

ỦY BAN NHÂN DÂN TP. HỒ CHÍ MINH
TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THỦ ĐỨC
KHOA CÔNG NGHỆ THÔNG TIN

GIÁO TRÌNH
HỌC PHẦN:LẬP TRÌNH DI ĐỘNG 1.
NGÀNH/NGHỀ:CÔNG NGHỆ THÔNG TIN
TRÌNH ĐỘ: CAO ĐẲNG

Ban hành kèm theo Quyết định số:... .. /QĐ-CNTĐ-CN ngày.... tháng.... năm...
của.....

TP. Hồ Chí Minh, năm 2019

TUYÊN BỐ BẢN QUYỀN

Tài liệu này thuộc loại sách giáo trình nên các nguồn thông tin có thể được phép dùng nguyên bản hoặc trích dùng cho các mục đích về đào tạo và tham khảo. Mọi mục đích khác mang tính lèch lạc hoặc sử dụng với mục đích kinh doanh thiếu lành mạnh là vi phạm quyền tác giả.

LỜI GIỚI THIỆU

Học phần Lập trình di động 1 là học phần bắt buộc của ngành Công nghệ thông tin. Đây là học phần chuyên ngành Công nghệ thông tin, cung cấp cho sinh viên có khả năng phát triển các ứng dụng vừa và nhỏ trên thiết bị di động. Sinh viên sẽ được trình bày các khái niệm cơ bản cũng như quy trình trong phát triển ứng dụng trên điện thoại di động; Thông qua các hoạt động học tập, sinh viên có thể hoàn thiện dần tính chủ động, tích cực, khả năng tự học, tư duy hệ thống và thói quen tuân thủ các quy định làm việc trong môi trường chuyên nghiệp.

Giáo trình này được biên soạn dựa theo đề cương học phần “Lập trình di động 1” của Khoa Công nghệ thông tin – Trường Cao đẳng Công nghệ Thủ Đức. Dù đã rất cố gắng, song sẽ không tránh khỏi những thiếu sót, rất mong nhận được sự góp ý chân thành từ các quý đọc giả để giáo trình được hoàn thiện hơn.

....., ngày...tháng.... năm.....

Tham gia biên soạn
1. Trương Bá Thái
2: Tiêu Kim Cương

GIÁO TRÌNH HỌC PHẦN

Tên học phần: Lập trình di động 1

Mã học phần: CNC107311

Vị trí, tính chất, ý nghĩa và vai trò của học phần:

- Vị trí: Đây là học phần chuyên ngành bắt buộc, hệ cao đẳng ngành công nghệ thông tin.
- Tính chất: Đây là học phần chuyên ngành Công nghệ thông tin, cung cấp cho sinh viên có khả năng phát triển các ứng dụng vừa và nhỏ trên thiết bị di động. Sinh viên sẽ được trình bày các khái niệm cơ bản cũng như quy trình trong phát triển ứng dụng trên điện thoại di động.
- Ý nghĩa và vai trò của học phần: Thông qua các hoạt động học tập, sinh viên có thể hoàn thiện dần tính chủ động, tích cực, khả năng tự học, tư duy hệ thống và thói quen tuân thủ các quy định làm việc trong môi trường chuyên nghiệp.

Mục tiêu của học phần:

 **Về kiến thức:**

- Tổng hợp được kiến thức cơ bản về ngôn ngữ lập trình di động
- Trình bày cách xây dựng ứng dụng di động trên IDE Android Studio.

 **Về kỹ năng:**

- Xây dựng được giao diện cho ứng dụng Android
- Xây dựng hệ thống điều hướng cho ứng dụng Android.
- Xây dựng hệ thống cơ sở dữ liệu cho ứng dụng Android.
- Sử dụng thành thạo IDE Android Studio để viết chương trình Android
- Sử dụng được IDE Android Studio để debug chương trình Android.
- Hình thành thói quen thiết kế chương trình theo tiếp cận Top-Down.

 **Về năng lực tự chủ và trách nhiệm:**

- Hình thành thói quen thiết kế chương trình theo tiếp cận Top-Down.
- Hình thành thói quen viết code theo chuẩn, không copy bài, chủ động hỏi bài và tuân thủ quy định về lịch làm việc của giảng viên trong suốt quá trình học.

MỤC LỤC

CHƯƠNG 1. TỔNG QUAN VỀ LẬP TRÌNH ANDROID.....	7
1.1 / <i>Tổng quan về Android.....</i>	7
1.1.1 Hệ điều hành Android	7
1.1.2 Tại sao lập trình trên Android	14
1.2 / <i>Giới thiệu nền tảng phát triển Android.....</i>	15
1.2.1 Kiến trúc	15
1.2.2 Ngôn ngữ lập trình.....	18
1.3 / <i>Môi trường phát triển ứng dụng</i>	20
1.3.1 Giới thiệu Java JDK, Android SDK, Android Studio	20
1.3.2 Thiết lập môi trường phát triển Android Studio.....	21
1.4 / <i>Tạo ứng dụng đầu tiên</i>	21
1.4.1 Khởi tạo dự án	21
1.4.2 Cấu trúc dự án.....	33
1.4.3 AndroidManifest.....	37
1.5 / <i>Quản lý trạng thái Activity.....</i>	39
1.5.1 Xây dựng Activity	39
1.5.2 Vòng đời của Activity	39
1.6 / <i>Debug chương trình trong Android Studio.....</i>	41
1.6.1 Sử dụng Log và Toast.....	41
1.6.2 Sử dụng Debug	43
1.7 / <i>Bài tập Chương 1</i>	45
Bài tập 1.1. Phân biệt reground Lifetime và Visible Lifetime	46
Bài tập 1.2. Sử dụng ConstraintLayout xây dựng ứng dụng	47
CHƯƠNG 2. Giao diện người dùng và xử lý sự kiện.....	47
2.1 / <i>Giao diện người dùng</i>	48
2.1.1 FrameLayout.....	50
2.1.2 RelativeLayout	53
2.1.3 LinearLayout	57
2.1.4 GridLayout.....	60
2.2 / <i>Các điều khiển cơ bản.....</i>	61
2.2.1 TextView	64
2.2.2 EditText	69
2.2.3 Button	77
2.2.4 Checkbox	87
2.2.5 RadioButton.....	96
2.2.6 Image	109
2.2.7 ToggleButton	115
2.2.8 Switch	119
2.2.9 ScrollView	126
2.3 / <i>Các điều khiển hiển thị danh sách</i>	127
2.3.1 Các dạng Adapter	127
2.3.2 Spinner.....	129
2.4 / <i>Các điều khiển danh sách listview.....</i>	134

2.4.1 ListView	134
2.4.2 Custom ListView	141
2.5 / Các điều khiển nâng cao.....	149
2.5.1 TimePickerDialog.....	149
2.5.2 DatePicker Dialog	153
2.5.3 TabHost	157
2.6 / Bài tập Chương 2	167
Bài tập 2.1. Sử dụng Linear Layout thiết kế giao diện như sau:	167
Bài tập 2.2. Xây dựng màn hình đăng nhập	169
Bài tập 2.3. Lật bài và tính điểm.....	170
Bài tập 2.4. Thay đổi font chữ.....	171
Bài tập 2.5. Thay đổi hình.....	173
Bài tập 2.6. Thu thập thông tin cá nhân	174
Bài tập 2.7. Thông tin cá nhân.....	176
Bài tập 2.8. Xây dựng trắc nghiệm	178
Bài tập 2.9. Xây dựng listview country	185
Bài tập 2.10. Xây dựng chương trình đặt đồ ăn	186
Bài tập 2.11. Xây dựng chương trình danh sách cầu thủ	189
Bài tập 2.12. Xây dựng chương trình hiển thị danh sách cầu thủ và cập nhật danh sách.....	190
Bài tập 2.13. Hiển thị danh sách film và cập danh sách film	192
Bài tập 2.14. Xây dựng chương trình quản lý danh bạ.....	193
CHƯƠNG 3. Xây dựng giao diện với Fragment	194
3.1 / Các khái niệm cơ bản.....	194
3.1.1 Fragment và phiên bản hỗ trợ.....	194
3.1.2 Giao diện Fragment	196
3.1.3 Vòng đời của một Fragment.....	196
3.2 / Xây dựng và sử dụng Fragment	198
3.2.1 Thực hiện xây dựng Fragment.....	198
3.2.2 Sử dụng Fragment	199
3.3 / Bài tập Chương 3	212
Bài tập 3.1. Tạo và sử dụng Fragment 1	212
Bài tập 3.2. Tạo và sử dụng Fragment 2	214
CHƯƠNG 4. Action Bar - Navigation Drawers	216
4.1 / Action bar.....	216
4.1.1 Giới thiệu	216
4.1.2 Tạo Action Bar	217
4.1.3 Thao tác với Action Bar	218
4.2 / Navigation Drawers.....	221
4.2.1 Giới thiệu	221
4.2.2 Tạo Navigation Drawers.....	222
4.2.3 Thao tác với Navigation Drawers.....	224
4.3 / Bài tập Chương 4	227
Bài tập 4.1. Tạo và sử dụng OptionMenu	228
Bài tập 4.2. Xây dựng chương trình sử dụng ActionBar	228

<i>Bài tập 4.3. Xây dựng chương trình sử dụng ActionBar SearchView</i>	229
CHƯƠNG 5. Lập trình cơ sở dữ liệu với SQLite	230
<i>5.1 / Giới thiệu về SQLite.....</i>	230
<i>5.2 / Sử dụng SQLite trong Android</i>	230
5.2.1 Tạo và xóa cơ sở dữ liệu.....	230
5.2.2 Mở Cơ sở dữ liệu.....	232
5.2.3 Đóng Cơ sở dữ liệu.....	233
5.2.4 Truy vấn dữ liệu trong các bảng.....	233
5.2.5 Sắp xếp dữ liệu	237
<i>5.3 / Bài tập Chương 5</i>	256
<i>Bài tập 5.1. Xây dựng ứng dụng quản lý sinh viên, dữ liệu SinhVien gồm: Mã sinh viên, Tên sinh viên, Lớp.</i>	256
<i>Bài tập 5.2. Xây dựng ứng dụng quản lý nhân viên.....</i>	258

Danh mục hình

Hình 1-1: Các phiên bản của hệ điều hành Android.....	7
Hình 1-2: Android 1.5 Cupcake.....	8
Hình 1-3: Android 1.6 Donut.....	8
Hình 1-4: Android 2.0 Eclair	9
Hình 1-5: Android 2.2 Froyo	10
Hình 1-6: Android 2.3 Gingerbread.....	10
Hình 1-7: Android 3.0 Honeycomb	11
Hình 1-8: Android 4.0 Ice Cream Sandwich	11
Hình 1-9: Android 4.1 Jelly Bean.....	12
Hình 1-10: Android 4.4 KitKat.....	12
Hình 1-11: Android 5.0 Lollipop.....	13
Hình 1-12: Android 6.0 Marshmallow	13
Hình 1-13: Android 7.0 Nougat.....	14
Hình 1-14: Android 8.0 Oreo.....	14
Hình 1-15: Minh họa sơ đồ cấu trúc Android.....	16
Hình 1-16: Download Java JDK.....	Error! Bookmark not defined.
Hình 1-17: Download Android Studio	Error! Bookmark not defined.
Hình 1-18: Chọn Next để cài đặt Android Studio	Error! Bookmark not defined.
Hình 1-19: Chọn các thành phần trong Android Studio	Error! Bookmark not defined.
Hình 1-20: Chọn đồng ý để cài đặt.....	Error! Bookmark not defined.
Hình 1-21: Chọn nơi lưu trữ Android Studio	Error! Bookmark not defined.
Hình 1-22: Cấu hình bộ nhớ ảo	Error! Bookmark not defined.
Hình 1-23: Đặt tên cho Android Studio.....	Error! Bookmark not defined.
Hình 1-24: Kết thúc cài đặt.....	Error! Bookmark not defined.
Hình 1-25: Bắt đầu khởi tạo Android Studio.....	Error! Bookmark not defined.
Hình 1-26: Complete Install	Error! Bookmark not defined.
Hình 1-27: chọn Theme cho Android Studio	Error! Bookmark not defined.
Hình 1-28: Download Component	Error! Bookmark not defined.
Hình 1-29: Finish download component	Error! Bookmark not defined.
Hình 1-30: Tạo mới Android Studio project	21
Hình 1-31: Đặt tên project	22
Hình 1-32: Nền tảng ứng dụng	23
Hình 1-33: Tạo mới ứng dụng	24
Hình 1-34: đặt tên cho Activity Name và Layout Name	25
Hình 1-35:Giao diện các vùng ứng dụng ban đầu	26
Hình 1-36: Cách 1 tạo máy áo	28
Hình 1-37: Cách 2 tạo máy áo	28
Hình 1-38: Tạo máy áo	28
Hình 1-39:Chọn máy áo.....	29
Hình 1-40: Config Hardware Profile	30
Hình 1-41: Verify Config	31
Hình 1-42: Chạy máy áo.....	31
Hình 1-43: Máy áo	32
Hình 1-44: Thực thi ứng dụng	32
Hình 1-45: Màn hình ứng dụng	33

Hình 1-46: Cấu trúc ứng dụng	34
Hình 1-47: Vòng đời Activity.....	40
Hình 1-48: Đọc log	42
Hình 1-49: Các hình dạng Toast.....	43
Hình 1-50: Đặt các breakpoint.....	44
Hình 1-51: Chọn Debug.....	44
Hình 1-52: Quy trình debug.....	45
Hình 2-1: Xây dựng giao diện bằng FrameLayout.....	52
Hình 2-2: xây dựng giao diện sử dụng RelativeLayout.....	54
Hình 2-3: xây dựng giao diện sử dụng RelativeLayout màn hình đăng nhập	56
Hình 2-4: Ví dụ TextView trước khi click	66
Hình 2-5: Ví dụ TextView Sau khi click	67
Hình 2-6: Các dạng EditText.....	70
Hình 2-7: Ví dụ EditText trước đăng ký.....	72
Hình 2-8: Ví dụ EditText sau đăng ký	72
Hình 2-9: Các dạng Button	78
Hình 2-10: Hiển thị Icon bên dưới chuỗi văn bản	81
Hình 2-11: Ví dụ Button trước khi tính	82
Hình 2-12: Ví dụ Button sau khi tính	82
Hình 2-13: Các dạng CheckBok	88
Hình 2-14: Ví dụ CheckBox trước khi chọn.....	90
Hình 2-15: Ví dụ CheckBox sau khi chọn.....	91
Hình 2-16: Các dạng RadioButton	97
Hình 2-17: Ví dụ RadioButton trước khi gửi thông tin	99
Hình 2-18: Ví dụ RadioButton sau khi gửi thông tin	99
Hình 2-19: Ví dụ Image	110
Hình 2-20: ví dụ minh họa ImageView	111
Hình 2-21: Ví dụ ImageView xem hình 1	112
Hình 2-22: Các trạng thái của ToggleButton.....	115
Hình 2-23: Ví dụ ToggleButton trạng thái tắt	117
Hình 2-24: Ví dụ ToggleButton trạng thái mở	117
Hình 2-25: Minh Họa Switch	122
Hình 2-26: Ví dụ Switch trước khi nhấn submit	122
Hình 2-27: Ví dụ Switch sau khi nhấn submit.....	123
Hình 2-28: Các dạng Adapter	127
Hình 2-29: Các dạng Spinner	130
Hình 2-30: ví dụ Spiner trước khi chọn	131
Hình 2-31: Ví dụ Spinner sau khi chọn	131
Hình 2-32: Minh họa ListView.....	135
Hình 2-33:Ví dụ ListView hiển thị danh sách sản phẩm.....	137
Hình 2-34: Ví dụ ListView khi chọn sản phẩm.....	137
Hình 2-35: Ví dụ hiển thị ListView và Custom ListView.....	141
Hình 2-36: Ví dụ Custom Listview hiển thị danh sách	142
Hình 2-37: Ví dụ Custom ListView khi chọn danh sách.....	143
Hình 2-38: Ví dụ TimePicker	151
Hình 2-39: Ví dụ DatePicker	154

Hình 2-40: Minh họa Tabhost.....	158
Hình 2-41: Ví dụ Tabhost	161
Hình 3-1: Các dạng Fragment màn hình đứng	195
Hình 3-2: Các dạng Fragment màn hình ngang.....	196
Hình 3-3: Vòng đời Fragment	197
Hình 3-4: Ví dụ xây dựng fragment tĩnh	201
Hình 3-5: Ví dụ xây dựng Fragment động.....	207
Hình 4-1: Các hình thức của Action Bar	217
Hình 4-2: Ví dụ sử dụng Action bar	219
Hình 4-3: Minh họa Navigation Drawers	222
Hình 4-4: Navigation drawer chọn template	223
Hình 4-5: Giao diện Navigation Drawer	224

CHƯƠNG 1. TỔNG QUAN VỀ LẬP TRÌNH ANDROID

MỤC TIÊU THỰC HIỆN

- ✓ Trình bày tổng quan về lập trình android
- ✓ Cài đặt được môi trường phát triển
- ✓ Tạo được ứng dụng đầu tiên
- ✓ Biết cách debug trong android studio

1.1 | Tổng quan về Android

1.1.1 | Hệ điều hành Android

Android là một Hệ điều hành mã nguồn mở và là một hệ điều hành dựa trên Linux cho các thiết bị mobile như Smartphone và máy tính bảng. Ban đầu Android được phát triển bởi Công ty Android với sự hỗ trợ tài chính từ Google, sau đó được Google mua lại vào năm 2005.

Android đưa ra một phương pháp thống nhất để phát triển ứng dụng cho các thiết bị di động, nghĩa là các lập trình viên chỉ cần phát triển Android, và các ứng dụng khác có thể chạy trên các thiết bị khác nhau mà đã được trang bị Android.

Các phiên bản của hệ điều hành Android:



Hình 1-1: Các phiên bản của hệ điều hành Android

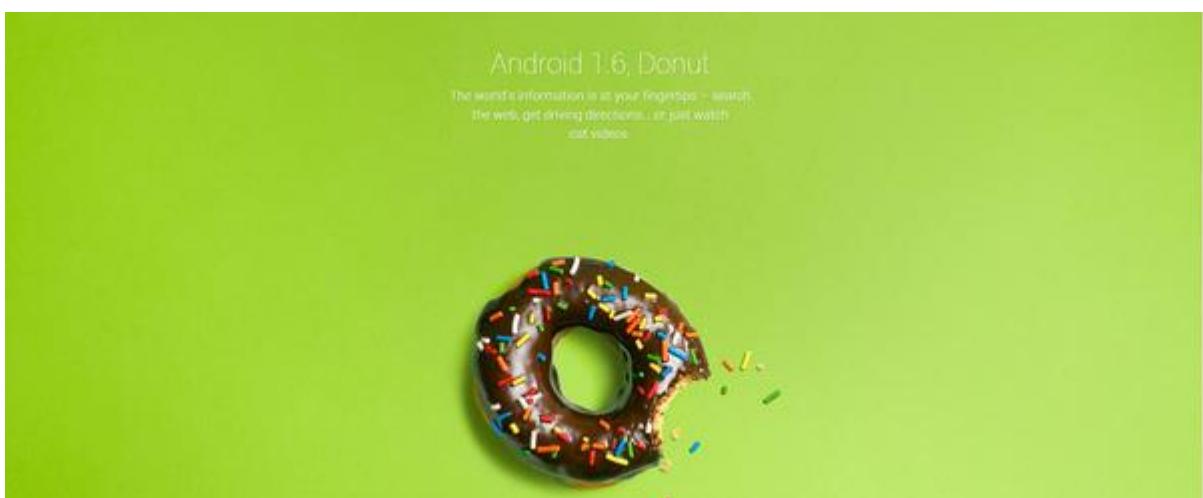
1.1.1.1 | Android 1.5 Cupcake



Hình 1-2: Android 1.5 Cupcake

Điểm nhấn đáng chú ý nhất ra đời cùng Android 1.5 Cupcake được cho là bàn phím ảo. Có thể bạn sẽ bất ngờ về điều này, tuy nhiên trở lại thời điểm những năm 2008/2009, smartphone chủ yếu được ra đời với bàn phím QWERTY vật lý. Sự dịch chuyển từ yêu thích bàn phím cứng sang bàn phím ảo chỉ bắt đầu nhen nhóm khi iPhone đời đầu ra mắt vào năm 2007.

1.1.1.2 | *Android 1.6 Donut*



Hình 1-3: Android 1.6 Donut

Android 1.6 Donut vào thời điểm ra mắt không được đánh giá cao bởi người dùng vì những thay đổi nền tảng này mang lại không thực sự rõ rệt. Tuy nhiên, Android 1.6 đánh dấu việc hệ điều hành này đã hỗ trợ nhiều độ phân giải màn hình khác nhau. Đặc điểm này thực sự quan trọng trong quá trình phát triển của Android.

1.1.1.3 | *Android 2.0 Eclair*



Hình 1-4: Android 2.0 Eclair

Trái ngược với Android 1.6 Donut, Android 2.0 có nhiều tính năng mới đến mức chỉ được chọn ra một tính năng ấn tượng nhất là rất khó khăn, dẫu vậy chuyên trang Phonearena đánh giá tính năng nổi bật nhất thuộc về phần mềm camera. Theo đó, trước khi Android 2.0 xuất hiện, Android không hỗ trợ đèn flash LED. Các tính năng như cân bằng trắng, chế độ lấy nét, chế độ chụp cảnh, hiệu ứng màu cũng hoàn toàn không khả dụng.

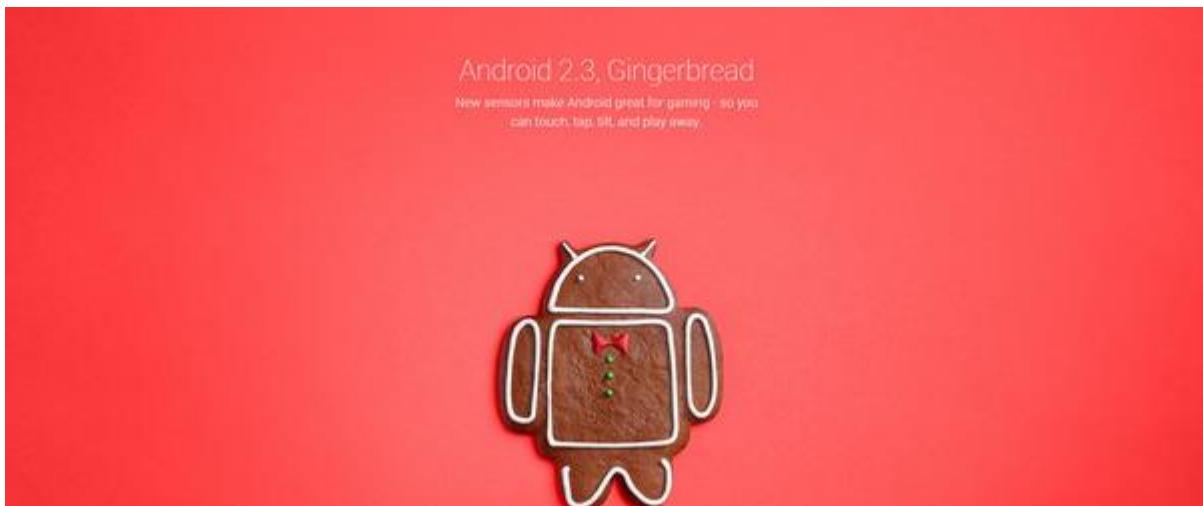
1.1.1.4 | *Android 2.2 Froyo*



Hình 1-5: Android 2.2 Froyo

Hai điểm nhấn xuất hiện trên Android 2.2 Froyo là hiệu năng xử lý tuyệt vời và hỗ trợ kết nối Wi-Fi. Ở thời điểm nền tảng này ra mắt, Google khẳng định hiệu năng thiết bị vận hành trên nền tảng Android 2.2 có thể tăng lên gấp 2,5 lần.

1.1.1.5 / Android 2.3 Gingerbread



Hình 1-6: Android 2.3 Gingerbread

Android 2.3 Gingerbread được coi là một trong những phiên bản Android phổ biến nhất trong lịch sử hệ điều hành này. Thậm chí cho đến tận hôm nay, vẫn có tới 11% các thiết bị Android chạy phiên bản 2.3 Gingerbread. Android 2.3 Gingerbread mang đến cho thiết bị khả năng hỗ trợ nhiều loại cảm biến, bao gồm những cảm biến như cảm biến áp suất khí quyển (barometer) hay con quay hồi chuyển (gyroscope). Những cảm biến này ảnh hưởng rất nhiều đến tính năng và khả năng tương tác của người dùng với thiết bị Android về sau.

1.1.1.6 | *Android 3.0 Honeycomb*



Hình 1-7: Android 3.0 Honeycomb

Những đặc điểm đáng chú ý nhất trên từng phiên bản Android 6 Honeycomb là một trong những bản cập nhật Android ít được biết đến nhất, tuy nhiên không phải vì thế mà nền tảng này không có những đóng góp chung cho sự phát triển của Android. Nền tảng này đánh dấu sự quan tâm của Google đến trải nghiệm của người dùng các thiết bị máy tính bảng.

1.1.1.7 | *Android 4.0 Ice Cream Sandwich*

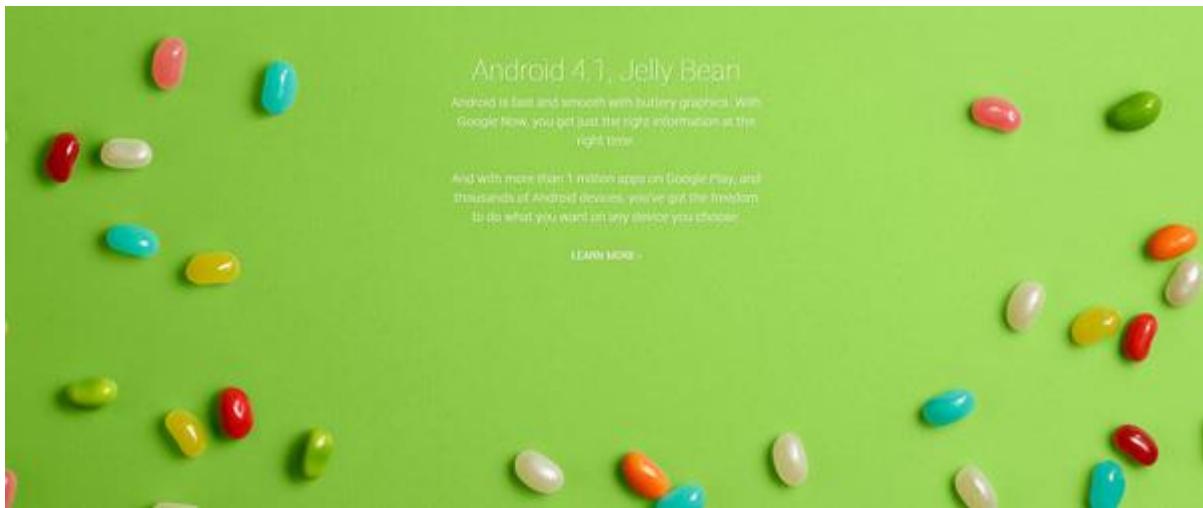


Hình 1-8: Android 4.0 Ice Cream Sandwich

Lần đầu tiên xuất hiện trong nền tảng Android 3.0 Honeycomb, giao diện "Holo" được Google cải thiện và trau chuốt thêm rất nhiều trong Android 4.0. Hiểu một cách đơn giản, giao diện người dùng Holo được Google đưa ra để giảm thiểu những khác biệt về thiết kế giữa phần mềm của các nhà sản xuất, nhà phát triển ứng dụng và

phần mềm của Google. Android 4.0 Ice Cream Sandwich để lại rất nhiều ảnh hưởng đến giao diện Android về sau.

1.1.1.8 / Android 4.1 Jelly Bean



Hình 1-9: Android 4.1 Jelly Bean

Android 4.1 Jelly Bean đánh dấu sự xuất hiện lần đầu tiên của Google Now, một thành phần quan trọng trong hệ sinh thái của Google. Tận dụng sức mạnh từ cỗ máy tìm kiếm của mình, Google Now mang đến cho người dùng những thông tin hữu ích ở giao diện dạng thẻ tùy thuộc vào ngữ cảnh. "Người trợ lý ảo" này còn có khả năng xử lý ngôn ngữ tự nhiên được đánh giá cao.

1.1.1.9 / Android 4.4 KitKat

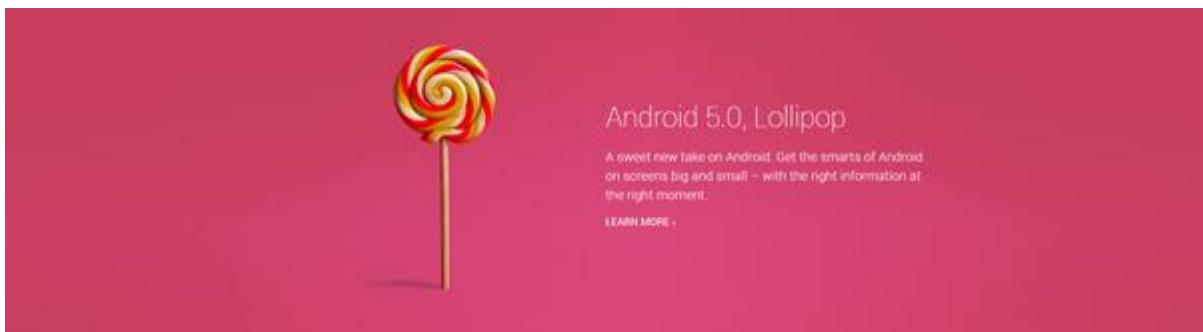


Hình 1-10: Android 4.4 KitKat

Với KitKat, Google không chỉ mang tới cho người dùng một giao diện tương tác hiện đại hơn mà còn tập trung phát triển một hệ điều hành có thể vận hành trên tru

trên cả các thiết bị giá rẻ, cấu hình khiêm tốn. Google khẳng định các thiết bị với chỉ 512MB cũng có thể cài đặt và chạy được hệ điều hành này. Đây là một bước tiến quan trọng trong việc giảm sự phân mảnh nền tảng trong thế giới các thiết bị Android.

1.1.1.10 | *Android 5.0 Lollipop*



Hình 1-11: Android 5.0 Lollipop

Lollipop rõ ràng là hệ điều hành Android có phần nhìn bắt mắt nhất từ trước đến nay với ngôn ngữ thiết kế mới được Google đặt tên là Material Design. Trong hệ điều hành này, giao diện Android được làm mới theo triết lý thiết kế phẳng cùng với hệ màu tươi sáng, khác hẳn với các hệ điều hành tiền nhiệm.

1.1.1.11 | *Android 6.0 Marshmallow*



Hình 1-12: Android 6.0 Marshmallow

Android Marshmallow đã được Google công bố vào tháng 9 năm 2015, cải thiện tuổi thọ pin và thêm các tính năng mới như Hỗ trợ tính năng Hiện hành trên Tap và hỗ trợ cảm biến vân tay.

1.1.1.12 | *Android 7.0 Nougat*



Hình 1-13: Android 7.0 Nougat

Các tính năng của Android Nougat được đưa ra trước tiên và cho cộng đồng chọn tên, được chính thức công bố vào cuối tháng 6 năm 2016. Cùng với việc cải thiện hiệu suất và quản lý pin nhờ một tính năng gọi là Doze on-the-go, Nougat cũng mang lại nhiều tiện ích như màn hình đa nhiệm. Phiên bản 7.1 đã sớm được tung ra trên các điện thoại thông minh Pixel và Pixel XL của Google. Nó hỗ trợ thêm cho trình tạo Pixel mới, Google Assistant, Night Light (chủ yếu là bộ lọc ánh sáng màu xanh), chế độ Daydream VR, các phím tắt được gọi là 'Moves' và nhiều tính năng khác.

1.1.1.13 | Android 8.0 Oreo



Hình 1-14: Android 8.0 Oreo

Vào năm 2017, Google chính thức ra phiên bản mới là Android Oreo. Phiên bản cập nhật mới nhất này cung cấp nhiều tính năng mới tuyệt vời. Trong phiên bản này Google đã có nhiều cải tiến bổ sung cho chất lượng âm thanh và nhập văn bản, cũng như quản lý tài nguyên.

1.1.2 | Tại sao lập trình trên Android

➤ Xu thế phát triển công nghệ di động

Theo nhận định của nhiều chuyên gia công nghệ từ các hãng công nghệ hàng đầu như Microsoft, Google, IBM, ... Ba xu hướng tất trên toàn cầu hiện nay là: Social and Security (mạng xã hội và bảo mật), Mobility (công nghệ di động), Analytics Big Data (phân tích dữ liệu lớn), Cloud (Điện toán đám mây).

➤ Thị trường thiết bị Android

Trong tất cả các hệ điều hành dành cho di động hiện nay, có thể nói: Android đã mang lại một cuộc cách mạng thật sự cho các lập trình viên. Nổi bật với tính mở, đơn giản nhưng mạnh mẽ, không tốn phí cho bất cứ bản quyền nào và đặc biệt cộng đồng lập trình viên vô cùng lớn mạnh. Android thật sự là một nền tảng mạnh mẽ cho phép các lập trình viên, những người chưa từng lập trình trên thiết bị di động có thể tạo ra các ứng dụng một cách nhanh chóng và dễ dàng. Có thể nói Android đang dần mang lại phong cách mới trong thói quen sử dụng điện thoại của người dùng.

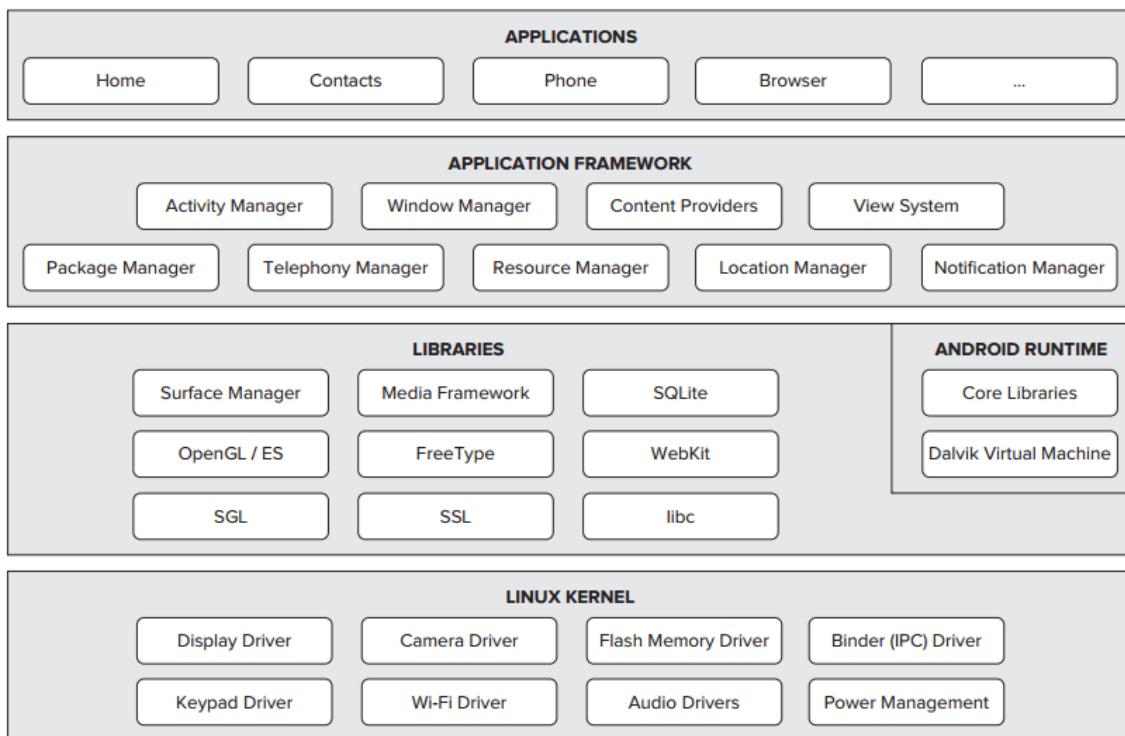
➤ Nhu cầu tuyển dụng lập trình viên Android

Với xu thế phát triển công nghệ di động nhanh và mạnh như hiện nay, thị trường thiết bị Android chiếm vị trí cao nhất không chỉ ở Việt Nam mà trên toàn thế giới, thì nhu cầu sử dụng các ứng dụng cho các thiết bị Android là rất lớn. Vì vậy, nhu cầu tuyển dụng lập trình viên Android cũng rất lớn và sẽ tăng nhanh.

1.2 | Giới thiệu nền tảng phát triển Android

1.2.1 | Kiến trúc

Hệ điều hành Android là một ngăn xếp của các thành phần phần mềm mà có thể đại khái phân chia thành 5 khu vực và 4 lớp chính.



Hình 1-15: Minh họa sơ đồ cấu trúc Android.

➤ Lớp Linux Kernel

Linux Kernel là lớp thấp nhất. Nó cung cấp các chức năng cơ bản như quản lý tiến trình, quản lý bộ nhớ, quản lý thiết bị như: Camera, bàn phím, màn hình, ... Ngoài ra, nó còn quản lý mạng, driver của các thiết bị, điều này gỡ bỏ sự khó khăn về giao tiếp với các thiết bị ngoại vi

➤ Libraries trong Android

Phía trên Linux Kernel là tập hợp các bộ thư viện mã nguồn mở WebKit, bộ thư viện nổi tiếng libc, cơ sở dữ liệu SQLite hữu ích cho việc lưu trữ và chia sẻ dữ liệu, bộ thư viện thể phát, ghi âm về âm thanh, hoặc video. Thư viện SSL chịu trách nhiệm cho bảo mật Internet

➤ Android Libraries

Phần này gồm các thư viện dựa trên Java. Nó bao gồm các Framework Library giúp xây dựng, vẽ đồ họa và truy cập cơ sở dữ liệu trở nên dễ dàng hơn. Dưới đây là một số Android Library cốt lõi có sẵn cho lập trình viên Android: –

➤ android.app: Cung cấp truy cập tới mô hình ứng dụng và nó là nền móng cho tất cả ứng dụng Android.

- android.content: Việc truy cập nội dung, các thông điệp giữa các ứng dụng và các thành phần ứng dụng trở nên dễ dàng hơn.
- android.database: Được sử dụng để truy cập dữ liệu được cung cấp bởi Provider và bao gồm các lớp quản lý cơ sở dữ liệu SQLite.
- android.opengl: Một Java Interface cho OpenGL ES 3D thông qua API.
- android.os: Cung cấp cho các ứng dụng sự truy cập tới các dịch vụ chuẩn của hệ điều hành như thông báo, dịch vụ hệ thống và giao tiếp nội tiền trình.
- android.text: Được sử dụng để phục hồi và thao tác text trên một thiết bị hiển thị.
- android.view: Các khái niệm trúc nền tảng của ứng dụng UI.
- android.widget: Một tập hợp các UI được xây dựng trước như button, label, list view, layout manager, radio button, ...
- android.webkit: Một tập hợp các lớp cho phép khả năng để trình duyệt trên web được xây dựng bên trong các ứng dụng.
- Android Runtime

Đây là thành phần thứ 3 trong cấu trúc, thuộc về lớp 2 tính từ dưới lên. Phần này cung cấp một thành phần quan trọng gọi là Dalvik Virtual Machine là một máy ảo Java đặc biệt, được thiết kế tối ưu cho Android.

Máy ảo Dalvik VM sử dụng các tính năng cốt lõi của Linux như quản lý bộ nhớ, đa luồng, mà thực chất là bên trong ngôn ngữ Java. Máy ảo Dalvik cho phép tất cả các ứng dụng Android chạy trong tiến trình riêng của nó

Android Runtime cũng cung cấp bộ thư viện cốt lõi, cho phép các lập trình viên Android sử dụng để viết các ứng dụng Android.

➤ Application Framework

Lớp Application Framework cung cấp nhiều dịch vụ cấp cao hơn cho các ứng dụng trong các lớp Java. Các lập trình viên cũng được phép sử dụng các dịch vụ này trong các ứng dụng của họ.

Application Framework bao gồm các dịch vụ chính sau:

- Activity Manager: Điều khiển các khía cạnh của vòng đời ứng dụng và Activity Stack.
- Content Providers: Cho phép các ứng dụng công bố và chia sẻ dữ liệu với các ứng dụng khác.
- Resource Manager: Cung cấp sự truy cập tới các resource được nhúng (không phải code) như chuỗi, thiết lập màu, UI layout.
- Notifications Manager: Cho phép các ứng dụng hiển thị thông báo tới người dùng.
- View System: Một tập hợp các view được sử dụng để tạo UI cho ứng dụng.

Application: sẽ thấy tất cả các ứng dụng Android ở lớp trên cùng. Ứng dụng viết sẽ được cài đặt vào lớp này.

