

BỘ GIÁO DỤC VÀ ĐÀO TẠO  
ĐẠI HỌC CÔNG NGHỆ TP.HCM

LẬP TRÌNH  
TRÊN THIẾT BỊ DI ĐỘNG

**Biên Soạn:**

TS. Nguyễn Hà Giang

ThS. Bùi Mạnh Toàn

ThS. Võ Tân Dũng

## LẬP TRÌNH TRÊN THIẾT BỊ DI ĐỘNG



\* 1 . 2 0 2 1 . C M P 1 7 7 \*

---

Các ý kiến đóng góp về tài liệu học tập này, xin gửi về e-mail của ban biên tập:  
[tailieuhoctap@hutech.edu.vn](mailto:tailieuhoctap@hutech.edu.vn)

# MỤC LỤC

MỤC LỤC .....	I
HƯỚNG DẪN .....	V
BÀI 1: TỔNG QUAN VỀ LẬP TRÌNH TRÊN THIẾT BỊ DI ĐỘNG .....	1
1.1 GIỚI THIỆU TỔNG QUAN .....	1
1.1.1 Thiết bị di động .....	1
1.1.2 Hệ điều hành Android và iOS .....	2
1.1.3 Android là gì? .....	3
1.1.4 Các tính năng của Android .....	4
1.1.5 Lịch sử Android .....	5
1.1.6 Giá trị thị trường hệ điều hành Android .....	7
1.2 KIẾN TRÚC ANDROID.....	7
1.2.1 Applications.....	8
1.2.2 Application Framework.....	8
1.2.3 Android Runtime .....	9
1.2.4 Platform Libraries.....	9
1.2.5 Linux Kernel .....	10
TÓM TẮT .....	11
BÀI TẬP .....	11
BÀI 2: PHÁT TRIỂN ỨNG DỤNG TRÊN ANDROID .....	12
2.1 CÀI ĐẶT MÔI TRƯỜNG PHÁT TRIỂN ANDROID.....	12
2.1.1 Hai cách cài đặt .....	12
2.1.2 Cài đặt Android Studio .....	13
2.2 TẠO ỨNG DỤNG ANDROID.....	20
2.2.1 Ứng dụng Hello World .....	20
2.2.2 Android Layout File.....	27
2.2.3 Android Main Activity File .....	28
2.2.4 Android Manifest File .....	29
2.2.5 Run Android Hello World App .....	29
2.3 THIẾT LẬP ANDROID EMULATOR.....	32
2.3.1 Tạo Android Virtual Device .....	32
2.3.2 Chạy Ứng dụng Android .....	35
2.4 THÀNH PHẦN TRONG ỨNG DỤNG ANDROID .....	36
2.4.1 Các thành phần cơ bản .....	36
2.4.2 Android Activities .....	37
2.4.3 Android Intents.....	37
2.4.4 Android Services .....	37
2.4.5 Android Broadcast Receivers.....	38

2.4.6 Android Content Providers.....	38
2.4.7 Các Componet bổ sung .....	38
<b>TÓM TẮT .....</b>	<b>40</b>
<b>BÀI TẬP .....</b>	<b>40</b>
<b>BÀI 3: XÂY DỰNG ACTIVITY.....</b>	<b>41</b>
<b>3.1 CÁC ACTIVITY TRONG ỨNG DỤNG ANDROID .....</b>	<b>41</b>
3.1.1 Activity là gì?.....	41
3.1.2 Chu kỳ sống của Android Activity .....	42
3.1.3 Các phương thức Callback .....	43
3.1.4 Sơ đồ chu kỳ sống của Android Activity.....	47
3.1.5 Ví dụ về Android Activity Lifecycle .....	49
<b>3.2 XỬ LÝ SỰ KIỆN TRONG ANDROID .....</b>	<b>54</b>
3.2.1 Sự kiện là gì?.....	54
3.2.2 Event Listeners và Event Handlers.....	55
3.2.3 Đăng ký Event Listeners.....	56
3.2.4 Chế độ cảm ứng và sự Focus .....	56
3.2.5 Ví dụ về xử lý sự kiện .....	57
<b>TÓM TẮT .....</b>	<b>63</b>
<b>BÀI TẬP .....</b>	<b>63</b>
<b>BÀI 4: THIẾT KẾ GIAO DIỆN.....</b>	<b>65</b>
<b>4.1 GIỚI THIỆU LAYOUT VÀ VIEW CỦA ANDROID .....</b>	<b>65</b>
4.1.1 Android View and ViewGroup.....	65
4.1.2 Các View của Android .....	65
4.1.3 Android ViewGroup.....	66
4.1.4 Các loại layout trong Android .....	67
4.1.5 Khai báo các UI Element trong XML.....	67
4.1.6 Nạp tập tin XML Layout từ một Activity .....	68
4.1.7 Khởi tạo các Layout Element trong thời gian chạy .....	69
4.1.8 Thuộc tính Width và Height .....	69
4.1.9 Các thuộc tính của Layout .....	70
<b>4.2 CÁC LOẠI LAYOUT.....</b>	<b>71</b>
4.2.1 LinearLayout.....	72
4.2.2 RelativeLayout.....	76
4.2.3 TableLayout.....	80
4.2.4 Frame Layout .....	84
<b>4.3 CÁC VIEWGROUP .....</b>	<b>87</b>
4.3.1 ListView .....	87
4.3.2 GridView .....	96
4.3.3 WebView.....	104
4.3.4 ScrollView .....	109
<b>TÓM TẮT .....</b>	<b>112</b>

<b>BÀI TẬP .....</b>	<b>112</b>
<b>BÀI 5: THÀNH PHẦN VIEW - WIDGET.....</b>	<b>116</b>
<b>5.1 VIEW CƠ BẢN .....</b>	<b>116</b>
<b>5.2 VIEW NÂNG CAO .....</b>	<b>128</b>
<b>5.3 TOAST VÀ DIALOG.....</b>	<b>144</b>
<b>TÓM TẮT .....</b>	<b>149</b>
<b>BÀI TẬP .....</b>	<b>149</b>
<b>BÀI 6: THÀNH PHẦN INTENT .....</b>	<b>153</b>
<b>6.1 CƠ CHẾ INTENT .....</b>	<b>153</b>
<b>6.2 ĐỔI TƯỢNG INTENT .....</b>	<b>154</b>
<b>6.3 PHÂN LOẠI INTENT .....</b>	<b>156</b>
6.3.1 Intent Filter.....	157
6.3.2 Cách thức xác định thành phần phù hợp với Intent.....	158
<b>6.4 SỬ DỤNG CÁC INTENT .....</b>	<b>161</b>
6.4.1 Explicit Intent thực thi Activity .....	161
6.4.2 Implicit Intent thực thi Activity .....	162
<b>6.5 GỌI HIỂN THỊ ACTIVITY.....</b>	<b>162</b>
6.5.1 Gọi hiển thị không có trao đổi dữ liệu.....	163
6.5.2 Gọi hiển thị và truyền dữ liệu cho Activity con.....	164
6.5.3 Gọi hiển thị và truyền/nhận dữ liệu với Activity con .....	165
<b>TÓM TẮT .....</b>	<b>167</b>
<b>BÀI TẬP .....</b>	<b>167</b>
<b>BÀI 7: LẬP TRÌNH VỚI SQLITE .....</b>	<b>169</b>
<b>7.1 CƠ SỞ DỮ LIỆU SQLITE.....</b>	<b>169</b>
7.1.1 Đặc trưng cơ bản của SQLite.....	170
7.1.2 Lập trình với SQLite .....	170
7.1.3 Kiểu dữ liệu lưu trữ.....	171
7.1.4 Các lớp cơ bản trong gói SQLite .....	172
7.1.5 Thư mục lưu trữ .....	173
<b>7.2 ỨNG DỤNG MINH HỌA .....</b>	<b>174</b>
7.2.1 Tạo lớp kế thừa từ SQLiteOpenHelper .....	174
7.2.2 Tạo lớp chứa dữ liệu DTO là Student .....	175
7.2.3 Tạo lớp truy cập dữ liệu .....	176
7.2.4 Thao tác insert dữ liệu .....	179
7.2.5 Hiển thị danh sách sinh viên .....	180
<b>TÓM TẮT .....</b>	<b>181</b>
<b>BÀI TẬP .....</b>	<b>181</b>
<b>BÀI 8: ỨNG DỤNG SERVICE .....</b>	<b>183</b>
<b>8.1 THÀNH PHẦN SERVICE .....</b>	<b>183</b>
8.1.1 Vòng đời của Service .....	184

8.1.2 Khuôn mẫu chung của lớp thực thi Service .....	186
8.1.3 Khởi tạo Service.....	187
8.1.4 Đăng ký Service vào tệp Manifest.....	188
8.1.5 Chạy một Service.....	188
<b>8.2 ỨNG DỤNG MẪU SERVICE: CHƠI NHẠC .....</b>	<b>189</b>
<b>TÓM TẮT .....</b>	<b>193</b>
<b>BÀI TẬP .....</b>	<b>194</b>
<b>BÀI 9: BROADCAST RECEIVER.....</b>	<b>195</b>
<b>9.1 CƠ CHẾ BROADCAST RECEIVER .....</b>	<b>195</b>
9.1.1 Phân loại Broadcast .....	196
9.1.2 Tạo một broadcast receiver .....	196
9.1.3 Đăng ký xử lý Broadcast Intent.....	196
9.1.4 Một số các Broadcast Intent .....	197
9.1.5 Phát sinh Broadcast Intent .....	198
<b>9.2 MINH HỌA BROADCASTRECEIVER .....</b>	<b>199</b>
<b>TÓM TẮT .....</b>	<b>203</b>
<b>BÀI TẬP .....</b>	<b>203</b>
<b>BÀI 10: THÀNH PHẦN NOTIFICATION .....</b>	<b>205</b>
<b>10.1 THÀNH PHẦN NOTIFICATION .....</b>	<b>205</b>
<b>10.2 THỂ HIỆN THÔNG ĐIỆP .....</b>	<b>206</b>
10.2.1 Dạng bình thường.....	206
10.2.2 Dạng lớn .....	207
<b>10.3 TẠO THÔNG ĐIỆP .....</b>	<b>208</b>
10.3.1 Các nội dung thông điệp bắt buộc.....	208
10.3.2 Hành động của thông điệp .....	209
10.3.3 Tạo thông điệp đơn giản .....	209
<b>10.4 SỬ DỤNG THANH TIỀN ĐỘ .....</b>	<b>210</b>
10.4.1 Thanh tiến độ với thời gian xác định .....	210
10.4.2 Thanh chỉ dẫn liên tục .....	211
<b>TÓM TẮT .....</b>	<b>213</b>
<b>BÀI TẬP .....</b>	<b>213</b>
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>214</b>

# HƯỚNG DẪN

## MÔ TẢ MÔN HỌC

Môn học trang bị các kiến thức về cách thức phát triển ứng dụng trên nền tảng thiết bị di động. Bao gồm các nhóm kiến thức: giới thiệu các nền tảng hay hệ điều hành trên thiết bị di động thông minh, các công cụ và môi trường để phát triển ứng dụng trên thiết bị di động thông minh.

## NỘI DUNG MÔN HỌC

- Bài 1. Tổng quan về lập trình trên thiết bị di động: kiến thức cơ bản về thiết bị di động và khái niệm cơ bản về môi trường và cách thức lập trình trên thiết bị di động.
- Bài 2. Phát triển ứng dụng trên Android: cung cấp các kiến thức cơ bản về hệ điều hành Android. Trình bày mô hình quy trình phát triển ứng dụng trên thiết bị di động hỗ trợ Android. Ngoài ra giới thiệu các thành phần chính của một ứng dụng Android điển hình.
- Bài 3. Xây dựng Activity trong ứng dụng Android: bài học cung cấp kiến thức để xây dựng ứng dụng có giao diện tương tác với người dùng; cách thức hoạt động của thành phần Activity trong ứng dụng Android. Làm quen với cách thức sử dụng layout XML để khởi tạo giao diện của Activity.
- Bài 4. Thiết kế giao diện: trong bài này trình bày các loại layout cơ bản và nâng cao để tạo giao diện cho Activity.
- Bài 5. Thành phần View-Widget: cung cấp các loại widget từ cơ bản đến nâng cao, bao gồm chức năng, cách sử dụng trong thiết kế giao diện và cách tham chiếu để lập trình trong mã nguồn Java.
- Bài 6. Thành phần Intent: bài này cung cấp kỹ thuật để truyền thông và giao tiếp giữa các thành phần trong ứng dụng Android. Cách thức gọi hiển thị Activity trong cùng một ứng dụng và gọi các ứng dụng hệ thống cũng được trình bày chi tiết trong bài học.

- Bài 7. Lập trình với SQLite: kỹ thuật để tạo và sử dụng cơ sở dữ liệu SQLite trong ứng dụng Android.
- Bài 8. Xây dựng ứng dụng Service: Cung cấp mô hình tổng quát để tạo ứng dụng Android chạy background, là dạng ứng dụng khi hoạt động không có giao diện tương tác với người dùng.
- Bài 9. Lập trình Broadcast Receiver: Cung cấp kỹ thuật đăng ký xử lý các thông điệp của hệ thống hay của ứng dụng.
- Bài 10. Thành phần Notification: Cách thức sử dụng Notification trong ứng dụng Android.

## KIẾN THỨC TIỀN ĐỀ

Sinh viên có kiến thức về lập trình hướng đối tượng, lập trình có giao diện tương tác với người dùng. Ngoài ra có kiến thức về cơ sở dữ liệu quan hệ và lập trình cơ sở dữ liệu.

## YÊU CẦU MÔN HỌC

Người học phải tham dự đầy đủ các buổi lên lớp và làm bài tập đầy đủ ở nhà. Ngoài ra trước mỗi buổi học, người học phải đọc trước bài học trong bài giảng để nắm các nội dung chính. Sau mỗi giờ lên lớp, người học cần đọc thêm tài liệu do giảng viên trực tiếp đứng lớp yêu cầu. Do giới hạn của bài giảng không thể cung cấp hết và chi tiết các thành phần trong lập trình Android, vì vậy yêu cầu người học cần phải đọc thêm các tài liệu tham khảo liệt kê trong phần cuối của bài giảng.

## CÁCH TIẾP NHẬN NỘI DUNG MÔN HỌC

Để học tốt môn này, người học cần ôn tập các bài đã học, trả lời các câu hỏi và làm nhiều bài tập; đọc trước bài mới và tự tìm kiếm các thông tin liên quan đến bài học.

Đối với mỗi bài học, người học đọc trước mục tiêu và tóm tắt bài học, sau đó đọc nội dung bài học. Kết thúc mỗi bài học người đọc cần phải làm tối thiểu các bài tập yêu cầu.

Ngoài ra đây là môn học trong đó có một số bài sẽ có phần thực hành là viết chương trình, do đó người học cần phải làm những bài tập phần thực hành tương ứng với những bài học.

## **PHƯƠNG PHÁP ĐÁNH GIÁ MÔN HỌC**

Môn học được đánh giá gồm:

- Điểm quá trình (50%): Hình thức và cách đánh giá do giảng viên dạy lý thuyết quyết định được phê duyệt của bộ môn.
- Điểm thi cuối kỳ (50%): Hình thức làm đồ án môn học và chấm thi văn답 dựa trên nội dung đồ án môn học kết hợp với lý thuyết trong các bài học. Danh sách đồ án do giảng viên quyết định được bộ môn kiểm duyệt và cung cấp vào đầu khóa học.



# BÀI 1: TỔNG QUAN VỀ LẬP TRÌNH TRÊN THIẾT BỊ DI ĐỘNG

Học xong bài này người học cần nắm được các nội dung sau.

- *Tổng quan về các nền tảng trên thiết bị di động thông minh.*
- *Các khái niệm cơ bản về phát triển các dạng ứng dụng trên thiết bị di động.*

## **1.1 GIỚI THIỆU TỔNG QUAN**

### **1.1.1 Thiết bị di động**

Điện thoại di động thông minh (smart phone) thực sự đã mang đến một cuộc cách mạng cho các thiết bị di động, trong thời kì mà công nghệ số phát triển với tốc độ chóng mặt như hiện nay.

Nhu cầu sử dụng thiết bị di động đã trở nên rất phổ biến không chỉ ở Việt Nam mà trên toàn thế giới. Sự tiến bộ vượt bậc của công nghệ đã làm thay đổi hoàn toàn thói quen cũng như hành vi của con người trong cuộc sống. Chỉ với một chiếc smart phone người ta có thể thỏa mãn hầu hết các nhu cầu cơ bản như: trò chuyện, gửi tin nhắn, chơi game, nghe nhạc, lướt web, thanh toán... Thiết bị di động là phân khúc đã có bước phát triển nhanh chóng trên thị trường. Theo nhiều cuộc khảo sát, hiện nay số lượng truy cập Internet từ thiết bị di động thông minh đã vượt qua máy tính cá nhân. Trong tương lai, thiết bị di động sẽ trở thành phương tiện giao tiếp và làm việc chủ yếu của con người. Và phần cốt lõi để tạo ra sức hấp dẫn từ những chiếc smart phone chính là hệ điều hành mạnh mẽ và đa dạng các ứng dụng phục vụ cho thiết bị.

Do sự phát triển vượt bậc của thiết bị di động nên xu hướng hiện nay các công ty phát triển phần mềm đã chuyển hướng sang phát triển phần mềm trên thiết bị di động. Vì thế trong những năm gần đây nhu cầu nhân lực phát triển ứng dụng trên thiết bị di động hay smart phone đang rất phát triển.



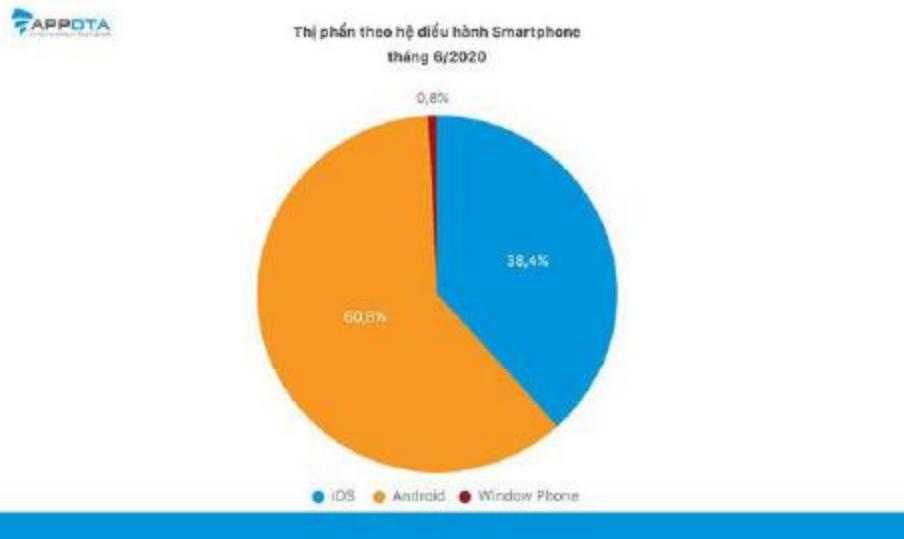
### 1.1.2 Hệ điều hành Android và iOS

Android là hệ điều hành có mã nguồn mở dựa trên nền tảng Linux do chính Google phát hành. Do có mã nguồn mở cùng giấy phép của Google không có quá nhiều ràng buộc đã cho phép các nhà phát triển, lập trình viên điều chỉnh và phân phối Android một cách tự do nhất.

Theo số liệu năm 2020, Android vẫn giữ vững vị trí hệ điều hành lớn nhất thế giới với 74,4%. Còn tại thị trường di động Việt Nam, báo cáo của Appota chỉ ra rằng, Android đang chiếm ưu thế hơn với hơn 60% thị phần.

Đứng thứ hai là hệ điều hành iOS với 38,4% ở mảng thị trường di động. Do iOS là hệ điều hành được phát triển bởi Apple và phân phối độc quyền cho phần cứng của Apple, iOS cũng không hỗ trợ các ứng dụng từ các nhà phát triển thứ 3 nên việc thị trường bị giới hạn cũng là điều dễ hiểu.

Windows Phone là hệ điều hành của Microsoft dành cho smartphone, kế tục nền tảng Windows Mobile, mặc dù chúng không giống nhau. Khác với Windows Mobile, Windows Phone tập trung vào sự phát triển của Marketplace - nơi các nhà phát triển có thể cung cấp sản phẩm tới người dùng. Windows Phone được ra mắt vào tháng 10 năm 2010. Sự chậm chạp trong việc đổi mới, kho ứng dụng nghèo nàn, kiểm soát phần cứng tệ là các lý do khiến Microsoft không thể thống trị thị trường smartphone và ngày càng biến mất khỏi thị trường.



Chúng ta sẽ tìm hiểu những điều cơ bản về hệ điều hành Android như: Android là gì, lịch sử của hệ điều hành android, tính năng của hệ điều hành Android, các phiên bản khác nhau của hệ điều hành Android với các ví dụ minh họa ở các phần tiếp theo.

### **1.1.3 Android là gì?**

Android là một hệ điều hành mã nguồn mở dựa trên Linux với giao diện lập trình Java cho các thiết bị di động như điện thoại thông minh (thiết bị màn hình cảm ứng hỗ trợ hệ điều hành Android) cũng như cho máy tính bảng.

Android được phát triển bởi Open Handset Alliance (OHA), do Google dẫn đầu. Open Handset Alliance (OHA) là một liên minh gồm nhiều công ty như Samsung, Sony, Intel và nhiều công ty khác để cung cấp dịch vụ và triển khai thiết bị cầm tay sử dụng nền tảng Android.

Vào năm 2007, Google đã phát hành phiên bản beta đầu tiên của bộ phát triển phần mềm Android (SDK) và phiên bản thương mại đầu tiên của Android 1.0 (với tên Alpha), được phát hành vào tháng 9 năm 2008.

Vào năm 2012, Google đã phát hành một phiên bản Android khác, 4.1 Jelly Bean. Đây là một bản cập nhật gia tăng và nó đã cải thiện rất nhiều về giao diện người dùng, chức năng và hiệu suất.

Vào năm 2014, Google đã công bố một phiên bản mới nhất khác, 5.0 Lollipop. Trong phiên bản Lollipop, Google đã cải tiến hoàn toàn giao diện người dùng bằng

cách sử dụng Material Designs, điều này tốt cho Giao diện người dùng cũng như cho các chủ đề liên quan.

Tất cả mã nguồn dành cho Android đều có sẵn miễn phí trên Git-Hub, Stack Overflow và nhiều trang web khác. Google xuất bản hầu hết mã theo Apache License version 2.0 (Giấy phép Apache phiên bản 2.0).

### **1.1.4 Các tính năng của Android**

Android là một hệ điều hành mã nguồn mở mạnh mẽ cung cấp rất nhiều tính năng tuyệt vời, đó là:

- Nó là mã nguồn mở và chúng ta có thể tùy chỉnh hệ điều hành dựa trên các yêu cầu của mình.
- Nó hỗ trợ kết nối GSM, CDMA, WIFI, NFC, Bluetooth, v...v... để thực hiện cuộc gọi điện thoại hoặc truyền dữ liệu. Nó sẽ cho phép chúng ta thực hiện hoặc nhận cuộc gọi hoặc tin nhắn SMS và chúng ta có thể gửi hoặc truy xuất dữ liệu trên các mạng di động
- Bằng cách sử dụng công nghệ WIFI, chúng ta có thể ghép cặp (pair) với các thiết bị khác bằng các ứng dụng.
- Android có nhiều API để hỗ trợ các dịch vụ dựa trên vị trí như GPS.
- Chúng tôi có thể thực hiện tất cả các hoạt động liên quan đến lưu trữ dữ liệu bằng cách sử dụng cơ sở dữ liệu nhẹ SQLite.
- Nó có một loạt các hỗ trợ đa phương tiện như AVI, MKV, FLV, MPEG4, v.v. để phát hoặc ghi nhiều loại âm thanh hoặc video và có định dạng hình ảnh khác nhau như JPEG, PNG, GIF, BMP, MP3, v...v...
- Nó có hỗ trợ rộng rãi cho điều khiển phần cứng đa phương tiện để thực hiện phát lại hoặc ghi âm bằng camera và micrô.
- Nó có một trình duyệt web dựa trên bộ cục WebKit mã nguồn mở tích hợp để hỗ trợ HTML5, CSS3.
- Nó hỗ trợ đa tác vụ, chúng ta có thể di chuyển từ cửa sổ tác vụ này sang cửa sổ tác vụ khác và nhiều ứng dụng có thể chạy đồng thời.

- Nó sẽ tạo cơ hội sử dụng lại các thành phần ứng dụng và thay thế các ứng dụng gốc.
- Chúng ta có thể truy cập các thành phần phần cứng như máy ảnh, GPS và Accelerometer.
- Nó có hỗ trợ cho đồ họa 2D/3D

### 1.1.5 Lịch sử Android

Ban đầu, Google tung ra phiên bản đầu tiên của nền tảng Android vào ngày 5 tháng 11 năm 2007, từ đó trở đi Google đã phát hành rất nhiều phiên bản Android với Code Name (tên mã) dựa trên các món tráng miệng, chẳng hạn như Apple Pie, Banana Bread, Cupcake, Donut, Éclair, Froyo, Gingerbread, Jellybeans, Kitkat, Lollipop, Marshmallow,...v...v... và đã thực hiện rất nhiều thay đổi và bổ sung cho nền tảng Android.

Bảng sau liệt kê chi tiết các phiên bản khác nhau của Android được Google phát hành từ năm 2007 đến nay.

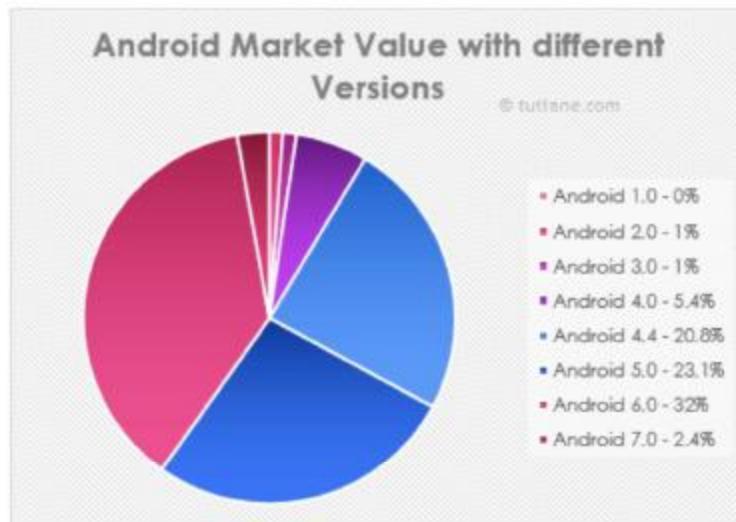
Release Date	Version	API Level	Name Code
September 23, 2008	Android 1.0	1	Apple Pie
February 9, 2009	Android 1.1	2	Banana Bread
April 30, 2009	Android 1.5	3	Cupcake
September 15, 2009	Android 1.6	4	Donut
October 26, 2009	Android 2.0	5	Eclair
December 3, 2009	Android 2.0.1	6	
January 12, 2009	Android 2.1	7	
May 20, 2010	Android 2.2	8	Froyo
January 18, 2011	Android 2.2.1	8	
January 22, 2011	Android 2.2.2	8	
November 21, 2011	Android 2.2.3	8	
December 6, 2010	Android 2.3	9	Gingerbread
February 9, 2011	Android 2.3.1	9	
July 25, 2011	Android 2.3.3	10	
September 2, 2011	Android 2.3.4	10	
February 22, 2011	Android 3.0.x	11	Honeycomb

Release Date	Version	API Level	Name Code
May 10, 2011	Android 3.1.x	12	
July 15, 2011	Android 3.2.x	13	
October 18, 2011	Android 4.0	14	Ice Cream Sandwich
October 19, 2011	Android 4.0.1	14	
November 28, 2011	Android 4.0.2	14	
December 16, 2011	Android 4.0.3	15	
February 4, 2012	Android 4.0.4	15	
July 9, 2012	Android 4.1	16	Jelly Bean
July 23, 2012	Android 4.1.1	16	
October 9, 2012	Android 4.1.2	16	
November 13, 2012	Android 4.2	17	
November 27, 2012	Android 4.2.1	17	
February 11, 2013	Android 4.2.2	17	
July 24, 2013	Android 4.3	18	
October 31, 2013	Android 4.4	19	Kitkat
June 23, 2014	Android 4.4.1, 4.4.2, 4.4.3, 4.4.4	19	
October 17, 2014	Android 5.0	21	Lollipop
March 09, 2015	Android 5.1	22	
October 5, 2015	Android 6.0	23	Marshmallow
December 7, 2015	Android 6.0.1	23	
August 22, 2016	Android 7.0	24	Nougat
October 4, 2016	Android 7.1	25	
August 21, 2017	Android 8.0	26	Oreo
December 5, 2017	Android 8.1	27	
August 6, 2018	Android 9.0	28	Pie
September 3, 2019	Android 10.0	29	Android 10
September 8, 2020	Android 11.0	30	Android 11

Đối với mỗi phiên bản, Google đã thực hiện rất nhiều thay đổi và giới thiệu rất nhiều tính năng mới bởi vì việc sử dụng Android trên thị trường di động tăng mạnh.

### 1.1.6 Giá trị thị trường hệ điều hành Android

Sau đây là mô tả bằng hình ảnh về việc sử dụng Android trong thị trường điện thoại di động với các phiên bản khác nhau.



Đây là cách Google phát hành nhiều phiên bản hệ điều hành android và giành được thị phần điện thoại di động không lồ với nhiều phiên bản khác nhau.

## 1.2 KIẾN TRÚC ANDROID

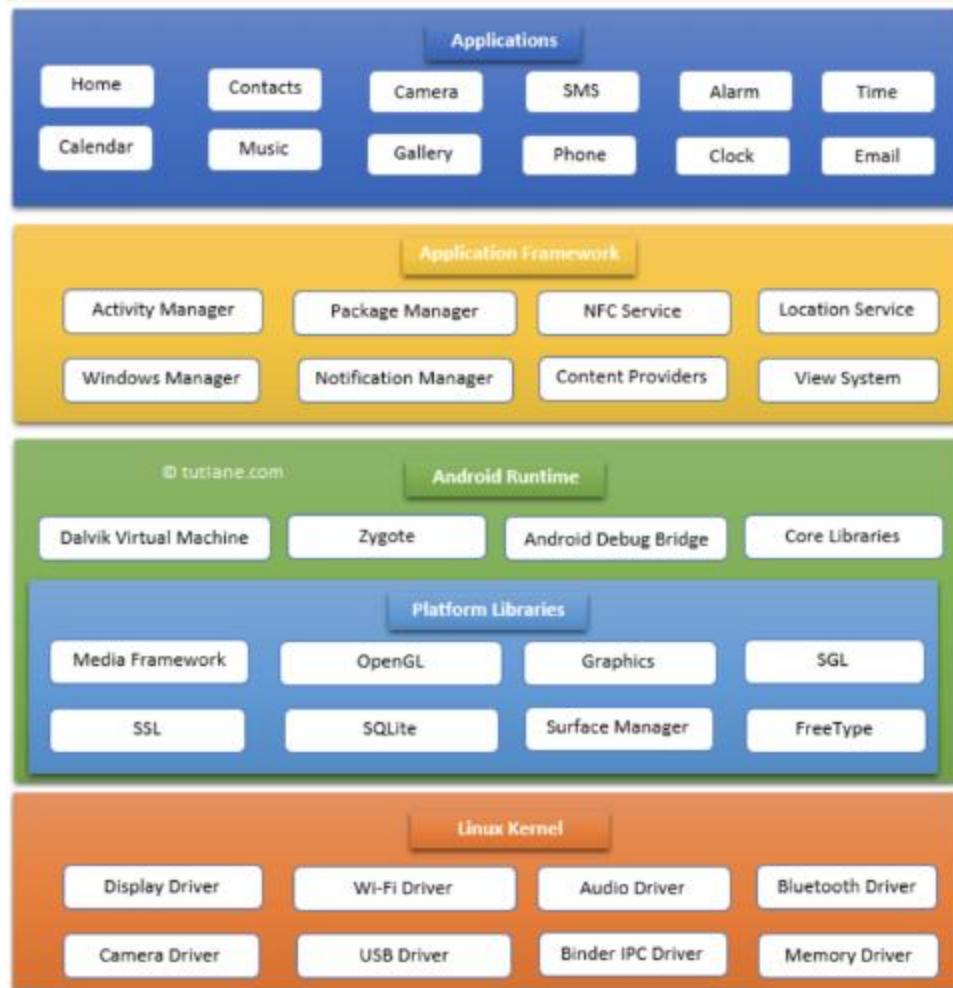
Kiến trúc Android là một tập hợp các thành phần phần mềm để hỗ trợ nhu cầu của thiết bị di động. Một chồng (stack) các phần mềm Android chứa Linux Kernel (nhân), tập hợp các thư viện C/C ++ được thể hiện thông qua application framework, các services, runtime và các ứng dụng.

Sau đây là các thành phần chính của kiến trúc Android, đó là

- Applications
- Android Framework
- Android Runtime
- Platform Libraries
- Linux Kernel

Trong các thành phần này, Linux Kernel là thành phần chính trong android để cung cấp các chức năng của hệ điều hành cho thiết bị di động và Máy ảo Dalvik (DVM-Dalvik Virtual Machine) chịu trách nhiệm chạy một ứng dụng di động.

Sau đây là mô tả bằng hình ảnh của kiến trúc android với các thành phần khác nhau.



## 1.2.1 Applications

Lớp trên cùng của kiến trúc android là Application (ứng dụng). Các ứng dụng nguyên thủy và ứng dụng của bên thứ ba như danh bạ, email, nhạc, thư viện, đồng hồ, trò chơi, v...v... bất cứ thứ gì ta sẽ xây dựng thì chúng sẽ chỉ được cài đặt tại lớp này.

Lớp ứng dụng chạy run-time trong Android bằng cách sử dụng các lớp (classes) và các dịch vụ (services) được tạo sẵn từ khung ứng dụng (application framework).

## 1.2.2 Application Framework

Application Framework (khung ứng dụng) cung cấp các lớp được sử dụng để tạo ứng dụng Android. Nó cũng cung cấp thành phần trừu tượng khái quát (generic

abstraction) để truy cập phần cứng và quản lý giao diện người dùng và các tài nguyên ứng dụng. Về cơ bản, nó cung cấp các dịch vụ mà qua đó chúng ta có thể tạo một lớp cụ thể nào đó và làm cho lớp đó hữu ích để hỗ trợ cho việc tạo ứng dụng.

Khung ứng dụng bao gồm các dịch vụ như dịch vụ điện thoại (telephony service), dịch vụ định vị (location services), trình quản lý thông báo (notification manager), dịch vụ NFC, hệ thống các khung nhìn (view), v...v... mà chúng ta có thể sử dụng để phát triển ứng dụng theo các yêu cầu của mình.

### 1.2.3 Android Runtime

Môi trường Android Runtime là một phần quan trọng của Android hơn là một phần bên trong và nó chứa các thành phần như các thư viện lõi và máy ảo Dalvik (Dalvik virtual machine). Android Runtime là engine cung cấp sức mạnh cho các ứng dụng của chúng ta cùng với các thư viện và nó tạo cơ sở cho khung ứng dụng (application framework).

Dalvik Virtual Machine (DVM) là một máy ảo tương tự như Java Virtual Machine (JVM). Nó được thiết kế đặc biệt và tối ưu hóa cho Android để đảm bảo rằng một thiết bị có thể chạy nhiều thực thể ứng dụng một cách hiệu quả. Nó dựa vào nhân Linux để phân luồng và quản lý bộ nhớ cấp thấp.

Các thư viện lõi trong Android Runtime sẽ cho phép chúng ta triển khai các ứng dụng Android bằng ngôn ngữ lập trình JAVA chuẩn.

### 1.2.4 Platform Libraries

Platform Libraries bao gồm các thư viện lõi C / C ++ khác nhau và các thư viện loại Java-based như SSL, libc, Graphics, SQLite, Webkit, Media, Surface Manger, OpenGL,v...v... để hỗ trợ phát triển Android.

Sau đây là chi tiết tóm tắt về một số thư viện Android lõi có sẵn để phát triển Android.

- Media library để phát và ghi các định dạng âm thanh và video. Surface manager library để cung cấp quản lý hiển thị.
- Thư viện SGL và OpenGL Graphics dành cho đồ họa 2D và 3D.

- SQLite để hỗ trợ cơ sở dữ liệu và FreeType để hỗ trợ phông chữ
- Web-Kit để hỗ trợ trình duyệt web và SSL để bảo mật Internet.

## 1.2.5 Linux Kernel

Linux Kernel là tầng dưới cùng và là trái tim của kiến trúc Android. Nó quản lý tất cả các trình điều khiển như trình điều khiển hiển thị (display drivers), trình điều khiển máy ảnh (camera drivers), trình điều khiển Bluetooth (Bluetooth drivers), trình điều khiển âm thanh (audio drivers), trình điều khiển bộ nhớ (memory drivers), v...v... là những yêu cầu chính yếu cho thiết bị Android chạy trong suốt thời gian run time.

Linux Kernel sẽ cung cấp một tầng trung tương giữa phần cứng thiết bị và phần còn lại của ngăn xếp. Nó chịu trách nhiệm quản lý bộ nhớ, quản lý nguồn, quản lý thiết bị, truy cập tài nguyên, v...v...

# TÓM TẮT

Bài học giới thiệu tổng quan về thiết bị di động cá nhân, điển hình là điện thoại di động và ngày nay thường được trang bị cấu hình mạnh nên được gọi là điện thoại thông minh. Do thiết bị này hiện tại khá phổ biến nên việc phát triển ứng dụng trên thiết bị này được yêu cầu khá nhiều. Bài học giới thiệu các hệ điều hành dành cho thiết bị di động phổ biến hiện nay là Android, iOS (và cả Windows Phone).

## BÀI TẬP

**Câu 1:** Hãy cho biết các nền tảng cho thiết bị di động thông minh hiện nay? Với mỗi nền tảng hãy cho biết đặc điểm, ưu và khuyết điểm.

**Câu 2:** Hãy cho biết tại sao giá trị phần của Android cao hơn iOS?

**Câu 3:** Hãy cho biết các đặc điểm cơ bản của nền tảng Android?

**Câu 4:** Cho biết các ngôn ngữ chính để phát triển ứng dụng trên nền tảng Android?

**Câu 5:** Cho biết các ngôn ngữ chính để phát triển ứng dụng trên nền tảng iOS?

**Câu 6:** Hãy cho biết tại sao thị phần của Windows Phone ngày càng thu hẹp?

**Câu 7:** Hãy tìm hiểu các ngôn ngữ để phát triển ứng dụng Web trên nền tảng thiết bị di động?

**Câu 8:** Hãy tìm hiểu nhu cầu nguồn nhân lực lập trình viên lập trình trên thiết bị di động?

# BÀI 2: PHÁT TRIỂN ỨNG DỤNG TRÊN ANDROID

Học xong bài này người học cần nắm được các nội dung sau.

- *Cài đặt được phần mềm Android Studio để lập trình trên Android.*
- *Tạo được ứng dụng cơ bản bằng Android Studio.*
- *Hiểu được các thành phần chính của một ứng dụng Android*
- *Hiểu được quy trình phát triển của ứng dụng Android.*
- *Tạo được thiết bị điện thoại thông minh giả lập để chạy ứng dụng.*

## **2.1 CÀI ĐẶT MÔI TRƯỜNG PHÁT TRIỂN ANDROID**

### **2.1.1 Hai cách cài đặt**

Nói chung, để xây dựng các ứng dụng cho Android, chúng ta nên có Bộ phát triển Java JDK (Java Development), Android SDK và môi trường phát triển.

Android SDK tương thích với các hệ điều hành Windows, Mac và Linux để tạo các ứng dụng Android theo yêu cầu của chúng ta.

Chúng ta có thể thiết lập môi trường phát triển Android bằng hai cách sau.

1. Cài đặt Eclipse IDE theo cách thủ công. Tóm lược ngay sau đây.
2. Android Studio.

Ban đầu, Google hỗ trợ cài đặt Eclipse IDE thủ công cho môi trường phát triển Android bằng cách tải xuống các thành phần bắt buộc như Eclipse IDE, Android SDK, Java Development Kit (JDK), v...v... từ trang web chính thức. Sau đó, Google đã giới

thiệu một thành phần có tên là Android Studio để làm cho quá trình thiết lập môi trường trở nên đơn giản hơn.

Bằng cách sử dụng gói Android Studio, chúng ta có thể dễ dàng cài đặt môi trường phát triển Android trong bất kỳ hệ điều hành nào để triển khai các ứng dụng Android.

Android Studio là sự kết hợp của các thành phần sau để cho phép người dùng triển khai các ứng dụng Android.

1. IDE Eclipse
2. SDK Android
3. Thiết bị ảo Android (Android Virtual Device)
4. Eclipse Plugin

Bằng cách tải xuống Android Studio trực tiếp từ trang web của Google để cài đặt, chúng ta có thể dễ dàng thiết lập môi trường phát triển các ứng dụng của mình.

### **2.1.2 Cài đặt Android Studio**

Android Studio là IDE chính thức để phát triển Android và nó dựa trên phần mềm IntelliJ IDEA. Nó có sẵn cho các hệ điều hành Windows, MAC và LINUX. Chúng ta có thể tải xuống phiên bản Android Studio mới nhất từ URL sau:

<https://developer.android.com/studio/index.html>

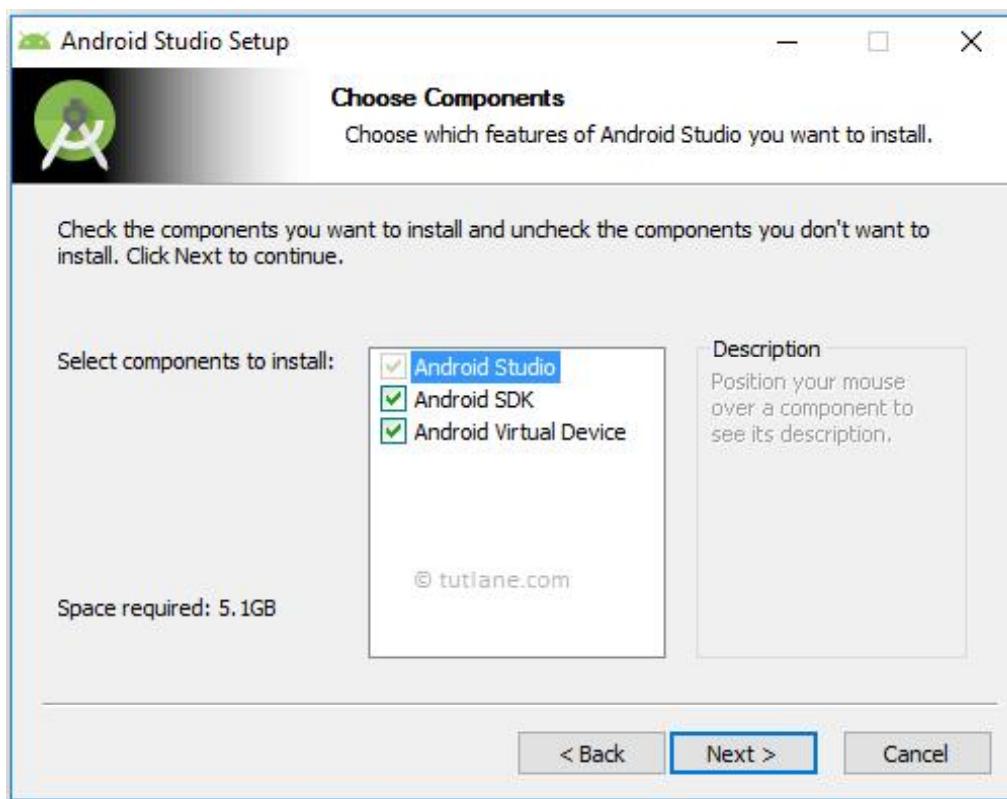
Trong hướng dẫn này, chúng ta sẽ giải thích cách cài đặt Android Studio trên máy tính Windows đang có hệ điều hành Windows 10.

Tải xuống phiên bản Android Studio mới nhất từ URL ở trên và khởi chạy tệp Android Studio.exe bằng cách nhấp đúp vào tệp đó.

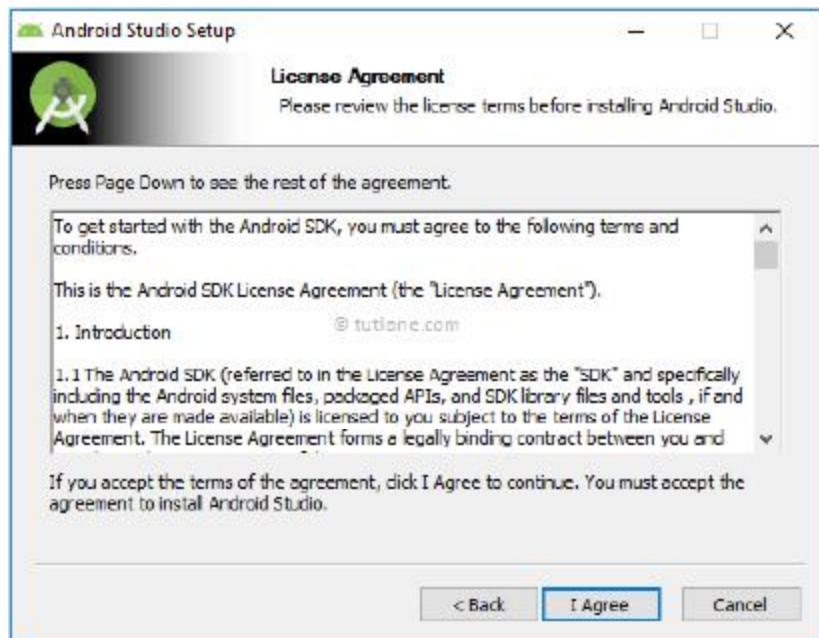
Màn hình thiết lập Android Studio ban đầu sẽ mở ra như hình dưới đây, nhấp vào Next để tiếp tục các bước thiết lập môi trường tiếp theo.



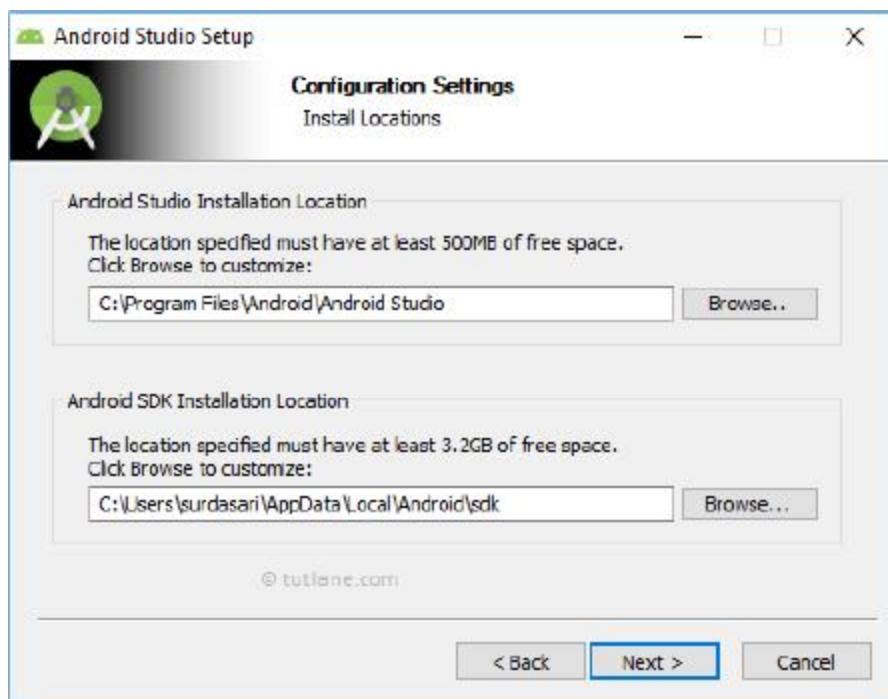
Bây giờ chúng ta cần chọn các thành phần cần thiết. Ở đây chúng ta đã chọn cả ba thành phần (Android Studio, Android SDK và Android Virtual Device) và nhấn Next như hình bên dưới.



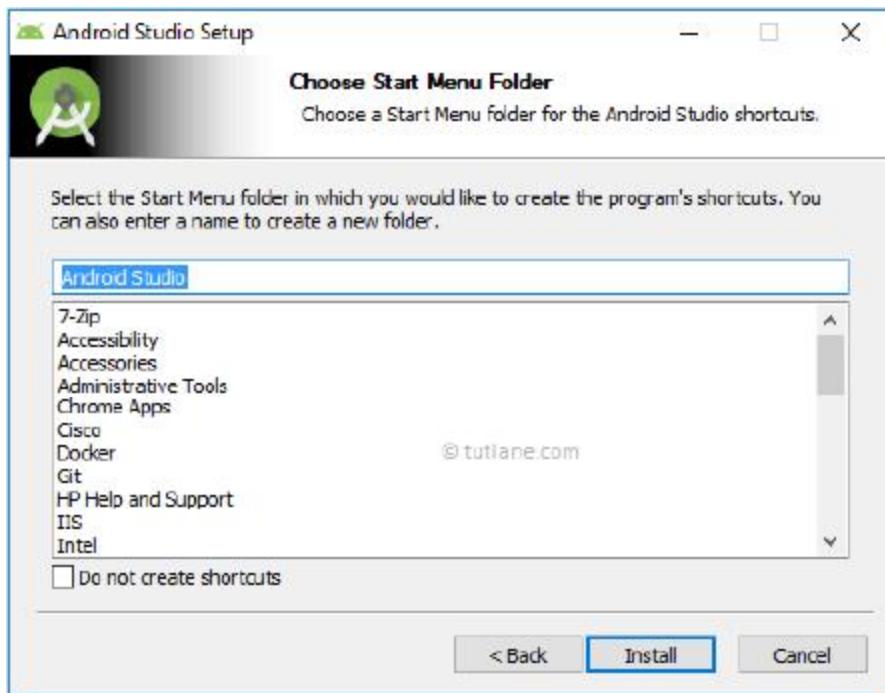
Bây giờ chúng ta cần đồng ý về các thỏa thuận Giấy phép để tiếp tục, hãy nhấp vào nút I Agree (tôi đồng ý) như hình bên dưới.



