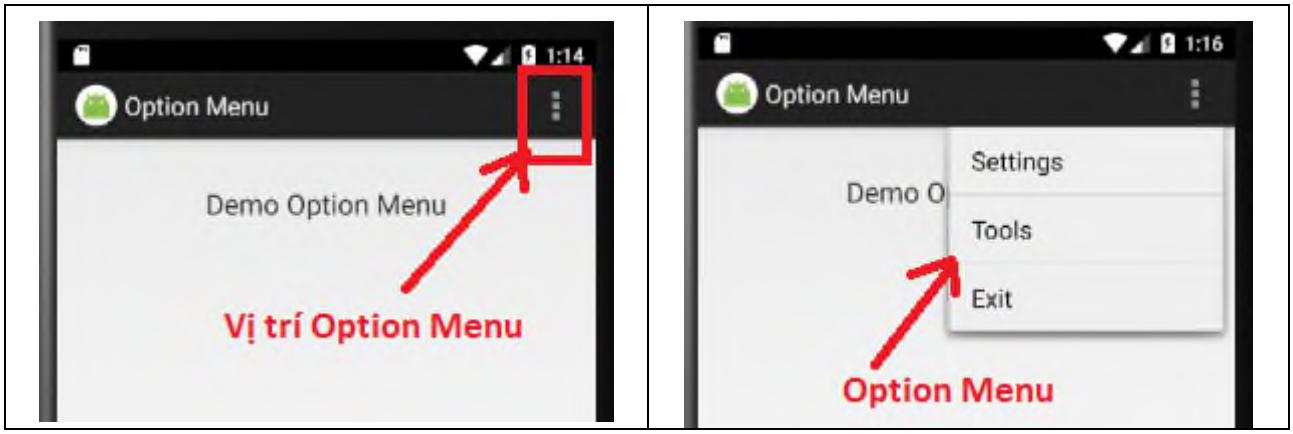
Bundle trong android là một đối tượng dữ liệu được tạo ra nhằm mục đích đóng gói các dữ liệu cần được truyền qua lại giữa các thành phần của ứng dụng trong Android thông qua Intent.

CHƯƠNG 7. CỬA SỔ THÔNG BÁO VÀ MENU

Menu là một trong những Control đặc biệt, dùng để chứa các chức năng của chương trình hoặc các tùy chỉnh dành riêng cho ứng dụng. Android cung cấp cho chúng ta các loại menu như : Option Menu, Context Menu, Popup Menu.

1. Option Menu

Option Menu là danh sách các chức năng chính của ứng dụng, thường được hiển thị ở góc trên, bên phải của điện thoại di động.



Màn hình Option Menu

Trên màn hình điện thoại dị động Ở màn hình Option Menu trên, chúng ta thấy vị trí và các mục chức năng của chương trình gồm Setting, Tools, Exit. Để tạo được Option Menu trên ứng dụng, chúng ta cần Override phương thức onCreateOptionsMenu và xử lý sự kiện Click trên menu chúng ta cần Override phương thức onOptionsItemSelected.

Phương thức của Option Menu thường dùng:

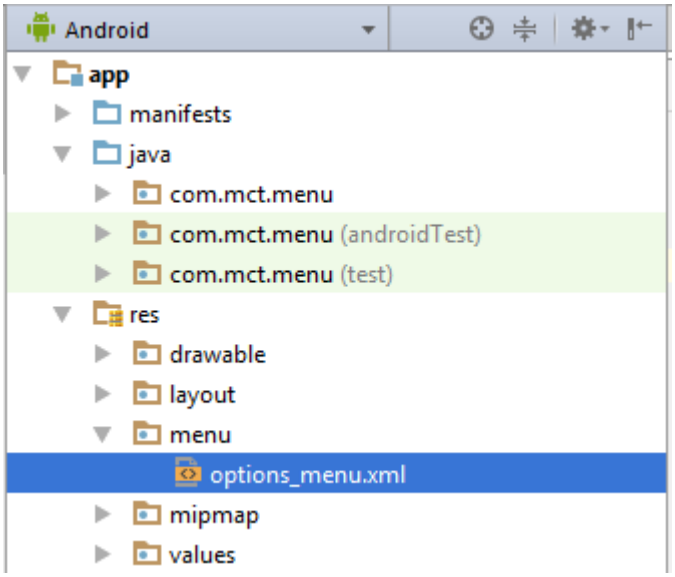
Thuộc tính	Mô tả
onCreateOptionsMenu	onCreateOptionsMenu(): Phương thức onCreateOptionsMenu() sẽ thực hiện các công việc khởi tạo menu cho đối tượng Activity, ở đây chúng ta dùng phương thức inflate() của lớp android.view.MenuInflater để lấy dữ liệu của menu từ file options_menu.xml về sử dụng. Phương thức này nhận vào một đối tượng android.view.Menu.
onOptionsItemSelected	onOptionsItemSelected(): Phương thức onOptionsItemSelected() sẽ xử lý sự kiện click menu. Phương thức này nhận vào một đối tượng android.view.MenuItem.