Ví dụ của những ứng dụng này là Contacts Books, Browser, Games, ...

1.2.2 | Ngôn ngữ lập trình

- Java

Ngôn ngữ lập trình Java là một trong những ngôn ngữ ưa thích nhất khi phát triển ứng dụng Android. Một ngôn ngữ lập trình hướng đối tượng được phát triển tại Sun Microsystems (nay thuộc sở hữu của Oracle), Java có thể chạy theo hai cách khác nhau: trong cửa sổ trình duyệt, hoặc trong một máy ảo có thể làm mà không có trình duyệt.

- C++

Đây là ngôn ngữ lập trình thích hợp và mạnh mẽ nhất khi xây dựng các ứng dụng di động cho Android và Windows chủ yếu dành cho lập trình cấp thấp, nó vẫn là ngôn ngữ đi vào nền tảng cho các nhà phát triển ứng dụng trên điện thoại di động. Là một ngôn ngữ lập trình mạnh, thừa hưởng những ưu điểm của ngôn ngữ lập trình C, C++ cho phép các ứng dụng di động được phát triển cho mọi mục đích trên mọi nền tảng tồn tại. Nó có thể không được sang trọng hoặc hợp thời trang, nhưng nó đã thống trị thế giới lập trình ngay cả trước khi cuộc cách mạng điện thoại thông minh.

- C#

C # là một ngôn ngữ tuyệt vời. C# là tất cả mọi thứ về Java mà không có bất kỳ phần xáu, được lập trình tốt hơn từ tất cả các cải tiến hàng đầu. Microsoft đã nhìn thấy tiềm năng của Java và quyết định tạo một phiên bản tốt hơn của riêng họ.

Trong quá khứ, nhược điểm lớn nhất của C # là nó chỉ có thể chạy trên các hệ thống Windows vì nó dựa vào .Net Framework. Nhưng tất cả điều đó đã thay đổi khi Microsoft mở nguồn .NET Framework vào năm 2014 và mua lại Xamarin vào năm 2016, công ty duy trì Mono (một dự án cho phép các chương trình C # chạy trên nhiều nền tảng). Do đó vẫn có thể dùng được C# để lập trình ứng dụng Android.

Ngày nay, có thể sử dụng Xamarin.Android và Xamarin.iOS để tạo các ứng dụng di động bản địa với Visual Studio hoặc Xamarin Studio. Đây là một điều tuyệt vời bởi vì có thể sử dụng ngôn ngữ trong các ngữ cảnh khác sau này, chẳng hạn như thiết kế các trò chơi phức tạp với Unity và C#. Ví dụ về một ứng dụng được thiết kế với Xamarin? MarketWatch.

➤ Kotlin:

Kotlin là một ngôn ngữ phát triển dựa vào Java Virtual Machine được phát triển bởi JetBrains⁵ Công ty phát triển IntelliJ IDE. Các tính năng thú vị của Kotlin đó là trực quan và dễ học, hầu hết các phần của Kotlin rất giống với những gì chúng ta đã biết, IDE Android studio đã được kết hợp Kotlin free.

➤ HTML5 + CSS + JavaScript

Ba ngôn ngữ lập trình này, ban đầu là trifecta cốt lõi cho việc phát triển front-end web, đã phát triển trở nên hữu dụng hơn. Bây giờ có thể thiết kế đa dạng nhiều loại apps, cả điện thoại di động và máy tính để bàn, chỉ cần sử dụng HTML5, CSS và JavaScript.

Để thiết kế ứng dụng Android theo cách này, có thể sử dụng Adobe Cordova, một khuôn khổ mã nguồn mở cũng hỗ trợ iOS, Windows 10 Mobile, Blackberry, Firefox và nhiều hơn nữa. Nhưng bên cạnh những tính hữu dụng của nó, Cordova đòi hỏi rất nhiều công sức để tạo ra được một ứng dụng chạy tốt, đó là lý do tại sao nhiều người chọn Ionic Framework để thay thế (vì nó sử dụng Cordova để triển khai cho các nền tảng khác nhau).

Một sự lựa chọn khác là sử dụng React Native. Thư viện này có thể triển khai trên Android, iOS và nền tảng Windows chung. Nó được duy trì và sử dụng bởi Facebook, Instagram, và các công ty lớn khác.

➤ Python

Mặc dù Android không hỗ trợ phát triển Python bản địa nhưng vẫn có những công cụ cho phép tạo apps trên Python và sau đó chuyển đổi chúng thành các APK chạy thành công trên thiết bị Android.

1.3 | Môi trường phát triển ứng dụng

1.3.1 | Giới thiệu Java JDK, Android SDK, Android Studio

Android SDK (Software Development Kit) và JDK (Java Development Kit) là hai công cụ cần thiết để chúng ta có thể lập trình nên các ứng dụng Android. Và tất nhiên nếu không muốn lập trình trên phần mềm soạn thảo văn bản thì một công cụ lập trình IDE (Integrated development environment) sẽ rất hữu ích và tiện lợi. Eclipse được xem là một công cụ hỗ trợ rất tốt trong việc lập trình ứng dụng Android.

Android SDK, JDK và Eclipse đều có mặt trên một số phiên bản hệ điều hành Windows, Mac OS và Linux do đó chúng ta có thể lập trình trên hệ điều hành mà chúng ta đã quen sử dụng.Thêm nữa, Android được thực thi trên máy ảo Dalvik nên việc phát triển ứng dụng là như nhau trên cả 3 môi trường

Android Studio được Google chính thức phát hành phiên bản đầu tiên Android Studio 0.1 vào tháng 5/ 2013 (Phiên bản hiện nay là 1.2.1 phát hành vào tháng 5/2015 và phiên bản 1.3 đã được công bố tại Google I/O 2015). Là công cụ lập trình dựa trên nền IntelliJ, cung cấp các tính năng mạnh mẽ hơn ADT như:

- Hỗ trợ xây dựng dự án dạng Gradle.
- Hỗ trợ sửa lỗi nhanh và tái sử dụng cấu trúc phương thức.
- Cung cấp các công cụ kiểm tra tính khả dụng, khả năng hoạt động của ứng dụng, tương thích nền tảng...
- Hỗ trợ bảo mật mã nguồn và đóng gói ứng dụng.
- Trình biên tập giao diện cung cấp tổng quan giao diện ứng dụng và các thành phần, cho phép tùy chỉnh trên nhiều cấu hình khác nhau.
- Cho phép tương tác với nền Google Cloud.

Với mục tiêu tạo ra môi trường phát triển tất cả trong một, trải nghiệm nhanh và mượt hơn các IDE khác, Android Studio không ngừng ra đời các phiên bản cải tiến.

1.3.2 | Thiết lập môi trường phát triển Android Studio

Để bắt đầu viết ứng dụng với Android Studio, chúng ta cần tải và cài đặt hai bộ phần mềm sau:

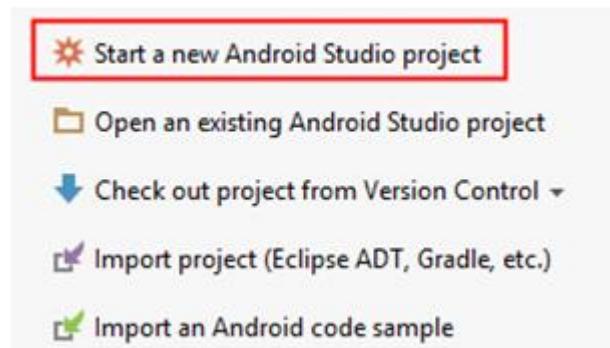
- Java JDK: <http://java.sun.com/javase/downloads/index.jsp> (Cài đặt trước hết và nên chọn phiên bản mới nhất).
- Android Studio: <http://developer.android.com/sdk/index.html> - tải gói Android Studio.

1.4 | Tạo ứng dụng đầu tiên

1.4.1 | Khởi tạo dự án

Bước 1: Tạo mới Project

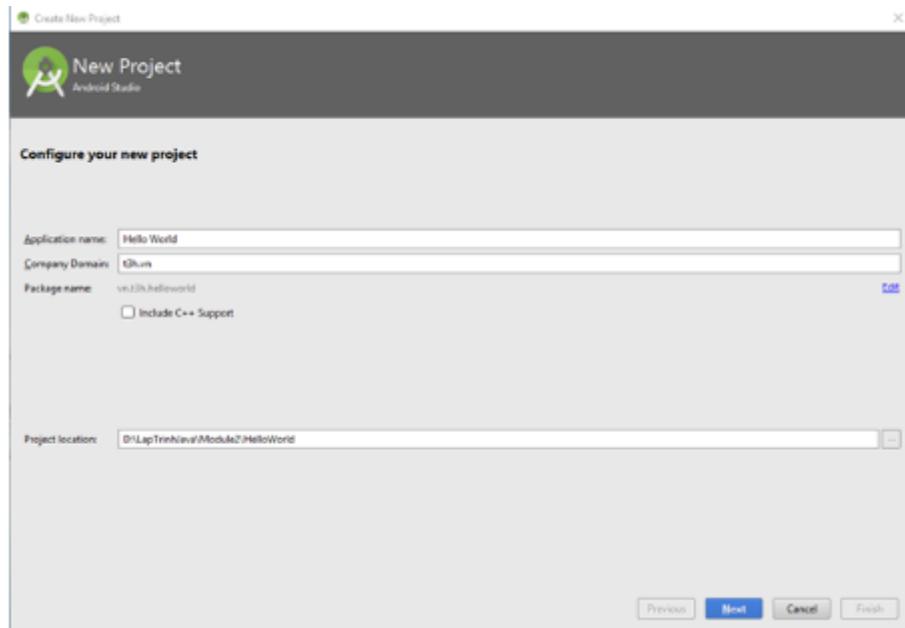
- Chọn Start a new Android Studio project



Hình 1-16: Tạo mới Android Studio project

Trong Android Studio, project giúp định nghĩa không gian làm việc của ứng dụng, bao gồm mã nguồn, các tài nguyên và các thông số cấu hình dùng để kiểm thử và build ứng dụng

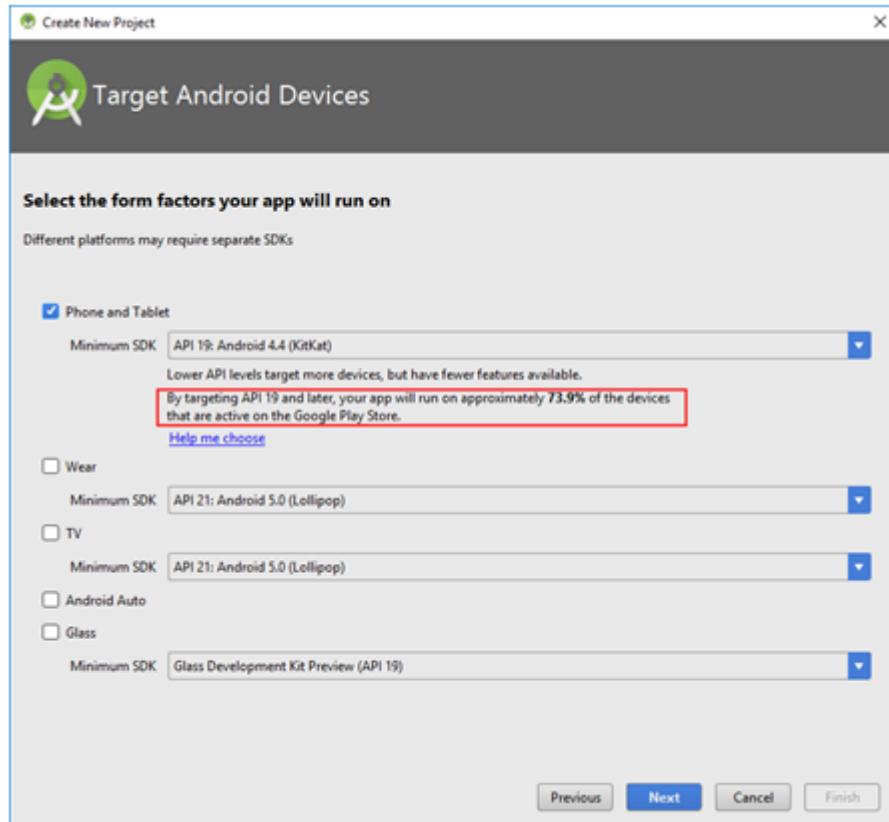
Bước 1.1: Đặt tên cho project



Hình 1-17: Đặt tên project

- **Application name:** Tên của ứng dụng, lưu ý phải viết HOA chữ cái đầu tiên của tên ứng dụng. Mặc định tên của ứng dụng cũng sẽ là tên Project.
- **Company Domain:** Tên domain của công ty. Dựa trên Application name và Company name, hệ thống sẽ tạo ra package name và thông tin này được sử dụng để đưa ứng dụng lên Google Play Project location: Đường dẫn trên máy dùng để lưu trữ ứng dụng.

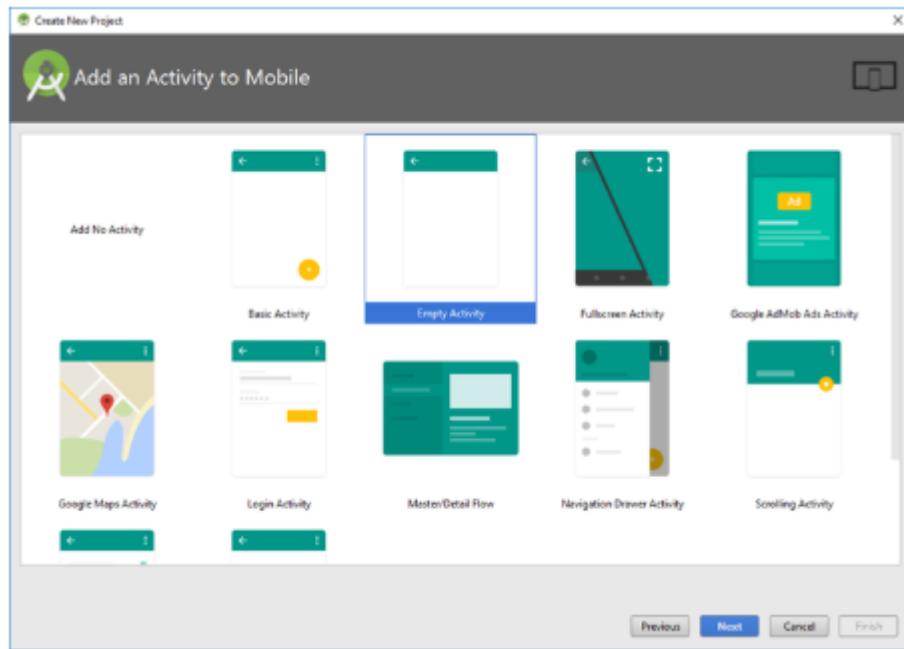
Bước 1.2: Chọn nền tảng để phát triển ứng dụng



Hình 1-18: Nền tảng ứng dụng

Phone and Tablet: chọn mục này để xác định mình đang phát triển ứng dụng trên điện thoại và máy tính bảng. Sau đó chọn **Minimum SDK**, là phiên bản API thấp nhất mà ứng dụng có thể cài đặt.

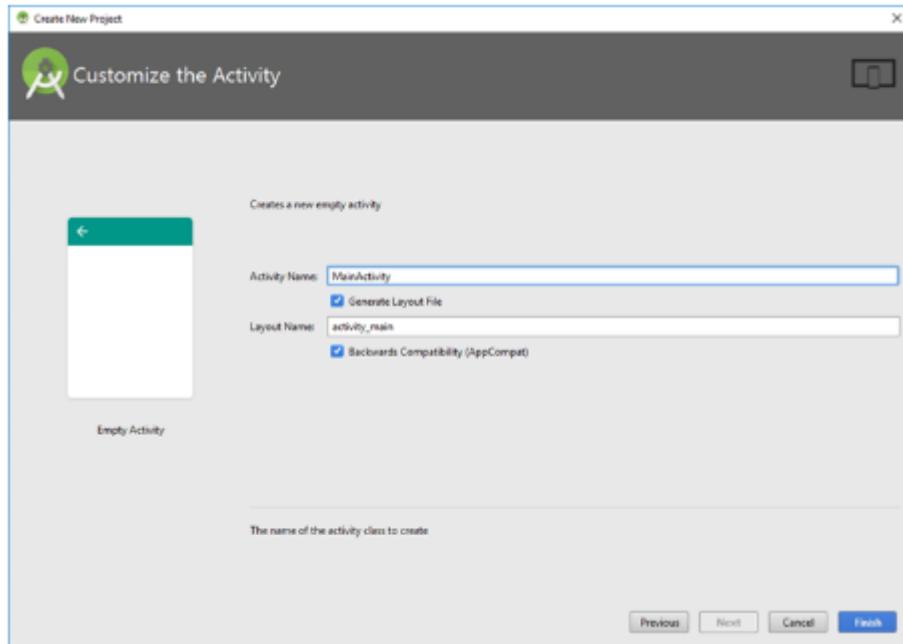
Bước 1.3: Tạo mới và đưa Activity vào ứng dụng



Hình 1-19: Tạo mới ứng dụng

Mỗi **Activity** là một màn hình giao diện người dùng, nơi người dùng tương tác, thực hiện một số thao tác tương ứng với chức năng của ứng dụng. Một ứng dụng có thể có nhiều **Activity** và sẽ có **Activity** hiển thị đầu tiên khi ứng dụng khởi động. Tương tự như khi lập trình Winform thì cũng có nhiều màn hình và sẽ có màn hình khởi động đầu tiên. Ở đây do chúng ta viết một ứng dụng đơn giản nên chúng ta chọn **Empty Activity**.

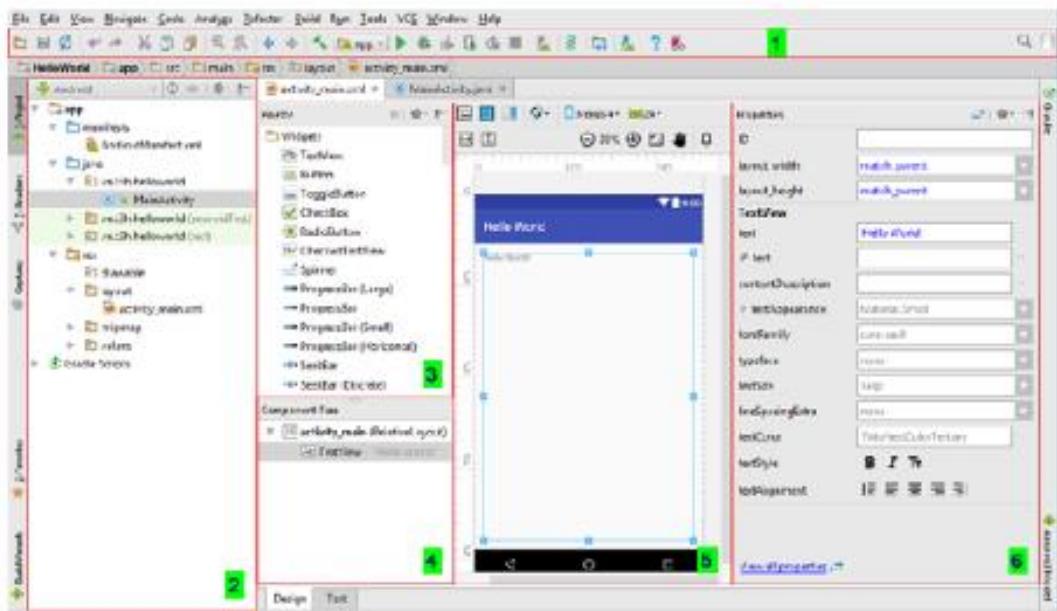
Bước 1.4: đặt tên cho Activity Name và Layout Name.



Hình 1-20: đặt tên cho Activity Name và Layout Name

Do ứng dụng chúng ta chỉ có một **Activity**, trên đó sẽ hiện dòng chữ “Hello world” nên có thể để mặc định các thông số như gợi ý. Trong Android, tương ứng với mỗi **Activity** khi tạo ra sẽ có một tập tin lưu source code (**.java**) và một tập tin là mô tả giao diện của Activity (**.xml**). Trong trường hợp này, **Activity** của chúng ta là **MainActivity** nên hai tập tin đó là **MainActivity.java** và view layout sẽ có tên là **activity_main.xml**. Nhấn nút **Finish** để hoàn tất các bước tạo ứng dụng đầu tiên.

Lúc này giao diện của Android Studio sẽ hiện ra như sau



Hình 1-21:Giao diện các vùng ứng dụng ban đầu

- + **Vùng 1:** Thanh công cụ giúp thao tác nhanh các chức năng thường dùng khi lập trình trong Android Studio. Trong đó, quan trọng là chức năng Run  , Debug ứng dụng  và quản lý máy ảo 
- + **Vùng 2:** Cấu trúc hệ thống tài nguyên của ứng dụng
 - ☞ **Thư mục manifests:** chứa thông tin cấu hình của ứng dụng
 - ☞ **AndroidManifest.xml:** tập tin XML chứa tất cả các thông tin cấu hình dùng để build ứng dụng và các thành phần của ứng dụng (activity, service,...). Mỗi ứng dụng đều có một tập tin *AndroidManifest.xml*. Trong ứng dụng, Activity nào muốn sử dụng đều bắt buộc phải có khai báo *AndroidManifest.xml*

```

<manifest xmlns:android='http://schemas.android.com/apk/res/android'
    package='vn.edu.tdc.chuong4vidu'>

    <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>

```

- ☞ Tập tin **AndroidManifest.xml** của ứng dụng mới tạo Hello world
- ☞ **Thu mục java:** chứa tất cả các file mã nguồn .java của ứng dụng
- ☞ Lúc này do ứng dụng của chúng ta chỉ có một màn hình màn hình **MainActivity** nên các chỉ thấy **MainActivity.java**. Tương ứng với mỗi **Activity** thì file mã nguồn sẽ chứa các xử lý trên Activity đó. Activity nào được khởi chạy đầu tiên khi ứng dụng hoạt động sẽ được khai báo đầu tiên trong tập tin **AndroidManifest.xml**.
- ☞ **Thu mục res:** chứa các tài nguyên của ứng dụng, bao gồm các tập tin hình ảnh, các thiết kế giao diện, thực đơn,... của ứng dụng
- + **Vùng 3:** Danh sách các control mà Android Studio hỗ trợ, có thể tạo giao diện bằng cách kéo thả trực tiếp các control này vào vùng giao diện (Vùng 5) và Android Studio sẽ phát sinh ra mã lệnh XML cho .
- + **Vùng 4:** hiển thị giao diện theo cấu trúc cây giúp dễ dàng quan sát và lựa chọn control.
- + **Vùng 5:** vùng giao diện của thiết bị cho phép kéo thả các control. Chúng ta có thể chọn cách hiển thị theo chiều nằm ngang, nằm đứng, phóng to, thu nhỏ, lựa chọn các loại thiết bị hiển thị...
- + **Vùng 6:** Cửa sổ thuộc tính của control đang chọn, cho phép thiết lập các thuộc tính cần thiết.

Bước 2: Tạo máy ảo

Quá trình tạo máy ảo tương đối mất thời gian nên để tiết kiệm thời gian nên làm trước, rồi trong thời gian chờ máy ảo khởi động sẽ viết code cho ứng dụng để đến lúc viết xong có thể build ứng dụng ngay.

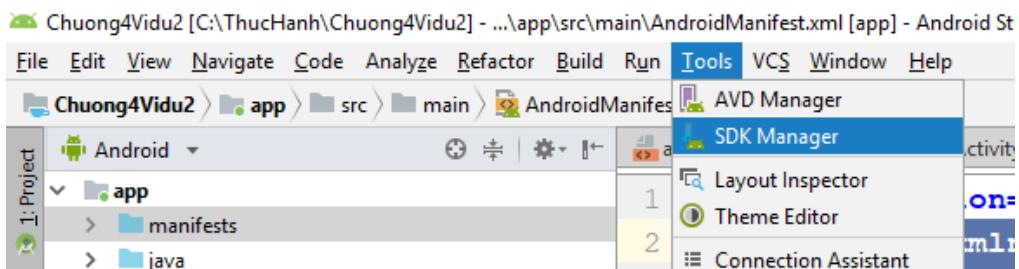
Có 2 cách để tạo máy ảo

☞ Cách 1: Chọn biểu tượng AVD Manager trên thanh Toolbar



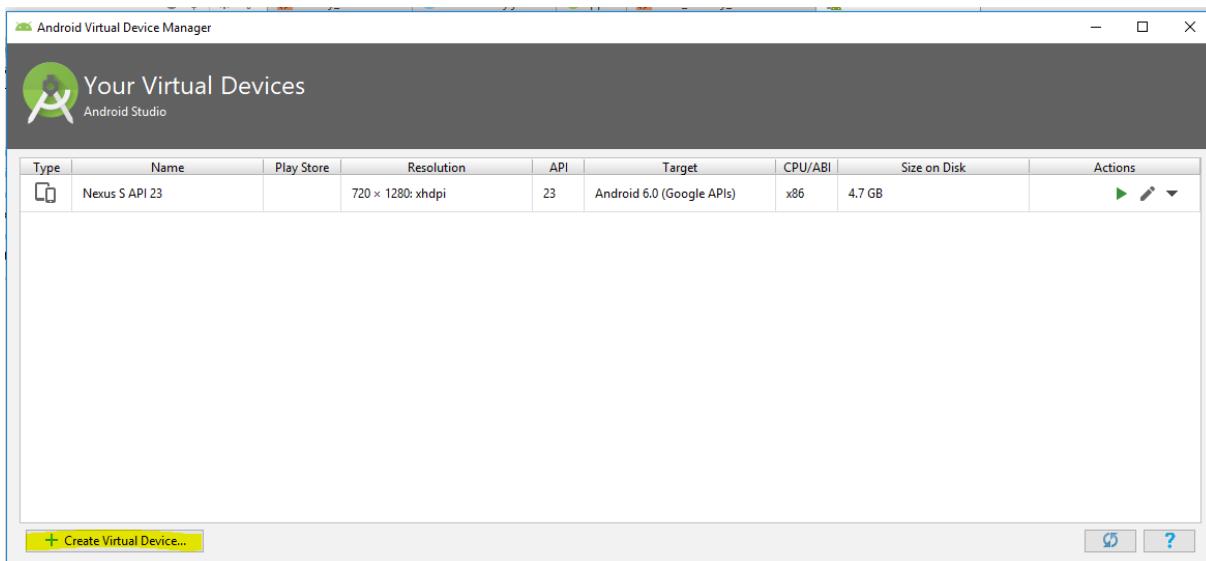
Hình 1-22: Cách 1 tạo máy ảo

☞ Cách 2: Chọn tab Tools → Android → AVD Manager



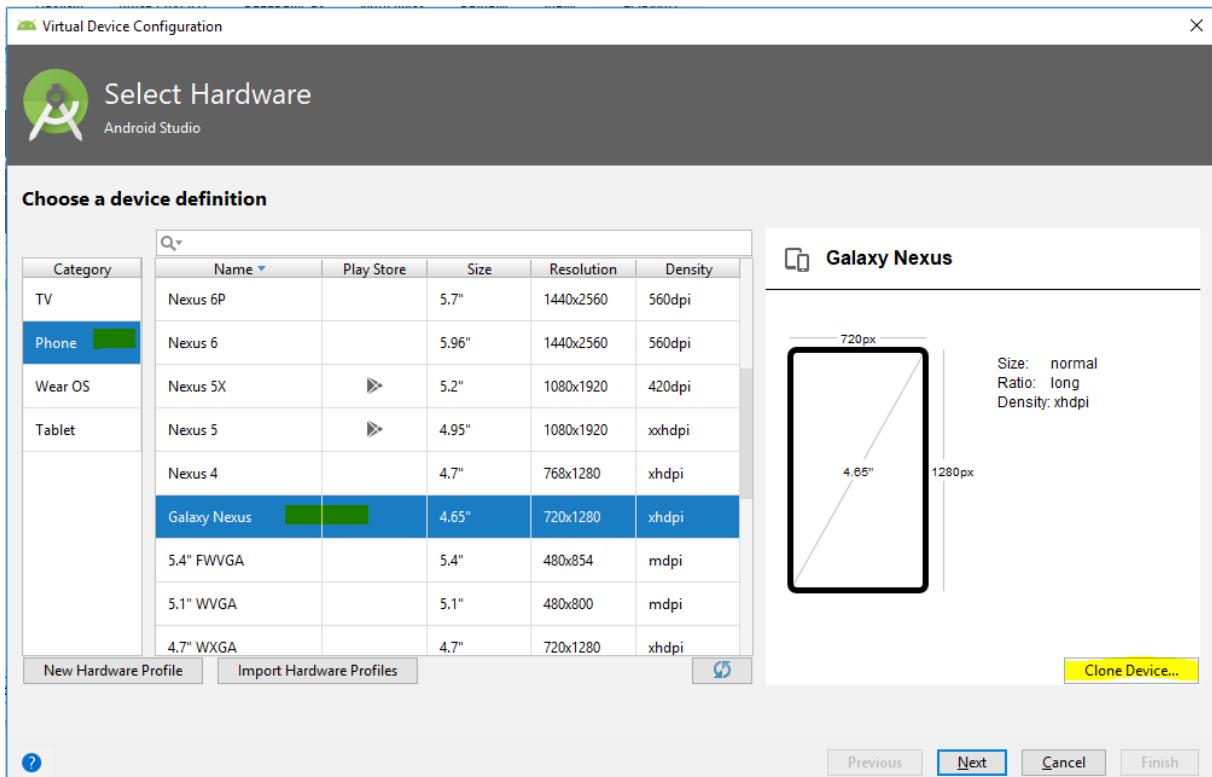
Hình 1-23: Cách 2 tạo máy ảo

Bước 2.1: Chọn Create Virtual Device



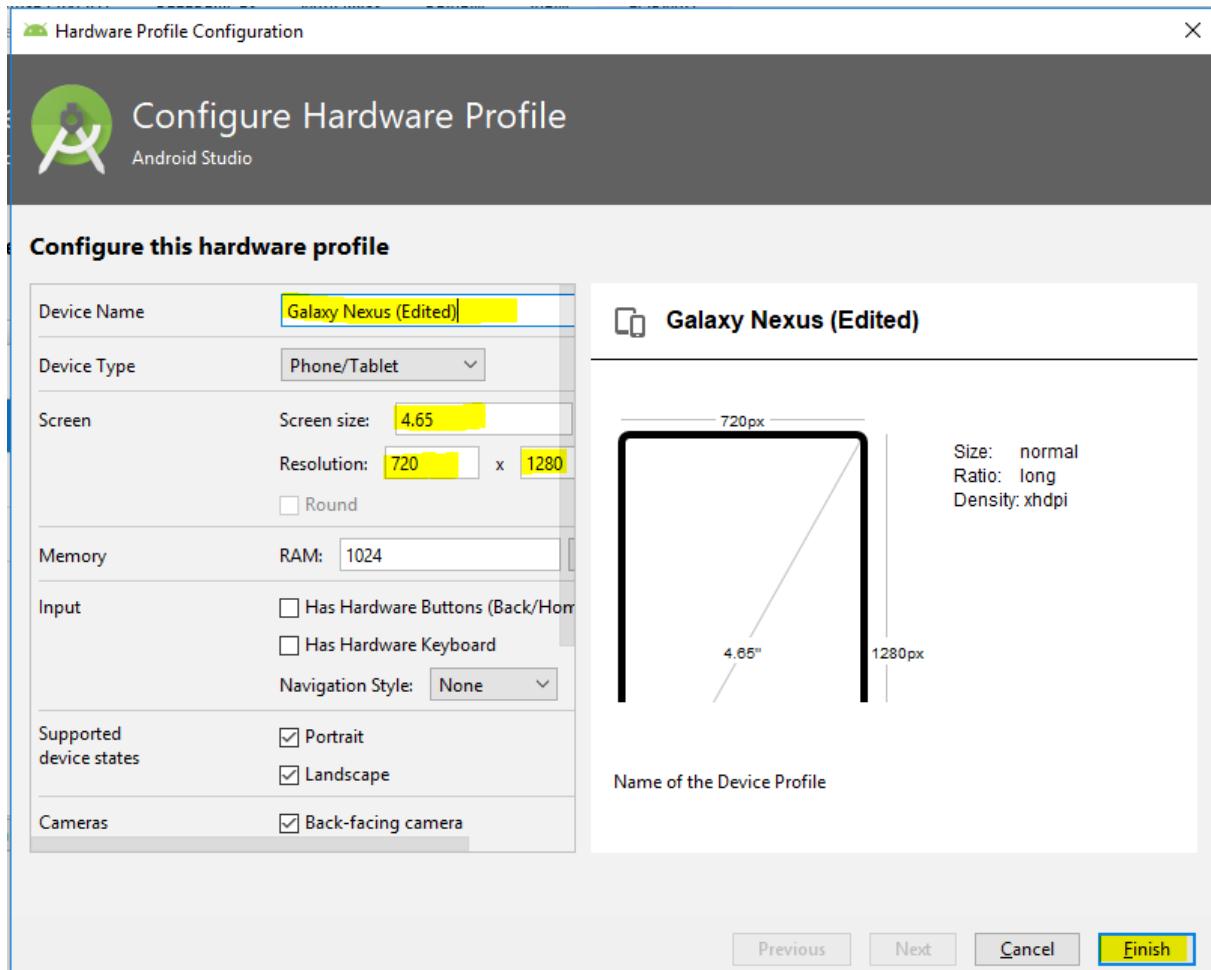
Hình 1-24: Tạo máy ảo

Bước 2.2: Chọn máy ảo muốn tạo. Ta có thể chỉnh sửa lại thông số máy ảo bằng cách click vào Clone Device



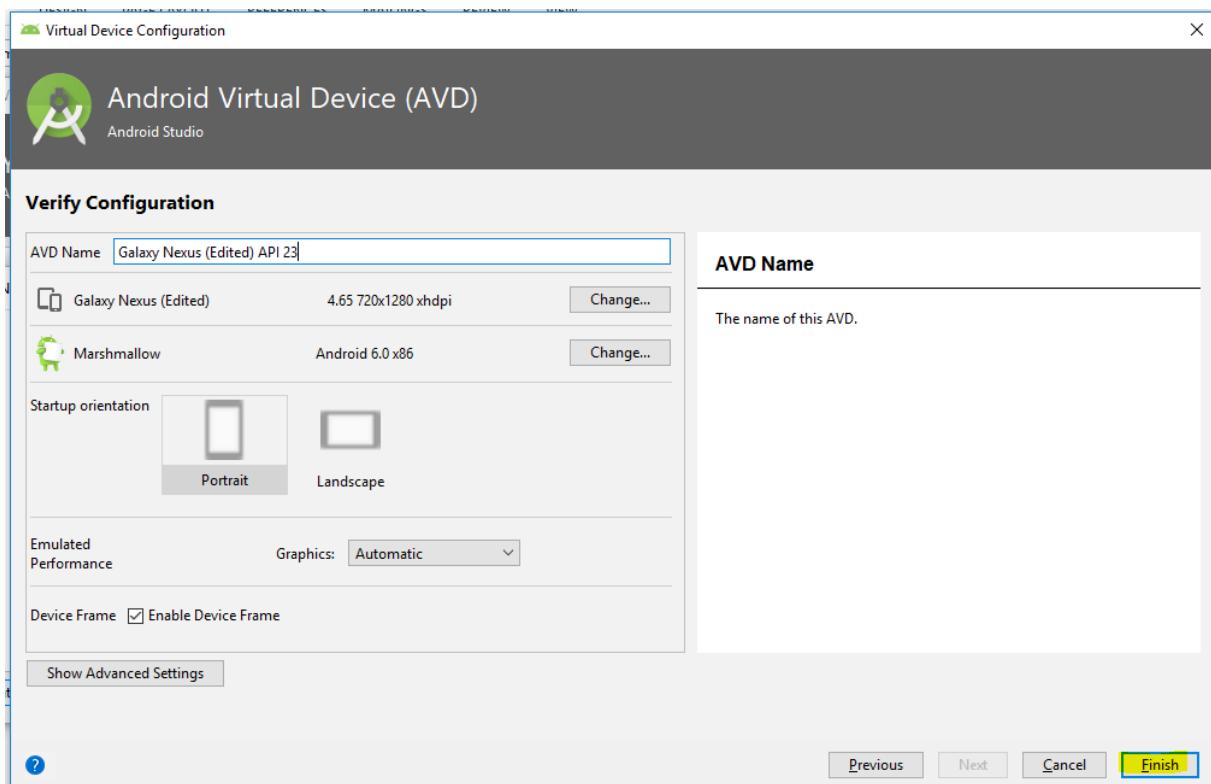
Hình 1-25: Chọn máy ảo

Bước 2.3: chọn hệ điều hành Android cho máy ảo. Trong bộ Android SDK đã download hệ điều hành có những API nào thì sẽ thấy có tất cả ở đây



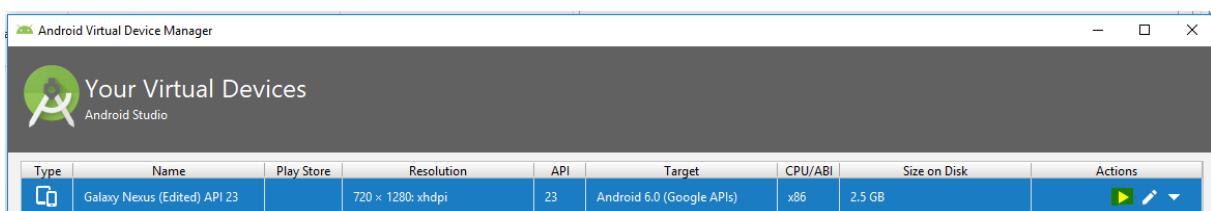
Hình 1-26: Config Hardware Profile

Bước 2.4: Click vào Finish để hoàn thành quá trình tạo máy ảo



Hình 1-27: Verify Config

Bước 2.5: click vào biểu tượng để chạy máy ảo



Hình 1-28: Chạy máy ảo

Bước 2.6: Máy ảo Android đã khởi động xong



Hình 1-29: Máy ảo

Bước 3: build và thực thi ứng dụng, có thể build ứng dụng bằng cách click vào biểu tượng ➤



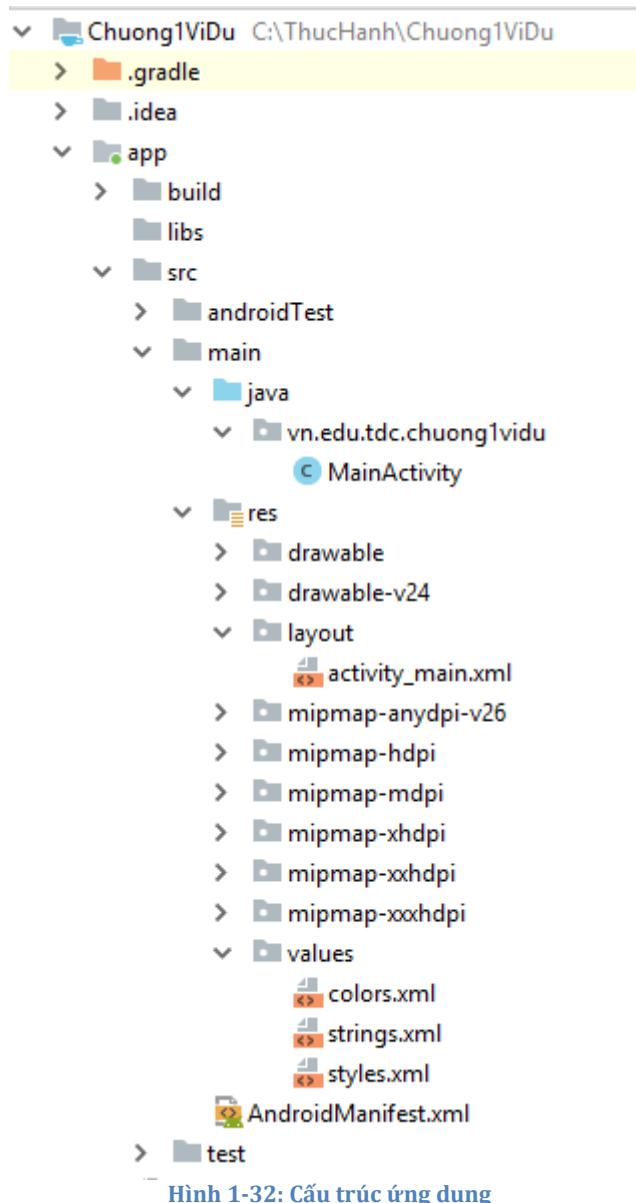
Hình 1-30: Thực thi ứng dụng

Lúc này chúng ta sẽ có kết quả như sau:



Hình 1-31: Màn hình ứng dụng

1.4.2 | Cấu trúc dự án



Hình 1-32: Cấu trúc ứng dụng

Main Activity code là một Java file với tên ***MainActivity.java***. Đây là một file ứng dụng thực sự mà cuối cùng được chuyển đổi thành một Dalvik có thể thực thi và chạy ứng dụng của. Sau đây là phần code mặc định được tạo cho *Hello World!*

```

public class MainActivity extends AppCompatActivity {

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

```

```
}
```

```
}
```

- ☞ *strings.xml* được đặt trong thư mục *res/values* và nó chứa tất cả text mà ứng dụng của sử dụng. Ví dụ, tên của các button, label, text mặc định, và các kiểu tương tự chuỗi vào trong file này. File này chịu trách nhiệm cho nội dung thuận văn bản. Ví dụ, một strings file mặc định sẽ trông như sau:

```
<resources>
    <string name="app_name">ChuongIViDu</string>
</resources>
```

- ☞ *activity_main.xml* là một layout file có sẵn trong thư mục *res/layout* mà được tham chiếu bởi ứng dụng của khi xây dựng giao diện. sẽ sửa đổi file này khá thường xuyên để thay đổi layout của ứng dụng. Với ứng dụng Hello World, file này sẽ có nội dung sau liên quan tới layout mặc định:

```
<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=".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>`

❖ Resource trong Android

Thư mục	Kiểu Resource
anim/	Các XML file định nghĩa thuộc tính hiệu ứng. Chúng được lưu giữ trong thư mục res/anim và được truy cập từ lớp R.anim .
color/	Các XML file định nghĩa danh sách trạng thái của màu. Chúng được lưu giữ trong res/color/ và được truy cập từ lớp R.color .
drawable/	Các Image file như .png, .jpg, .gif hoặc XML file mà được biên dịch vào trong bitmap, state list, shape, animation drawable. Chúng được lưu giữ trong res/drawable/ và được truy cập từ lớp R.drawable
layout/	XML files that define a user interface layout. They are saved in res/layout/ and accessed from the R.layout class.
menu/	XML file định nghĩa một UI layout. Chúng được lưu giữ trong res/layout/ và được truy cập từ lớp R.menu
raw/	Các file riêng để lưu giữ trong dạng raw from. cần gọi Resources.openRawResource() với resource ID, mà là R.raw.filename để mở các raw file này
values/	XML file chứa các giá trị đơn giản, ví dụ chuỗi, số nguyên và màu. Ví dụ, dưới đây là một số tên file qui ước cho các Resource có thể tạo trong thư mục này: – <ul style="list-style-type: none">✓ arrays.xml cho các mảng và được truy cập từ lớp R.array✓ integers.xml cho số nguyên và được truy cập từ lớp R.integer

	<ul style="list-style-type: none"> ✓ <i>bools.xml</i> cho boolean, và được truy cập từ lớp R.bool class. ✓ <i>colors.xml</i> cho các giá trị màu, và được truy cập từ lớp R.color ✓ <i>dimens.xml</i> cho các giá trị chiều, và được truy cập từ lớp R.dimens ✓ <i>strings.xml</i> cho các giá trị chuỗi, và được truy cập từ lớp R.string ✓ <i>styles.xml</i> cho các style, và được truy cập từ lớp R.style
xml/	Các XML file riêng có thể được đọc tại runtime bởi gọi phương thức Resources.getXML() . có thể lưu giữ các file cấu hình đa dạng tại đây, các file này sẽ được sử dụng tại runtime

1.4.3 | **AndroidManifest**

Tập tin *AndroidManifest.xml* chứa thông tin về package của ứng dụng, bao gồm các thành phần của ứng dụng như *activities*, *services*, *broadcast receivers*, *content providers*

- ☞ Nó thực hiện một số nhiệm vụ khác nhau:
- ☞ Cấp quyền một số phần trong ứng dụng
- ☞ Khai báo các API mà ứng dụng sẽ sử dụng
- ☞ Khai báo các thông tin về ứng dụng

Dưới đây là một tập tin *AndroidManifest.xml* đơn giản:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="vn.edu.tdc.chuong1vidu">
```

```
<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>

```

☞ Các phần tử của tập tin *AndroidManifest.xml*

Các phần tử được sử dụng trong tập tin xml trên được mô tả:

- ☞ **<manifest>:manifest** là phần tử đầu tin trong tập tin AndroidManifest.xml, có thuộc tính **package** mô tả tên **package** của class activity.
- ☞ **<application>: application** là phần tử con của manifest. Nó khai báo namespace, phần tử này chứa nhiều phần tử con được khai báo trong thành phần (component) của ứng dụng như: Activity ..
- ☞ Các thuộc tính của các phần tử thường được sử dụng: **icon**, **label**, **theme** v.v
- ☞ **android:icon**: Thuộc tính này là nơi bạn thiết lập icon cho ứng dụng. Icon này sẽ xuất hiện trên màn hình chính của thiết bị. Hiện tại thì icon này đang được để mặc định là file .
- ☞ **android:label**: Thuộc tính này trỏ đến một giá trị *string* trong resource, hoàn toàn tương tự như cách bạn khai báo text cho *TextView* ở bài trước vậy. String này chính là tên ứng dụng .
- ☞ **android:theme**: Thuộc tính này khá hay, nó giúp chúng ta định nghĩa “*giao diện chủ đề*” cho ứng dụng. Và vì nó hay quá nên mình sẽ không nói ở đây, mình dành hẳn một bài để nói về chủ đề này để cho các bạn hiểu rõ nhất về nó.
- ☞ **ic_launcher.png**: để trong thư mục *res/mipmap/*. Cách sử dụng icon cho ứng dụng.