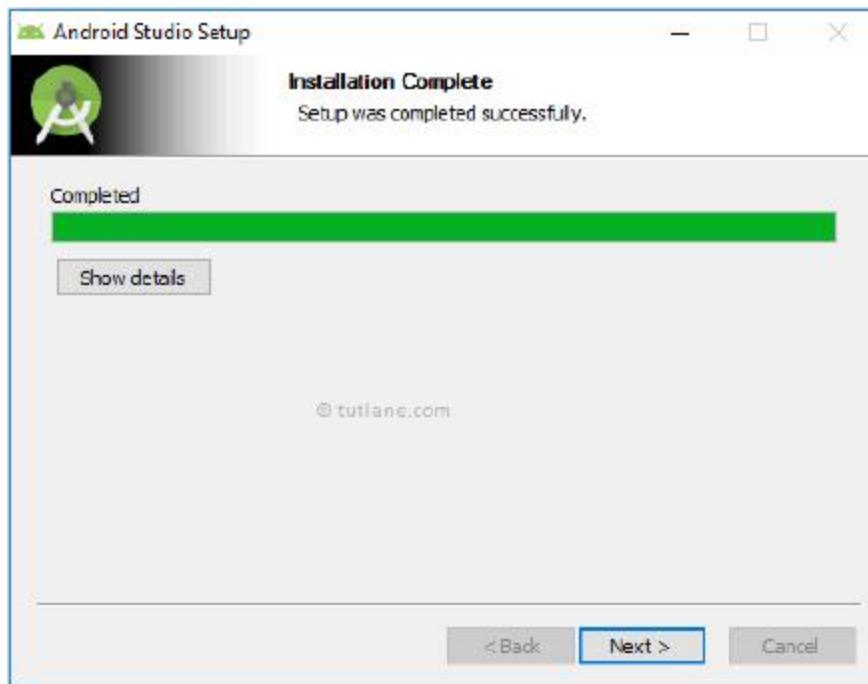
Bây giờ chúng ta cần chỉ định vị trí ổ đĩa máy cục bộ để cài đặt Android Studio và Android SDK. Sau khi chọn đường dẫn vị trí để cài đặt các thành phần cần thiết, bạn nhấn Next như hình bên dưới.



Bây giờ bạn chọn Start Menu Folder để tạo phím tắt cho Android Studio và nhấp vào Install như hình dưới đây.



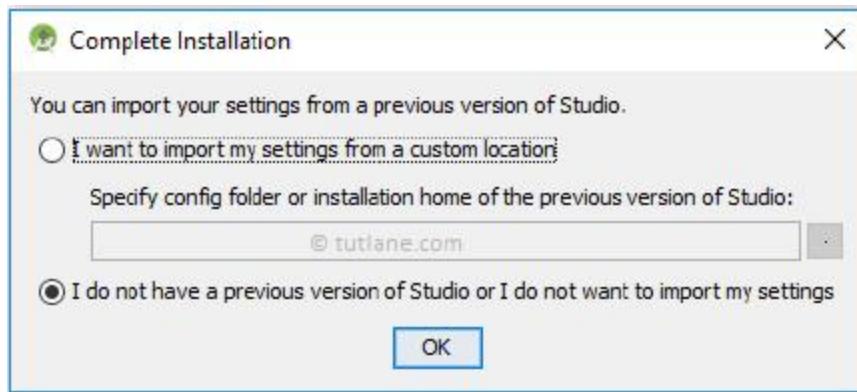
Sau khi chúng ta nhấp vào nút Install, quá trình cài đặt sẽ bắt đầu và nhấp vào nút Next sau khi hoàn thành cài đặt như hình dưới đây.



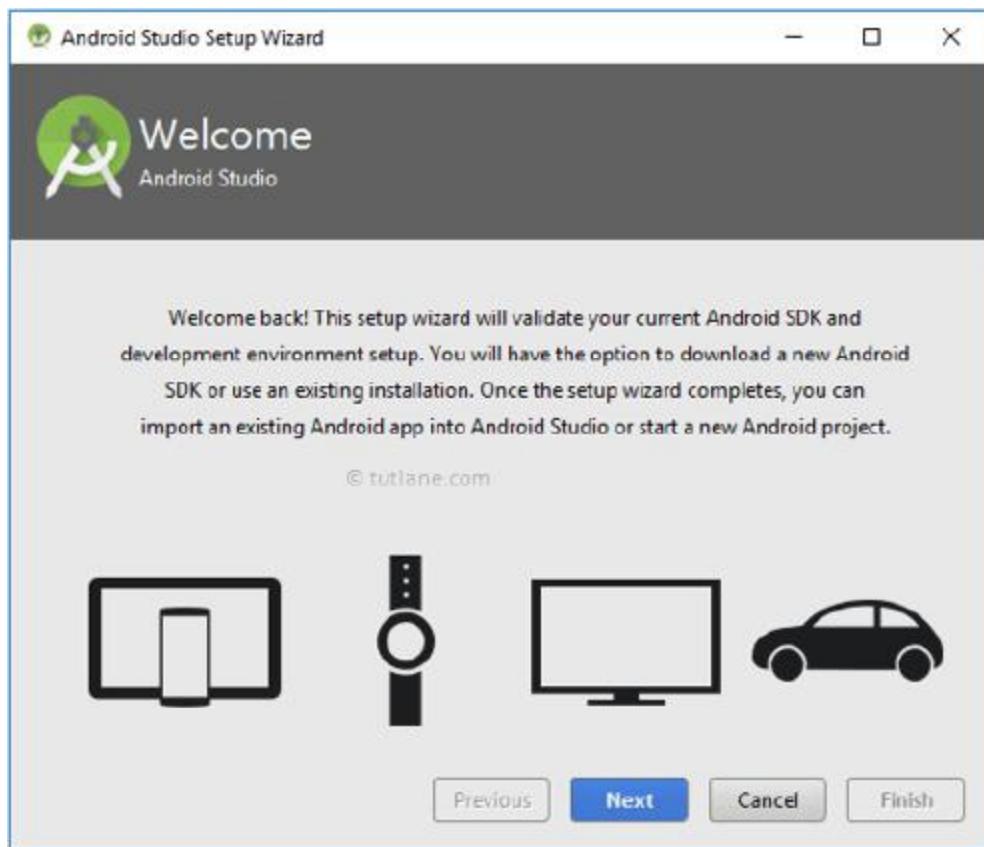
Sau đó, nó sẽ hiển thị thông báo hoàn thành cài đặt, nhấn vào Finish để khởi chạy Android Studio như hình dưới đây.



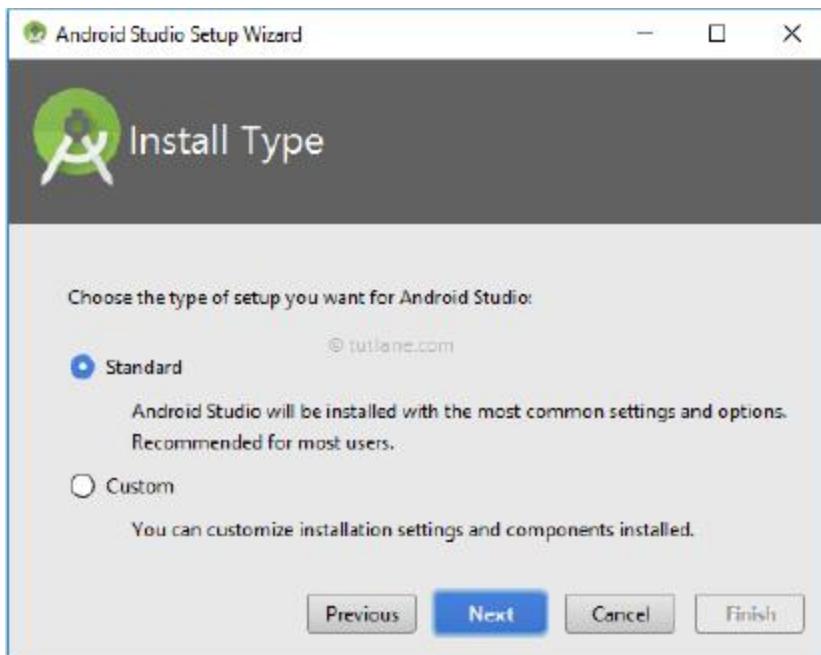
Trong khi khởi chạy Android Studio, nó sẽ cung cấp cho bạn một tùy chọn để nhập cài đặt (import settings) từ phiên bản trước của Studio. Trong trường hợp nếu bạn chưa có bất kỳ phiên bản nào trước đó, hãy chọn tùy chọn thứ hai và bấm OK như hình dưới đây.



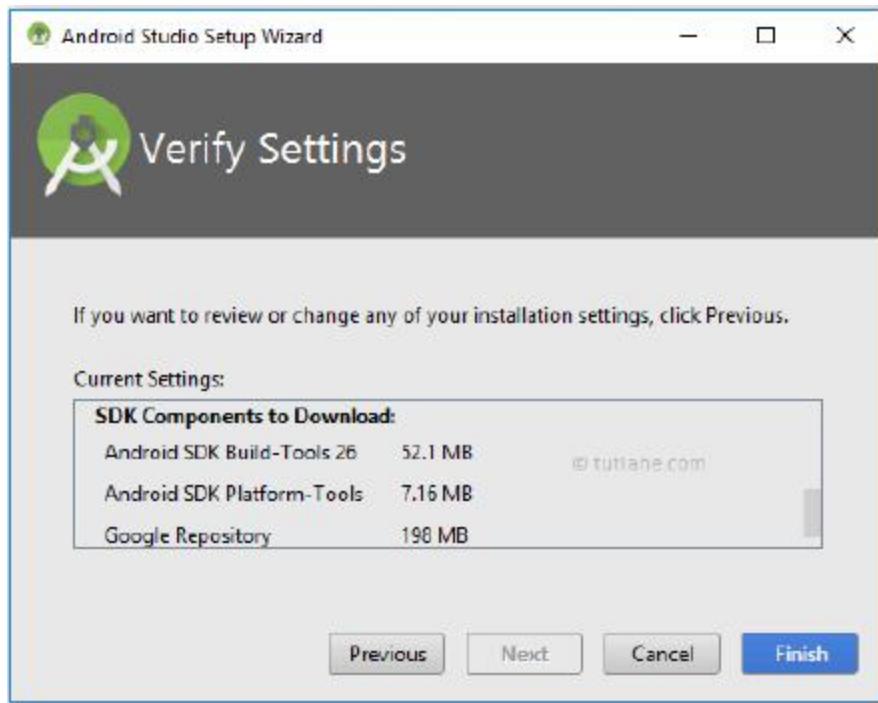
Bây giờ Android Studio sẽ mở một cửa sổ hướng dẫn chào mừng, nhấp vào Next để xác thực Android SDK và thiết lập môi trường phát triển như hình dưới đây.



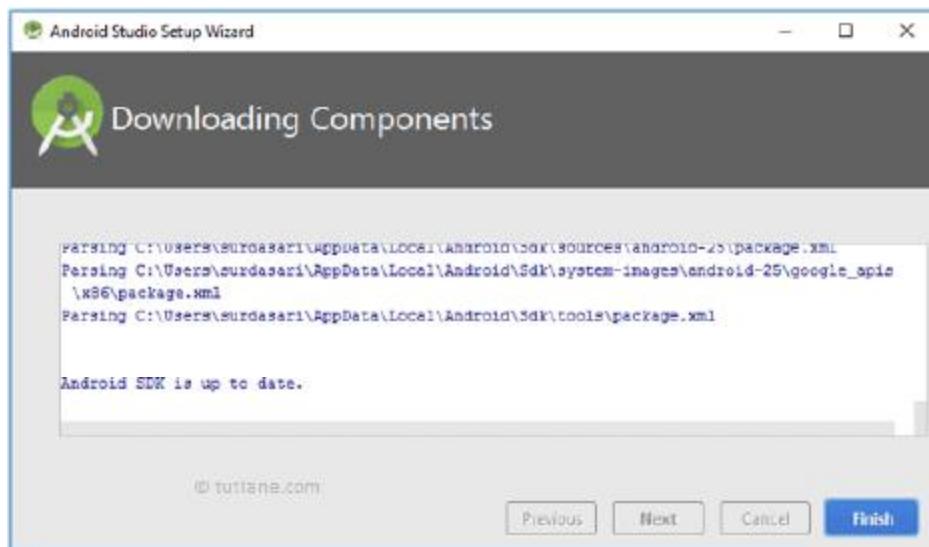
Bây giờ hãy chọn kiểu cài đặt chuẩn (Standard installation) và nhập vào Next để cài đặt các Settings và tùy chọn phổ biến như hình dưới đây.



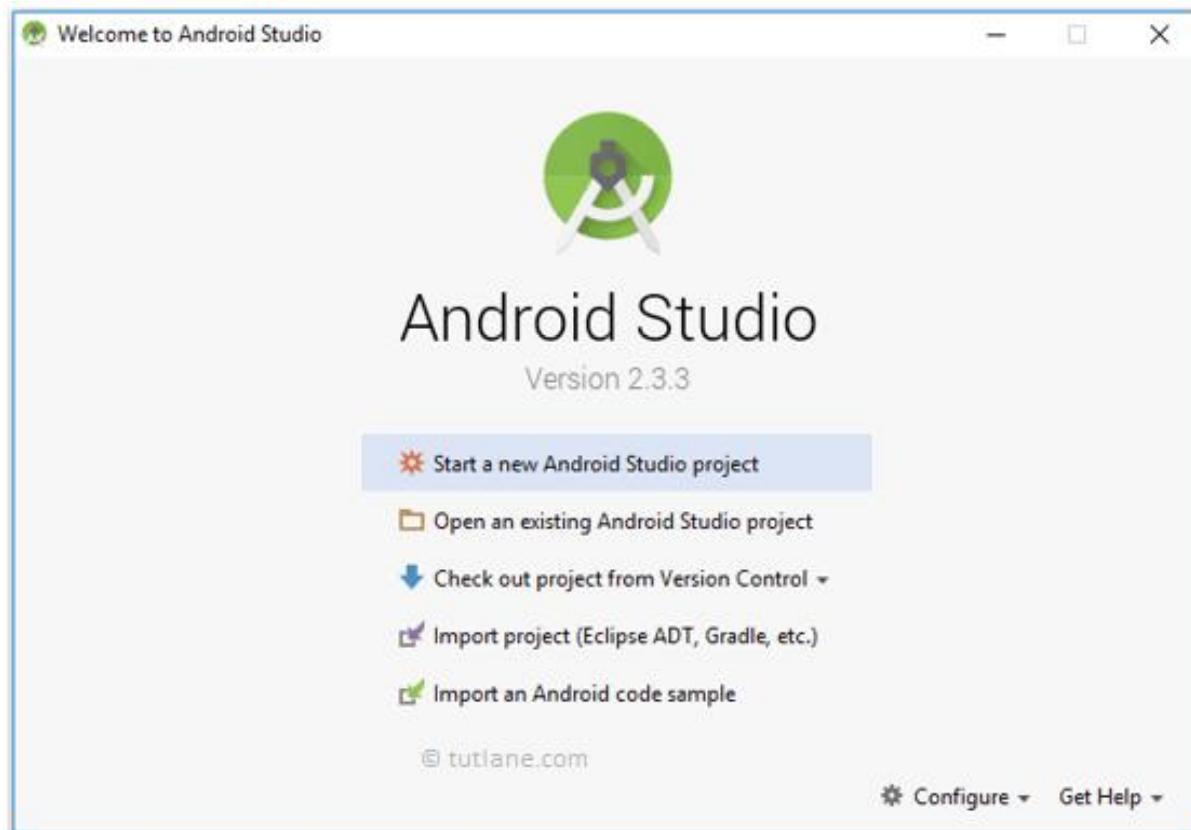
Bây giờ xác minh cài đặt và nhấp vào Finish để hoàn tất quá trình cài đặt Android Studio như hình dưới đây.



Sau khi hoàn thành việc cài đặt các thành phần cần thiết, click vào Finish như hình bên dưới.



Sau khi hoàn thành cài đặt tất cả các thành phần cần thiết, chúng ta sẽ có thể thấy cửa sổ chào mừng của Android Studio như hình dưới đây.



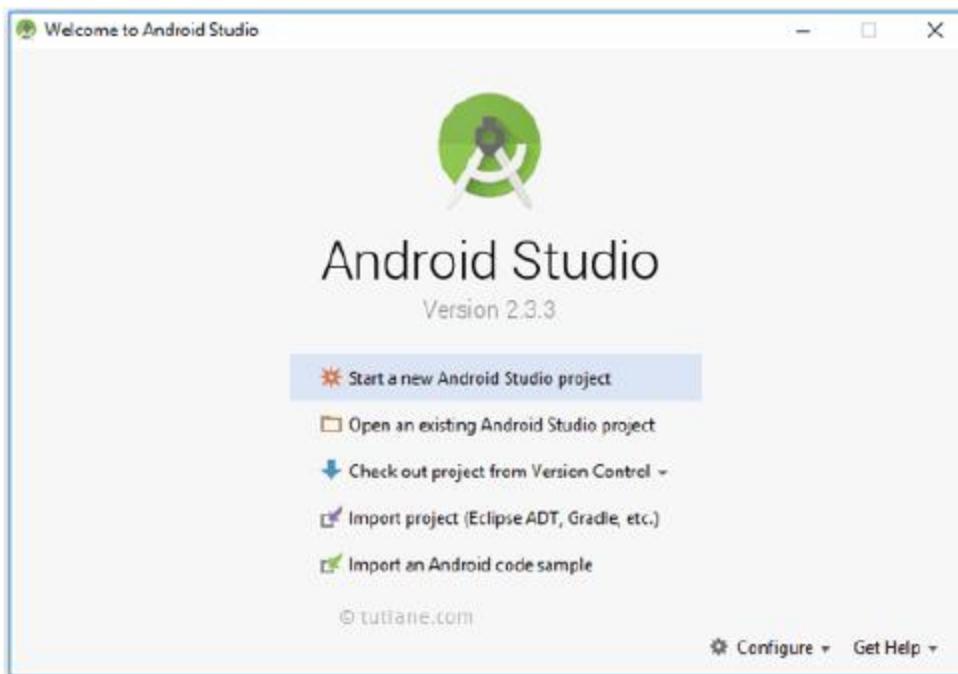
## **2.2 TẠO ỨNG DỤNG ANDROID**

### **2.2.1 Ứng dụng Hello World**

Bằng cách sử dụng Android Studio IDE (Integrated Development Environment - môi trường phát triển tích hợp), chúng ta có thể triển khai các ứng dụng Android cần thiết theo yêu cầu của chúng ta.

Để hiện thực ứng dụng Android Hello World, trước tiên chúng ta cần thiết lập một môi trường phát triển bằng cách sử dụng Android Studio IDE được Google cung cấp miễn phí cho các nhà phát triển Android (Android developers). Trong trường hợp nếu bạn không biết cách thiết lập môi trường phát triển Android, hãy từng bước làm theo hướng dẫn Android Environment Setup.

Khi chúng ta đã hoàn tất cài đặt Android Studio, hãy mở Android Studio và nó sẽ giống như hình dưới đây.



Ở đây, chúng ta sẽ chọn tùy chọn New Project vì chúng ta chưa tạo bất kỳ dự án nào khác và chúng ta cần tạo một cái mới. Vì vậy, chúng ta sẽ chọn New Project từ các tùy chọn đã cho.

Tuy nhiên, chúng ta có thể chọn Import Project nếu muốn nhập (import) dự án từ bất kỳ cách nào khác, chẳng hạn như dự án Eclipse vào Android Studio. Android Studio sẽ chuyển đổi dự án Eclipse thành dự án Android Studio, thêm các tệp cấu hình cần thiết cho chúng ta.

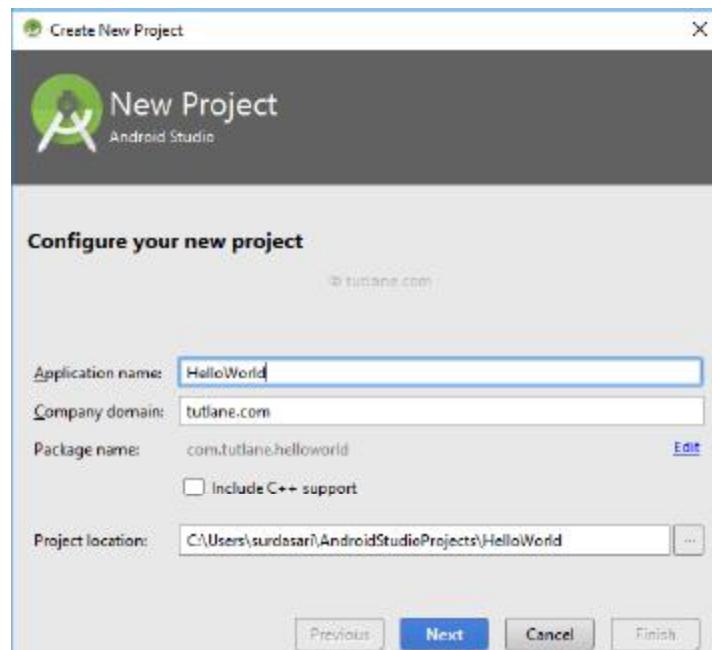
Nếu chúng tôi chọn Open Project từ danh sách tùy chọn, chúng ta có thể mở các dự án được tạo bằng Android Studio hoặc IntelliJ IDEA.

Kiểm tra từ Version Control, chúng ta có thể kiểm tra bản sao của một dự án đang được kiểm soát phiên bản. Đây là một cách tuyệt vời để nhanh chóng bắt kịp với một dự án hiện có.

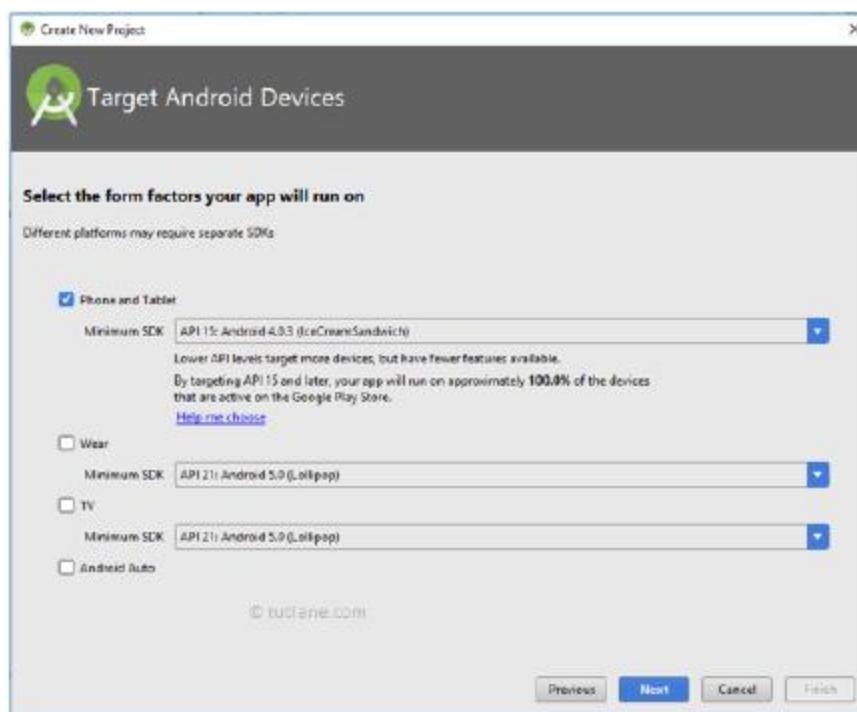
Để bắt đầu, hãy chọn New Project từ danh sách các tùy chọn. Điều này sẽ hiển thị cho chúng ta danh sách các tùy chọn để cấu hình dự án mới của chúng ta.

Khi chúng ta nhấp vào "New Project" từ tùy chọn trên, màn hình tiếp theo sẽ mở ra như thế này, nơi chúng ta phải đề cập đến tên Project, tên miền của công ty và vị trí lưu Project (chúng ta gọi đó là đường dẫn chính nơi ứng dụng này sẽ được lưu) vì tên gói (Package Name) sẽ được tạo tự động khi chúng ta tạo dự án trong Android Studio.

Sau khi nhập tất cả các chi tiết, nếu chúng ta nhấp vào nút Next, một màn hình khác sẽ xuất hiện, nơi chúng ta đã chọn các nền tảng (platforms) và mục tiêu SDK khác nhau như hình dưới đây dựa trên yêu cầu của chúng ta.

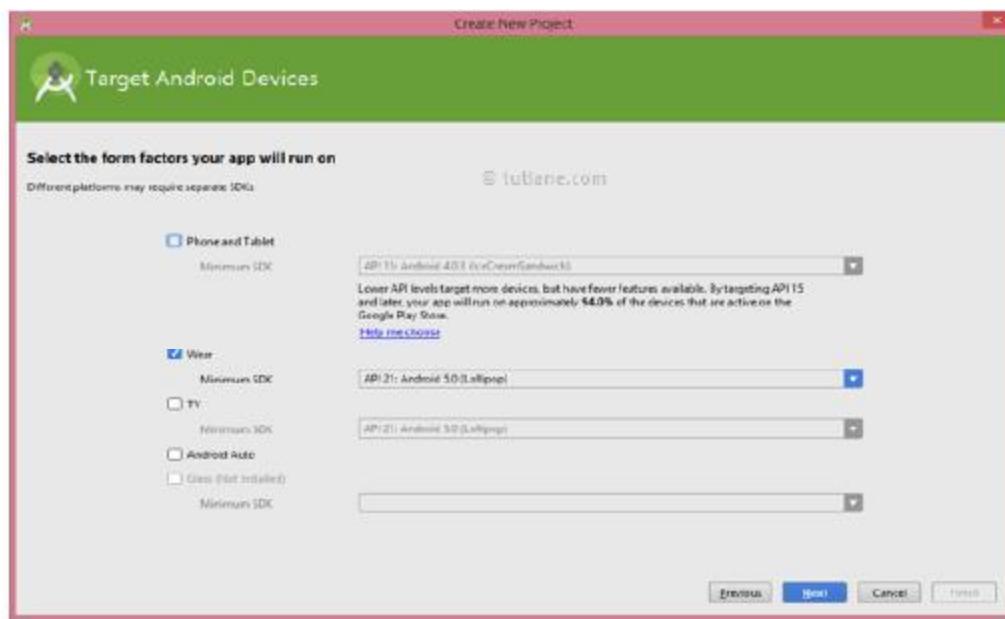


Sau khi nhập tất cả các chi tiết, nếu chúng ta nhấp vào nút Next, một màn hình khác sẽ xuất hiện, nơi chúng ta đã chọn các nền tảng (platforms) và mục tiêu SDK khác nhau như hình dưới đây dựa trên yêu cầu của chúng ta.

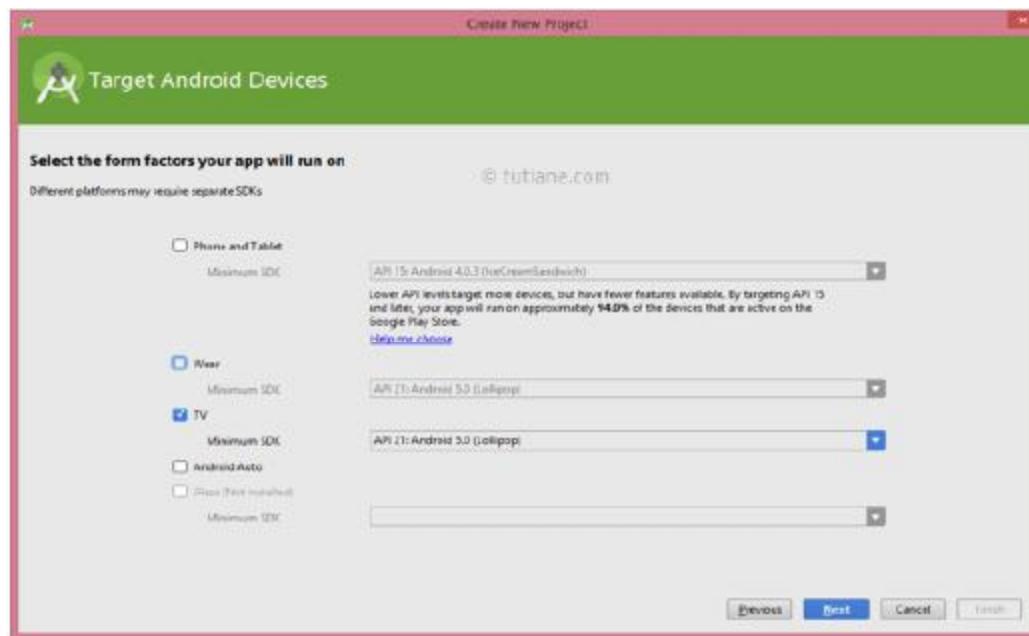


Ở đây, chúng ta cần chọn loại platforms mà chúng ta sẽ sử dụng để phát triển ứng dụng, chẳng hạn như nếu chúng ta chọn “Phone and Tablet”, thì nó sẽ hiển thị phiên bản API và SDK khác nhau.

Nếu chúng ta chọn “Wear”, thì nó sẽ hiển thị các phiên bản API và SDK của nó như hình bên dưới.



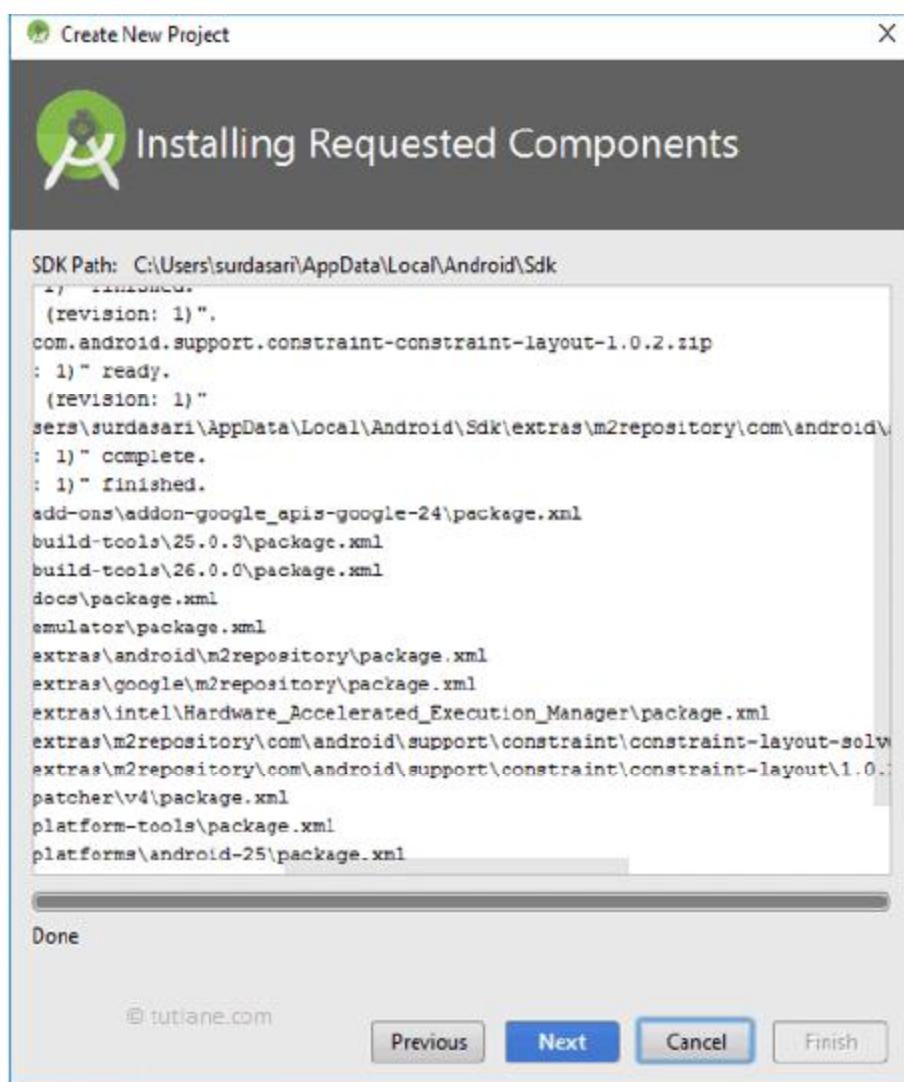
Trong trường hợp nếu chúng ta chọn “TV”, thì nó sẽ hiển thị các phiên bản API và SDK của tivi như hình bên dưới.



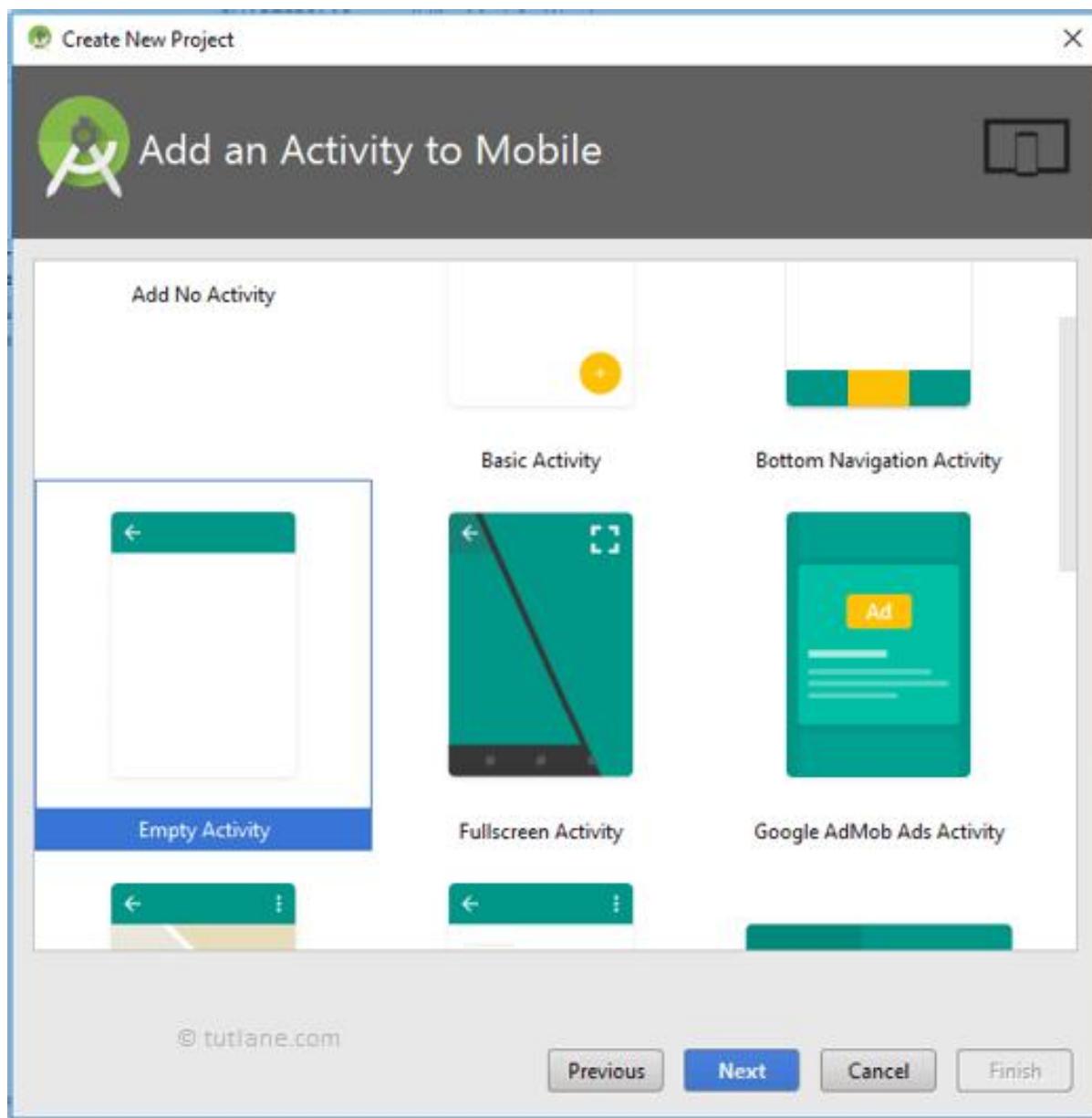
Wear: chúng ta sử dụng tùy chọn này cho Android Watches (đồng hồ Android) mà chúng ta có thể đeo trên tay và sử dụng chức năng tương tự như chúng tôi làm với các thiết bị Android. Ta có thể gọi điện, đặt báo thức, chụp ảnh và nhiều thứ khác một cách dễ dàng.

TV: chúng ta sử dụng tùy chọn này cho SmartIPTV rất phổ biến hiện nay. Chúng ta có thể xem các kênh yêu thích của mình như chúng ta thấy trên ti-vi tại nhà và thực hiện các chuyển kênh một cách dễ dàng.

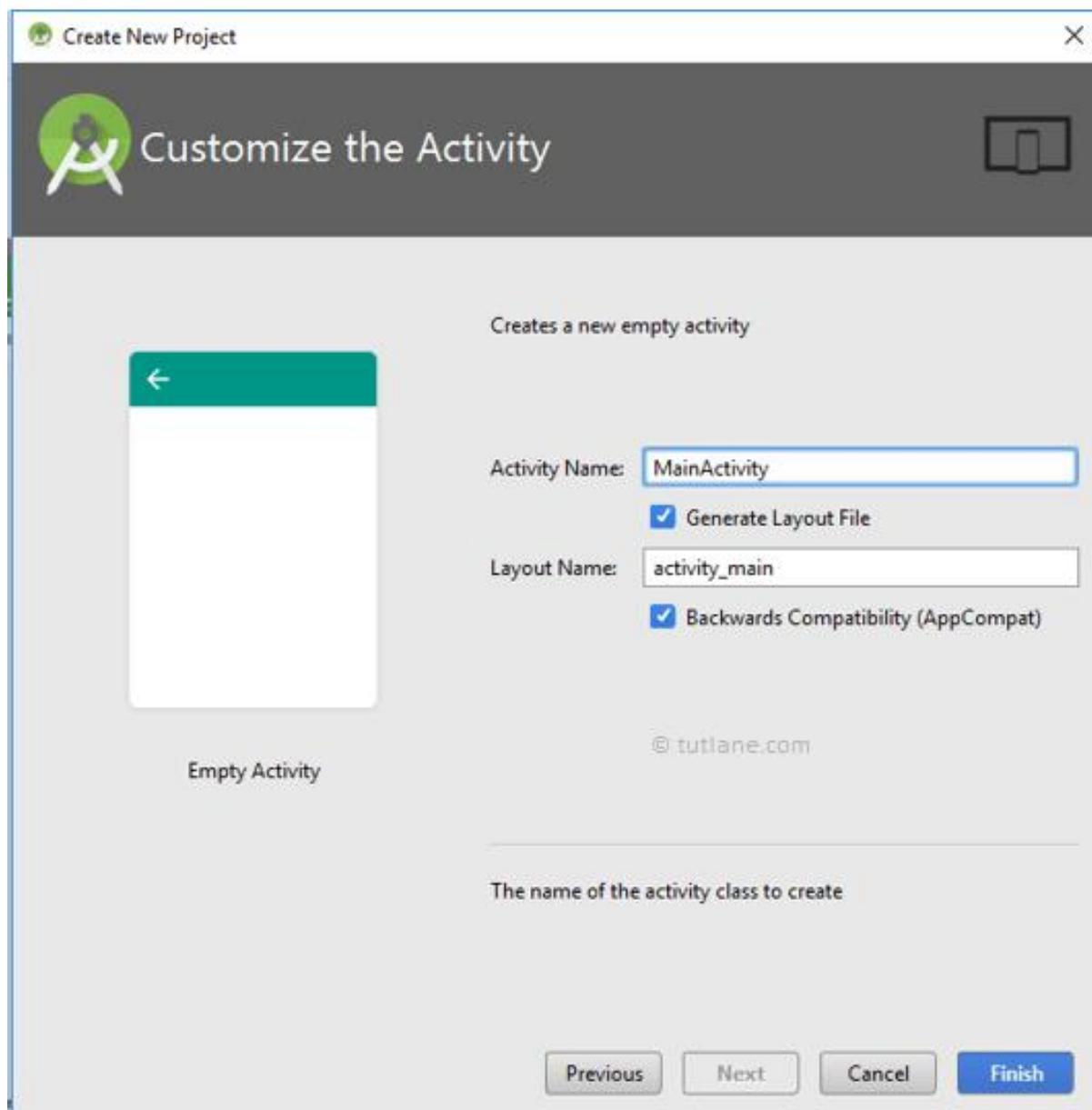
Ở đây chúng ta sẽ triển khai một ứng dụng cho điện thoại và máy tính bảng, vì vậy chúng ta đã chọn tùy chọn điện thoại và máy tính bảng và nhấp vào Next và nó sẽ cài đặt các thành phần bắt buộc như hình dưới đây.



Bây giờ hãy nhấp vào Next để chọn Activity cụ thể theo yêu cầu của chúng ta. Nếu chúng ta chọn Empty Activity, thì nó sẽ hiển thị Empty Activity trong bố cục (layout) của chúng ta. Trong trường hợp nếu chúng ta chọn các tùy chọn khác, thì nó sẽ hiển thị Activity mà chúng ta đã chọn. Ở đây chúng ta đang chọn Empty Activity trông như hình bên dưới.

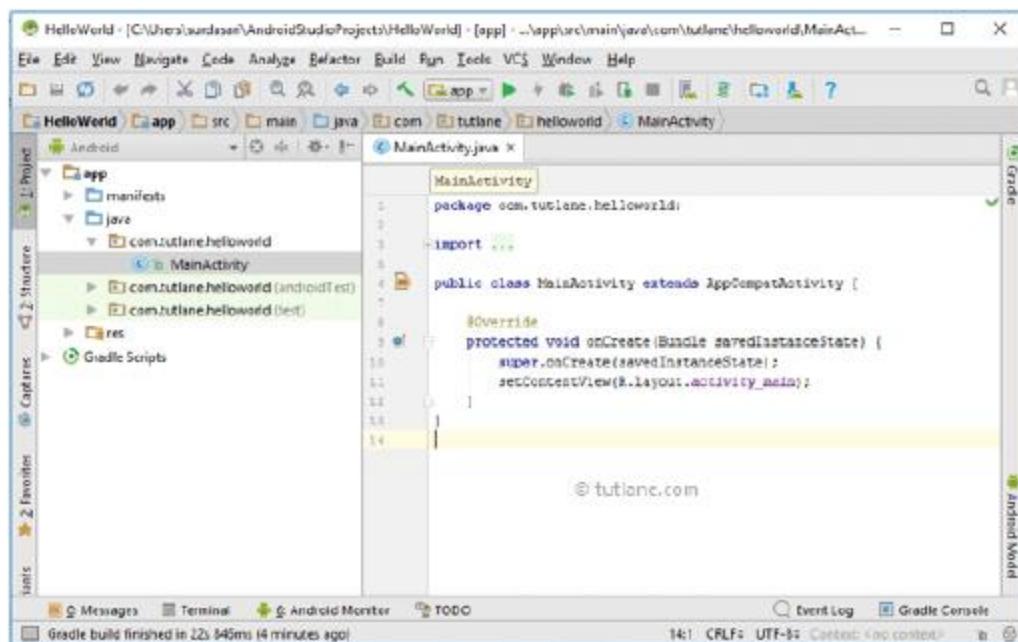


Sau khi chọn Activity cho ứng dụng của chúng ta, hãy nhấp vào nút Next và nó sẽ đưa bạn đến màn hình tiếp theo như hình dưới đây.



Ở đây chúng ta có thể thấy Activity, tức là Empty Activity mà chúng ta đã chọn trong phần trước và tên tập tin java là "MainActivity". Bây giờ chúng ta đã sẵn sàng cho bước cuối cùng, chỉ cần nhấp vào nút "Finish" và nó sẽ đưa bạn đến Main Page (trang chính), nơi chúng ta phải viết mã và tạo bô cục (layouts) mới ở đó.

Sau khi nhấp vào Finish, chúng ta sẽ thấy giao diện người dùng của Android Studio với Project Explorer (trình khám phá dự án) ở bên trái và workspace (không gian làm việc) ở bên phải như hình dưới đây.

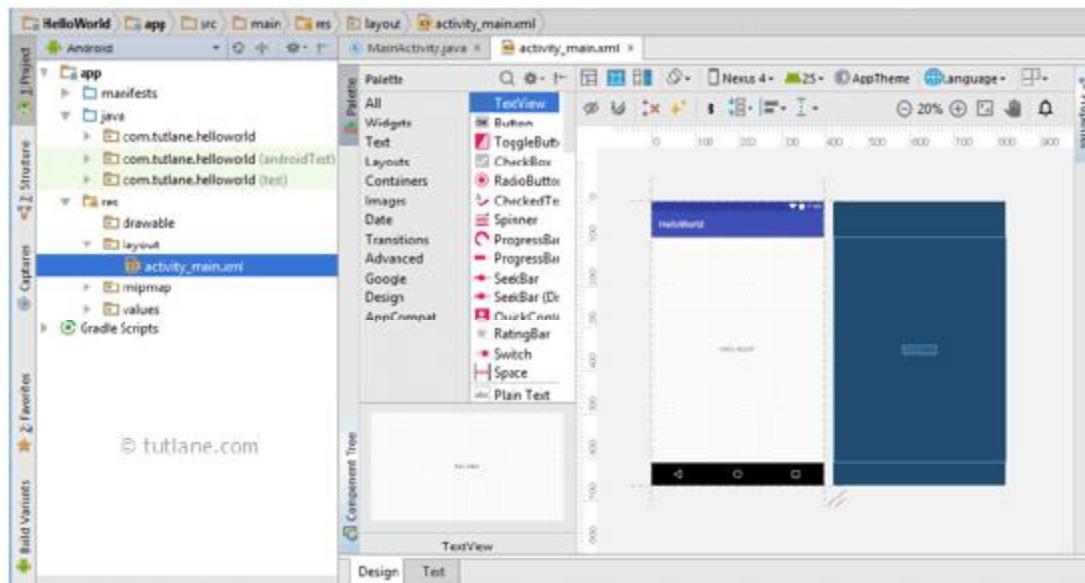


Để biết thêm về các thư mục trong ứng dụng Android, hãy kiểm tra Android App Folder Structure này (cấu trúc thư mục ứng dụng Android). Sau đây là các tập tin quan trọng mà chúng ta cần để xây dựng ứng dụng của mình trong Android Studio.

## 2.2.2 Android Layout File

Tập tin bô cục Android (activity\_main.xml)

Giao diện người dùng của ứng dụng của chúng ta sẽ được thiết kế trong file này và nó sẽ chứa các chế độ Design và Text. Nó sẽ tồn tại trong thư mục layouts và cấu trúc của file activity\_main.xml ở chế độ Design như hình bên dưới.



Chúng ta có thể thực hiện các sửa đổi thiết kế bắt buộc trong tệp activity\_main.xml bằng cách sử dụng chế độ Design hoặc Text. Nếu chúng ta chuyển sang chế độ Text thì tập tin activity\_main.xml sẽ chứa đoạn mã như hình bên dưới.

```
<?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.tutlane.helloworld.MainActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

## 2.2.3 Android Main Activity File

(MainActivity.java)

Tệp hoạt động chính trong ứng dụng android là MainActivity.java và nó sẽ tồn tại trong thư mục java. Tệp MainActivity.java sẽ chứa mã java để xử lý tất cả các hoạt động liên quan đến ứng dụng của chúng ta.

Sau đây là mã mặc định của tệp MainActivity.java được tạo bởi ứng dụng HelloWorld của chúng ta.

```
package com.tutlane.helloworld;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

## 2.2.4 Android Manifest File

(AndroidManifest.xml)

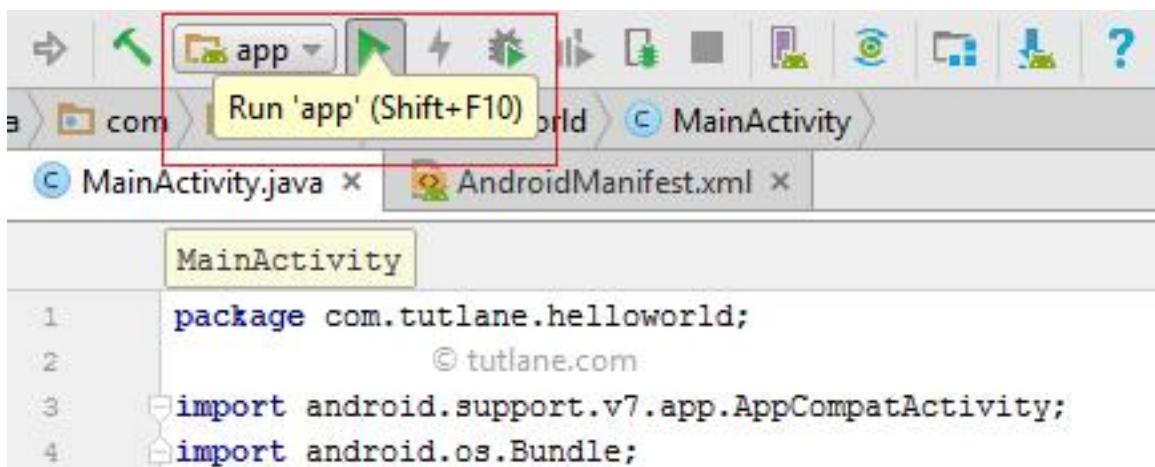
Nói chung, ứng dụng của chúng ta sẽ chứa nhiều hoạt động và chúng ta cần xác định tất cả các hoạt động đó trong tệp AndroidManifest.xml. Trong tệp kê khai của chúng ta, chúng ta cần đề cập đến hoạt động chính cho ứng dụng của mình bằng cách sử dụng thuộc tính danh mục MAIN action và LAUNCHER trong intent filters (<intent-filter>). Trong trường hợp nếu chúng ta không đề cập đến hành động MAIN hoặc danh mục LAUNCHER cho hoạt động chính, biểu tượng ứng dụng của chúng ta sẽ không xuất hiện trong danh sách ứng dụng của màn hình chính.