Để xây dựng ứng dụng Option Menu và xử lý các sự kiện trên Option Menu như hình trên, chúng ta thực hiện các bước như sau:

Bước 1: Tạo project có tên Option Menu và thiết kế giao diện chương trình như hình trên.

Bước 2: Trong thư mục res chúng ta tạo mới thư mục có tên menu bằng cách nhấn chuột phải tại thư mục res chọn new->Directory->nhập vào tên là menu. Tiếp theo, nhấn chuột phải tại thư mục menu chọn new->Menu resource file-> nhập vào option_menu. Mở file option_menu.xml nhập vào nội dung như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/settings"
        android:title="@string/menu_Settings" />
    <item android:id="@+id/tools"
        android:title="@string/menu_Tools" />
    <item android:id="@+id/Thoat"
        android:title="@string/menu_Thoat" />
</menu>
```



Màn hình vị trí file option_menu.xml

Bước 3: Tạo Option Menu và xử lý các sự kiện trên Option Menu như hình trên bằng cách Override phương thức onCreateOptionsMenu và xử lý sự kiện Click trên menu chúng ta Override phương thức onOptionsItemSelected trong file .java như sau:

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.options_menu, menu);
    return true;
}
@Override
```

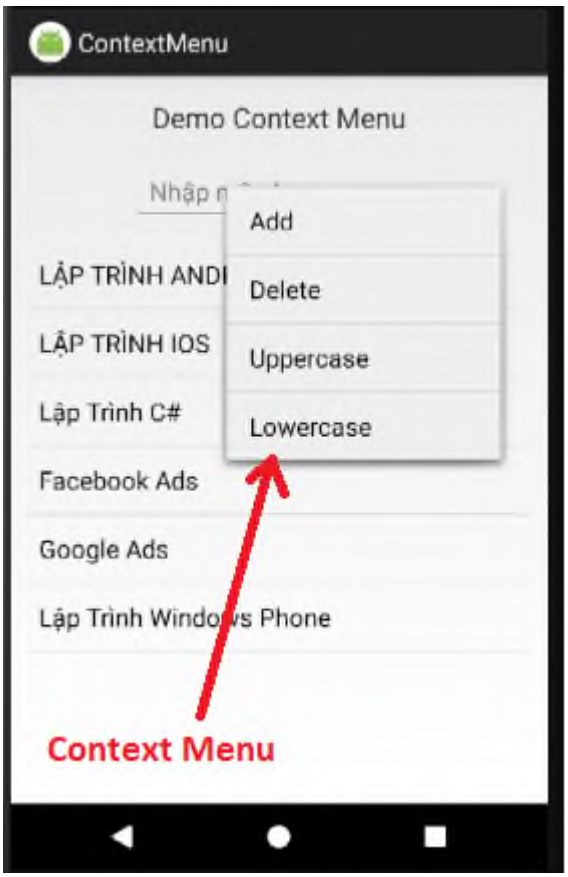
```
public boolean onOptionsItemSelected(MenuItem item) {
   switch (item.getItemId())
   {
      case R.id.settings:
         Toast.makeText(MainActivity.this, "Bạn đã chọn Settings",
               Toast.LENGTH_SHORT).show();
         return true;
      case R.id.tools:
         Toast.makeText(MainActivity.this, "Bạn đã chọn Tools",
               Toast.LENGTH_SHORT).show();
         return true;
      case R.id.Thoat:
         Toast.makeText(MainActivity.this, "Bạn đã chọn Exit",
               Toast.LENGTH_SHORT).show();
         return true;
      default:
         return super.onOptionsItemSelected(item);
   }
}
}
```