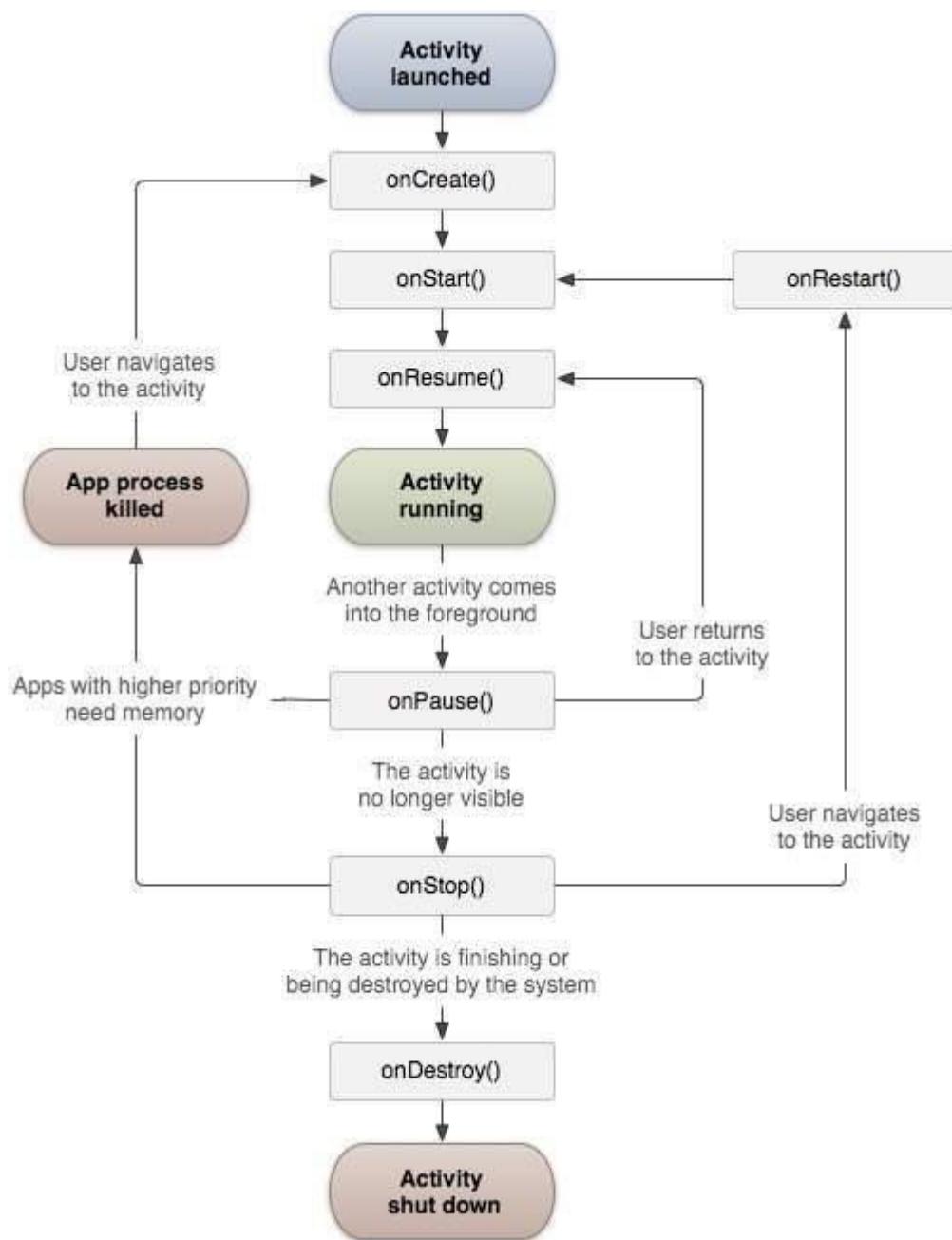
- ☞ **<activity>:** activity là phần tử con của ứng dụng, một activity phải được định nghĩa trong tập tin AndroidManifest.xml. Nó có nhiều thuộc tính: **label, name, theme, launchMode**
 - ☞ **android:label:** Tiêu đề của activity được hiển thị trên màn hình.
 - ☞ **android:name:** Tên class của activity. Thuộc tính này bắt buộc.
 - ☞ **<intent-filter>:** intent-filter là phần tử giúp cho hệ thống Android biết được ứng dụng của bạn có thể làm được những gì.
 - ☞ **<action>:** Phần tử này một hành động cho intent-filter. intent-filter có ít nhất một action:
 - ☞ **<category>:** Thêm tên category cho một intent-filter:

1.5 | Quản lý trạng thái Activity.

1.5.1 | Xây dựng Activity

Một Activity biểu diễn một màn hình với một giao diện UI giống như Window hoặc Frame của Java.Android activity, mà là một lớp con của lớp ContextThemeWrapper. Nếu đã từng làm việc với ngôn ngữ lập trình C, C++ hoặc Java thì phải thấy rằng chương trình của bắt đầu từ hàm main(). Tương tự, hệ điều hành Android khởi tạo chương trình của nó bên trong một Activity bắt đầu với một lời gọi trên phương thức callback là onCreate(). Có một dãy các phương thức callback mà khởi động một Activity và một dãy phương thức callback khác sẽ hủy một Activity như sau trong sơ đồ vòng đời của Activity. (image courtesy : android.com)

1.5.2 | Vòng đời của Activity



Hình 1-33: Vòng đời Activity

Khi Activity được kích hoạt, và được hệ thống để vào BackStack. Sau khi kích hoạt, lần lượt các callback `onCreate()`, `onStart()`, `onResume()` sẽ được hệ thống gọi đến. Sau khi gọi đến các callback trên, thì Activity mới chính thức được xem là đang chạy (Activity running). Lúc này, nếu có bất kỳ Activity nào khác chiếm quyền hiển thị, thì Activity hiện tại sẽ rơi vào trạng thái `onPause()`. Nếu cái sự hiển thị của Activity khác làm cho Activity mà chúng ta đang nói đến không còn nhìn thấy nữa thì `onStop()` sẽ được gọi tiếp theo nữa. Nếu Activity đã vào `onPause()` rồi, tức là đang bị Activity khác đè lên, mà người dùng sau đó quay về lại Activity cũ, thì `onResume()` được gọi. Còn nếu Activity đã vào `onStop()` rồi, mà người dùng quay về lại Activity cũ thì `onRestart()`

được gọi. Trong cả hai trường hợp Activity rời vào onPause() hoặc onStop(), nó sẽ rất dễ bị hệ thống thu hồi (tức là bị hủy) để giải phóng tài nguyên, khi này nếu quay lại Activity cũ, onCreate() sẽ được gọi chứ không phải onResume() hay onRestart(). Và cuối cùng, nếu một Activity bị hủy một cách có chủ đích, chẳng hạn như người dùng nhấn nút Back ở System Bar, hay hàm finish() được gọi,... thì onDestroy() sẽ được kích hoạt và Activity kết thúc vòng đời của nó.

Callback	Miêu tả
onCreate()	Đây là phương thức callback đầu tiên và được gọi khi Activity được tạo đầu tiên
onStart()	Sau khi gọi đến onCreate(), hệ thống sẽ gọi đến onStart(). Hoặc hệ thống cũng sẽ gọi lại onStart() sau khi gọi onRestart() nếu trước đó nó bị che khuất bởi Activity nào khác (một màn hình khác hoặc một ứng dụng khác) che hoàn toàn và rời vào onStop().
onResume()	Được gọi khi người dùng bắt đầu tương tác với ứng dụng
onPause()	Activity tạm dừng không nhận input từ người dùng và không thể thực thi bất cứ code nào và được gọi khi activity hiện tại đang được dừng và activity trước đó đang được phục hồi
onStop()	Callback này được gọi trước khi activity bị hủy bởi hệ thống
onDestroy()	Callback này được gọi trước khi activity bị hủy bởi hệ thống
onRestart()	Được gọi khi activity tái khởi động sau khi dừng nó

1.6 | Debug chương trình trong Android Studio

1.6.1 | Sử dụng Log và Toast

Khi thực thi ứng dụng muốn kiểm tra xem đã xảy ra lỗi gì hay hệ thống có cảnh báo gì, hoặc chỉ là các thông tin thực thi bình thường chúng ta có thể sử dụng

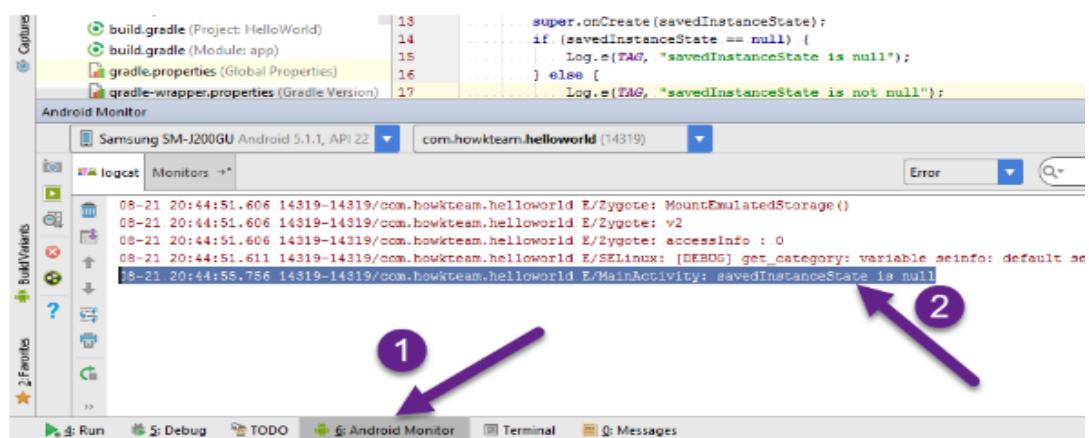
LogCat. Trong quá trình xây dựng ứng dụng, nếu muốn in ra giá trị của một biến đơn nào đó tại một thời điểm bất kỳ chúng ta có sử dụng:

1.6.1.1 / Log

Bước 1: Đặt Log trong code:

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    Log.d("Test", "Tdc");  
}
```

Bước 2: Đọc Log

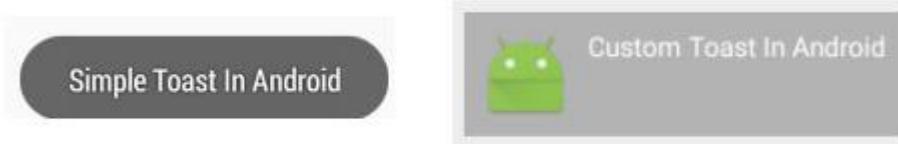


Hình 1-34: Đọc log

- ☞ Số 1: Chọn Android Monitor.
- ☞ Số 2: Đọc thông tin

1.6.1.2 / Toast

Trong Android, Toast dùng để hiển thị thông tin trong khoảng thời gian ngắn. nó giống như một thông báo nổi trên ứng dụng, không ngăn cản tương tác người dùng. chúng ta có thể tùy biến lại **Toast**



Toast

Custom Toast With Image

Hình 1-35: Các hình dạng Toast

- Các phương quan trọng của Toast
 - ☞ **makeText(Context context, CharSequence text, int duration)**: Phương thức này thường sử dụng để hiện thị thông báo. Phương thức này có 3 tham số:
 - **Context context**: thường là YourActivity.this của bạn vào nếu như bạn đang sử dụng ở Activity.
 - **CharSequence text**: đây chính là nội dung bạn muốn show lên, ở đây là kiểu String
 - **int duration**: Khoảng thời gian Toast cần hiển thị, nó là hằng số. Hằng số của Toast: Dưới đây là các hằng số của Toast được sử dụng để thiết lập thời gian cho Toast.
 - **LENGTH_LONG**: Toast sẽ hiển thị trong 3,5 giây
 - **LENGTH_SHORT**: Toast sẽ hiển thị trong 2 giây.
 - ☞ **show()**: phương thức này hiển thị thông báo ra màn hình. Phương pháp này hiển thị thông báo bằng cách sử dụng phương thức **makeText()** của Toast.

1.6.2 | Sử dụng Debug

Android Studio có cung cấp khả năng debug ứng dụng rất hiệu quả cho các ứng dụng chạy trên máy thật lẫn máy ảo

- Chọn thiết bị để debug.
- Đặt các breakpoint (điểm dừng) trong code.
- Quan sát và kiểm tra các giá trị biến / biểu thức trong runtime.



```

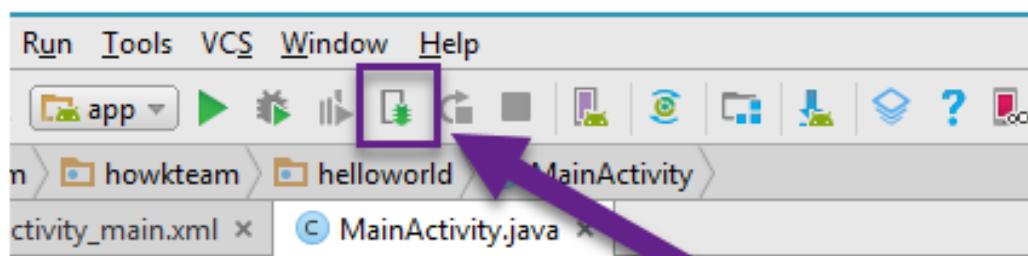
7 public class MainActivity extends AppCompatActivity {
8     ...
9     ...
10    ...
11    ...
12    ...
13    ...
14    ...
15    ...
16    ...
17    ...
18    ...
19    ...
20    ...
21    ...
22    ...
23

```

Click vào đây hoặc Ctrl+F8

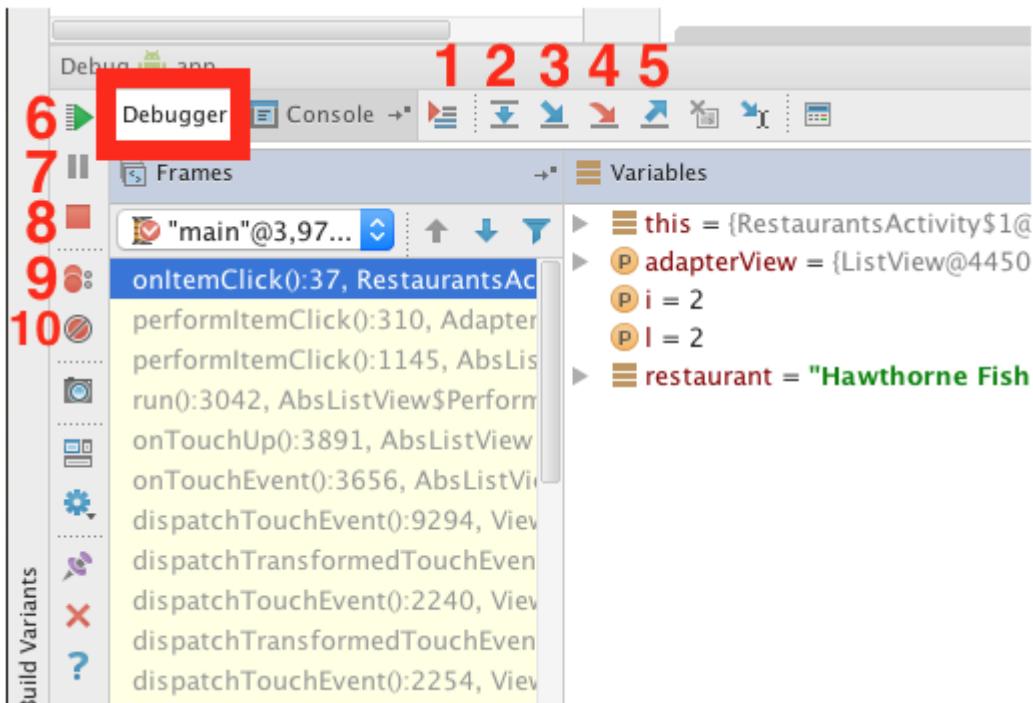
Hình 1-36: Đặt các breakpoint

Để bắt đầu debug, các nhấn vào nút **Debug**  trên thanh công cụ (khi trỏ vào sẽ có chữ “Debug ‘app’”).



Hình 1-37: Chọn Debug

Lúc này Android Studio sẽ build ứng dụng ra file APK, ký (sign) file APK bằng key debug, và cài đặt lên thiết bị của. Cuối cùng, cửa sổ **Debug** sẽ được mở ra:



Hình 1-38: Quy trình debug

- ☞ Số 1: Nút hiển thị Breakpoint đang active
- ☞ Số 2. Step Over, nút này sẽ giúp debug nhảy xuống dòng code tiếp theo
- ☞ Số 3. Step Into, nút này sẽ nhảy vào bên trong hàm
- ☞ Số 4. Force Step Into, nút này sẽ cho phép nhảy thẳng đến dòng đầu tiên bên trong của hàm được gọi
- ☞ Số 5. Thoát ra ngoài
- ☞ Số 6. Tiếp tục chương trình(Resume Program), nút này sẽ tiếp tục chạy ứng dụng một cách bình thường. Tạm thời bỏ qua debug
- ☞ Số 7. Tạm dừng chương trình(Pause Program)
- ☞ Số 8. Dừng ứng dụng (Stop App)
- ☞ Số 9. Xem các Breakpoints
- ☞ 10. Mute Breakpoint, tắt tạm thời tất cả các breakpoint.

1.7 | Bài tập Chương 1

Bài tập 1.1. Phân biệt reground Lifetime và Visible Lifetime

Đề bài:

Tạo 3 Activity đặt tên là: ManHinhChinh, ManHinh1, ManHinh2.

Giao diện màn hình chính như sau:



Khi ManHinh1 được kích hoạt thì nó sẽ nằm phía trên ManHinhChinh vẫn thấy được màn hình chính



Khi ManHinh2 hiển thị thì nó sẽ chiếm toàn bộ màn hình không thể thấy được màn hình chính.

Bắt các sự kiện khi Activity chuyển đổi trạng thái và thông báo lên màn hình Android.

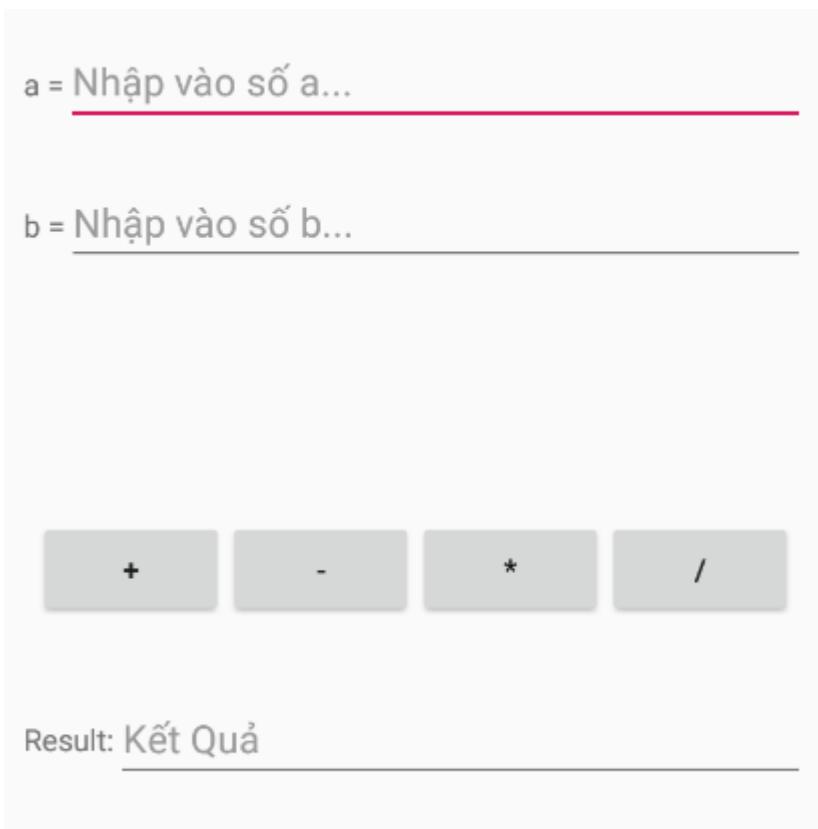
Mục tiêu:

- Hiểu sâu thêm về các phương thức đặt trạng thái trong Activity.
- Phân biệt được giữa Foreground Lifetime và Visible Lifetime.

Gợi ý thực hiện:

- Tạo mới Android Application Project, đặt tên là VongDoiActivity.
- Tạo 2 activity đặt tên lần lượt như sau: ManHinh1 và ManHinh2.
- Thiết kế giao diện cho ManHinhChinh, ManHinh1, ManHinh2 như yêu cầu của đề bài (sử dụng Button).
- Viết code xử lý trong lớp ManHinhChinh.java.

Bài tập 1.2. Sử dụng ConstraintLayout xây dựng ứng dụng



Mục tiêu:

- Xây dựng giao diện bằng ConstraintLayout
- Xây dựng chức năng xử lý cơ bản

CHƯƠNG 2. Giao diện người dùng và xử lý sự kiện

MỤC TIÊU THỰC HIỆN

- Xây dựng được giao diện cho ứng dụng

- Xử lý được các sự kiện cho ứng dụng
- Sử dụng thành thạo IDE Android Studio để viết chương trình Android
- Hình thành thói quen thiết kế chương trình theo tiếp cận Top-Down

2.1 | Giao diện người dùng

Giao diện người dùng là một trong những yếu tố quan trọng, quyết định sự thành công của một ứng dụng Android. Ứng dụng Android muốn thành công thì phải có giao diện trực quan, dễ hiểu và dễ sử dụng. Ở bài này, chúng ta sẽ tìm hiểu cấu trúc giao diện, các thành phần trên giao diện và các thuộc tính của chúng. Giao diện được tạo bởi nhiều tài nguyên như: layout, các điều khiển (control), tài nguyên hình ảnh, màu sắc, v.v... Và tất cả các tài nguyên dùng để thiết kế giao diện sẽ được lưu trong thư mục **res/**.

Các loại Layout trong Android

Số	Mô tả
1	<p>Linear Layout</p> <p>LinearLayout là một view group mà căn chỉnh các view con theo một hướng nào đó: chiều dọc hay chiều ngang</p>
2	<p>Relative Layout</p> <p>RelativeLayout là một view group mà hiển thị các view con trong các vị trí cân xứng với nhau</p>
3	<p>Table Layout</p> <p>TableLayout là một view mà nhóm tất cả các view vào trong các hàng và các cột</p>

4	Absolute Layout AbsoluteLayout cho phép xác định vị trí chính xác của các view con
5	Frame Layout FrameLayout là một placeholder trên màn hình mà có thể sử dụng để hiển thị một view đơn

Các thuộc tính trong Android

Attribute	Miêu tả
android:id	Đây là ID mà nhận diện duy nhất View
android:layout_width	Đây là độ rộng của Layout
android:layout_height	Đây là chiều cao của Layout
android:layout_marginTop	Đây là không gian phụ (extra space) trên cạnh trên của Layout
android:layout_marginBottom	Đây là extra space trên cạnh dưới của Layout
android:layout_marginLeft	Đây là extra space trên cạnh trái của Layout
android:layout_marginRight	Đây là extra space trên cạnh phải Layout
android:layout_gravity	Xác định cách các view con được đặt tại đâu
android:layout_weight	Xác định có bao nhiêu extra space trong Layout

	nên được cấp phát tới View đó
android:layout_x	Xác định tọa độ x của Layout
android:layout_y	Xác định tọa độ y của Layout
android:layout_width	Đây là độ rộng Layout
android:layout_height	Đây là chiều cao Layout
android:paddingLeft	Đây là left padding được điền cho Layout
android:paddingRight	Đây là right padding được điền cho Layout
android:paddingTop	Đây là top padding được điền cho Layout
android:paddingBottom	Đây là bottom padding được điền cho Layout

2.1.1 | FrameLayout.

FrameLayout là một ViewGroup được sử dụng rất nhiều trong android. Bởi vì nó là ViewGroup đơn giản nhất, và thời gian tính toán của nó để layout ra các view con trong nó là thấp nhất nên performance của ViewGroup này là cao nhất.

FrameLayout được định nghĩa bắt đầu bởi thẻ `<FrameLayout>` và thẻ đóng `</FrameLayout>`. Ở giữa thẻ đóng và thẻ mở chính là các view con của nó.

```

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- View Child-->

</FrameLayout>

```

Quy tắc layout các view con trong FrameLayout là các view sẽ nằm chồng lên nhau, view thêm vào sau sẽ nằm đè lên view nằm phía dưới.

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:id="@+id/FrameLayout1"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Hello World"  
        android:textColor="#2c3e50"  
        android:textSize="32sp" />  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="TDC"  
        android:textColor="#16a085"  
        android:textSize="32sp" />  
/</FrameLayout>
```

Ví dụ 1. xây dựng giao diện như hình 2.1



Hình 2-1: Xây dựng giao diện bằng FrameLayout

```

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/FrameLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

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

        <TextView
            android:layout_gravity="bottom|center"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Trường Cao Đẳng Công Nghệ Thủ Đức"
            android:textColor="#16a085"
            android:textSize="18sp" />

    </FrameLayout>
</FrameLayout>

```

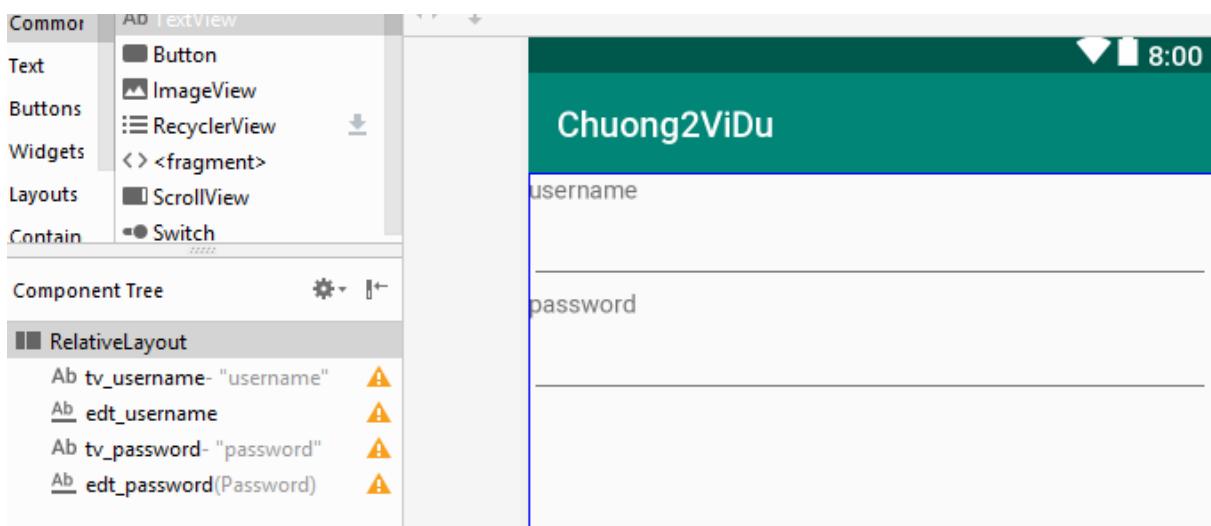
- ✓ Ưu điểm: Là ViewGroup đơn giản nên thời gian tính toán để layout các view con nhanh.
- ✓ Nhược điểm: Không thiết kế được cái giao diện phức tạp.

2.1.2 | **RelativeLayout**.

Các View khi được đặt lên Layout sẽ có vị trí phụ thuộc vào View đã đặt vào trước nó, do đó khi thay đổi vị trí của một View sẽ làm thay đổi vị trí của các View còn lại. Đối với các giao diện phức tạp hơn thì việc dùng Relative Layout sẽ dễ dàng hơn. Các View khi được đặt lên Layout sẽ có vị trí phụ thuộc vào vị trí các View đã đặt vào trước đó và đối tượng đang chứa nó.

- ❖ Ví dụ ở đây ta có thể xây dựng Layout cho TextView nằm phía trên ba Button theo vị trí tùy ý.
- ✓ Sử dụng trong các trường hợp xây dựng bộ cục tổ chức hiển thị các đối tượng theo mối quan hệ vị trí.
- ✓ Đối tượng được đặt vào ***RelativeLayout*** đầu tiên sẽ xác định vị trí cho các đối tượng sau đó.

Ví dụ 2. Xây dựng giao diện sử dụng RelativeLayout như hình 2.2



Hình 2-2: xây dựng giao diện sử dụng RelativeLayout

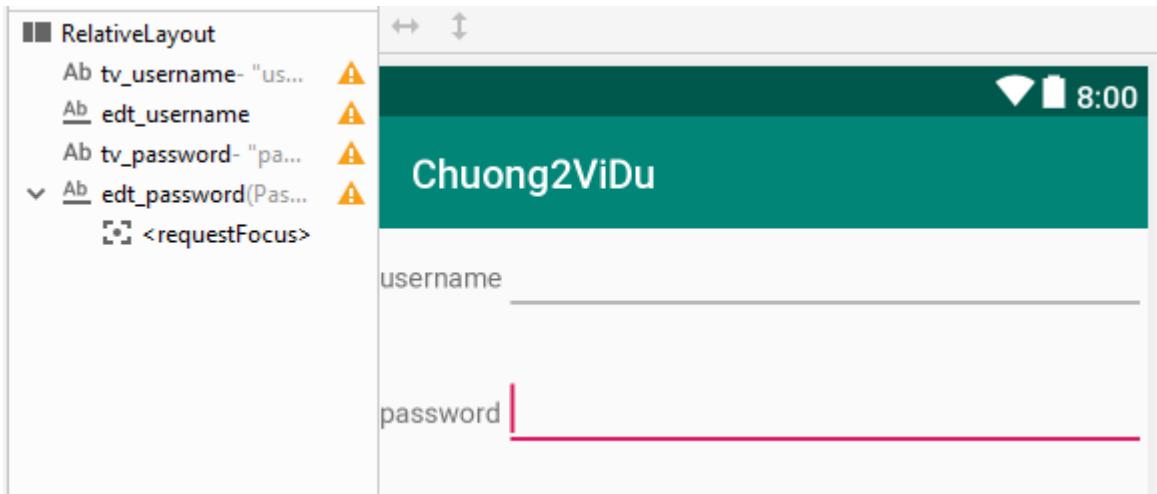
Ta thấy rằng EditText, username nằm dưới TextView, username. Và tương tự TextView, password nằm dưới Edittext, username, EditText password sẽ nằm dưới TextView, password.