Sau đây là mã mặc định của tệp AndroidManifest.xml được tạo bởi ứng dụng HelloWorld của chúng ta.

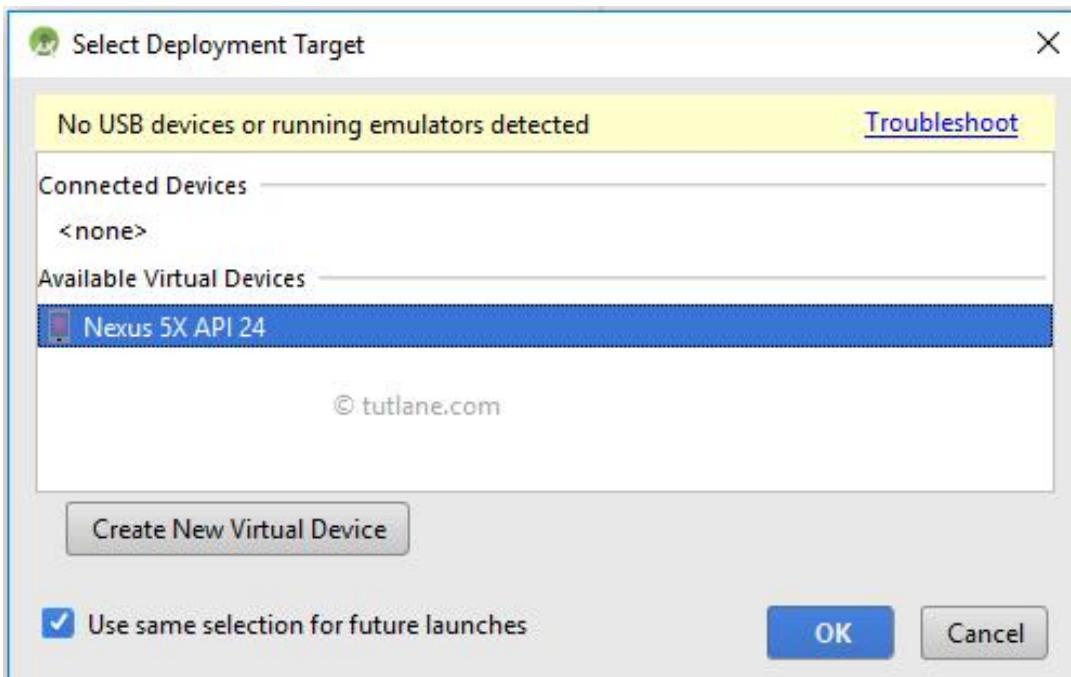
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tutlane.helloworld" >
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
        <activity android:name=".MainActivity" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

## 2.2.5 Run Android Hello World App

Để chạy các ứng dụng android chúng ta cần nhấn vào nút Run hoặc nhấn tổ hợp phím Shift + F10 như hình dưới đây.



Sau khi bấm vào nút play cửa sổ mới sẽ mở ra trong đó bạn chọn Android Virtual Device (AVD) và bấm OK như hình bên dưới.

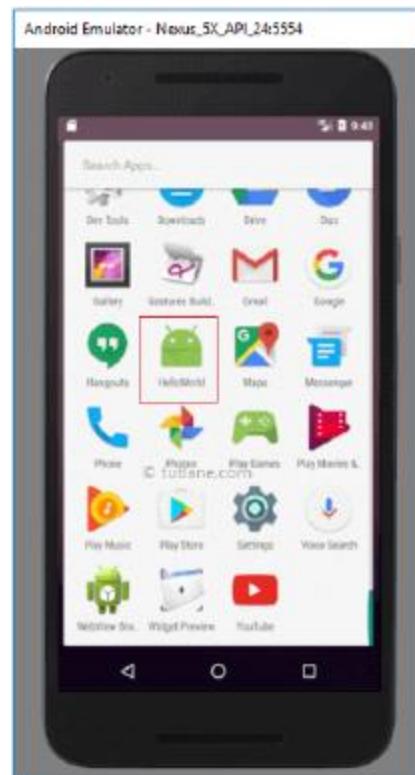


Trong trường hợp nếu bạn không thể nhìn thấy bất kỳ thiết bị ảo nào, thì bạn cần tạo một thiết bị ảo để chạy ứng dụng của mình để kiểm tra Android Virtual Device Setup này.

Lúc này ứng dụng hello world cho android của chúng ta sẽ hiện ra kết quả như hình bên dưới



Trong tệp AndroidManifest.xml của chúng ta, chúng ta đã đề cập đến thuộc tính danh mục MAIN action và LAUNCHER cho tệp hoạt động chính của chúng ta do biểu tượng ứng dụng của chúng ta sẽ tạo trong danh sách các ứng dụng trên màn hình HOME như được hiển thị bên dưới.



Đây là cách chúng ta có thể tạo ứng dụng trong Android và thực thi các ứng dụng dựa trên yêu cầu của chúng ta.

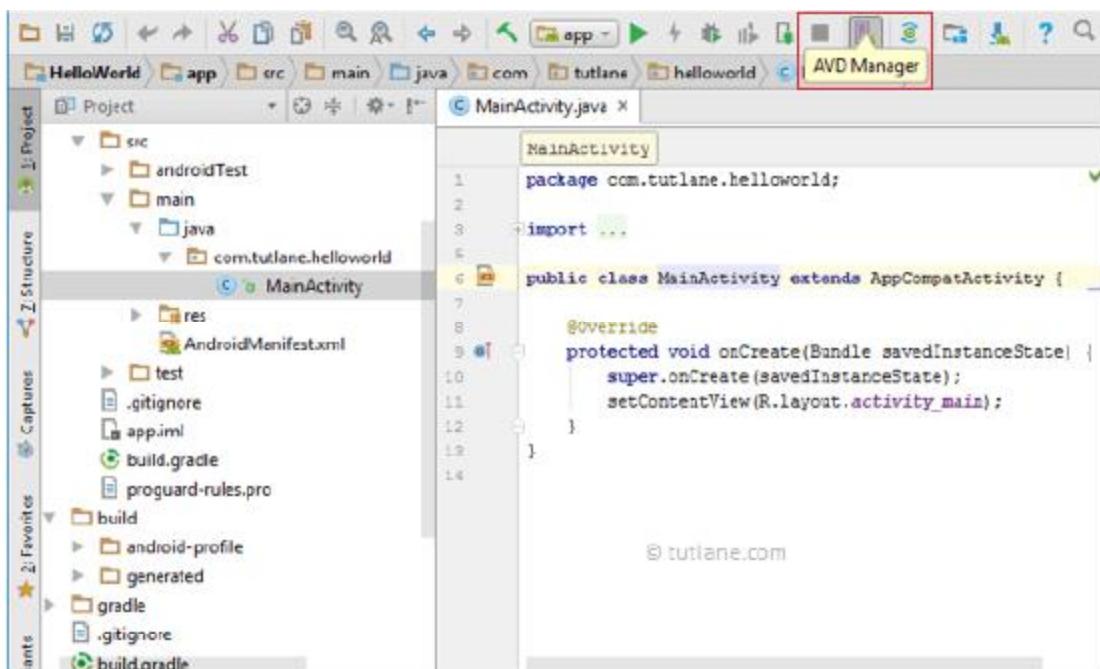
## 2.3 THIẾT LẬP ANDROID EMULATOR

(Create AVD - Android Virtual Device)

Thiết bị ảo Android (AVD) là một trình giả lập được sử dụng để tái tạo chức năng của điện thoại, máy tính bảng, Android Wear hoặc TV để kiểm tra cục bộ các ứng dụng Android của chúng ta. Bằng cách sử dụng giao diện trình quản lý AVD trong studio android, chúng ta có thể thiết lập trình giả lập thiết bị ảo Android để kiểm tra các ứng dụng của chúng ta.

### 2.3.1 Tạo Android Virtual Device

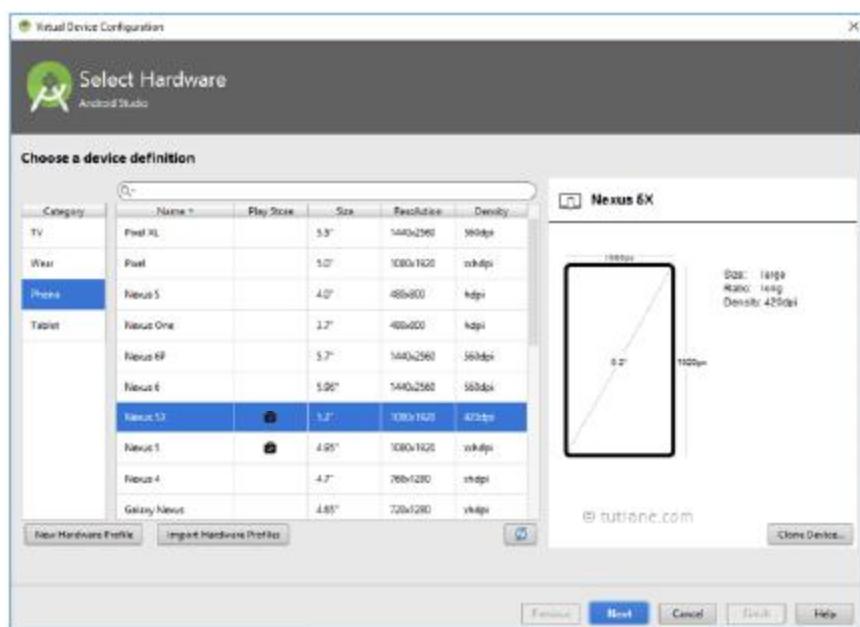
Để kiểm tra ứng dụng Android của chúng ta, chúng ta nên có Thiết bị ảo Android (AVD). Chúng ta có thể tạo thiết bị ảo bằng cách nhấp vào AVD Manager như hình dưới đây.



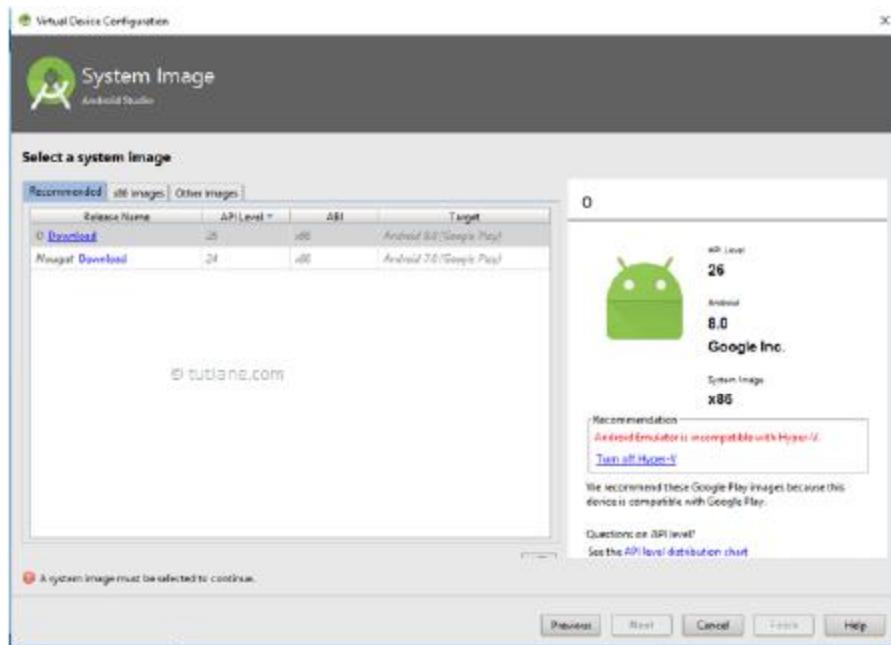
Khi chúng ta nhấp vào AVD Manager,, một cửa sổ mới sẽ mở ra trong đó nhấp vào Create Virtual Device như hình dưới đây.



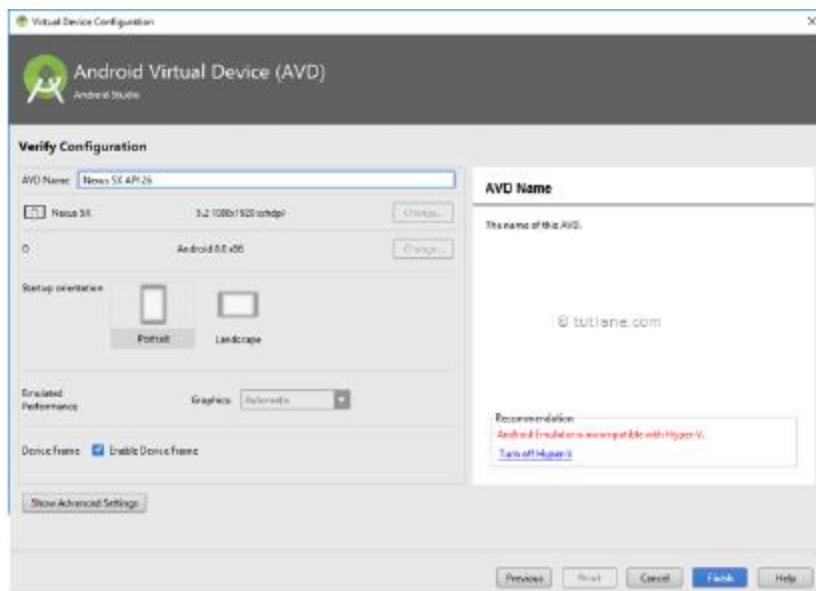
Bây giờ hãy chọn loại thiết bị cần thiết và Click Next để tạo một thiết bị ảo như hình bên dưới.



Bây giờ chúng ta cần tải về và chọn system image và nhấn Next như hình bên dưới.



Bây giờ xác minh cấu hình của thiết bị ảo android (AVD) và nhấp vào Finish như hình dưới đây.

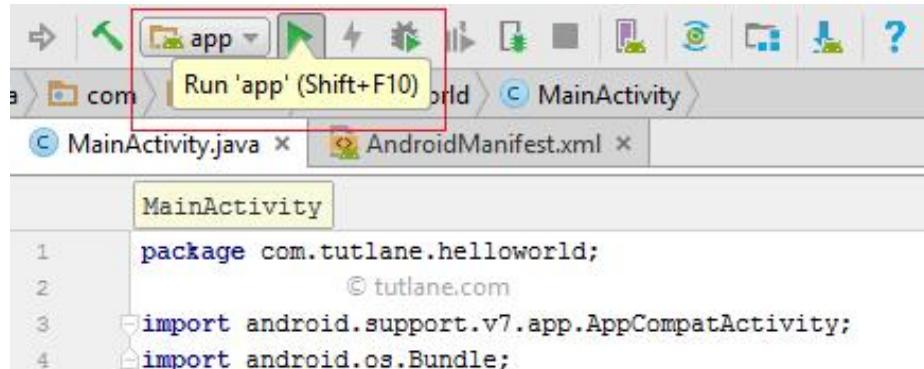


Đây là cách chúng ta cần thêm thiết bị ảo Android (AVD) vào một studio android để kiểm tra các ứng dụng Android của chúng ta.

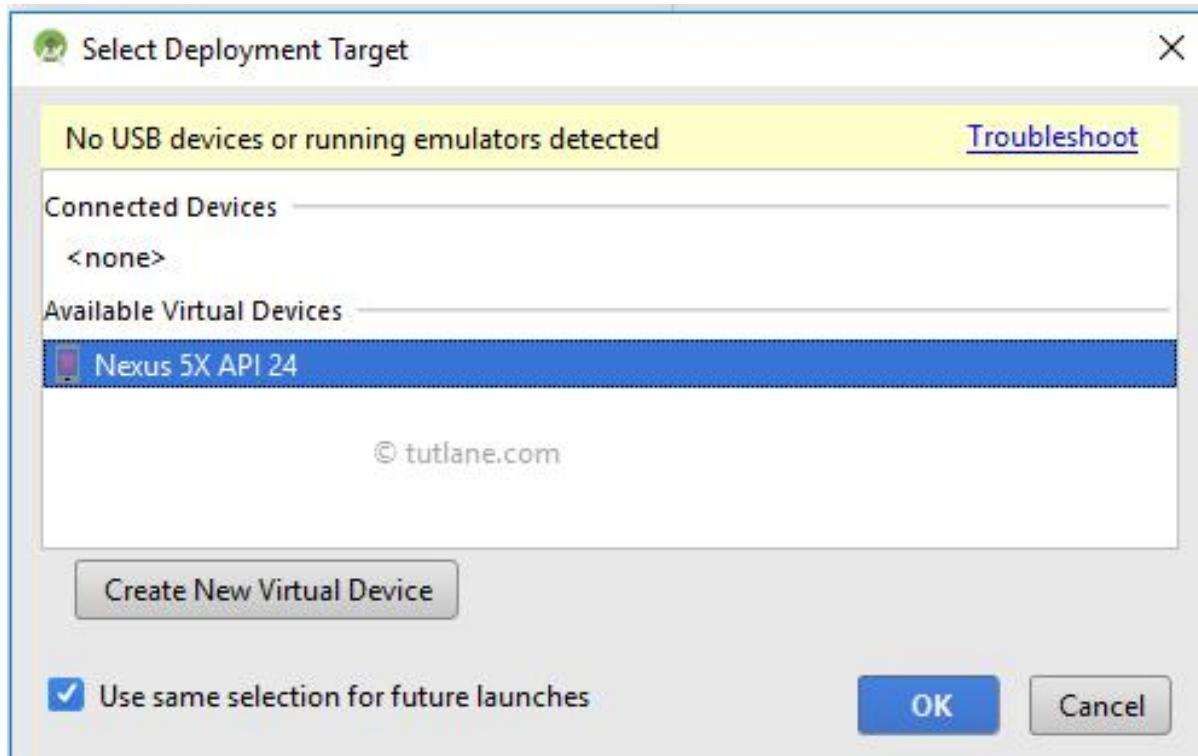
Khi chúng ta đã hoàn tất việc thiết lập thiết bị ảo android trong android studio, hãy tạo một ứng dụng mẫu trong android studio và chạy ứng dụng bằng trình quản lý AVD. Trong trường hợp nếu bạn không biết về việc tạo ứng dụng, hãy kiểm tra Ứng dụng Android Hello World này.

### 2.3.2 Chạy Ứng dụng Android

Để chạy các ứng dụng android chúng ta cần nhấp vào nút Run hoặc nhấn tổ hợp phím Shift + F10 như hình dưới đây



Sau khi bấm vào nút play cửa sổ mới sẽ mở ra trong đó bạn chọn Android Virtual Device (AVD) và bấm OK như hình bên dưới.



Đây là cách chúng ta có thể thiết lập trình giả lập thiết bị ảo Android (AVD) trong android studio để tái tạo chức năng của thiết bị Android thực.

## 2.4 THÀNH PHẦN TRONG ỨNG DỤNG ANDROID

Trong android, các thành phần ứng dụng là các khối xây dựng cơ bản của một ứng dụng và các thành phần này sẽ hoạt động như một điểm vào để cho phép hệ thống hoặc người dùng truy cập vào ứng dụng của chúng ta. Các thành phần trong một ứng dụng Android: Activities, Intents, Views, Layouts, Services,...

### 2.4.1 Các thành phần cơ bản

Sau đây là các thành phần ứng dụng cốt lõi cơ bản có thể được sử dụng trong ứng dụng Android.

- Activities
- Intents
- Content Providers
- Broadcast Receivers
- Services

Tất cả các thành phần ứng dụng này được xác định trong tệp mô tả ứng dụng android (AndroidManifest.xml) như hình dưới đây.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ....>
    <application android:allowBackup="true" android:icon="@mipmap/ic_launcher" ....>
        <activity android:name=".MainActivity" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        ....
    </application>
</manifest>
```

Đây là cách chúng ta có thể xác định các thành phần ứng dụng Android trong tệp AndroidManifest.xml.

## 2.4.2 Android Activities

Trong android, Activity đại diện cho một màn hình duy nhất có giao diện người dùng (UI) và nó sẽ đóng vai trò là điểm vào để người dùng tương tác với ứng dụng.

Ví dụ: một ứng dụng danh bạ đang có nhiều hoạt động như hiển thị danh sách liên hệ, thêm liên hệ mới và một hoạt động khác để tìm kiếm địa chỉ liên hệ. Tất cả các hoạt động này trong ứng dụng liên hệ độc lập với nhau nhưng sẽ hoạt động cùng nhau để mang lại trải nghiệm người dùng tốt hơn.

Để biết thêm về các Activity trong Android, hãy kiểm tra Android Activities trên Android.

## 2.4.3 Android Intents

Trong android, Intent là một đối tượng truyền thông tin được sử dụng để yêu cầu một hành động từ một thành phần khác.

Trong android, các intent chủ yếu được sử dụng để thực hiện những điều sau đây.

- Khởi động một Activity
- Khởi động một Service
- Phân phát một Broadcast

Có hai loại intent có sẵn trong Android, đó là:

- Implicit Intents (intent không tường minh)
- Explicit Intents (intent tường minh)

## 2.4.4 Android Services

Trong android, service là một thành phần giữ cho ứng dụng chạy trong chế độ background để thực hiện các hoạt động lâu dài dựa trên yêu cầu của chúng ta. Đối với Service, chúng ta không có bất kỳ giao diện người dùng nào và giao diện này sẽ chạy các ứng dụng ở chế độ background như phát nhạc background khi người dùng đang chạy các ứng dụng khác nhau.

Chúng ta có hai loại service có sẵn trong Android, đó là

- Local Services
- Remote Services

## 2.4.5 Android Broadcast Receivers

Trong android, Broadcast Receiver là một thành phần sẽ cho phép hệ thống gửi các sự kiện đến ứng dụng như gửi thông báo pin yếu đến ứng dụng. Các ứng dụng cũng có thể bắt đầu phát sóng để cho các ứng dụng khác biết rằng dữ liệu cần thiết có sẵn trong thiết bị để sử dụng.

Nói chung, chúng ta sử dụng Intents để cung cấp các sự kiện theo hình thức broadcast đến các ứng dụng khác và Broadcast Receivers sử dụng thông báo trên thanh notifications để cho người dùng biết rằng sự kiện broadcast xảy ra.

## 2.4.6 Android Content Providers

Trong android, Content Providers hữu ích để trao đổi dữ liệu giữa các ứng dụng dựa trên các yêu cầu. Content Providers có thể chia sẻ dữ liệu ứng dụng được lưu trữ trong hệ thống tệp, cơ sở dữ liệu SQLite, trên web hoặc bất kỳ vị trí lưu trữ nào khác mà ứng dụng của chúng tôi có thể truy cập.

Bằng cách sử dụng Content Providers, các ứng dụng khác có thể truy vấn hoặc sửa đổi dữ liệu của ứng dụng của chúng ta dựa trên các quyền do Content Providers cung cấp. Ví dụ: android cung cấp Content Providers (`ContactsContract.Data`) để quản lý thông tin liên hệ, bằng cách sử dụng các quyền thích hợp, bất kỳ ứng dụng nào cũng có thể truy vấn nhà cung cấp nội dung để thực hiện các thao tác đọc và ghi thông tin liên hệ.

## 2.4.7 Các Componet bổ sung

Trong android, chúng ta có các component bổ sung được sử dụng để xây dựng mối quan hệ giữa các thành phần trên (Activities, Intents, Content Providers, Services và Broadcast Receivers) để triển khai logic ứng dụng của chúng ta, đó là

Component	Description
Fragments	These are used to represent the portion of user interface in an activity
Layouts	These are used to define the user interface (UI) for an activity or app
Views	These are used to build a user interface for an app using UI elements like buttons, lists, etc.
Resources	To build an android app we required external elements like images, audio files, etc. other than coding
Manifest File	It's a configuration file ( <b>AndroidManifest.xml</b> ) for the application and it will contain the information about <a href="#">Activities</a> , <a href="#">Intents</a> , <a href="#">Content Providers</a> , <a href="#">Services</a> , <a href="#">Broadcast Receivers</a> , permissions, etc.

Đây là các component ứng dụng chính được yêu cầu để xây dựng một ứng dụng Android dựa trên các yêu cầu của chúng ta.

## TÓM TẮT

Trong bài học này cung cấp các kiến thức cơ bản về nền tảng Android bao gồm các đặc điểm chính, các thành phần trong kiến trúc hệ thống Android. Phần chính là trình bày quy trình phát triển các ứng dụng Android, đây là quy trình rất quan trọng đối với người phát triển ứng dụng.

Các công cụ và môi trường để phát triển ứng dụng Android được đề cập cụ thể giúp cho người học nhanh chóng tiếp cận đến phần chính của bài học.

Phần cuối cùng của bài học trình bày các nội dung chính của một ứng dụng Android. Các thành phần này sẽ được trình bày cụ thể trong các bài học tiếp theo.

## BÀI TẬP

**Câu 1:** Hãy mô tả các thành phần chính của kiến trúc Android?

**Câu 2:** Hãy cho biết cách thức thiết lập môi trường để xây dựng ứng dụng Android?

**Câu 3:** Cho biết cụ thể từng giai đoạn của quy trình phát triển ứng dụng Android?

**Câu 4:** Hãy so sánh quy trình phát triển ứng dụng trên máy tính và quy trình phát triển ứng dụng trên Android?

**Câu 5:** Hãy tìm hiểu khái niệm Application Store? Cho biết một số các Application Store điển hình?

**Câu 6:** Các thành phần chính trong ứng dụng Android?

# BÀI 3: XÂY DỰNG ACTIVITY

Học xong bài này người học cần nắm được các nội dung sau.

- Các đặc điểm cơ bản của thành phần chính trong ứng dụng Android là Activity
- Vòng đời hoạt động của các Activity trong ứng dụng Android
- Cách thức tạo ứng dụng Android có giao diện là Activity
- Cách thức xây dựng giao diện người dùng đồ họa trong ứng dụng Android
- Sử dụng XML Layout để xây dựng giao diện cho Activity
- Cách thức sử dụng thành phần resource trong project Android
- Cơ chế xử lý sự kiện cho các thành phần view trong giao diện

## 3.1 CÁC ACTIVITY TRONG ỨNG DỤNG ANDROID

### 3.1.1 Activity là gì?

Trong android, Activity đại diện cho một màn hình duy nhất với giao diện người dùng (UI) của một ứng dụng và nó sẽ đóng vai trò là một điểm vào để người dùng tương tác với một ứng dụng.

Nói chung, các ứng dụng android sẽ chứa nhiều màn hình và mỗi màn hình của ứng dụng của chúng ta sẽ là một phần mở rộng của lớp Activity. Bằng cách sử dụng các activities, chúng ta có thể đặt tất cả các thành phần giao diện người dùng ứng dụng Android của mình trong một màn hình duy nhất.

Từ nhiều activity trong ứng dụng android, một activity có thể được đánh dấu là activity chính và đó là màn hình đầu tiên xuất hiện khi chúng ta khởi chạy ứng dụng. Trong ứng dụng android, mỗi activity có thể khởi động một activity khác để thực hiện các hành động khác nhau dựa trên yêu cầu của chúng ta.

Ví dụ: một ứng dụng danh bạ đang có nhiều hactivity, trong đó màn hình activity chính sẽ hiển thị danh sách các contact và từ màn hình activity chính, chúng ta có thể

khởi chạy các activity khác cung cấp màn hình để thực hiện các tác vụ như thêm một contact mới và tìm kiếm contact. Tất cả các activity này trong ứng dụng liên hệ có liên kết lồng léo với các hoạt động khác nhưng sẽ hoạt động cùng nhau để mang lại trải nghiệm người dùng tốt hơn.

Nói chung, trong Android có một sự phụ thuộc tối thiểu giữa các activity trong một ứng dụng. Để sử dụng các activity trong ứng dụng, chúng ta cần đăng ký thông tin activity đó trong tệp kê khai của ứng dụng (AndroidManifest.xml) và cần quản lý vòng đời hoạt động đúng cách.

Để sử dụng các activity trong ứng dụng của chúng ta, chúng ta cần xác định một activity với các thuộc tính bắt buộc trong tệp kê khai (AndroidManifest.xml) như hình dưới đây

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ....>
    <application ....>
        <activity android:name=".MainActivity" >
            ....
            ....
        </activity>
    ....
</application>
</manifest>
```

Thuộc tính activity android: name sẽ đại diện cho tên của lớp và chúng ta cũng có thể thêm nhiều thuộc tính như icon, label, theme, permissions, v...v... vào một phần tử hoạt động dựa trên yêu cầu của chúng ta.

Trong ứng dụng android, các activity có thể được thực hiện dưới dạng một lớp con của lớp Activity như hình dưới đây.

```
public class MainActivity extends Activity {
}
```

### 3.1.2 Chu kỳ sống của Android Activity

Nói chung, các hoạt động trong ứng dụng Android của chúng ta sẽ trải qua các giai đoạn khác nhau trong vòng đời của chúng (life cycle). Trong android, lớp Activity có 7

phương thức callback như `onCreate()`, `onStart()`, `onPause()`, `onRestart()`, `onResume()`, `onStop()` và `onDestroy()` để mô tả activity sẽ hoạt động như thế nào ở các giai đoạn khác nhau.

Bằng cách sử dụng các phương thức callback của activity, chúng ta có thể xác định cách activity của chúng ta có thể phản ứng khi người dùng truy cập hoặc rời khỏi ứng dụng của chúng ta.

### 3.1.3 Các phương thức Callback

Trong android, một activity trải qua một loạt trạng thái trong suốt thời gian tồn tại của nó. Bằng cách sử dụng các phương thức callback, chúng ta có thể nhận thấy sự chuyển đổi giữa các trạng thái của activity.

Hệ thống Android khởi tạo chương trình của nó trong một activity bắt đầu bằng một lệnh gọi trên phương thức callback tên là `onCreate()`. Có một chuỗi các phương thức callback khởi động một hoạt động và một chuỗi các phương thức callback chia tách một hoạt động.

Phần này sẽ cung cấp cho ta thông tin chi tiết về các phương thức callback để xử lý chuyển đổi trạng thái hoạt động của một activity trong vòng đời hoạt động của nó (lifecycle).

#### 3.1.3.1 Phương thức `onCreate`

Đây là phương thức callback đầu tiên và nó kích hoạt khi hệ thống tạo một activity lần đầu tiên. Trong quá trình tạo ra activity, activity được chuyển sang trạng thái `Created`.

Nếu chúng ta có các công việc khởi động ứng dụng chỉ cần thực hiện một lần trong suốt vòng đời của một activity, thì chúng ta có thể viết code cho các công việc đó trong phương thức `onCreate()`.

Dưới đây là ví dụ về việc xác định phương thức `onCreate()` trong activity của android.

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

Sau khi thực thi xong phương thức `onCreate()`, activity sẽ chuyển sang trạng thái Started và hệ thống gọi phương thức `onStart()`.

### 3.1.3.2 Phương thức onStart

Phương thức callback `onStart()` sẽ gọi khi một activity đi vào trạng thái Started bằng cách hoàn thành phương thức `onCreate()`. Phương thức `onStart()` sẽ hiển thị một activity cho người dùng và việc thực thi phương thức này sẽ kết thúc rất nhanh chóng.

Sau đây là ví dụ về việc xác định phương thức `onStart()` trong hoạt động android.

```
@Override
protected void onStart()
{
    super.onStart()
}
```

Sau khi hoàn thành việc thực thi phương thức `onStart()`, activity chuyển sang trạng thái Resumed và hệ thống gọi phương thức `onResume()`.

### 3.1.3.3 Phương thức onResume

Khi một hoạt động được đưa vào trạng thái Resumed, hệ thống sẽ gọi lại phương thức callback `onResume()`. Ở trạng thái này, activity bắt đầu tương tác với người dùng có nghĩa là người dùng có thể thấy phần chức năng và thiết kế của một ứng dụng trên một màn hình.

Phần lớn chức năng cốt lõi của một ứng dụng được triển khai trong phương thức `onResume()`.

Ứng dụng sẽ ở trạng thái Resumed này cho đến khi một activity khác xảy ra làm mất sự focus khỏi ứng dụng như nhận cuộc gọi điện thoại hoặc màn hình tắt, v...v...

Trong trường hợp nếu có bất kỳ sự kiện gián đoạn nào xảy ra ở trạng thái Resumed, Activity sẽ chuyển sang trạng thái Paused và hệ thống sẽ gọi phương thức `onPause()`.

Sau khi một activity được trả về từ trạng thái Paused sang trạng thái Resumed, hệ thống sẽ gọi lại phương thức onResume() do đó chúng ta cần triển khai phương thức onResume() để khởi tạo các thành phần mà chúng ta phát hành trong phương thức onPause()

Dưới đây là ví dụ về việc xác định phương thức onResume() trong hoạt động android.

```
@Override
public void onResume() {
    super.onResume();
    if (mCamera == null) {
        initializeCamera();
    }
}
```

Nếu bất kỳ gián đoạn nào xảy ra ở trạng thái Resumed, hactivity sẽ chuyển sang trạng thái Paused và hệ thống sẽ gọi phương thức onPause().

### 3.1.3.4 Phương thức onPause

Bất cứ khi nào người dùng rời khỏi một activity hoặc activity hiện tại đang bị Paused thì hệ thống sẽ gọi phương thức onPause(). Phương thức onPause() được sử dụng để tạm dừng các activity như dừng phát nhạc khi hoạt động ở trạng thái tạm dừng hoặc chuyển một activity trong khi chuyển từ ứng dụng này sang ứng dụng khác vì mỗi lần chỉ có thể focus vào một ứng dụng.

Dưới đây là ví dụ về việc xác định phương thức onPause() trong activity của android.

```
@Override
public void onPause() {
    super.onPause();
    if (mCamera != null) {
        mCamera.release();
        mCamera = null;
    }
}
```

Sau khi hoàn thành việc thực thi phương thức onPause(), phương thức tiếp theo là onStop() hoặc onResume() tùy thuộc vào điều gì xảy ra sau khi một activity được đưa vào trạng thái Paused.

### 3.1.3.5 Phương thức onStop

Hệ thống sẽ gọi phương thức callback onStop() khi một activity không còn hiển thị cho người dùng nữa, activity sẽ chuyển sang trạng thái Stopped. Điều này xảy ra do activity hiện tại được chuyển sang trạng thái Resumed hoặc activity mới khởi chạy bao phủ toàn bộ màn hình hoặc activity đó đã bị phá hủy (destroyed).

Phương thức onStop() hữu ích để giải phóng tất cả các tài nguyên ứng dụng không còn cần thiết cho người dùng.

Sau đây là ví dụ về việc xác định phương thức onStop() trong activity của android.

```
@Override  
protected void onStop()  
{  
    super.onStop();  
}
```

Phương thức callback tiếp theo được hệ thống đưa ra là onRestart(), trong trường hợp nếu activity quay lại để tương tác với người dùng hoặc onDestroy(), trong trường hợp nếu activity đã chạy xong.

### 3.1.3.6 Phương thức onRestart

Hệ thống sẽ gọi phương thức onRestart() khi một activity tự khởi động lại sau khi dừng nó. Phương thức onRestart() sẽ khôi phục trạng thái activity từ thời điểm đang bị dừng.

Phương thức callback onRestart() trong activity của android sẽ luôn được theo sau bởi phương thức onStart().

### 3.1.3.7 Phương thức onDestroy

Hệ thống sẽ gọi phương thức onDestroy() trước khi một activity bị hủy và đây là phương thức callback cuối cùng mà activity của android nhận được.

Hệ thống sẽ gọi phương thức callback onDestroy() này hoặc activity đang kết thúc hoặc hệ thống hủy activity để tiết kiệm không gian bộ nhớ, tài nguyên.

Dưới đây là ví dụ về việc xác định phương thức `onDestroy()` trong activity của android.

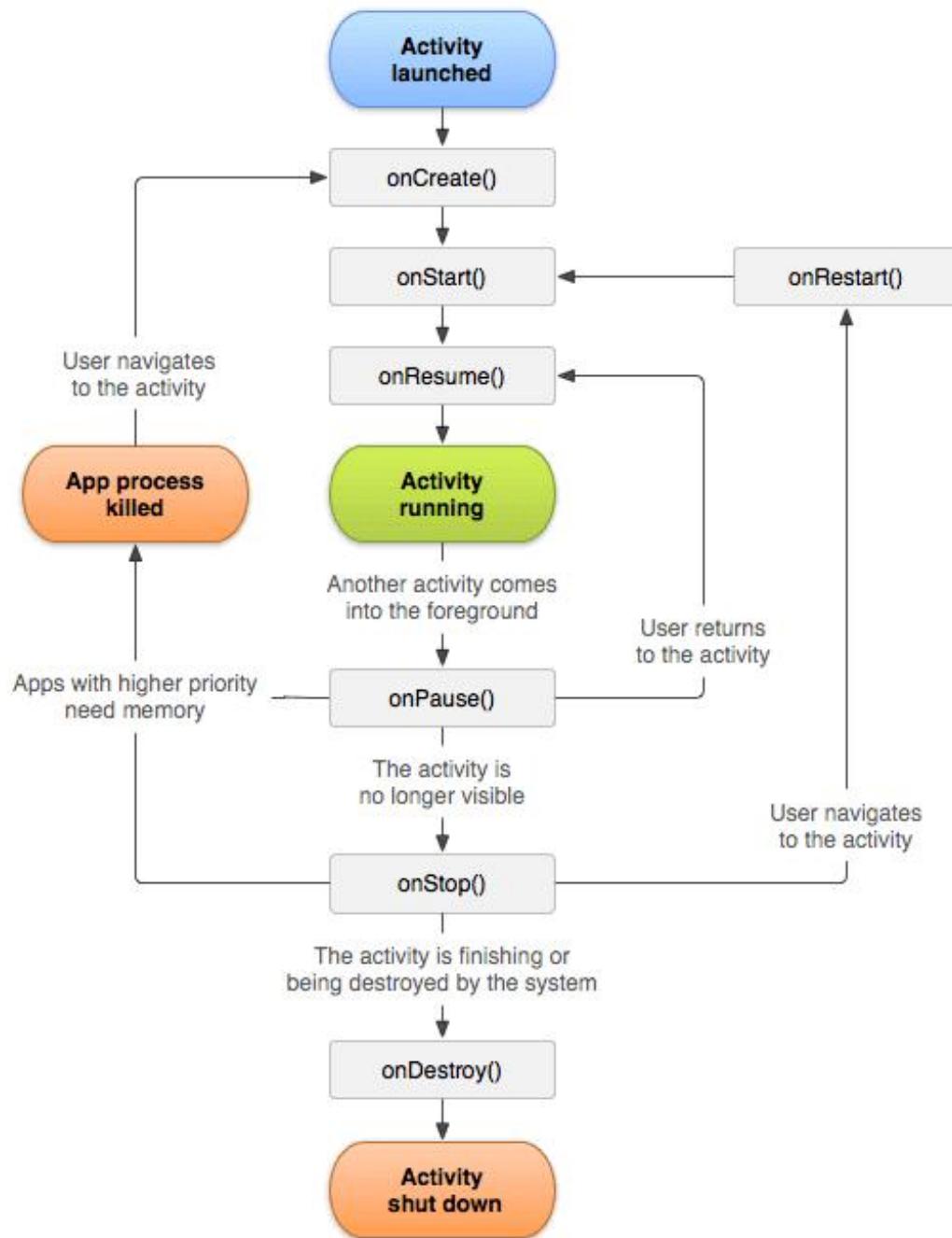
```
@Override  
public void onDestroy()  
{  
    super.onDestroy();  
}
```

Phương thức `onDestroy()` sẽ giải phóng tất cả các tài nguyên không được giải phóng bởi phương thức callback `onStop()` trước đó.

### 3.1.4 Sơ đồ chu kỳ sống của Android Activity

Nói chung, trong lớp activity android sử dụng các phương thức callback khác nhau như `onCreate()`, `onStart()`, `onPause()`, `onRestart()`, `onResume()`, `onStop()` và `onDestroy()` để trải qua các giai đoạn khác nhau của vòng đời (life cycle) của một activity.

Sau đây là phần trình bày bằng hình ảnh về Vòng đời của một activity trong Android cho thấy activity sẽ hoạt động như thế nào trong các giai đoạn khác nhau bằng cách sử dụng các phương thức callback.



Bất cứ khi nào người dùng cổ gắng rời khỏi một activity như chuyển từ ứng dụng này sang ứng dụng khác, hệ thống sẽ sử dụng các phương thức callback để loại bỏ hoàn toàn hoặc một phần activity đó để tiếp tục activity từ nơi người dùng đã dừng lại.

Dựa trên các đòi hỏi của chúng ta, chúng ta có thể triển khai activity trong ứng dụng android bằng phương thức callback và không cần thiết phải sử dụng tất cả các phương thức callback trong mỗi ứng dụng android.

### 3.1.5 Ví dụ về Android Activity Lifecycle

Bây giờ chúng ta sẽ xem, vòng đời activity của android sẽ hoạt động như thế nào bằng một ví dụ. Sau đây là ví dụ về việc gọi các phương thức callback để gọi một activity để xem quá trình vòng đời của activity trong ứng dụng Android.

Ở đây, chúng ta sẽ sử dụng ví dụ về Ứng dụng Android Hello World đã tạo trước đó và thực hiện một số sửa đổi đối với tệp MainActivity.java như hình dưới đây để nắm bắt quy trình vòng đời hoạt động của activity trong Android.

#### 3.1.5.1 Code trong tập tin MainActivity.java

Sau đây là các sửa đổi code được thực hiện để bao gồm tất cả các phương thức callback của vòng đời trong tệp MainActivity.java

```
package com.tutlane.helloworld;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d("Activity Lifecycle","onCreate invoked");
    }
    @Override
    protected void onStart() {
        super.onStart();
        Log.d("Activity Lifecycle","onStart invoked");
    }
}
```

```

@Override
protected void onResume() {
    super.onResume();
    Log.d("Activity Lifecycle","onResume invoked");
}
@Override
protected void onPause() {
    super.onPause();
    Log.d("Activity Lifecycle","onPause invoked");
}
@Override
protected void onStop() {
    super.onStop();
    Log.d("Activity Lifecycle","onStop invoked");
}
@Override
protected void onRestart() {
    super.onRestart();
    Log.d("Activity Lifecycle","onRestart invoked");
}
@Override
protected void onDestroy() {
    super.onDestroy();
    Log.d("Activity Lifecycle","onDestroy invoked");
}
}
)

```

Nếu bạn quan sát đoạn mã trên, chúng ta đã xác định một câu lệnh như "setContentView (R.layout.activity\_main);" sẽ giúp tải tất cả các thành phần giao diện người dùng được xác định trong tệp activity\_main.xml và sử dụng phương thức Log.d() để tạo thông báo nhật ký.

Trong ứng dụng của chúng ta, chúng ta có thể có nhiều hơn một tệp activity và chúng ta cần khai báo tất cả các activity trong tệp AndroidManifest.xml. Trong tệp manifest XML, bằng cách sử dụng thuộc tính danh mục MAIN action và LAUNCHER trong bộ các intent filter (<intent-filter>), chúng ta có thể đề cập đến activity chính mở ra khi người dùng khởi chạy ứng dụng của chúng ta lắn đầu với launcher icon. Trong trường hợp nếu chúng ta không đề cập đến MAIN action, thì hệ thống sẽ quyết định activity nào cần bắt đầu và nếu chúng ta không thêm danh mục LAUNCHER cho activity chính, biểu tượng ứng dụng (app icon) của chúng ta sẽ không xuất hiện trong danh sách ứng dụng của màn hình chính .

Code của tệp AndroidManifest.xml sẽ như hình dưới đây.

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

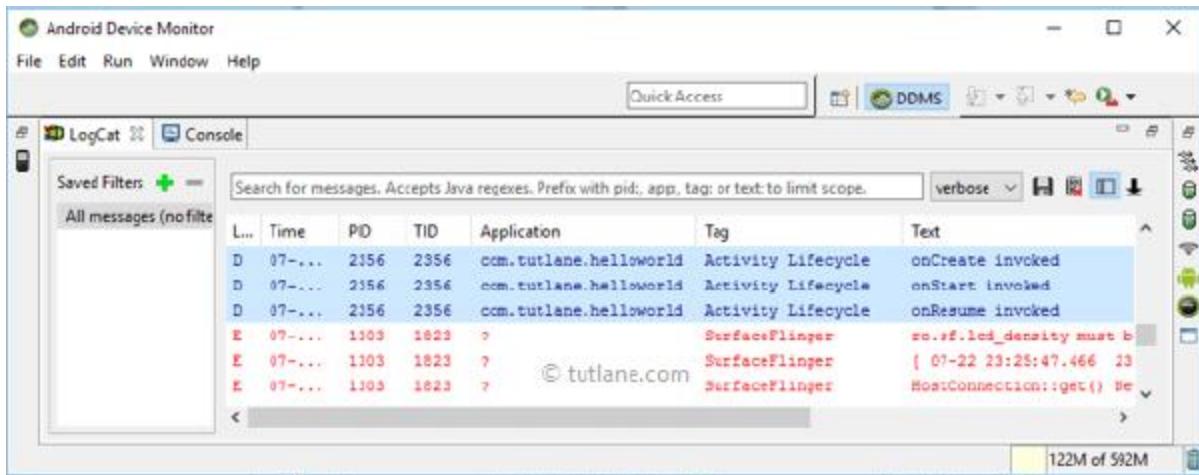
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
        <activity android:name=".MainActivity"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

### 3.1.5.2 Kết quả của ví dụ về Lifecycle

Bây giờ, hãy chạy ứng dụng của bạn bằng Thiết bị ảo Android (AVD) trong Android Studio bằng cách nhấp vào biểu tượng Play trên thanh công cụ hoặc nhấn Shift + F10. Kết quả ứng dụng của chúng ta sẽ như hình dưới đây.

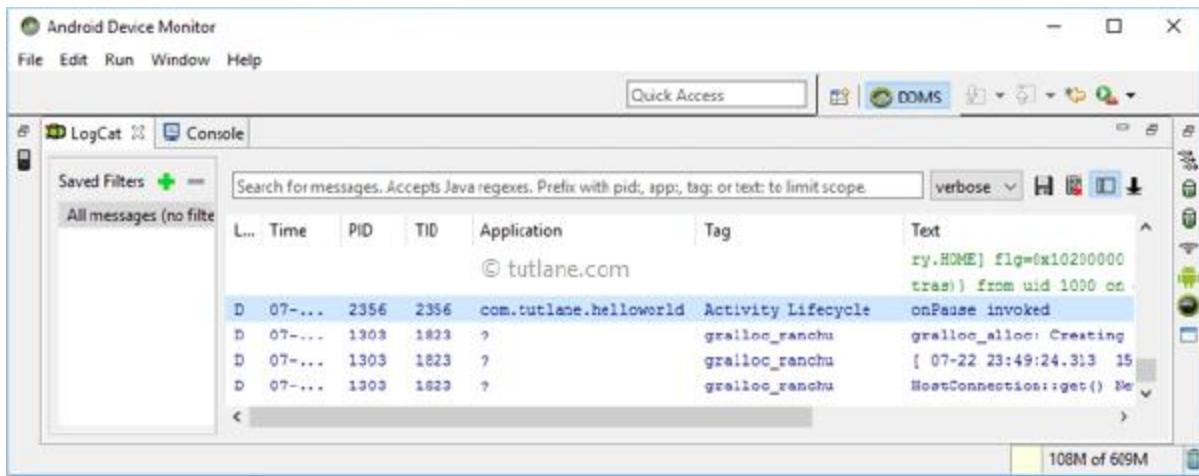


Bây giờ hãy mở trình theo dõi thiết bị Android (Android Device Monitor) để xem thông báo nhật ký của chúng ta trong cửa sổ LogCat trong Android Studio như hình dưới đây.

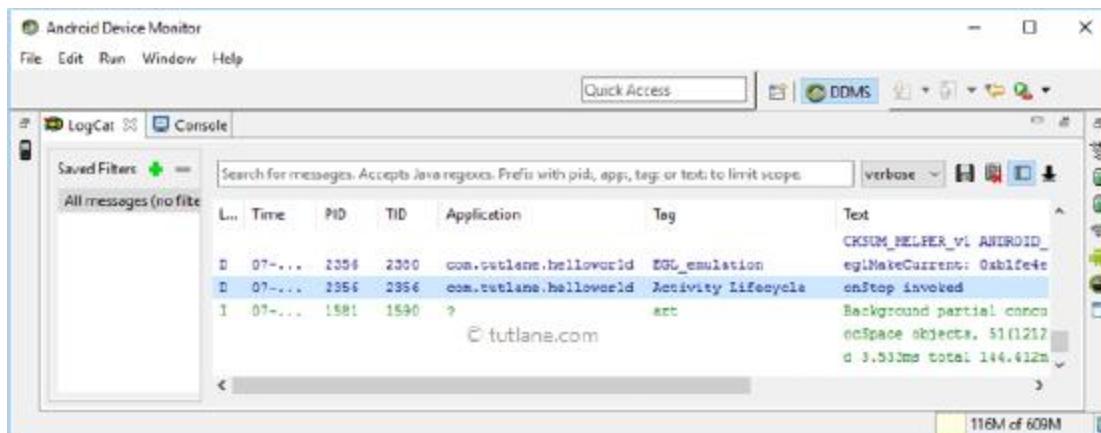


Nếu bạn quan sát các thông báo nhật ký trong cửa sổ LogCat onCreate, các phương thức onStart và onResume được hệ thống gọi ra.