Trong phương thức onCreateOptionsMenu chúng ta dùng hàm getMenuInflater() để lấy đối tượng MenuInflater và sau đó dùng hàm Inflate để nạp XML Menu cho Activity. Trong hàm Inflate có 2 tham số, tham số thứ 1 là R.menu.*options_menu* và tham số thứ 2 là biến menu trong hàm onCreateOptionsMenu.

Trong phương thức onOptionsItemSelected chúng ta dùng cấu trúc switch để kiểm tra Menu Item nào đang được người dùng chọn để xử lý sự kiện tương ứng. Trong hàm onOptionsItemSelected có biến item được truyền vào, biến item này chính là item đang được người dùng chọn. Để kiểm tra dễ dàng chúng ta dùng phương thức getItemId để so sánh.

2.Context Menu

Context Menu là loại menu xuất hiện khi người dùng nhấn một khoảng thời gian khá lâu trên một View bất kỳ nhằm mục đích thực hiện một số chức năng phù hợp với các giá trị hiển thị trên view cần xử lý khi.



Màn hình Context Menu với sự kiện LongClick

Phương thức của Context Menu thường dùng:

Thuộc tính	**Mô tả**
registerForContextMenu	registerForContextMenu(view): Phương thức này được sử dụng để gắn một ContextMenu cho một View
onCreateContextMenu	onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo menuInfo): Phương thức này sẽ thực hiện các công việc khởi tạo menu cho đối tượng Activity, ở đây chúng ta dùng phương thức inflate() của lớp android.view.MenuInflater để lấy dữ liệu của menu từ file options_menu.xml về sử dụng. Phương thức này nhận vào một đối tượng android.view.Menu.
onContextItemSelected	onContextItemSelected(MenuItem item): Phương thức onContextItemSelected() sẽ làm nhiệm vụ xử lý sự kiện click trên từng item.

Để xây dựng ứng dụng Context Menu và xử lý các sự kiện trên Context Menu như hình trên, chúng ta thực hiện các bước như sau:

Bước 1: Tạo project có tên Context Menu và thiết kế giao diện chương trình như hình trên. Trong màn hình chương trình chính có 1 TextView, 1 EditText và 1 ListView chứa các danh sách môn học.

File XML của chương trình chính như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.mct.contextmenu.MainActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/app_title"
        android:textSize="20dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.016"
        android:layout_marginRight="8dp"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_marginTop="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        android:id="@+id/textView" />
    <EditText
        android:id="@+id/edtmonhocId"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPersonName"
        android:hint="@string/monhoc_hinttitle"
        android:layout_marginRight="8dp"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_marginLeft="8dp"
        app:layout_constraintLeft_toLeftOf="parent"
        android:layout_marginTop="20dp"
        app:layout_constraintTop_toBottomOf="@+id/textView"
        app:layout_constraintHorizontal_bias="0.503" />
    <ListView
        android:id="@+id/lvmonhocid"
        android:layout_width="400dp"
        android:layout_height="414dp"
        android:padding="20dp"
        app:layout_constraintBottom_toBottomOf="parent"
        android:layout_marginBottom="8dp"
        android:layout_marginRight="8dp"
        app:layout_constraintRight_toRightOf="parent"
```

```
        android:layout_marginLeft="8dp"
        app:layout_constraintLeft_toLeftOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Tạo danh sách các môn học trong File String.XML có nội dung như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">ContextMenu</string>
    <string name="app_title">Demo Context Menu</string>
    <string name="monhoc_hinttitle"> Nhập môn học </string>
    <string-array name="arrmonhoc">
        <item> Lập Trình Android </item>
        <item> Lập Trình IOS </item>
        <item> Lập Trình C# </item>
        <item> Facebook Ads </item>
        <item> Google Ads </item>
        <item> Lập Trình Windows Phone </item>
    </string-array>
</resources>
```

Nạp dữ liệu từ file XML lên ListView như sau:

```
public class MainActivity extends Activity {
    //Khai báo biến
    private ListView lstmonhoc;
    private ArrayAdapter<String> adapter;
    String [] monhoc;
    List<String> monhoc_list;

    EditText editTextmonhoc;
    String emonhoc;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        lstmonhoc  =  (ListView)findViewById(R.id.lvmonhocid);
        //Lấy danh sách môn học từ XML
        monhoc =  getResources().getStringArray(R.array.arrmonhoc);
        monhoc_list    = new ArrayList<String>(Arrays.asList(monhoc));
        adapter      =   new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,
monhoc_list);
        lstmonhoc.setAdapter(adapter);
        //Đăng ký ContextMenu cho ListView Môn học
        registerForContextMenu(lstmonhoc);
```

```
        editTextmonhoc=(EditText) findViewById(R.id.edtmonhocId);
    }
```

Bước 2: Trong thư mục res chúng ta tạo mới thư mục có tên menu bằng cách nhấn chuột phải tại thư mục res chọn new->Directory->nhập vào tên là menu. Tiếp theo, nhấn chuột phải tại thư mục menu chọn new->Menu resource file-> nhập vào context_menu. Mở file context_menu.xml nhập vào nội dung như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/addId"
        android:title="Add"/>
    <item android:id="@+id/delId"
        android:title="Delete"/>
    <item android:id="@+id/upId"
        android:title="Uppercase"/>
    <item android:id="@+id/lowId"
        android:title="Lowercase"/>
</menu>
```

Bước 3: Tạo Context Menu và xử lý các sự kiện trên Context Menu như hình trên bằng cách Override phương thức onCreate ContextMenu trong file .java như sau:

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                                ContextMenu.ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.context_menu, menu);
}
```

Xử lý sự kiện Click trên Item Context menu chúng ta Override phương thức onContextItemSelected trong file .java như sau:

```
@Override
public boolean onContextItemSelected(MenuItem item) {
    AdapterView.AdapterContextMenuInfo info =
(AdapterView.AdapterContextMenuInfo) item.getMenuInfo();
    int pos = info.position;
    String i = adapter.getItem(pos);
    switch (item.getItemId())
    {
        case R.id.addId:
            emonhoc=editTextmonhoc.getText().toString();
            monhoc_list.add(emonhoc);
            adapter.notifyDataSetChanged();
            return true;
        case R.id.delId:
            monhoc_list.remove(pos);
```