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/tv_username"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="username" />
    <EditText
        android:id="@+id/edt_username"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/tv_username" />
    <TextView
        android:id="@+id/tv_password"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/edt_username"
        android:text="password" />
    <EditText
        android:id="@+id/edt_password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/tv_password"
        android:inputType="textPassword" />
</RelativeLayout>

```

Ví dụ 3. Xây dựng giao diện sử dụng RelativeLayout màn hình đăng nhập như hình 2.3



Hình 2-3: xây dựng giao diện sử dụng RelativeLayout màn hình đăng nhập

EditText username sẽ nằm bên phải TextView username EditText password sẽ nằm bên phải TextView password TextView password và EditText password sẽ nằm phía dưới TextView username

```
<?xml version='1.0' encoding='utf-8'?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/tv_username"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/edt_username"
        android:text="username" />
    <EditText
        android:id="@+id/edt_username"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@+id/tv_username" />
```

```

<TextView
    android:id="@+id/tv_password"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@id/edit_password"
    android:layout_below="@id/edit_username"
    android:text="password" />

<EditText
    android:id="@+id/edit_password"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/edit_username"
    android:layout_alignParentRight="true"
    android:layout_below="@+id/edit_username"
    android:layout_marginTop="22dp"
    android:ems="5"
    android:inputType="textPassword" >
</EditText>
</RelativeLayout>

```

- Ưu điểm: Thiết kế được các giao diện phức tạp.
- Nhược điểm: Muốn sử dụng các thuộc tính như `android:layout_above`, `android:layout_toLeftOf` thì phải đặt id cho các view mà view hiện tại xác định vị trí tương đối với các view đó.

2.1.3 | **LinearLayout.**

LinearLayout sắp xếp các view con theo hai hướng:

- ✓ **Vertical:** Sắp xếp view con theo chiều dọc.
- ✓ **Horizontal:** Sắp xếp các view con theo chiều ngang:

Sử dụng trong các trường hợp xây dựng bố cục tổ chức hiển thị các đối tượng theo một chiều duy nhất (chiều dọc hoặc ngang).

Đối tượng mặc định vị trí top left trên **LinearLayout**, có thể sử dụng thuộc tính Gravity để thiết lập lại vị trí.

☞ Chia tỉ lệ layout

Ngoài cách sử dụng **LinearLayout** thông thường thì **LinearLayout** thường được sử dụng để phân chia tỉ lệ layout bằng cách sử dụng thuộc tính **layout_weight** và **weightSum**.

☞ **weightSum**: Xác định trọng số của **LinearLayout** hiện tại.

☞ **layout_weight**: Trọng số mà view con trong **LinearLayout** chiếm giữ.

Ví dụ 4. Chia tỉ lệ có weightSum



Hình 2-4: Ví dụ chia tỷ lệ màn hình

```
<?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:weightSum="10"
    android:orientation="vertical">
    <View
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="5"
        android:background="#af5656" />
    <View
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="2"
        android:background="#28c6a6" />
    <View
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="#7c145f" /></LinearLayout>
```

❖ Ưu điểm

Thiết kế được các giao diện phức tạp.

Chia tỉ lệ layout, phù hợp với việc phát triển UI trên nhiều device có kích thước màn hình khác nhau.

❖ Nhược điểm

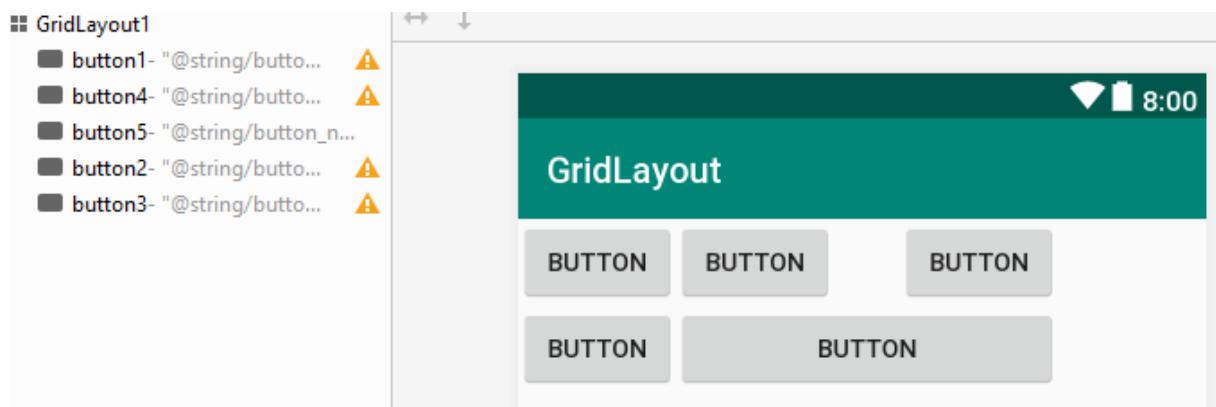
Thời gian tính toán và layout view con tốn chi phí hơn so với FrameLayout và RelativeLayout. Đây là ViewGroup tính toán phức tạp nhất trong bộ ba ViewGroup thường xuyên được sử dụng (FrameLayout, RelativeLayout, LinearLayout).

2.1.4 | GridLayout

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

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

Ví dụ 5. **Button** lên **GridLayout** có 3 dòng, 3 cột



```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:id="@+id/GridLayout1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:columnCount="3"  
    android:orientation="horizontal"  
    android:rowCount="3" >
```

```
<Button  
    android:id="@+id/button1"  
    android:layout_gravity="left/top"  
    android:text="@string/button_name" />
```

```
<Button  
    android:id="@+id/button4"  
    android:layout_column="0"  
    android:layout_gravity="left|top"  
    android:layout_row="2"  
    android:text="@string/button_name" />  
  
<Button  
    android:id="@+id/button5"  
    android:layout_width="213dp"  
    android:layout_column="1"  
    android:layout_columnSpan="2"  
    android:layout_gravity="fill_horizontal|fill_vertical"  
    android:layout_row="2"  
    android:text="@string/button_name" />  
  
<Button  
    android:id="@+id/button2"  
    android:layout_column="2"  
    android:layout_gravity="left|top"  
    android:layout_row="0"  
    android:text="@string/button_name" />  
  
<Button  
    android:id="@+id/button3"  
    android:layout_column="2"  
    android:layout_gravity="right|top"  
    android:layout_row="0"  
    android:text="@string/button_name" />  
  
</GridLayout>
```

2.2 | Các điều khiển cơ bản

Trong Android, để tạo giao diện, chúng ta phải sử dụng hai lớp cốt lõi là: View và ViewGroup. Chúng rất quan trọng và được sử dụng thường xuyên trong quá trình phát triển ứng dụng Android. View và ViewGroup là hai lớp Java kế thừa từ lớp Java Object. View là một lớp cha trên cùng. ViewGroup, TextView, AnalogClock, ImageView,... là các lớp con kế thừa từ lớp cha View. Và các lớp: Frame Layout, Linear layout và Relative Layout lại kế thừa từ ViewGroup. Tương tự, TextView cũng có 3 lớp con: EditText, Button và CheckedTextView.

View được sử dụng để tạo ra các điều khiển trên màn hình cho phép nhận các tương tác từ người dùng cũng như hiển thị các thông tin cần thiết, View bao gồm hai dạng:

- View: các điều khiển đơn lẻ.
- ViewGroup: tập hợp nhiều điều khiển đơn lẻ.

Sđt	UI Control & Miêu tả
1	TextView Control này được sử dụng để hiển thị text tới người dùng
2	EditText EditText là một lớp con được định nghĩa trước của TextView mà bao gồm các khả năng chỉnh sửa đa dạng
3	AutoCompleteTextView AutoCompleteTextView là một view tương tự như EditText, ngoại trừ rằng nó hiển thị một danh sách các đề nghị tự động trong khi người dùng soạn text
4	Button Một nút có thể được nhấn, hoặc click bởi người dùng để thực hiện một hành động

5	ImageButton Là một AbsoluteLayout cho khả năng xác định vị trí chính xác của các view con
6	CheckBox On/Off có thể được chuyển đổi bởi người dùng. nên sử dụng nó khi biểu diễn cho người dùng với một nhóm các tùy chọn có thể chọn mà không loại trừ lẫn nhau
7	ToggleButton Hiển thị trạng thái checked/unchecked giống một nút on/off với một light indicator
8	ProgressBar ProgressBar view cung cấp một phản hồi có thể nhìn thấy về một số tác vụ, như khi chúng ta thực hiện tác vụ ra ngoài trong background
9	TimePicker TimePicker view cho phép người sử dụng lựa chọn thời gian của một ngày: hoặc chế độ 24 h hoặc chế độ AM/PM
10	DatePicker DatePicker view cho phép người dùng lựa chọn một date

Các đối tượng View được thể hiện trên màn hình giao diện như một hình chữ nhật tuỳ thuộc vị trí, kích thước, màu sắc và nhận vào cũng như xử lý các tương tác có liên quan.

Một số thể hiện của lớp View: TextView, ImageView, SurfaceView... ViewGroup cũng là một thể hiện của View. Có thể xây dựng đối tượng View theo 2 cách:

- ☞ Kéo thả và tuỳ chỉnh thuộc tính trong XML.
- ☞ Thiết lập thông số và truy xuất trực tiếp trong Java Code.

2.2.1 | TextView

TextView dùng để hiển thị các đoạn văn bản mà không muốn người dùng có thể chỉnh sửa được nội dung, có thể khai báo **TextView** trong file layout **XML** hoặc trong đoạn code Java.

TextView là đối tượng cho phép hiển thị các nội dung văn bản ở 4 dạng:

- ☞ Normal: : dạng văn bản kích thước font chữ mặc định.
- ☞ SmallText: dạng văn bản kích thước font chữ nhỏ.
- ☞ MediumText: dạng văn bản kích thước font chữ vừa.
- ☞ LargeText: dạng văn bản kích thước font chữ to.



- ☞ Code TextView trong XML

```

<TextView
    android:id="@+id/txtTdc"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="tdc it" />
  
```

- ☞ Code TextView trong JAVA

```

TextView textView = (TextView) findViewById(R.id.txtTdc);
textView.setText("tdc it");
  
```

- ☞ Một số phương thức quan trọng khác:

- ***setTextColor***: android:textColor: thiết lập màu chữ.
- ***setTextSize***: android:textSize: thiết lập kích cỡ chữ.
- ***setTypeFace***: android:typeface: thiết lập các tuỳ chọn khác về font hay áp dụng một định dạng font ngoài cho TextView.

↳ Thuộc tính thường dùng của TextView

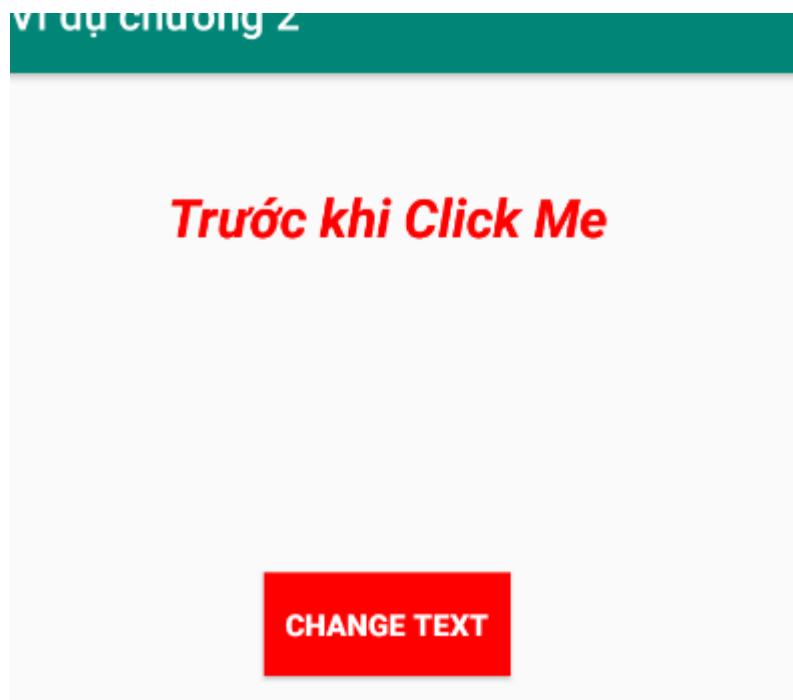
- ***android:id***: Là thuộc tính duy nhất của TextView.
- ***android:gravity***: Thuộc tính này thường sử dụng để canh nội dung trên TextView: left, right, center, top, bottom, center_vertical, center_horizontal
- ***android:text***: Thuộc tính này dùng xuất chuỗi văn bản lên TextView, Chúng ta có thể khai báo trong XML hoặc code Java
- ***android:textColor***: Thuộc tính này dùng xác định màu chữ, dạng màu chữ: "#argb", "#rgb", "#rrggbb", hoặc "#aarrggbb".
- ***android:textSize***: Thuộc tính textSize xác định kích thước văn bản của TextView. Chúng ta có thể đăt kích thước văn bản theo sp(scale independent pixel) hoặc dp(density pixel).
- ***android: textStyle***: Thuộc tính xác định loại văn bản của TextView, thông thường có các loại văn bản:bold, italic và normal. Nếu chúng ta muốn sử nhiều hơn một loại văn bản thì phải thêm phép toán hoặc "|" vào giữa các loại văn bản:
- ***android:background***: Thuộc tính này xác định màu nền cho TextView.
- ***android:padding***: Thuộc tính này xác định khoảng cách từ đường viền của TextView với nội dung nó chứa: left, right, top or bottom.

<***TextView***

```
    android:id="@+id/txtTDC"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:background="#000"
    android:gravity="center_horizontal"
    android:padding="10dp"
```

```
    android:text="TDC IT"  
    android:textColor="#fff"  
    android:textSize="40sp"  
    android:textStyle="bold/italic" />
```

Ví dụ 6. Tạo một **TextView** trong XML, sau đó thay đổi nội dung của nó thông qua một **button** được lập trình xử lý sự kiện trong Java Class



Hình 2-5: Ví dụ TextView trước khi click

Ví dụ chương 2

Sau khi click me

CHANGE TEXT

Hình 2-6: Ví dụ TextView Sau khi click

Bước 1: Tạo một project tên là TextView: File → New →Android Application Project điền các thông tin →Next →Finish

Bước 2: Mở res→layout→xml (hoặc) activity_main.xml và thêm code, chúng ta sẽ tạo một TextView và Button.

```
<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/simpleTextView"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_centerHorizontal="true"  
    android:text="Trước khi Click Me"  
    android:textColor="#f00"  
    android:textSize="25sp"
```

```
    android:textStyle="bold/italic"  
    android:layout_marginTop="50dp"/>
```

```
<Button  
    android:id="@+id/btnChangeText"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_centerInParent="true"  
    android:background="#f00"  
    android:padding="10dp"  
    android:text="Change Text"  
    android:textColor="#fff"  
    android:textStyle="bold" />  
</RelativeLayout>
```

Bước 3: Mở app → src → MainActivity.java và thêm code Nội dung của TextView sẽ thay đổi khi click vào Button.

```
public class MainActivity extends AppCompatActivity {  
    TextView txtmsg;  
    Button changeText;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        setControl();  
        setEvent();  
    }  
    private void setControl() {  
        txtmsg = findViewById(R.id.simpleTextView);  
        changeText = findViewById(R.id.btnChangeText);  
    }  
    private void setEvent() {
```

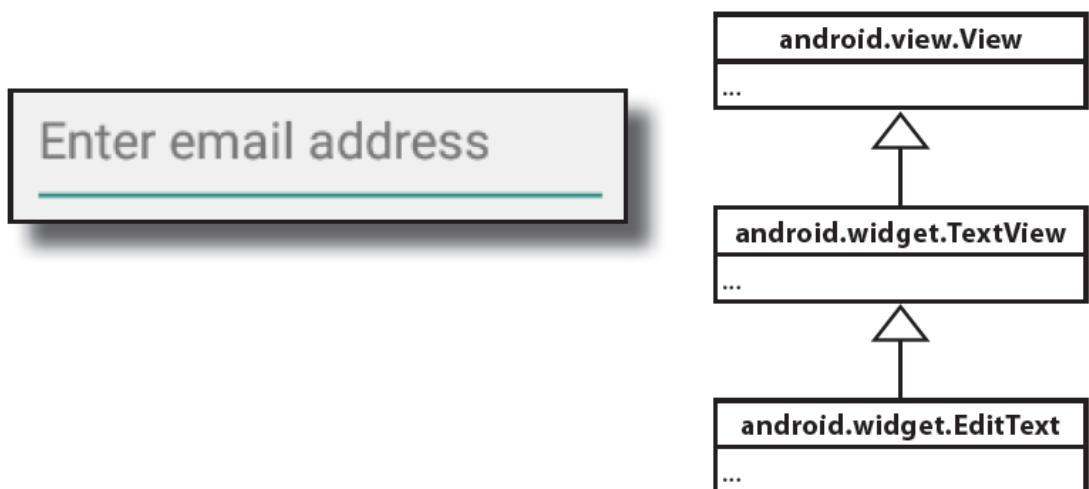
```

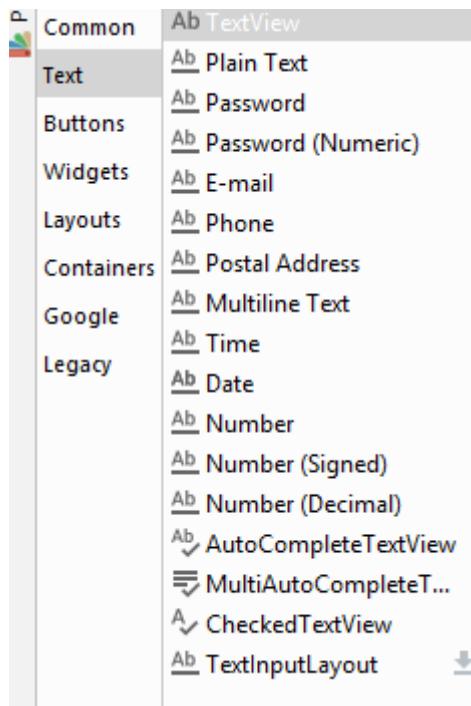
changeText.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        txtmsg.setText("Sau khi click me");
    }
});
}

```

2.2.2 | EditText

Trong lập trình ứng dụng một widget không thể thiếu đó là Text box, trong android gọi là **EditText**. **EditText** sử dụng cho phép người dùng nhập thông tin để ứng dụng xử lý dưới dạng các text box. **EditText** là lớp kế thừa từ **TextView**, được dùng thay đổi nội dung text, chứa tất cả thuộc tính của **TextView**.





Hình 2-7: Các dạng EditText

⇒ Thuộc tính thường dùng của EditText

- ☞ **android:id:** Là thuộc tính duy nhất của EditText.
- ☞ **android:gravity:** Thuộc tính này thường sử dụng để canh nội dung trên EditText: left, right, center, top, bottom, center_vertical, center_horizontal
- ☞ **android:text:** Thuộc tính này dùng xuất chuỗi văn bản lên EditText, Chúng ta có thể khai báo trong XML hoặc code Java
- ☞ **android:hint:** Thuộc tính hint để hiển thị thông tin gợi ý trong vùng nhập dữ liệu khi bạn chưa nhập bất kỳ dữ liệu nào vào, chỉ cần có dữ liệu là phần hint sẽ tự động mất đi
- ☞ **android:textColor:** Thuộc tính này dùng xác định màu chữ, dạng màu chữ: #argb”, “#rgb”, “#rrggbb”, hoặc #aarrggbb”.
- ☞ **android:textColorHint:** là thuộc tính set màu cho hint
- ☞ **android:textSize:** Thuộc tính textSize xác định kích thước văn bản của EditText. Chúng ta có thể đặt kích thước văn bản theo sp(scale independent pixel) hoặc dp(density pixel).

- ☞ **android:textStyle:** Thuộc tính xác định loại văn bản của EditText, thông thường có các loại văn bản:bold, italic và normal. Nếu chúng ta muốn sử nhiều hơn một loại văn bản thì phải thêm phép toán hoặc "|" vào giữa các loại văn bản
- ☞ **android:background:** Thuộc tính này xác định màu nền cho EditText.
- ☞ **android:padding:** Thuộc tính này xác định khoảng cách từ đường viền của EditText với nội dung nó chứa: left, right, top or bottom.
- ☞ **android:inputType:** Định dạng kiểu văn bản khi người dùng nhập vào(kiểu mật khẩu, kiểu số, kiểu email, phone, ...).Trong ví dụ sau EditText chỉ được nhập số. Chúng ta thêm thuộc tính android:inputType="number"
- ☞ **android:maxLines:** Cho phép người dùng nhập tối đa bao nhiêu dòng

<EditText

```

    android:id="@+id/simpleEditText"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:hint="Enter Your Name Here"
    android:textColorHint="#fff"
    android:textStyle="bold/italic"
    android:background="#000"
    android:ems="10"
    android:maxLines="1"
    android:inputType="number"
    android:padding="15dp"

```

/>

Ví dụ 7. Lấy giá trị từ các **EditText**, sau đó hiển thị các giá trị này lên **TextView** thông qua một **button**

Tài khoản _____

Mật khẩu _____

Ngày sinh _____

Email _____

ĐĂNG KÝ

Hình 2-8: Ví dụ EditText trước đăng ký

Tài khoản **tdc** _____

Mật khẩu ******* _____

Ngày sinh **15/01/1980** _____

Email **fit@tdc.edu.vn** _____

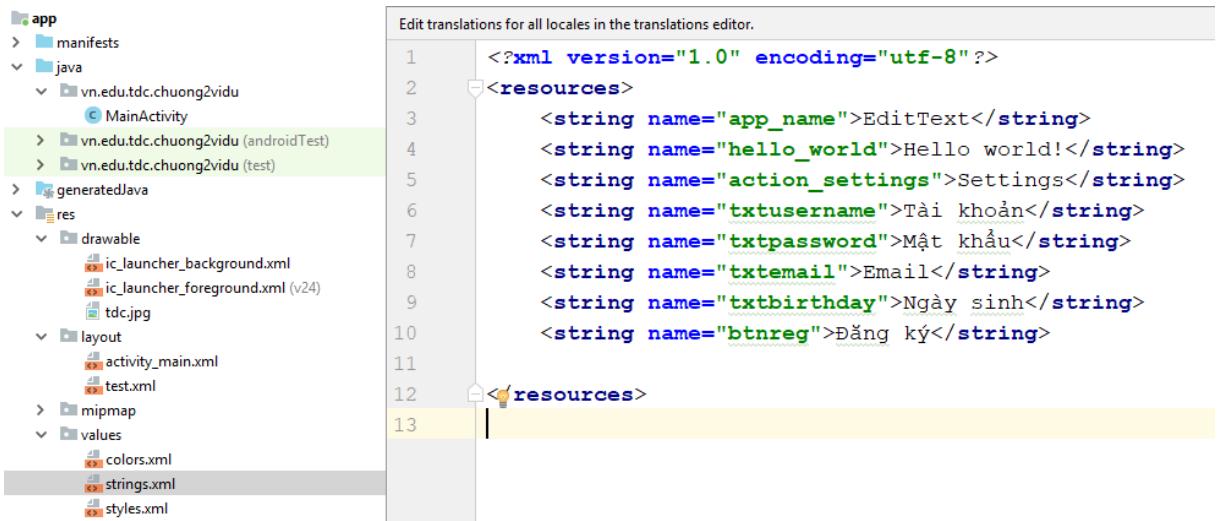
ĐĂNG KÝ

Thông tin tài khoản
Tài Khoản:tdc
Mật khẩu:123
Ngày sinh: 15/01/1980
Email:fit@tdc.edu.vn

Hình 2-9: Ví dụ EditText sau đăng ký

Bước 1: .Tiến hành tạo project,

Bước 2: Vào thư mục **res/values** bổ sung **string.xml**



Bước 3: vào thư mục res /layout → activity_main.xml. Thiết kế giao diện 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">
```

<LinearLayout

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
```

<TextView

```
    android:id="@+id/txtName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/txtusername" />
```

<EditText

```
    android:id="@+id/editName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:ems="10">

    <requestFocus />
</EditText>
</LinearLayout>

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

    <TextView
        android:id="@+id/txtPassword"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/txtpassword" />

    <EditText
        android:id="@+id/edtPassword"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPassword">

        <requestFocus />
</EditText>
</LinearLayout>

<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
        android:orientation="horizontal">

<TextView
    android:id="@+id/txtBirthday"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/txtbirthday" />

<EditText
    android:id="@+id/editBirthday"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:ems="10"
    android:inputType="date" />

</LinearLayout>

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

<TextView
    android:id="@+id/txtEmail"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/txtemail" />

<EditText
    android:id="@+id/editEmail"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```

    android:ems="10"
    android:inputType="textEmailAddress" />
</LinearLayout>

<Button
    android:id="@+id	btnReg"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/btnreg" />

<TextView
    android:id="@+id	txtShowMessage"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="0.21"
    android:inputType="textMultiLine"
    android:textAppearance="?android:attr/textAppearanceLarge" />

</LinearLayout>
```

Bước 4: : Mở app → src → **MainActivity.java** và thêm code. Khi click vào Button sẽ lấy các giá trị của **EditText**, sau đó hiển thị lên **TextView**.

```

public class MainActivity extends AppCompatActivity {
    //Khai báo các widget
    private Button btnReg;
    private TextView txtShowMessage;
    private EditText edtName, edtPassword, edtBirthday, edtEmail;

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

```

    setControl();
    setEvent();
}

private void setControl() {
    edtName = (EditText) findViewById(R.id.editName);
    edtPassword = (EditText) findViewById(R.id.editPassword);
    edtBirthday = (EditText) findViewById(R.id.editBirthday);
    edtEmail = (EditText) findViewById(R.id.editEmail);
    txtShowMessage = (TextView) findViewById(R.id.txtShowMessage);
    btnReg = (Button) findViewById(R.id.btnReg);
}

private void setEvent() {
    btnReg.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String str = "Thông tin tài khoản \n";
            str += "Tài Khoản:" + edtName.getText().toString() + "\n Mật khẩu:" +
                edtPassword.getText().toString();
            str += "\n Ngày sinh: " + edtBirthday.getText().toString() + "\n Email:" +
                edtEmail.getText().toString();
            txtShowMessage.setText(str);
            txtShowMessage.setBackgroundColor(Color.GREEN);
        }
    });
}

```

2.2.3 | Button

Button là một loại View, nó hiển thị nút bấm để chờ người dùng bấm vào. Button kế thừa từ TextView nên các thuộc tính, thiết lập cho TextView ở phần trước là có hiệu quả như đối với Button.



Hình 2-10: Các dạng Button

↳ Bắt sự kiện khi người dùng ấn vào Buton: có 3 cách chính như sau:

❖ Cách 1:

```
view.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        Log.i(tag: "view", msg: "Click");  
    }  
});
```

❖ Cách 2: Cho Activity implement OnClickListener của android.view.View

```

public class MainActivity extends ActionBarActivity implements OnClickListener {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btnHello = (Button) findViewById(R.id.btnHello);
        btnHello.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
        case R.id.btnHello:
            //do something
            break;
        default:
            break;
        }
    }
}

```

☞ Cách 3: Thêm thuộc tính onClick vào thẻ Button:

```

<Button
        android:id="@+id	btnHello"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="doHello"
        android:text="Hello" />

```

☞ Viết phương thức public void doHello() trong MainActivity

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
public void doHello(View v){
    switch (v.getId()) {
    case R.id.btnHello:
        //do something
        break;

    default:
        break;
    }
}

```

- ↳ Thuộc tính thường dùng của Button
- ↳ **android:id:** Là thuộc tính duy nhất của Button.
- ↳ **android:gravity:** Thuộc tính này thường sử dụng để canh nội dung trên EditText: left, right, center, top, bottom, center_vertical, center_horizontal.
- ↳ **android:text:** Thuộc tính này dùng xuất chuỗi văn bản lên Button, Chúng ta có thể khai báo trong XML hoặc code Java.
- ↳ **android:textColor:** Thuộc tính này dùng xác định màu chữ, dạng màu chữ: "#argb", "#rgb", "#rrggb", hoặc "#aarrggbb".
- ↳ **android:textSize:** Thuộc tính textSize xác định kích thước văn bản của Button. Chúng ta có thể đặt kích thước văn bản theo: sp(scale independent pixel) hoặc dp(density pixel).
- ↳ **android:textSize:** Thuộc tính textSize xác định kích thước văn bản của Button. Chúng ta có thể đặt kích thước văn bản theo: sp(scale independent pixel) hoặc dp(density pixel).
- ↳ **android:background:** Thuộc tính này xác định màu nền cho Button.
- ↳ **android:padding:** Thuộc tính này xác định khoảng cách từ đường viền của Button với nội dung nó chứa: left, right, top or bottom.
- ↳ **android:background:** Thuộc tính này xác định màu nền cho Button.
- ↳ **android:padding:** Thuộc tính này xác định khoảng cách từ đường viền của Button với nội dung nó chứa: left, right, top or bottom.
- ↳ **android:drawableBottom:** drawableBottom hiển thị icon sau chuỗi văn bản: **android:drawableTop, android:drawableRight và android:drawableLeft:** Hiển thị Icon bên trái,bên phải hoặc phía trên của chuỗi văn bản.

Ví dụ 8. hiển thị Icon bên dưới chuỗi văn bản:

<Button

```

    android:id="@+id(btnSimple")
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"

```

```
    android:layout_alignParentTop="true"  
    android:layout_marginTop="28dp"  
    android:textSize="40sp"  
    android:text="@string/btnreg"  
    android:padding="15dp"  
    android:textStyle="bold/italic"  
    android:background="#FFCC"  
    android:drawableBottom="@drawable/tdc"/>
```



Hình 2-11: Hiển thị Icon bên dưới chuỗi văn bản

Ví dụ 9. Xây dựng ứng dụng tính tổng 2 số nguyên, kết quả hiển thị các giá trị này lên **TextView** thông qua một **button** được lập trình xử lý sự kiện trong Java Class

Tổng hai số

Nhập số a: _____

Nhập số b: _____

TỔNG **XÓA TRẮNG**

Hình 2-12: Ví dụ Button trước khi tính

Tổng hai số

Nhập số a: 3 _____

Nhập số b: 4 _____

Tổng 2 số là: 7.0

TỔNG **XÓA TRẮNG**

Hình 2-13: Ví dụ Button sau khi tính

Bước 1: Tạo một project tên là TextView: File → New →Android Application Project
điền các thông tin →Next→Finish

Bước 2: Vào thư mục res/values bô sung string.xml

<resources>

```

<string name="app_name">Button</string>
<string name="txtlabel">Tổng hai số</string>
<string name="txtnuma">Nhập số a:</string>
<string name="txtnumb">Nhập số b:</string>
<string name="btnsum">Tổng</string>

```

```
<string name="btnclear">Xóa trống</string>
</resources>
```

Bước 3: Mở res → layout → xml (hoặc) activity_main.xml và thêm code, chúng ta sẽ tạo các TextView, EditText và Button

```
<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"
    >

    <TextView
        android:id="@+id/txtLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="@string/txtlabel"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textColor="#ff00"
        android:textSize="30sp"
        android:textStyle="bold/italic" />

    <TextView
        android:id="@+id/txtNumA"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_marginTop="56dp"
        android:text="@string/txtnuma" />

    <EditText
```

```
    android:id="@+id/edtNumA"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/txtNumA"
    android:layout_alignLeft="@+id/txtLabel"
    android:layout_alignBottom="@+id/txtNumA"
    android:ems="10"
    android:inputType="number">

    <requestFocus />
</EditText>

<TextView
    android:id="@+id/txtNumB"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/edtNumA"
    android:layout_alignLeft="@+id/txtNumA"
    android:layout_marginTop="35dp"
    android:text="@string/txtnumb" />

<EditText
    android:id="@+id/edtNumB"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/txtNumB"
    android:layout_alignLeft="@+id/edtNumA"
    android:layout_alignBottom="@+id/txtNumB"
    android:ems="10"
    android:inputType="number" />
```

```

<TextView
    android:id="@+id/txtResult"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/editNumB"
    android:layout_alignLeft="@+id/txtNumB"
    android:layout_marginTop="25dp" />

<Button
    android:id="@+id	btnClear"
    android:layout_width="130sp"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id	btnSum"
    android:layout_alignBottom="@+id	btnSum"
    android:layout_marginLeft="30dp"
    android:layout_toRightOf="@+id	btnSum"
    android:text="@string/btnClear" />

<Button
    android:id="@+id	btnSum"
    android:layout_width="130sp"
    android:layout_height="wrap_content"
    android:layout_below="@+id/txtResult"
    android:layout_alignTop="@+id/txtResult"
    android:layout_marginTop="28dp"
    android:text="@string/btnSum" />

</RelativeLayout>

```

Bước 4: Mở **app → src→MainActivity.java** và thêm code. Khi click vào **Button** gọi phương thức **sum(a,b)**, trong phương thức này chúng ta truyền 2 tham số lấy từ 2 **EditText**, sau đó hiển thị lên **TextView**.

```

public class MainActivity extends AppCompatActivity {
    //Khai báo các widget
    Button btnSum, btnClear;
    EditText edtNumA, edtNumB;
    TextView txtResult;

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

    private void setControl() {
        edtNumA = (EditText)findViewById(R.id.edtNumA);
        edtNumB = (EditText)findViewById(R.id.edtNumB);
        txtResult= (TextView)findViewById(R.id.txtResult);
        btnSum = (Button)findViewById(R.id.btnSum);
        btnClear= (Button)findViewById(R.id.btnClear);
    }

    private void setEvent() {
        btnSum.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                float a, b, c;
                a = Float.parseFloat(edtNumA.getText().toString());
                b = Float.parseFloat(edtNumB.getText().toString());
            }
        });
    }
}

```

```

c = a + b;
txtResult.setText("Tổng 2 số là: "+c);
txtResult.setBackgroundColor(Color.GREEN);

}

});

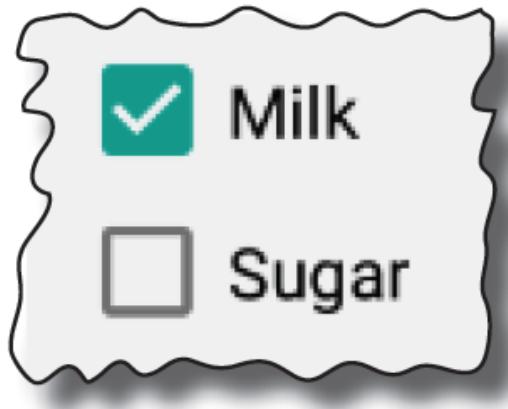
btnClear.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        edtNumA.setText("");
        edtNumB.setText("");
    }
});
}

```

2.2.4 | Checkbox.

CheckBox là thành phần thẻ hiện trạng thái chọn (checked) hoặc không chọn (unchecked) **CheckBox** thường dùng khi người dùng có nhiều lựa chọn và được phép chọn một hoặc nhiều lựa chọn cùng lúc.



Hình 2-14: Các dạng CheckBok

Các phương của CheckBox: Lớp CheckBox kế thừa nhiều phương thức của View, TextView hoặc Button

Một vài phương thức thường sử dụng:

Phương thức	Ý nghĩa
public boolean isChecked()	True nếu CheckBox là checked, ngược lại false
public void setChecked(boolean status)	Thay đổi trạng thái của CheckBox

↳ Thuộc tính thường dùng của CheckBox

- ☞ **android:id:** Là thuộc tính duy nhất của CheckBox
- ☞ **android:checked:** checked là thuộc tính của CheckBox dùng để set trạng thái của CheckBox. Giá trị là true hoặc false, nếu giá trị là true thì trạng thái CheckBox là checked, ngược lại là false thì trạng thái của CheckBox là unchecked. Chúng ta cũng có thể set trạng thái của CheckBox bên Java Code bằng cách dùng phương thức **setChecked(boolean status)**
- ☞ **android:gravity:** Thuộc tính này thường sử dụng để canh nội dung trong CheckBox: left, right, center, top, bottom, center_vertical, center_horizontal.
- ☞ **android:text:** thuộc tính text dùng hiển thị nội dung trong một CheckBox. Chúng ta có thể set thuộc tính này trong tập tin xml hoặc java code

- ☞ **android:textSize:** Thuộc tính textSize xác định kích thước nội dung văn bản của CheckBox. Chúng ta có thể đặt kích thước văn bản theo: sp (scale independent pixel) hoặc dp(density pixel).
- ☞ **android:textStyle:** Thuộc tính xác định loại văn bản của CheckBox, thông thường có các loại văn bản:bold, italic và normal. Nếu chúng ta muốn sử dụng nhiều hơn một loại văn bản thì phải thêm phép toán hoặc "|" vào giữa các loại văn bản
- ☞ **android:background:** Thuộc tính này xác định màu nền cho CheckBox.
- ☞ **android:padding:** Thuộc tính này xác định khoảng cách từ đường viền của CheckBox với nội dung nó chứa:left, right, top or bottom.

<CheckBox

```
    android:id="@+id/chksimpleCheckBox"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Attribute Of Check Box"  
    android:textColor="#44f"  
    android:textSize="20sp"  
    android:textStyle="bold/italic"  
    android:checked="true"  
    android:padding="30dp"/>
```

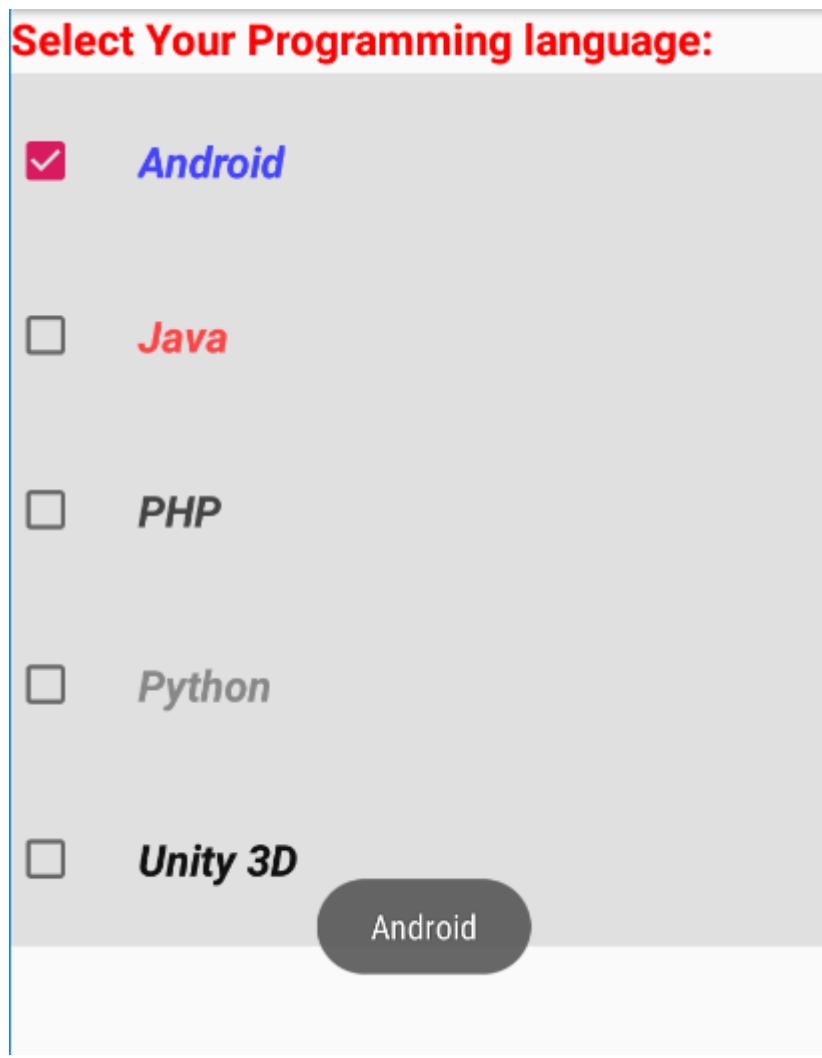


Ví dụ 10. Chọn 1 ngôn ngữ lập trình trên các đối tượng **CheckBox**. Khi người sử dụng click lên **CheckBox** sẽ hiển thị ngôn ngữ vừa chọn, thông qua việc sử dụng đối tượng **TOAST**.

Select Your Programming language:

- Android*
- Java*
- PHP*
- Python*
- Unity 3D***

Hình 2-15: Ví dụ CheckBox trước khi chọn



Hình 2-16: Ví dụ CheckBox sau khi chọn

Bước 1: Tạo một project tên là DemoCheckBox: **File** → **New** → **Android Application Project** điền các thông tin → **Next** → **Finish**.

Bước 2: Vào thư mục res/values bổ sung string.xml

```
<resources>
    <string name="app_name">CheckBoxExample</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="android">Android</string>
    <string name="java">Java</string>
    <string name="php">PHP</string>
    <string name="python" >Python</string>
    <string name="unity">Unity 3D</string>
</resources>
```

Bước 3: Mở res → layout → xml (hoặc) activity_main.xml

```
<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">
```

```
<TextView
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:text="Select Your Programming language: "
```

```
    android:textColor="#f00"
```

```
    android:textSize="20sp"
```

```
    android:textStyle="bold" />
```

```
<LinearLayout
```

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

```
    android:layout_width="fill_parent"
```

```
    android:layout_height="wrap_content"
```

```
    android:layout_marginTop="30dp"
```

```
    android:background="#e0e0e0"
```

```
    android:orientation="vertical">
```

```
<CheckBox
```

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

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:layout_centerHorizontal="true"
```

```
    android:checked="false"
```

```
    android:padding="27dp"
```

```
    android:text="@string/android"
```

```
    android:textColor="#44f"
```

```
    android:textSize="20sp"
    android:textStyle="bold/italic" />

<CheckBox
    android:id="@+id/chkJava"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:checked="false"
    android:padding="27dp"
    android:text="@string/java"
    android:textColor="#f44"
    android:textSize="20sp"
    android:textStyle="bold/italic" />

<CheckBox
    android:id="@+id/chkPHP"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:checked="false"
    android:padding="27dp"
    android:text="@string/php"
    android:textColor="#444"
    android:textSize="20sp"
    android:textStyle="bold/italic" />

<CheckBox
    android:id="@+id/chkPython"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```

    android:layout_centerHorizontal="true"
    android:checked="false"
    android:padding="27dp"
    android:text="@string/python"
    android:textColor="#888"
    android:textSize="20sp"
    android:textStyle="bold/italic" />

<CheckBox
    android:id="@+id/chkUnity"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:checked="false"
    android:padding="27dp"
    android:text="@string/unity"
    android:textColor="#101010"
    android:textSize="20sp"
    android:textStyle="bold/italic" />

```

</LinearLayout>

</RelativeLayout>

Bước 4: Mở app → src -> **MainActivity.java** và thêm code. Khi click vào **CheckBox** sẽ lấy các giá trị của **CheckBox**, sau dùng đối tượng **TOAST** để hiển thị

```

public class MainActivity extends AppCompatActivity implements
View.OnClickListener {

    //Khai báo các widget
    CheckBox android, java, python, php, unity3D;

    @Override

```

```

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

private void setControl() {
    android = (CheckBox) findViewById(R.id.chkAndroid);
    java = (CheckBox) findViewById(R.id.chkJava);
    python = (CheckBox) findViewById(R.id.chkPython);
    php = (CheckBox) findViewById(R.id.chkPHP);
    unity3D = (CheckBox) findViewById(R.id.chkUnity);
}

private void setEvent() {
    android.setOnClickListener(this);
    java.setOnClickListener(this);
    python.setOnClickListener(this);
    php.setOnClickListener(this);
    unity3D.setOnClickListener(this);
}

@Override
public void onClick(View view) {

    switch (view.getId()) {
        case R.id.chkAndroid:
            if (android.isChecked())
                Toast.makeText(getApplicationContext(), "Android",
                Toast.LENGTH_LONG).show();
    }
}

```

```

        break;

    case R.id.chkJava:
        if (java.isChecked())
            Toast.makeText(getApplicationContext(), "Java",
Toast.LENGTH_LONG).show();
        break;

    case R.id.chkPHP:
        if (php.isChecked())
            Toast.makeText(getApplicationContext(), "PHP",
Toast.LENGTH_LONG).show();
        break;

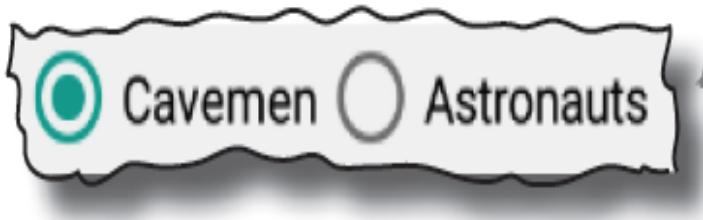
    case R.id.chkPython:
        if (python.isChecked())
            Toast.makeText(getApplicationContext(), "Python",
Toast.LENGTH_LONG).show();
        break;

    case R.id.chkUnity:
        if (unity3D.isChecked())
            Toast.makeText(getApplicationContext(), "Unity 3D",
Toast.LENGTH_LONG).show();
        break;
    }
}
}
}

```

2.2.5 | RadioButton

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



Hình 2-17: Các dạng RadioButton

RadioButton là một Button gồm có 2 trạng thái checked và unchecked, nếu một button đang ở trạng thái unchecked thì người sử dụng có thể check vào, còn button đang ở trạng thái checked thì người sử dụng không thể unchecked.