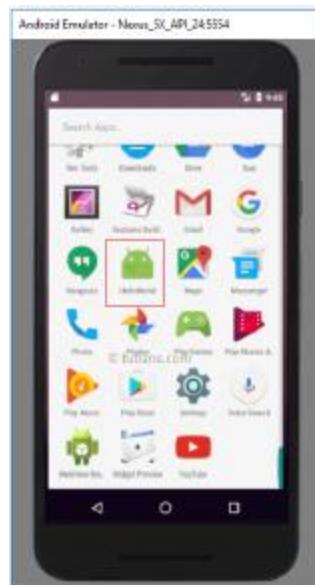
Bây giờ nhấn vào nút Home trong Android Emulator, ngay lập tức activity được đưa vào trạng thái Paused và hệ thống sẽ gọi ra phương thức onPause() như hình bên dưới.



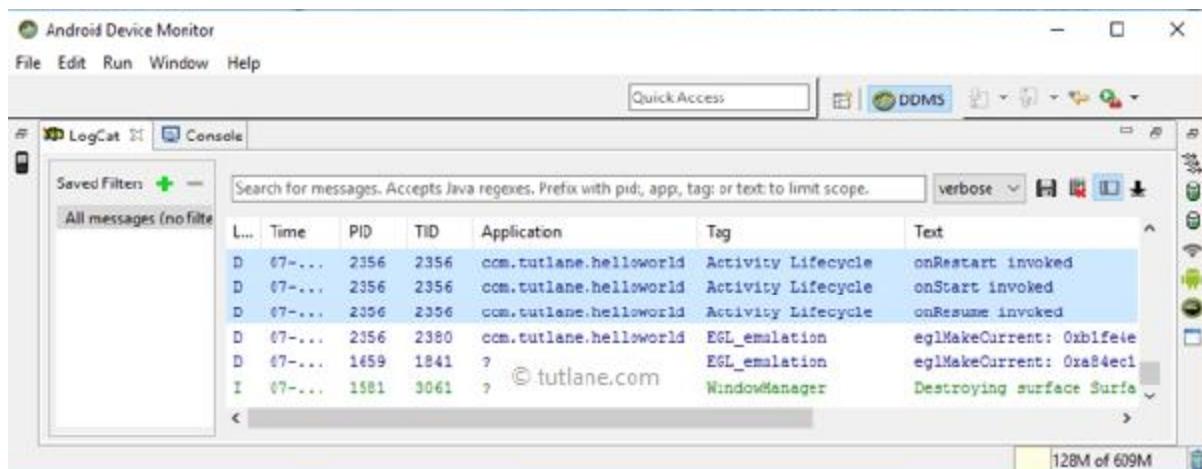
Sau một thời gian, hoạt động sẽ chuyển sang trạng thái Stopped và hệ thống sẽ gọi phương thức onStop() như hình dưới đây.



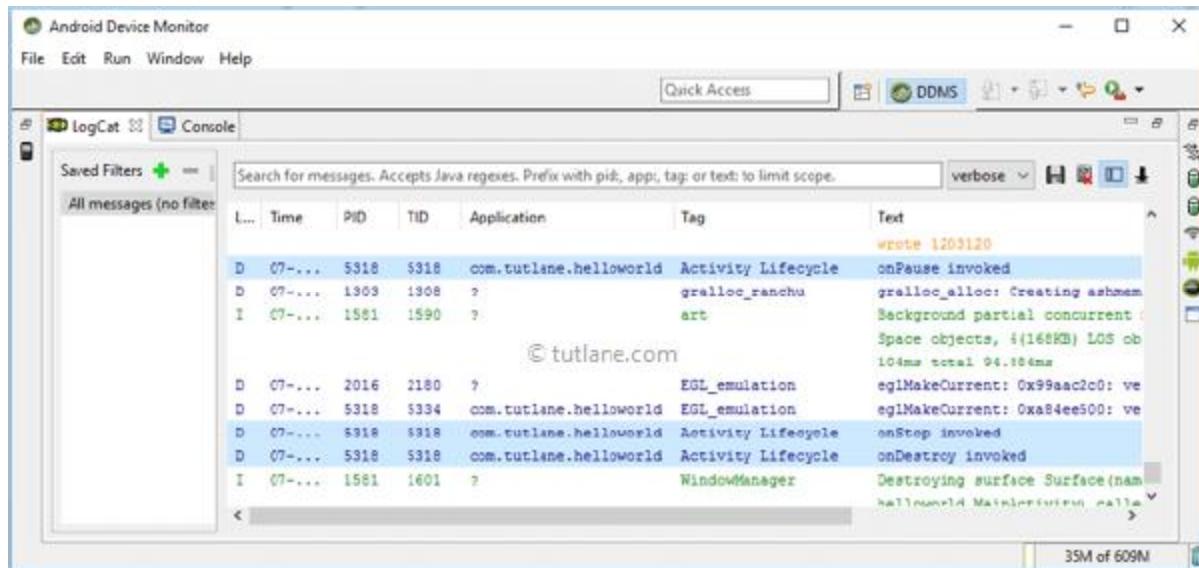
Bây giờ một lần nữa khởi chạy ứng dụng của chúng ta từ danh sách ứng dụng trên màn hình Home như hình dưới đây.



Nếu bạn quan sát các thông báo nhật ký trong cửa sổ LogCat lần nữa, các phương thức onRestart, onStart và onResume được hệ thống gọi ra như hình dưới đây.



Bây giờ bấm vào nút Back trong thiết bị giả lập Android, hệ thống sẽ gọi phương thức onPause và một lúc sau onStop, phương thức onDestroy sẽ được gọi như hình dưới đây.



Ở đây chúng ta cần nhớ rằng các phương thức onCreate và onDestroy sẽ chỉ gọi một lần trong suốt vòng đời hoạt động.

Đây là cách quy trình vòng đời hoạt động android sẽ gọi các phương thức khác nhau trong khi chuyển đổi từ giai đoạn này sang giai đoạn khác.

## 3.2 XỬ LÝ SỰ KIỆN TRONG ANDROID

### 3.2.1 Sự kiện là gì?

Sự kiện là một cách hữu ích để thu thập dữ liệu về tương tác của người dùng với các thành phần tương tác của Ứng dụng. Ví dụ như các lần nhấn nút hoặc chạm vào màn hình, v...v... Android framework duy trì hàng đợi sự kiện dựa theo hàng đợi queue dưới dạng cơ sở nhập trước, xuất trước (FIFO). Ta có thể nắm bắt các sự kiện này trong chương trình của mình và thực hiện hành động thích hợp theo yêu cầu.

Có ba khái niệm sau liên quan đến Quản lý sự kiện trên Android:

**Event Listeners** - (rình xử lý sự kiện) là một giao diện trong lớp View có chứa một phương thức callback duy nhất. Các phương thức này sẽ được gọi bởi Android

framework khi một View (đã được đăng ký listener) được kích hoạt bởi sự tương tác của người dùng với một thành phần trong giao diện người dùng.

**Event Listeners Registration** - Đăng ký sự kiện là quá trình mà một Event Handler được đăng ký với Event Listener để trình xử lý được gọi khi Event Listener kích hoạt sự kiện.

**Event Handlers** - Khi một sự kiện xảy ra và trước đó chúng ta đã đăng ký một listener cho sự kiện, Event listener sẽ gọi Event Handlers, đây mới là phương thức thực sự xử lý sự kiện.

### 3.2.2 Event Listeners và Event Handlers

Event Handler	Event Listener & sự mô tả
onClick()	<b>OnClickListener()</b> Hàm này được gọi khi người dùng nhấp hoặc chạm hoặc focus (tập trung) vào bất kỳ widget con nào như button, text, image, v...v... Ta sẽ sử dụng hàm event handler onClick() để xử lý sự kiện đó.
onLongClick()	<b>OnLongClickListener()</b> Hàm này được gọi khi người dùng nhấp hoặc chạm hoặc focus (tập trung) vào bất kỳ widget nào như button, text, image, v...v... trong một hoặc nhiều giây. Bạn sẽ sử dụng hàm event handler onLongClick () để xử lý sự kiện đó.
onFocusChange()	<b>OnFocusChangeListener()</b> Hàm này được gọi khi widget bị mất focus (mất tập trung) tức là người dùng thoát khỏi một view. Bạn sẽ sử dụng hàm event handler onFocusChange() để xử lý sự kiện đó.
onKey()	<b>OnFocusChangeListener()</b> Hàm này được gọi khi người dùng focus (tập trung) vào một view và nhấn hoặc nhả một phím phần cứng trên thiết bị. Bạn sẽ sử dụng hàm event handler onKey() để xử lý sự kiện đó.
onTouch()	<b>OnTouchListener()</b> Hàm được gọi khi người dùng nhấn phím, nhả phím hoặc bất kỳ movement gesture (hành vi cử chỉ chuyển động)

Event Handler	Event Listener & sự mô tả
	nào trên màn hình. Bạn sẽ sử dụng hàm event handler <code>onTouch()</code> để xử lý sự kiện đó.
<code>onMenuItemClick()</code>	<b>OnMenuItemClickListener()</b> Hàm này được gọi khi người dùng chọn một item (một mục) trên menu. Bạn sẽ sử dụng hàm event handler <code>onMenuItemClick()</code> để xử lý sự kiện đó.
<code>onCreateContextMenu()</code>	<b>onCreateContextMenuListener()</b> Hàm này được gọi khi menu ngữ cảnh (context menu) đang được tạo (do kết quả của một nhấp chuột dài liên tục).

Có nhiều event listeners khác có sẵn như một phần của lớp View như `OnHoverListener`, `OnDragListener`, v...v... có thể cần thiết cho ứng dụng của bạn. Vì vậy, khuyên bạn nên tham khảo tài liệu chính thức về phát triển ứng dụng Android trong trường hợp bạn định phát triển một ứng dụng phức tạp.

### 3.2.3 Đăng ký Event Listeners

Đăng ký sự kiện (Event Registration) là quá trình mà Event Handler được đăng ký với Event Listener để trình xử lý được gọi khi Event Listener kích hoạt sự kiện. Mặc dù có một số cách nghiêm ngặt để đăng ký event listener của bạn cho bất kỳ sự kiện nào, nhưng ở đây sẽ chỉ liệt kê 3 cách hàng đầu, trong đó bạn có thể sử dụng bất kỳ cách nào trong số đó tùy theo tình hình.

- Sử dụng lớp bên trong ẩn danh (Anonymous Inner Class)
- Lớp Activity thực hiện giao diện Listener.
- Sử dụng tệp Layout `activity_main.xml` để chỉ định event handler trực tiếp.

Các ví dụ chi tiết về cả ba trường hợp sẽ được đề cập ở các phần sau đây.

### 3.2.4 Chế độ cảm ứng và sự Focus

**Touch Mode** (chế độ cảm ứng): người dùng có thể tương tác với thiết bị của mình bằng cách sử dụng các phím hoặc nút phản ứng hoặc chạm vào màn hình. Chạm vào màn hình sẽ đưa thiết bị vào chế độ cảm ứng. Sau đó, người dùng có thể

tương tác với nó bằng cách chạm vào các nút ảo trên màn hình, hình ảnh, v.v. Bạn có thể kiểm tra xem thiết bị có ở chế độ cảm ứng hay không bằng cách gọi phương thức `isInTouchMode ()` của lớp View.

**Focus (tập trung):** một view hoặc một Widget thường được đánh dấu hoặc hiển thị con trỏ nhấp nháy khi nó đang được focus (tập trung). Điều này cho thấy rằng nó đã sẵn sàng chấp nhận input từ người dùng.

- `isFocusable ()` - nó trả về true hoặc false
- `isFocusableInTouchMode ()` - kiểm tra xem nếu một view có thể focus ở chế độ cảm ứng hay không. (Một view có thể được focus khi sử dụng phím phan cứng nhưng không thể focus khi thiết bị ở chế độ cảm ứng)

```
android:focusable="@+id/button_1"
```

### onTouchEvent()

```
public boolean onTouchEvent(MotionEvent event){
    switch(event.getAction()){
        case TOUCH_DOWN:
            Toast.makeText(this,"you have clicked down touch button",Toast.LENGTH_LONG).show();
            break;

        case TOUCH_UP:
            Toast.makeText(this,"you have clicked up touch button",Toast.LENGTH_LONG).show();
            break;

        case TOUCH_MOVE:
            Toast.makeText(this,"you have clicked move touch button",Toast.LENGTH_LONG).show();
            break;
    }
    return super.onTouchEvent(event);
}
```

### 3.2.5 Ví dụ về xử lý sự kiện

Đăng ký Event Listeners bằng cách sử dụng lớp bên trong ẩn danh (Anonymous Inner Class).

Ở đây, bạn sẽ tạo một triển khai ẩn danh của listener và sẽ hữu ích nếu mỗi lớp chỉ được áp dụng cho một điều khiển duy nhất và bạn có lợi thế để chuyển các đối số cho event handler. Trong cách tiếp cận này, các phương thức event handler có thể truy cập dữ liệu private của Activity. Không cần tham chiếu để gọi đến Activity.

Nhưng nếu bạn áp dụng handler cho nhiều hơn một điều khiển, bạn sẽ phải cắt và dán mã cho handler và nếu mã cho handler dài, điều đó làm cho mã khó bảo trì hơn.

Sau đây là các bước đơn giản để chỉ ra cách chúng ta sẽ sử dụng lớp Listener riêng biệt để đăng ký và bắt sự kiện click chuột. Cách tương tự, bạn có thể triển khai listener của mình cho bất kỳ loại sự kiện bắt buộc nào khác.

<b>Bước Mô tả</b>	
1	Bạn sẽ sử dụng Android studio IDE để tạo một ứng dụng Android và đặt tên nó là myapplication trong gói com.example.myapplication như được giải thích trong phần ví dụ về chương trình Hello World ở các phần trước.
2	Sửa đổi tệp src / MainActivity.java để thêm event listeners và handlers sự kiện nhấp chuột cho hai buttons được xác định.
3	Sửa đổi nội dung mặc định của tệp res/layout/activity_main.xml để bao gồm các controls trên giao diện.
4	Không cần khai báo hằng số chuỗi mặc định. Android studio giúp cho bạn vẽ các hằng số mặc định.
5	Chạy ứng dụng để khởi chạy Android emulator và xác minh kết quả của những thay đổi được thực hiện trong ứng dụng.

Sau đây là nội dung của tệp main activity đã sửa đổi

**src/com.example.myapplication/MainActivity.java.** Tệp này có thể bao gồm từng phương thức callback cơ bản của lifecycle.

```
package com.example.myapplication;

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

public class MainActivity extends ActionBarActivity {
    private ProgressDialog progress;
    Button b1,b2;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        progress = new ProgressDialog(this);

        b1=(Button)findViewById(R.id.button);
        b2=(Button)findViewById(R.id.button2);
        b1.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                TextView txtView = (TextView) findViewById(R.id.textView);
                txtView.setTextSize(25);
            }
        });

        b2.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                TextView txtView = (TextView) findViewById(R.id.textView);
                txtView.setTextSize(55);
            }
        });
    }
}
```

Sau đây sẽ là nội dung của tệp res/layout/activity\_main.xml. Biển abc là một logo.

```
<?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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Event Handling "
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:textSize="30dp"/>

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Tutorials point "
        android:textColor="#ff87ff09"
        android:textSize="30dp"
        android:layout_above="@+id/imageButton"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="40dp" />
```

```
<ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/imageButton"
    android:src="@drawable/abc"
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Small font"
    android:id="@+id/button"
    android:layout_below="@+id/imageButton"
    android:layout_centerHorizontal="true" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Large Font"
    android:id="@+id/button2"
    android:layout_below="@+id/button"
    android:layout_alignRight="@+id/button"
    android:layout_alignEnd="@+id/button" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    android:id="@+id/textView"
    android:layout_below="@+id/button2"
    android:layout_centerHorizontal="true"
    android:textSize="25dp" />

</RelativeLayout>
```

Sau đây sẽ là nội dung của res/values/string.xml để định nghĩa hai hằng số mới

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">myapplication</string>
</resources>
```

Sau đây là nội dung mặc định của AndroidManifest.xml

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

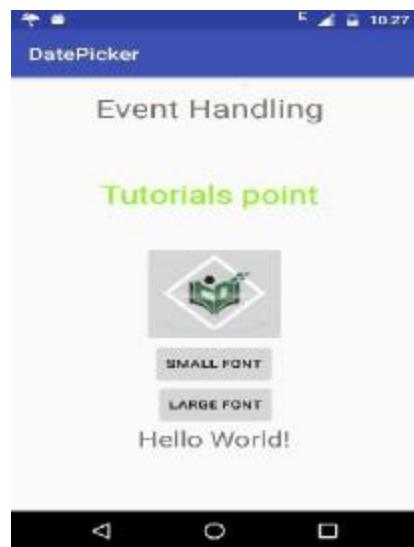
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name="com.example.myapplication.MainActivity"
            android:label="@string/app_name" >

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

        </activity>
    </application>
</manifest>
```

Hãy thử chạy ứng dụng myapplication của bạn. Giả sử bạn đã tạo AVD của mình trong khi thiết lập môi trường. Để chạy ứng dụng từ Android Studio, hãy mở một trong các tệp hoạt động của dự án của bạn và nhấp vào biểu tượng Play từ thanh công cụ. Android Studio cài đặt ứng dụng trên AVD của bạn và khởi động nó và nếu mọi thứ đều ổn với thiết lập và ứng dụng của bạn, nó sẽ hiển thị sau cửa sổ của Emulator.



Bây giờ bạn thử nhấp vào hai nút lần lượt và bạn sẽ thấy phông chữ của văn bản Hello World sẽ thay đổi, điều này xảy ra vì phương thức xử lý sự kiện nhấp chuột đã đăng ký đang được gọi đối với mỗi sự kiện nhấp chuột.

# TÓM TẮT

Trong bài học này giới thiệu thành phần cơ bản nhất trong ứng dụng Android, thành phần Activity. Đây là thành phần thường dùng nhiều nhất do có giao diện đồ họa tương tác với người dùng. Các đặc điểm chính của Activity như vòng đời hoạt động, cơ chế back stack được trình bày để người học hiểu rõ hơn về cơ chế hoạt động của Activity. Thông qua đó cách thức tạo ứng dụng có Activity được thể hiện cụ thể, người học lần lượt tìm hiểu cách quản lý và phát triển một ứng dụng Android Studio, cách thức tạo và sử dụng resource, cách thức xử lý các sự kiện của các thành phần view trong Activity.

## BÀI TẬP

**Câu 1:** Viết ứng dụng Android khảo sát vòng đời hoạt động của một Activity từ lúc tạo ra cho đến lúc kết thúc. Sử dụng thành phần Toast để hiển thị, các trạng thái của vòng đời của Activity này.

**Câu 2:** Viết ứng dụng Android có Activity có giao diện chứa một TextView có nội dung là "HUTECH". Tạo các button trong Activity như sau:

- Button 1: thiết lập kích thước font của TextView là 20
- Button 2: thiết lập Kích thước font của TextView là 40

Viết phần xử lý các button 1 và 2 theo cách Activity thực thi giao diện OnClickListener.

**Câu 3:** Viết ứng dụng Android có Activity với giao diện chứa một TextView có nội dung là "Khoa CNTT- HUTECH". Tạo các Button trong Activity như sau:

- Button 1: thiết lập màu văn bản của TextView là đỏ
- Button 2: thiết lập màu văn bản của TextView là xanh
- Button 3: thiết lập màu văn bản của TextView là vàng

Viết phần xử lý các Button theo cách khai báo hàm xử lý trong XML Layout và Activity không thực thi giao diện OnClickListener.

**Câu 4:** Viết ứng dụng Android có Activity với giao diện chứa một TextView có nội dung là "Số lần click: 0". Tạo một Button có xử lý là tăng số lần click mỗi khi click lên Button và cập nhật số lần click hiện tại lên TextView.

Viết phần xử lý button theo cách khai báo hàm xử lý trong lớp riêng thực thi giao diện OnClickListener.

# BÀI 4: THIẾT KẾ GIAO DIỆN

Học xong bài này người học cần nắm được các nội dung sau.

- *Khái niệm cơ bản về thành phần tổ chức giao diện trong Activity.*
- *Khái niệm View và ViewGroup trong việc tổ chức phân cấp giao diện.*
- *Cách thức xây dựng giao diện thông qua XML*
- *Hiểu các thuộc tính XML cơ bản trong thiết kế giao diện*
- *Các dạng tổ chức giao diện trong XML Layout.*
- *Hiểu và sử dụng tốt các thành phần layout trong việc xây dựng giao diện*

## **4.1 GIỚI THIỆU LAYOUT VÀ VIEW CỦA ANDROID**

### **4.1.1 Android View and ViewGroup**

Trong android, Layout được sử dụng để định nghĩa giao diện người dùng cho một ứng dụng hoặc activity và nó sẽ chứa các phần tử giao diện người dùng (UI) sẽ xuất hiện cho người dùng.

Giao diện người dùng trong ứng dụng android được tạo bằng một tập hợp các đối tượng View và ViewGroup. Nói chung, các ứng dụng android sẽ chứa một hoặc nhiều activity và mỗi activity là một màn hình ứng dụng. Các activity sẽ chứa nhiều thành phần giao diện người dùng và các thành phần giao diện người dùng đó là các thể hiện của các lớp con View và ViewGroup.

### **4.1.2 Các View của Android**

View là một lớp cơ sở cho tất cả các thành phần giao diện người dùng trong Android. Ví dụ: lớp EditText được sử dụng để chấp nhận input từ người dùng nhập vào trong các ứng dụng android, là lớp con của View.

Sau đây là một số lớp con View phổ biến sẽ được sử dụng trong các ứng dụng Android.

- TextView
- EditText
- Button
- CheckBox
- RadioButton
- ImageButton
- Progress Bar
- Spinner
- vân... vân...

Như vậy, chúng ta có nhiều lớp con View có sẵn trong android.

### 4.1.3 Android ViewGroup

Ví dụ: Linear Layout là ViewGroup có chứa các control giao diện người dùng như button, textview, v...v... và các layout khác cũng có.

Sau đây là các lớp con ViewGroup thường được sử dụng trong các ứng dụng Android.

- Linear Layout
- Relative Layout
- Table Layout
- Frame Layout
- Web View
- List View
- Grid View

Cả hai lớp con View và ViewGroup cùng nhau sẽ đóng vai trò quan trọng để tạo bố cục trong các ứng dụng Android.

#### 4.1.4 Các loại layout trong Android

Các loại layout gồm có: Linear, Relative, Frame, Table, ListView, GridView, WebView

Trong android, layout được sử dụng để xác định giao diện người dùng cho một ứng dụng hoặc hoạt động và nó sẽ chứa các phần tử giao diện người dùng sẽ xuất hiện cho người dùng.

Giao diện người dùng trong ứng dụng android được tạo với một bộ sưu tập các đối tượng View và ViewGroup. Nói chung, các ứng dụng android sẽ chứa một hoặc nhiều hoạt động và mỗi hoạt động là một màn hình của ứng dụng. Các hoạt động sẽ chứa nhiều thành phần giao diện người dùng và các thành phần giao diện người dùng đó là các thể hiện của các lớp con View và ViewGroup.

View là lớp cơ sở cho tất cả các thành phần UI trong android và nó được sử dụng để tạo các thành phần UI (user interface) tương tác như TextView, EditText, Checkbox, Radio Button,... và nó chịu trách nhiệm xử lý sự kiện và vẽ.

ViewGroup là một lớp con của View và nó sẽ hoạt động như một lớp cơ sở cho các bố cục và tham số bố cục. ViewGroup sẽ cung cấp một vùng chứa vô hình để chứa các View hoặc ViewGroup khác và để xác định các thuộc tính bố cục.

Trong android, chúng ta có thể xác định bố cục theo hai cách, đó là:

1. Khai báo các phần tử giao diện (UI Elements) người dùng trong XML.
2. Khởi tạo các phần tử layout trong thời gian chạy (runtime).

Android framework sẽ cho phép chúng ta sử dụng một trong hai hoặc cả hai phương pháp này để xây dựng giao diện người dùng của ứng dụng của chúng ta.

#### 4.1.5 Khai báo các UI Element trong XML

Trong android, chúng ta có thể tạo các bố cục giống như các trang web trong HTML bằng cách sử dụng các View và ViewGroup mặc định trong tệp XML. Tệp layout chỉ được chứa một phần tử gốc, phần tử này phải là một đối tượng View hoặc ViewGroup. Sau khi chúng ta xác định phần tử gốc, sau đó chúng ta có thể thêm các

đối tượng bô cục hoặc widget bô sung làm phần tử con để xây dựng cấu trúc View phân cấp xác xây dựng nên khung sườn của ứng dụng.

Sau đây là ví dụ về xây dựng layout trong tệp XML (activity\_main.xml) bằng cách sử dụng LinearLayout để chứa TextView, EditText và Button.

```
<?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">
    <TextView
        android:id="@+id/fstTxt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Enter Name"
        />
    <EditText
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10">
    </EditText>
    <Button
        android:id="@+id/getName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Get Name" />
</LinearLayout>
```

Chúng ta cần tạo một tệp layout trong thư mục của project là /res/layout, sau đó chỉ các tệp layout mới được biên dịch đúng.

#### 4.1.6 Nạp tệp tin XML Layout từ một Activity

Khi chúng ta đã hoàn tất việc tạo layout, chúng ta cần tải tài nguyên bô cục XML này bằng phương thức callback onCreate() từ activity của chúng ta như hình dưới đây.

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

Nếu bạn quan sát đoạn mã trên, chúng ta đang gọi layout của mình bằng cách sử dụng phương thức setContentView ở dạng R.layout.layout\_file\_name. Ở đây tên tệp

xml của chúng ta là activity\_main.xml vì vậy chúng ta đã sử dụng tên tệp là activity\_main.

Nói chung, trong quá trình khởi chạy activity của chúng ta, phương onCreate () sẽ được gọi bởi android framework để có được layout cần thiết cho một activity.

#### 4.1.7 Khởi tạo các Layout Element trong thời gian chạy

Nếu chúng ta muốn khởi tạo các phần tử layout trong thời gian chạy (runtime), chúng ta cần tạo các đối tượng View và ViewGroup tùy chỉnh của riêng mình theo lập trình với các bối cảnh bắt buộc.

Sau đây là ví dụ về cách tạo layout bằng cách sử dụng LinearLayout để chứa TextView, EditText và Button trong một activity sử dụng các đối tượng View và ViewGroup tùy chỉnh theo lập trình.

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView textView1 = new TextView(this);
        textView1.setText("Name:");
        EditText editText1 = new EditText(this);
        editText1.setText("Enter Name");
        Button button1 = new Button(this);
        button1.setText("Add Name");
        LinearLayout linearLayout = new LinearLayout(this);
        linearLayout.addView(textView1);
        linearLayout.addView(editText1);
        linearLayout.addView(button1);
        setContentView(linearLayout);
    }
}
```

Bằng cách tạo View xem tùy chỉnh và ViewGroup theo lập trình, chúng ta có thể xác định layout dựa trên yêu cầu của chúng ta trong các ứng dụng Android.

#### 4.1.8 Thuộc tính Width và Height

Khi chúng ta xây dựng một layout bằng cách sử dụng tệp XML, chúng ta cần đặt chiều rộng và chiều cao cho mọi phần tử View và ViewGroup bằng cách sử dụng các thuộc tính layout\_width và layout\_height.

Sau đây là ví dụ về thiết lập chiều rộng và chiều cao cho các phần tử View và ViewGroup trong tệp XML layout.

```
<?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">
    <TextView
        android:id="@+id/fstTxt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Enter Name" />
</LinearLayout>
```

Nếu bạn quan sát ví dụ trên, chúng ta đã sử dụng các giá trị khác nhau để đặt layout\_width và layout\_height, đó là

- match\_parent
- wrap\_content

Nếu chúng ta đặt giá trị match\_parent, thì View hoặc ViewGroup sẽ cố gắng khớp với chiều rộng hoặc chiều cao gốc.

Nếu chúng ta đặt giá trị wrap\_content, thì View hoặc ViewGroup sẽ cố gắng điều chỉnh chiều rộng hoặc chiều cao của nó dựa theo nội dung.

#### 4.1.9 Các thuộc tính của Layout

Trong android, giống như layout\_width và layout\_height, chúng ta có một loại thuộc tính khác có sẵn cho các đối tượng View và ViewGroup để xác định hình thức xuất hiện của layout dựa trên nhu cầu của chúng ta.

Sau đây là một số thuộc tính layout phổ biến được sử dụng trong ứng dụng android.

Thuộc tính	Mô tả
android:id	Dùng để đặt tên định danh cho một view và ViewGroups
android:layout_width	Dùng định nghĩa độ rộng cho các thành phần View và ViewGroup trong layout

Thuộc tính	Mô tả
android:layout_height	Dùng định nghĩa chiều cao cho các thành phần View và ViewGroup trong layout
android:layout_marginLeft	Dùng định nghĩa khoảng cách từ cạnh trái của View và ViewGroup đến cạnh tương ứng của layout đang chứa chúng
android:layout_marginRight	Dùng định nghĩa khoảng cách từ cạnh phải của View và ViewGroup đến cạnh tương ứng của layout đang chứa chúng
android:layout_marginTop	Dùng định nghĩa khoảng cách từ cạnh trên của View và ViewGroup đến cạnh tương ứng của layout đang chứa chúng
android:layout_marginBottom	Dùng định nghĩa khoảng cách từ cạnh dưới của View và ViewGroup đến cạnh tương ứng của layout đang chứa chúng
android:paddingLeft	Dùng định nghĩa khoảng cách từ cạnh trái của View và ViewGroup đến cạnh tương ứng của nội dung mà chúng đang chứa
android:paddingRight	Dùng định nghĩa khoảng cách từ cạnh phải của View và ViewGroup đến cạnh tương ứng của nội dung mà chúng đang chứa
android:paddingTop	Dùng định nghĩa khoảng cách từ cạnh trên của View và ViewGroup đến cạnh tương ứng của nội dung mà chúng đang chứa
android:paddingBottom	Dùng định nghĩa khoảng cách từ cạnh dưới của View và ViewGroup đến cạnh tương ứng của nội dung mà chúng đang chứa
android:layout_gravity	Dùng để xác định vị trí của các view con bên trong layout

## 4.2 CÁC LOẠI LAYOUT

Chúng ta có một số loại layout khác nhau có sẵn trong Android để triển khai giao diện người dùng cho các ứng dụng Android của chúng ta với các thiết kế khác nhau dựa trên yêu cầu của chúng ta.

Sau đây là các bộ cục (layout) thường được sử dụng trong các ứng dụng Android để thực hiện các thiết kế theo yêu cầu.

- Linear Layout

- Relative Layout
- Frame Layout
- Table Layout
- Web View
- List View
- Grid View

**Linear Layout:** trong android, LinearLayout là một lớp con ViewGroup được sử dụng để hiển thị lần lượt tất cả các View con theo hướng ngang (horizontal) hoặc hướng dọc (vertical) dựa trên thuộc tính hướng (orientation).

**Relative Layout:** trong android, RelativeLayout là một ViewGroup được sử dụng để chỉ định vị trí của các View con tương đối với nhau (Con A ở bên trái Con B) hoặc tương đối với cha (Căn chỉnh Aligned trên top của cha).

**Frame Layout:** trong android, FrameLayout là một lớp con ViewGroup được sử dụng để chỉ định vị trí của các View mà nó chứa phía trên top của nhau để chỉ hiển thị các View đơn bên trong FrameLayout.

**Table Layout:** trong android, TableLayout là một lớp con ViewGroup được sử dụng để hiển thị các phần tử View con theo các hàng và cột.

**Web View:** trong android, WebView là một trình duyệt được sử dụng để hiển thị các trang web như một phần của bối cảnh hoạt động của chúng ta.

**List View:** trong android, ListView là một ViewGroup được sử dụng để hiển thị danh sách các mục trong một cột đơn có thể cuộn được.

**Grid View:** trong android, GridView là một ViewGroup được sử dụng để hiển thị các mục trong một lưới có thể cuộn gồm các cột và hàng.

## 4.2.1 LinearLayout

Trong android, LinearLayout là lớp con ViewGroup được sử dụng để hiển thị lần lượt tất cả các View con theo hướng ngang (horizontal) hoặc hướng dọc (vertical) dựa trên thuộc tính hướng (orientation).

Trong android, chúng ta có thể chỉ định hướng linear layout bằng cách sử dụng thuộc tính android: orientation.

Sau đây là biểu diễn bằng hình ảnh của linear layout trong các ứng dụng Android.



Horizontal



Vertical

Trong LinearLayout, các View con được sắp xếp từng cái một nên danh sách ngang sẽ chỉ có một hàng gồm nhiều cột và danh sách dọc sẽ có một cột gồm nhiều hàng.

#### 4.2.1.1 Khai báo LinearLayout

Sau đây là cách chúng ta căn định nghĩa LinearLayout trong các ứng dụng Android.

```
<?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" >
    <!-- Add Child Views Here -->
</LinearLayout>
```

Nếu bạn quan sát đoạn mã ở trên, thì ở đây chúng ta đã xác định hướng là chiều dọc, vì vậy điều này sẽ căn chỉnh tất cả các layout con và các view con của nó theo chiều dọc.

### 4.2.1.2 Ví dụ về LinearLayout

Sau đây là ví dụ về việc tạo LinearLayout với các control khác nhau trong ứng dụng Android.

Tạo một ứng dụng android mới bằng cách sử dụng Android Studio và đặt tên là LinearLayout.

Bây giờ, hãy mở tệp activity\_main.xml từ đường dẫn \res\layout và viết code như hình dưới đây.

#### activity\_main.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:paddingLeft="20dp"
    android:paddingRight="20dp"
    android:orientation="vertical" >
    <EditText
        android:id="@+id/txtTo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="To"/>
    <EditText
        android:id="@+id/txtSub"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Subject"/>
    <EditText
        android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="Message"/>
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="Send"/>
</LinearLayout>
```

Sau khi hoàn tất việc tạo layout, chúng ta cần tải tài nguyên XML layout từ phương thức onCreate() của activity, mở tệp main activity MainActivity.java từ đường dẫn \java\com.tutlane.linearLayout và viết code như như hình bên dưới.

## MainActivity.java

```
package com.tutlane.linearLayout;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
  
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
}
```

Nếu bạn quan sát đoạn mã trên, chúng ta đang gọi layout của mình bằng cách sử dụng phương thức `setContentView` ở dạng `R.layout.layout_file_name`. Ở đây tên tệp xml của chúng ta là `activity_main.xml` vì vậy chúng ta đã sử dụng tên tệp là `activity_main`.

Nói chung, trong quá trình khởi chạy activity của chúng ta, phương thức `onCreate()` sẽ được gọi bởi android framework để có được layout cần thiết cho một activity.

### Kết quả của ví dụ **LinearLayout**

Khi chạy ví dụ trên bằng thiết bị ảo android (AVD), chúng ta sẽ nhận được kết quả như hình bên dưới.



### Thuộc tính weight của layout

Nếu bạn quan sát ví dụ trên, chúng ta đã sử dụng thuộc tính layout weight (android: layout\_weight) trong View con. Trên thực tế, thuộc tính này được các View con sử dụng để chỉ định lượng không gian View sẽ chiếm trên màn hình. Nếu chúng ta gán giá trị trọng số lớn hơn cho View con, thì nó sẽ mở rộng để lấp đầy bất kỳ khoảng trống nào còn lại trong View cha.

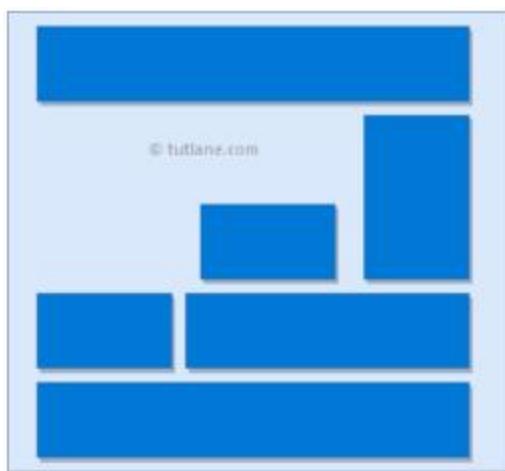
Nếu bạn quan sát ví dụ trên, chúng ta đã sử dụng ba text field và chúng ta chỉ định giá trị trọng số (weight) cho một text field. Hai text field không có trọng số (weight) sẽ chỉ chiếm diện tích cần thiết cho nội dung của nó và text field khác có giá trị trọng số (weight) sẽ mở rộng để lấp đầy khoảng trống còn lại sau khi không gian đã cung cấp cho text field.

Đây là cách chúng ta có thể sử dụng LinearLayout trong các ứng dụng android để hiển thị lần lượt tất cả các View theo hướng ngang hoặc hướng dọc dựa trên thuộc orientation.

### 4.2.2 RelativeLayout

Trong android, RelativeLayout là một ViewGroup được sử dụng để chỉ định vị trí của các thể hiện View con tương đối với nhau (con A ở bên trái con B) hoặc liên quan đến cha mẹ (căn chỉnh với top của cha mẹ).

Sau đây là biểu diễn bằng hình ảnh của relative layout trong các ứng dụng Android.



Trong android, RelativeLayout rất hữu ích để thiết kế giao diện người dùng vì bằng cách sử dụng bố cục tương đối, chúng ta có thể loại bỏ các nhóm chế độ xem lồng

nhau và giữ cho hệ thống phân cấp bố cục của chúng ta phẳng, giúp cải thiện hiệu suất của ứng dụng.

#### 4.2.2.1 Xác định vị trí của các Views trong Relative Layout

Như chúng ta đã thảo luận, trong RelativeLayout, chúng ta cần chỉ định vị trí của các view liên quan tương đối (relative) với nhau hoặc liên quan đến cha mẹ. Trong trường hợp nếu chúng ta không chỉ định vị trí của các View con, thì theo mặc định, tất cả các View con được đặt ở trên cùng bên trái của layout.

Sau đây là một số thuộc tính layout hữu ích nhất có sẵn cho các dạng View trong RelativeLayout.

Thuộc tính	Mô tả
layout_alignParentTop	Nếu nó chỉ định "true", cạnh trên cùng của view sẽ canh lề với cạnh trên của cha.
layout_alignParentBottom	Nếu nó chỉ định "true", cạnh trên cùng của view sẽ canh lề với cạnh dưới của cha.
layout_alignParentLeft	Nếu nó chỉ định "true", cạnh trên cùng của view sẽ canh lề với cạnh trái của cha.
layout_alignParentRight	Nếu nó chỉ định "true", cạnh trên cùng của view sẽ canh lề với cạnh phải của cha.
layout_centerInParent	Nếu nó chỉ định "true", cạnh trên cùng của view sẽ canh lề ở giữa của cha.
layout_centerHorizontal	Nếu nó chỉ định "true", cạnh trên cùng của view sẽ canh giữa theo chiều ngang của cha.
layout_centerVertical	Nếu nó chỉ định "true", cạnh trên cùng của view sẽ canh giữa theo chiều dọc của cha.
layout_above	Nằm trên một view được chỉ định bởi id.
layout_below	Nằm dưới một view được chỉ định bởi id.
layout_toLeftOf	Nằm bên trái một view được chỉ định bởi id.
layout_toRightOf	Nằm bên phải một view được chỉ định bởi id.
layout_toStartOf	Nằm ở vị trí bắt đầu của một view được chỉ định bởi id.
layout_toEndOf	Nằm ở vị trí cuối của một view được chỉ định bởi id.

### 4.2.2.2 Ví dụ về RelativeLayout

Sau đây là ví dụ về việc tạo RelativeLayout với các control khác nhau trong ứng dụng Android.

Tạo một ứng dụng android mới bằng cách sử dụng Android Studio và đặt tên là RelativeLayout.

Bây giờ, hãy mở tệp activity\_main.xml từ đường dẫn \res\layout và viết code như hình dưới đây.

#### activity\_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="10dp"
    android:paddingRight="10dp">
    <Button
        android:id="@+id	btn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:text="Button1" />
    <Button
        android:id="@+id	btn2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"
        android:text="Button2" />
    <Button
        android:id="@+id	btn3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_centerVertical="true"
        android:text="Button3" />
```

```

<Button
    android:id="@+id	btn4"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:text="Button4" />
<Button
    android:id="@+id	btn5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBottom="@+id	btn2"
    android:layout_centerHorizontal="true"
    android:text="Button5" />
<Button
    android:id="@+id	btn6"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id	btn4"
    android:layout_centerHorizontal="true"
    android:text="Button6" />
<Button
    android:id="@+id	btn7"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_toEndOf="@+id	btn1"
    android:layout_toRightOf="@+id	btn1"
    android:layout_alignParentRight="true"
    android:text="Button7" />
</RelativeLayout>

```

Sau khi hoàn tất việc tạo layout, chúng ta cần tải tài nguyên XML layout từ phương thức `onCreate()` của chúng ta, cho tệp mở tệp main activity tên là `MainActivity.java` từ đường dẫn `\java\com.tutlane.relativelayout` và viết code như hình bên dưới.

### MainActivity.java

```

package com.tutlane.linearLayout;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}

```

Nếu bạn quan sát đoạn mã trên, chúng ta đang gọi layout của mình bằng cách sử dụng phương thức `setContentView` ở dạng `R.layout.layout_file_name`. Ở đây tên tệp xml của chúng ta là `activity_main.xml` vì vậy chúng ta đã sử dụng tên tệp là `activity_main`.

Nói chung, trong quá trình khởi chạy activity của chúng ta, phương thức `onCreate()` sẽ được gọi bởi android framework để có được layout cần thiết cho một activity.

### Kết quả của ví dụ **RelativeLayout**

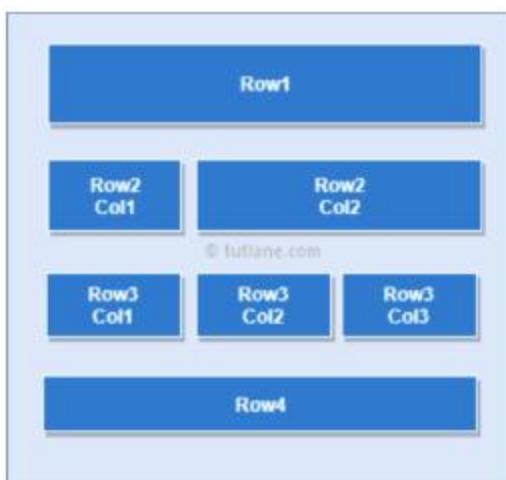
Khi chạy ví dụ trên bằng thiết bị ảo android (AVD), chúng ta sẽ nhận được kết quả như hình bên dưới. Đây là cách chúng ta có thể sử dụng `RelativeLayout` trong các ứng dụng Android dựa trên yêu cầu của chúng ta.



### 4.2.3 TableLayout

Trong android, `TableLayout` là một lớp con `ViewGroup` được sử dụng để hiển thị các phần tử View con trong các hàng và cột.

Sau đây là biểu diễn bằng hình ảnh của `TableLayout` trong các ứng dụng Android.



Trong android, TableLayout sẽ định vị các phần tử con của nó thành các hàng và cột và nó sẽ không hiển thị bất kỳ đường viền nào cho các hàng, cột hoặc ô.

TableLayout trong android sẽ hoạt động giống như table trong HTML và table sẽ có nhiều cột như hàng có nhiều ô nhất. TableLayout có thể được giải thích là <table> và TableRow giống như phần tử <tr> trong HTML.

#### 4.2.3.1 Ví dụ TableLayout

Sau đây là ví dụ về việc tạo TableLayout với các control khác nhau trong ứng dụng Android.

Tạo một ứng dụng android mới bằng cách sử dụng Android Studio và đặt tên là TableLayout.

Bây giờ, hãy mở tệp activity\_main.xml từ đường dẫn \res\layout và viết code như hình dưới đây

##### activity\_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="100dp"
    android:paddingLeft="10dp"
    android:paddingRight="10dp" >
    <TableRow android:background="#0079D6" android:padding="5dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="UserId" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="User Name" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Location" />
    </TableRow>
```

```
<TableRow android:background="#DAE8FC" android:padding="5dp">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="1" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Suresh Dasari" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Hyderabad" />
</TableRow>
```

```
<TableRow android:background="#DAE8FC" android:padding="5dp">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="2" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Rohini Alavala" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Guntur" />
</TableRow>
```

```
<TableRow android:background="#DAE8FC" android:padding="5dp">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="3" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Trishika Dasari" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Guntur" />
</TableRow>
</TableLayout>
```

Sau khi hoàn tất việc tạo layout, chúng ta cần tải tài nguyên XML layout từ phương thức `onCreate()` của chúng ta, mở tệp main activity có tên là `MainActivity.java` từ đường dẫn `\java\com.tutlane.tablelayout` và viết code như như hình bên dưới.

### MainActivity.java

```
package com.tutlane.linearLayout;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Nếu bạn quan sát đoạn code trên, chúng ta đang gọi layout của mình bằng phương thức `setContentView` ở dạng `R.layout.layout_file_name`. Ở đây tên tệp xml của chúng ta là `activity_main.xml` vì vậy chúng ta đã sử dụng tên tệp là `activity_main`.

Nói chung, trong quá trình khởi chạy activity của chúng ta, phương thức `onCreate()` sẽ được gọi bởi android framework để có được layout cần thiết cho một activity.

### Kết quả của ví dụ TableLayout

Khi chạy ví dụ trên bằng thiết bị ảo android (AVD), chúng ta sẽ nhận được kết quả như hình bên dưới. Đây là cách chúng ta có thể sử dụng Table Layout trong các ứng dụng Android dựa trên yêu cầu của chúng ta.

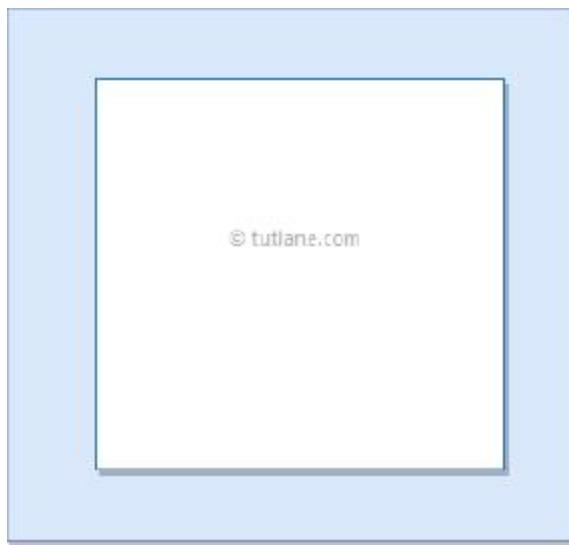


## 4.2.4 Frame Layout

Trong android, Framelayout là một lớp con ViewGroup được sử dụng để chỉ định vị trí của các View mà nó chứa phía trên của nhau để hiển View bên trong FrameLayout.

Nói một cách đơn giản, có thể nói FrameLayout được thiết kế để chặn một vùng trên màn hình để hiển thị một mục duy nhất.

Sau đây là mô tả bằng hình ảnh của frame layout trong các ứng dụng Android.



Trong android, FrameLayout sẽ hoạt động như một trình giữ chỗ (placeholder) trên màn hình và nó được sử dụng để giữ một View con duy nhất.

Trong FrameLayout, các View con được thêm vào một ngăn xếp (stack) và View con được thêm gần nhất sẽ hiển thị ở trên cùng. Chúng ta có thể thêm nhiều dạng View con vào FrameLayout và kiểm soát vị trí của chúng bằng cách sử dụng các thuộc tính gravity trong FrameLayout.

### 4.2.4.1 Ví dụ FrameLayout

Sau đây là ví dụ về việc tạo FrameLayout với các control khác nhau trong ứng dụng Android.

Tạo một ứng dụng android mới bằng cách sử dụng Android Studio và đặt tên là FrameLayout.

Bây giờ, hãy mở tệp activity\_main.xml từ đường dẫn \res\layout và viết code như hình dưới đây.

## activity\_main.xml

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <ImageView
        android:id="@+id/imgvw1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scaleType="centerCrop"
        android:src="@drawable/flimg" />
    <TextView
        android:id="@+id/txtvw1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp"
        android:background="#4C374A"
        android:padding="10dp"
        android:text="Grand Palace, Bangkok"
        android:textColor="#FFFFFF"
        android:textSize="20sp" />
    <TextView
        android:id="@+id/txtvw2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right|bottom"
        android:background="#AA000000"
        android:padding="10dp"
        android:text="21/Aug/2017"
        android:textColor="#FFFFFF"
        android:textSize="18sp" />
</FrameLayout>
```

Nếu bạn quan sát đoạn code trên, chúng ta đã sử dụng ImageView để hiển thị hình ảnh (flimg) từ thư mục drawable trong framelayou. Vì vậy, hãy thêm hình ảnh của bạn vào thư mục drawable và thay thế đường dẫn @drawable/flimg bằng đường dẫn hình ảnh của bạn.

Sau khi hoàn tất việc tạo layout, chúng ta cần tải tài nguyên XML layout từ phương thức onCreate () của chúng ta, mở tệp main activity tên là MainActivity.java từ đường dẫn \java\com.tutlane.framelayout và viết code như hình bên dưới.