```
            adapter.notifyDataSetChanged();
            return true;
        case R.id.upId:
            String upln = i.toUpperCase();
            monhoc_list.set(pos, upln);
            adapter.notifyDataSetChanged();
            return true;
        case R.id.lowId:
            String lpln = i.toLowerCase();
            monhoc_list.set(pos,lpln);
            adapter.notifyDataSetChanged();
            return true;
        default:
            return super.onContextItemSelected(item);
    }
}// Kết thúc onContextItemSelected
```

3.Popup Menu

Popup Menu hiển thị menu bên dưới văn bản neo nếu không gian có sẵn nếu không ở trên các văn bản neo. Nó biến mất nếu bạn nhấp vào bên ngoài popup menu. Các android.widget.PopupMenu là lớp con trực tiếp của lớp java.lang.Object.

Màn hình Popup Menu

Phương thức của Popup Menu thường dùng:

Thuộc tính	**Mô tả**
setOnMenuItemClickListener	setOnMenuItemClickListener: Phương thức này được sử dụng để gắn listener cho đối tượng PopupMenu, khác với menu ngữ cảnh và menu tùy chọn là 2 loại menu này đã có sẵn trong Activity và View nên chúng ta không cần gọi trực tiếp ra như đối tượng menu popup.
show	show(): Sau khi tạo PopupMenu, chúng ta phải gọi phương thức show() nếu muốn hiện menu này ra.
onMenuItemClick	onMenuItemClick(): Phương thức onOptionsItemSelected() sẽ xử lý sự kiện click menu. Phương thức này nhận vào một đối tượng android.view.MenuItem.

CHƯƠNG 8. SQLITE TRONG ANDROID

8.1 Giới thiệu SQLite

Là cơ sở dữ liệu nhỏ tương tự như MySQL, SQL server… với đặc điểm là nhỏ , gọn và nhẹ.Trong lập trình android thì SQLite sẽ được lưu trong ổ đĩa của thiết bị chứ không phải lưu trên server hay bất kì nơi nào khác.

Việc truy vấn trong SQLite sẽ tương tự như các hệ cơ sở dữ liệu khác nên bạn sẽ không phải gặp khó khăn gì khi sử dụng và bạn cũng chẳng cần user , mật khẩu khi truy cập vào trong database gì cả.

Tại sao dùng SQLite để lưu trữ

Với các bài trước ta học thì việc lưu trữ một đối tượng khá là khó, với Shared Preferences thì bạn phải chuyển nó sang String rồi sau đó dưới dạng **Key** – **Value**, còn với kiểu lưu External hay Internal thì lại khá cực.

Bây giờ có SQLite sẽ giải quyết cho bạn vấn đề này, nó được sử dụng khi nào các bạn biết không?

SQLite được sử dụng khi nào?

Nó được sử dụng khi yêu cầu bài toán đưa ra là hãy lưu những dữ liệu lớn ở dạng Object, dạng text… xuống thiết bị để có thể lấy ra bất kỳ khi nào.Dữ liệu lớn ở đây không phải là nặng đâu nhé các bạn mà nó nhiều ấy vì dữ kiệu dạng text, object lưu xuống cũng khá nhẹ cho dù nó nhiều.

Mình sẽ đưa ra những ví dụ kinh điển bạn sẽ hình dung ra ngay như sau:

Ex1:

Ứng dụng **ZingMp3** tiếp đi ha, bạn sẽ có chức năng là lưu bài hát vào danh sách yêu thích đúng không? Một bài hát thường sẽ có các thuộc tính như: tên, url, lyric, ca sĩ… Đây là một Oject đúng không nào? Ứng dụng sẽ lưu danh sách bài hát yêu thích vào database để sau này truy xuất ta hiển thị thôi.

Ex2:

Ứng dụng có chức năng lưu trữ lịch sử tìm kiếm thì cũng có thể sử dụng sql để lưu lại danh sách này.

Ex3:

Ứng dụng báo thức có thể sử dụng sqlite để lưu danh sách các báo thức đã cài đặt.

Tuy nhiên theo mình thấy thì SQLite lưu trữ dữ liệu không được lớn như những hệ cơ sở dữ liệu khác vì nó tích hợp trong thiết bị android mà, nếu quá nặng thì ứng dụng đó sẽ rất chậm.Nói gì nói thì nó chỉ đáp ứng với khoảng dữ liệu nhỏ vài ngàn dòng gì đó thôi chứ lớn quá thì nó sẽ không dùng SQLite để lưu đâu các bạn.

8.2 SQLite trong Android

Cách sử dụng SQLite trong lập trình Android

Tương tự như các hệ quản trị cơ sở dữ liệu khác thì khi thao tác với Database thì bạn sẽ có những hành động cổ điển đó là **CRUD** mà mình đã nói ở trên rồi, ở loạt này chúng ta cũng sẽ tìm hiểu những cái chính đó thôi chứ không đi quá sâu được vì còn rất nhiều bài ở đằng sau nữa.

Bài toán mình đưa ra là lưu lại danh sách Student vào trong SQLite, chính vì thế mình sẽ đi phân tích tất cả những thuộc tính mà chúng ta sẽ phải khởi tạo cho lớp Student từ đó đi xây dựng Database cho dữ liệu này.

Dự kiến class model **Student.java** sẽ có bao gồm những thuộc tính như sau: id, name, number, email, address.

Vậy class **Student.java** sẽ có những thuộc tính và các phương thức getter, setter sau:

```
package cheng.com.bai4sqlite.model;


/**
 * Created by chien on 11/13/16.
 */


public class Student {
    private int id;
    private String name;
```

```
    private String number;
    private String email;
    private String address;

    public Student() {
    }

    public Student(String name, String number, String email, String address) {
        this.name = name;
        this.number = number;
        this.email = email;
        this.address = address;
    }

    public Student(int id, String name, String number, String email, String address) {
        this.id = id;
        this.name = name;
        this.number = number;
        this.email = email;
        this.address = address;
    }
```

```
public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getNumber() {
    return number;
}
```

```
public void setNumber(String number) {
    this.number = number;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getAddress() {
    return address;
}

public void setAddress(String address) {
    this.address = address;
}

@Override
```

```
    public String toString() {

        return "Name:" +getName() + " \n" +"Number: "+getNumber() +"\n" + "Email: "+getEmail() +"\n" + "Address:" +getAddress();

    }

}
```

Vậy bản mô phỏng các thuộc tính trong Table Student khi lưu vào database sẽ như sau:

Show 10 entries

Search:

Field	Type	Key
id	int	primary key
name	Text	
number	Text	
email	Text	
address	Text	

Showing 1 to 5 of 5 entries

PreviousNext

Vậy quy trình chúng ta sẽ làm làm gì nhỉ? Khởi tạo một database có tên là "**student_list**", một table tên là "**student**" và sau đó là có câu truy vấn **CRUD**.

Đây là class **DBManager.java** , ở đây mình đã khởi tạo một SQLite tên là ***student_list*** và một table là ***student***, và sau đó là tất cả các phương thức truy vấn và bên dưới mình sẽ giải thích cụ thể cho các bạn hiểu nhé.

```
package com.cheng.sqlite.database;
```

```
import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import android.util.Log;

import android.widget.Toast;


import com.cheng.sqlite.model.Student;


import java.util.ArrayList;

import java.util.List;



/**

 * Created by chien on 11/13/16.

 */


public class DBManager extends SQLiteOpenHelper {


    public static final String DATABASE_NAME ="student_list";
```

```
private static final String TABLE_NAME ="student";

private static final String ID ="id";

private static final String NAME ="name";

private static final String EMAIL ="email";

private static final String NUMBER ="number";

private static final String ADDRESS ="address";


private Context context;


public DBManager(Context context) {

    super(context, DATABASE_NAME,null, 1);

    Log.d("DBManager", "DBManager: ");

    this.context = context;

}


@Override

public void onCreate(SQLiteDatabase db) {

    String sqlQuery = "CREATE TABLE "+TABLE_NAME +" (" +

            ID +" integer primary key, "+

            NAME + " TEXT, "+

            EMAIL +" TEXT, "+
```

```
            NUMBER+" TEXT," +
            ADDRESS +" TEXT)";
    db.execSQL(sqlQuery);
    Toast.makeText(context, "Create successfylly", Toast.LENGTH_SHORT).show();
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS "+TABLE_NAME);
    onCreate(db);
    Toast.makeText(context, "Drop successfylly", Toast.LENGTH_SHORT).show();
}

//Add new a student
public void addStudent(Student student){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(NAME, student.getName());
    values.put(NUMBER, student.getNumber());
    values.put(EMAIL, student.getEmail());
    values.put(ADDRESS, student.getAddress());
```

```
    //Neu de null thi khi value bang null thi loi


    db.insert(TABLE_NAME,null,values);


    db.close();
}


/*
Select a student by ID
 */


public Student getSdtudentById(int id){
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.query(TABLE_NAME, new String[] { ID,
                NAME, EMAIL,NUMBER,ADDRESS }, ID + "=?",
            new String[] { String.valueOf(id) }, null, null, null, null);
    if (cursor != null)
        cursor.moveToFirst();


    Student student = new Student(cursor.getString(1),cursor.getString(2),cursor.getString(3),cursor.getString(4));
```

```
        cursor.close();
        db.close();
        return student;
    }

    /*
    Update name of student
     */

    public int Update(Student student){
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();

        values.put(NAME,student.getName());

        return db.update(TABLE_NAME,values,ID +"=?",new String[] {
String.valueOf(student.getId())});