Một vài phương thức thường sử dụng:

Phương thức	Ý nghĩa
public boolean isChecked()	True nếu CheckBox là checked, ngược lại false
public void setChecked(boolean status)	Thay đổi trạng thái của CheckBox

- ↳ Thuộc tính thường dùng của RadioButton
 - ☞ **android:id:** Là thuộc tính duy nhất của RadioButton
 - ☞ **android:checked:** checked là thuộc tính của RadioButton dùng để set trạng thái của CheckBox. Giá trị là true hoặc false, nếu giá trị là true thì trạng thái RadioButton là checked, ngược lại là false thì trạng thái của RadioButton là unchecked. Chúng ta cũng có thể set trạng thái của RadioButton bên Java Code bằng cách dùng phương thức setChecked(boolean status).
 - ☞ **android:checked:** checked là thuộc tính của RadioButton dùng để set trạng thái của CheckBox. Giá trị là true hoặc false, nếu giá trị là true thì trạng thái RadioButton là checked, ngược lại là false thì trạng thái của RadioButton là unchecked. Chúng ta cũng có thể set trạng thái của RadioButton bên Java Code bằng cách dùng phương thức setChecked(boolean status).
 - ☞ **android:gravity:** Thuộc tính này thường sử dụng để canh nội dung trong RadioButton: left, right, center, top, bottom, center_vertical, center_horizontal.

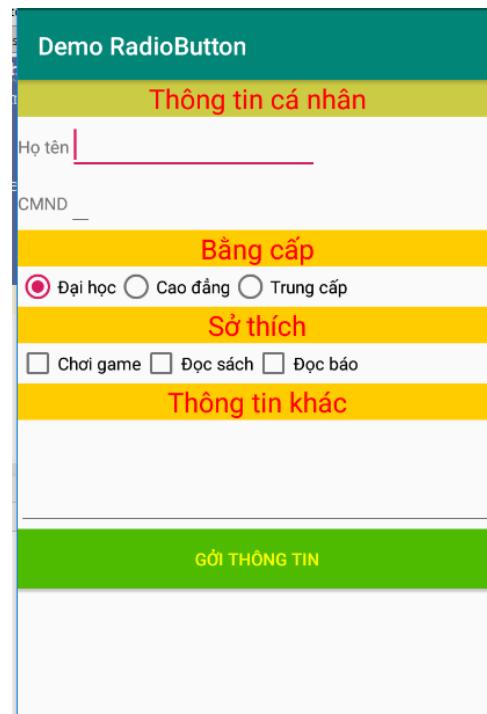
- ☞ **android:text:** thuộc tính text dùng hiển thị nội dung trong một RadioButton.
Chúng ta có thể set thuộc tính này trong tập tin xml hoặc java code
- ☞ **android:textColor:** Thuộc tính này dùng xác định màu chữ, dạng màu chữ: "#argb", "#rgb", "#rrggb", hoặc "#aarrggbb".
- ☞ **android:textSize:** Thuộc tính textSize xác định kích thước nội dung văn bản của RadioButton. Chúng ta có thể đặt kích thước văn bản theo: sp (scale independent pixel) hoặc dp (density pixel).
- ☞ **android:textStyle:** Thuộc tính xác định loại văn bản của RadioButton, thông thường có các loại văn bản:bold, italic và normal. Nếu chúng ta muốn sử nhiều hơn một loại văn bản thì phải thêm phép toán hoặc "|" vào giữa các loại văn bản.
- ☞ **android:background:** Thuộc tính này xác định màu nền cho RadioButton.
- ☞ **android:padding:** Thuộc tính này xác định khoảng cách từ đường viền của RadioButton với nội dung nó chứa: left, right, top or bottom.

<**RadioButton**

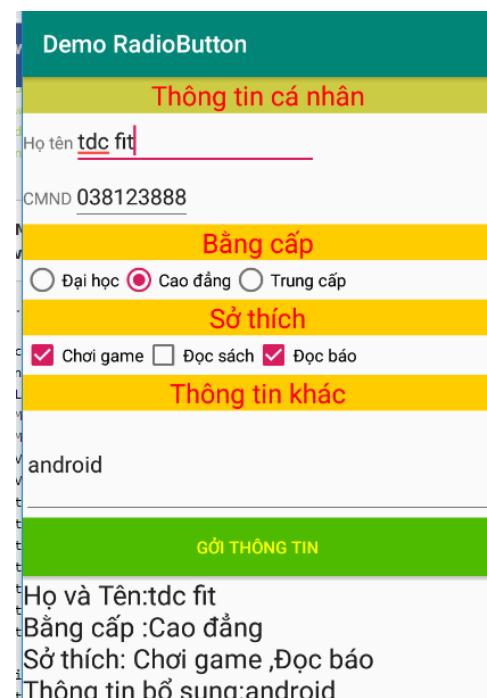
```
    android:id="@+id/rdisimpleRadioButton"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:text ="TDC IT"  
    android:textSize="40sp"  
    android:textColor="#f00"  
    android:textStyle="bold/italic"  
    android:checked="true"  
    android:padding="30dp"/>
```



Ví dụ 11. Xây dựng ứng dụng khảo sát đơn giản sau:



Hình 2-18: Ví dụ RadioButton trước khi gửi thông tin



Hình 2-19: Ví dụ RadioButton sau khi gửi thông tin

Yêu cầu:

- ☞ Tên người không được để trống và phải có ít nhất 3 ký tự
- ☞ Chứng minh nhân dân chỉ được nhập kiểu số và phải có đúng 9 chữ số

- ☞ Bằng cấp mặc định sẽ chọn là Đại học
- ☞ Sở thích phải chọn ít nhất 1 chọn lựa
- ☞ Thông tin bổ sung có thể để trống
- ☞ Khi bấm gửi thông tin, chương trình sẽ hiển thị toàn bộ thông tin cá nhân cho người sử dụng biết thông qua TextView:

Bước 1: Tạo một project tên là DemoRadioButton: File → New →Android Application Project điền các thông tin → Next →Finish

Bước 2: Vào thư mục res/values bổ sung string.xml

<resources>

```

<string name="app_name">Demo RadioButton</string>
<string name="action_settings">Settings</string>
<string name="information">Thông tin cá nhân</string>
<string name="firstname">Họ tên</string>
<string name="id">CMND</string>
<string name="degree">Bằng cấp</string>
<string name="university">Đại học</string>
<string name="college">Cao đẳng</string>
<string name="college1">Trung cấp</string>
<string name="favorate">Sở thích</string>
<string name="game">Chơi game</string>
<string name="book">Đọc sách</string>
<string name="newspaper">Đọc báo</string>
<string name="addinformation">Thông tin khác</string>
<string name="btninsert">Gửi thông tin</string>

```

</resources>

Bước 3: Mở res→layout→xml

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="fill_parent"

```

```
    android:layout_height="fill_parent"
    android:orientation="vertical">

<TextView
    android:id="@+id/txtInfomation"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="#cc4"
    android:gravity="center"
    android:text="@string/information"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:textColor="#f00" />

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

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

<TextView
    android:id="@+id/txtFirstName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/firstname" />

<EditText
    android:id="@+id/edtFirstName"
```

```
    android:layout_width="200sp"
    android:layout_height="wrap_content">

    <requestFocus />
</EditText>
</LinearLayout>

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/txtID"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/id" />

    <EditText
        android:id="@+id/edtID"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:maxLength="9" />
</LinearLayout>
</LinearLayout>

<TextView
    android:id="@+id/txtDegree"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
```

```
    android:background="#fc0"
    android:gravity="center"
    android:text="@string/degree"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:textColor="#f00" />
```

```
<RadioGroup
    android:id="@+id/rdiGroupDegree"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
```

```
<RadioButton
    android:id="@+id/rdiUniversity"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:text="@string/university" />
```

```
<RadioButton
    android:id="@+id/rdiCollege"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/college" />
```

```
<RadioButton
    android:id="@+id/rdiCollege1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/college1" />
</RadioGroup>
```

```
<TextView  
    android:id="@+id/txtFavorate"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:background="#fc0"  
    android:gravity="center"  
    android:text="@string/favorate"  
    android:textAppearance="?android:attr/textAppearanceLarge"  
    android:textColor="#f00" />
```

```
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:orientation="horizontal">
```

```
<CheckBox  
    android:id="@+id/chkGame"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/game" />
```

```
<CheckBox  
    android:id="@+id/chkBook"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/book" />
```

```
<CheckBox  
    android:id="@+id/chkNewspaper"  
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"  
    android:text="@string/newspaper" />  
  
</LinearLayout>
```

```
<TextView  
    android:id="@+id/txtAddInformaton"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:background="#fc0"  
    android:gravity="center"  
    android:text="@string/addinformation"  
    android:textAppearance="?android:attr/textAppearanceLarge"  
    android:textColor="#f00" />
```

```
<EditText  
    android:id="@+id/edtAddInformation"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:ems="10"  
    android:inputType="textMultiLine"  
    android:lines="3" />
```

```
<Button  
    android:id="@+id/btninsert"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:background="#4ebb00"  
    android:text="@string/btninsert"  
    android:textColor="#ff0" />
```

```

<TextView
    android:id="@+id/txtResult"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:inputType="textMultiLine"
    android:lines="6"
    android:textAppearance="?android:attr/textAppearanceLarge" />

```

</LinearLayout>

Bước 4: Mở app →src →MainActivity.java và thêm code. Khi click vào Button Gõi thông tin, các thông tin được hiển thị qua TextView.

```
public class MainActivity extends AppCompatActivity {
```

```

    //Khai báo các widget
    EditText edtFirstName, edtID, edtInformation;
    CheckBox chkGame, chkBook, chkNewspaper;
    TextView txtResult;
    Button btnInsert;

```

@Override

```

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

```

```
private void setControl() {
```

```

    edtFirstName = (EditText) findViewById(R.id.edtFirstName);
    edtID      = (EditText) findViewById(R.id.edtID);
    edtInformation = (EditText) findViewById(R.id.edtAddInformation);
    chkGame      = (CheckBox) findViewById(R.id.chkGame);

```

```

chkBook      = (CheckBox) findViewById(R.id.chkBook);
chkNewspaper = (CheckBox) findViewById(R.id.chkNewspaper);
btnInsert    = (Button) findViewById(R.id.btninsert);
txtResult   = (TextView) findViewById(R.id.txtResult);
}

private void setEvent() {
    btnInsert.setOnClickListener(new View.OnClickListener() {

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

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

```

```

String strID = edtID.getText().toString().trim();
if(strID.length()!=9){
    edtID.selectAll();
    edtID.requestFocus();
    Toast.makeText(getApplicationContext(), "Số chứng minh nhân dân phải 9 số",
    Toast.LENGTH_LONG).show();
    return;
}
String degree;
RadioGroup rdigroupDegree = (RadioGroup)
findViewById(R.id.rdiGroupDegree);
int id = rdigroupDegree.getCheckedRadioButtonId();

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

if(favorate.trim().length()==0){
    Toast.makeText(getApplicationContext(), "Xin vui lòng nhập thông tin",
    Toast.LENGTH_LONG).show();
}

```

```

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

```

2.2.6 | Image

Hầu hết tất cả các ứng dụng, web, game trên desktop, web, mobile đều sử dụng hình ảnh. Và trong Android để hiển thị hình ảnh chúng ta sử dụng view được xây dựng sẵn trong Android SDK có tên là **ImageView**. ImageView là một view có chức năng loading, và hiển thị hình ảnh, hoặc bắt cứ drawable. ImageView handle cho chúng ta tất cả các công việc để hiển thị hình ảnh.

ImageView là một view sử dụng để hiển thị ảnh, mà nguồn ảnh có thể là một file ảnh trên ứng dụng, trên thiết bị hoặc từ **URL**.



Hình 2-20: Ví dụ Image

- ↳ Thuộc tính thường dùng của ImageView
- ↳ **android:id:** Là thuộc tính duy nhất của ImageView
- ↳ **android:src:** là thuộc tính chứa hình ảnh cần hiển thị
- ↳ **android:background:** Thuộc tính này xác định màu nền cho ImageView
- ↳ **android:padding:** Thuộc tính này xác định khoảng cách từ đường viền của ImageView với nội dung nó chứa: left, right, top or bottom.
- ↳ **paddingRight:** thiết khoảng cách bên phải của ImageView.
- ↳ **paddingLeft:** thiết khoảng cách bên trái của ImageView.
- ↳ **paddingTop:** thiết khoảng cách phía trên của ImageView.
- ↳ **paddingBottom:** thiết khoảng cách phía bên dưới của ImageView.
- ↳ **padding:** thiết khoảng cách tất cả 4 phía của ImageView.
- ↳ **android:scaleType:** ScaleType là thuộc tính xác định các thức mà hình ảnh sẽ được scale như thế nào để phù hợp với view của chúng ta. ImageView có thể hiển thị image theo nhiều cách khác nhau phụ thuộc vào các giá trị của thuộc tính scaleType. Giá trị của scaleType :fit_xy, center_crop, fitStart v.v .

<ImageView

android:id="@+id/simpleImageView"

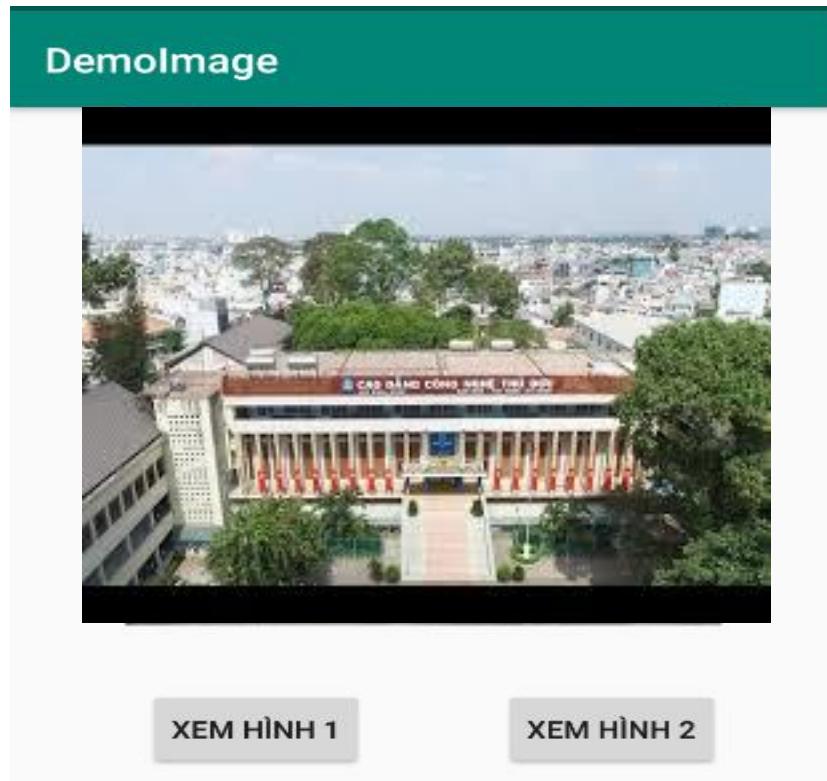
```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#000"
    android:src="@drawable/tdc"
    android:padding="30dp"
    android:scaleType="fitXY"
/>

```



Hình 2-21: ví dụ minh họa ImageView

Ví dụ 12. xây dựng ứng dụng hiển thị hình ảnh lên một ImageView. Để xem hình người sử dụng click vào 2 Button trên màn hình



Hình 2-22: Ví dụ ImageView xem hình 1

Bước 1: Tạo một project tên là DemoImages: **File** → **New** → **Android Application Project** điền các thông tin → **Next** → **Finish**.

Bước 2: Vào thư mục res/values bô sung string.xml

```
<resources>
    <string name="app_name">DemoImage</string>
    <string name="btndemo1">Xem hình 1</string>
    <string name="btndemo2">Xem hình 2</string>
</resources>
```

Bước 3: Mở res → layout → xml (hoặc) activity_main.xml

```
<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">
```

```
<ImageView
    android:id="@+id/imgDemo"
```

```
    android:layout_width="wrap_content"
    android:layout_height="300sp"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:scaleType="fitXY"
    android:src="@drawable/tdc" /><!--set scaleType fit XY-->
```

```
<Button
    android:id="@+id/btnDemo1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/imgDemo"
    android:layout_alignParentLeft="true"
    android:layout_marginLeft="63dp"
    android:layout_marginTop="36dp"
    android:text="@string/btndemo1" />

<Button
    android:id="@+id/btnDemo2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/btnDemo1"
    android:layout_alignRight="@+id/imgDemo"
    android:layout_alignBottom="@+id/btnDemo1"
    android:text="@string/btndemo2" />
```

```
</RelativeLayout>
```

Bước 4: Mở app → src -> **MainActivity.java** và thêm code. Khi click vào **CheckBox** sẽ lấy các giá trị của **CheckBox**, sau dùng đối tượng **TOAST** để hiển thị

```

public class MainActivity extends AppCompatActivity {

    //Khai báo các widget
    ImageView imgDemo;
    Button btnDemo1, btnDemo2;

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

    private void setControl() {
        imgDemo = (ImageView) findViewById(R.id.imgDemo);
        btnDemo1= (Button) findViewById(R.id.btnDemo1);
        btnDemo2= (Button)findViewById(R.id.btnDemo2);
    }

    private void setEvent() {
        btnDemo1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                imgDemo.setImageResource(R.drawable.tdc);
            }
        });

        btnDemo2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

```

```

    // TODO Auto-generated method stub
    imgDemo.setImageResource(R.drawable.fit);
}

});

}

}

```

2.2.7 | ToggleButton

ToggleButton có thể nói là dạng đặc biệt của button, nó có 2 trạng thái cơ bản là **check** và **not check**. Khi ở trạng thái check chúng ta click nó sẽ chuyển sang trạng thái not check và ngược lại



Hình 2-23: Các trạng thái của ToggleButton

Tùy theo phiên bản android mà hình ảnh ToggleButton khác nhau(check và not check).

- ↳ Thuộc tính quan trọng:
 - ☞ **textOn**: trạng thái nút đang bật
 - ☞ **textOff**: trạng thái nút đang tắt
- ↳ Một vài phương thức thường sử dụng:

Phương thức	Ý nghĩa
CharSequence getTextOff()	Trả về một chuỗi khi giá trị trạng thái button là unchecked
CharSequence getTextOn()	Trả về một chuỗi giá trị khi trạng thái button là

	checked
public void setChecked (boolean status)	Thay đổi trạng thái của ToggleButton
public Boolean isChecked()	Kiểm tra trạng thái của ToggleButton, có 2 giá trị true hoặc false

↳ Thuộc tính thường dùng của ToggleButton

- ☞ **android:id:** Là thuộc tính duy nhất của ToggleButton
- ☞ **android:checked:** Checked là thuộc tính của ToggleButton dùng để set trạng thái của ToggleButton. Giá trị là true hoặc false, nếu giá trị là true thì trạng thái ToggleButton là checked, ngược lại là false thì trạng thái của ToggleButton là unchecked. Chúng ta cũng có thể set trạng thái của ToggleButton bên Java Code bằng cách dùng phương thức setChecked(boolean status)
- ☞ **android:checked:** checked là thuộc tính của ToggleButton dùng để set trạng thái của ToggleButton. Giá trị là true hoặc false, nếu giá trị là true thì trạng thái ToggleButton là checked, ngược lại là false thì trạng thái của ToggleButton là unchecked. Chúng ta cũng có thể set trạng thái của ToggleButton bên Java Code bằng cách dùng phương thức setChecked(boolean status)
- ☞ **android:textOn** Và **android:textOff:** thuộc tính textOn được sử dụng để hiển thị câu thông báo khi ở trạng thái checked. Chúng ta có thể set textOn trong XML, hoặc trong Java Class
- ☞ **android:textColor:** Thuộc tính này dùng xác định màu chữ, dạng màu chữ: “#argb”, “#rgb”, “#rrggb”, hoặc “#aarrggbb”
- ☞ **android:textSize:** Thuộc tính textSize xác định kích thước nội dung văn bản của ToggleButton. Chúng ta có thể đặt kích thước văn bản theo: sp(scale independent pixel) hoặc dp(density pixel)
- ☞ **android:textStyle:** Thuộc tính xác định loại văn bản của ToggleButton, thông thường có các loại văn bản:bold, italic và normal. Nếu chúng ta muốn sử nhiều hơn một loại văn bản thì phải thêm phép toán hoặc “|” vào giữa các loại văn bản
- ☞ **android:background:** Thuộc tính này xác định màu nền cho ToggleButton

- ☞ **android:padding**: Thuộc tính này xác định khoảng cách từ đường viền của ToggleButton với nội dung nó chứa: left, right, top or bottom.
- ☞ **android:drawableBottom**, **android:drawableTop**, **android: drawableRight** và **android:drawableLeft**: Các thuộc tính này hiển thị hình trong res/drawable theo các hướng: bottom, top, right và left nội dung văn bản của ToggleButton.

Ví dụ 13. Trong ví dụ này chúng ta sẽ làm app có một **ToggleButton**. Khi người sử dụng click **ToggleButton** sẽ hiện thị trạng thái hiện tại của **ToggleButton** qua đối tượng TOAST



Hình 2-24: Ví dụ ToggleButton trạng thái tắt



Hình 2-25: Ví dụ ToggleButton trạng thái mở

Bước 1: Tạo một project tên là DemoToggleButton. **File** → **New** → **Android Application Project** điền các thông tin → **Next** → **Finish**.

Bước 2: Mở **res** → **layout** → **xml** (hoặc) **activity_main.xml**

```

<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"
    >

<ToggleButton
    android:id="@+id/simpleToggleButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:textSize="25sp"
    android:layout_centerHorizontal="true"
    android:textColor="#f00"
    android:textOn="Mở"
    android:textOff="Tắt"
    android:drawableTop="@drawable/on"
/>

```

Bước 3: Mở app → src -> **MainActivity.java** và thêm code. Khi click vào **ToggleButton** sẽ lấy chuỗi trạng thái của nó bằng cách dùng phương thức **getText()**, sau đó dùng đối tượng **TOAST** để hiển thị.

```

public class MainActivity extends AppCompatActivity {
    //Khai báo các widget
    ToggleButton tglbtnSimple;

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

```

```

    setEvent();
}

private void setControl() {
    tglbtnSimple = (ToggleButton) findViewById(R.id.simpleToggleButton);
}

private void setEvent() {
    tglbtnSimple.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub
            Toast.makeText(getApplicationContext(), "Trạng thái đã chọn " +
                tglbtnSimple.getText(), Toast.LENGTH_LONG).show();
        }
    });
}

}

```

2.2.8 | Switch

Switch là một widget trong android, nó có 2 trạng thái chọn lựa. Nó được sử dụng để hiển thị trạng thái **checked** và **unchecked** của một nút thông qua một thanh trượt cho người dùng. Nó có 2 Button cơ bản là on/off cho biết trạng thái của Switch. Switch thường được sử dụng hai nút on/off trong trường hợp sau: âm thanh, WiFi, Bluetooth,

Switch: đối tượng nút bấm hai trạng thái “bật” và “tắt”, có thể thao tác bằng cách trượt ngón tay trên đối tượng.



☞ Một vài phương thức thường sử dụng:

Phương thức	Ý nghĩa
public boolean isChecked()	Trạng thái hiện tại của Switch
public void setChecked(boolean status)	Thay đổi trạng thái của Switch

↳ Thuộc tính thường dùng của Switch

- ☞ **android:id:** Là thuộc tính duy nhất của Switch
- ☞ **android:checked:** checked là thuộc tính của Switch dùng để set trạng thái của Switch. Giá trị là true hoặc false, nếu giá trị là true thì trạng thái Switch là checked, ngược lại là false thì trạng thái của Switch là unchecked. Chúng ta cũng có thể set trạng thái của Switch bên Java Code bằng cách dùng phương thức `setChecked(boolean status)`
- ☞ **android:text:** thuộc tính text dùng hiển thị nội dung trong một Switch. Chúng ta có thể set thuộc tính này trong tập tin xml hoặc java code.
- ☞ **android:gravity:** Thuộc tính này thường sử dụng để canh nội dung trong CheckBox: left, right, center, top, bottom, center_vertical, center_horizontal.
- ☞ **android:textOn** và **android:textOff:** thuộc tính textOn được sử dụng để hiển thị câu thông báo khi Switch ở trạng thái checked. Chúng ta có thể set textOn trong XML, hoặc trong Java Class.
- ☞ **android:textColor:** Thuộc tính này dùng xác định màu chữ, dạng màu chữ: “#argb”, “#rgb”, “#rrggbb”, hoặc “#aarrggbb”

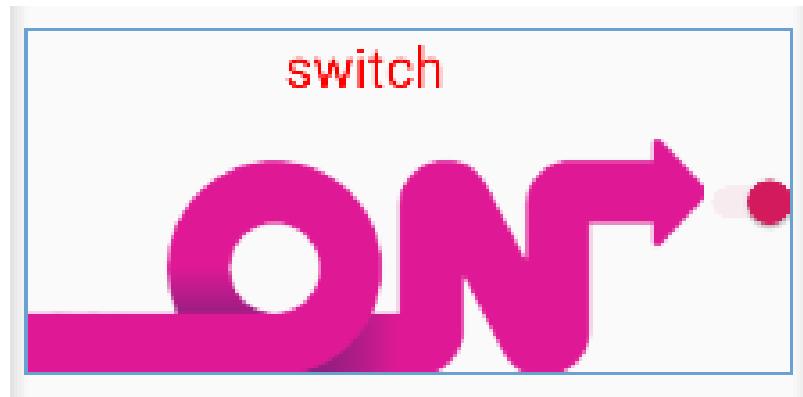
- ☞ **android:textSize:** Thuộc tính textSize xác định kích thước nội dung văn bản của Switch. Chúng ta có thể đặt kích thước văn bản theo: sp(scale independent pixel) hoặc dp(density pixel).
- ☞ **android:textStyle:** Thuộc tính xác định loại văn bản của Switch, thông thường có các loại văn bản:bold, italic và normal. Nếu chúng ta muốn sử nhiều hơn một loại văn bản thì phải thêm phép toán hoặc "|" vào giữa các loại văn bản.
- ☞ **android:background:** Thuộc tính này xác định màu nền cho Switch.
- ☞ **android:padding:** Thuộc tính này xác định khoảng cách từ đường viền của Switch với nội dung nó chứa: left, right, top or bottom.
- ☞ **android:drawableBottom, android:drawableTop, android: drawableRight và android:drawableLeft:** Các thuộc tính này hiển thị hình trong res/drawable theo các hướng: bottom, top, right và left nội dung văn bản của Switch.

<Switch

```

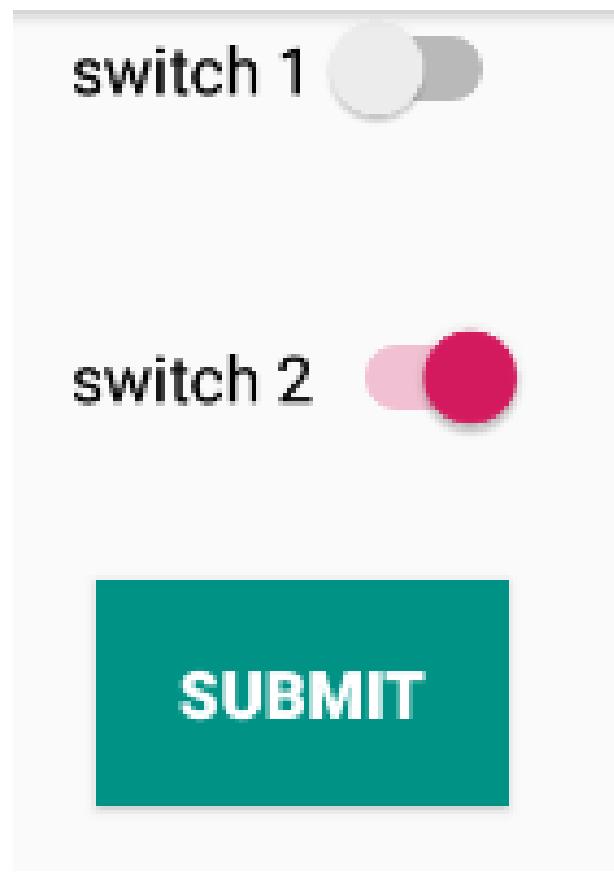
android:id="@+id/simpleSwitch"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:checked="true"
android:drawableBottom="@drawable/on"
android:gravity="center"
android:text="switch"
android:textColor="#f00"
android:textOff="Off"
android:textOn="On"
android:textSize="25sp" />

```

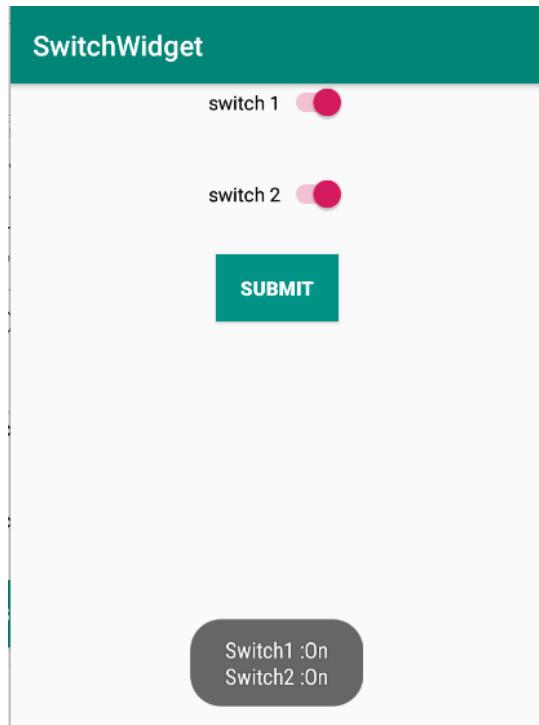


Hình 2-26: Minh Họa Switch

Ví dụ 14. Xây dựng ứng dụng có 2 Switch và một Button. Khi người sử dụng chọn Switch và click Button sẽ hiện thị trạng thái hiện tại của Switch qua đối tượng TOAST



Hình 2-27: Ví dụ Switch trước khi nhấn submit



Hình 2-28: Ví dụ Switch sau khi nhấn submit

Bước 1: Tạo một project tên là DemoSwitch: **File** → **New** → **Android Application Project** điền các thông tin → **Next** → **Finish**.

Bước 2: Vào thư mục **res/values** bổ sung **string.xml**

<**resources**>

```

<string name="app_name">SwitchWidget</string>
<string name="hello_world">Hello world!</string>
<string name="action_settings">Settings</string>
<string name="switch1">Switch 1</string>
<string name="switch2">Switch 2</string>
<string name="btndsubmit">Submit</string>

```

</**resources**>

Mở **res** → **layout** → **xml** (hoặc) **activity_main.xml**

```

<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:gravity="center_horizontal"
    android:orientation="vertical">

<Switch
    android:id="@+id/simpleSwitch1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="false"
    android:text="switch 1"
    android:textOff="Off"
    android:textOn="On" />

<Switch
    android:id="@+id/simpleSwitch2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="39dp"
    android:checked="true"
    android:text="switch 2"
    android:textOff="Off"
    android:textOn="On" />

<Button
    android:id="@+id/btnSubmit"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/simpleSwitch2"
    android:layout_marginTop="30dp"
    android:background="#009284"
    android:padding="10dp"
    android:text="Submit"
```

```
    android:textColor="#fff"
    android:textStyle="bold" />

</LinearLayout>
```

Bước 3: Mở app → src -> MainActivity.java và thêm code. Khi click vào CheckBox sẽ lấy các giá trị của CheckBox, sau dùng đối tượng TOAST để hiển thị

```
public class MainActivity extends AppCompatActivity {
    //Khai báo các widget
    Switch simpleSwitch1, simpleSwitch2;
    Button submit;

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

    private void setControl() {
        simpleSwitch1 = (Switch) findViewById(R.id.simpleSwitch1);
        simpleSwitch2 = (Switch) findViewById(R.id.simpleSwitch2);
        submit = (Button) findViewById(R.id.btnSubmit);
    }
}
```

```

private void setEvent() {
    submit.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String statusSwitch1, statusSwitch2;
            if (simpleSwitch1.isChecked())
                statusSwitch1 = simpleSwitch1.getTextOn().toString();
            else
                statusSwitch1 = simpleSwitch1.getTextOff().toString();
            if (simpleSwitch2.isChecked())
                statusSwitch2 = simpleSwitch2.getTextOn().toString();
            else
                statusSwitch2 = simpleSwitch2.getTextOff().toString();
            Toast.makeText(getApplicationContext(), "Switch1 :" + statusSwitch1 +
                    "|n" + "Switch2 :" + statusSwitch2, Toast.LENGTH_LONG).show(); // display the
            current state for switch's
        }
    });
}
}

```

2.2.9 | ScrollView.

ScrollView là một dạng đặc biệt của FrameLayout ở chỗ nó cho phép người dùng di chuyển qua một danh sách các quan điểm mà chiếm nhiều không gian hơn màn hình vật lý. Các ScrollView có thể chứa chỉ một Child View hay ViewGroup, mà thường là một LinearLayout. Cho phép người dùng di chuyển qua một danh sách chiếm nhiều không gian hơn màn hình vật lý.

- ❖ Thuộc tính thường dùng của ScrollView
- ❖ **android:id:** Là thuộc tính duy nhất của ScrollView.

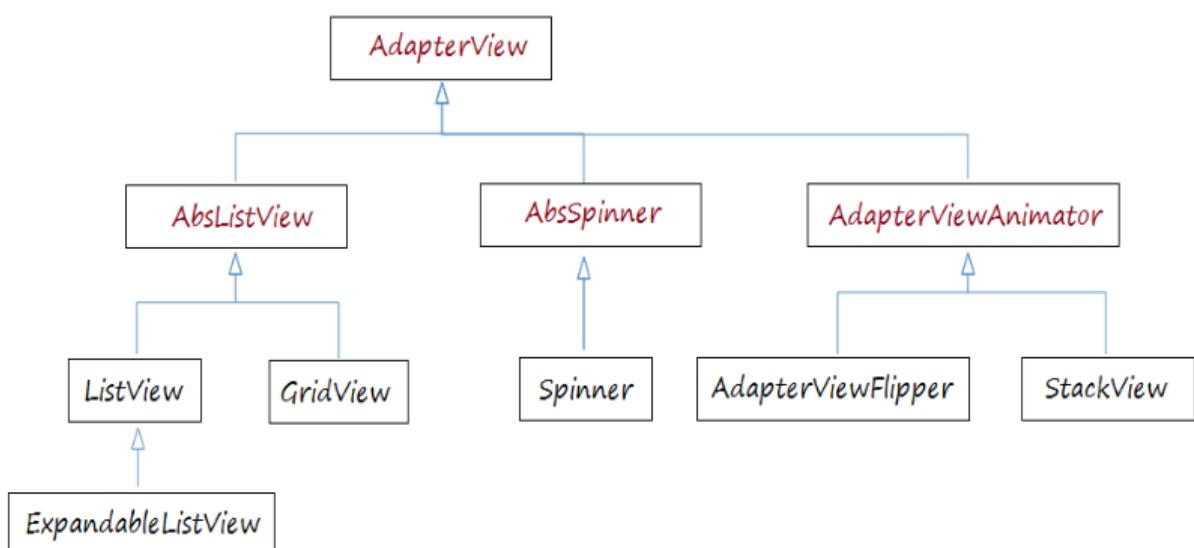
- ☞ **android:scrollbars**: Thuộc tính này được sử dụng hiển thị thanh cuộn trong ScrollView, giá trị có thể là "vertical, horizontal". Mặc định cuộn theo thẳng đứng

2.3 | Các điều khiển hiển thị danh sách

2.3.1 | Các dạng Adapter

Adapter giữ vai trò như cầu nối giữa các đối tượng điều khiển dạng danh sách (AdapterView) và các dữ liệu bên dưới, cung cấp các phương thức truy xuất dữ liệu. Cho phép thực hiện quản lý giao diện, số lượng chỉ mục trên AdapterView và thực hiện truy vấn dữ liệu, sắp xếp dữ liệu. Bên cạnh đó, Adapter cũng chịu trách nhiệm tạo ra các View con trong tập các dữ liệu.

AdapterView là các đối tượng điều khiển dạng tập hợp, cho phép hiển thị thông tin cơ bản theo dạng danh sách và thực hiện quản lý thông tin theo từng mục riêng biệt.



Hình 2-29: Các dạng Adapter

✚ Adapter trong Android

- ☞ **BaseAdapter** : Là lớp adapter cơ sở cho các Adapter khác
- ☞ **ArrayAdapter**: MộtListAdapter có thể quản lý một ListView chứa danh sách các phần tử có kiểu bất kỳ
- ☞ **Custom ArrayAdapter**: Thường được sử dụng để hiển thị một danh sách tùy chỉnh

- ❖ **SimpleAdapter:** Nó là một Adapter đơn giản và dễ hiểu để ánh xạ dữ liệu vào những View được định nghĩa trong một tập tin XML
- ❖ **Custom SimpleAdapter:** Nó được sử dụng để hiển thị một danh sách được tùy chỉnh để truy cập các mục con của List hoặc Grid.
- ❖ **BaseAdapter:** Là lớp adapter cơ sở cho các Adapter thường dùng khác như ArrayAdapter<T>, CursorAdapter, SimpleAdapter. BaseAdapter thường đóng vai trò Adapter cho các ListView và Spinner sẽ được tìm hiểu trong các phần tiếp theo.

```
class CustomAdapter extends BaseAdapter {
```

```
@Override
```

```
public int getCount() {  
    return 0;  
}
```

```
@Override
```

```
public Object getItem(int i) {  
    return null;  
}
```

```
@Override
```

```
public long getItemId(int i) {  
    return 0;  
}
```

```
@Override
```

```
public View getView(int i, View view, ViewGroup viewGroup) {  
  
    return null;  
}  
}
```

```

public class MyAdapter extends ArrayAdapter {

    public MyAdapter(Context context, int resource, int textViewResourceId, List
objects) {
        super(context, resource, textViewResourceId, objects);
    }

    @Override
    public int getCount() {
        return super.getCount();
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        return super.getView(position, convertView, parent);
    }
}

```

Ví dụ: Tạo một ArrayAdapter chứa các dữ liệu chuỗi.