## MainActivity.java

```
package com.tutlane.linearLayout;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Nếu bạn quan sát đoạn code trên, chúng ta đang gọi layout của mình bằng phương thức `setContentView` ở dạng `R.layout.layout_file_name`. Ở đây tên tệp xml của chúng ta là `activity_main.xml` vì vậy chúng ta đã sử dụng tên tệp là `activity_main`.

Nói chung, trong quá trình khởi chạy hactivity của chúng ta, phương thức `onCreate()` sẽ được gọi bởi android framwork để có được layout cần thiết cho một activity.

### Kết quả của ví dụ FrameLayout

Khi chạy ví dụ trên bằng thiết bị ảo android (AVD) chúng ta sẽ nhận được kết quả như hình bên dưới. Đây là cách chúng ta có thể sử dụng frame layout trong các ứng dụng Android dựa trên yêu cầu của chúng ta.



## 4.3 CÁC VIEWGROUP

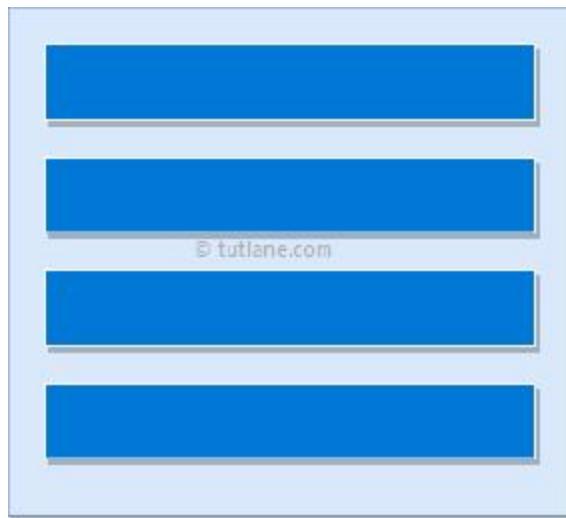
ViewGroup là một lớp con của View và nó sẽ hoạt động như một lớp cơ sở cho các bối cục và các tham số của bối cục. ViewGroup sẽ cung cấp một bộ chứa (container) vô hình để chứa các View hoặc ViewGroup khác và để khai báo các thuộc tính của bối cục.

### 4.3.1 ListView

Trong android, ListView là một ViewGroup được sử dụng để hiển thị list (danh sách) có thể cuộn của các item (mục) trong nhiều hàng và các item được tự động chèn vào danh sách bằng bộ adapter.

Nói chung, adapter lấy dữ liệu từ các nguồn như mảng hoặc cơ sở dữ liệu và chuyển đổi từng mục thành các View kết quả và các view được đưa vào danh sách thành các item.

Sau đây là đại diện bằng hình ảnh của listview trong các ứng dụng Android.



#### 4.3.1.1 Android Adapter

Trong android, Adapter sẽ hoạt động như một trung gian giữa các nguồn dữ liệu và các view của adapter như ListView, Gridview để điền dữ liệu vào các view của adapter. Bộ Adapter sẽ chứa dữ liệu và lấy từng mục dữ liệu trong tập dữ liệu để sinh ra các view cho mỗi item trong list view.

Nói chung, trong android, chúng ta có sẵn các loại bộ adapter khác nhau để tìm nạp dữ liệu từ các nguồn dữ liệu khác nhau để điền dữ liệu vào các view của adapter, đó là:

Adapter	Mô tả
ArrayAdapter	Input của nó là một mảng hay List
CurosrAdapter	Input của nó là Cursor.
SimpleAdapter	Input của nó là các dữ liệu tĩnh (static data) trong resources.
BaseAdapter	Đây là một triển khai chung cho cả ba loại adapter và nó có thể được sử dụng cho ListView, Gridview hoặc Spinners tùy theo nhu cầu của chúng ta

#### 4.3.1.2 Ví dụ về ListView

Dưới đây là ví dụ về cách tạo ListView bằng cách sử dụng arrayadapter trong ứng dụng android.

Tạo một ứng dụng android mới bằng cách sử dụng Android Studio và đặt tên là ListView.

Bây giờ, hãy mở tệp activity\_main.xml từ đường dẫn \res\layout và viết code như hình dưới đây.

##### activity\_main.xml

```
<?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">
    <ListView
        android:id="@+id/userlist"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
    </ListView>
</LinearLayout>
```

Khi chúng ta đã hoàn tất việc tạo layout, bây giờ chúng ta sẽ liên kết dữ liệu với ListView của mình bằng ArrayAdapter, cho tệp main activity mở MainActivity.java từ đường dẫn \java\com.tutlane.listview và viết code như hình dưới đây.

## MainActivity.java

```
package com.tutlane.listview;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;

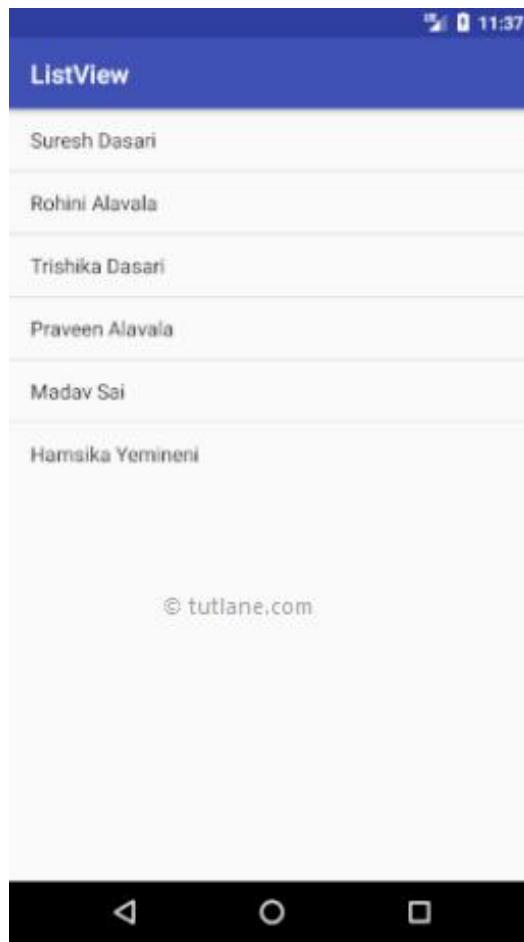
public class MainActivity extends AppCompatActivity {
    private ListView mListview;
    private ArrayAdapter aAdapter;
    private String[] users = { "Suresh Dasari", "Rohini Alavala", "Trishika
Dasari", "Praveen Alavala", "Madav Sai", "Hamsika Yemineni"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mListview = (ListView) findViewById(R.id.userlist);
        aAdapter = new ArrayAdapter(this, android.R.layout.simple_list_item_
1, users);
        mListview.setAdapter(aAdapter);
    }
}
```

Nếu bạn quan sát đoạn code trên, chúng ta đang liên kết chi tiết mảng tĩnh (users) với ListView bằng ArrayAdapter và gọi layout của chúng ta bằng phương thức setContentView ở dạng R.layout.layout\_file\_name. Ở đây tên tệp xml của chúng ta là activity\_main.xml vì vậy chúng ta đã sử dụng tên tệp là activity\_main.

Nói chung, trong quá trình khởi chạy activity của chúng ta, phương thức onCreate() sẽ được gọi bởi android framework để có được layout cần thiết cho một activity.

### Kết quả của ví dụ ListView

Khi chạy ví dụ trên bằng thiết bị ảo android (AVD) chúng ta sẽ nhận được kết quả như hình dưới đây. Đây là cách chúng ta có thể liên kết dữ liệu với ListView bằng ArrayAdapter trong các ứng dụng Android dựa trên yêu cầu của chúng ta.



#### 4.3.1.3 Ví dụ về ListView với Custom Adapter

Trong ví dụ trước, chúng ta đã học một cách đơn giản để liên kết dữ liệu với ListView bằng cách sử dụng ArrayAdapter trong ứng dụng android. Nay giờ chúng ta sẽ xem cách tạo Adapter tùy chỉnh của riêng mình (custom) và liên kết dữ liệu với ListView bằng một ví dụ.

Đối với điều này, chúng ta cần tạo lớp custom adapter của riêng mình bằng cách thừa kế lớp BaseAdapter và tạo một lớp sẽ chứa các tham số cho các item nằm theo dòng trong list.

Bây giờ, hãy tạo một ứng dụng android mới bằng cách sử dụng Android Studio và đặt tên là ListView. Mở tệp activity\_main.xml từ đường dẫn \res\layout và viết code như hình dưới đây.

## activity\_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <ListView
        android:id="@+id/user_list"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:dividerHeight="1dp" />
</LinearLayout>
```

Bây giờ chúng ta cần tạo một layout cho các item theo hàng của listview, nhấp chuột phải vào thư mục layouts → chọn New → tệp Layout resource → đặt tên là list\_row.xml và nhấn OK. Bây giờ mở tệp mới tạo (list\_row.xml) và viết code như hình dưới đây.

## list\_row.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="5dip" >
    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textStyle="bold"
        android:textSize="17dp" />
    <TextView
        android:id="@+id/designation"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/name"
        android:layout_marginTop="7dp"
        android:textColor="#343434"
        android:textSize="14dp" />
    <TextView
        android:id="@+id/location"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/designation"
        android:layout_alignBottom="@+id/designation"
        android:layout_alignParentRight="true"
        android:textColor="#343434"
        android:textSize="14dp" />
</RelativeLayout>
```

Bây giờ chúng ta cần tạo một lớp tùy chỉnh (custom class - ListItem.java) để đại diện cho mỗi hàng trong danh sách (list), nhấp chuột phải vào thư mục java à chọn New → Java Class → đặt tên là ListItem.java rồi nhấp OK. Mở tệp ListItem.java và viết code như hình dưới đây.

### **ListItem.java**

```
package com.tutlane.listview;
/**
 * Created by tutlane on 23-08-2017.
 */
public class ListItem {
    private String name;
    private String designation;
    private String location;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getDesignation() {
        return designation;
    }
    public void setDesignation(String designation) {
        this.designation = designation;
    }
    public String getLocation() {
        return location;
    }
    public void setLocation(String location) {
        this.location = location;
    }
}
```

Bây giờ chúng ta cần tạo một custom adapter. (CustomListAdapter.java) và cho nó thừa kế lớp BaseAdapter. Trong trường hợp nếu chúng ta đang cho thừa kế lớp của mình bằng cách sử dụng BaseAdapter, chúng ta cần ghi đè các phương thức sau từ lớp BaseAdapter.

Phương thức	Mô tả
getCount()	Trả về tổng số dòng có trong listview
getItem()	Nó được sử dụng để chỉ định dữ liệu đối tượng của mỗi item
getItemId()	Trả về id của mỗi item nằm trên dòng
getView()	Được sử dụng để trả về một view đại diện cho một dòng của ListView.

Để tạo custom adapter, nhấp chuột phải vào thư mục java → chọn New → Java Class → đặt tên là CustomListAdapter.java rồi nhấp vào OK.

Mở tệp CustomListAdapter.java và viết code như hình dưới đây.

### CustomListAdapter.java

```
package com.tutlane.listview;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;

import java.util.ArrayList;
/**
 * Created by tutlane on 23-08-2017.
 */
public class CustomListAdapter extends BaseAdapter {
    private ArrayList<ListItem> listData;
    private LayoutInflater layoutInflater;
    public CustomListAdapter(Context aContext, ArrayList<ListItem> listData) {
        this.listData = listData;
        layoutInflater = LayoutInflater.from(aContext);
    }
    @Override
    public int getCount() {
        return listData.size();
    }
    @Override
    public Object getItem(int position) {
        return listData.get(position);
    }
}
```

```

@Override
public long getItemId(int position) {
    return position;
}
public View getView(int position, View v, ViewGroup vg) {
    ViewHolder holder;
    if (v == null) {
        v = LayoutInflater.inflate(R.layout.list_row, null);
        holder = new ViewHolder();
        holder.uName = (TextView) v.findViewById(R.id.name);
        holder.uDesignation = (TextView) v.findViewById(R.id.designation)
    };
    holder.uLocation = (TextView) v.findViewById(R.id.location);
    v.setTag(holder);
} else {
    holder = (ViewHolder) v.getTag();
}
holder.uName.setText(listData.get(position).getName());
holder.uDesignation.setText(listData.get(position).getDesignation());
holder.uLocation.setText(listData.get(position).getLocation());
return v;
}
static class ViewHolder {
    TextView uName;
    TextView uDesignation;
    TextView uLocation;
}
}

```

Nếu bạn quan sát lớp trên, chúng ta đang cho custom adapter của mình thừa kế giao diện BaseAdapter và chúng ta ghi đè tất cả các phương thức BaseAdapter trong bộ custom adapter của chúng ta.

Bây giờ chúng ta cần kết hợp tất cả custom classes trong tệp main activity (MainActivity.java) để liên kết dữ liệu với listview của chúng ta.

Mở tệp hoạt động chính (MainActivity.java) và viết code như hình dưới đây.

### **MainActivity.java**

```
package com.tutlane.listview;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.Toast;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ArrayList userList = getListData();
        final ListView lv = (ListView) findViewById(R.id.user_list);
        lv.setAdapter(new CustomListAdapter(this, userList));
        lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> a, View v, int position,
            long id) {
                ListItem user = (ListItem) lv.getItemAtPosition(position);
                Toast.makeText(MainActivity.this, "Selected :" + " " + user.
                getName() + ", " + user.getLocation(), Toast.LENGTH_SHORT).show();
            }
        });
    }

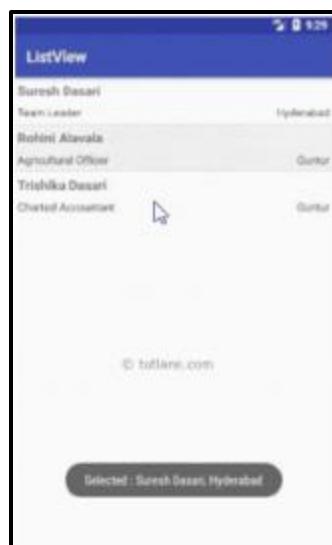
    private ArrayList getListData() {
        ArrayList<ListItem> results = new ArrayList<>();
        ListItem user1 = new ListItem();
        user1.setName("Suresh Dasari");
        user1.setDesignation("Team Leader");
        user1.setLocation("Hyderabad");
        results.add(user1);
        ListItem user2 = new ListItem();
        user2.setName("Rohini Alavala");
        user2.setDesignation("Agricultural Officer");
        user2.setLocation("Guntur");
        results.add(user2);
        ListItem user3 = new ListItem();
        user3.setName("Trishika Dasari");
        user3.setDesignation("Charted Accountant");
        user3.setLocation("Guntur");
        results.add(user3);
        return results;
    }
}
```

Nếu bạn quan sát đoạn code trên, chúng ta đang xây dựng và liên kết dữ liệu với ListView bằng cách sử dụng custom adapter của chúng ta và gọi layout của chúng ta bằng phương thức setContentView ở dạng R.layout.layout\_file\_name. Ở đây tên tệp xml của chúng ta là activity\_main.xml vì vậy chúng ta đã sử dụng tên tệp là activity\_main.

Nói chung, trong quá trình khởi chạy activity của chúng ta, phương thức onCreate() sẽ được gọi bởi android framework để có được layout cần thiết cho một activity.

### Kết quả của ví dụ Custom ListView

Khi chạy ví dụ trên bằng thiết bị ảo android (AVD) chúng ta sẽ nhận được kết quả như hình bên dưới.



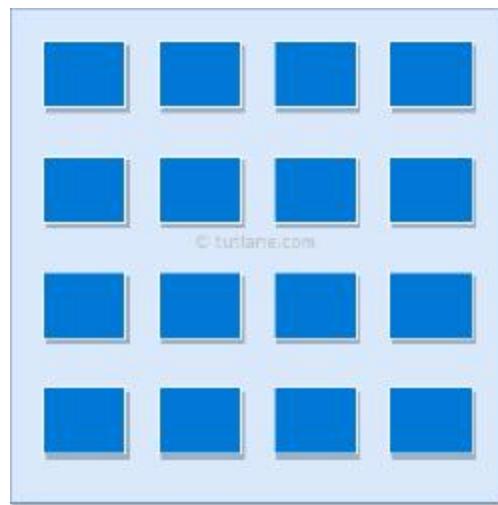
Đây là cách chúng ta có thể liên kết dữ liệu với ListView bằng cách sử dụng custom adapter trong các ứng dụng Android tùy theo yêu cầu của chúng ta.

## 4.3.2 GridView

Trong android, Grid View là một ViewGroup được sử dụng để hiển thị các mục trong lưới hai chiều, lưới có thể cuộn và các grid item được tự động chèn vào bối cảnh gridview bằng list adapter.

Nói chung, adapter lấy dữ liệu từ các nguồn như mảng hoặc cơ sở dữ liệu và chuyển đổi từng mục thành view kết quả và được đưa vào danh sách.

Sau đây là mô tả bằng hình ảnh của GridView trong các ứng dụng Android.



## Adapter

Trong android, Adapter sẽ hoạt động như một trung gian giữa các nguồn dữ liệu và adapter views như ListView, Gridview để điền dữ liệu vào adapter views. Adapter sẽ giữ dữ liệu và duyệt lần lượt các item trong tập dữ liệu để sinh ra các view cho từng item trong list.

Nói chung, trong android, chúng ta có sẵn các loại adapter khác nhau để tìm nạp dữ liệu từ các nguồn dữ liệu khác nhau để điền dữ liệu vào adapter views, đó là:

Adapter	Mô tả
ArrayAdapter	Input của nó là một mảng hay List
CursorAdapter	Input của nó là Cursor.
SimpleAdapter	Input của nó là các dữ liệu tĩnh (static data) trong resources.
BaseAdapter	Đây là một triển khai chung cho cả ba loại adapter và nó có thể được sử dụng cho ListView, Gridview hoặc Spinners tùy theo nhu cầu của chúng ta

### 4.3.2.1 Ví dụ về GridView

Dưới đây là ví dụ đơn giản hiển thị chi tiết người dùng bằng GridView và hiển thị vị trí của một hình ảnh cụ thể khi nhấp vào hình ảnh đó trong các ứng dụng Android.

Tạo một ứng dụng android mới bằng cách sử dụng Android Studio và đặt tên là GridView.

Sau khi chúng ta tạo một ứng dụng, hãy thêm một số hình ảnh mẫu vào thư mục project /res/drawable để hiển thị hình ảnh trong GridView.

Bây giờ, hãy mở tệp activity\_main.xml từ đường dẫn /res/layout và viết code như hình dưới đây.

### **activity\_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<GridView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gridview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnWidth="110dp"
    android:numColumns="auto_fit"
    android:verticalSpacing="10dp"
    android:horizontalSpacing="10dp"
    android:stretchMode="columnWidth"
    android:gravity="center" />
```

Khi chúng ta đã hoàn tất việc tạo layout, chúng ta cần tạo một custom adapter (ImageAdapter.java) bằng cách thừa kế từ lớp BaseExtender để hiển thị tất cả các item trong lưới (grid), nhấp chuột phải vào thư mục java --> Đặt tên là ImageAdapter.java và bấm OK.

Mở tệp ImageAdapter.java và viết code như hình dưới đây.

## ImageAdapter.java

```
package com.tutlane.gridview;
import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;
/**
 * Created by tutlane on 24-08-2017.
 */
public class ImageAdapter extends BaseAdapter {
    private Context mContext;
    public ImageAdapter(Context c) {
        mContext = c;
    }
    public int getCount() {
        return thumbImages.length;
    }
    public Object getItem(int position) {
        return null;
    }
    public long getItemId(int position) {
        return 0;
    }
    // create a new ImageView for each item referenced by the Adapter
    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView imageView = new ImageView(mContext);
        imageView.setLayoutParams(new GridView.LayoutParams(200, 200));
        imageView.setScaleType(ImageView.ScaleType.CENTER_CROP);
        imageView.setPadding(8, 8, 8, 8);
        imageView.setImageResource(thumbImages[position]);
        return imageView;
    }
    // Add all our images to arraylist
    public Integer[] thumbImages = {
        R.drawable.img1, R.drawable.img2,
        R.drawable.img3, R.drawable.img4,
        R.drawable.img5, R.drawable.img6,
        R.drawable.img7, R.drawable.img8,
        R.drawable.img1, R.drawable.img2,
        R.drawable.img3, R.drawable.img4,
        R.drawable.img5, R.drawable.img6,
        R.drawable.img7, R.drawable.img8,
        R.drawable.img1, R.drawable.img2,
        R.drawable.img3, R.drawable.img4,
        R.drawable.img5
    };
}
```

Nếu bạn quan sát đoạn mã trên, chúng ta đã giới thiệu một số hình ảnh, thực sự đó là những hình ảnh mẫu mà chúng ta đã thêm vào thư mục /res/drawable.

Bây giờ, chúng ta sẽ liên kết hình ảnh với GridView bằng cách sử dụng custom adapter của chúng ta (ImageAdapter.java), mở tệp main activity tên là MainActivity.java từ đường dẫn \java\com.tutlane\gridview và viết code như hình bên dưới.

### MainActivity.java

```
package com.tutlane\gridview
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.GridView;
import android.widget.Toast;

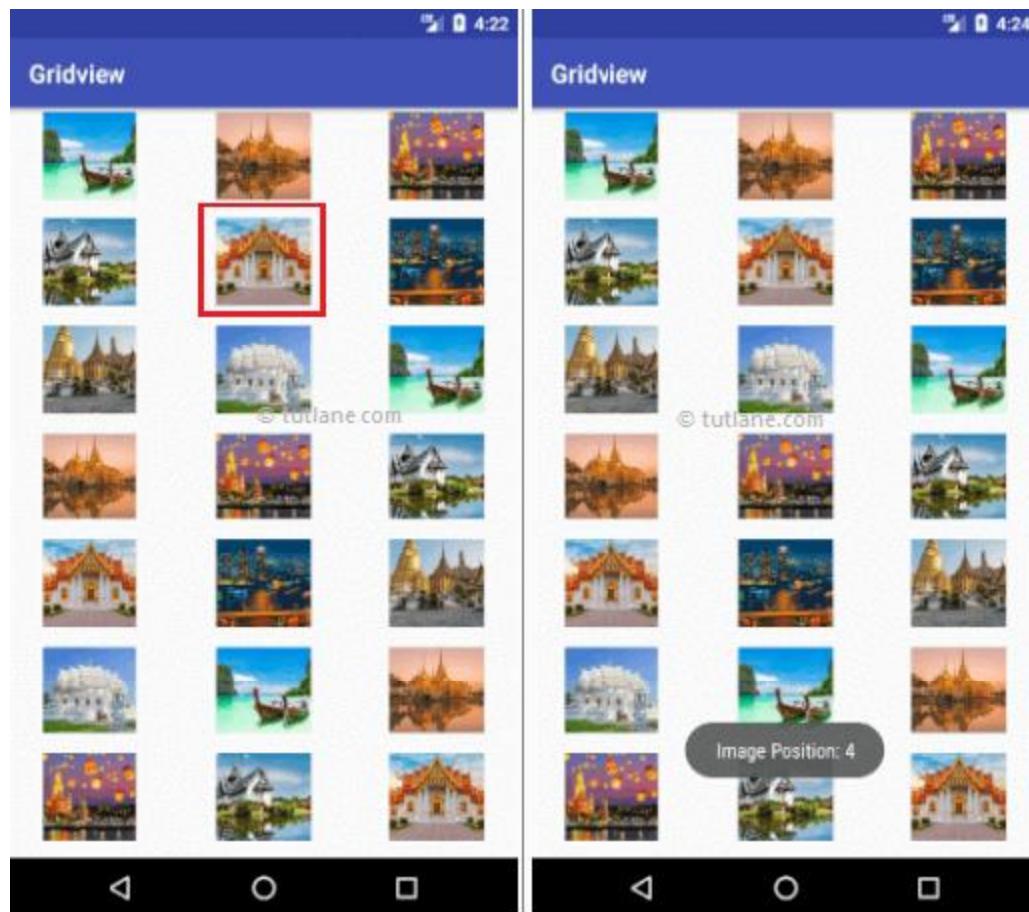
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        GridView gv = (GridView) findViewById(R.id.gvDetails);
        gv.setAdapter(new ImageAdapter(this));
        gv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            public void onItemClick(AdapterView<?> parent, View v, int position, long id) {
                Toast.makeText(MainActivity.this, "Image Position: " + position, Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

Nếu bạn quan sát thấy code trên, chúng ta đang liên kết các chi tiết hình ảnh với GridView bằng cách sử dụng custom adapter của chúng ta (ImageAdapter.java) và gọi layout của chúng ta bằng phương thức setContentView ở dạng R.layout.layout\_file\_name. Ở đây tên tệp xml của chúng ta là activity\_main.xml vì vậy chúng tôi đã sử dụng tên tệp là activity\_main.

Nói chung, trong quá trình khởi chạy activity của chúng ta, phương thức onCreate() sẽ được gọi bởi android framework để có được layout cần thiết cho một activity.

### Kết quả của ví dụ GridView

Khi chạy ví dụ trên bằng thiết bị ảo android (AVD), chúng ta sẽ nhận được kết quả như hình bên dưới.



Đây là cách chúng ta có thể liên kết hình ảnh với GridView bằng Adapter trong các ứng dụng Android dựa trên yêu cầu của chúng ta.

#### 4.3.2.2 Ví dụ về GridView Details

Trong ví dụ trên, chúng ta đã triển khai một thư viện hình ảnh bằng cách sử dụng gridView trong ứng dụng Android. Bây giờ chúng ta sẽ mở rộng chức năng của ví dụ trên để hiển thị hình ảnh lướt đã chọn trên toàn màn hình.

Bây giờ chúng ta cần tạo một tệp layout mới (image\_details.xml) trong thư mục project /res/layout để hiển thị chi tiết hình ảnh, nhấp chuột phải vào thư mục layouts → chọn New → Layout resource file → Đặt tên file là image\_details.xml và bấm OK. Bây giờ mở tệp mới tạo (image\_details.xml) và viết mã như hình dưới đây.

**image\_details.xml**

```
<?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">
    <ImageView android:id="@+id/full_image_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

Bây giờ chúng ta cần tạo một custom activity (FullImageActivity.java) để hiển thị chi tiết hình ảnh trong tệp layout mới tạo của chúng ta (image\_details.xml), nhấp chuột phải vào thư mục java và chọn New → Java Class → Đặt tên là FullImageActivity.java và bấm OK.

Mở tệp FullImageActivity.java và viết code như hình dưới đây.

**FullImageActivity.java**

```
package com.tutlane.gridview;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.widget.ImageView;

/**
 * Created by tutlane on 24-08-2017.
 */
public class FullImageActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.image_details);
        // Get intent data
        Intent i = getIntent();
        // Get Selected Image Id
        int position = i.getExtras().getInt("id");
        ImageAdapter imageAdapter = new ImageAdapter(this);
        ImageView imageView = (ImageView) findViewById(R.id.full_image_view);
        imageView.setImageResource(imageAdapter.thumbImages[position]);
    }
}
```

Bây giờ chúng ta cần đưa tệp activity mới tạo của mình (FullImageActivity.java) vào tệp AndroidManifest.xml như hình dưới đây. Đổi với điều đó, hãy mở tệp AndroidManifest.xml và viết code như hình dưới đây.

## AndroidManifest.xml

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

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

Bây giờ chúng ta cần sửa đổi chức năng nhấp vào gridview imange trong tệp main activity (MainActivity.java) để lấy chi tiết hình ảnh và hiển thị nó trong activity mới.

Mở tệp main activity (MainActivity.java) và viết code như hình dưới đây.

## MainActivity.java

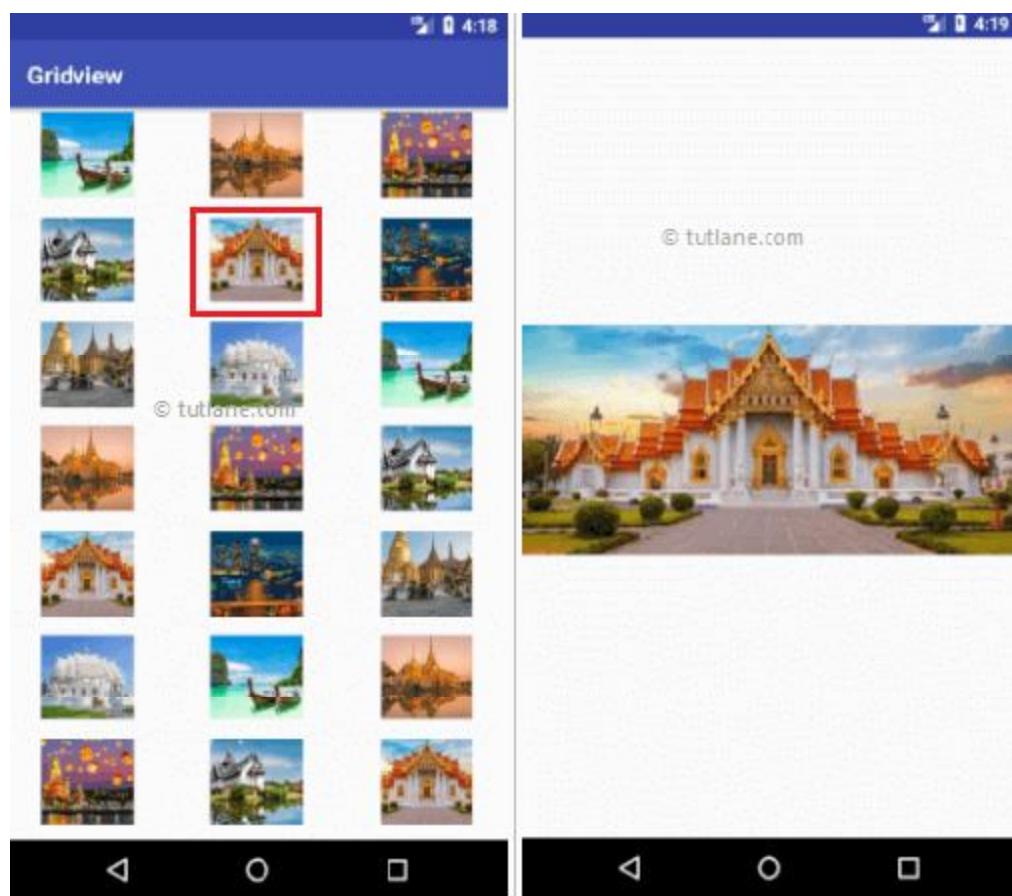
```
package com.tutlane.gridview;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.GridView;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        GridView gv = (GridView) findViewById(R.id.gvDetails);
        gv.setAdapter(new ImageAdapter(this));
        gv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            public void onItemClick(AdapterView<?> parent, View v, int position, long id) {
                // Sending image id to FullScreenActivity
                Intent i = new Intent(getApplicationContext(), FullImageActivity.class);
                // passing array index
                i.putExtra("id", position);
                startActivity(i);
            }
        });
    }
}
```

Nếu bạn quan sát đoạn code trên, chúng ta đang nhận các chi tiết hình ảnh đã chọn khi nhập vào hình ảnh và gửi các chi tiết đó đến tệp activity mới được tạo của chúng ta để hiển thị hình ảnh trên toàn màn hình.

### Kết quả của ví dụ GridView Details

Khi chạy ví dụ trên bằng thiết bị ảo android (AVD) chúng ta sẽ nhận được kết quả như hình dưới đây. Đây là cách chúng ta có thể xây dựng thư viện hình ảnh trong gridview và hiển thị hình ảnh đã chọn trong các ứng dụng Android dựa trên yêu cầu của chúng ta.

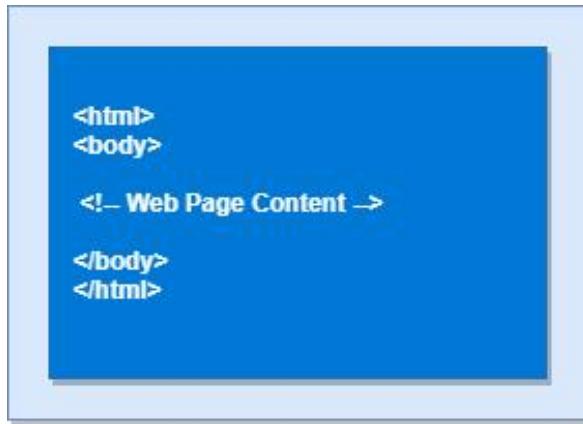


### 4.3.3 WebView

Trong android, WebView là một phần mở rộng của lớp View và nó được sử dụng để hiển thị nội dung trang web HTML tĩnh hoặc nội dung trang web từ xa có URL trong các ứng dụng android như một phần của layout hoạt động của chúng ta.

Nói chung, trong android, WebView sẽ hoạt động như một trình duyệt nhúng để đưa nội dung trang web vào layout hoạt động của chúng ta và nó sẽ không chứa bất kỳ tính năng nào của các trình duyệt thông thường, chẳng hạn như thanh địa chỉ, điều khiển điều hướng, v...v...

Sau đây là biểu diễn bằng hình ảnh của WebView trong các ứng dụng Android.



Nói chung, trong android WebView rất hữu ích để bao gồm và hiển thị nội dung của các trang web khác hoặc nội dung ứng dụng trong các trang được yêu cầu của chúng ta, chẳng hạn như thỏa thuận người dùng cuối (end-user), v...v...

### 4.3.3.1 Ví dụ WebView

Dưới đây là ví dụ về hiển thị nội dung HTML tĩnh trong WebView trong các ứng dụng Android.

Tạo một ứng dụng android mới bằng cách sử dụng Android studio và đặt tên là WebView.

Bây giờ, hãy mở tệp activity\_main.xml từ đường dẫn /res/layout và viết code như hình dưới đây

#### activity\_main.xml

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

Khi chúng ta đã hoàn tất việc thêm WebView vào layout, bây giờ chúng ta sẽ hiển thị nội dung HTML tĩnh trong WebView, cho tệp hoạt động chính đang mở MainActivity.java từ đường dẫn \java\com.tutlane.webview và viết code như hình dưới đây.

### MainActivity.java

```
package com.tutlane.webview;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.webkit.WebView;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        WebView wv = (WebView) findViewById(R.id.webview);
        String customHtml = "<html><body><h1>Welcome to Tutlane</h1>" +
            "<h2>Welcome to Tutlane</h2><h3>Welcome to Tutlane</h3>" +
            "<p>It's a Static Web HTML Content.</p>" +
            "</body></html>";
        wv.loadData(customHtml, "text/html", "UTF-8");
    }
}
```

Nếu bạn quan sát đoạn code trên, chúng ta đang gọi layout của mình bằng phương thức setContentView ở dạng R.layout.layout\_file\_name. Ở đây tên tệp xml của chúng ta là activity\_main.xml vì vậy chúng tôi đã sử dụng tên tệp là activity\_main và cố gắng hiển thị nội dung HTML tĩnh trong WebView.

Nói chung, trong quá trình khởi chạy activity của chúng ta, phương thức gọi lại onCreate() sẽ được gọi bởi khuôn khổ android framework để có được layout cần thiết cho một activity.

### Kết quả của ví dụ WebView

Khi chúng ta chạy ví dụ trên bằng thiết bị ảo Android (AVD), chúng ta sẽ nhận được kết quả như hình dưới đây. Đây là cách chúng ta có thể hiển thị nội dung HTML tĩnh bằng cách sử dụng WebView trong các ứng dụng android dựa trên yêu cầu của chúng ta.



### 4.3.3.2 Ví dụ về hiển thị nội dung của web trong Web View

Nói chung, trong Android WebView sẽ hoạt động như một trình duyệt nhúng để hiển thị nội dung trang web tĩnh hoặc từ xa trong các ứng dụng Android của chúng ta.

Bây giờ chúng ta sẽ xem cách tải nội dung URL từ xa trong WebView với ví dụ trong ứng dụng android.

Bằng cách sử dụng thuộc tính WebView LoadURL, chúng ta có thể tải nội dung URL từ xa trong các ứng dụng Android của mình. Để hiển thị nội dung URL từ xa trong webview, hãy sửa đổi mã tệp MainActivity.java như được hiển thị bên dưới.

#### MainActivity.java

```
package com.tutlane.webview;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.webkit.WebView;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        WebView webView = (WebView) findViewById(R.id.webview);
        webView.getSettings().setJavaScriptEnabled(true);
        webView.loadUrl("http://tutlane.com");
    }
}
```

Nếu bạn quan sát ví dụ trên, chúng ta đang cố gắng tải nội dung URL từ xa (<http://tutlane.com>) trong ứng dụng Android của chúng ta bằng cách sử dụng

WebView và chúng ta đặt thuộc tính `setJavaScriptEnabled()` để bật JavaScript vì JavaScript bị tắt trong WebView theo mặc định.

Nói chung, trang web mà chúng tôi đang tải trong WebView có thể sử dụng JavaScript. Trong trường hợp nếu chúng ta không bật JavaScript, thì chức năng liên quan đến JavaScript trong trang web sẽ không hoạt động, đó là lý do chúng tôi bật JavaScript bằng `setJavaScriptEnabled()`.

Để tải nội dung URL web từ xa, ứng dụng của chúng ta phải có quyền truy cập vào Internet. Chúng ta cần thiết lập quyền truy cập internet như hình bên dưới.

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

Bây giờ mở ứng dụng của chúng ta tệp `AndroidManifest.xml` trong thư mục `/manifests` và viết code như hình dưới đây.

### **AndroidManifest.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tutlane.webview">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-permission android:name="android.permission.INTERNET" />
</manifest>
```

Khi chúng ta đã hoàn tất các cài đặt cần thiết, bây giờ chúng ta sẽ chạy và xem kết quả của ứng dụng của chúng ta.

### Kết quả của ví dụ **WebView** hiển thị nội dung web

Khi chạy ví dụ trên bảng thiết bị ảo android (AVD) chúng ta sẽ nhận được kết quả như hình dưới đây.



Đây là cách chúng ta có thể hiển thị nội dung trang web URL từ xa bằng cách sử dụng **WebView** trong các ứng dụng android dựa trên yêu cầu của chúng ta.

### 4.3.4 ScrollView

**ScrollView** là một trường hợp đặc biệt của **FrameLayout**. Người sử dụng cuộn danh sách chứa các đối tượng View hoặc ViewGroup khi các đối tượng này có vùng không gian hiển thị lớn hơn so với màn hình hỗ trợ.

**ScrollView** chỉ có thể chứa duy nhất một đối tượng View hoặc ViewGroup (thường là **LinearLayout**). Lưu ý là không dùng chung **ListView** cùng với **ScrollView** do **ListView** được thiết kế để hiển thị một danh sách các đối tượng có liên hệ với nhau và đã tối ưu hóa để phù hợp với danh sách lớn.

Ví dụ sau minh họa **ScrollView** chứa một **LinearLayout**:

```
<?xml version="1.0" encoding="utf-8"?>  
<ScrollView  
    android:id="@+id/widget54"  
    android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android" >
<LinearLayout
    android:layout_width="310px"
    android:layout_height="wrap_content"
    android:orientation="vertical" >
    <Button
        android:id="@+id/button1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 1" />
    <Button
        android:id="@+id/button2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 2" />
    <Button
        android:id="@+id/button3"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 3" />
<EditText
    android:id="@+id/txt"
    android:layout_width="fill_parent"
    android:layout_height="300px" />
<Button
```

```
    android:id="@+id/button4"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:text="Button 4" />  
  
<Button  
    android:id="@+id/button5"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:text="Button 5" />  
  
</LinearLayout>  
  
</ScrollView>
```

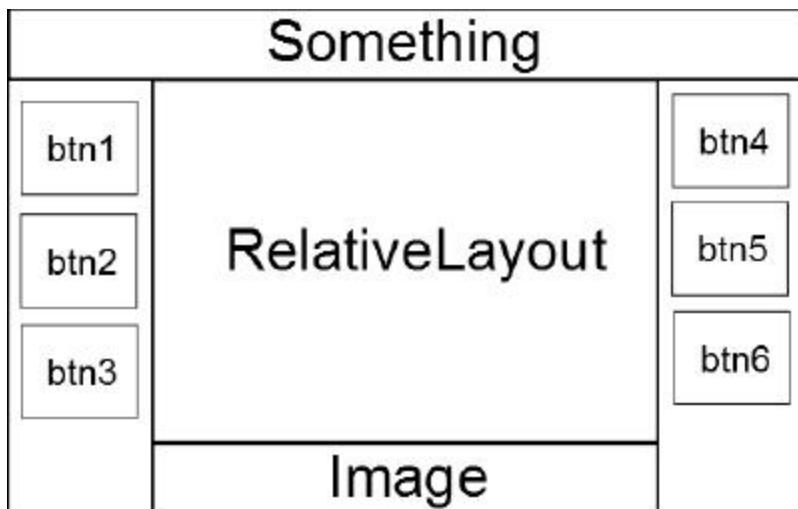
Khi thực thi ứng dụng thì người dùng có thể dùng tay để vuốt trên màn hình theo chiều từ dưới lên.

## TÓM TẮT

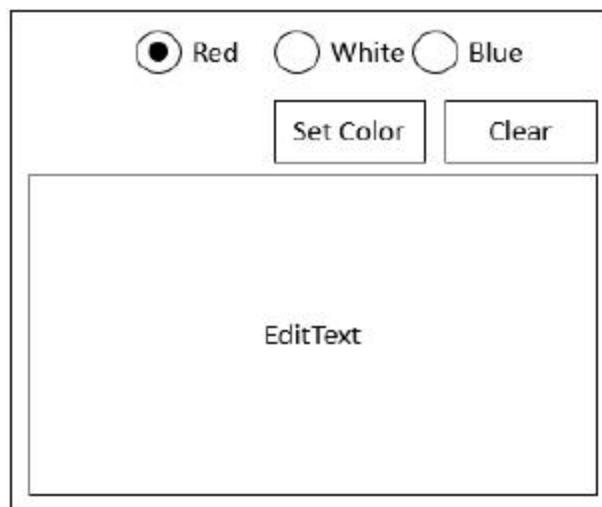
Bài học trình bày cách thức cơ bản để tổ chức giao diện của Activity. Cách tiếp cận chính là dùng XML Layout chứa các View và ViewGroup bên trong. View là thành phần đơn nhất, trong khi ViewGroup cho phép chứa các View/ViewGroup khác bên trong. Mỗi thành phần ViewGroup sẽ có cách thức bố trí các thành phần bên trong theo trật tự quy định sẵn. ViewGroup xem như là thành phần chính trong việc tổ chức giao diện của Activity, cùng một giao diện có thể có nhiều cách bố trí và sử dụng các thành phần layout khác nhau, tuy nhiên cần chọn một cách bố trí nào đó hợp lý và đơn giản nhất để tối ưu hiệu năng thực thi của ứng dụng.

## BÀI TẬP

**Câu 1:** Tạo ứng dụng Android có giao diện như hình vẽ sau:



**Câu 2:** Dùng LinearLayout tạo giao diện có dạng như hình sau:



Trong đó các RadioButton nằm giữa ở dòng trên cùng, tiếp theo 2 button cạnh về bên phải và cuối cùng là EditText chiếm phần không gian còn lại. Mặc định là Radio Red được chọn, do đó màu của EditText ban đầu là màu đỏ. Viết phần xử lý cho hai button như sau:

- Button Set Color: thiết lập màu text cho EditText là màu chọn trong RadioButton.
- Button Clear: thiết lập màu text cho EditText là màu đen.

**Câu 3:** Sử dụng TableLayout để giao diện như hình sau:



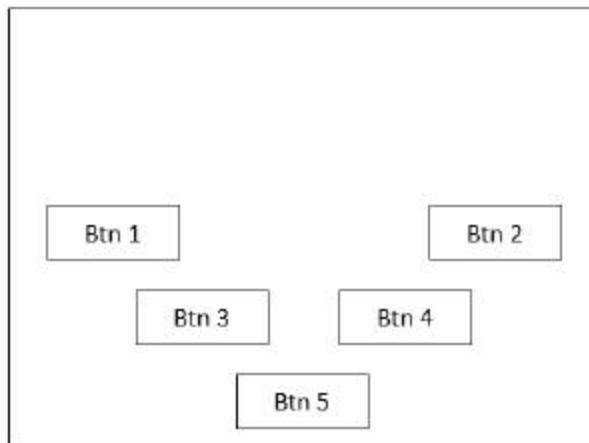
Bao gồm 3 dòng:

- Dòng 1 chứa 3 button

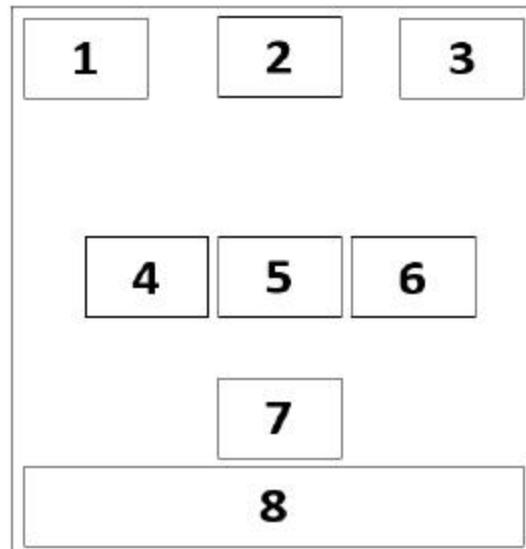
- Dòng 2 chứa 1 TextView duy nhất
- Dòng 3 chứa 1 button.

Viết phần xử lý cho các button: 3 button màu thì thiết lập màu tương ứng, button Clear thì thiết lập màu đen cho nội dung của TextView.

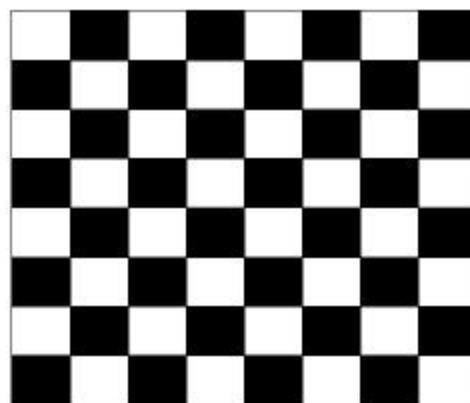
**Câu 4:** Sử dụng RelativeLayout xây dựng giao diện như hình sau:



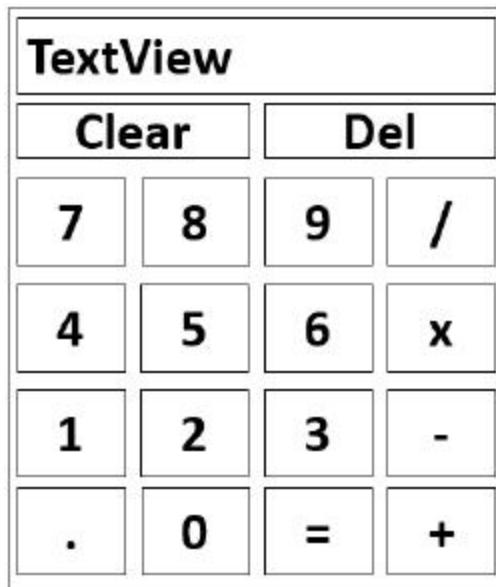
**Câu 5:** Dùng RelativeLayout tạo giao diện ứng dụng như sau:



**Câu 6:** Tạo ứng dụng Android hiển thị bàn cờ vua như sau:



**Câu 7:** Tạo giao diện của ứng dụng Calculator như sau:



# BÀI 5: THÀNH PHẦN VIEW - WIDGET

Học xong bài này, người học cần nắm được các nội dung sau.

- *Cách sử dụng các thành phần View-Widget cơ bản.*
- *Biết cách lựa chọn View thích hợp trong việc xây dựng GUI.*
- *Cách thức xử lý các sự kiện thông dụng trên View.*
- *Sử dụng các View nâng cao có nhiều tính năng tương tác với người dùng.*

## 5.1 VIEW CƠ BẢN

---

Đây là các View cơ bản trong Android cho phép hiển thị nội dung văn bản, hình ảnh cũng như thực hiện một số các lựa chọn cơ bản. Các View cơ bản như sau:

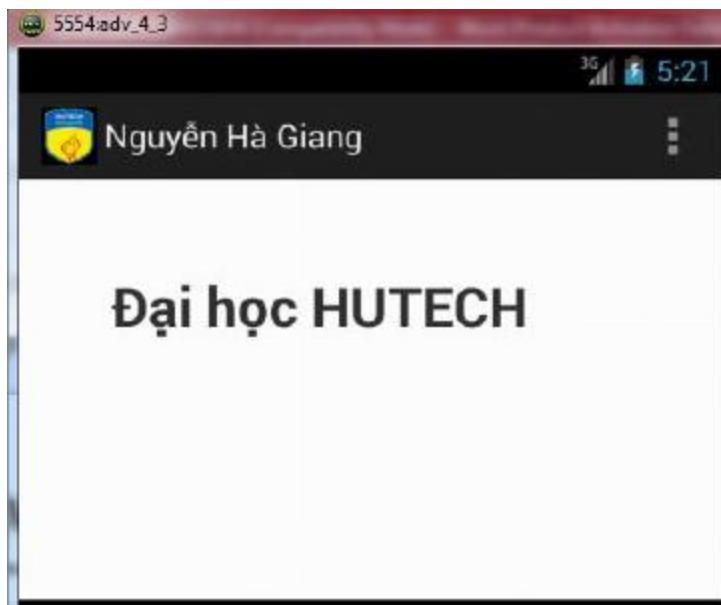
- |            |                |               |
|------------|----------------|---------------|
| - TextView | - ImageButton  | - RadioButton |
| - EditText | - ToggleButton | - RadioGroup  |
| - Button   | - CheckBox     |               |

❖ TextView: dùng để hiển thị thông tin cho người dùng và không cho người dùng chỉnh sửa. Đây là dạng View cơ bản nhất và hay được dùng trong Activity.

<TextView

```
    android:id="@+id/tvHutech"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignParentLeft="true"  
    android:layout_alignParentTop="true"  
    android:layout_marginLeft="50dp"
```

```
    android:layout_marginTop="50dp"  
    android:textStyle="bold"  
    android:text="Đại học HUTECH"  
    android:textSize="30sp"/>
```



**Hình 5.1: Minh họa TextView.**

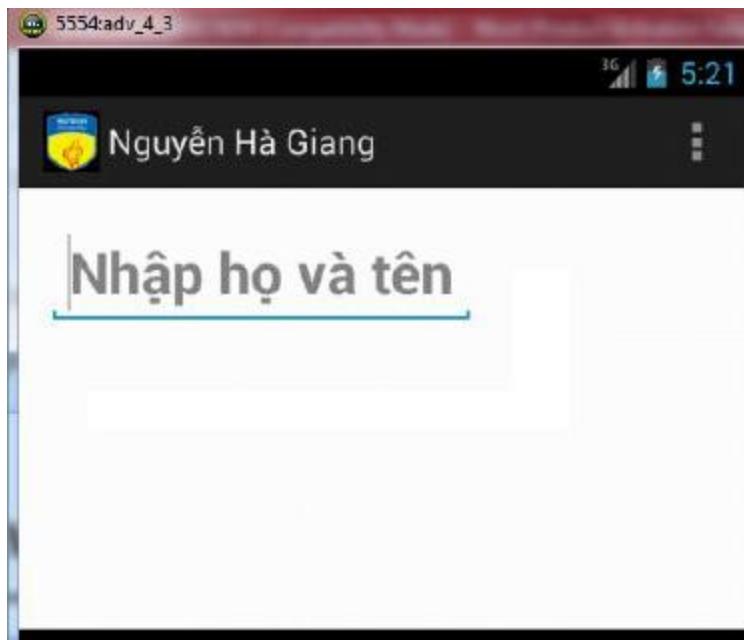
Để thay đổi nội dung thể hiện của TextView khi ứng dụng đang chạy thì phải dùng thuộc tính ID của TextView:

```
TextView tvHutech;  
  
tvHutech = (TextView) findViewById(R.id.tvHutech);  
  
tvHutech.setText("Đại học Công nghệ TP.HCM");
```

- ❖ EditText: đây là View cho phép người dùng có thể nhập thông tin vào, thông thường dùng để thu thập thông tin từ người dùng.