    }
```

```
/*
 Getting All Student
  */

public List<Student> getAllStudent() {
    List<Student> listStudent = new ArrayList<Student>();
    // Select All Query
    String selectQuery = "SELECT  * FROM " + TABLE_NAME;

    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery(selectQuery, null);

    if (cursor.moveToFirst()) {
        do {
            Student student = new Student();
            student.setId(cursor.getInt(0));
            student.setName(cursor.getString(1));
            student.setEmail(cursor.getString(2));
            student.setNumber(cursor.getString(3));
            student.setAddress(cursor.getString(4));
            listStudent.add(student);
```

```
        } while (cursor.moveToNext());
    }
    cursor.close();
    db.close();
    return listStudent;
  }
  /*
  Delete a student by ID
   */
  public void deleteStudent(Student student) {
    SQLiteDatabase db = this.getWritableDatabase();
    db.delete(TABLE_NAME, ID + " = ?",
          new String[] { String.valueOf(student.getId()) });
    db.close();
  }
/*
Get Count Student in Table Student
 */
  public int getStudentsCount() {
    String countQuery = "SELECT  * FROM " + TABLE_NAME;
    SQLiteDatabase db = this.getReadableDatabase();
```

```
        Cursor cursor = db.rawQuery(countQuery, null);
        cursor.close();


        // return count
        return cursor.getCount();
    }
}
```

Đây là class chúng ta sẽ thao tác với SQLite nhé các bạn, bạn muốn làm gì chỉ cần khởi tạo đối tượng **DBManager** rồi gọi phương thức tương ứng ra thôi.Tuy nhìn thấy nó nhiều vậy thôi chứ thực chất nó không có gì ghê gớm cả, mình sẽ giải thích từng phương thức cho các bạn hiểu ngay thôi.

Để sử dụng được SQLite trong Android thì bạn phải tạo một Class implement class **SQLiteOpenHelper**, đây là điều bắt buộc nhé và khi implement class này thì bạn phải ghi đè các phương thức onCreate() và phương thức onUpgrade() và phải khởi tạo constructor cho class đó.Ở ví dụng trên là mình tạo class **DBManager** và implement SQLiteOpenHelper và các phương thức mà mình phải ghi đè đó là:

```
@Override
    public void onCreate(SQLiteDatabase db) {
        String sqlQuery = "CREATE TABLE "+TABLE_NAME +" (" +
                ID +" integer primary key, "+
                NAME + " TEXT, "+
                EMAIL +" TEXT, "+
                NUMBER+" TEXT," +
```

```
            ADDRESS +" TEXT)";
        db.execSQL(sqlQuery);
        Toast.makeText(context, "Create successfylly", Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS "+TABLE_NAME);
        onCreate(db);
        Toast.makeText(context, "Drop successfylly", Toast.LENGTH_SHORT).show();
    }
```

Và constructor:

```
public DBManager(Context context) {
        super(context, DATABASE_NAME,null, 1);
        this.context = context;
    }
```

Tại sao phải ghi đè các phương thức trên nhỉ? Như chúng ta đã biết thì một quá trình lưu một Student xuống database thì phải trải qua các bước là:

- Tạo database
- Tạo table
- Add student

Vậy ở trên constructer **DBManager()** chính là quá trình tạo database, còn hàm **onCreate()** chính là quá trình tạo table.Còn thằng **OnUpdate()** sẽ được gọi khi có một sự thay đổi liên quan đến cấu trúc dữ liệu như thêm 1 table, xoá table…

Vậy nó được chạy khi nào, bây giờ nếu như ban gọi như sau trong MainActivity:

```
DBManager dbManager = new DBManager(this);
```

Thì sẽ chẳng có chuyện gì xảy ra cả đâu nhé, 2 phương thức trên chẳng cái nào chạy cả đâu và nó sẽ chạy khi các bạn thực hiện các lệnh truy vấn như thêm, tìm kiếm, sửa… và sau đó nó sẽ chẳng chạy vào nữa nếu như nó đã được gọi 1 lần.

Dưới đây là các phức mà chúng ta sẽ viết tuỳ theo mục đích của mình, cùng đi phân tích nhé:

Add – Thêm mới