```

String arr[]={ "One", "Two", "Three", "Four"};
ArrayAdapter<String> adapter = new
    ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, arr));

```

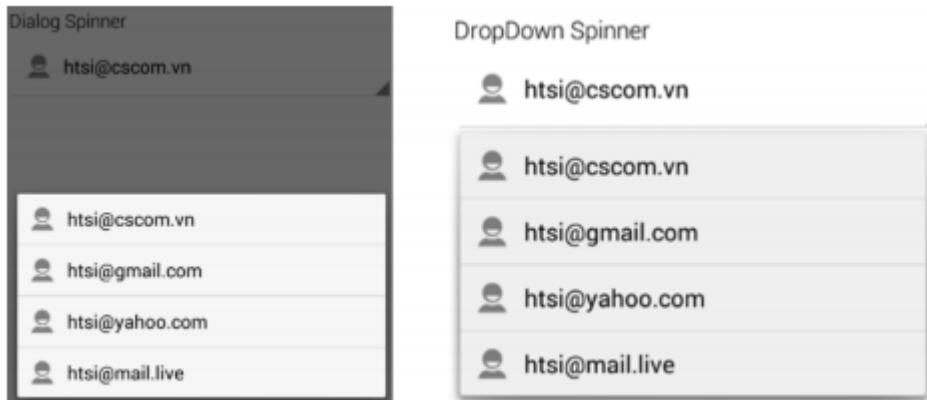
Trong đó:

- ☞ *android.R.layout.simple_list_item_1*: là layout mặc định do Android xây dựng sẵn cho 1 item của ListView.
- ☞ Tuy nhiên, nếu muốn sử dụng với những giao diện phức tạp hơn như ImageView, hoặc nhiều TextView ta nên ghi đè lại *getView (int, View, ViewGroup)* để trả lại kiểu giao diện mà muốn.

2.3.2 | Spinner.

Spinner là đối tượng điều khiển hiển thị một danh mục ở một thời điểm, người dùng có thể lựa chọn một trong nhiều danh mục để hiển thị. Bao gồm hai chế độ hiển thị pop-up lựa chọn (*spinnerMode*)

- ❖ Dialog
- ❖ Dropdown

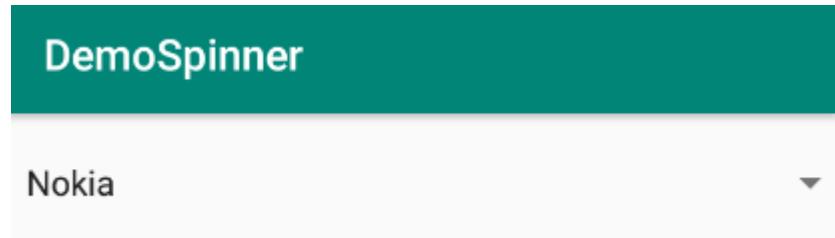


Hình 2-30: Các dạng Spinner

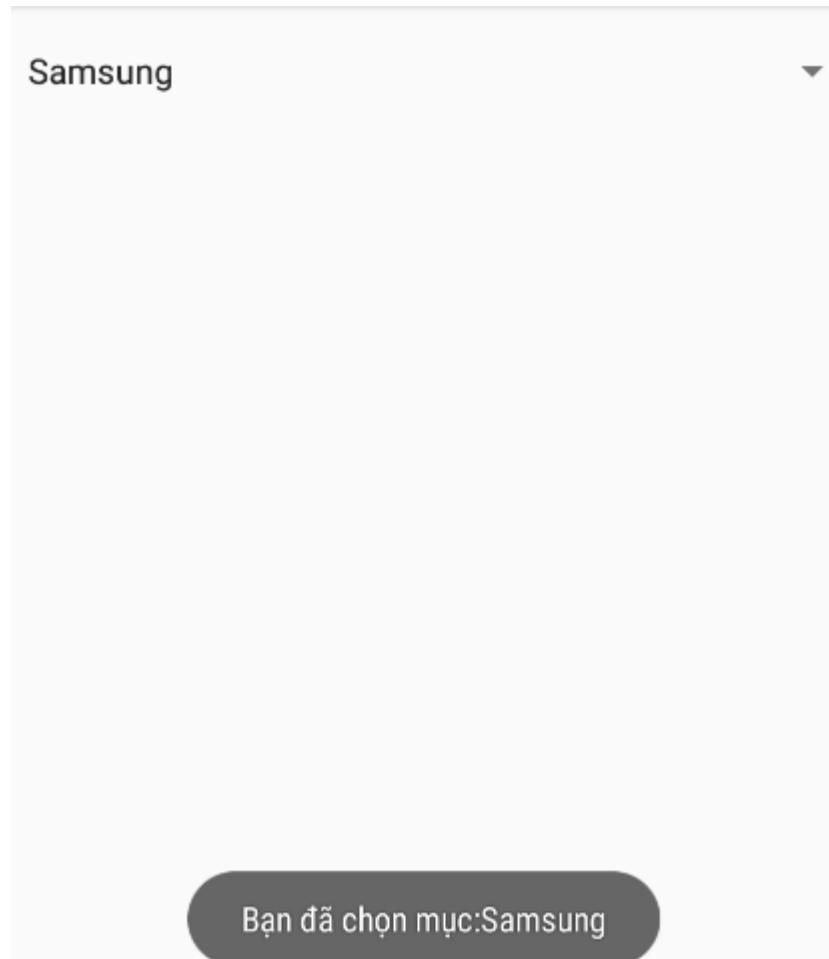
- ❖ Thuộc tính XML quan trọng:
 - ❖ **spinnerMode:** dialog | dropdown
 - ❖ **prompt:** string
 - ❖ **popupBackground:** drawable | color
 - ❖ **entries:** string-array
- ❖ Một số phương thức quan trọng:
 - ❖ **setAdapter(SpinnerAdapter)**
 - ❖ **setPrompt(CharSequence) – setPrompt(int resId)** (Dialog Mode)
 - ❖ **setPopupBackgroundResource(int)**
 - ❖ **setPopupBackgroundDrawable(Drawable)**
- ❖ Các sự kiện thường dùng trong Spinner
 - ❖ **setOnItemSelectedListener** Sự kiện này xảy ra khi người dùng click chọn Spinner
 - ❖ **onItemSelected:** gọi phương thức này được gọi khi có một sự kiện chọn item nào đó.

- ☞ **onNothingSelected**: phương thức này được gọi khi click vào Spinner mà không chọn item nào cả.

Ví dụ 15. xây dựng Spinner hiển thị danh sách các mặt hàng, khi chọn một mặt hàng sẽ hiển thị thông báo.



Hình 2-31: ví dụ Spiner trước khi chọn



Hình 2-32: Ví dụ Spinner sau khi chọn

Bước 1: Tạo một project tên là DemoSpinner: File → New → Android Application Project điền các thông tin → Next → Finish.

Bước 2: Mở res → layout → xml (hoặc) activity_main.xml

```
<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"  
>  
  
<Spinner  
    android:id="@+id/spinner1"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="20dp"  
/>  
</LinearLayout>
```

Bước 3: Mở app → src -> MainActivity.java và thêm code. Khởi tạo Spinner. Tiếp theo, tạo 1 mảng dữ liệu và kết nối Spinner thông qua đối tượng ArrayAdapter. Trong bước này chúng ta cũng thiết lập sự kiện setOnItemSelectedListener(new OnItemSelectedListener()) cho Spinner

```
public class MainActivity extends AppCompatActivity {  
    //Khai báo các widget  
    String mobilePhones[] = {"Nokia", "Samsung", "Iphone", "HTC", "BPhone",  
    "MobileStar"};  
    Spinner spnMobile;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        setControl();  
        setEvent();  
    }  
  
    private void setControl() {  
        spnMobile = (Spinner) findViewById(R.id.spinner1);  
    }  
}
```

```

private void setEvent() {
    ArrayAdapter adapter = new ArrayAdapter<String>(this,
    android.R.layout.simple_spinner_item, mobilePhones);
    adapter.setDropDownViewResource(android.R.layout.select_dialog_singlechoice);
    spnMobile.setAdapter(adapter);
    spnMobile.setOnItemSelectedListener(new
        AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> arg0, View arg1,
                int arg2, long arg3) {
                // TODO Auto-generated method stub
                Toast.makeText(getApplicationContext(), "Bạn đã chọn mục:" +
                    mobilePhones[arg2], Toast.LENGTH_LONG).show();
            }
            @Override
            public void onNothingSelected(AdapterView<?> arg0) {
                // TODO Auto-generated method stub
                Toast.makeText(MainActivity.this, "Bạn chưa chọn",
                    Toast.LENGTH_SHORT).show();
            }
        });
}

```

2.4 | Các điều khiển danh sách listview

2.4.1 | ListView.

ListView là một dạng điều khiển nâng cao có chức năng hỗ trợ hiển thị dữ liệu theo dạng từng dòng. ListView cần một đối tượng Adapter để quản lý và giúp hiển thị dữ liệu.



Hình 2-33: Minh họa ListView

↳ Thuộc tính thường dùng của ListView

- ☞ **android:id:** Là thuộc tính duy nhất của ListView.
- ☞ **android:divider:** Thuộc tính này có thể là một image hay màu dùng để phân chia giữa các dòng trong ListView.
- ☞ **android:dividerHeight:** Thuộc tính này xác định chiều cao của thuộc tính android:divider.
- ☞ **android:listSelector:** Thuộc tính này thường được sử dụng để thiết lập màu hoặc hình dòng được chọn trong listView. Thường nó sử dụng màu cam hoặc màu xanh dương, nhưng chúng ta cũng có thể thiết lập lại màu hoặc image cho ListView

↳ Sự kiện thường dùng trong ListView

- ☞ **setOnItemClickListener:** Sự kiện này xảy ra khi người dùng click lên ListView.

- ☞ **setOnItemLongClickListener**: Sự kiện được gắn cho ListView Item, khi nhấn lâu từ 2.5 tới 3 giây thì sự kiện này sẽ xảy ra

↳ Các xử lý trên ListView

- Cách lấy ListView thông qua Id của ListView

```
ListView lvTenLV=(ListView) findViewById(R.id.idcuaListView);
```

↳ Cách tạo ArrayAdapter và đưa dữ liệu từ mảng vào ArrayAdapter

```
ArrayAdapter<[Kiểu mảng]>[Tên mảng adapter]  
= new ArrayAdapter<Kiểu mảng>  
(this, android.R.layout.simple_list_item_1, [tenMangDuLieu]);
```

↳ Giải thích ý nghĩa các đối số trong phương thức ArrayAdapter:

- Đối số thứ 1: **this**

Đại diện cho **context** của Activity hiện tại, có thể viết MainActivity.this (nếu viết như thế này thì ở bất kỳ vị trí nào nó cũng hiểu là context của MainActivity).

- Đối số thứ 2: **android.R.layout.simple_list_item_1**

Là layout Listview mà được Android xây dựng sẵn. Nó được lưu trong SDK/platforms/ android-api(x)/ data/ res/ layout/ simple_list_item_1.xml.

- Đối số thứ 3: **[tenMangDuLieu]**

Là mảng chứa dữ liệu cần hiển thị lên ListView.

↳ Thuộc tính XML quan trọng:

☞ **listSelector**: drawable

☞ **divider**: drawable

☞ **dividerHeight**: dimen

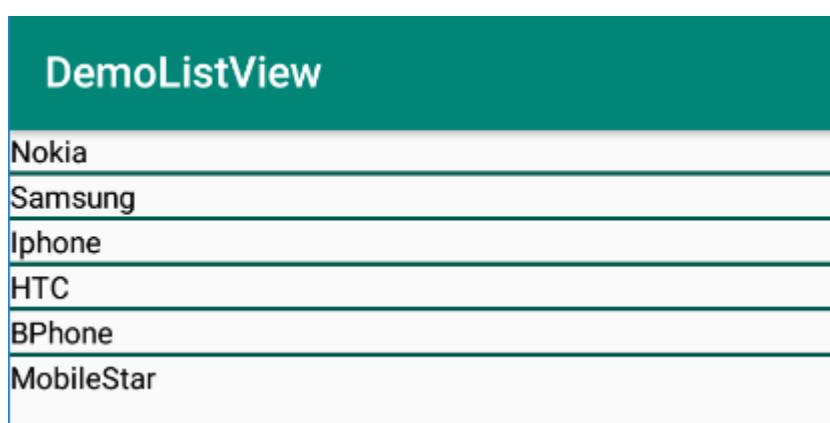
☞ **entries**: string-array

↳ Một số phương thức quan trọng:

- ☞ **setAdapter(Class Extends)**
- ☞ **addHeaderView(View) – removeHeaderView(View)**
- ☞ **addFooterView(View) – removeFooterView(View)**
- ☞ **setSelection(int)**
- ☞ **smoothScrollToPosition(int)**

Ví dụ 16. Xây dựng ứng dụng gồm hiển thị danh sách sản phẩm sử dụng ListView.

Khi chọn sản phẩm sẽ hiển thị thông báo



Hình 2-34: Ví dụ ListView hiển thị danh sách sản phẩm



Hình 2-35: Ví dụ ListView khi chọn sản phẩm

Bước 1: Tạo một project tên là DemoListView: **File → New → Android Application Project** điền các thông tin → **Next →Finish**.

Bước 2: Mở res → layout → xml (hoặc) activity_main.xml

```
<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/simpleListView"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:divider="@color/colorPrimaryDark"  
    android:dividerHeight="2dp" />  
  
</LinearLayout>
```

Bước 3: Tạo mới một Activity activity_listview.xml vào trong thư mục layout và thêm vào 1 TextView.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical">  
  
    <TextView  
        android:id="@+id/textView"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:layout_gravity="center"  
        android:text="@string/app_name"  
        android:textColor="#000" />  
  
</LinearLayout>
```

Bước 4: Mở app → src -> **MainActivity.java** và thêm code, tạo 1 mảng dữ liệu và kết nối **ListView** thông qua đối tượng **ArrayAdapter**.

```

public class MainActivity extends AppCompatActivity {
    //Khai báo các widget
    String mobilePhones[] = {"Nokia", "Samsung", "Iphone", "HTC", "BPhone",
    "MobileStar"};
    ListView simpleList;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        setControl();
        setEvent();
    }
    private void setControl() {
        simpleList = (ListView) findViewById(R.id.simpleListView);
    }

    private void setEvent() {
        ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(this,
R.layout.activity_listview, R.id.textView, mobilePhones);
        simpleList.setAdapter(arrayAdapter);
        simpleList.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
                Toast.makeText(MainActivity.this, "Bạn chọn " +
mobilePhones[position], Toast.LENGTH_SHORT).show();
            }
        });
    }
}

```

2.4.2 | Custom ListView.

Trong thực tế, các ứng dụng Android có liên quan đến ListView thì đa phần chúng ta phải điều chỉnh lại layout cho đúng với yêu cầu của khách hàng. Và cách điều chỉnh lại layout là tạo Custom Layout cho ListView

Sự khác nhau giữa ListView thông thường và ListView có layout được điều chỉnh lại là ở việc khởi tạo Adapter.

Sau khi đã có ListView trên giao diện, chúng ta có thể tạo Custom Layout cho ListView như sau:

Bước 1: Tạo một lớp dùng để quản lý dữ liệu.

Bước 2: Tạo thêm một layout cho một item của ListView.

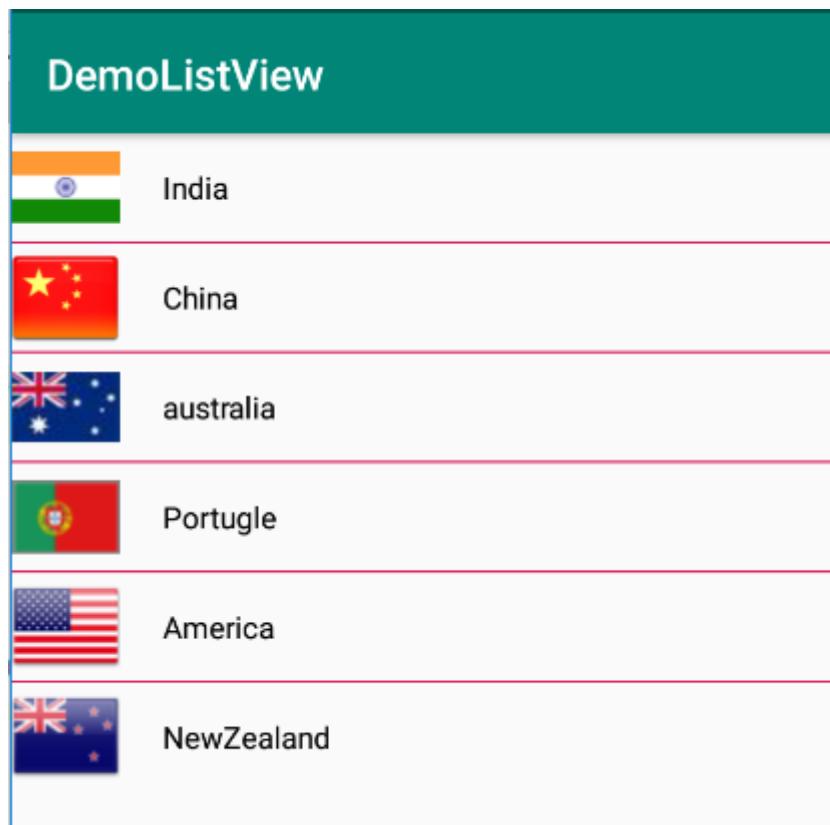
Bước 3: Tạo lớp Custom Adapter kế thừa từ lớp ArrayAdapter.

Bước 4: Hiển thị dữ liệu lên ListView

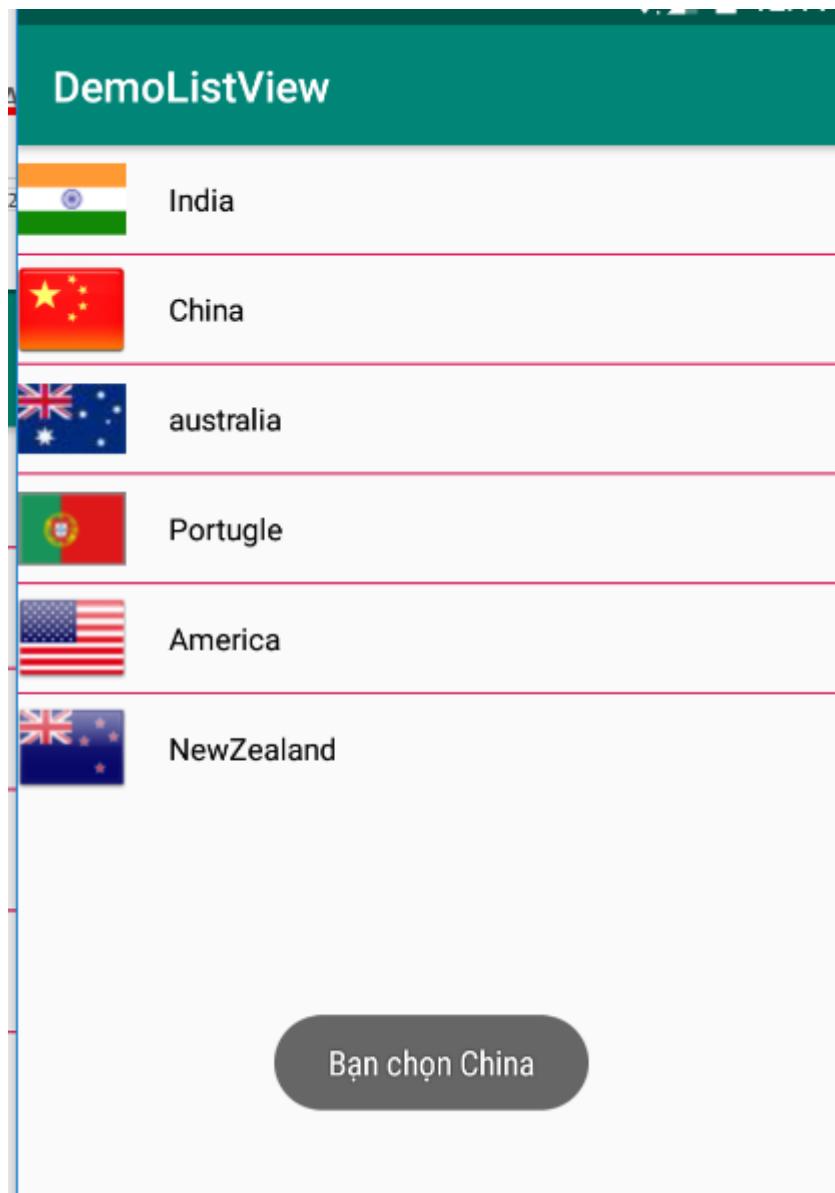


Hình 2-36: Ví dụ hiển thị ListView và Custom ListView

Ví dụ 17. Xây dựng ứng dụng gồm có một **ListView** để hiển thị tên các nước cùng với lá cờ nước đó.



Hình 2-37: Ví dụ Custom Listview hiển thị danh sách



Hình 2-38: Ví dụ Custom ListView khi chọn danh sách

Bước 1: Tạo một project tên là ListViewCustom: **File → New → Android Application Project** điền các thông tin → **Next →Finish**.

Bước 2: Mở res → layout → xml (hoặc) activity_main.xml

```
<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/simpleListView"  
        android:layout_width="fill_parent"  
        android:layout_height="wrap_content"  
        android:divider="@color/colorAccent"  
        android:dividerHeight="1dp"  
        android:footerDividersEnabled="false" />  
  
</LinearLayout>
```

Bước 3: Tạo mới một activity_listview.xml vào trong thư mục layout và thêm code sau

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:orientation="horizontal">  
    <ImageView  
        android:id="@+id/icon"  
        android:layout_width="50dp"  
        android:layout_height="50dp"  
        android:src="@drawable/america" />  
    <TextView  
        android:id="@+id/textView"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_gravity="center"  
        android:text="America"  
        android:layout_marginLeft="20dp"  
        android:textColor="#000" />  
    </LinearLayout>
```

Bước 4: Tạo mới một lớp CustomAdapter.java bên trong package và thêm code sau

```
public class CustomAdapter extends ArrayAdapter {  
    Context context;  
    String countryList[];  
    int flags[];  
    int layoutID;  
    public CustomAdapter(Context context, int layoutID, String[] countryList, int  
    flags[]) {  
        super(context, layoutID);  
        this.context = context;  
        this.countryList = countryList;  
        this.flags = flags;  
        this.layoutID = layoutID;  
    }  
}
```

```

@Override
public int getCount() {
    // TODO Auto-generated method stub
    return countryList.length;
}

@Override
public View getView(int position, View convertView, ViewGroup parent) {
    // TODO Auto-generated method stub
    LayoutInflator inflater = (LayoutInflator.from(context));
    convertView = inflater.inflate(layoutID, null);
    TextView country = (TextView) convertView.findViewById(R.id.textView);
    ImageView icon = (ImageView) convertView.findViewById(R.id.icon);
    country.setText(countryList[position]);
    icon.setImageResource(flags[position]);
    return convertView;
}

```

Bước 5: Mở app → src -> **MainActivity.java** và thêm code Trong bước này chúng ta khởi tạo **ListView**. Tiếp theo, chúng ta tạo 1 mảng cho image, 1 mảng cho tên image. Lưu các hình ảnh vào thư mục drawable.

```
public class MainActivity extends AppCompatActivity {  
    //Khai báo các widget  
    ListView simpleList;  
    String countryList[] = {"India", "China", "australia", "Portugle", "America",  
    "NewZealand"};  
    int flags[] = {R.drawable.india, R.drawable.china, R.drawable.australia,  
    R.drawable.portugle, R.drawable.america, R.drawable.new_zealand};  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        setControl();  
        setEvent();  
    }  
    private void setControl() {  
        simpleList = (ListView) findViewById(R.id.simpleListView);  
    }  
}
```

```

private void setEvent() {
    CustomAdapter adapter = new CustomAdapter(this,R.layout.activity_listview,
countryList, flags);
    simpleList.setAdapter(adapter);

    simpleList.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
            Toast.makeText(MainActivity.this, "Bạn chọn " + countryList[position],
Toast.LENGTH_SHORT).show();
        }
    });
}
}

```

2.5 | Các điều khiển nâng cao

2.5.1 | TimePickerDialog .

Time Picker trong Android cho phép lựa chọn thời gian của ngày trong chế độ hoặc 24 h hoặc AM/PM. Thời gian bao gồm các định dạng hour, minute, và clock. Android cung cấp tính năng này thông qua lớp TimePicker.

↳ Các phương thức thường dùng của TimePicker

- ↳ **setCurrentHour(Integer currentHour)**: Phương thức này dùng để thiết lập ngày hiện tại cho mộtTimePicker
- ↳ **setCurrentMinute(Integer currentMinute)**: Phương thức này dùng để thiết lập ngày hiện tại cho mộtTimePicker
- ↳ **getCurrentHour()**: Phương thức này lấy giờ hiện tại của TimePicker
- ↳ **getCurrentMinute()**: Phương thức này lấy phút hiện tại của TimePicker

- ☞ **setIs24HourView(Boolean is24HourView)**: Phương thức dùng thiết lập chế độ giờ hiển thị là 24h hay là hiển thị dạng AM/PM. Trong phương thức này thiết lập chế độ true/false. Nếu giá trị true chế độ hiển thị theo 24h, nếu false hiển thị theo chế độ AM/PM
- ☞ **is24HourView()**: Phương này kiểm tra xem chế độ hiện tại là 24h hay AM/PM. Phương thức này trả về true nếu nó đang ở chế độ 24h ngược lại false đang ở chế độ AM/PM
- ☞ **setOnTimeChangedListener(TimePicker.OnTimeChangedListener onTimeChangedListener)**: Phương thức này thiết lập hàm callback mà chỉ rằng thời gian đã được chỉnh sửa bởi người dùng.

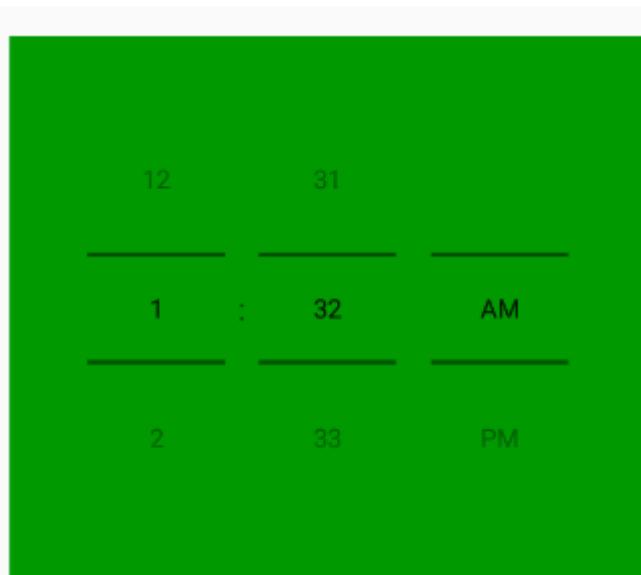
↳ Một số thuộc tính thường dùng của TimePicker

- ☞ **android:id**: Là thuộc tính duy nhất của TimePicker
- ☞ **android:timePickerMode**: thuộc tính này thường được sử dụng để thiết lập chế độ hiển thị dạng đồng hồ là spinner hay là clock.
- ☞ **android:background**: Thuộc tính này thiết lập màu nền hoặc image trong thư mục drawable cho TimePicker.
- ☞ **android:padding**: Thuộc tính này xác định khoảng cách từ đường viền của TimePicker với nội dung nó chứa: left, right, top or bottom.

<TimePicker

```
    android:id="@+id/simpleTimePicker"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:timePickerMode="spinner"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="50dp"
    android:padding="50dp"/> 
```

Ví dụ 18. Xây dựng ứng dụng gồm có một TimePicker và một TextView. Khi người sử dụng tùy chỉnh giờ trên TimePicker chúng ta sẽ lấy giá trị của nó rồi hiển thị trên TextView và Toast.



Hình 2-39: Ví dụ TimePicker

Bước 1: Tạo một project tên là DemoTimePicker: **File** → **New** → **Android Application Project** điền các thông tin → **Next** → **Finish**.

Bước 2: Mở res → layout → xml (hoặc) activity_main.xml

```
<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"  
    android:gravity="center"  
    tools:context=".MainActivity">
```

<TimePicker

```
    android:id="@+id/simpleTimePicker"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:background="#090"  
    android:padding="20dp"  
    android:timePickerMode="spinner" />
```

```

<TextView
    android:id="@+id/time"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Time Is ::"
    android:textColor="#090"
    android:textSize="20sp"
    android:textStyle="bold" />

```

</LinearLayout>

Bước 3: Mở app → src -> **MainActivity.java** và thêm code. Trong bước này chúng ta khởi tạo TimePicker và TextView. TextView dùng để hiển thị thời gian của TimePicker. Trong bước này chúng ta thiết lập sự kiện cho TimePicker thông qua sự kiện **setOnTimeChangedListener()** chúng ta sẽ lấy giá trị thời gian hiện tại của TimePicker sau đó hiển thị ra TextView và Toast.

```

public class MainActivity extends AppCompatActivity {
    //Khai báo các widget
    TextView time;
    TimePicker simpleTimePicker;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        setControl();
        setEvent();
    }
    private void setControl() {
        time = (TextView) findViewById(R.id.time);
        simpleTimePicker = (TimePicker) findViewById(R.id.simpleTimePicker);
    }
}

```

```

private void setEvent() {
    simpleTimePicker.setIs24HourView(false); // used to display AM/PM mode
    // perform set on time changed listener event
    simpleTimePicker.setOnTimeChangedListener(new
    TimePicker.OnTimeChangedListener() {
        @Override
        public void onTimeChanged(TimePicker view, int hourOfDay, int minute) {
            // display a toast with changed values of time picker
            Toast.makeText(getApplicationContext(), hourOfDay + " " + minute,
            Toast.LENGTH_SHORT).show();
            time.setText("Time is :: " + hourOfDay + ":" + minute); // set the
            current time in text view
        }
    });
}
}

```

2.5.2 | DatePicker Dialog.

Date Picker trong Android cho phép lựa chọn date bao gồm ngày, tháng, và năm trong Custom UI của. Với tính năng này, Android cung cấp các thành phần DatePicker và DatePickerDialog.

⇒ Các phương thức thường dùng của DatePicker

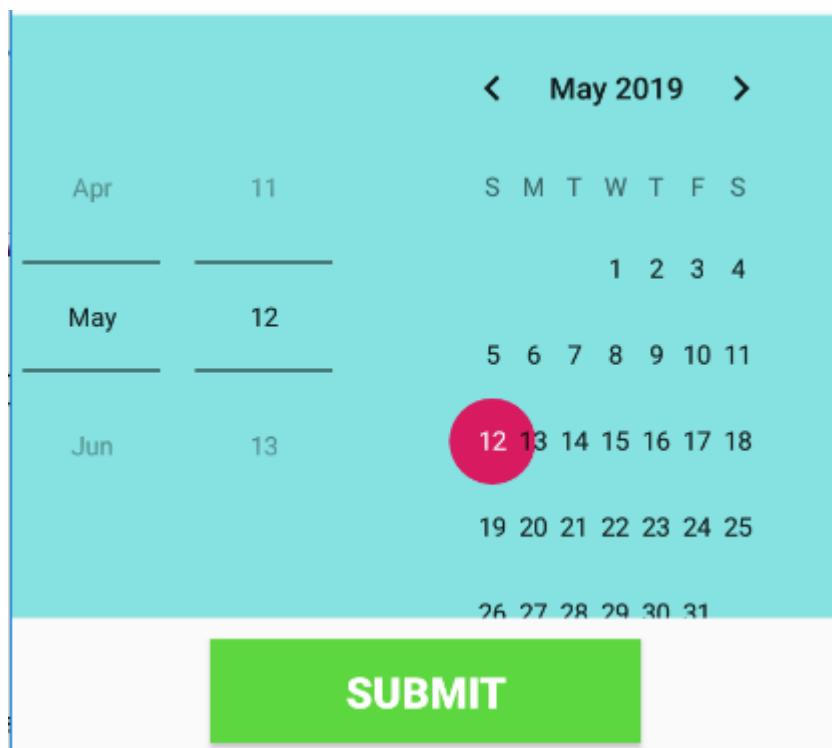
- ☛ **setSpinnersShown(boolean shown):** Phương thức này thường sử dụng để hiện thi DatePicker ở dạng spinner hay không. Trong phương thức này chúng ta có thể thiết lập giá trị true hoặc false. Nếu giá trị là false thì spinner không hiển thị. Mặc định là true
- ☛ **getDayOfMonth():** Phương thức này được sử dụng để lấy ngày trong tháng từ DatePicker. Phương thức này trả về một số nguyên.
- ☛ **getMonth():** Phương thức này được sử dụng để lấy tháng từ DatePicker. Phương thức này trả về một số nguyên.

- ☞ **getYear()**: Phương thức này được sử dụng để lấy năm từ DatePicker. Phương thức này trả về một số nguyên.
- ☞ **getFirstDayOfWeek()**: Phương thức này lấy ngày đầu tiên trong tuần. Phương thức này trả về một số nguyên.

↳ Một số thuộc tính thường dùng của DatePicker

- ☞ **android:id**: Là thuộc tính duy nhất của DatePicker.
- ☞ **android: datePickerMode**: Thuộc tính này thường dùng để thiết lập DatePicker theo chế độ spinner hay calendar.
- ☞ **android:background**: Thuộc tính này thiết lập màu nền hoặc image trong thư mục drawable cho DatePicker.
- ☞ **android:padding**: Thuộc tính này xác định khoảng cách từ đường viền của TimePicker với nội dung nó chứa: left, right, top or bottom.

Ví dụ 19. Xây dựng ứng dụng gồm có một **DatePicker** và một **Button**. Khi người sử dụng chọn ngày trên **DatePicker** và click vào **Button "Submit"** chúng ta sẽ lấy giá trị của **DatePicker** rồi hiển thị lên **Toast**.



Hình 2-40: Ví dụ DatePicker

Bước 1: Tạo một project tên là DemoDatePicker: File → New → Android Application Project điền các thông tin → Next → Finish.

Bước 2: Mở res → layout → xml (hoặc) activity_main.xml

```
<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"  
    android:gravity="center"  
    tools:context=".MainActivity">  
  
<DatePicker  
    android:id="@+id/simpleDatePicker"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:background="#86e1e1"  
    android:datePickerMode="spinner" />  
  
<Button  
    android:id="@+id	btnSubmit"  
    android:layout_width="200dp"  
    android:layout_height="wrap_content"  
    android:layout_below="@+id/simpleDatePicker"  
    android:layout_marginTop="10dp"  
    android:background="#5dd73f"  
    android:text="SUBMIT"  
    android:textColor="#fff"  
    android:textSize="20sp"  
    android:textStyle="bold" />  
  
</LinearLayout>
```

Bước 3: Mở app → src -> **MainActivity.java** và thêm code. Trong bước này chúng ta khởi tạo **DatePicker** và **Button**. Sau đó, thiết lập sự kiện **onClickListener()** cho **Button**. Khi người sử dụng click vào Button này chúng ta lấy giá trị ngày, tháng, năm của **DatePicker** và hiển thị nó thông qua **Toast**.

```
public class MainActivity extends AppCompatActivity {  
    //Khai báo các widget  
    DatePicker simpleDatePicker;  
    Button btnSubmit;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        setControl();  
        setEvent();  
    }  
    private void setControl() {  
        simpleDatePicker = (DatePicker) findViewById(R.id.simpleDatePicker);  
        btnSubmit = (Button) findViewById(R.id.btnSubmit);  
    }  
}
```

```

private void setEvent() {
    btnSubmit.setOnClickListener(new View.OnClickListener() {

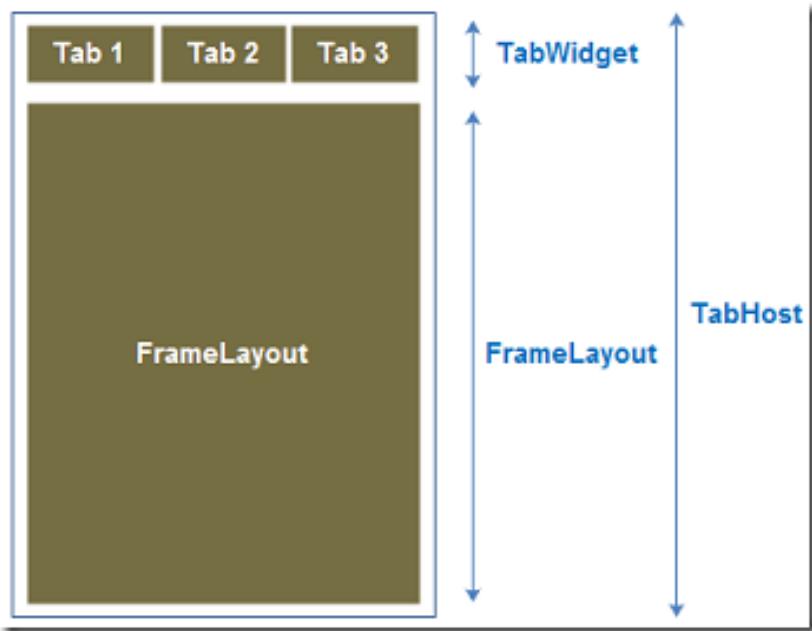
        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub
            String day = "Ngày: " + simpleDatePicker.getDayOfMonth();
            String month = "\nTháng: " + simpleDatePicker.getMonth();
            String year = "\n Năm:" + simpleDatePicker.getYear();
            Toast.makeText(MainActivity.this, day + month + year,
                    Toast.LENGTH_LONG).show();
        }
    });
}
}

```

2.5.3 | TabHost.

Tabhost trong android là một dạng giao diện điều khiển bằng thẻ tab cho phép người dùng có thể chuyển đổi các khung hình khác nhau trong cùng một giao diện Activity.

Một TabHost Layout gồm có 3 phần : TabHost, TabWidget và FrameLayout



Hình 2-41: Minh họa Tabhost

- ❖ TabHost: Là container chính chứa các Tab buttons và Tab contents
- ❖ TabWidget : Để định dạng cho các Tab buttons : Nhãn, icon, ...
- ❖ FrameLayout : là container để chứa các layout cho Tab contents. Chỉ có FrameLayout là view group được dùng cho Tab contents, không thể dùng các loại Layout khác.

2.5.3.1 / Các phương thức quan trọng của TabSpec.

- ↳ **setIndicator(CharSequence label):** Phương thức này được sử dụng để thiết lập chuỗi nhãn lên trên tab.
 - ↳ **setIndicator(CharSequence label, Drawable icon):** Phương thức này được sử dụng để thiết lập chuỗi nhãn và một icon lên trên tab
- ví dụ thiết lập một chuỗi nhãn “Tab 1” và một icon cho tab

```
TabHost tabHost = (TabHost)findViewById(android.R.id.tabhost);
TabHost.TabSpec tabSpec = tabHost.newTabSpec("tab1");
tabSpec.setIndicator("Tab 1", getResources().getDrawable(R.drawable.ic_launcher));
```

- ↳ **setContent(Intent intent):** Phương thức này thường được sử dụng mở một Activity. Click người sử dụng click vào tab sử dụng phương thức này nó sẽ mở một Activity để hiển.

Ví dụ sau chúng ta mở một activity bằng phương thức *setContent*

```
TabHost tabHost = (TabHost) findViewById(android.R.id.tabhost);
TabHost.TabSpec tabSpec = tabHost.newTabSpec("tab1");
tabSpec.setIndicator(view);
Intent intent = new Intent(this, MyActivity.class);
tabSpec.setContent(intent);
```

2.5.3.2 / **Một số phương thức quan trọng của Tabhost.**

- ↳ *.addTab(TabSpec tabSpec)*: Phương thức này được sử dụng để thêm một tab mới cho một tab widget.

Ví dụ: Xác định một tab bằng cách sử dụng lớp TabSpec và sau đó thêm tab đó vào tabhost bằng cách sử dụng phương thức addTab.

```
TabHost tabHost = (TabHost) findViewById(android.R.id.tabhost);
TabHost.TabSpec tabSpec = tabHost.newTabSpec("tab1");
tabSpec.setIndicator(view);
Intent intent = new Intent(this, MyActivity.class);
tabSpec.setContent(intent);
tabHost.addTab(tabSpec);
```

- ↳ *clearAllTabs()*: Phương thức này được sử dụng xóa tất cả các tab trên TabHost

Ví dụ: Bước đầu tiên chúng ta thêm 2 tabs, bước tiếp theo chúng ta xóa các tab này

```
TabHost tabHost = (TabHost) findViewById(android.R.id.tabhost);
TabHost.TabSpec tabSpec = tabHost.newTabSpec("tab1");
tabSpec.setIndicator("Tab 1");
Intent intent = new Intent(this, MyActivity.class);
tabSpec.setContent(intent);
TabHost.TabSpec tabSpec1 = tabHost.newTabSpec("tab2");
tabSpec1.setIndicator("Tab 2");
Intent intent1 = new Intent(this, MyActivity.class);
```

```
tabSpec1.setContent(intent1  
tabHost.addTab(tabSpec);  
tabHost.addTab(tabSpec1);  
tabHost.clearAllTabs();
```

- ↳ **setCurrentTab(int index):** Phương thức này được sử dụng để thiết lập tab được chọn. Mặc định trong TabHost tab đầu tiên là tab hiện tại.

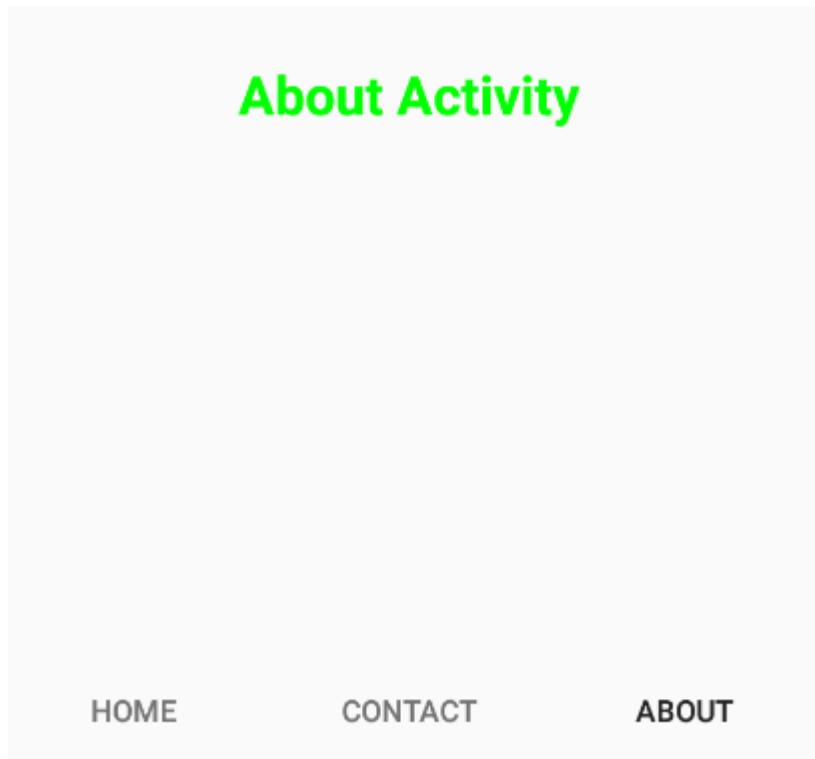
Ví dụ sau chúng ta thêm 2 tabs, sau đó thiết lập tab được chọn là tab thứ 2

```
TabHost tabHost = (TabHost) findViewById(android.R.id.tabhost);  
TabHost.TabSpec tabSpec = tabHost.newTabSpec("tab1");  
tabSpec.setIndicator("Tab 1");  
Intent intent = new Intent(this, MyActivity.class);  
tabSpec.setContent(intent);  
TabHost.TabSpec tabSpec1 = tabHost.newTabSpec("tab2");  
tabSpec1.setIndicator("Tab 2");  
Intent intent1 = new Intent(this, MyActivity.class);  
tabSpec1.setContent(intent1);  
tabHost.addTab(tabSpec);  
tabHost.addTab(tabSpec1);  
tabHost.setCurrentTab(1);
```

- ↳ **setOnTabChangedListener(OnTabChangeListener):** Phương thức này được sử dụng khi một tab thay đổi

```
tabHost.setOnTabChangedListener(new TabHost.OnTabChangeListener() {  
    @Override  
    public void onTabChanged(String tabId) {  
        // Add code Here  
    }  
});
```

Ví dụ: xây dựng ứng dụng sử dụng tabhost dùng để hiển thị 3 tab : Home, Contact, About



Hình 2-42: Ví dụ Tabhost

Bước 1: Tạo một project tên là DemoTabHost: **File → New → Android Application Project** điền các thông tin → **Next →Finish**.

Bước 2: Mở **res → layout → xml** tạo layout

➤ activity_main.xml

```
<TabHost xmlns:android="http://schemas.android.com/apk/res/android"  
    android:id="@+id/tabhost"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
    <LinearLayout  
        android:layout_width="fill_parent"  
        android:layout_height="fill_parent"  
        android:orientation="vertical">  
        <FrameLayout  
            android:id="@+id/tabcontent"  
            android:layout_width="fill_parent"  
            android:layout_height="0dip"  
            android:layout_weight="1" />  
        <TabWidget  
            android:id="@+id/tabs"  
            android:layout_width="fill_parent"  
            android:layout_height="wrap_content"  
            android:layout_marginBottom="-4dp"  
            android:layout_weight="0" />  
    </LinearLayout>  
</TabHost>
```

☞ **activity_about.xml**

```
<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" >  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_centerInParent="true"  
        android:textSize="25sp"  
        android:textColor="#0f0"  
        android:textStyle="bold"  
        android:text="About Activity" />  
    </RelativeLayout>  
    &lt; activity_contact.xml  
    <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">  
        <TextView  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:layout_centerInParent="true"  
            android:text="Contact Activity"  
            android:textColor="#00f"  
            android:textSize="25sp"  
            android:textStyle="bold" />  
        </RelativeLayout>  
        &lt; activity_home.xml
```

```
<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">  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_centerInParent="true"  
        android:text="Home Activity"  
        android:textColor="#f00"  
        android:textSize="25sp"  
        android:textStyle="bold" />  
    </RelativeLayout>
```

Bước 3: Mở app → src và thêm code.

¤ MainActivity.java

```
public class MainActivity extends TabActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        TabHost tabHost = (TabHost) findViewById(android.R.id.tabhost);  
        TabSpec spec;  
        Intent intent;  
        spec = tabHost.newTabSpec("home");  
        spec.setIndicator("HOME");  
        intent = new Intent(this, HomeActivity.class);  
        spec.setContent(intent);  
        tabHost.addTab(spec);  
        spec = tabHost.newTabSpec("Contact");  
        spec.setIndicator("CONTACT");
```

```

intent = new Intent(this, ContactActivity.class);
spec.setContent(intent);
tabHost.addTab(spec);
spec = tabHost.newTabSpec("About");
spec.setIndicator("ABOUT");
intent = new Intent(this, AboutActivity.class);
spec.setContent(intent);
tabHost.addTab(spec);
tabHost.setCurrentTab(1);
tabHost.setOnTabChangedListener(new TabHost.OnTabChangeListener() {
    @Override
    public void onTabChanged(String tabId) {
        // display the name of the tab whenever a tab is changed
        Toast.makeText(getApplicationContext(), tabId,
        Toast.LENGTH_SHORT).show();
    }
});
}

```

☞ AboutActivity.java.