```
<EditText  
    android:id="@+id/edName"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignParentLeft="true"  
    android:layout_alignParentTop="true"
```

```
    android:layout_marginLeft="50dp"
    android:layout_marginTop="50dp"
    android:textStyle="bold"
    android:textSize="30sp"
    android:hint="Nhập họ và tên"/>
```



**Hình 5.2: EditText**

Trong code Java sẽ tham chiếu đến EditText thông qua giá trị ID:

```
EditText edName;
edName = (EditText) findViewById(R.id.edName);
String Name = edName.getText().toString();
```

- ❖ Button: chức năng tương tự như Button trong lập trình giao diện trên máy tính, dùng để bắt sự kiện lựa chọn của người dùng.

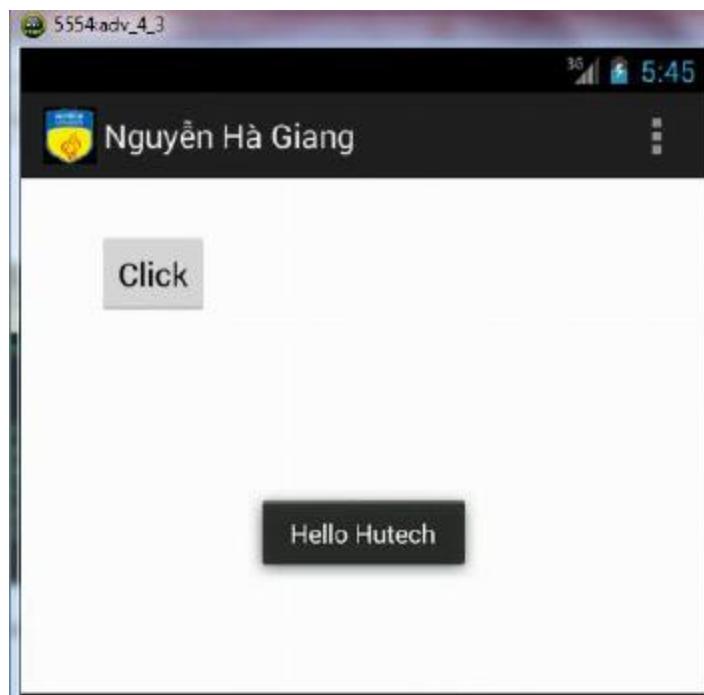
<Button

```
    android:id="@+id	btnSayHello"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
```

```
    android:layout_marginLeft="40dp"  
    android:layout_marginTop="40dp"  
    android:text="Click" />
```

Phần khai báo xử lý sự kiện click trong code Java.

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    Button btn = (Button)findViewById(R.id.btnSayHello);  
    btn.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View v) {  
            Toast.makeText(getApplicationContext(), "Hello Hutech",  
                Toast.LENGTH_LONG).show();  
        }  
    });  
}
```

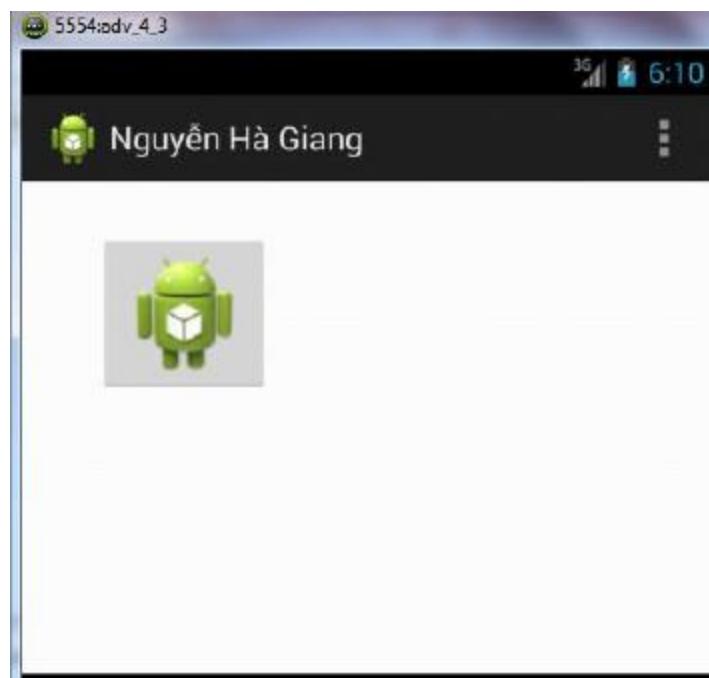


**Hình 5.3:** Xử lý khi button được chọn.

- ❖ ImageButton: Tương tự như Button nhưng dùng ảnh thay thế cho văn bản, sử dụng thuộc tính android:src để thiết lập ảnh cho Button.

```
<ImageButton
```

```
    android:id="@+id/btnAndroid"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignParentLeft="true"  
    android:layout_alignParentTop="true"  
    android:layout_marginLeft="40dp"  
    android:layout_marginTop="30dp"  
    android:src="@drawable/ic_launcher"/>
```



**Hình 5.4: Minh họa ImageButton**

- ❖ ToggleButton: là dạng Button có hai trạng thái on/off.

```
<ToggleButton
```

```
    android:id="@+id/toggleButton1"  
    android:layout_width="wrap_content"
```

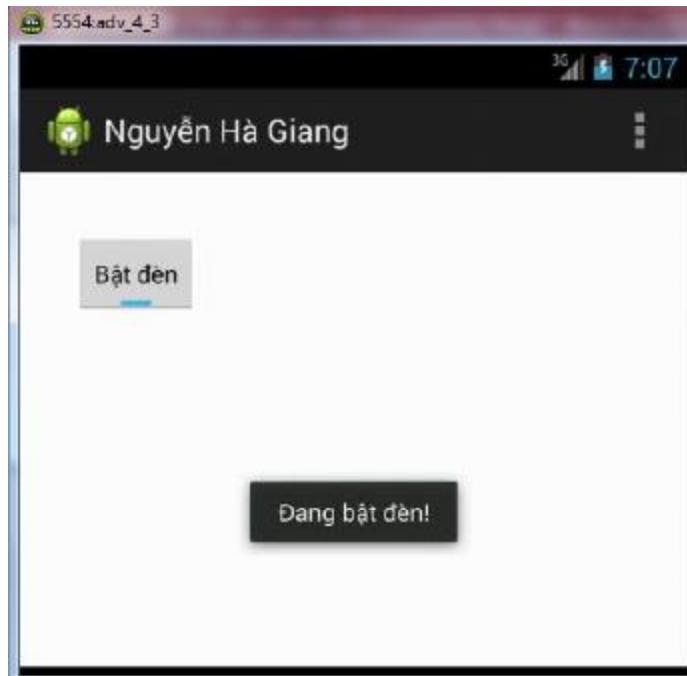
```
    android:layout_height="wrap_content"  
    android:layout_alignParentLeft="true"  
    android:layout_alignParentTop="true"  
    android:layout_marginLeft="31dp"  
    android:layout_marginTop="34dp"  
    android:textOn="Bật đèn"  
    android:textOff="Tắt đèn"  
/>
```

Phần code xử lý cho ToggleButton:

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    final ToggleButton bt = (ToggleButton)findViewById(R.id.tbLight);  
    bt.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View v) {  
            String str;  
            if (bt.isChecked())  
                str = "Đang bật đèn!";  
            else  
                str = "Đang tắt đèn!";  
            Toast.makeText(getApplicationContext(), str,  
                    Toast.LENGTH_LONG).show();  
        }  
    });  
}
```

Khi nhấn vào Button thì tùy theo trạng thái là on hay off mà có xử lý tương ứng. Trong hàm xử lý sự kiện click của ToggleButton, dùng phương thức isChecked của

ToggleButton để kiểm tra trạng thái của Button là on hay off, nếu là on thì isChecked trả về true, và là false trong trường hợp ngược lại. Tùy theo phiên bản của Android mà ToggleButton có phần thể hiện trạng thái on hay off có thể khác nhau.



**Hình 5.5: Xử lý khi ToggleButton là on.**

- ❖ CheckBox: là kiểu Button đặc biệt thể hiện hai trạng thái check hay uncheck.

Xem thử ví dụ minh họa sau:

1. Tạo file strings.xml chứa các chuỗi sử dụng trong XML layout

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, MyAndroidAppActivity!</string>
    <string name="app_name">Nguyễn Hà Giang</string>
    <string name="chk_ios">iPhone</string>
    <string name="chk_android">Android</string>
    <string name="chk_windows">Windows Mobile</string>
    <string name="btn_display">Display</string>
</resources>
```

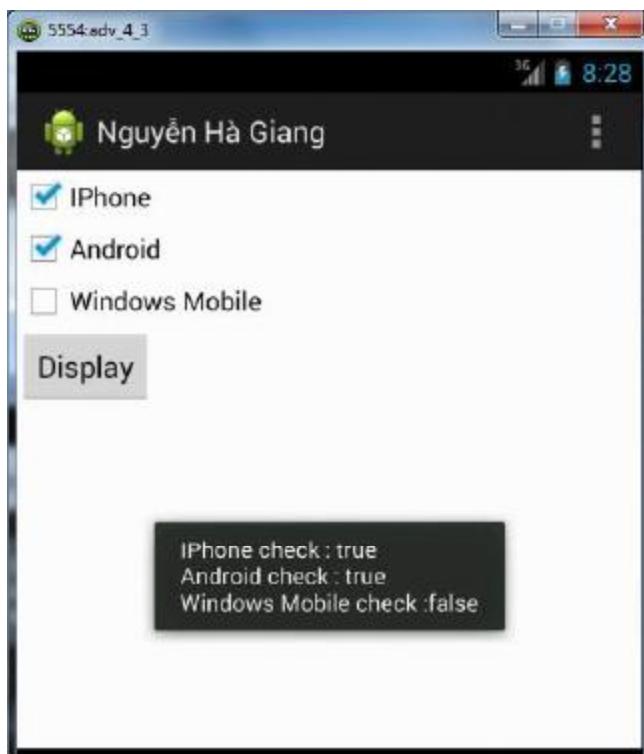
2. Tạo file XML layout với 3 CheckBox như sau:

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:orientation="vertical" >  
  
    <CheckBox  
        android:id="@+id/chkIos"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/chk_ios" />  
  
    <CheckBox  
        android:id="@+id/chkAndroid"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/chk_android"  
        android:checked="true" />  
  
    <CheckBox  
        android:id="@+id/chkWindows"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/chk_windows" />  
  
    <Button  
        android:id="@+id/btnDisplay"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/btn_display" />  
  
</LinearLayout>
```

3. Phần xử lý trong Activity như sau:

```
public class DemoCheckBox extends Activity {  
    private CheckBox chkIos, chkAndroid, chkWindows;  
    private Button btnDisplay;  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
        addListenerOnButton();  
    }  
  
    public void addListenerOnButton() {  
        chkIos = (CheckBox) findViewById(R.id.chkIos);  
        chkAndroid = (CheckBox) findViewById(R.id.chkAndroid);  
        chkWindows = (CheckBox) findViewById(R.id.chkWindows);  
        btnDisplay = (Button) findViewById(R.id.btnDisplay);  
        btnDisplay.setOnClickListener(new View.OnClickListener() {  
            //Chạy khi button click  
  
            @Override  
            public void onClick(View v) {  
                StringBuffer result = new StringBuffer();  
                result.append("iPhone check : ").append(chkIos.isChecked());  
                result.append("\nAndroid check : ").append(chkAndroid.isChecked());  
                result.append("\nWindows Mobile check :")  
                    .append(chkWindows.isChecked());  
                Toast.makeText(DemoCheckBox.this, result.toString(),  
                    Toast.LENGTH_LONG).show();  
            }  
        });  
    }  
}
```

#### 4. Chương trình khi chạy demo



**Hình 5.6: Sử dụng CheckBox.**

##### ❖ RadioButton & RadioGroup

Trong Android có thể dùng RadioButton để thể hiện phần chọn theo dạng radio, các RadioButton cùng nhóm với nhau thì chứa bên trong RadioGroup. Trong một nhóm button (RadioGroup) thì khi một button được chọn thì các Button khác trong nhóm sẽ tự động bị uncheck, đây là đặc tính chỉ được chọn một option trong số các option cùng nhóm, khác với CheckBox là cho phép chọn nhiều.

#### 1. Tạo file strings.xml có nội dung sau

```
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <string name="app_name">Nguyễn Hà Giang</string>  
    <string name="radio_male">Male</string>  
    <string name="radio_female">Female</string>  
    <string name="btn_display">Display</string>  
</resources>
```

2. Tạo file XML layout có chứa 2 RadioButton và 1 RadioGroup như sau

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <RadioGroup
        android:id="@+id/radioSex"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >
        <RadioButton
            android:id="@+id/radioMale"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/radio_male"
            android:checked="true" />
        <RadioButton
            android:id="@+id/radioFemale"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/radio_female" />
    </RadioGroup>
    <Button
        android:id="@+id/btnDisplay"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/btn_display" />
</LinearLayout>
```

### 3. Phần xử lý trong code Java

```
public class MainActivity extends Activity {  
    private RadioGroup radioSexGroup;  
    private RadioButton radioSexButton;  
    private Button btnDisplay;  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        addListenerOnButton();  
    }  
    public void addListenerOnButton() {  
        radioSexGroup = (RadioGroup) findViewById(R.id.radioSex);  
        btnDisplay = (Button) findViewById(R.id.btnDisplay);  
        btnDisplay.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View v) {  
                // get selected radio button from radioGroup  
                int selectedId = radioSexGroup.getCheckedRadioButtonId();  
                // find the radiobutton by returned id  
                radioSexButton = (RadioButton) findViewById(selectedId);  
                Toast.makeText(getApplicationContext(),  
                    radioSexButton.getText(), Toast.LENGTH_SHORT).show();  
            }  
        });  
    }  
}
```

## 4. Biên dịch và chạy trên máy ảo

**Hình 5.7: Sử dụng RadioButton.**

## **5.2 VIEW NÂNG CAO**

---

Trong phần này giới thiệu một số các View nâng cao thường dùng, do giới hạn của giáo trình, các dạng View nâng cao khác xin người đọc xem thêm trong tài liệu tham khảo ở cuối chương.

❖ **ProgressBar:** là dạng View hiển thị trực quan cho biết tiến độ thi hành của một tác vụ nào đó. Có hai dạng thể hiện thanh tiến độ:

- Dạng biết trước tiến độ công việc, nên có thể hiển thị phần trăm hoàn thành công việc. Ví dụ như việc download có thể dựa trên kích thước hiện tại download được để tính chính xác phần trăm hoàn thành.
- Dạng không biết trước khi nào tác vụ hoàn thành, có thể hiển thị biểu tượng dạng animation chỉ cho biết công việc đang thực hiện.

1. Tạo button trong XML Layout như sau:

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

```
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
<Button
    android:id="@+id	btnStartProgress"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Download File" />
</LinearLayout>
```

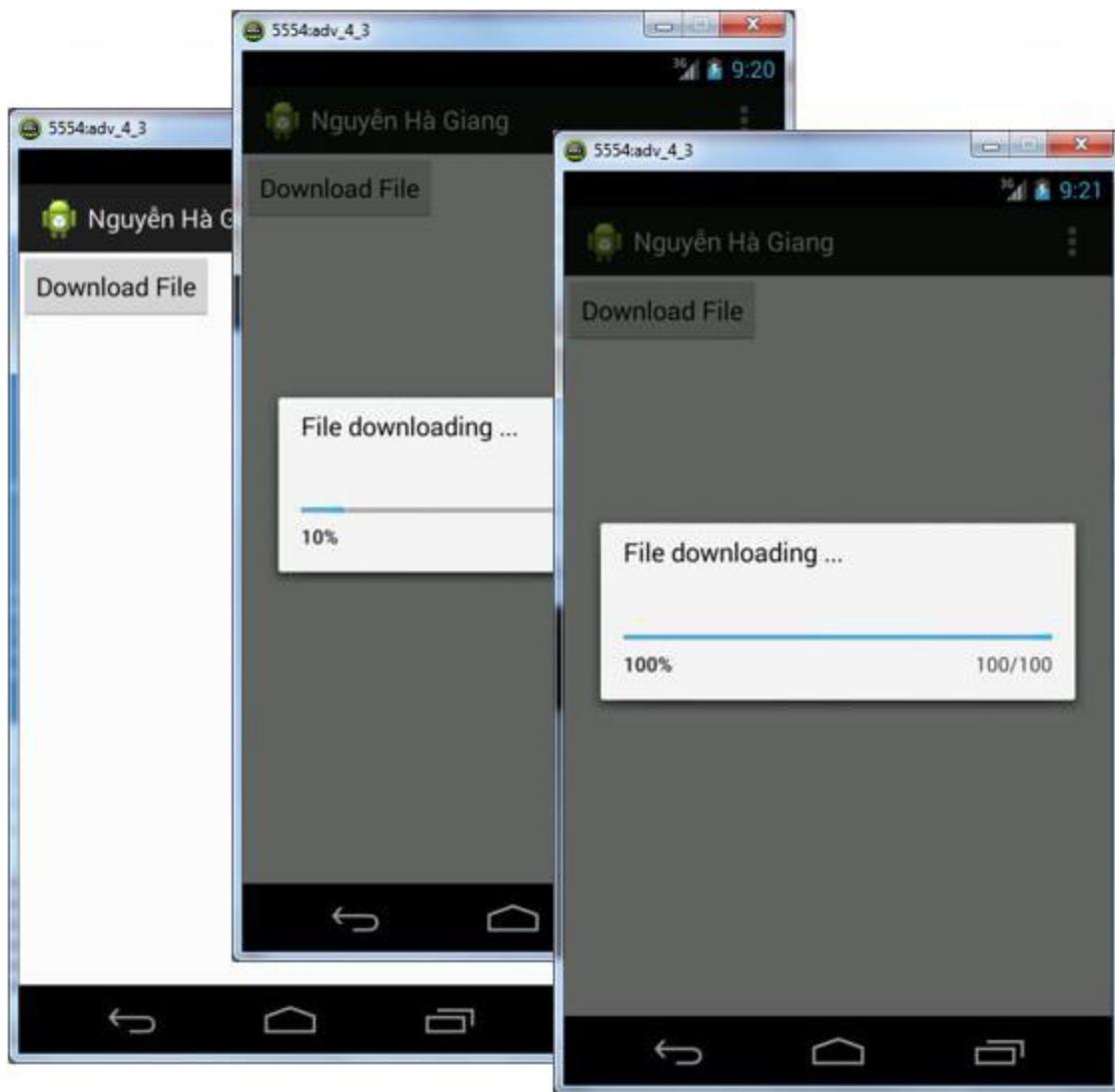
2. Phần code xử lý, có dùng thread để minh họa tiến độ hoàn thành của việc download.

```
public class MainActivity extends Activity {
    Button btnStartProgress;
    ProgressDialog progressBar;
    private int progressBarStatus = 0;
    private Handler progressBarHandler = new Handler();
    private long fileSize = 0;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        addListenerOnButton();
    }
    public void addListenerOnButton() {
        btnStartProgress = (Button) findViewById(R.id.btnStartProgress);
        btnStartProgress.setOnClickListener(
            new View.OnClickListener() {
                @Override
```

```
public void onClick(View v) {  
    // khởi tạo dialog  
    progressBar = new ProgressDialog(v.getContext());  
    progressBar.setCancelable(true);  
    progressBar.setMessage("File downloading ...");  
    progressBar.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);  
    progressBar.setProgress(0);  
    progressBar.setMax(100);  
    progressBar.show();  
    progressBarStatus = 0;  
    fileSize = 0;  
    new Thread(new Runnable() {  
        public void run() {  
            while (progressBarStatus < 100) {  
                // xử lý nhiệm vụ, demo nên không xử lý cụ thể  
                progressBarStatus = doSomeTasks();  
                // chờ 1 giây  
                try {  
                    Thread.sleep(1000);  
                } catch (InterruptedException e) {  
                    e.printStackTrace();  
                }  
                // Cập nhật progress bar  
                progressBarHandler.post(new Runnable() {  
                    public void run() {  
                        progressBar.setProgress(progressBarStatus);  
                    }  
                });  
            }  
            // download 100%,
```

```
if (progressBarStatus >= 100) {  
    // chờ 2 giây, để thấy việc hoàn thành 100%  
    try {  
        Thread.sleep(2000);  
    } catch (InterruptedException e) {  
        e.printStackTrace();  
    }  
    // đóng progress bar dialog  
    progressBar.dismiss();  
}  
}  
}).start();  
}  
});  
}  
// mô phỏng tiến trình download.  
public int doSomeTasks() {  
    while (fileSize <= 1000000) {  
        fileSize++;  
        if (fileSize == 100000) {  
            return 10;  
        } else if (fileSize == 200000) {  
            return 20;  
        } else if (fileSize == 300000) {  
            return 30;  
        }  
        // ...thêm code tùy thích  
    }  
    return 100;  
}
```

### 3. Kết quả chạy demo



**Hình 5.8: Minh họa ProgressBar.**

- ❖ **AutoCompleteTextView:** là lớp con của EditText cho phép hỗ trợ người dùng hoàn thành một chuỗi nhập dựa trên source có sẵn. Tính năng này rất hữu dụng, giảm thiểu việc nhập liệu từ người dùng và làm cho thao tác của người dùng nhanh gọn hơn.

Để tạo ra nguồn dữ liệu hỗ trợ trong AutoCompleteTextView thì Android dùng mảng chuỗi và ArrayAdapter để đưa vào AutoCompleteTextView. Ta xem ví dụ minh họa sau:

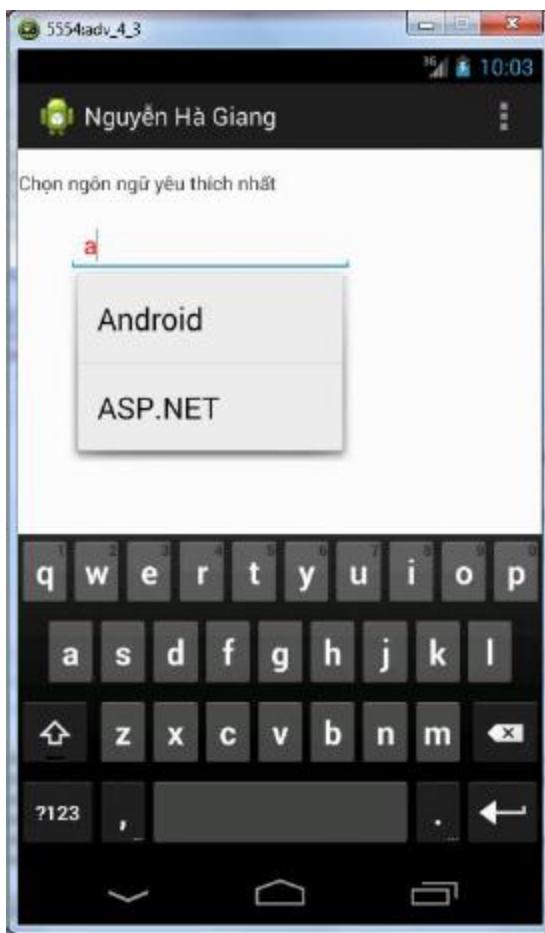
1. Phần XML layout chứa một AutoCompleteTextView:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    tools:context=".MainActivity" >  
  
    <TextView  
        android:id="@+id/textView1"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_alignParentLeft="true"  
        android:layout_alignParentTop="true"  
        android:layout_marginTop="15dp"  
        android:text="Chọn ngôn ngữ yêu thích nhất" />  
  
    <AutoCompleteTextView  
        android:id="@+id/autoCompleteTextView1"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_alignParentLeft="true"  
        android:layout_below="@+id/textView1"  
        android:layout_marginLeft="35dp"  
        android:layout_marginTop="15dp"  
        android:ems="10"  
        android:text="">  
        <requestFocus />  
    </AutoCompleteTextView>  
</RelativeLayout>
```

## 2. Phần code trong Activity của ứng dụng

```
public class MainActivity extends Activity {  
  
    String[] language = {"C", "C++", "Java", ".NET", "iPhone",  
                        "Android", "ASP.NET", "PHP"};  
  
    @Override  
  
    protected void onCreate(Bundle savedInstanceState) {  
  
        super.onCreate(savedInstanceState);  
  
        setContentView(R.layout.activity_main);  
  
        //Tạo adapter  
  
        ArrayAdapter<String> adapter = new ArrayAdapter<String>  
            (this, android.R.layout.select_dialog_item, language);  
  
        AutoCompleteTextView actv=  
            (AutoCompleteTextView) findViewById(R.id.autoCompleteTextView1);  
  
        actv.setThreshold(1); // hỗ trợ khi gõ 1 ký tự  
  
        actv.setAdapter(adapter);  
  
        actv.setTextColor(Color.RED);  
  
    }  
  
}
```

### 3. Chạy demo trên máy ảo



**Hình 5.9:** Chức năng Auto trong EditText

#### ❖ **ListView**

ListView là một trong những View rất quen thuộc đối với những ai lập trình ứng dụng cho Android. ListView giúp hiển thị một danh sách nào đó trong ứng dụng. Cách thể hiện của từng dòng (hay item) trong ListView cũng khá linh hoạt, cho phép người dùng có thể tùy chọn theo nhu cầu cụ thể của ứng dụng.

Ví dụ minh họa một ListView hiển thị một danh sách từ một mảng có sẵn.

Phần code XML layout gồm `TextView` chứa tiêu đề, một thanh gạch ngang dùng `View` và cuối cùng là `ListView`.

```
<TextView  
    android:id="@+id/tvTieuDe"  
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:textSize="30sp"
    android:text="@string/tieude"
    android:gravity="center"
    android:background="#00d"
    android:textColor="#fff"/>
<View
    android:layout_width="fill_parent"
    android:layout_height="2dp"
    android:layout_marginTop="5dp"
    android:background="#000000" />
```

```
<ListView
    android:id="@+id/lvDanhSach"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Phần code Java của Activity: trong code này tham chiếu đến ListView, sử dụng một ArrayAdapter<String>, adapter là thành phần trung gian, có nhiệm vụ lấy dữ liệu từ nguồn danh sách chuỗi và đưa vào ListView.

Khi bạn khởi tạo adapter lên thì sẽ có sẵn phần từ đó là:

*android.R.layout.simple\_list\_item\_1*, phần tử này được Android xây dựng sẵn cho chương trình và có nhiệm vụ lưu trữ dữ liệu được truyền vào.

```
public String [] danhSach = {"Công nghệ phần mềm", "Hệ thống thông tin",
    "Khoa học máy tính", "Mạng máy tính", "Hệ thống nhúng"};
private ListView lvDanhSach;
private ArrayAdapter<String> mainAdapter;
@Override
```

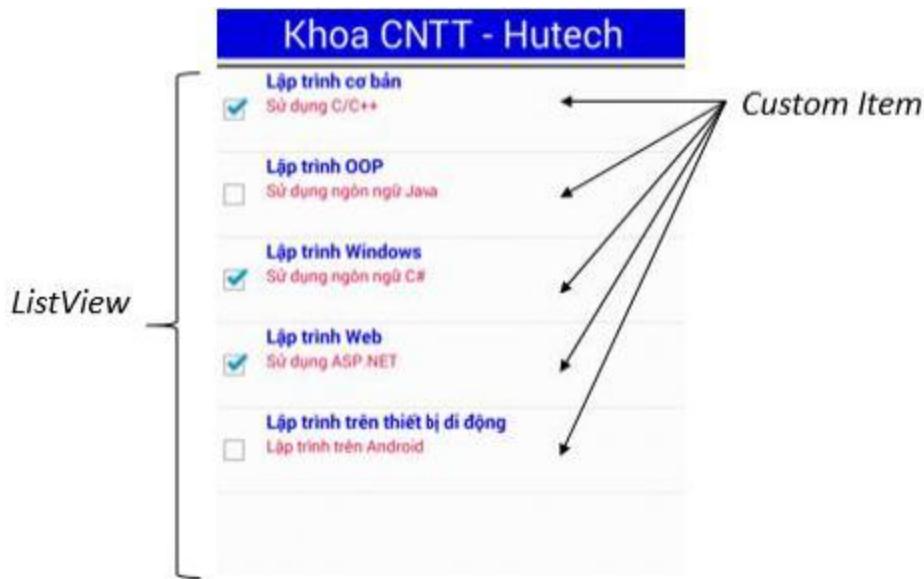
```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    lvDanhSach = (ListView) findViewById(R.id.lvDanhSach);  
    mainAdapter = new ArrayAdapter<String>(this,  
        android.R.layout.simple_list_item_1,danhSach);  
    lvDanhSach.setAdapter(mainAdapter);  
}
```



**Hình 5.10: Minh họa ListView cơ bản.**

Phần tiếp theo sẽ minh họa cách tạo một Custom ListView: đây là dạng ListView thường dùng nhất, do nhu cầu tùy biến giao diện ListView theo ý thích của người dùng.

Ứng dụng minh họa thiết kế một ListView có cách thể hiện từng item như hình 5.11 bên dưới:



**Hình 5.11: Mô tả thiết kế Custom ListView.**

Mỗi item trong ListView được thể hiện thông qua 3 View là: CheckBox, TextView chứa tiêu đề và TextView chứa nội dung của item. Cách thực hiện như sau:

Tạo giao diện của Activity như sau:

```
<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=".MainActivity" >

    <TextView
        android:id="@+id/tvTieuDe"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="30sp"
        android:text="@string/tieude"
        android:gravity="center"
        android:background="#00d" />

```

```
    android:textColor="#fff"/>  
  
<View  
    android:layout_width="fill_parent"  
    android:layout_height="2dp"  
    android:layout_marginTop="5dp"  
    android:background="#000000" />  
  
<ListView  
    android:id="@+id/lvDanhSach"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:scrollbars="vertical" />  
  
</LinearLayout>
```

Tiếp theo tạo layout mới có tên: list\_view.xml, layout này là giao diện mẫu của một item có trong ListView cần hiển thị.

```
<?xml version="1.0" encoding="utf-8"?>  
  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content">  
  
    <CheckBox  
        android:id="@+id/cbItem"  
        android:layout_width="wrap_content"  
        android:layout_height="66dp"  
        android:paddingRight="10sp"  
        android:paddingTop="20sp"  
        android:text="" />  
  
    <LinearLayout  
        android:layout_width="fill_parent"  
        android:layout_height="wrap_content"  
        android:orientation = "vertical">
```

```
<TextView  
    android:id="@+id/tvItem"  
    android:textSize = "15sp"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:lines="1"  
    android:textStyle="bold"  
    android:textColor="#0101DF" />  
  
<TextView  
    android:id="@+id/tvContent"  
    android:textSize="13sp"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:lines="1"  
    android:textColor="#DF013A" />  
  
</LinearLayout>  
</LinearLayout>
```

Tạo lớp Item chứa nội dung của một dòng dữ liệu trong ListView

```
public class Item implements Serializable {  
    private String title;  
    private String content;  
    private boolean checked;  
    public Item(String title) {this.title=title;}  
    public String getTitle() {return title;}  
    public void setContent(String content) {this.content=content;}  
    public String getContent() {return content;}}
```

```
@Override  
public boolean equals(Object o) {  
    if (o instanceof Item)  
    {  
        Item i = (Item)o;  
        return this.getTitle().equalsIgnoreCase(i.getTitle());  
    }  
    return false;  
}  
  
@Override  
public int hashCode() {  
    return title.hashCode();  
}  
}
```

Tạo lớp CustomItem, lớp này để load toàn bộ nội dung có trong file list\_view.xml đã xây dựng ở bước trên.

```
public class CustomItem extends LinearLayout  
{  
    CheckBox cbItem;  
    TextView tvItem;  
    TextView tvContent;  
    Context context;  
    public CustomItem(Context context){  
        super(context);  
        this.context = context;  
        LayoutInflator li = (LayoutInflator)this.getContext().  
            getSystemService(Context.LAYOUT_INFLATER_SERVICE);  
        li.inflate(R.layout.list_view, this,true);  
        cbItem = (CheckBox)findViewById(R.id.cbItem);  
        tvItem = (TextView)findViewById(R.id.tvItem);  
        tvContent = (TextView)findViewById(R.id.tvContent);
```

```
    }  
}
```

Tạo mới một lớp có tên ListItemAdapter. Lớp này sẽ extends từ ArrayAdapter. Mục đích là thiết kế lại cách hiển thị của ListView sao cho đúng yêu cầu của chương trình. Do đó cần override lại phương thức getView của lớp ArrayAdapter để có thể thực hiện lại việc này.

```
public class ListItemAdapter extends ArrayAdapter<Item> {  
  
    ArrayList<Item> array;  
  
    int resource;  
  
    CheckBox check;  
  
    TextView tvItem;  
  
    TextView tvContent;  
  
    Context context;  
  
    Item item;  
  
    public ListItemAdapter(Context context, int resource, ArrayList<Item> array) {  
        super(context, resource, array);  
        this.context = context;  
        this.resource = resource;  
        this.array = array;  
    }  
    @Override  
    public View getView(int position, View convertView, ViewGroup parent) {  
        View friendView = convertView;  
        if (friendView == null)  
            friendView = new CustomItem(getContext());  
        item = array.get(position);  
        if (item != null)
```

```
{  
    tvItem = ((CustomItem)friendView).tvItem;  
    tvContent = ((CustomItem)friendView).tvContent;  
    check = ((CustomItem)friendView).cbItem;  
    tvItem.setText(item.getTitle());  
    tvContent.setText(item.getContent());  
}  
return friendView;  
}  
}
```

Bước cuối cùng viết code cho Activity như sau:

```
public class MainActivity extends Activity {  
    private ArrayList<Item> array;  
    private ListItemAdapter arrayAdapter;  
    private ListView list;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        list = (ListView) findViewById(R.id.lvDanhSach);  
        array = new ArrayList<Item>();  
        arrayAdapter = new ListItemAdapter(this, R.layout.list_view, array);  
        list.setAdapter(arrayAdapter);  
        Item f = new Item("Lập trình cơ bản");  
        f.setContent("Sử dụng C/C++");  
        this.addItem(f);  
        this.addItem(f = new Item("Lập trình OOP"));  
        f.setContent("Sử dụng ngôn ngữ Java");  
        this.addItem(f = new Item("Lập trình Windows"));  
        f.setContent("Sử dụng ngôn ngữ C#");  
    }  
}
```

```

        this.addItem(f = new Item("Lập trình Web"));
        f.setContent("Sử dụng ASP.NET");
        this.addItem(f = new Item("Lập trình trên thiết bị di động"));
        f.setContent("Lập trình trên Android");
    }
    public void addItem(Item item) {
        array.add(item);
        arrayAdapter.notifyDataSetChanged();
    }
}

```

Biên dịch và chạy chương trình có kết quả như hình 5.12 sau:



**Hình 5.12: Custom ListView.**

## 5.3 TOAST VÀ DIALOG

- ❖ Thành phần Toast đơn giản

Toast cho phép hiển thị thông tin ngắn gọn ra màn hình theo dạng popup trong một khoảng thời gian nhất định và tự mất dần. Thường dùng để thông báo điều gì đó

cho người dùng biết. Khi hiển thị Toast không ảnh hưởng đến Activity khác và không bắt các sự kiện người dùng.

Cách sử dụng Toast cơ bản:

- Khởi tạo một đối tượng từ class Toast bằng hàm makeText():
  - `Toast t = Toast.makeText(context, message, duration)`
  - Context: application context
  - Message:nội dung thông điệp
  - Duration: thời gian hiển thị
- Để hiển thị, gọi hàm show: `t.show()`

Có thể né việc tạo đối tượng bằng cách gọi "*chain method*":

```
Toast.makeText(context, text, duration).show();
```

Vị trí xuất hiện mặc định của Toast thường là gần biên dưới màn hình, tuy nhiên có thể thay đổi vị trí với phương thức `setGravity(int , int, int)`. Ví dụ, nếu muốn Toast hiển thị góc trên bên trái thì thiết lập:

```
t.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```

Tham số thứ hai là giá trị tọa độ x và tham số thứ ba là tọa độ y.

#### ❖ Thành phần Custom Toast

Nếu chỉ hiển thị thông điệp đơn giản thì dùng Toast cơ bản như trên là được. Tuy nhiên, có thể thiết kế lại layout để hiển thị thông điệp theo ý người lập trình, lúc này ta gọi là Custom Toast. Các bước thực hiện sau để tạo Custom Toast.

- Tạo View layout thông qua XML hay code Java
- Truyền đối tượng View gốc của layout trên vào phương thức `setView(View)`.

Minh họa tạo Custom Toast như sau:

Tạo layout XML cho phần hiển thị của Toast, lưu thành file `toast_layout.xml`

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/toast_layout_root"
```

```
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="8dp"
    android:background="#DAAA" >
<ImageView android:src="@drawable/droid"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginRight="8dp" />
<TextView android:id="@+id/text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textColor="#FFF" />
</LinearLayout>
```

Lưu ý giá trị ID của LinearLayout là toast\_layout\_root, giá trị này dùng để inflate layout từ XML.

```
LayoutInflater inflater = getLayoutInflater();
View layout = inflater.inflate(R.layout.toast_layout,
    ( ViewGroup ) findViewById(R.id.toast_layout_root));
TextView text = ( TextView ) layout.findViewById(R.id.text);
text.setText("This is a custom toast");
Toast toast = new Toast(getApplicationContext());
toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
toast.setDuration(Toast.LENGTH_LONG);
toast.setView(layout);
```

```
toast.show();
```

❖ Dialog

Dialog là dạng cửa sổ nhỏ xuất hiện bên trên Activity hiện tại, người dùng bắt buộc phải xử lý thông tin trên dialog này trước khi quay lại với Activity. Dialog thường được sử dụng theo các dạng sau:

- AlertDialog: chứa thông tin text, list item và button để hỏi ý kiến người dùng.
- ProgressDialog: hiển thị thanh tiến trình
- DatePickerDialog: cho phép người dùng chọn ngày
- TimePickerDialog: cho phép người dùng chọn time.

Với AlertDialog, nội dung có thể chứa các phần: tiêu đề, nội dung thông điệp, tối đa có 3 Buttons, danh sách các item có thể chọn lựa kèm theo CheckBox hay RadioButton. Để khởi tạo AlertDialog, ta phải sử dụng lớp con là AlertDialog.Builder.

Minh họa sử dụng AlertDialog, các dạng dialog khác người đọc xem trong các e-book tham khảo.

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);  
  
    //Thiết lập tiêu đề hiển thị  
    builder.setTitle("Tiêu đề AlertDialog");  
  
    //Thiết lập thông báo hiển thị  
    builder.setMessage("Nội dung thông báo! ");  
  
    builder.setCancelable(false);  
  
    //Tạo nút Activity2  
    builder.setPositiveButton("Activity 2",  
        new DialogInterface.OnClickListener() {  
            public void onClick(DialogInterface dialog,  
                int which) {  
  
                Intent myIntent = new Intent(MainActivity.this, Activity2.class);  
            }  
        }  
    );
```

```
        startActivity(myIntent);

        finish();

    }

});

//Tạo nút Hủy

builder.setNeutralButton("Hủy", null);

//Tạo nút Thoát

builder.setNegativeButton("Thoát",

    new DialogInterface.OnClickListener() {

        public void onClick(DialogInterface dialog, int which) {

            finish();

        }

    });

AlertDialog dialog = builder.create();