```
//Add new a student
    public void addStudent(Student student){
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(NAME, student.getName());
        values.put(NUMBER, student.getNumber());
        values.put(EMAIL, student.getEmail());
        values.put(ADDRESS, student.getAddress());
        //Neu de null thi khi value bang null thi loi


        db.insert(TABLE_NAME,null,values);


        db.close();
```

```
    }
```

Phương thức trên dùng để thêm một Student vào trong sqlite, đầu tiên nó sẽ gọi phương thức getWritableDatabase() để mở database lên để đọc và ghi, nếu như ở đây database chưa được tạo thì phương onCreate() sẽ được gọi rồi sau đó mới thực hiện mở database lên.

Sau đó bạn tạo một đối tượng thuộc ContentValues để có thể put các giá trị vào trong các column, thằng này kiểu tương tự như Editor mà chúng ta đã học ở bài Shared Preferences ấy các bạn.

```
values.put(NAME, student.getName());
```

Ở trên mình đang đưa vào column "name" với giá trị là tên của student (student.getName()), Sau khi đã put hết giá trị rồi bạn gọi phương thức **insert()** và truyền vào các params là xong.Nhớ là nên đóng database lại mỗi lần mở nó ra nhé các bạn!

Read – Đọc

Sau khi đã lưu xuống được một danh sách student thì chúng ta tiến hành đọc lên thôi:

```
    public Student getSdtudentById(int id){
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.query(TABLE_NAME, new String[] { ID,
                NAME, EMAIL,NUMBER,ADDRESS }, ID + "=?",
                new String[] { String.valueOf(id) }, null, null, null, null);
//        Cursor cursor;
//            cursor = db.rawQuery("SELECT * FROM "+TABLE_NAME +" WHERE id = '1'",null);
        if (cursor != null)
            cursor.moveToFirst();
```

```
        Student student = new Student(cursor.getString(1),cursor.getString(2),cursor.getString(3),cursor.getString(4));
        cursor.close();
        db.close();
        return student;
    }
```

Ở đây chúng ta sẽ get thông tin một sinh viên ra theo id nhé các bạn, bạn sẽ thực hiện các câu query quen thuộc rồi lấy ra thôi.

Lấy toàn bộ danh sách Student thì làm như sau:

```
public List<Student> getAllStudent() {
    List<Student> listStudent = new ArrayList<Student>();
    // Select All Query
    String selectQuery = "SELECT  * FROM " + TABLE_NAME;

    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery(selectQuery, null);

    if (cursor.moveToFirst()) {
        do {
            Student student = new Student();
            student.setId(cursor.getInt(0));
```

```
            student.setName(cursor.getString(1));
            student.setEmail(cursor.getString(2));
            student.setNumber(cursor.getString(3));
            student.setAddress(cursor.getString(4));
            listStudent.add(student);
        } while (cursor.moveToNext());
    }
    cursor.close();
    db.close();
    return listStudent;
}
```

Update – Cập nhật

```
public int Update(Student student){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();

    values.put(NAME,student.getName());

    return db.update(TABLE_NAME,values,ID +"=?",new String[] {
String.valueOf(student.getId())});
```

```
}
```

Ở trên mình đang thay đổi tên của một Student với điệu kiện đang xét theo id, cái này nếu bạn không hiểu mình sẽ giải thích kĩ ở trong video nhé.

Delete – Xoá

```
public void deleteStudent(Student student) {

    SQLiteDatabase db = this.getWritableDatabase();

    db.delete(TABLE_NAME, ID + " = ?",

            new String[] { String.valueOf(student.getId()) });

    db.close();

}
```

Câu lệnh trên để xoá đi một Student khi theo điệu kiện là id luôn.

Lấy ra tổng số dòng trong Table

```
public int getStudentsCount() {
    String countQuery = "SELECT  * FROM " + TABLE_NAME;
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.rawQuery(countQuery, null);
    cursor.close();
    // return count
    return cursor.getCount();
}
```