// hiển thị dialog ra màn hình

dialog.show();
```

Trong ví dụ trên khi dialog hiển thị sẽ có 3 Button cho phép người dùng chọn: Button "Thoát" sẽ kết thúc ứng dụng Android, "Hủy" sẽ đóng dialog và quay lại Activity hiện hành, và "Activity 2" cho phép hiển thị một Activity khác (chi tiết cách gọi hiển thị Activity xem trong bài thành phần Intent).

# TÓM TẮT

Trong bài học này giới thiệu các thành phần tương tác cơ bản trên giao diện người dùng trong ứng dụng Android. Người học lần lượt tiếp cận với các Widget cơ bản như *EditText*, *TextView*, *Button*, *RadioButton*, *CheckBox*... Các thành phần này cung cấp các sự kiện cơ bản cho phép người dùng tương tác. Đây là các phần cơ bản và thiết yếu nhất trong ứng dụng Android khi cần tương tác với người dùng. Các ứng dụng Android có thành phần *Activity* đều chứa các Widget cơ bản như trên. Song song với các phần cơ bản là các View nâng cao, cho phép nâng cao mức độ tương tác với người dùng. Đây là các Widget được thiết kế chuyên biệt cho một số tính năng nào đó. Người lập trình Android cần thiết tìm hiểu và bổ sung các Widget để nâng cao tính năng tương tác với người dùng.

## BÀI TẬP

**Câu 1:** Viết ứng dụng Android cho phép người dùng nhập vào hai số nguyên. Thực hiện các chức năng: tính tổng, hiệu, tích, thương, ước số chung lớn nhất và các số nguyên tố nằm giữa hai số đó. Yêu cầu mỗi chức năng thể hiện trên các Button và kết quả hiển thị trên *TextView*.

**Câu 2:** Viết ứng dụng Android cho phép giải phương trình bậc 2 có mô tả như sau: cho phép user nhập các hệ số từ các *EditText*, có chức năng giải phương trình và kết quả hiển thị ra *TextView*.

**Câu 3:** Viết ứng dụng Android minh họa form đăng nhập, cho phép user nhập vào Account Name và Password. Kiểm tra thông tin đăng nhập nếu có Account Name là "hutech" và password là "hutech123" thì xuất thông tin chào mừng đăng nhập thành công trên *Toast*. Ngược lại thông báo đăng nhập thất bại.

**Câu 4:** Viết ứng dụng Android cho phép chuyển đơn vị từ độ C sang độ F và ngược lại. Công thức chuyển đổi như sau:

$$\langle \text{Độ F} \rangle = \langle \text{Độ C} \rangle * 9/5 + 32$$

**Câu 5:** Viết ứng dụng chuyển đổi năm Dương lịch sang Âm lịch. Ứng dụng cho phép nhập vào năm dương lịch ( $>=1900$ ) sau đó chuyển sang ngày âm lịch theo công thức như sau:

Ngày Âm lịch = Can + Chi

Can được tính như sau:

Can = năm Dương Lịch % 10

Canh = 0, Tân = 1, Nhâm = 2, Quý = 3, Giáp = 4, Ất = 5, Bính = 6, Đinh = 7, Mậu = 8, Kỷ = 9.

Chi được tính như sau:

Chi = năm Dương lịch %12

Tí = 0, Sửu = 1, Dần = 2, Mão = 3, Thìn = 4, Tị = 5, Ngọ = 6, Mùi = 7, Thân = 8, Dậu = 9, Tuất = 10, Hợi = 11.

Viết ứng dụng Android minh họa chức năng đăng ký học phần của sinh viên. Các học phần có thể đăng ký bao gồm:

- Học phần bắt buộc, mỗi môn là 3 tín chỉ:
  - Lập trình Windows
  - Lập trình CSDL
  - Công cụ & môi trường phát triển phần mềm
  - Công nghệ phần mềm
  - Quản lý dự án phần mềm
- Học phần tự chọn, số trong ngoặc là số tín chỉ:
  - Phát triển PM MNM (3TC)
  - Lập trình Web (2TC)
  - HTML 5 & CSS (2TC)
  - Lập trình Android (3TC)
  - Lập trình Web mobile (2TC)

- Kiểm nghiệm phần mềm (2TC)

Yêu cầu khi đăng ký nhập vào họ tên, mã số sinh viên, cho phép chọn đăng ký theo quy tắc sau: đối với môn bắt buộc thì phải đăng ký tối thiểu 9TC (phần còn lại có thể đăng ký ở HK sau), môn tự chọn thì tối thiểu là 5TC. Sau khi đăng ký theo đúng yêu cầu trên thì ứng dụng hiển thị kết quả dạng Toast các thông tin: họ tên SV, MSSV, các môn bắt buộc đăng ký và tổng số TC, các môn tự chọn và tổng số TC.

**Câu 6:** Viết ứng dụng Pizza Order cho phép chọn để mua bánh Pizza. Giao diện của ứng dụng có dạng như hình vẽ bên dưới, tuy nhiên người lập trình có thể tùy ý bổ sung thay thiết kế giao diện theo ý riêng.



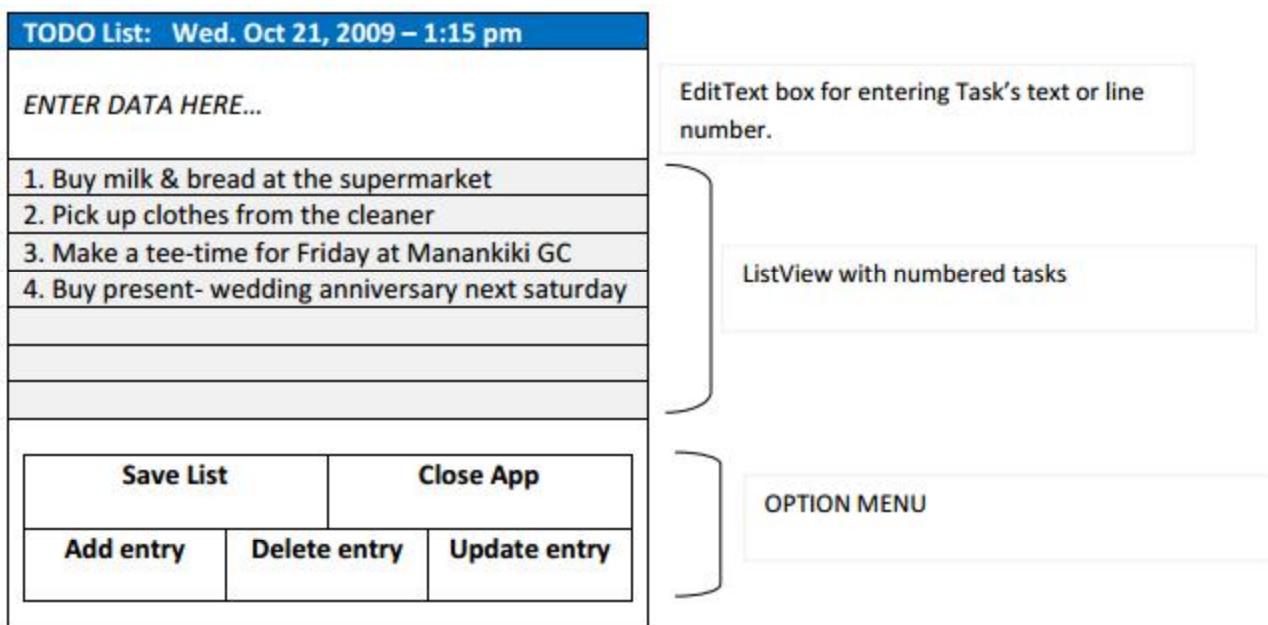
Chức năng "SMS-Place you order" ở đây chỉ đơn giản là hiển thị các thông tin đã chọn ra một Toast.

**Câu 7:** Viết ứng dụng thực thi danh sách TODO đơn giản. Ứng dụng hiển thị ListView với danh sách đánh số tuân tự các công việc cần làm.

Ứng dụng sử dụng Option Menu đưa ra các chức năng sau:

1. Thêm entry mới vào danh sách
2. Delete một entry trong danh sách (hỏi số entry rồi xóa)
3. Cập nhật nội dung của một công việc
4. Lưu danh sách (tạm thời có thể chỉ hiển thị ra Toast là đã lưu)
5. Đóng ứng dụng

Giao diện ứng dụng có mô tả theo dạng sau:



# BÀI 6: THÀNH PHẦN INTENT

Học xong bài này, người học cần nắm được các nội dung sau.

- Cơ chế truyền thông điệp giữa các thành phần trong ứng dụng Android.
- Phân loại các dạng Intent và cách sử dụng để gọi các thành phần chính của ứng dụng Android.
- Cách sử dụng đối tượng Intent để gọi và hiển thị các Activity.
- Cơ chế gởi nhận dữ liệu giữa hai Activity thông qua đối tượng Intent.

## 6.1 CƠ CHẾ INTENT

Trong android, Intent là một đối tượng gửi thông điệp được sử dụng để yêu cầu một hành động từ một thành phần ứng dụng khác. Ba thành phần cốt lõi của một ứng dụng: Activity, Service, và Broadcast Receiver được kích hoạt thông qua các thông điệp, được gọi là Intent. Thông điệp Intent là phương tiện cho việc kết buộc trễ (vào lúc runtime) giữa các thành phần trong cùng hay khác ứng dụng.

Bản thân Intent là đối tượng, là cấu trúc dữ liệu mô tả khái quát một hành động thực thi. Thông thường trong trường hợp của Broadcast, thì Intent là mô tả một việc gì đó được diễn ra và sẽ được thông cáo.

Có một số cơ chế khác nhau cho việc phân phát các Intent đến mỗi thành phần.

1. Một đối tượng Intent được truyền cho phương thức startActivity hay startActivityForResult để phát sinh ra Activity mới hoặc gọi Activity đã tồn tại.
2. Một đối tượng Intent được truyền cho phương thức startService để khởi tạo một Service hoặc đưa các chỉ dẫn mới cho Service. Tương tự như vậy một Intent có thể được truyền cho bindService để thiết lập kết nối giữa thành phần gọi và Service.
3. Những đối tượng Intent được truyền cho bất cứ phương thức broadcast như là: sendBroadcast(), sendOrderedBroadcast(), hay sendStickyBroadcast() để gọi các broadcast receiver đã đăng ký.

Trong các trường hợp trên, hệ thống Android sẽ tìm các Activity, service, hay broadcast receiver tương ứng để phản hồi với Intent, khởi động các thành phần này nếu cần. Không có việc chồng chéo giữa các thông điệp hệ thống này: broadcast Intent thì đưa cho broadcast receiver, chứ không thể đưa cho Activity hay service. Ngược lại một Intent được truyền startActivity() thì chỉ phân phát đến một Activity, không bao giờ phát đến service hay broadcast receiver.

Nói chung, trong android, Intent sẽ giúp chúng ta duy trì giao tiếp giữa các app component từ cùng một ứng dụng cũng như với các app component của các ứng dụng khác. Intent chủ yếu hữu ích để thực hiện những việc sau.

Khả năng	Mô tả
Bắt đầu một Activity	Bằng cách gửi một đối tượng "Intent" đến phương thức startActivity(), chúng ta có thể bắt đầu một "Activity" mới hoặc "Activity" hiện có để thực hiện những việc cần thiết.
Bắt đầu một Service	Bằng cách gửi một đối tượng Intent tới phương thức startService (), chúng ta có thể bắt đầu một "Service" mới hoặc gửi các hướng dẫn cần thiết đến một "Service" hiện có.
Gửi một Broadcast	Bằng cách gửi một đối tượng Intent tới phương thức sendBroadcast(), chúng ta có thể gửi thông điệp của mình đến các ứng dụng Broadcast khác.

## 6.2 ĐỐI TƯỢNG INTENT

Một đối tượng Intent là một gói thông tin. Nó chứa thông tin cần thiết cho thành phần nhận Intent (bao gồm hành động thực thi và dữ liệu để hành động diễn ra) và các thông tin thêm cho hệ thống Android biết (loại thành phần xử lý và các chỉ dẫn để chạy các thành phần).

Đối tượng Intent gồm các thành phần sau:

- Tên thành phần (component name): Tên của thành phần xử lý Intent, trường này là đối tượng ComponentName, là một sự kết hợp đầy đủ tên lớp của thành phần đích ví dụ như hutech.fit.fit.DemoIntent. Chúng ta có thể đặt tên thành phần bằng cách sử dụng setComponent(), setClass(), setClassName() hoặc bằng cách sử dụng hàm tạo Intent.

2. Hành động (action): Chuỗi hành động sẽ thực thi, hoặc trong trường hợp là broadcast thì có một số hằng action được định nghĩa trước.

**Bảng 6.1: Các hằng Action được định nghĩa trước.**

Hằng	Thành phần đích	Hành động
ACTION_CALL	Activity	Tạo cuộc gọi
ACTION_EDIT	Activity	Hiển thị dữ liệu cho user edit
ACTION_MAIN	Activity	Khởi tạo Activity ban đầu cho stack, không có dữ liệu ban đầu và không trả về dữ liệu
ACTION_SYNC	Activity	Đồng bộ dữ liệu trên server với dữ liệu trên thiết bị
ACTION_BATTERY_LOW	Broadcast receiver	Cảnh báo khi pin yếu
ACTION_HEADSET_PLUG	Broadcast receiver	Cảnh báo khi headset được cắm vào thiết bị, hoặc bị remove
ACTION_SCREEN_ON	Broadcast receiver	Màn hình bật
ACTION_TIMEZONE_CHANGED	Broadcast receiver	Thiết lập thời gian thay đổi

Trong danh sách bên trên là các hằng đã định nghĩa trước cho các hành động chung, còn một số hành động khác được định nghĩa trong Android API. Ngoài ra người lập trình có thể định nghĩa chuỗi hành động riêng để kích hoạt các thành phần trong ứng dụng. Có thể đặt chuỗi theo kiểu thêm package của ứng dụng vào trước tên hành động (tên package làm tiền tố), ví dụ như:

hutech.fit.nguyenha.giang.MY\_ACTION.

Các hành động quyết định đến cấu trúc của phần còn lại trong Intent, cụ thể là phần data và các trường extras, cũng như là tên phương thức sẽ quyết định các tham số và giá trị trả về.

Hành động trong đối tượng Intent được thiết lập thông qua phương thức `setAction()` và phương thức đọc là `getAction()`.

### 3. Dữ liệu (data):

URI của dữ liệu và kiểu MIME của dữ liệu. Những hành động khác nhau thì có những loại khác nhau của việc đặc tả dữ liệu. Ví dụ nếu hành động là ACTION\_EDIT thì trường dữ liệu chứa URI của tài liệu để hiển thị cho việc soạn thảo. Nếu hành động

là ACTION\_CALL, thì trường dữ liệu là tel:URI với số được gọi. Tương tự như vậy, nếu hành động là ACTION\_VIEW và trường dữ liệu là http:URI, khi đó Activity nhận thông tin sẽ download và hiển thị dữ liệu mà URI tham chiếu đến.

#### 4. Phân loại (category):

Chuỗi chứa thông tin thêm vào cho biết loại thành phần xử lý Intent. Chúng ta có thể chỉ định một danh mục bằng cách sử dụng addCategory () .

#### 5. Extras:

Thông tin bổ sung theo dạng key-value cung cấp cho thành phần xử lý Intent. Đây là dạng dữ liệu bổ sung, có thể có hoặc không tùy theo hành động cụ thể. Ví dụ, hành động ACTION\_TIMEZONE\_CHANGED có thêm thông tin "time\_zone" lưu vùng thời gian mới; hành động ACTION\_HEADSET\_PLUG có thêm thông tin "state" chỉ ra headset được cắm hay rút.

#### 6. Flags:

Các cờ hiệu được thiết lập cho Intent để chỉ dẫn cho hệ thống Android cách chạy Activity. Các cờ hiệu này được định nghĩa trong lớp Intent.

## **6.3 PHÂN LOẠI INTENT**

---

Các Intent được chia vào hai nhóm chính:

#### 1. Explicit Intent:

Xác định rõ tên của thành phần đích (Activity, service, broadcast) thông qua thuộc tính name, thành phần đích này là thành phần sẽ thực thi các hành động được đặc tả trong Intent. Thường được dùng để truyền thông giữa các thành phần bên trong ứng dụng (application-internal message) như một Activity khởi tạo service hay một Activity khác. Thông thường thì các Intent này không chứa thêm các thông tin nào khác.

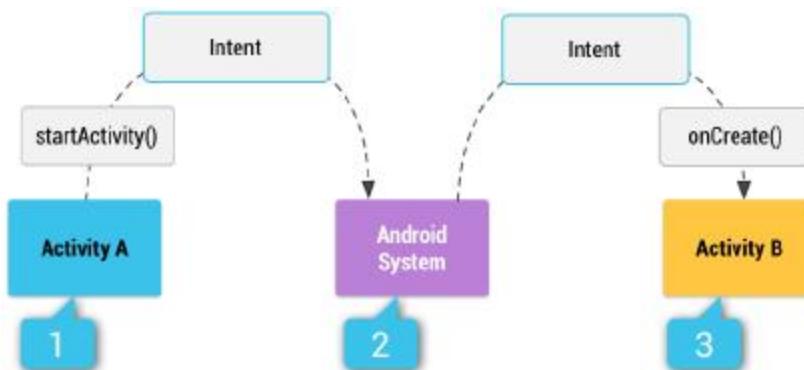
#### 2. Implicit Intent:

Không xác định tên thành phần, trường tên thành phần để trống. Dạng này dùng để kích hoạt các thành phần trong các ứng dụng khác. Dạng Intent này ngoài việc không xác định tên thành phần thì nó sẽ chứa đủ thông tin để hệ thống có thể xác định thành phần nào có sẵn là tốt nhất để thực thi Intent.

Hệ thống Android sẽ phân phát explicit Intent đến một thể hiện của lớp đích thông qua tên của thành phần đích được xác định trong đối tượng Intent.

Trong trường hợp implicit Intent, thông tin về thành phần đích không xác định, do đó hệ thống Android phải tìm thành phần thích hợp nhất để xử lý Intent. Khi đó một Activity, service thực thi hành động yêu cầu hoặc tập broadcast receiver đáp ứng lại các thông điệp broadcast.

Bất cứ thành phần nào (Activity, Broadcast Receiver, Service) khi muốn sử dụng trong ứng dụng đều phải được đăng ký trong file AndroidManifest.xml. Trong đó cần định nghĩa một thẻ <intent-filter> cung cấp các thông tin để hệ thống có thể xác định được cái mà các thành phần này có thể xử lý được (những action mà thành phần này có thể thực hiện được).



**Hình 6.1: Minh họa thực thi Implicit Intent trong hệ thống.**

Hình 6.1 minh họa cách một Implicit Intent được phát ra hệ thống để start một Activity khác. (1): Activity A tạo một Intent với hành động xác định và truyền cho phương thức startActivity(). (2): hệ thống Android tìm tất cả các ứng dụng trong hệ thống có Intent filter hợp với Intent đang xử lý, khi tìm thấy thì hệ thống sẽ start Activity B đó bằng cách gọi phương thức onCreate của nó và truyền Intent đang xử lý vào.

### 6.3.1 Intent Filter

Một thành phần (Activity, Service, Broadcast receiver) cung cấp thông tin cho hệ điều hành biết loại đối tượng Intent mà nó có thể xử lý bằng cách khai báo một hoặc nhiều Intent Filter. Mỗi Intent Filter cung cấp các thông tin về các hành động mà thành phần này có thể xử lý (ví dụ như hiển thị một contact, gọi điện thoại...). Các thành phần khai báo Intent Filter bằng cách sử dụng thẻ <intent-filter> đặt trong thẻ khai báo thành phần (<activity>, <service>, <receiver>) như ví dụ sau:

Ví dụ XX:

```
<activity android:name=".MainActivity">  
    <intent-filter>  
        <action android:name="android.intent.action.MAIN" />  
        <category android:name="android.intent.category.LAUNCHER" />  
        <data android:mimeType="text/plain"/>  
    </intent-filter>  
</activity>
```

Ví dụ trên khai báo một Activity với thẻ `<intent-filter>` chứa các thông tin dùng để chọn lựa Intent.

Với Intent tường minh thì thành phần nhận Intent đó đã được xác định cụ thể. Tuy nhiên, với các Intent không tường minh thì hệ thống sẽ tiến hành so sánh các thông tin phụ được khai báo trong Intent với các thành phần được khai báo trong tập `AndroidManifest.xml` để tìm ra thành phần phù hợp.

Một Intent Filter có các thành phần chính sau:

- **action:** Tên hành động mà thành phần có thể thực thi.
- **type:** Kiểu dữ liệu thành phần có thể thực thi.
- **category:** Phân nhóm các thành phần.
- **data:** Dữ liệu thành phần.

Đối với những dữ liệu không phải là nội dung cụ thể (như URI) thì việc xem xét lựa chọn Intent phù hợp sẽ dựa vào lược đồ (Scheme) của dữ liệu được cung cấp (như `http://mailto: ...`).

### 6.3.2 Cách thức xác định thành phần phù hợp với Intent

Như đã đề cập trong phần trước nếu Intent là dạng implicit thì hệ thống phải xác định được thành phần phù hợp với Intent đó. Để xác định xem một thành phần có hợp với Intent hay không thì hệ thống Android xem xét dựa theo các nguyên tắc sau:

- Trước tiên khi một Intent được gửi đi, Android sẽ tìm kiếm những thành phần Activity, Broadcast Receiver, Service có action-name phù hợp với Intent. Nếu có thành phần phù hợp thì Android sẽ mở thành phần đó lên để thực thi các hành động theo yêu cầu. Ngược lại nếu có nhiều hơn một thành phần có action-name phù hợp thì Android sẽ yêu cầu người dùng chọn một thành phần.
- Ngược lại nếu không có thành phần nào phù hợp thì Android sẽ tiến hành xem xét kiểu dữ liệu của Intent cung cấp xem có thành phần nào có đủ năng lực để xử lý kiểu dữ liệu đó không. Nếu không được Android sẽ tiến hành xem xét scheme của dữ liệu đó để tìm kiếm thành phần phù hợp. Nếu vẫn không tìm được thành phần phù hợp Android sẽ tiến hành xem xét các thành phần có cùng category với category được xác định trong đối tượng Intent để chọn thành phần.

Ví dụ: Khai báo Activity chính sẽ được mở khi một ứng dụng được khởi chạy:

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

Ví dụ trên khai báo một Activity tên là hutech\_main. Intent-Filter của Activity này có các thuộc tính:

- `<action android:name="android.intent.action.MAIN"/>`: Khai báo Activity main của thể thực thi được một hành động là mở một Activity.
- `<category android:name="android.intent.category.LAUNCHER"/>` : Khai báo Activity main thuộc nhóm các Activity được mở ra khi ứng dụng được chạy bởi người dùng.

Bằng cách khai báo như trên Activity tên main sẽ là Activity đầu tiên (hay còn gọi là Activity chính) sẽ được mở ra khi ứng dụng được chạy bởi người dùng.

Trong trường hợp nếu chúng ta muốn xử lý nhiều Intent với sự kết hợp của hành động, danh mục và dữ liệu, chúng ta cần tạo nhiều bộ lọc Intent.

Đây là đoạn mã xác định nhiều bộ lọc Intent trong tệp AndroidManifest.xml để xử lý nhiều Intent.

```
<activity android:name=".MainActivity">  
    <intent-filter>  
        <action android:name="android.intent.action.MAIN" />  
        <category android:name="android.intent.category.LAUNCHER" />  
        <data android:mimeType="text/plain"/>  
    </intent-filter>  
    </activity>  
    <activity android:name=".ResultActivity">  
        <intent-filter>  
            <action android:name="android.intent.action.SEND"/>  
            <category android:name="android.intent.category.DEFAULT"/>  
            <data android:mimeType="text/plain"/>  
        </intent-filter>  
    </activity>
```

Ý nghĩa đoạn mã trên, activity “MainActivity” sẽ activity khi bắt đầu ứng dụng vì chúng ta đã khai báo activity này bằng cách sử dụng các thuộc tính danh mục MAIN action và LAUNCHER trong bộ lọc (<intent-filter>).

MAIN - Cho biết activity nào để khởi động chính của ứng dụng, có nghĩa là sẽ chạy activity có action MAIN khi người dùng khởi chạy ứng dụng lần đầu bằng biểu tượng icon LAUNCHER.

LAUNCHER - Chỉ ra biểu tượng activity nào sẽ được đặt trên danh sách ứng dụng trên màn hình chính. Trong trường hợp nếu <activity> không chỉ định biểu tượng, thì hệ thống sẽ sử dụng biểu tượng từ phần tử <application>.

Hai phần tử (MAIN, LAUNCHER) này phải được ghép nối với nhau để activity xuất hiện trong trình khởi chạy ứng dụng.

Activity thứ hai “ResultActivity” nhằm giúp chúng ta chia nội dung. Người dùng có thể mở activity này bằng cách điều hướng từ MainActivity và họ cũng có thể nhấp trực tiếp từ một ứng dụng khác bằng cách sử dụng Implicit Intent đang nối với một trong hai filter đã khai báo.

## 6.4 SỬ DỤNG CÁC INTENT

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

**Bảng 6.2: Các phương thức khởi động thành phần**

Tên phương thức	Mô tả
startActivity(Intent)	Thực thi Activity theo mô tả trong Intent, không lấy giá trị trả về.
startActivityForResult(Intent)	Thực thi Activity theo mô tả trong Intent, có lấy giá trị trả về.
sendBroadcast(Intent)	Phân phát Intent tới bất cứ thành phần Receiver nào quan tâm.
startService(Intent)	Chạy một Service
bindService(Intent, ServiceConnection, int)	Bind một Service

### 6.4.1 Explicit Intent thực thi Activity

Như đã trình bày ở phần trên, Intent có thể dùng thuộc tính phụ để chỉ định đích danh tên Activity sẽ được mở. Để thực hiện điều này, lớp Intent cung cấp các phương thức:

- setComponent(ComponentName)
- setClass(Context, Class)
- setClassName(Context, String)
- setClassName(String, String).

Cách gọi này chỉ có thể được dùng để gọi các Activities trong cùng một ứng dụng.

Ví dụ:

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

```
startActivity(intent);
```

## 6.4.2 Implicit Intent thực thi Activity

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

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

Ví dụ:

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

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

## 6.5 GỌI HIỂN THỊ ACTIVITY

---

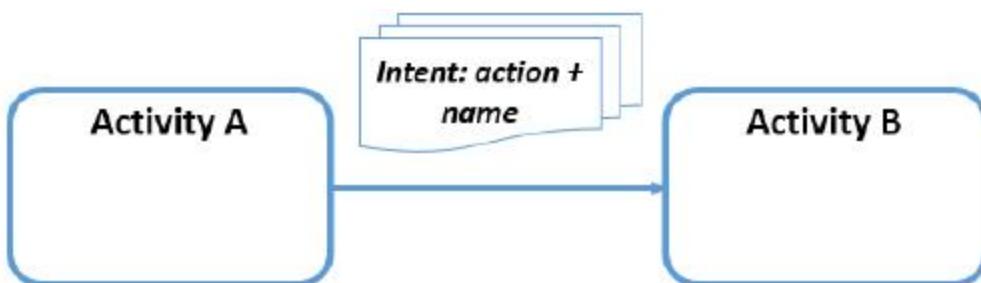
Trong phần này ta sẽ tìm hiểu cách thức gọi hiển thị và có trau đổi dữ liệu giữa hai Activity trong ứng dụng Android. Thông thường khi một Activity đang làm việc với người dùng có các tình huống sau có thể diễn ra :

1. Activity A đang làm việc, trong A có chức năng gọi và hiển thi Activity B thực hiện chức năng của riêng B.

2. Activity A đang làm việc, A có chức năng gọi-hiển thị Activity B thực hiện và có gởi dữ liệu qua cho B.
3. Activity A đang làm việc, A có chức năng gọi-hiển thị Activity B thực hiện (có thể gởi hoặc không gởi dữ liệu cho B). Khi B thực hiện xong thì gởi dữ liệu về cho A làm việc tiếp. Do đó khi B thực hiện thì A sẽ chờ kết quả trả về từ B.

### 6.5.1 Gọi hiển thị không có trao đổi dữ liệu

Activity A đang hoạt động, khi đó người dùng chọn một chức năng nào đó trên giao diện A, ứng dụng sẽ gọi và hiển thị Activity B để tương tác với người dùng.



**Hình 6.2: Không truyền dữ liệu**

Để gọi Activity B trong một chức năng nào đó của A, có thể là phần xử lý sự kiện click của button trong A như sau:

```

Intent activityIntent = new Intent(this, ActivityB.class);
startActivity(activityIntent);
  
```

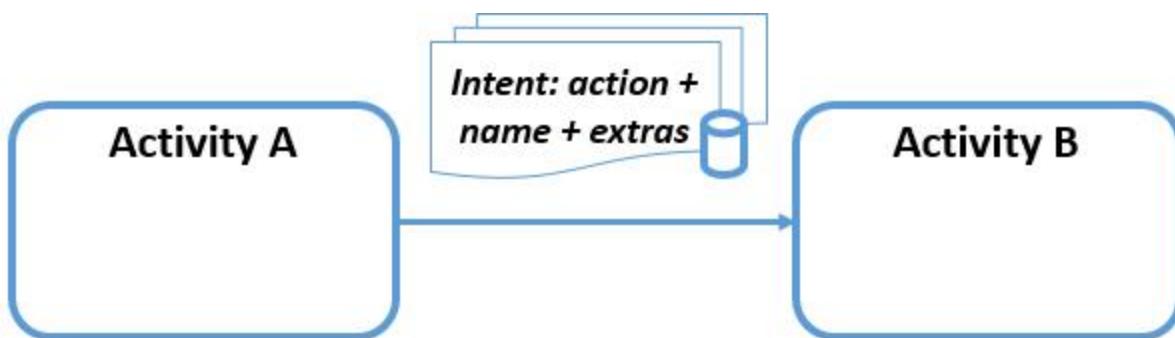
Để thành phần Activity B hoạt động được trong hệ thống thì phải đăng ký trong AndroidManifest.xml. Phần khai báo cho Activity B như sau:

```

<activity android:name = "ActivityB">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
  
```

## 6.5.2 Gọi hiển thị và truyền dữ liệu cho Activity con

Trong tình huống này, Activity A ngoài việc gọi và hiển thị Activity B thì A sẽ gởi dữ liệu sang cho Activity B xử lý. Để thực hiện thì Activity A tạo một đối tượng Bundle chứa dữ liệu cần gởi cho Activity B, sau đó gắn Bundle này kèm theo đối tượng Intent. Khi Activity B thực thi thì sẽ nhận được dữ liệu gởi từ Activity A.



**Hình 6.3: Gởi dữ liệu qua Activity B.**

❖ Các bước để đưa dữ liệu vào Intent, phần này thực thi trong Activity A:

1. Tạo đối tượng Bundle

```
Bundle data = new Bundle();
```

2. Đưa các dữ liệu dạng key-value vào đối tượng Bundle thông qua các phương thức putXYZ(), trong đó XYZ là các kiểu dữ liệu cơ bản như Int, String, Double...

```
data.putString("Name", "Nguyen Ha Giang");
```

```
data.putInt("DOB", 1978);
```

3. Đưa đối tượng Bundle vào trong đối tượng Intent

```
activityIntent.putExtras(data);
```

❖ Các bước thực hiện khi Activity B nhận dữ liệu:

1. Lấy đối tượng Intent do A gởi qua

```
Intent intent = getIntent();
```

2. Lấy đối tượng bundle từ Intent

```
Bundle info = intent.getExtras();
```

3. Lấy các dữ liệu từ Bundle thông qua các phương thức getXYZ(), trong đó XYZ là các kiểu dữ liệu cơ bản. Khi gọi phương thức getXYZ() ta truyền vào khóa của dữ liệu và nhận được giá trị.

### 6.5.3 Gọi hiển thị và truyền/nhận dữ liệu với Activity con

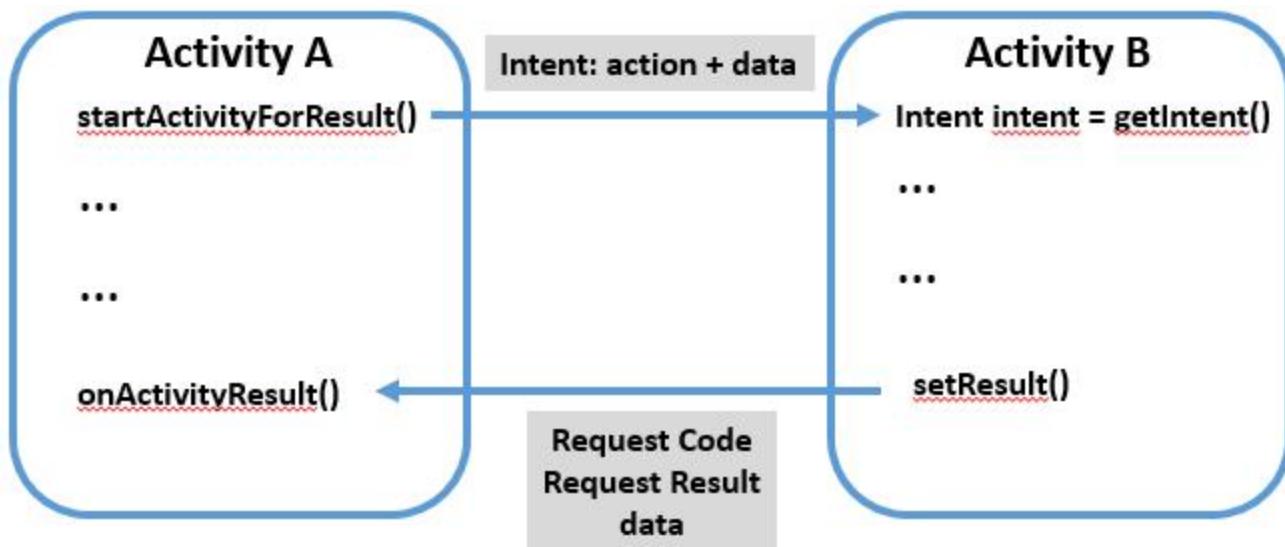
Trong tình huống này, Activity A khi gọi và hiển thị Activity B thì A chờ lấy dữ liệu trả về từ Activity B. Ở mô hình này có sự phối hợp hoạt động của hai Activity như sau: Activity A đang hoạt động, A gọi và gửi dữ liệu cho Activity B thông qua chức năng nào đó. Lúc này Activity B hiển thị và tương tác với người dùng, sau khi B xử lý xong chức năng của nó thì B sẽ gửi dữ liệu trả về cho Activity A. Do đó Activity A sẽ chờ lấy dữ liệu trả về từ Activity B.

Phần xử lý gọi hiển thị B được đặt trong lớp A như sau, do phần trước đã đề cập đến truyền dữ liệu từ A sang B nên phần này ta không xét tới.

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

```
final int result = 1;
```

```
// nếu có truyền dữ liệu sang B thì làm ở đây
```



**Hình 6.4:** Gởi-nhận dữ liệu giữa hai Activity.

```
// gọi Activity B hiển thị và chờ kết quả trả về
```

```
startActivityForResult(intent, result);
```

Do Activity A có nhu cầu nhận dữ liệu trả về từ B nên A phải viết phần xử lý khi B hoàn thành và trả về cho A. Phần xử lý này được thực thi trong phương thức onActivityResult() của Activity A.

```
@override
```

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    super.onActivityResult(requestCode, resultCode, data);  
  
    // lấy Bundle chứa dữ liệu  
  
    Bundle bundle = data.getExtras();  
  
    int data1 = bundle.getInt("<tên dữ liệu 1>");  
  
    int data2 = bundle.getInt("<tên dữ liệu 2>");  
  
    String data3 = bundle.getString("<tên dữ liệu 3>");  
  
    ...  
}
```

Phần xử lý trong Activity B khi xử lý xong và chuẩn bị dữ liệu để gửi về cho A.

```
// gửi dữ liệu về Activity trước  
  
Intent intent = new Intent();  
  
Bundle bundle = new Bundle();  
  
// gửi dữ liệu vào bundle  
  
bundle.putInt("<tên dữ liệu 1>", data1);  
  
bundle.putInt("<tên dữ liệu 2>", data2);  
  
bundle.putString("<tên dữ liệu 3>", data3);  
  
intent.putExtras(bundle); // gửi kèm dữ liệu  
  
setResult(RESULT_OK, intent); // gửi kết quả về  
  
finish(); // đóng Activity
```

# TÓM TẮT

Trong bài học này giới thiệu một thành phần Intent rất quan trọng trong kiến trúc của ứng dụng Android. Intent là cơ chế truyền thông điệp cho phép kết nối các thành phần của ứng dụng Android lại với nhau như Activity, Broadcast Receiver, Service. Ngoài ra Intent cho phép gọi các ứng dụng hay các thành phần có sẵn trong hệ thống Android. Song song với việc lập trình sử dụng Intent, bài học cung cấp có cơ chế để truyền/nhận dữ liệu giữa các thành phần với nhau, một trong cách đơn giản để truyền/nhận dữ liệu là dùng Bundle chứa dữ liệu đã định dạng kèm theo đối tượng Intent. Mô hình truyền/nhận dữ liệu được giới thiệu cụ thể theo 3 cách giữa hai Activity: gởi từ Activity qua Activity khác, nhận dữ liệu từ Activity khác, gởi/nhận dữ liệu giữa hai Activity. Tùy theo yêu cầu cụ thể của ứng dụng mà người lập trình có thể chọn tương ứng mô hình truyền/nhận dữ liệu nào đó.

## BÀI TẬP

**Câu 1:** Viết ứng dụng Android có 2 Activity theo yêu cầu sau:

Activity A (main Activity) cho phép người dùng nhập vào 2 số nguyên, sau đó có chức năng gởi 2 số nguyên này cho Activity B.

Activity B: nhận 2 số nguyên từ Activity A, cho phép người dùng chọn các phép toán cộng, trừ, nhân, chia giữa hai số. Tính và hiển thị kết quả cho người dùng.

**Câu 2:** Viết ứng dụng Android có 2 Activity, với các yêu cầu sau:

Activity A cho phép người dùng nhập vào 3 số nguyên, sau đó có chức năng gởi 3 số nguyên này cho Activity B.

Activity B: nhận 3 số nguyên từ A, cho phép chọn một trong 3 chức năng: tìm số lớn nhất trong 3 số, tìm số nhỏ nhất, giải phương trình với 3 số lần lượt là a, b, và c. Sau khi thực thi xong chức năng thì gởi kết quả về cho A, và cũng cho A biết chức năng nào đã thực thi. A sẽ hiển thị kết quả của chức năng tương ứng đã chọn trong B.

**Câu 3:** Viết ứng dụng NotePad đơn giản, trong đó có hai Activity, Activity A chính chứa EditText cho phép người dùng soạn thảo nội dung văn bản. A có chức năng

option, option này chứa trong Activity B, có các mục chọn màu văn bản, màu nền văn bản, font size... Sau khi chọn xong thì các option này có hiệu lực trên Activity A.

**Câu 4:** Viết ứng dụng cho phép nhập vào số điện thoại và thực hiện cuộc gọi

**Câu 5:** Viết ứng dụng cho phép nhập vào số điện thoại và nội dung tin nhắn. Thực thi việc gửi tin nhắn có nội dung đã nhập cho số điện thoại đó.

**Câu 6:** Viết ứng dụng đơn giản cho phép gửi email, thông tin cần nhập là: địa chỉ email, tiêu đề, và nội dung. Thực thi việc gửi email đến địa chỉ đã cho.

**Câu 7:** Viết ứng dụng cho phép nhập vào chuỗi nội dung tìm kiếm và thực thi tìm kiếm thông qua Google.

# BÀI 7: LẬP TRÌNH VỚI SQLITE

Học xong bài này, người học cần nắm được các nội dung sau.

- *Kiến thức cơ bản về cơ sở dữ liệu SQLite: Đặc điểm, tính năng, kiểu dữ liệu hỗ trợ, cách xây dựng dữ liệu.*
- *Mô hình làm việc với SQLite trong ứng dụng Android.*
- *Các lớp thư viện để lập trình kết nối SQLite trong ứng dụng Android.*
- *Hiểu được ví dụ minh họa kết nối CSDL SQLite.*

## 7.1 CƠ SỞ DỮ LIỆU SQLITE

SQLite là dạng cơ sở dữ liệu nhúng có kích thước nhỏ gọn phù hợp với ứng dụng chạy trên thiết bị di động có dung lượng bộ nhớ hạn chế. SQLite được xây dựng dựa trên chuẩn SQL-92, tên gọi là "lite" nói lên đúng bản chất của cơ sở dữ liệu là gọn nhẹ kích thước khoảng gần 300KB. Do kích thước nhỏ gọn như vậy nên việc truy xuất dữ liệu nhanh chóng, không chiếm dụng quá nhiều tài nguyên hệ thống.

Ngoài ra SQLite là dạng phần mềm mã nguồn mở nên không bị hạn chế về vấn đề tác quyền. Chính vì vậy nó được dùng nhiều bởi các công ty như Apple, Adobe, Google, Sun, Symbian... Trong lập trình Android thì việc dùng SQLite làm cơ sở dữ liệu rất phổ biến và thực tế chứng minh SQLite rất thích hợp cho việc lưu trữ dữ liệu client trên Android.

SQLite tuy gọn nhẹ hơn so với các hệ cơ sở dữ liệu khác nhưng cũng không khác biệt nhiều. SQLite cũng sử dụng ngôn ngữ truy vấn SQL (SELECT, INSERT, DELETE...). SQLite có một hệ thống câu lệnh SQL đầy đủ với các trigger, transaction như các hệ cơ sở dữ liệu khác. SQLite như một bản thu nhỏ của so với các hệ CSDL khác, vì vậy nó không thể có đầy đủ các chức năng trên chiếc điện thoại di động của bạn.

SQLite là một lựa chọn thích hợp dành cho ứng dụng trên hệ điều hành Android. Ngoài dung lượng lưu trữ nhỏ gọn, SQLite còn cho phép sử dụng Unicode, kiểu dữ liệu không được cài đặt trong một số phiên bản Android.

### 7.1.1 Đặc trưng cơ bản của SQLite

- SQLite được hiện thực từ tiêu chuẩn SQL-92 của một ngôn ngữ SQL nhưng vẫn còn chứa một số khiếm khuyết.
- Tuy SQLite hỗ trợ triggers nhưng bạn không thể viết trigger cho View. Hoặc SQLite không hỗ trợ lệnh ALTER TABLE, do đó, bạn không thể thực hiện chỉnh sửa hoặc xóa cột trong bảng.
- SQLite không hỗ trợ ràng buộc khóa ngoại, các transactions lồng nhau, phép kết RIGHT OUTER JOIN, FULL OUTER JOIN.
- SQLite sử dụng kiểu dữ liệu khác biệt so với hệ quản trị cơ sở dữ liệu tương ứng. Bạn có thể insert dữ liệu kiểu string vào cột kiểu integer mà không gặp phải bất kỳ lỗi nào.
- Vài tiến trình hoặc luồng có thể truy cập cùng một cơ sở dữ liệu. Việc đọc dữ liệu có thể chạy song song, còn việc ghi dữ liệu thì không được phép chạy đồng thời.
- Ngoài các khiếm khuyết trên thì Sqlite cung cấp cho người dùng gần như đầy đủ các chức năng mà một hệ cơ sở dữ liệu cần có như tạo database; tạo bảng; thêm, xóa, sửa dữ liệu.

### 7.1.2 Lập trình với SQLite

Để thao tác với cơ sở dữ liệu lưu trữ SQLite trên Android ta sẽ làm việc với hai loại đối tượng:

1. **SQLiteDatabase**: Đối tượng dùng để tạo, nâng cấp, đóng mở kết nối trên một cơ sở dữ liệu.
2. **SQLiteDatabase**: Đối tượng dùng để thực thi các câu lệnh SQL trên một cơ sở dữ liệu.

Một ứng dụng sẽ có một đối tượng **SQLiteDatabase** để tạo mới, nâng cấp cũng như đóng mở cơ sở dữ liệu. Thông qua việc sử dụng đối tượng **SQLiteDatabase** sẽ

Lấy về một đối tượng kiểu SQLiteDatabase, đây chính là thể hiện của cơ sở dữ liệu cần thao tác. SQLiteOpenHelper hỗ trợ hai phương thức để lấy về một đối tượng SQLiteDatabase là:

- getReadableDatabase(): Lấy về một đối tượng SQLiteDatabase ở dạng "chỉ đọc".
- getWritableDatabase(): Lấy về một đối tượng SQLiteDatabase ở dạng "đọc và ghi".

Để tạo mới một cơ sở dữ liệu ta sẽ kế thừa lớp SQLiteOpenHelper.

SQLiteOpenHelper hỗ trợ cho chúng ta 3 phương thức chính:

- Phương thức khởi tạo (constructor): Phương thức này cung cấp các tham số cần thiết để SQLiteOpenHelper có thể làm việc với cơ sở dữ liệu trên Android.
- Phương thức onCreate() được dùng để tạo file lưu trữ cơ sở dữ liệu, tạo các bảng có trong cơ sở dữ liệu cũng như nhập các dữ liệu ban đầu.
- Phương thức onUpgrade() được dùng để giúp bạn cập nhật lại các bảng (table) trong cơ sở dữ liệu.

Mỗi cơ sở dữ liệu trên một ứng dụng Android có nhiều phiên bản. Các phiên bản này có thể tương ứng với các phiên bản của ứng dụng.

### 7.1.3 Kiểu dữ liệu lưu trữ

SQLite hỗ trợ các kiểu dữ liệu như sau:

1. NULL: giá trị null.
2. INTEGER: kiểu số nguyên có dấu, lưu trữ trong 1, 2, 4, 6 hay 8 byte phụ thuộc vào độ lớn của dữ liệu.
3. REAL: giá trị số thực 8 byte.
4. TEXT: dạng text có hỗ trợ Unicode.
5. BLOB: lưu trữ chính xác dữ liệu nhập vào dạng blob.

Trong SQLite việc quản lý kiểu dữ liệu có khác biệt với so với các DBMS thông thường. Có thể cho phép insert dữ liệu chuỗi vào cột số nguyên hay ngược lại.

## 7.1.4 Các lớp cơ bản trong gói SQLite

### 1. SQLiteOpenHelper

Tạo và cập nhật dữ liệu trong ứng dụng, phải tạo lớp kế thừa từ lớp này trong ứng dụng và override các phương thức:

- onCreate()
- onUpgrade()
- getReadableDatabase(), getWritableDatabase()
- close(), onOpen()

### 2. ContentValues

Dùng để lưu trữ dữ liệu dạng "key-value", key thể hiện cột trong bảng và value là giá trị tương ứng.

Thường dùng để insert và update record trong cơ sở dữ liệu.

### 3. SQLiteDatabase: cung cấp các phương thức thao tác trên CSDL

- insert(String tableName, String nullColumnHack, ContentValues contentValues)
- delete(String table, String whereClause, String[] whereArgs)
- update(String table, ContentValues contentValues, String whereClause, String[] whereArgs)
- query(String table, String[] columns, String selection, String[] Args, String groupByClause, String havingClause, String orderByClause, String NoOfRowsToBeFethed)
- rawQuery(String sql, String[] selectionArgs)

### 4. Cursor: chứa toàn bộ record trả về từ câu truy vấn. Một số phương thức thao tác trên dữ liệu trả về.

**Bảng 7.1: Các phương thức trên Cursor**

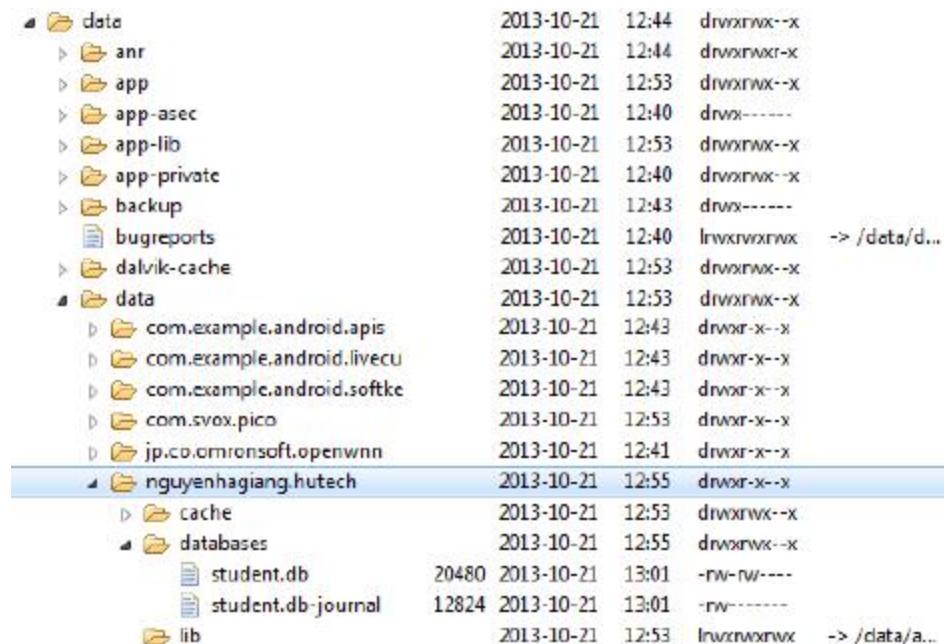
Phương thức	Ý nghĩa
moveToNext()	Di chuyển qua record kế tiếp
moveToPrevious	Di chuyển về record trước đó
moveToFirst	Di chuyển về đầu
moveToLast	Di chuyển về cuối
moveToPosition(int p)	Di chuyển đến vị trí p
isFirst()	Kiểm tra record ở đầu
isLast()	Kiểm tra record ở cuối
getInt(int columnIndex)	Lấy giá trị nguyên ở cột columnIndex
getString(int columnIndex)	Lấy chuỗi ở cột columnIndex
getDouble(int columnIndex)	Lấy giá trị thực ở cột columnIndex

### 7.1.5 Thư mục lưu trữ

Cơ sở dữ liệu SQLite được lưu trữ trong thư mục data của app trên Android. Cụ thể đường dẫn như sau:

```
/data/data/app name/databases/<database name>
```

Ví dụ: Ứng dụng có app name là fit.hutech và tập tin chứa cơ sở dữ liệu là student.db thì vị trí lưu trữ như hình vẽ.

**Hình 7.1: Vị trí lưu trữ dữ liệu.**

## 7.2 ỨNG DỤNG MINH HỌA

Trong phần này ta sẽ minh họa một ứng dụng quản lý thông tin sinh viên đơn giản. Cơ sở dữ liệu của ứng dụng chỉ duy nhất một bảng thể hiện như sau:

**Bảng 7.2: Bảng Student.**

<b>Student</b>		
studentID	INTEGER	Primary key autoincrement
studentName	TEXT	
mark	INTEGER	

### 7.2.1 Tạo lớp kế thừa từ SQLiteOpenHelper

Lớp này có nhiệm vụ tạo cơ sở dữ liệu, tạo bảng Student

```
public class DBCreator extends SQLiteOpenHelper {

    // Khai báo tên CSDL, tên bảng, tên version
    public static final String DATABASE_NAME = "student.db";

    public static final String TABLE_NAME = "Student";

    private static int DATABASE_VERSION = 1;

    public DBCreator(Context context)
    {
        // gọi constructor của lớp cơ sở để tạo database
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // sql tạo bảng dữ liệu
        String sql = "CREATE TABLE "+ TABLE_NAME +
                    " (studentID INTEGER PRIMARY KEY
```

```
AUTOINCREMENT, studentName TEXT, mark INTEGER);";  
// hiển thị log  
Log.d(DATABASE_NAME, "Create "+ TABLE_NAME);  
db.execSQL(sql);  
}  
  
@Override  
public void onUpgrade(SQLiteDatabase db, int oldVersion,  
int newVersion) {  
// TODO Auto-generated method stub  
}  
}  
} // class DBCreator
```

## 7.2.2 Tạo lớp chứa dữ liệu DTO là Student

Lớp dùng để chứa thông tin của một đối tượng sinh viên, tương ứng với một dòng dữ liệu trong bảng Student

```
public class Student  
{  
// data member  
private int id;  
private String name;  
private int mark;  
// constructor  
public Student(String name, int mark)  
{  
this.name = name;  
this.mark = mark;
```

```
}

public Student(int id, String name, int mark)

{

    this.id = id;

    this.name = name;

    this.mark = mark;

}

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 int getMark() {return mark; }

public void setMark(int mark) { this.mark = mark; }

}// class Student
```

### 7.2.3 Tạo lớp truy cập dữ liệu

Đây là lớp DAO, cho phép làm việc với cơ sở dữ liệu Student, lớp có biến đối tượng là DBCreator đã tạo trong bước trước.

```
public class DAO {

    // tạo đối tượng SQLiteDatabase để thao tác trên CSDL

    SQLiteDatabase db;

    // tạo đối tượng DBCreator

    DBCreator dbCreator;

    // tạo đối tượng kiểu context

    Context context;

    public DAO(Context context)
```

```
{  
    // tạo đối tượng Database  
    dbCreator = new DBCreator(context);  
    // trả về đối tượng cho phép write db  
    db = dbCreator.getWritableDatabase();  
  
}  
  
public int insertRecord(Student s) {  
    try {  
        // tạo câu query insert dữ liệu  
        String sql = "INSERT INTO "+ DBCreator.TABLE_NAME +  
                    "(studentName, mark) VALUES('"+ s.getName()+"','"+  
                    s.getMark()+"');"  
        // thực thi câu query  
        db.execSQL(sql);  
        return 1;  
    }  
    catch (Exception e) {  
        Log.e("DB", e.toString()+"DAO.Insert");  
        return -1;  
    }  
}  
  
// Phương thức xóa một sinh viên  
public int deleteRecord(int id)  
{
```

```
try
{
    // sử dụng hàm delete có sẵn trong SQLiteDatabase
    return db.delete(DBCreator.TABLE_NAME, "studentID=?",
        new String[] {" "+id});
}

catch (Exception e)
{
    Log.e("DB", e.toString()+" DAO.Delete");
    return -1;
}

public int updateRecord(Student s) {
    try {
        // tạo đối tượng ContentValues
        ContentValues values = new ContentValues();
        values.put("studentName", s.getName());
        values.put("mark", s.getMark());
        // gọi hàm update
        return db.update(DBCreator.TABLE_NAME, values, "studentID=?",
            new String[] {" "+s.getID()});
    }

    catch (Exception e)
    {
        Log.e("DB", e.toString()+" DAO.updateRecord");
    }
}
```

```
        return -1;  
    }  
}  
  
// lấy toàn bộ mẫu tin  
  
public Cursor getRecords()  
{  
    Cursor c = db.rawQuery("SELECT * FROM" + DBCreator.TABLE_NAME, null);  
    return c;  
}  
  
public void close()  
{  
    db.close();  
}  
}  
} // end DAO
```

### 7.2.4 Thao tác insert dữ liệu

Phần code này minh họa việc thêm dữ liệu là một đối tượng sinh viên vào trong CSDL. Code này trích ra trong một Activity có chức năng thêm đối tượng sinh viên.

Giả sử trong Activity này có hai EditText là edtName và edtMark, chứa thông tin họ tên và điểm do người dùng nhập vào.

```
String name = ((EditText)findViewById(R.id.edtName)).getText().toString();  
  
String tmp = ((EditText)findViewById(R.id.edtMark)).getText().toString();  
  
int mark = Integer.parseInt(tmp);  
  
DAO dao = new DAO(InsertActivity.this);  
  
Student s = new Student(name, mark);  
  
int i = dao.insertRecord(s);
```

```
if (i>=0) {  
  
    Toast.makeText(this, "1 record has been added",  
    Toast.LENGTH_LONG).show();  
  
}  
  
else {  
  
    Toast.makeText(this, "Record has not been added",  
    Toast.LENGTH_LONG).show();  
  
}
```

## 7.2.5 Hiển thị danh sách sinh viên

Để hiển thị danh sách sinh viên thì tạo một Activity có ListView để chứa nội dung danh sách sinh viên.

Đoạn code minh họa như sau:

```
ListView lv = (ListView)findViewById(R.id.list1);  
  
//tạo arraylist  
  
ArrayList<String> al = new ArrayList<String>();  
  
Cursor c = new DAO(this).getRecords();  
  
while (c.moveToNext()) {// đưa toàn bộ record vào list.  
  
    int id = c.getInt(0);  
  
    String name = c.getString(1);  
  
    int mark = c.getInt(2);  
  
    al.add(id+" "+name+" "+mark); // thêm vào list  
  
}  
  
// tạo adapter  
  
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
    android.R.layout.simple_list_item_1, al);  
  
lv.setAdapter(adapter); // thiết lập adapter cho ListView
```

# TÓM TẮT

Trong bài học này trình bày các phần cơ bản về cơ sở dữ liệu SQLite bao gồm các đặc điểm cơ bản, tính chất của SQLite, các kiểu dữ liệu hỗ trợ. Từ các kiến thức cơ bản đó thì bài học cung cấp cách lập trình kết nối với CSDL SQLite. Bao gồm các lớp cơ bản như `SQLiteOpenHelper`, `SQLiteDatabase`, `ContentValues`, `Cursor`. Phần cuối cùng của bài học là một ví dụ cụ thể về ứng dụng Android có thao tác với CSDL SQLite gồm : thêm, xóa, sửa, và tìm kiếm.

# BÀI TẬP

**Câu 1:** Viết ứng dụng minh họa form đăng nhập, dữ liệu xác thực đăng nhập được lưu trữ trong CSDL SQLite.

**Câu 2:** Viết ứng dụng TODO quản lý các công việc cần làm của cá nhân. Ứng dụng cho phép thêm, xóa, sửa, tìm kiếm và hiển thị toàn bộ công việc. Dữ liệu công việc được lưu trữ trong CSDL SQLite.

**Câu 3:** Viết ứng dụng quản lý các món ăn yêu thích: cho phép nhập vào tên món ăn, vật liệu (hay công thức nấu) và cách thức nấu món ăn đó. Ứng dụng hỗ trợ các chức năng thêm, xóa, sửa, tìm kiếm và hiển thị danh sách các món ăn.

**Câu 4:** Viết ứng dụng quản lý chi tiêu cá nhân. Cho phép thêm, xóa, sửa, xem, tính tổng các khoản chi và thu.

**Câu 5:** Viết ứng dụng Android minh họa quản lý mượn trả sách.

**Câu 6:** Viết ứng dụng M-Order cho phép order các món ăn, thức uống trong một nhà hàng.

**Câu 7:** Viết ứng dụng minh họa book vé máy bay.

**Câu 8:** Viết ứng dụng minh họa book tour du lịch.

**Câu 9:** Viết ứng dụng minh họa book phòng khách sạn.

**Câu 10:** Viết ứng dụng m-marketing, cho phép quản lý danh sách các khách hàng, mỗi khách hàng gồm các thông tin: họ tên, ngày sinh, địa chỉ, số điện thoại, địa chỉ

email. Mỗi khách hàng sẽ phân vào một nhóm nào đó để dễ dàng cho công việc tiếp thị, có các loại khách hàng như sau: {điện máy, thực phẩm, thời trang, vật dụng gia đình, mỹ phẩm}. Ứng dụng có các chức năng cơ bản như cho phép thêm, xóa, sửa, tìm kiếm, hiển thị toàn bộ danh sách. Ngoài ra ứng dụng có thêm 2 chức năng gởi tin nhắn SMS và gởi email đến các nhóm khách hàng.

**Câu 11:** Viết ứng dụng minh họa quản lý tin nhắn SMS, ứng dụng cho phép thêm, xóa, tìm kiếm và hiển thị các tin nhắn. Mỗi tin nhắn gồm các thông tin: {số điện thoại, ngày giờ nhận, nội dung tin nhắn}. Ngoài ra ứng dụng có chức năng thống kê số điện thoại nào nhắn tin nhiều nhất...

# BÀI 8: ỨNG DỤNG SERVICE

Học xong bài này, người học cần nắm được các nội dung sau.

- Hiểu cách thức hoạt động của dạng ứng dụng chạy nền trong Android là Service
- Nắm vững vòng đời hoạt động của thành phần Service Android và các trạng thái hoạt động của Service trong suốt vòng đời
- Hiểu được lớp nền tảng Service và các phương thức xử lý của lớp này
- Cách tạo một ứng dụng Android có Service bao gồm: xây dựng lớp kế thừa, thực thi các phương thức trạng thái và đăng ký lớp Service trong AndroidManifest.

## 8.1 THÀNH PHẦN SERVICE

Service là thành phần nền tảng của Android. Đôi khi ứng dụng chạy với khoảng thời gian lâu và không có hoặc hiếm khi có sự can thiệp từ người dùng. Tiến trình nền này có thể xử lý ngay cả khi thiết bị được sử dụng cho Activity/task khác.

Service cũng giống như các thành phần khác của ứng dụng (Activity, Broadcast Receiver...), sẽ chạy trên luồng chính của tiến trình ứng dụng. Điều này có nghĩa là nếu cần thực hiện một công việc nào đó tốn nhiều thời gian như chơi nhạc, tải dữ liệu từ trên mạng v.v thì bạn phải đưa công việc đó vào một luồng riêng để thực thi. Việc này sẽ tránh cho các công việc đang thực thi trên luồng chính không bị gián đoạn. Chúng ta cần xác định rõ các đặc trưng của Service:

- Một Service không phải là một tiến trình tách biệt. Một đối tượng Service không chạy trên tiến trình của riêng nó mà là chạy trên tiến trình của ứng dụng.
- Một Service không phải là một luồng. Điều này có nghĩa là mọi công việc sẽ được luồng chính thực thi. Chính vì vậy một đối tượng Service thường định nghĩa một luồng của riêng nó để thực hiện các công việc nhằm tránh tình trạng gián đoạn các công việc đang thực thi ở luồng chính.

Một Service có thể sử dụng theo hai cách sau:

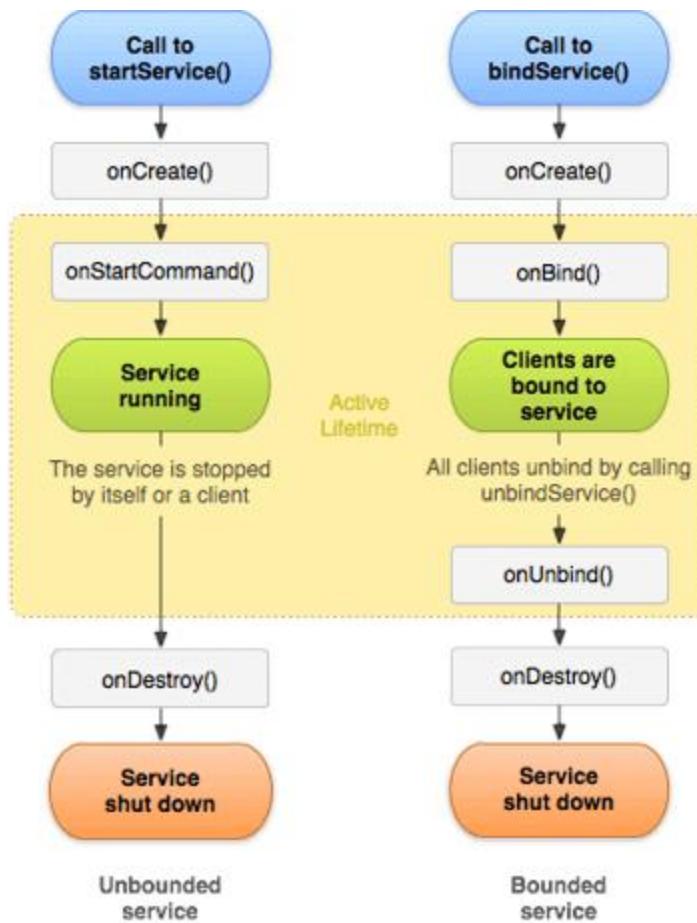
1. Một Service có thể được dùng để thực thi một công việc nền mà không cần hiển thị giao diện người dùng. Loại Service này được bắt đầu và được cho phép hoạt động cho đến khi một người nào đó dừng lại hoặc tự ngắt. Ở chế độ này, Service được bắt đầu bằng cách gọi `Context.startService()` và dừng bằng lệnh `Context.stopService()`. Một Service có thể tự ngắt bằng lệnh `Service.stopSelf()` hoặc `Service.stopSelfResult()`. Mỗi Service chỉ có một thể hiện duy nhất, do đó chỉ cần một lệnh `stopService()` để ngừng một Service lại cho dù lệnh `startService()` được gọi ra bao nhiêu lần.
2. Một Service còn có thể được sử dụng để cung cấp một tính năng nào đó cho ứng dụng khác kết nối và sử dụng. Một ứng dụng có thể thiết lập một đường truyền tới đối tượng Service và sử dụng đường kết nối đó để điều khiển Service. Kết nối này được thiết lập bằng cách gọi lệnh `Context.bindService()` và được đóng lại bằng cách gọi lệnh `Context.unbindService()`. Nhiều ứng dụng có thể kết nối tới cùng một đối tượng Service. Nếu Service được một ứng dụng kết nối đến vẫn chưa được chạy thì lệnh `bindService()` có thể tùy ý khởi chạy nó.

Hai chế độ này thì không tách biệt toàn bộ, có thể kết nối với một Service đã được bắt đầu với lệnh `startService()`.

Ví dụ: Một Service nghe nhạc ở chế độ nền có thể được bắt đầu bằng cách gọi lệnh `startService()` cùng với một đối tượng Intent xác định bài hát cần chơi. Sau đó, có thể người sử dụng muốn kiểm soát trình chơi nhạc hoặc biết thêm thông tin về bài hát hiện tại đang chơi, thì sẽ có một Activity tạo lập một đường truyền tới Service bằng cách gọi `bindService()`. Trong trường hợp này, `stopService()` sẽ không thực sự ngừng Service cho đến khi liên kết cuối cùng được đóng lại.

### **8.1.1 Vòng đời của Service**

Hai vòng lặp quan trọng trong vòng đời của một đối tượng Service cục bộ:



**Hình 8.1: Vòng đời của Service.**

1. Vòng đời toàn diện (entire lifetime): Bắt đầu từ lúc gọi phương thức `onCreate()` đến lúc gọi phương thức `onDestroy()`. Cũng giống như Activity, đối tượng Service sẽ khởi tạo các giá trị tại phương thức `onCreate()` và dọn dẹp bộ nhớ tại phương thức `onDestroy()`.

Ví dụ: Một đối tượng Service dùng để chơi nhạc sẽ tạo một luồng để chơi nhạc ở phương thức `onCreate()` và ngưng luồng chơi nhạc này ở phương thức `onDestroy()` cũng như dọn dẹp bộ nhớ trước khi hủy đối tượng.

2. Vòng đời thực thi (active lifetime): Bắt đầu từ lúc gọi phương thức `onStart()`.

Ví dụ: Phương thức `onStart(...)` sẽ đón đối tượng Intent được gởi đi khi khởi động Service bằng phương thức `startService(...)`. Với đối tượng Service dùng để chơi nhạc sẽ gởi tên bài hát cần phát trong đối tượng Intent.

Nếu một đối tượng Service cho phép một ứng dụng khác kết nối thì phải cài đặt 3 phương thức sau:

1. IBinder onBind(Intent intent): Khi một đối tượng muốn tạo kết nối đến một đối tượng Service thì gọi phương thức Context.bindService(...) và gửi đi một đối tượng Intent. Phương thức onBind() sẽ được gọi để xử lý yêu cầu kết nối này. Kết quả sẽ trả về các kênh giao tiếp mà đối tượng cản kết nối có thể sử dụng để tương tác với Service.
2. boolean onUnbind(Intent intent): Phương thức này tương tự như phương thức onBind(). Tuy nhiên nó sẽ được gọi khi có một đối tượng gọi phương thức Context.unbindService() để ngắt kết nối với Service. Lúc này phương thức onUnbind() sẽ được gọi để xử lý yêu cầu ngắt kết nối đến Service.
3. void onRebind(Intent intent): Phương thức này được gọi khi có một đối tượng mới muốn kết nối đến Service.

### **8.1.2 Khuôn mẫu chung của lớp thực thi Service**

Để xây dựng một ứng dụng nền là Service trong Android thì phải thực thi lớp kế thừa từ Service trong Android SDK.

Lớp kế thừa này có dạng như sau:

```
public class MyService extends Service {  
    // phần khai báo dữ liệu riêng và các phương thức xử lý khác trong lớp service  
    // ....  
    // phần override các phương thức callback của lớp cơ sở Service  
    @Override  
    public void onCreate() {  
        // TODO Auto-generated method stub  
    }  
    @Override  
    public void onDestroy() {  
        // TODO Auto-generated method stub  
    }  
}
```

```
}

@Override

public void onStart(Intent intent, int startId) {

// TODO Auto-generated method stub

}

@Override

public IBinder onBind(Intent intent) {

// TODO Auto-generated method stub

return null;

}

}// end of MyService
```

Trong phần thực thi của lớp Service này các phương thức dạng callback của lớp cơ sở Service cần phải override, các phương thức này là phần chính trong các giai đoạn của vòng đời hoạt động của các service:

- onCreate()
- onDestroy()
- onBind()
- onStart()

### 8.1.3 Khởi tạo Service

Trong android để tạo một dịch vụ (service), chúng ta phải tạo một lớp con của Service hoặc sử dụng một trong các lớp con hiện có. Trong android, thành phần ứng dụng chẳng hạn như một activity có thể khởi động service bằng cách gọi startService(), dẫn đến việc gọi phương thức onStartCommand() của service này.

Sau đây là ví dụ tạo một dịch vụ (service), chúng ta phải tạo một lớp con

```
public class SampleService extends Service {

@Override

public int onStartCommand(Intent intent, int flags, int startId) {

//TODO write your own code
```

```
        return Service.START_NOT_STICKY;  
    }  
  
    @Override  
    public IBinder onBind(Intent intent) {  
        //TODO for communication return IBinder implementation  
        return null;  
    }  
}
```

### 8.1.4 Đăng ký Service vào tệp Manifest

Khi chúng ta t null; service, chúng ta cần service, chúng ta có trong t chúng ta c android bchúng ta c android bchúng ta cice> như dưới đây

```
<manifest ... >  
    ...  
    <application ... >  
        <service android:name=".SampleService" />  
    </application>  
    ...  
</manifest>
```

### 8.1.5 Chạy một Service

Trong android, một thành phần android, activity, service hoặc broadcast có thể là android, activity, service hoặc vice" />rtService(). Sau đây là đoạn mã mẫu về việc bắt đầu một service bằng phương thức startService.

```
Intent intent = new Intent(this, MyService.class);
```

```
startService(intent);
```

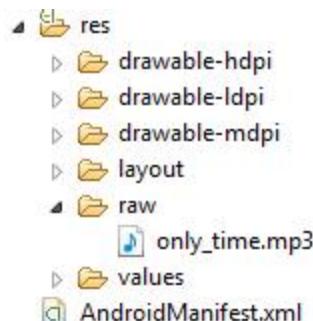
## **8.2 ỨNG DỤNG MẪU SERVICE: CHƠI NHẠC**

Trong phần này minh họa một ứng dụng đơn giản cho phép chạy một tập tin mp3 mặc định có sẵn trong ứng dụng. Phần chơi tập tin nhạc do một Service đảm nhận, để bật/tắt bài hát thì ứng dụng cung cấp UI thông qua một Activity.

Các bước thực hiện tóm lược như sau:

1. Tạo ứng dụng Android.
2. Import file âm nhạc mp3 vào trong project:

Tạo một thư mục có tên raw trong thư mục resource là res.



**Hình 8.2: Thư mục raw chứa file nhạc**

3. Tạo layout cho Activity chính của ứng dụng, Activity này có chức năng start/stop bài hát.

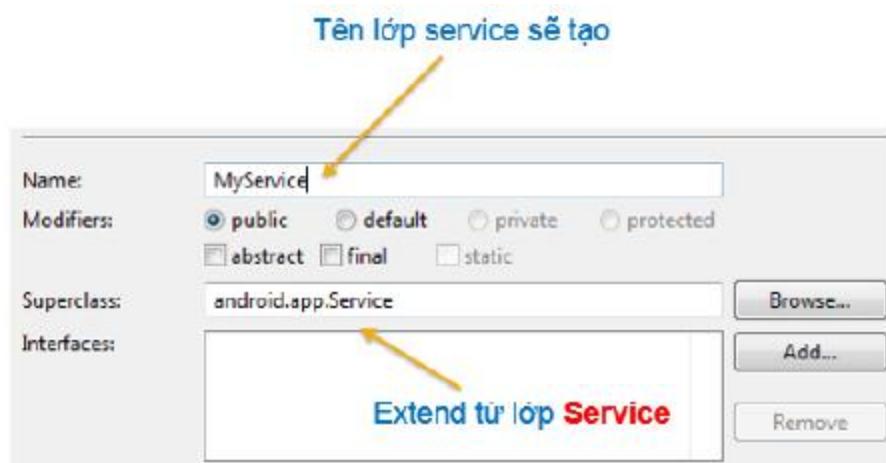
```
<LinearLayout
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center_horizontal" >
    <TextView android:text="Services Demo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="35dp" >
    </TextView>
```

```

<ToggleButton android:text="ToggleButton"
    android:id="@+id/toggleButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="doService" >
</ToggleButton>
</LinearLayout>

```

4. Tạo một service, lớp này bắt buộc phải kế thừa từ lớp Service cơ sở trong Android SDK.



**Hình 8.3: Tạo lớp Service.**

5. Thực thi các phần xử lý trong lớp MyService, cụ thể là override các phương thức callback như onCreate, onDestroy, và onStart. Để phát bài hát mp3, ta tạo một đối tượng của lớp MediaPlayer.

```

public class MyService extends Service {
    private static final String TAG = "HaGService";
    private MediaPlayer player;
    @Override
    public IBinder onBind(Intent intent) {
        // TODO Auto-generated method stub
        return null;
    }
    @Override

```

```
public void onCreate() {  
    // TODO Auto-generated method stub  
}  
  
@Override  
  
public void onDestroy() {  
    // TODO Auto-generated method stub  
}  
  
@Override  
  
public void onStart(Intent intent, int startId) {  
    // TODO Auto-generated method stub  
}  
}  
} // end of MyService
```

#### 6. Viết các phần xử lý cho phương thức onCreate()

Bao gồm các thao tác:

- Hiển thị thông báo "Service created"
- Hiển thị thông tin debug trong LogCat
- Load file mp3 từ resource vào đối tượng player
- Thiếp lập cờ loop

Đoạn code như sau:

```
@Override  
  
public void onCreate() {  
    // TODO Auto-generated method stub  
  
    Toast.makeText(this, "Service created", Toast.LENGTH_LONG).show();  
  
    Log.d(TAG, "onCreate");  
  
    player = MediaPlayer.create(this, R.raw.only_time);  
  
    player.setLooping(false);  
}
```

7. Thực thi hàm onStart và onDestroy

```
@Override  
public void onStart(Intent intent, int startId) {  
    // TODO Auto-generated method stub  
    Toast.makeText(this, "Service started", Toast.LENGTH_LONG).show();  
    Log.d(TAG, "onStart");  
    player.start(); // chạy file nhạc  
}  
  
@Override  
public void onDestroy() {  
    // TODO Auto-generated method stub  
    Toast.makeText(this, "Service stopped", Toast.LENGTH_LONG).show();  
    Log.d(TAG,"onDestroy");  
    player.stop(); // dừng file nhạc  
}
```

8. Viết phần xử lý cho sự kiện onClick của ToggleButton trong Activity chính của ứng dụng

```
public void doService(View view)  
{  
    // tham chiếu đến ToggleButton  
    ToggleButton btn = (ToggleButton) view;  
    if (btn.isChecked()==true) {  
        startService( new Intent(this,MyService.class) );  
    }  
    else {
```

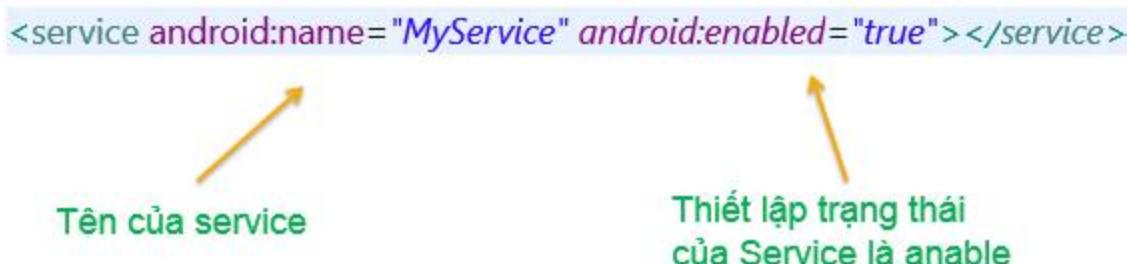
```
stopService( new Intent(this,MyService.class) );
```

```
}
```

```
}
```

9. Khai báo service trong AndroidManifest.xml, đây là phần khai báo bắt buộc, vì service là thành phần chính của ứng dụng Android.

Trong thẻ service cần khai báo thuộc tính android:enable là true để service có thể hoạt động được.



**Hình 8.4: Đăng ký Service trong AndroidManifest.xml**

10. Biên dịch và chạy ứng dụng trên máy ảo!

## TÓM TẮT

Bài học giới thiệu thành phần hoạt động nền trong hệ thống Android là Service. Thành phần Service thì không có giao diện tương tác với người dùng, chạy nền và hoạt động lâu dài trong hệ thống. Để điều khiển sự hoạt động của Service trong ứng dụng Android thì có thể bổ sung thêm Activity cho phép tương tác với người dùng, do Service không có thành phần giao diện. Bài học trình bày cụ thể vòng đời hoạt động của Service trong ứng dụng Android. Để xây dựng Service Android thì ứng dụng phải tạo lớp kế thừa từ lớp cơ sở Service được cung cấp trong Android SDK. Tùy theo yêu cầu cụ thể của ứng dụng mà thực thi các phần xử lý trạng thái tương ứng trong lớp kế thừa Service. Phần cuối cùng của bài học là ví dụ cụ thể cách xây dựng ứng dụng Service Android. Dựa trên ví dụ này người học có thể xem như là khuôn mẫu để xây dựng các Service khác.

## BÀI TẬP

**Câu 1:** Viết ứng dụng Android minh họa vòng đồi hoạt động của một Service, với mỗi trạng thái của Service thì dùng đối tượng Toast để hiện thị lên màn hình.

**Câu 2:** Viết ứng dụng MiniAlarm cho phép user quản lý (thêm, xóa, sửa) các mốc thời gian để cảnh báo, ứng dụng có Service chạy nền, kiểm tra nếu có mốc thời gian nào đến hẹn thì xuất cảnh báo cho người dùng, có thể dùng Toast, Notification và kết hợp với nhạc cảnh báo.

**Câu 3:** Viết ứng dụng quản lý danh sách ngày sinh nhật, cho phép thêm, xóa, sửa một entry sinh nhật. Ứng dụng có Service chạy nền, khi đến một ngày sinh nhật nào đó trong danh sách thì hiển thị thông báo, việc kiểm tra vào lúc 7h00 AM của mỗi ngày.

**Câu 4:** Viết ứng dụng Android cho phép đếm lùi một cột mốc thời gian nào đó trong tương lai. Mỗi ngày kiểm tra vào lúc 8h00 AM và thông báo cho người dùng biết là còn bao nhiêu nữa đến mốc thời gian đó.

**Câu 5:** Viết ứng dụng Android cho phép chạy các tập tin mp3 (được import vào trong project). Ứng dụng có giao diện Activity cho phép chọn các kiểu nghe như: nghe lần lượt các bài hát, nghe duy nhất một bài và lặp, nghe ngẫu nhiên các bài hát trong danh sách. Xây dựng Service thực thi bài hát theo các yêu cầu mà người dùng chọn trong Activity.

**Câu 6:** Viết ứng dụng Android cho phép nhận tin nhắn SMS đến máy và lưu trữ riêng trong CSDL SQLite, ứng dụng có giao diện Activity cho phép người dùng quản lý các danh sách các tin nhắn SMS đã lưu trong CSDL của ứng dụng.

**Câu 7:** Viết ứng dụng Android cho phép lưu trữ các cuộc gọi đến máy, các thông tin lưu trữ gồm: số máy gọi đến, có nhận hay không nhận và thời gian hội thoại nếu có nhận cuộc gọi. Các thông tin này lưu trữ trong CSDL SQLite và có giao diện cho phép quản lý danh sách các cuộc điện thoại đến này.

# BÀI 9: BROADCAST RECEIVER

Học xong bài này, người học cần nắm được các nội dung sau.

- Cơ chế xử lý các thông điệp phát sinh từ hệ thống Android hay phát sinh từ ứng dụng Android.
- Cơ chế tạo và phát sinh thông điệp từ ứng dụng.
- Nắm vững cách thực thi phát sinh thông điệp và cách thức bắt và xử lý thông điệp.
- Hiểu một số các Broadcast Intent cơ bản của hệ thống.
- Viết được các ứng dụng Android cho phép gửi và bắt các thông điệp được phát sinh.

## 9.1 CƠ CHẾ BROADCAST RECEIVER

Đây là một thành phần trong bốn thành phần chính (Activity, Service, Content Provider, Broadcast Receiver) của ứng dụng Android. Như tên gọi của thành phần này, cho phép nhận thông điệp (message) được gửi đi từ hệ thống hay từ các ứng dụng khác. Bản thân thông điệp là một dạng Broadcast Intent của ứng dụng Android. Một thông điệp khi được gửi quảng bá lên thì có thể có nhiều receiver nhận và đáp ứng lại với thông điệp đó, việc đáp ứng chính là có những xử lý tùy theo receiver cụ thể.

Một thành phần như Activity hay Service sử dụng phương thức sendBroadcast() trong lớp Context để phát sinh một sự kiện Broadcast. Tham số truyền vào cho phương thức này là một đối tượng Intent.

Các thành phần nhận thông điệp (receiver) của một Broadcast Intent cần thiết phải kế thừa từ lớp BroadcastReceiver trong Android SDK. Ta gọi lớp này là Broadcast Receiver, lớp này phải được đăng ký trong manifest thông qua thẻ Receiver, nhằm cho hệ thống biết lớp có đăng ký xử lý một Broadcast Intent nào đó.

### 9.1.1 Phân loại Broadcast

Có hai dạng Broadcast trong hệ thống Android:

- Broadcast phát sinh từ chính nền tảng Android, ví dụ thông điệp phát sinh khi có cuộc gọi đến, thông điệp phát sinh khi có tin nhắn gửi tới, thông điệp phát sinh khi nguồn pin yếu... Các ứng dụng chạy trên Android nếu quan tâm đến các thông điệp này thì phải đăng ký xử lý.
- Broadcast phát sinh từ các ứng dụng Android, ví dụ ứng dụng A sao khi thực thi xong một nhiệm vụ nào đó thì phát sinh thông điệp cho phép ứng dụng khác chẳng hạn như B biết và có xử lý tùy theo yêu cầu cụ thể của B.

### 9.1.2 Tạo một broadcast receiver

Đây là lớp xử lý khi thông điệp được phát sinh, lớp này là lớp kế thừa từ lớp cơ sở là BroadcastReceiver. Trong lớp này cần phải override phương thức onReceive(), phương thức này sẽ được gọi tự động khi Receiver nhận thông điệp.

Đoạn code minh họa việc tạo một Broadcast Receiver cơ bản

```
public class MyReceiver extends BroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent intent)
    {
        Toast.makeText(context, "Intent Detected.", Toast.LENGTH_LONG)
            .show();
    }
}
```

Trong đoạn code trên do tính chất minh họa nên chỉ xuất ra thông điệp dạng message box có nội dung "Intent Detected", còn với ứng dụng cụ thể thì phần xử lý sẽ đặt trong onReceive().

### 9.1.3 Đăng ký xử lý Broadcast Intent

Một ứng dụng Android muốn xử lý sự kiện nào đó thì phải đăng ký trong androidmanifest.xml.

Đoạn code sau minh họa việc đăng ký sự kiện "boot completed" của Android.

```
<application>
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <receiver android:name="MyReceiver">
        <intent-filter>
            <action android:name="android.intent.action.BOOT_COMPLETED">
            </action>
        </intent-filter>
    </receiver>
</application>
```

Trong thẻ Receiver có khai báo MyReceiver chính là tên của lớp xử lý khi có sự kiện BOOT\_COMPLETED được phát sinh.

### 9.1.4 Một số các Broadcast Intent

Trong phần trên đã minh họa chẵn và xử lý một sự kiện của hệ thống Android là android.intent.action.BOOT\_COMPLETED, đây chỉ là một trong số rất nhiều các sự kiện trong hệ thống Android.

**Bảng 9.1: Các Broadcast Intent cơ bản**

Broadcast Intent	Ý nghĩa
android.intent.action.BATTERY_LOW	Sự kiện khi pin yếu
android.intent.action.BATTERY_OKAY	Xuất hiện khi pin bình thường (sau khi sự kiện pin yếu phát sinh)
android.intent.action.BOOT_COMPLETED	Sự kiện khi hoàn tất việc boot máy
android.intent.action.DATA_SMS_RECEIVED	Khi nhận tin nhắn
android.intent.action.NEW_OUTGOING_CALL	Khi gọi điện thoại
android.intent.action.REBOOT	Xuất hiện khi máy boot
android.intent.action.DATE_CHANGED	Xuất hiện khi ngày thay đổi
android.intent.action.POWER_CONNECTED	Xuất hiện khi cắm vào nguồn điện
android.intent.action.POWER_DISCONNECTED	Xuất hiện khi rút nguồn điện
android.intent.action.PHONE_STATE	Xuất hiện khi có cuộc gọi đến máy.

## 9.1.5 Phát sinh Broadcast Intent

Như đã đề cập trong phần trước Broadcast Intent có thể phát sinh từ một ứng dụng Android bất kỳ chứ không chỉ là đặc quyền của hệ thống Android. Trong phần này sẽ tìm hiểu cách phát sinh ra một Broadcast Intent từ chính ứng dụng.

Các bước để tạo và phát sinh Broadcast Intent là:

1. Tạo một đối tượng Intent.
2. Thiết lập action cho Intent là chuỗi mô tả Broadcast Intent hay thông điệp.
3. Gọi phương thức sendBroadcast() với tham số là Intent đã tạo trong bước trên.

Đoạn code sau minh họa chức năng tạo riêng một Broadcast Intent của ứng dụng và phát sinh sự kiện Broadcast Intent trong một chức năng xử lý của một Activity nào đó, trong trường hợp này là hàm xử lý sự kiện click của một Button trên Activity.

```
public void broadcastIntent(View view)
{
    Intent intent = new Intent();
    intent.setAction("hutech.fit.CUSTOM_INTENT");
    sendBroadcast(intent);
}
```

Giả sử có một lớp Receiver là MyReceiver, lớp này sẽ xử lý sự kiện hutech.fit.CUSTOM\_INTENT đã phát sinh trong hàm trên. Ta có phần khai báo xử lý trong androidmanifest.xml như sau:

```
<application android:icon="@drawable/ic_launcher"
            android:label="@string/app_name"
            android:theme="@style/AppTheme" >
    <receiver android:name="MyReceiver">
        <intent-filter>
            <action android:name="hutech.fit.CUSTOM_INTENT">
            </action>
        </intent-filter>
    </receiver>
</application>
```

## 9.2 MINH HỌA BROADCASTRECEIVER

Để thiết lập Broadcast Receiver trong ứng dụng Android, ta cần thực hiện những việc sau.

- Tạo một BroadcastReceiver.
- Đăng ký BroadcastReceiver vào ứng dụng.

Để bắt đầu ta cần tạia cần tău roadcastReceiver vào ứng dụng. dụng android studio và đio tău à đio tă là BroadcastReceiver.

Bây giờ ta cần tạo tệp nội dung truyền phát mới để xác định provider thực tế của chúng ta, với tên tệp là MyBroadcastReceiver.java trong đường dẫn \src\main\java\com.<package của bạn>.broadcastreceiver và khai báo các phương thức liên quan bằng cách thực hiện nhấp chuột phải vào thư mục ứng dụng của bạn > Go > New > chọn Java Class và đặt tên là MyBroadcastReceiver.java.

Sau đó hãy mở tệp MyBroadcastReceiver.java và viết các đoạn mã sau:

```
package com.<tên package của bạn>.broadcastreceiver;  
import android.content.BroadcastReceiver;  
import android.content.Context;  
import android.content.Intent;  
import android.widget.Toast;
```

```
public class MyBroadcastReceiver extends BroadcastReceiver {  
  
    @Override  
  
    public void onReceive(Context context, Intent intent){  
  
        CharSequence iData = intent.getCharSequenceExtra("msg");  
  
        Toast.makeText(context,"I'm Received Message: "+  
iData,Toast.LENGTH_LONG).show();  
  
    }  
}
```

Bây giờ, hãy mở tệp activity\_main.xml từ đường dẫn \src\main\res\layout và viết mã sau.

```
<?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">
    <EditText
        android:id="@+id/txtMsg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:layout_marginTop="180dp"
        android:ems="10"/>
    <Button
        android:id="@+id/btnShow"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onClickShowBroadcast"
        android:layout_marginLeft="130dp"
        android:text="Show Broadcast"/>
</LinearLayout>
```

Tiếp theo, hãy mở tệp MainActivity.java từ đường dẫn \java\com.<package của bạn>.broadcastreceiver và viết phần sau để thực hiện các intent broadcast.

```
package com.<package của bạn>.broadcastreceiver;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
```

```
import android.os.Bundle;  
import android.view.View;  
import android.widget.EditText;  
  
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
    public void onClickShowBroadcast(View view){  
        EditText st = (EditText)findViewById(R.id.txtMsg);  
        Intent intent = new Intent();  
        intent.putExtra("msg", (CharSequence)st.getText().toString());  
        intent.setAction("com.<package của bạn>.CUSTOM_INTENT");  
        sendBroadcast(intent);  
    }  
}
```

Bây giờ ta cần đăng ký bộ broadcast receiver trong tệp manifest android (AndroidManifest.xml) bằng cách sử dụng thuộc tính <receiver> như dưới đây.

```
<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    package="com.<package của bạn>.broadcastreceiver">  
  
    <application  
        android:allowBackup="true"  
        android:icon="@mipmap/ic_launcher"
```

```
    android:label="@string/app_name"  
    android:roundIcon="@mipmap/ic_launcher_round"  
    android:supportsRtl="true"  
    android:theme="@style/AppTheme">  
        <activity android:name=".MainActivity">  
            <intent-filter>  
                <action android:name="android.intent.action.MAIN" />  
  
                <category android:name="android.intent.category.LAUNCHER" />  
            </intent-filter>  
        </activity>  
        <receiver android:name="MyBroadcastReceiver">  
            <intent-filter>  
                <action android:name="com.<package của bạn>.CUSTOM_INTENT">  
                </action>  
            </intent-filter>  
        </receiver>  
    </application>  
</manifest>
```

Sau đó thực hiện chạy ứng dụng và nhập nội dung cần gửi broadcast để xem kết quả trong Toast hiển thị bên dưới.

## TÓM TẮT

*Broadcast Receiver là thành phần rất quan trọng trong ứng dụng Android, cho phép bắt và xử lý các sự kiện được phát sinh từ hệ thống Android hay sự kiện phát sinh từ một ứng dụng nào đó.*

*Thông qua bài học này người học có thể hiểu được cơ chế phát sinh các sự kiện trong hệ thống Android, hiểu được cách chặn và xử lý một sự kiện nào đó.*

## BÀI TẬP

**Câu 1:** Viết ứng dụng Android cho phép lưu các tin nhắn SMS mà máy nhận được. Các tin nhắn gửi đến máy sẽ được lưu trong CSDL SQLite riêng của ứng dụng. Ứng dụng cho phép người dùng có thể thêm, xóa, và gửi nội dung các tin nhắn vào một địa chỉ email xác định.

**Câu 2:** Viết ứng dụng Android lưu các số điện thoại gọi đến máy, các thông tin lưu trữ gồm: {số điện thoại, thời gian gọi} được lưu trong CSDL SQLite. Ứng dụng cho phép quản lý các cuộc gọi lưu trong CSDL riêng này.

**Câu 3:** Viết hai ứng dụng Android thực hiện yêu cầu sau:

Ứng dụng Android A: giả lập máy rút tiền ATM, cho phép user xem thông tin tài khoản, xem số dư, xem lịch sử giao dịch, cho phép rút tiền. Trong đó chức năng rút tiền sau khi thực hiện xong thì phát sinh một Broadcast lên hệ thống.

Ứng dụng Android B: đăng ký xử lý sự kiện khi A có giao dịch rút tiền hoàn tất, khi đó B sẽ lấy thông tin {mã tài khoản, tên tài khoản, số tiền rút, và số dư cuối}. B sẽ kiểm tra trong CSDL khách hàng để lấy số điện thoại của khách hàng. Cuối cùng B thực hiện nhắn tin cho khách hàng biết giao dịch vừa diễn ra với số tiền rút và số dư cuối.

**Câu 4:** Bổ sung thêm dựa trên ứng dụng trong câu 3 theo yêu cầu sau:

Bổ sung thêm chức năng chuyển khoản cho ứng dụng A, khi đó sẽ cho người dùng chuyển vào tài khoản khác số tiền nhỏ hơn số dư hiện tại. Khi đó A sẽ phát sinh 2 sự kiện lên hệ thống:

1. Sự kiện rút tiền của tài khoản đang làm việc (sự kiện này đã xử lý trong câu 3).
2. Sự kiện chuyển khoản

Sự kiện chuyển khoản sẽ được xử lý trong ứng dụng B (phần này viết bổ sung). B thực hiện chức năng nhắn tin cho khách hàng được chuyển khoản.

# BÀI 10: THÀNH PHẦN NOTIFICATION

Học xong bài này, người học cần nắm được các nội dung sau.

- Cơ chế Notification trong ứng dụng Android, đây là thành phần rất cần thiết trong các hệ điều hành dành cho thiết bị di động.
- Các dạng thông điệp được hiển thị trong nền tảng Android.
- Hiểu được cách thức tạo và phát sinh thông điệp trong lập trình Android.
- Làm việc được với các chức năng nâng cao như thể hiện dạng progress trong ứng dụng Android.

## 10.1 THÀNH PHẦN NOTIFICATION

Notification là một dạng thông điệp đặc biệt để hiển thị thông tin cho người dùng, thông điệp này xuất hiện bên ngoài giao diện thông thường của ứng dụng. Khi ứng dụng báo với hệ thống phát sinh một Notification, thì trên màn hình xuất hiện một biểu tượng trong vùng Notification. Để nhìn thấy chi tiết của Notification, người dùng mở Notification Drawer. Cả hai thành phần này đều là vùng quản lý của hệ thống và người dùng có thể mở bất cứ lúc nào.



**Hình 10.1: Các thông điệp trong Notification Area**



**Hình 10.2: Các Notification trong Notification Drawer**

Trong nền tảng Android thành phần Notification khá hữu dụng và quen thuộc với người dùng, cho phép người dùng có thể biết được những gì mà các ứng dụng hay hệ thống muốn thông báo. Ví dụ như: có sms được nhận, có cuộc gọi nhỡ, mạng Wi-Fi, Bluetooth...

## **10.2 THỂ HIỆN THÔNG ĐIỆP**

---

Các thông điệp trong vùng Notification Area có thể xuất hiện theo hai dạng cơ bản như sau, điều này còn tùy thuộc vào phiên bản và trạng thái của Drawer.

- Dạng bình thường (normal view): đây là giao diện chuẩn của các thông điệp khi hiển thị trong Notification Drawer.
- Dạng lớn (big view): dạng này xuất hiện khi thông điệp được mở ra. Cách thể hiện này là đặc tính mới xuất hiện trong bản Android 4.1.

### **10.2.1 Dạng bình thường**

Một thông điệp trong chế độ này xuất hiện trong vùng có kích thước khoảng 64 dp chiều cao. Thậm chí nếu tạo một thông điệp với dạng lớn, thì vẫn xuất hiện ở dạng bình thường cho đến khi được mở ra. Hình 10.3 bên dưới minh họa cho dạng thể hiện này.



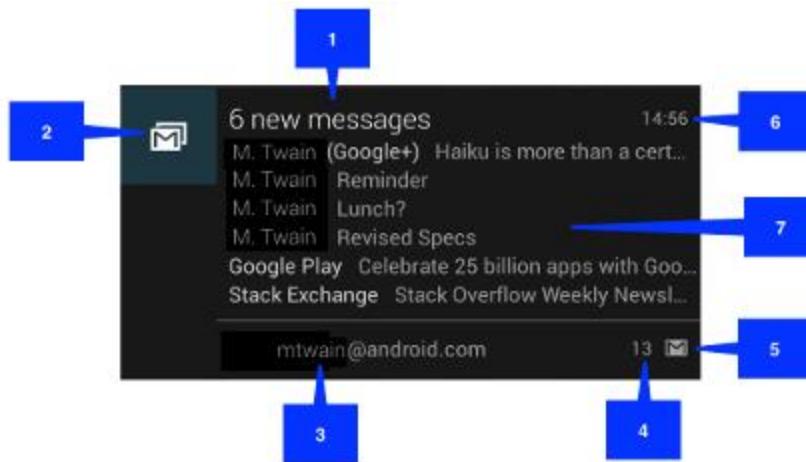
**Hình 10.3: Các thông điệp ở dạng bình thường.**

Các thành phần theo nhãn trên hình 10-3 như sau:

1. Nội dung tiêu đề.
2. Icon lớn.
3. Nội dung thông điệp.
4. Thông tin nội dung.
5. Icon nhỏ.
6. Thời gian mà thông điệp đưa ra.

## 10.2.2 Dạng lớn

Dạng thông điệp lớn này chỉ hiển thị khi thông điệp được mở rộng, điều này xuất hiện khi thông điệp là trên cùng của Notification Drawer, hoặc khi người dùng mở rộng. Chức năng này chỉ xuất hiện từ phiên bản 4.1. Hình 10-4 minh họa phần thể hiện dạng này.



**Hình 10.4: Dạng thể hiện lớn.**

Như ta thấy trên hình 10.4, hầu hết chúng đều giống các thành phần trong dạng bình thường, chỉ khác là thành phần có nhãn số 7, đây là vùng chi tiết. Có một số kiểu thể hiện khác nhau của dạng view này:

1. Dạng ảnh lớn (big picture style): vùng chi tiết chứa một ảnh bitmap có kích thước đến 256 chiều cao.
2. Dạng văn bản lớn (big text style): hiển thị khối văn bản lớn trong vùng chi tiết.
3. Dạng inbox (inbox style): hiển thị các dòng văn bản trong vùng chi tiết.

Ngoài ra trong dạng view này cũng cho phép các nội dung tùy chọn sau, các nội dung này không có trong view bình thường.

1. Dạng tiêu đề nội dung lớn: cho phép ta viết lại tiêu đề nội dung đã xuất hiện trong view bình thường.
2. Văn bản tóm tắt: cho phép thêm một dòng văn bản bên dưới vùng chi tiết.

## **10.3 TẠO THÔNG ĐIỆP**

---

Người lập trình thiết lập thông tin giao diện và các hành động cho một thông điệp thông qua đối tượng NotificationCompat.Builder. Để tạo ra thông điệp thì phải gọi phương thức NotificationCompat.Builder.build(), phương thức này trả về một đối tượng Notification chứa phần mô tả của chương trình. Sau khi thiết lập xong các thông số cho thông điệp thì phát sinh thông điệp thì truyền đối tượng Notification này cho hệ thống bằng cách gọi phương thức NotificationManager.notify().

### **10.3.1 Các nội dung thông điệp bắt buộc**

Một đối tượng notification phải chứa các thông tin sau:

1. Icon nhỏ: Dùng phương thức setSmallIcon()
2. Tiêu đề: Thiết lập với setContentTitle()
3. Nội dung chi tiết: setContentText()

Các thông tin còn lại là tùy chọn, tùy theo yêu cầu cụ thể mà thông điệp có thiết lập hay không thiết lập.

### 10.3.2 Hành động của thông điệp

Mặc dù đây là thành phần tùy chọn, nhưng ta nên thêm tối thiểu một hành động cho thông điệp. Một hành động cho phép người dùng đi từ thông điệp đến một Activity trong ứng dụng.

Một thông điệp có thể cung cấp nhiều hành động, thông thường là định nghĩa hành động ràng buộc khi người dùng chạm vào thông điệp, phổ biến là mở một Activity của ứng dụng. Ngoài ra ta cũng có thể thêm các button vào trong thông điệp để thực thi các hành động thêm vào như tắt alarm hay trả lời tin nhắn (từ phiên bản 4.1).

### 10.3.3 Tạo thông điệp đơn giản

Đoạn code sau minh họa cách tạo một thông điệp đơn giản cho phép user mở Activity khi chạm hay kích vào thông điệp.

```
NotificationCompat.Builder mBuilder =  
    new NotificationCompat.Builder(this)  
        .setSmallIcon(R.drawable.notification_icon)  
        .setContentTitle("My notification")  
        .setContentText("Hello World!");  
  
Intent resultIntent = new Intent(this, ResultActivity.class);  
  
TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);  
stackBuilder.addParentStack(ResultActivity.class);  
stackBuilder.addNextIntent(resultIntent);  
  
PendingIntent resultPendingIntent =  
    stackBuilder.getPendingIntent( 0,  
        PendingIntent.FLAG_UPDATE_CURRENT );  
  
mBuilder.setContentIntent(resultPendingIntent);  
  
NotificationManager mNotificationManager =  
    (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
```

```
mNotificationManager.notify(mId, mBuilder.build());
```

## 10.4 SỬ DỤNG THANH TIẾN ĐỘ

---

Thông điệp có thể chứa một thanh chỉ dẫn tiến độ dạng animation cho người dùng biết được tình trạng hoạt động của chức năng nào đó. Nếu có thể ước lượng được thời gian mà hoạt động này diễn ra và thời gian để hoàn thành thì dùng hình thức xác định là thanh tiến độ (progress bar). Nếu không xác định được thời gian hoàn thành của hoạt động thì dùng hình thức không xác định (dạng minh họa hoạt động liên tục).

Từ phiên bản Android 4.0 để sử dụng thanh tiến độ thì gọi phương thức `setProgress()`. Trong phiên bản trước thì phải tạo layout của thông điệp chứa `ProgressBar`.

### 10.4.1 Thanh tiến độ với thời gian xác định

Để hiển thị thanh tiến độ với thời gian xác định, gọi phương thức `setProgress()` theo tham số `setProgress(max, progress, false)` và phát sinh thông điệp. Khi hoạt động diễn ra giá trị `progress` gia tăng và cập nhật thông điệp. Khi hoạt động hoàn thành thì giá trị `progress` bằng với `max`. Cách thực hiện chung là gọi `setProgress()` với `max` là 100 và `progress` như là mức độ hoàn thành công việc.

Sau khi công việc hoàn thành thì có thể cập nhật lại nội dung của thông điệp để báo hoàn thành và xóa thanh tiến độ bằng cách gọi `setProgress(0, 0, false)`

Đoạn code sau minh họa việc tạo thanh tiến độ:

```
mNotifyManager =  
    (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);  
  
mBuilder = new NotificationCompat.Builder(this);  
  
mBuilder.setContentTitle("Picture Download")  
    .setContentText("Download in progress")  
    .setSmallIcon(R.drawable.ic_notification);  
  
new Thread(  
    () -> {  
        int max = 100;  
        int progress = 0;  
        while (progress < max) {  
            progress++;  
            mNotifyManager.setProgress(max, progress, false);  
            try {  
                Thread.sleep(100);  
            } catch (InterruptedException e) {}  
        }  
        mNotifyManager.setProgress(0, 0, false);  
    }).start();
```

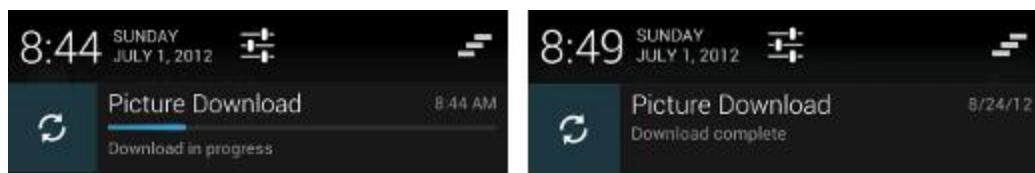
```

new Runnable() {

    @Override
    public void run() {
        int incr;
        for (incr = 0; incr <= 100; incr+=5) {
            mBuilder.setProgress(100, incr, false);
            mNotifyManager.notify(0, mBuilder.build());
            try {
                Thread.sleep(5*1000);
            } catch (InterruptedException e) {
                Log.d(TAG, "sleep failure");
            }
        }
        mBuilder.setContentText("Download complete")
            .setProgress(0,0,false);
        mNotifyManager.notify(ID, mBuilder.build());
    }
}
).start();

```

Kết quả của chương trình đầy đủ với đoạn code trên như hình bên dưới, trong đó bên trái là chụp khi chưa hoàn thành và bên phải thể hiện khi công việc hoàn thành.



**Hình 10.5: Thanh tiến độ**

## 10.4.2 Thanh chỉ dẫn liên tục

Để hiển thị một hoạt động liên tục và không xác định chính xác thời gian thì gọi phương thức `setProgress(0, 0, true)`, hai tham số đầu thiết lập là 0. Kết quả là thanh

chỉ dẫn cùng kiểu với thanh tiến độ bên trên nhưng khác là cách thể hiện animation liên tục.

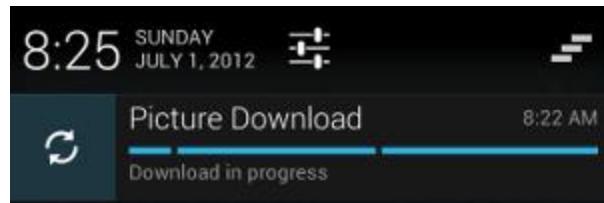
Trong đoạn code trên ta tìm đoạn code:

```
mBuilder.setProgress(100, incr, false);  
mNotifyManager.notify(0, mBuilder.build());
```

Sau đó thay đoạn code trên bằng

```
mBuilder.setProgress(0, 0, true);  
mNotifyManager.notify(0, mBuilder.build());
```

Kết quả thể hiện như hình bên dưới



**Hình 10.6: Minh họa thanh hoạt động liên tục**

## TÓM TẮT

Các dạng Notification là thành phần quan trọng và được người dùng đánh giá là cần thiết và tiện dụng. Cho phép người sử dụng thiết bị nhanh chóng có được các thông tin chung về tình trạng của các ứng dụng đang hoạt động trên nền tảng Android. Ngoài ra với các ứng dụng chạy nền, không có giao diện tương tác với người dùng thì tính năng Notification là rất cần thiết. Người phát triển ứng dụng Android có thể bổ sung cho ứng dụng tính năng hỗ trợ Notification sẽ nâng cao tính tương tác và tính tiện dụng.

Trong bài học này cung cấp các kiến thức để xây dựng thành phần Notification để bổ sung cho ứng dụng Android. Các dạng thể hiện của Notification được trình bày chi tiết, tùy theo yêu cầu cụ thể của ứng dụng mà người phát triển ứng dụng có thể chọn cách thể hiện nào đó cho thích hợp nhất.

## BÀI TẬP

**Câu 1:** Viết ứng dụng Android chặn các sự kiện của hệ thống Android và hiển thị lên Notification. Ví dụ khi pin yếu, cắm nguồn, điện thoại gọi tới...thì báo.

**Câu 2:** Viết ứng dụng Android cho nhận các tin nhắn SMS, khi nhận tin nhắn thì dùng Notification hiển thị cho người dùng xem.

**Câu 3:** Viết ứng dụng Android cơ bản cho phép check email, khi có mail thì hiển thị tóm tắt trên Notification Bar.

**Câu 4:** Viết ứng dụng minh họa xử lý thao tác nào đó (chỉ minh họa nên dùng sleep) trong 100s. Mỗi giây sẽ cập nhật tiến độ hoàn thành của công việc lên thanh Notification của ứng dụng.

**Câu 5:** Viết ứng dụng Alarm cảnh báo lên thanh Notification.

**Câu 6:** Xem lại tất cả ứng dụng của các bài học trước, nếu bài tập nào có hiển thị thông tin dạng Toast thì sửa lại hiển thị lên thanh Notification.

## TÀI LIỆU THAM KHẢO

1. Grant Allen (2012), *Beginning Android 4*, Apress, New York.
2. Marziah Karch (2010), *Android for Work: Productivity for Professionals*, Apress, New York.
3. Wei-Meng-Lee (2013), *Android Application Development Cookbook*, John Wiley & Sons, Inc.
4. Jonathan Simon (2011), *Head First Android Development*, O'reilly Media, Sebastopol.
5. Lucas Jordan, Pieter Greyling (2011). *Practical Android Projects*, Apress, New York.
6. Mark Rollins (2011). *The Business of Android Apps Development*, Apress, New York.
7. Satya Komatineni, Dave MacLean (2012). *Pro Android 4*, Apress, New York.
8. Trần Duy Thành (2013), *Hướng dẫn thực hành – Lập trình cho thiết bị di động*, Đại học Công nghiệp TP.HCM.
9. Lê Đức Huy (2012), *Phái triển ứng dụng smartphone*, Đại học Hoa Sen TP. HCM.
10. Nguyễn Hà Giang (2017), *Lập trình trên thiết bị di động*, Khoa CNTT – HUTECH
11. Website Tutlane.com, *Android tutorial*