```

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

```

ContactActivity.java.

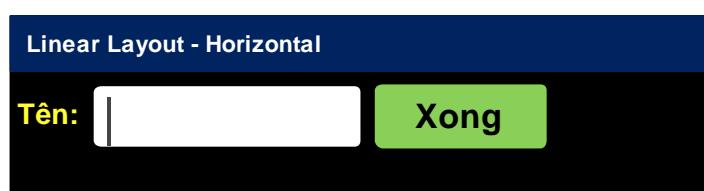
```
public class ContactActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_contact);  
    }  
}
```

HomeActivity.java

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

2.6 | Bài tập Chương 2

Bài tập 2.1. Sử dụng Linear Layout thiết kế giao diện như sau:





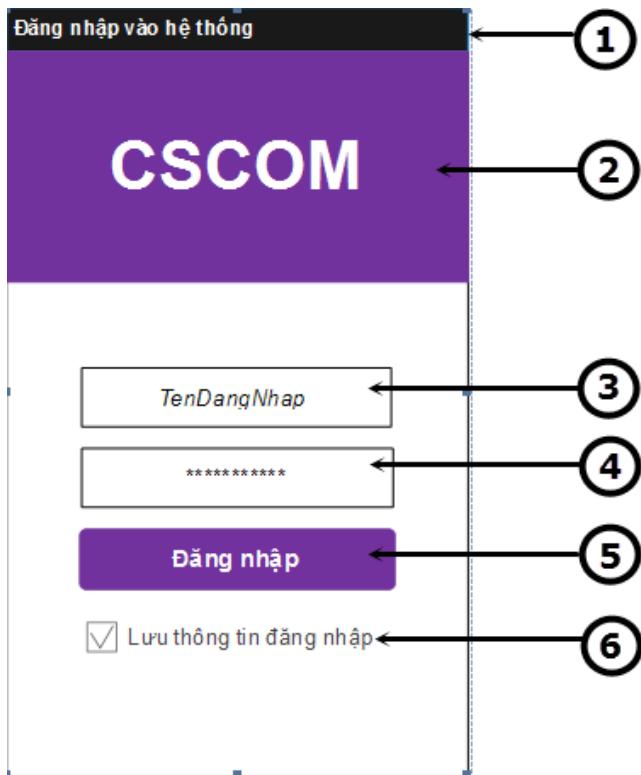
Mục tiêu:

- Biết cách sử dụng các loại tài nguyên giao diện phù hợp.
- Tạo được các giao diện theo yêu cầu.

Gợi ý thực hiện:

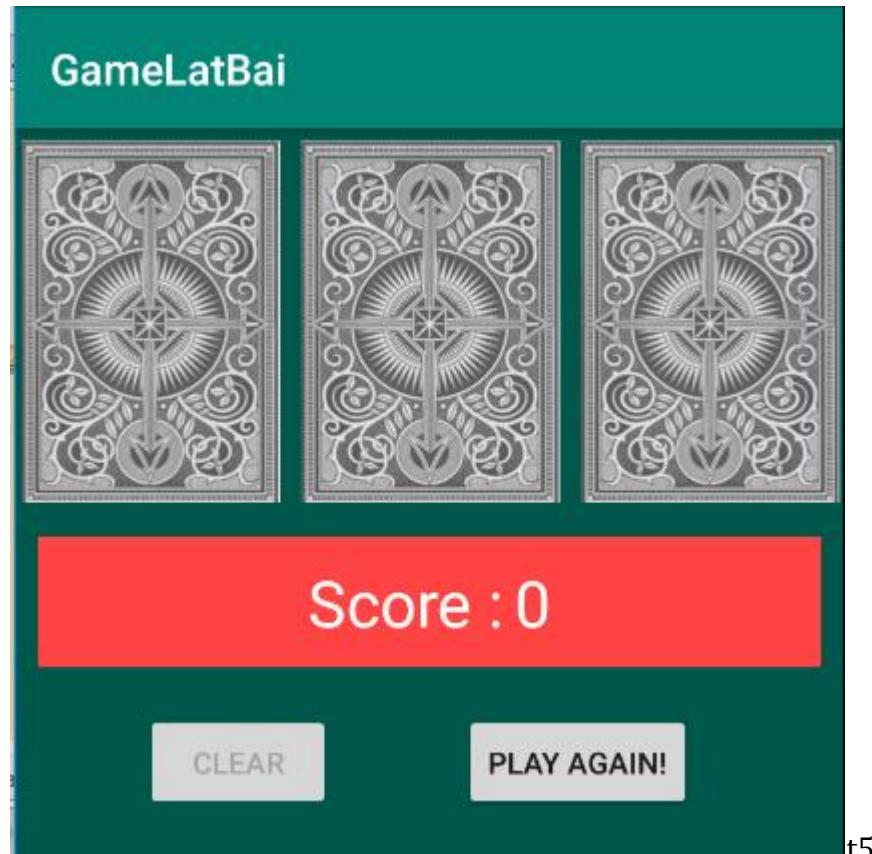
- Tạo 1 Android Application Project.
- Tạo Layout và kết nối Layout vào Activity.
- Đặt các điều khiển lên Layout vừa tạo và thiết lập, tùy chỉnh các thông số của các Layout để được giao diện như ý muốn.

Bài tập 2.2. Xây dựng màn hình đăng nhập

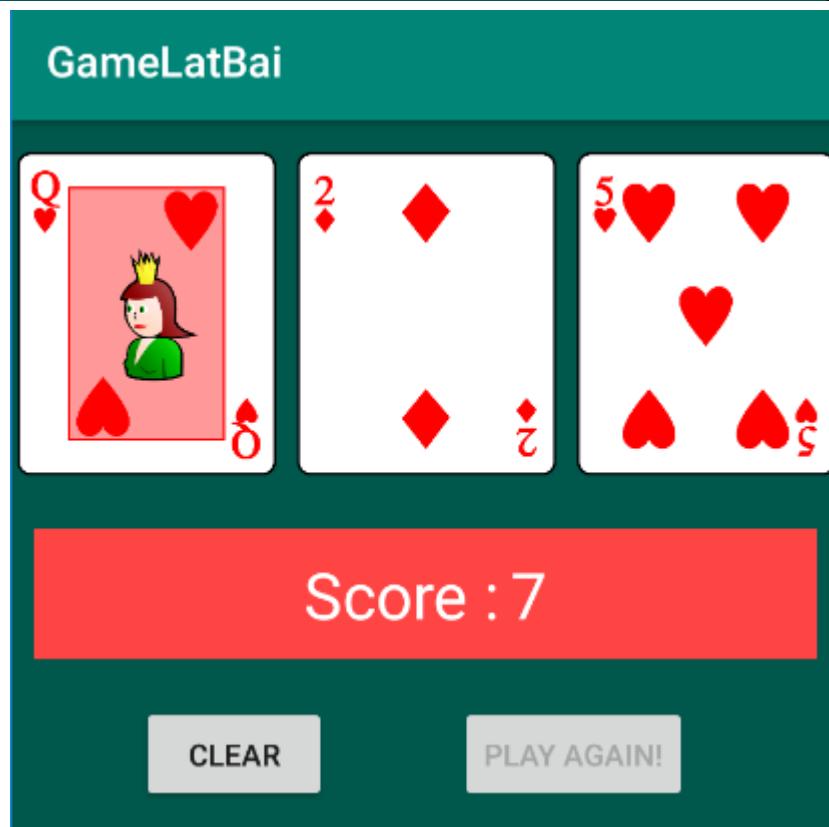


- Trong đó:
 - o Số 1 và số 2: tiêu đề và tên màn hình - sử dụng TextView để tạo.
 - o Số 3: dùng để nhập tên đăng nhập – sử dụng EditText để tạo và đặt tên id là edt_TenDangNhap.
 - o Số 4: dùng để nhập mật khẩu – sử dụng EditText để tạo và đặt tên id là edt_MatKhau.
 - o Số 5: khi người dùng chọn vào nút này, nó sẽ xử lý thông tin đăng nhập mà người dùng đã nhập ở số 3 và 4 – sử dụng Button để tạo và đặt tên id là btn_DangNhap.
 - o Số 6: nếu người dùng bấm chọn phần này, thông tin đăng nhập sẽ được lưu cho những lần sau, không cần phải nhập thông tin đăng nhập lại – sử dụng Checkbox để tạo và đặt tên id là chkb_LuuDangNhap.
- Nhưng ở bài tập này, chúng ta chỉ dùng lại ở việc thiết kế giao diện theo yêu cầu và thiết lập các thuộc tính cho các điều khiển.

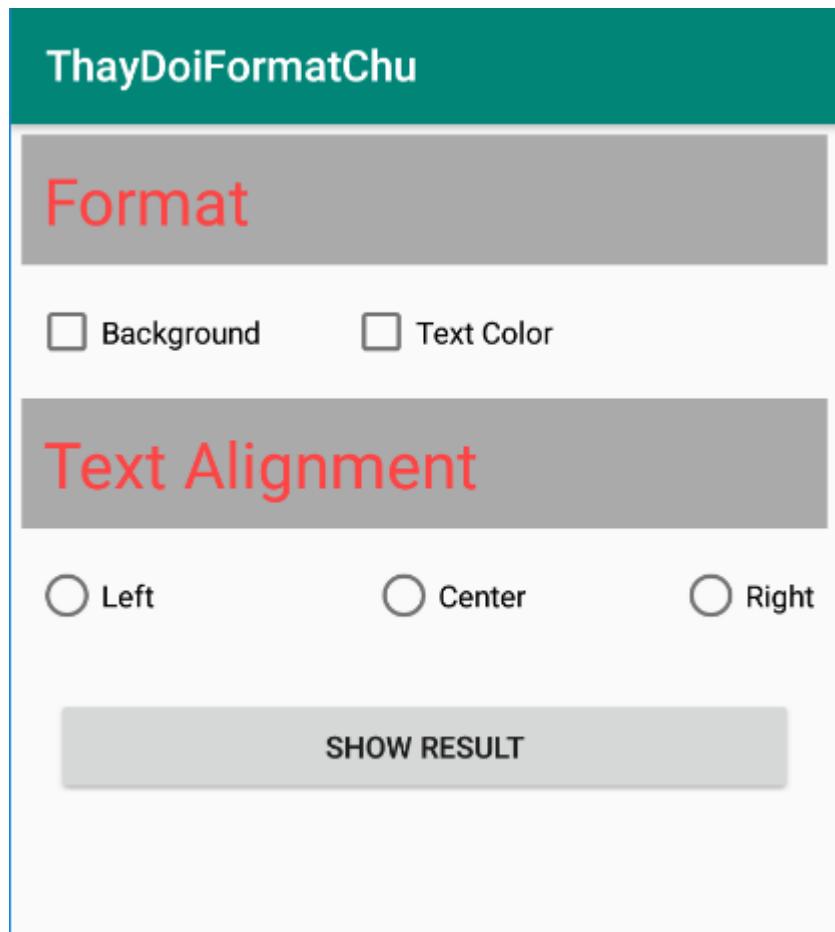
Bài tập 2.3. Lật bài và tính điểm.



t5



Bài tập 2.4. Thay đổi font chữ.



ThayDoiFormatChu

Format

Background Text Color

Text Alignment

Left Center Right

SHOW RESULT

TDC FIT

Bài tập 2.5. Thay đổi hình.

ThayDoiHinhAnh

- Pig
- Cat
- Dog
- Rabbit
- tdc



Bài tập 2.6. Thu thập thông tin cá nhân.

Thông tin cá nhân TDC

Bạn nhập tên đi

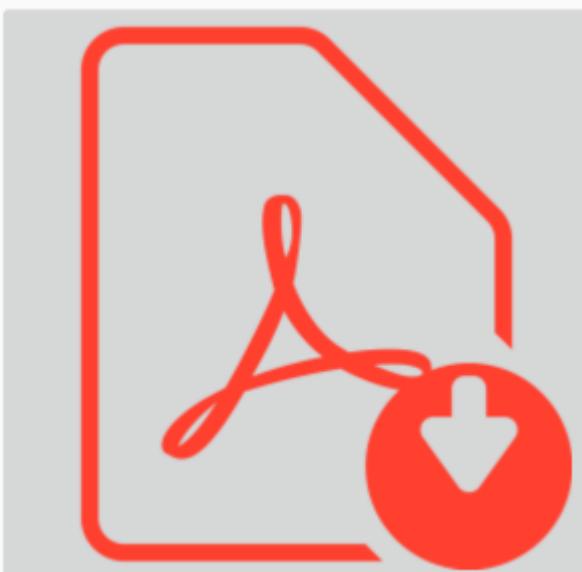
Nam Nữ

Yêu màu tím

Thích màu hồng

Sống nội tâm

Hay Khóc Thầm



Thông tin cá nhân TDC

Nguyễn Văn A

Nam Nữ

Yêu màu tím

Thích màu hồng

Sống nội tâm

Hay Khóc Thầm

Họ tên: Nguyễn Văn A
Giới tính :Nam
Sở thích:
Yêu màu tím
Sống nội tâm

Bài tập 2.7. Thông tin cá nhân.

ThongTinCaNhan TDC

Thông Tin Cá Nhân

Họ tên: Tên phải >= 3 ký tự

CMND: Nhập đúng 9 chữ số

Ngày sinh: Nhập ngày sinh

Nơi sinh: Nhập nơi sinh

Bằng cấp

Trung Cấp Cao Đẳng Đại Học

Sở thích

Đọc Báo Đọc Sách Đọc coding

Thông tin bổ sung

Nhập thông tin bổ sung - có thể để trống

GỬI THÔNG TIN

ThongTinCaNhan TDC

Thông Tin Cá Nhân

Họ tên: Nguyễn Văn A

CMND: 123456789

Thông tin cá nhân

Nguyễn Văn A

123456789

Cao Đẳng

Đọc Sách

Thông tin bổ sung:

Du lịch

ĐÓNG

GỬI THÔNG TIN

Bài tập 2.8. Xây dựng trắc nghiệm.

Trắc Nghiệm TDC

2019

Sat, Jun 8

June 2019						
S	M	T	W	T	F	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

ĐĂNG NHẬP **THOÁT**

Trắc Nghiệm TDC

$5 * 10 = ?$

- 10 20
- 40 50

< | RESULT | >

Trắc Nghiệm TDC

Chọn tất cả con vật !!!

- Chó Ô Tô
- Bò Máy bay

< **RESULT** >

Trắc Nghiệm TDC

Hãy chọn đáp án đúng !!!

$1 + 2 = \underline{1} \quad \downarrow$

$1 + 3 = \underline{1} \quad \downarrow$

$2 + 2 = \underline{1} \quad \downarrow$

<

RESULT

>

Trắc Nghiệm TDC

Câu trả lời nào là một con vật ?

Chó

SAI

Ô tô

SAI

Cây

SAI

< | RESULT | >

Trắc Nghiệm TDC

Câu trả lời nào là một con vật ?

- Chó SAI
- Ô tô SAI
- Cây SAI

< **RESULT** >

Trắc Nghiệm TDC

Kết quả của bạn

Câu 1: 0 câu đúng

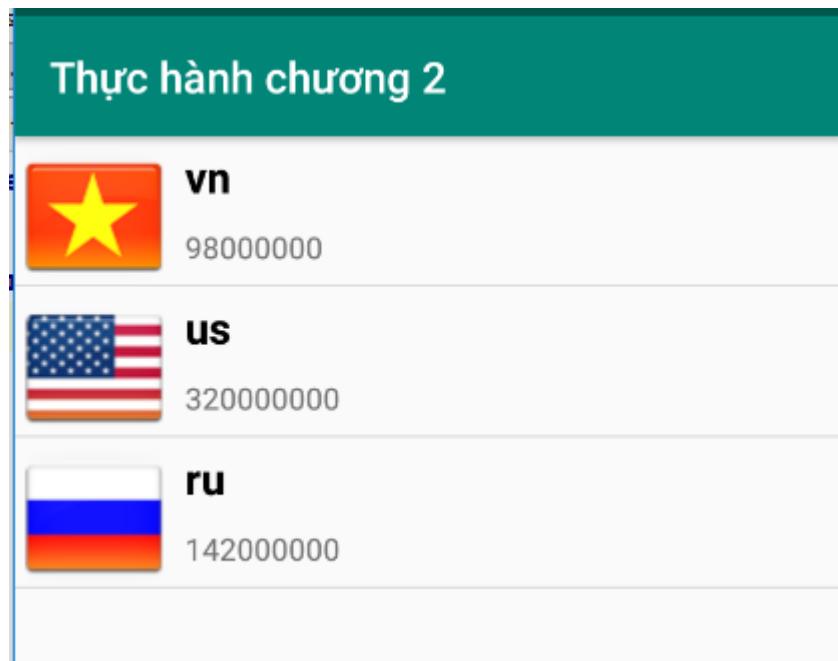
Câu 2: 0 câu đúng

Câu 3: 0 câu đúng

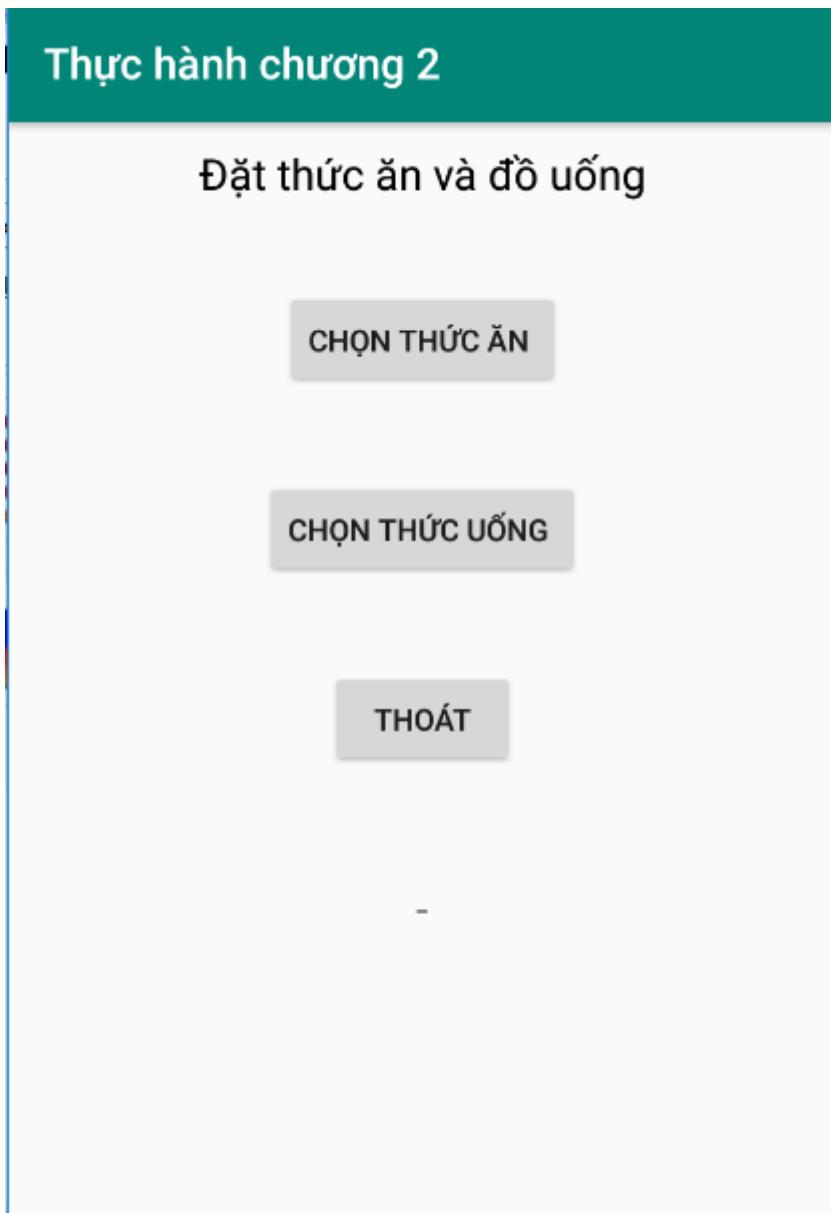
Câu 4: 0 câu đúng

Câu 5: 0 câu đúng

Bài tập 2.9. Xây dựng listview country.



Bài tập 2.10. Xây dựng chương trình đặt đồ ăn.



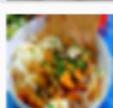
Thực hành chương 2



Bún bò



Cháo



Bánh canh



Phở

Thực hành chương 2



Cô ca cô la



Pepsi



Nước lọc



Bia Heniken

Thực hành chương 2

Đặt thức ăn và đồ uống

CHỌN THỨC ĂN

CHỌN THỨC UỐNG

THOÁT

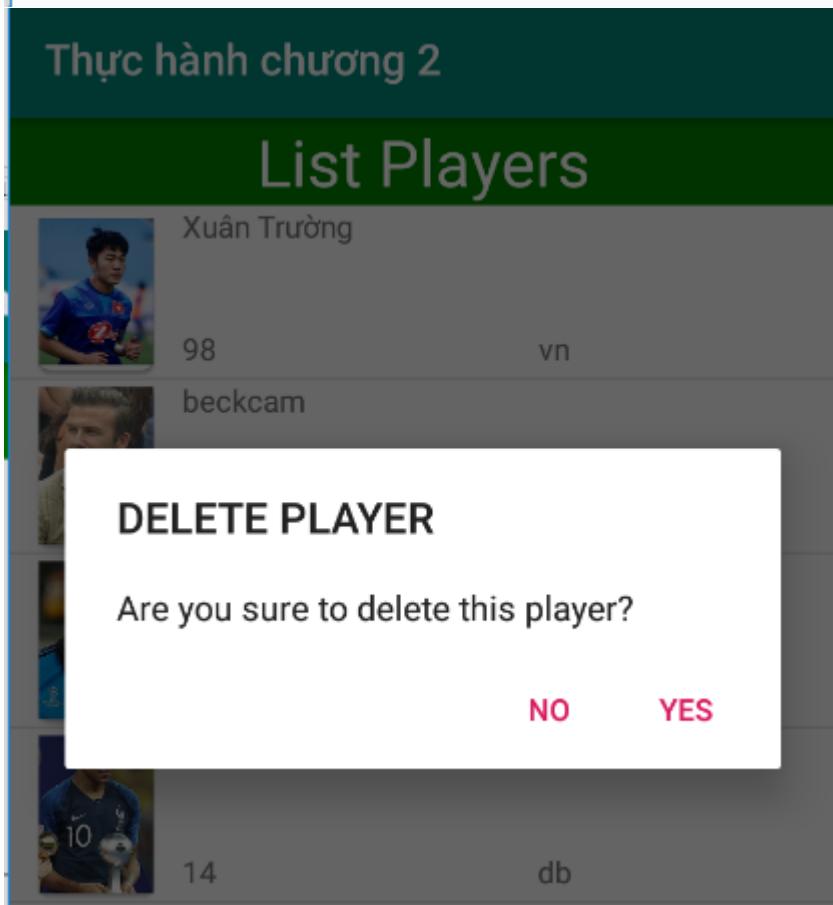
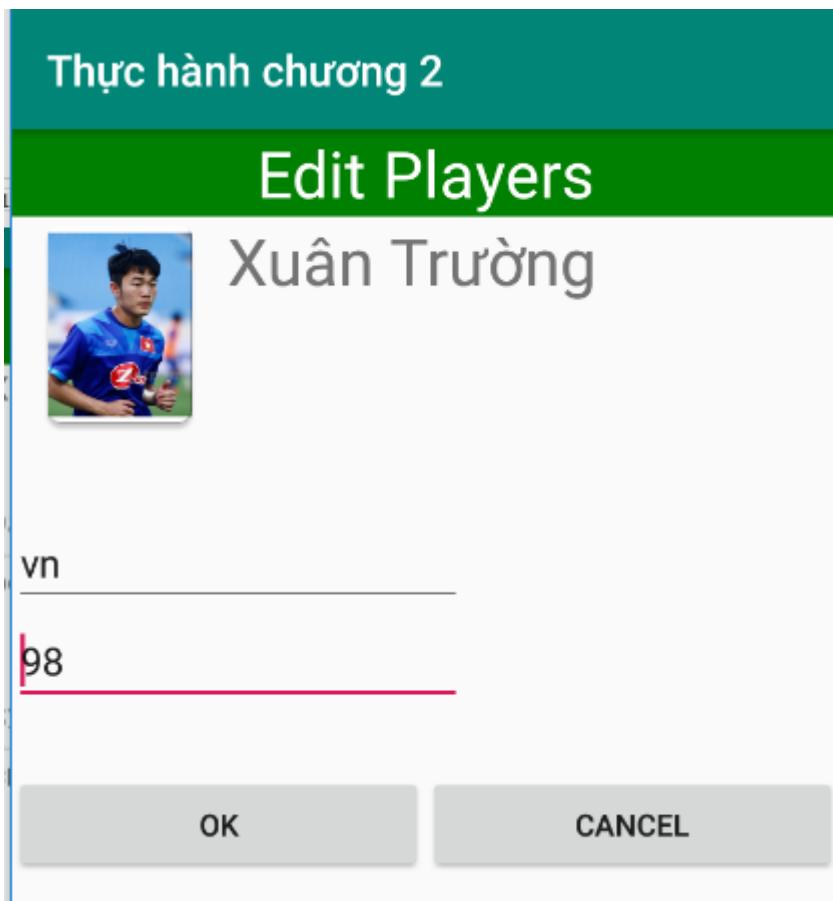
Bún bò - Pepsi

Bài tập 2.11. Xây dựng chương trình danh sách cầu thủ.

Thực hành chương 2		
	Xuan Truong 28/4/1995 (23 age)	
	David Beckham 2/5/1975 (43 age)	
	Cris Ronaldo 5/2/1985 (34 age)	
	Kylian Mbappe 20/12/1998 (20 age)	

Bài tập 2.12. Xây dựng chương trình hiển thị danh sách cầu thủ và cập nhật danh sách.

Thực hành chương 2		
List Players		
	Xuân Trường	
	98	vn
	beckcam	
	320	fc
	cr7	
	142	db
	mbabe	
	14	db



Bài tập 2.13. Hiển thị danh sách film và cập danh sách film.

Thực hành chương 2

List Films

	Chú chó nhỏ <i>(Bichon là chú chó thông minh, vui vẻ, đáng yêu, giàu tình cảm và chúng cũng rất cần được yêu thương. Chúng rất thích gần gũi)</i>
	Chú mèo to <i>(Sau khi chủ nhân ban đầu qua đời, Bronson bị bỏ rơi tại West Michigan Humane Society. Kích thước của chú mèo khiến mọi</i>

Thực hành chương 2

Edit Films

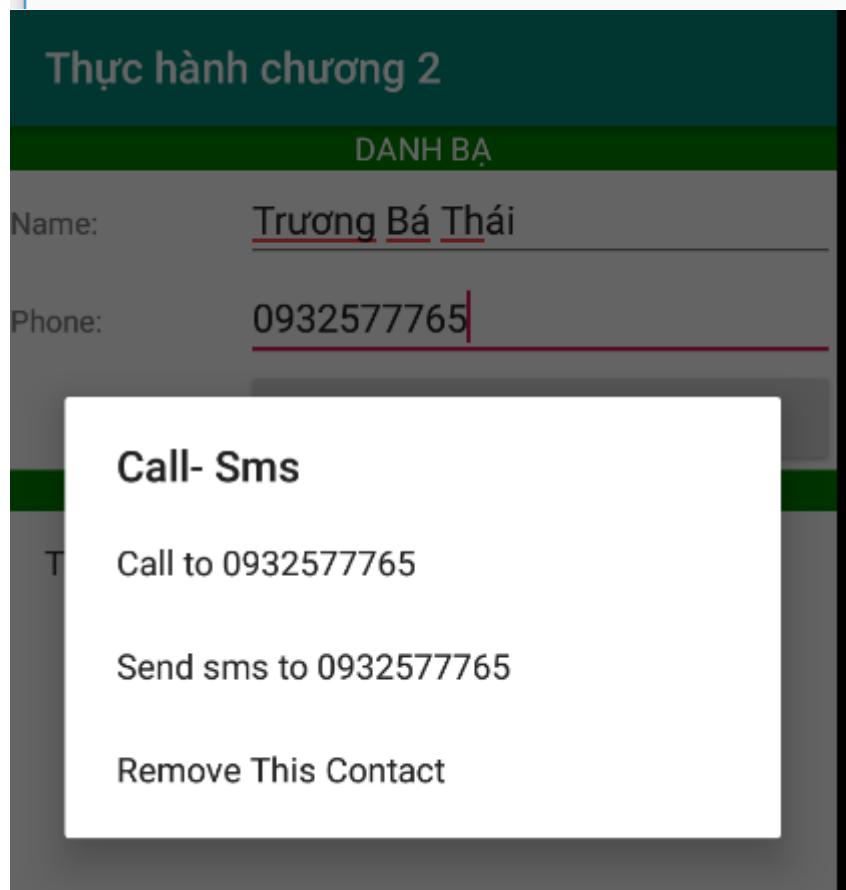
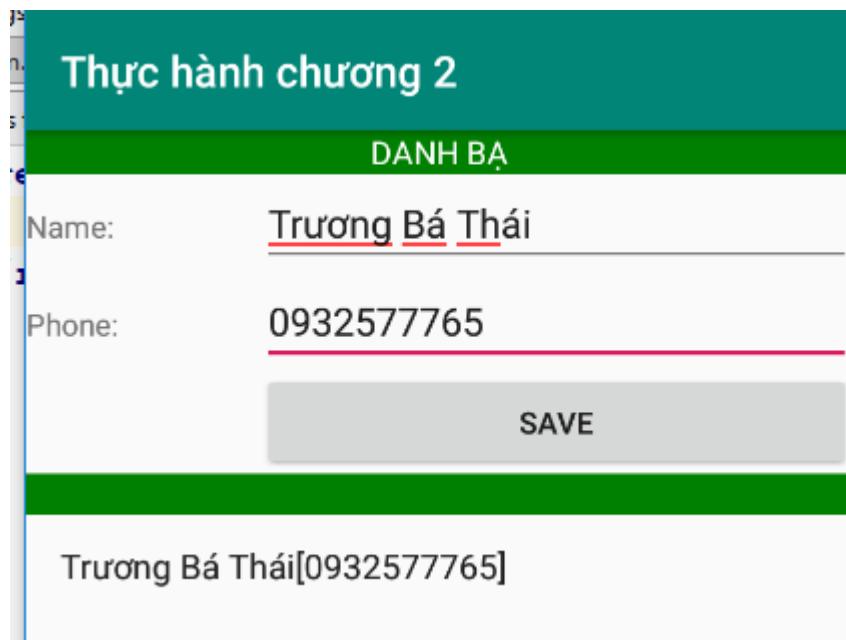
Chú chó nhỏ

Bichon là chú chó thông minh, vui vẻ, đáng yêu, giàu tình cảm và chúng cũng rất cần được yêu thương. Chúng rất thích gần gũi, thân thiện

Score 

OK **CANCEL**

Bài tập 2.14. Xây dựng chương trình quản lý danh bạ.



CHƯƠNG 3. Xây dựng giao diện với Fragment .

MỤC TIÊU THỰC HIỆN

- Trình bày được các khái niệm cơ bản của fragment
- Xây dựng hệ thống điều hướng cho ứng dụng Android sử dụng fragment

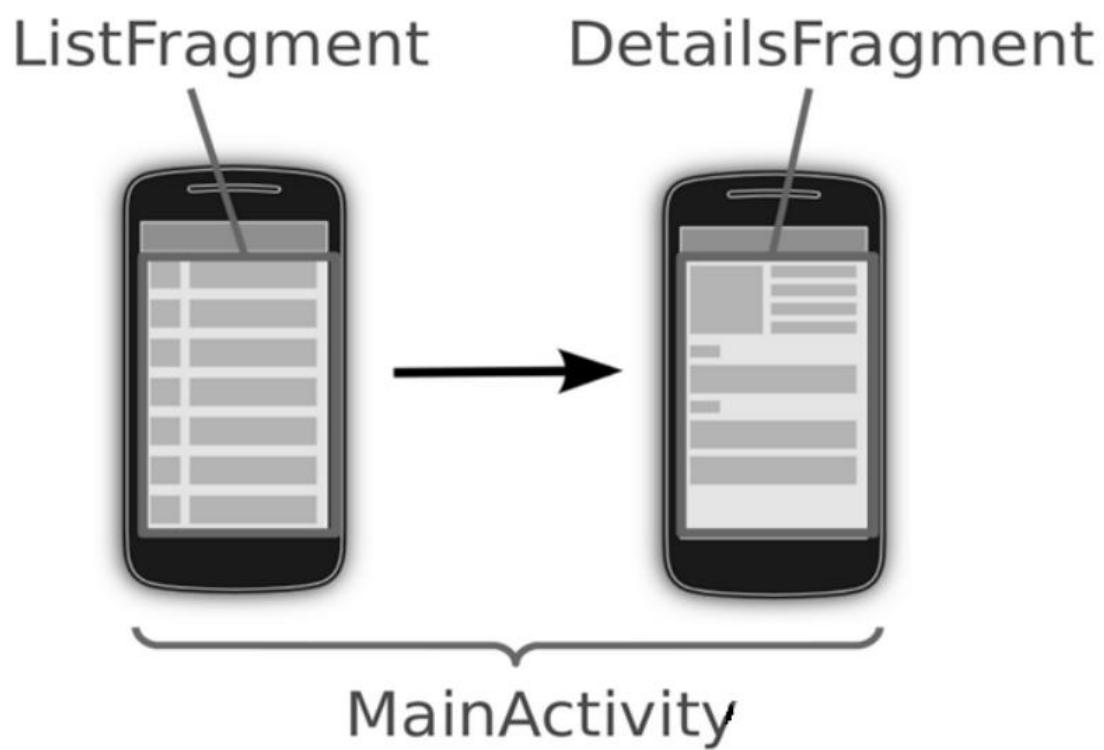
3.1 | Các khái niệm cơ bản.

3.1.1 | Fragment và phiên bản hỗ trợ.

Fragment được giới thiệu trong phiên bản Android 3.0, tuy nhiên do được hỗ trợ trong gói Android Support Library nên có thể sử dụng trong các phiên bản từ Android 1.6 trở lên.

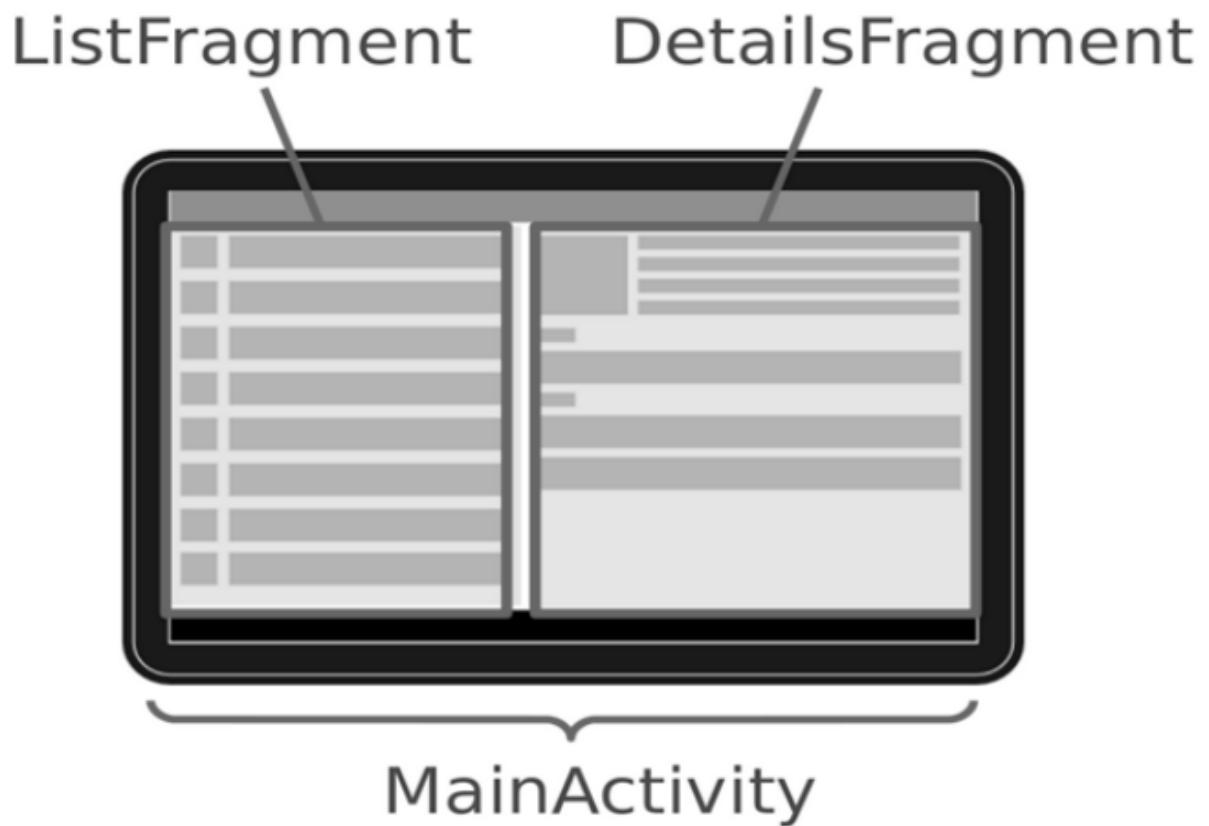
Fragment là đối tượng được nhúng trong Activity, cho phép thực hiện nhận tương tác có vòng đời riêng và thực hiện trao đổi thông tin với Activity và các Fragment khác.

Yếu tố chính trong việc sử dụng Fragment là nó có thể giúp chúng ta xây dựng các giao diện một cách chủ động và linh hoạt để có thể thích ứng với các kiểu màn hình có kích thước khác nhau từ điện thoại cho đến máy tính bảng, nói chính xác hơn là tối ưu hóa giao diện cho từng loại thiết bị, kích thước và độ phân giải



Hình 3-1: Các dạng Fragment màn hình đứng

Tính linh động của Fragment cho phép chúng ta bố cục lại các phần giao diện của một ứng dụng theo cách hợp lý nhất.



Hình 3-2: Các dạng Fragment màn hình ngang

Mỗi Fragment sẽ không bị bó buộc trong một Activity nhất định và có thể tái sử dụng nhiều lần, giúp chủ động hơn trong việc xây dựng giao diện từ việc thêm hoặc xóa các thành phần vào các cửa sổ khác nhau.

3.1.2 | Giao diện Fragment .

Để tạo mới đối tượng Fragment ta cần tạo lớp kế thừa từ lớp Fragment, khai báo các điều khiển và thực thi các chức năng.

Fragment là một thành phần android độc lập, được sử dụng bởi một activity, giống như một sub-activity. Fragment có vòng đời và giao diện riêng. Các Fragment thường có một file java đi kèm với file giao diện xml. Các fragment không có file giao diện xml thường được gọi là headless fragments.

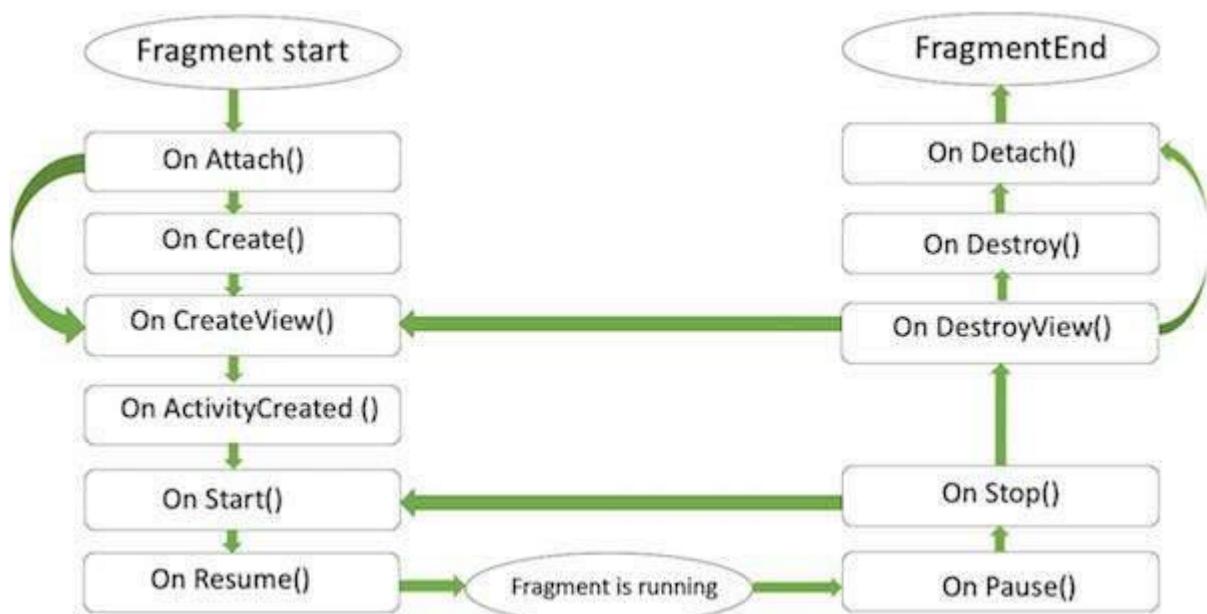
Fragment sử dụng phương thức `getActivity()` để lấy ra Activity cha

Fragment được định nghĩa trong file xml của activity (static definition) hoặc có thể sửa đổi fragment khi đang chạy (dynamic definition)

3.1.3 | Vòng đời của một Fragment .

Vòng đời của một Fragment sẽ được quản lý bởi các phương thức giống như một Activity. Chúng cũng có các phương thức khởi tạo (created), bắt đầu thực thi (started), phục hồi trạng thái (resumed), tạm dừng tương tác (paused), đóng (stopped) và hủy (destroyed). Bên cạnh đó các phương thức được gọi thực thi lại từ lớp Activity sẽ được dùng để cho biết việc kết nối và hủy kết nối Fragment đến Activity, khởi tạo và hủy khởi tạo các điều khiển trên Fragment, thông báo việc hoàn thành việc khởi tạo Activity chứa Fragment.

Fragment trong Android có vòng đời riêng của nó, tương tự như một Activity trong Android. Sơ đồ sau miêu tả ngắn gọn các giai đoạn trong vòng đời của Fragment.



Hình 3-3: Vòng đời Fragment

Vòng đời của một Fragment bắt đầu từ lúc chúng được gắn lên một Activity và kết thúc khi được gỡ bỏ. Hai sự kiện này sẽ được giám sát bởi hai phương thức tương ứng là **onAttach** và **onDetach**.

- ﴿ **onAttach()**: Sự thể hiện (instance) của Fragment được gắn kết với một sự thể hiện của activity. Fragment và Activity không hoàn toàn được khởi tạo. Đặc biệt khi lấy trong phương thức này một tham chiếu tới activity mà sử dụng Fragment cho công việc khởi tạo xa hơn.
- ﴿ **onCreate()**: Hệ thống gọi phương thức này khi tạo Fragment. nên khởi tạo các thành phần cơ bản của Fragment mà muốn duy trì khi Fragment bị dừng hoặc tạm dừng, sau đó được phục hồi lại.

- ↳ ***onCreateView()***: Hệ thống gọi phương này khi cần Fragment đó để vẽ giao diện UI lần đầu tiên. Để vẽ một UI cho Fragment của, phải trả về một thành phần View từ phương thức này. Đó là root của layout. có thể trả về null nếu Fragment không cung cấp một giao diện UI.
- ↳ ***onActivityCreated()***: Được gọi sau phương thức *onCreateView()* khi host activity được tạo. Sự thể hiện của Activity và Fragment đã được tạo cùng với cấu trúc view của activity đó. Tại điểm này, View có thể được truy cập với phương thức *findViewById()*.
- ↳ ***onStart()***
- ↳ ***onResume()***: Fragment hoạt động.
- ↳ ***onPause()***: Hệ thống gọi phương thức này khi có dấu hiệu chỉ rằng người dùng đang rời khỏi Fragment này.
- ↳ ***onStop()***: Fragment đang bị dừng bằng cách gọi phương thức này.
- ↳ ***onDestroyView()***: Fragment view sẽ hủy sau khi gọi phương thức này.
- ↳ ***onDestroy()***: Được gọi để xóa trạng thái của Fragment.

3.2 | Xây dựng và sử dụng Fragment .

3.2.1 | Thực hiện xây dựng Fragment.

Khai báo lớp kế thừa từ lớp Fragment, gọi phương thức *onCreateView* thực hiện tạo giao diện cho Fragment.

Ví dụ về tạo lớp Fragment và giao diện:

```

public class MyFragment1 extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
        super.onCreateView(inflater, container, savedInstanceState);
        View view = inflater.inflate(R.layout.fragment1, container, false);
        EditText txtFlagment = getActivity().findViewById(R.id.txtFlagment);
        TextView tvFlagment = view.findViewById(R.id.tvFlagment);
        tvFlagment.setText(txtFlagment.getText().toString());
        return view;
    }
}

```

3.2.2 | Sử dụng Fragment .

Sử dụng Fragment trong Activity, bao gồm hai cách:

- ↳ Thực hiện tham chiếu Fragment từ giao diện XML của Activity (**Static Fragment**).
- ↳ Khai báo đối tượng FragmentManager, cho phép nhúng Fragment vào Activity từ JavaCode.(**dynamic definition**)

3.2.2.1 / *Static Fragment.*

Static Fragment là kiểu fragment được khai báo (định nghĩa) trực tiếp trong file **activity_main.xml**

Trong file **activity_main.xml** khai báo tĩnh

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <fragment
        android:id="@+id/listFragment"
        class="vn.edu.tdc.chuong3vidu.MyFragment1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"></fragment>
</LinearLayout>

```

- ☞ Thuộc tính **class** chỉ đến đường dẫn chứa file java tương ứng.
- ☞ Bắt buộc phải có thuộc tính **android:id**.
- ☞ Class **MyFragment1.java** phải **extends Fragment** và phải ghi đè phương thức **onCreateView**

```

public class MyFragment1 extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
        super.onCreateView(inflater, container, savedInstanceState);
        View view = inflater.inflate(R.layout.fragment1, container, false);
        return view;
    }
}

```

Ví dụ 20. Xây dựng ứng dụng hiển thị 2 Fragment



Hình 3-4: Ví dụ xây dựng fragment tĩnh

Bước 1: Tạo một project tên là Flagment: **File → New → Android Application Project** điền các thông tin → **Next →Finish**.

Bước 2: Mở res → layout → xml

fragment1.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical" android:layout_width="match_parent"  
    android:background="#FF0000"  
    android:layout_height="match_parent">  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:id="@+id/tvFlagment"  
        android:text="fragment 1"  
        android:textSize="30dp"  
        android:layout_gravity="center"  
    />  
</LinearLayout>
```

fragment2.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical" android:layout_width="match_parent"  
    android:background="#AA0099"  
    android:layout_height="match_parent">  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:id="@+id/tvFlagment"  
        android:text="fragment 2"  
        android:textSize="30dp"  
        android:layout_gravity="center"  
    />  
/</LinearLayout>
```

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:orientation="horizontal"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
    <fragment  
        android:id="@+id/listFragment"  
        class="vn.edu.tdc.chuong3vidu.MyFragment1"  
        android:layout_width="0dp"  
        android:layout_height="match_parent"  
        android:layout_weight="1"  
        tools:layout="@layout/fragment1">  
    </fragment>
```

```
<fragment
    android:id="@+id/detailFragment"
    class='vn.edu.tdc.chuong3vidu.MyFragment2'
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="1"
    tools:layout="@layout/fragment2">
</fragment>
</LinearLayout>
```

Bước 3: Mở app → src -> thêm code.

MyFragment1.java

```
public class MyFragment1 extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
        super.onCreateView(inflater, container, savedInstanceState);
        View view = inflater.inflate(R.layout.fragment1, container, false);
        EditText
        return view;
    }
}
```

MyFragment2.java

```

public class MyFragment1 extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
        super.onCreateView(inflater, container, savedInstanceState);
        View view = inflater.inflate(R.layout.fragment1, container, false);
        EditText
        return view;
    }
}

```

☞ *MainActivity.java.*

```

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

```

3.2.2.2 / Dynamics Fragment

Lớp **FragmentManager** cho phép thêm, xóa, thay thế fragment trong layout của activity. Sử dụng phương thức **getFragmentManager()** hoặc **getSupportFragmentManager()** để lấy ra một đối tượng FragmentManager.

Việc sửa đổi phải được thực hiện trong một giao dịch thông qua lớp **FragmentTransaction**

Để làm như thế, thông thường, chúng ta định nghĩa một **FrameLayout** giữ chỗ trong file layout, sau đó dùng FragmentManager để ghi đè một fragment vào FrameLayout giữ chỗ đó

Mỗi Activity sẽ có một Fragment Manager dùng để quản lý các Fragment chứa trong nó. Để sử dụng ta cần gọi phương thức **getFragmentManager**

```
android.support.v4.app.FragmentManager fm = getSupportFragmentManager();
```

Đối tượng này sẽ cung cấp một số phương thức như thêm, gỡ bỏ và cập nhật Fragment vào Activity, quản lý các phiên làm việc của Fragment...

Cách đơn giản nhất để thêm một vào một Activity là khai báo thẻ fragment trong tập tin giao diện củ Activity đó. Fragment được xem như một đối tượng View Group khi được gắn vào trong Layout. Thường chúng ta sẽ sử dụng cách này để định nghĩ một tập các giao diện tĩnh cho các màn hình có kích cỡ khác nhau.

Trong trường hợp ta cần thay đổi giao diện dựa trên trạng thái của ứng dụng thì cách tốt nhất ta nên sử dụng các giao diện động dựa trên **FragmentTransaction**.

Lớp FragmentTransaction hỗ trợ chúng ta trong việc thêm, gỡ bỏ hoặc thay thế các Fragment trên Activity ngay khi ứng dụng đang thực thi. Dựa vào các tương tác của người dùng mà ta có thể xây dựng các giao diện hợp lý. Ngoài ra, để giúp cho ứng dụng có thể đáp ứng các trải nghiệm người dùng, các chuyển hoạt cũng được sử dụng giữa các phiên làm việc của Fragment Transaction và các Transaction sẽ được lưu trữ trong stack để tái sử dụng.

Mỗi phiên làm việc của FragmentTrasaction được khởi tạo bằng phương thức beginTransaction. Các hoạt động của Fragment sẽ được định nghĩa bằng các phương thức *add*, *remove* hoặc *replace*. Sau đó chúng ta có thể tùy chỉnh các chuyển hoạt hoặc cách thức hoạt động trong stack trước khi thực hiện phương thức *commit* để thêm Transaction vào hàng đợi.

```
android.support.v4.app.FragmentTransaction ft_add = fm.beginTransaction();  
ft_add.add(R.id.frame_layout, new MyFragment1());  
ft_add.commit();
```

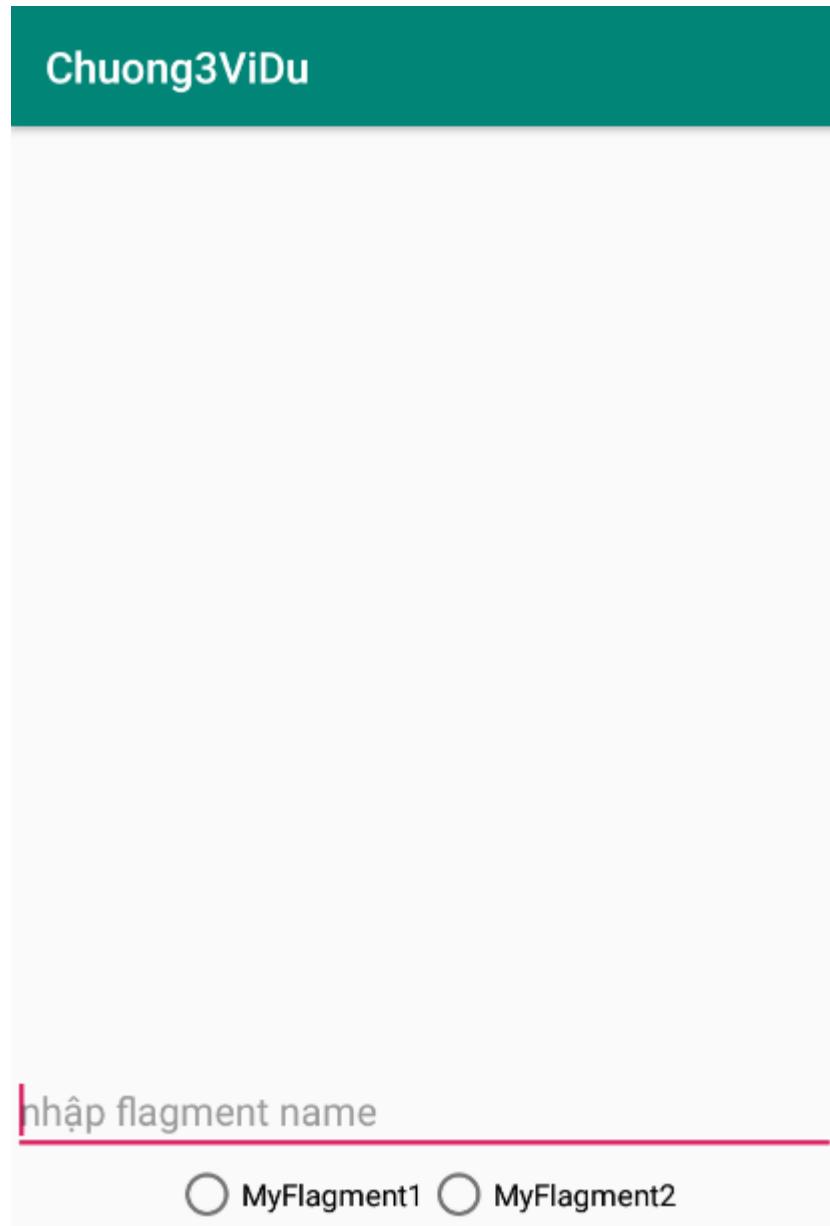
Để thực hiện các chuyển hoạt cho các Fragment trong một Transaction ta có 2 cách, một là dùng phương thức **setTransaction** và truyền vào các tham số có sẵn trong lớp **FragmentTransaction**.

```
ft_add.setCustomAnimations(android.R.animator.fade_in, android.R.animator.fade_out);
```

Trong lớp Fragment hộ trợ phương thức getActivity cho phép chúng có thể truy cập vào các đối tượng của Activity mà chúng được nhúng vào. Điều này giúp chúng ta có thể lấy về các View mà ta cần tương tác ngay trong lớp Fragment.

```
public class MyFragment1 extends Fragment {  
  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  
    Bundle savedInstanceState) {  
        super.onCreateView(inflater, container, savedInstanceState);  
        View view = inflater.inflate(R.layout.fragment1, container, false);  
        EditText txtFlagment = getActivity().findViewById(R.id.txtFlagment);  
        TextView tvFlagment = view.findViewById(R.id.tvFlagment);  
        tvFlagment.setText(txtFlagment.getText().toString());  
  
        return view;  
    }  
}
```

Ví dụ 21. Xây dựng ứng dụng hiển thị Fragment động



Hình 3-5: Ví dụ xây dựng Fragment động

Bước 1: Tạo một project tên là Fragment: **File → New → Android Application Project** điền các thông tin → **Next →Finish**.

Bước 2: Mở res → layout → xml

fragment1.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical" android:layout_width="match_parent"  
    android:background="#FF0000"  
    android:layout_height="match_parent">  
    <TextView
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/tvFlagment"
    android:text="fragment 1"
    android:textSize="30dp"
    android:layout_gravity="center"
/>
</LinearLayout>
```

fragment2.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:background="#AA0099"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/tvFlagment"
        android:text="fragment 2"
        android:textSize="30dp"
        android:layout_gravity="center"
    /
</LinearLayout>
```

activity_main.xml

```
<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">
```

<FrameLayout

```
    android:id="@+id/frame_layout"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"></FrameLayout>
```

```
<EditText
    android:id="@+id/txtFlagment"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="nhập flagment name" />
```

```
<RadioGroup
    android:id="@+id/radGroup"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:orientation="horizontal">
```

```
<RadioButton
    android:id="@+id/radMyFlagment1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="MyFlagment1" />
```

```
<RadioButton
    android:id="@+id/radMyFlagment2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="MyFlagment2" />
```

```
</RadioGroup>
```

```
</LinearLayout>
```

Bước 3: Mở app → src -> thêm code.

MyFragment1.java

```
public class MyFragment1 extends Fragment {  
  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  
    Bundle savedInstanceState) {  
        super.onCreateView(inflater, container, savedInstanceState);  
        View view = inflater.inflate(R.layout.fragment1, container, false);  
        EditText txtFlagment= getActivity().findViewById(R.id.txtFlagment);  
        TextView tvFlagment = view.findViewById(R.id.tvFlagment);  
        tvFlagment.setText(txtFlagment.getText().toString());  
  
        return view;  
    }  
}
```

MyFragment2.java

```
public class MyFragment1 extends Fragment {  
  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  
    Bundle savedInstanceState) {  
        super.onCreateView(inflater, container, savedInstanceState);  
        View view = inflater.inflate(R.layout.fragment1, container, false);  
        EditText txtFlagment= getActivity().findViewById(R.id.txtFlagment);  
        TextView tvFlagment = view.findViewById(R.id.tvFlagment);  
        tvFlagment.setText(txtFlagment.getText().toString());  
    }  
}
```

```

    return view;
}
}

>MainActivity.java
public class MainActivity extends AppCompatActivity {
    //Khai báo các widget
    RadioGroup radioGroup;
    FragmentManager fm;
    FragmentTransaction ft_add;

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

    private void setEvent() {
        fm = getSupportFragmentManager();
        radioGroup.setOnCheckedChangeListener(new
    RadioGroup.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(RadioGroup group, int checkedId) {
            if (checkedId == R.id.radMyFragment1) {
                ft_add = fm.beginTransaction();
                ft_add.setCustomAnimations(android.R.animator.fade_in,
                android.R.animator.fade_out);
                ft_add.replace(R.id.frame_layout, new MyFragment1());
                ft_add.addToBackStack(null);
                ft_add.commit();
            }
        }
    });
}

```

```

        }

        if (checkedId == R.id.radMyFlagment2) {
            ft_add = fm.beginTransaction();
            ft_add.replace(R.id.frame_layout, new MyFragment2());
            ft_add.addToBackStack(null);
            ft_add.commit();
        }
    });

}

}

```

```

private void setConTrol() {
    radioGroup = findViewById(R.id.radGroup);
}

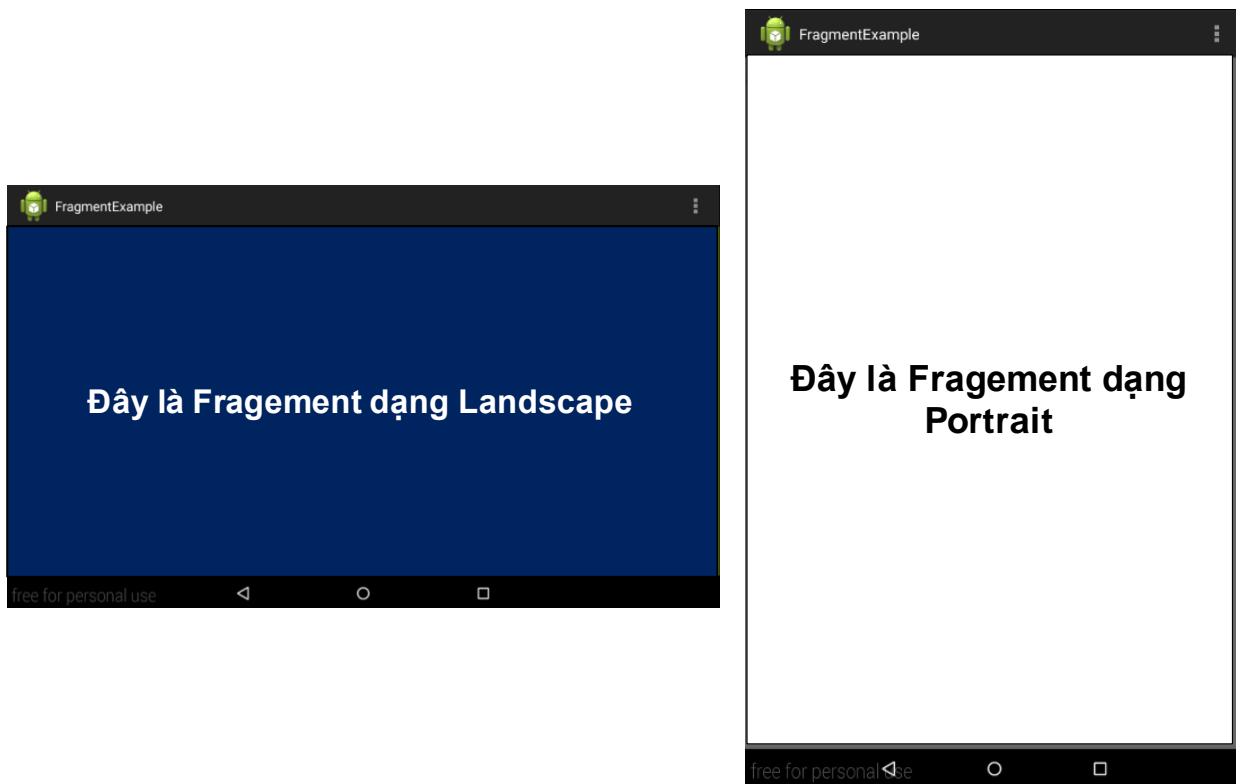
```

3.3 | Bài tập Chương 3

Bài tập 3.1. Tạo và sử dụng Fragment 1

Đề bài:

- Tạo 2 Fragment:
 - o Fragment 1 được dùng khi màn hình thiết bị nằm ngang.
 - o Fragment 2 được dùng khi màn hình thiết bị nằm dọc.
- Yêu cầu:
 - o Khi xoay màn hình thiết bị nằm ngang sẽ hiển thị giao diện như hình 1, còn khi xoay màn hình thiết bị nằm dọc sẽ hiển thị giao diện như hình 2.



Mục tiêu:

- Hiểu được khái niệm Fragment.
- Biết cách tạo và dùng Fragment.
- Ứng dụng của Fragment.

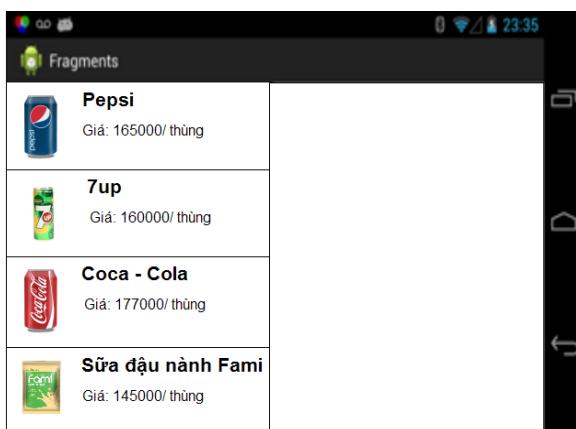
Gợi ý thực hiện:

- Tạo 1 tập tin .xml là giao diện của Fragment để trong thư 2 thư mục layout-port và layout-land.
- Trong tập tin .xml main tạo 1 ViewGroup để chứa giao diện Fragment.
- Tạo lớp kế thừa Fragment và cài đặt giao diện là tập tin .xml vừa tạo ở trên.
- Trong lớp main (.xml), để Fragment lên ViewGroup để chứa giao diện Fragment.

Bài tập 3.2. Tạo và sử dụng Fragment 2

Đề bài:

Sử dụng Fragment khi người dùng nhấn chọn vào 1 dòng trên ListView sẽ hiển thị nội dung của dòng đó



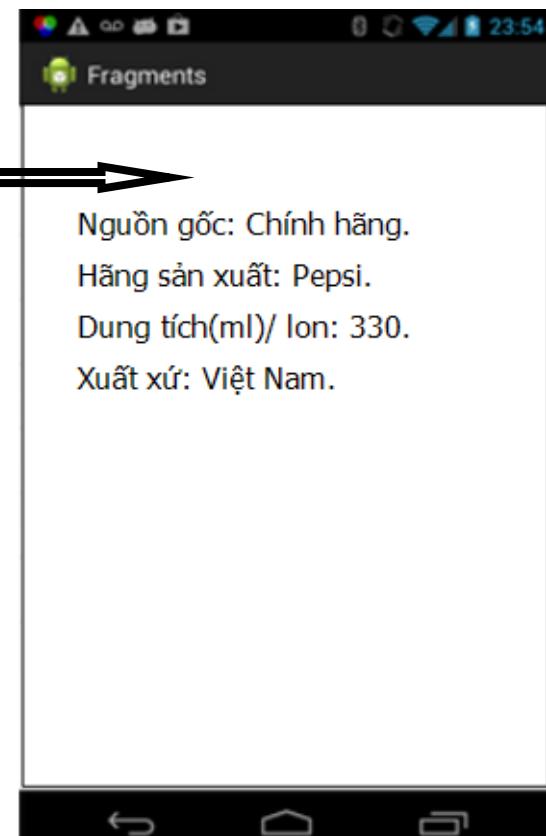
Giao diện khi thiết bị nằm ngang



Giao diện khi người dùng chọn 1 dòng trên ListView trên thiết bị nằm ngang.



Giao diện khi thiết bị nằm dọc.



Giao diện khi người dùng chọn 1 dòng trên ListView khi thiết bị nằm dọc.

Mục tiêu:

- Biết cách gửi dữ liệu giữa Activity và Fragment thông qua Interface.

Gợi ý thực hiện:

- Tạo 2 Layout cho màn hình nằm ngang và đứng. Màn hình nằm ngang thì có 1 ListView bên trái và Fragment bên phải. Còn màn hình đứng thì 2 Layout: 1 Layout chứa ListView và 1 Layout chứa Fragment.
- Tạo Interface để truyền nhận dữ liệu giữa Activity và Fragment .

CHƯƠNG 4. Action Bar - Navigation Drawers

MỤC TIÊU THỰC HIỆN

- Trình bày được các khái niệm Action bar
- Tạo được Action bar và sử dụng được các thao tác với Action bar
- Trình bày được các khái niệm Navigation Drawers
- Tạo được Action bar và sử dụng được các thao tác với Navigation Drawers

4.1 | Action bar

4.1.1 | Giới thiệu

Trong lần đầu giới thiệu chiếc Samsung Galaxy Nexus, Google đã loại bỏ nút Menu ra khỏi cụm phím điều hướng chính của hệ thống, chỉ còn lại nút Quay về (Back), nút trở về màn hình chính (Home) và nút liệt kê các ứng dụng đã chạy gần đây (Recents Apps). Nó biến thành một thứ gọi là "**Action Bar**", nơi mà người dùng có thể khởi chạy các hoạt động tùy thuộc vào ứng dụng.

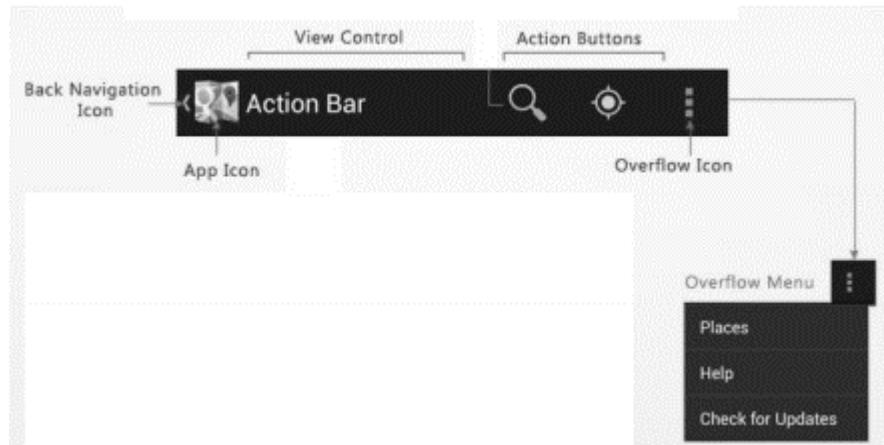
↳ Action Bar có chức năng:

- ☞ Điều hướng giao diện ứng dụng.
- ☞ Hiển thị các thao tác trên toàn bộ hệ thống ứng dụng hoặc thao tác tại màn hình hiển thị (tìm kiếm, chỉnh sửa, xóa...).
- ☞ Thao tác chuyển đến các giao diện khác nhau trong cùng một màn hình hiển thị (tabhost, list navigation...).

Những tính năng nào không thể hiện hết thì sẽ nằm trong dấu ba chấm dạng dọc ở cuối Action Bar, gọi là Action Overflow.

Các tổ hợp điều khiển trên ActionBar:

- ☞ **App Icon:** thường là logo của ứng dụng hoặc bất kỳ Icon nào mà muốn.
- ☞ **View Control:** thường hiển thị Title cho nội dung màn hình hiện tại, có thể tùy chỉnh thành một Spinner.
- ☞ **Action Button:** chứa một số Button để thực hiện chức năng quan trọng.
- ☞ **Action Overflow:** Chứa một Menu Dropdown các chức năng hiển thị.



Hình 4-1: Các hình thức của Action Bar

4.1.2 | Tạo Action Bar

Tạo một tập tin menu cho ActionBar trong thư mục **res/menu**:

main_activity_actions.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">

    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_search_black_24dp"
        android:title="action_search"
        app:showAsAction="ifRoom"
    />

    <item
        android:id="@+id/action_compose"
        android:icon="@drawable/ic_record_voice_over_black_24dp"
        android:title="action_compose"
        app:showAsAction="ifRoom"
    />

</menu>
```

Đối với mỗi thẻ item sẽ có một số thuộc tính quan trọng sau:

- ☞ **android:id** - id của Action Button.
- ☞ **android:icon** - icon hiển thị cho chức năng. o android:title - tiêu đề cho chức năng.
- ☞ **android:showAsAction** - Xác định việc hiển thị cho Action Button.

Tiếp tục, mở MainActivity ra và thực hiện kéo tập tin Menu vừa tạo vào cho ứng dụng. Chúng ta sẽ thực hiện việc này trong phương thức onCreateOptionsMenu():

```
public boolean onCreateOptionsMenu(Menu menu) {  
    // Inflate the menu items for use in the action bar  
    MenuInflater inflater = getMenuInflater();  
    inflater.inflate(R.menu.main_activity_actions, menu);  
    return super.onCreateOptionsMenu(menu);  
}
```

4.1.3 | Thao tác với Action Bar

Chúng ta sẽ kích hoạt các chức năng khi chạm vào các Action Button. Việc này sẽ được thực hiện trong onOptionsItemSelected():

@Override

```
public boolean onOptionsItemSelected(MenuItem item) {
```

// Handle presses on the action bar items

```
switch (item.getItemId()) {
```

```
    case R.id.action_search:
```

```
        Toast.makeText(getApplicationContext(), "openSearch()", Toast.LENGTH_SHORT).show();
```

```
        return true;
```

```
    case R.id.action_compose:
```

```
        Toast.makeText(getApplicationContext(), "composeMessage()", Toast.LENGTH_SHORT).show();
```

```

    HORT).show();
    return true;
case android.R.id.home:
    Toast.makeText(getApplicationContext(),"Go
home()",Toast.LENGTH_SHORT).show();
    return true;
default:
    return super.onOptionsItemSelected(item);
}
}

```

Tiếp theo, chúng ta tìm hiểu việc hiển thị và ẩn thành Action Bar. Có 2 phương thức rất đơn giản, đó là hide() và show()

```

ActionBar actionBar = getSupportActionBar();
actionBar.hide();
actionBar.show();

```

Thay đổi App Icon cho Action Bar. Chúng ta sử dụng phương thức setIcon() để thay đổi App Icon trên Action Bar

```
actionBar.setIcon(R.drawable.tdc);
```

Ví dụ 22. Xây dựng ứng dụng sử dụng ActionBar để điều hướng



Hình 4-2: Ví dụ sử dụng Action bar

Bước 1: Tạo một project tên là DemoActionBar: **File → New → Android Application Project** điền các thông tin → **Next →Finish.**

Bước 2: Vào thư mục **res/menu** tạo file **main_activity_actions.xml**

```

<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">
    <item

```

```

    android:id="@+id/action_search"
    android:icon="@drawable/ic_search_black_24dp"
    android:title="action_search"
    app:showAsAction="ifRoom"
/>
<item
    android:id="@+id/action_compose"
    android:icon="@drawable/ic_record_voice_over_black_24dp"
    android:title="action_compose"
    app:showAsAction="ifRoom"
/>
</menu>

```

Bước 3: Mở app → src -> MainActivity.java và thêm code.

```

public class MainActivity extends AppCompatActivity {

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

        ActionBar actionBar = getSupportActionBar();
        actionBar.setDisplayHomeAsUpEnabled(true);
    }
}

```

```

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu items for use in the action bar
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.main_activity_actions, menu);
    }
}

```

```

    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle presses on the action bar items
    switch (item.getItemId()) {
        case R.id.action_search:
            Toast.makeText(getApplicationContext(), "openSearch()", Toast.LENGTH_SHORT).show();
            return true;
        case R.id.action_compose:
            Toast.makeText(getApplicationContext(), "composeMessage()", Toast.LENGTH_SHORT).show();
            return true;
        case android.R.id.home:
            Toast.makeText(getApplicationContext(), "Go home()", Toast.LENGTH_SHORT).show();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}

```

4.2 | Navigation Drawers

4.2.1 | Giới thiệu

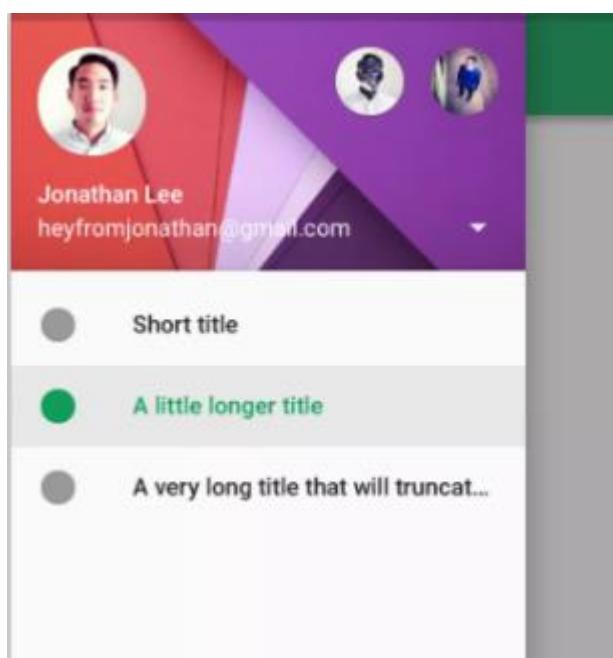
Navigation Drawer, hay vẫn gọi là Left Menu, hoặc Slide Menu. Navigation Drawer được Google giới thiệu vào năm 2013, ngay sau khi ActionBar được trình làng khoảng 2 năm. Navigation Drawer này chỉ đơn giản là một thanh menu được ẩn đi về

phía bên trái màn hình. Nó được hiển thị ra khi người dùng nhấn vào icon menu trên



Action Bar.

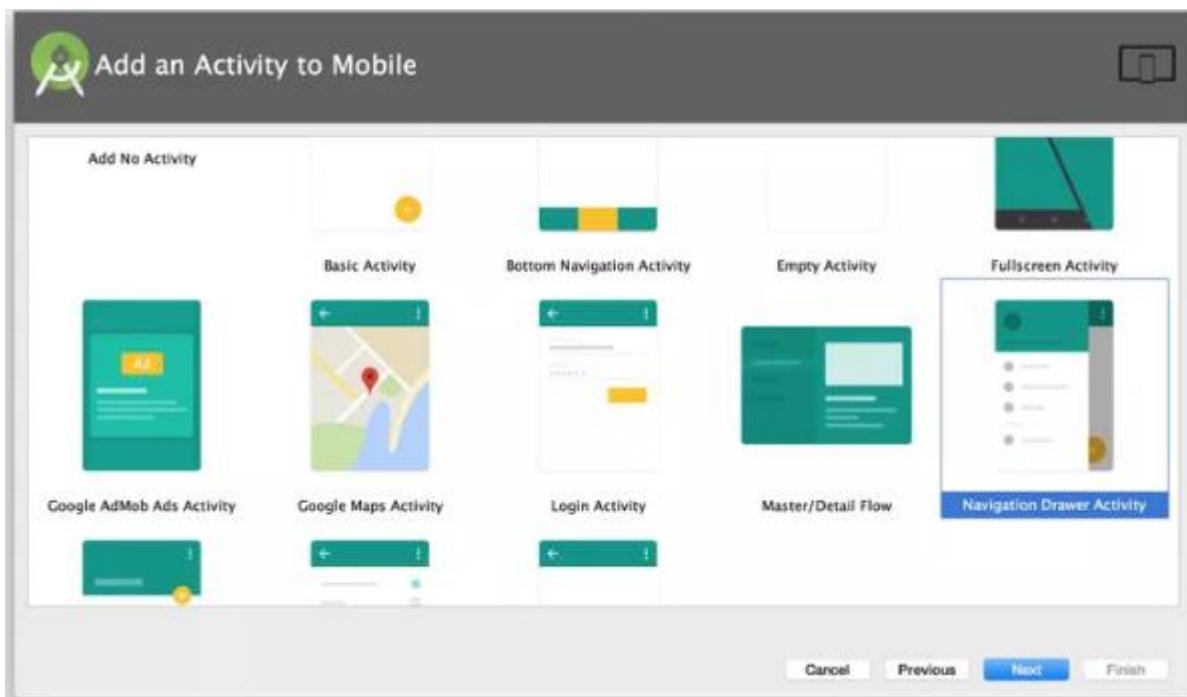
Cũng giống như ActionBar, Navigation Drawer mong muốn đem lại cho user một trải nghiệm rõ ràng hơn trên các ứng dụng phức tạp. Khi đó các chức năng chính của ứng dụng dường như được để hết cả lên thanh này, Navigation Drawer này mới chính thức là “xương sống” rõ ràng nhất cho ứng dụng.



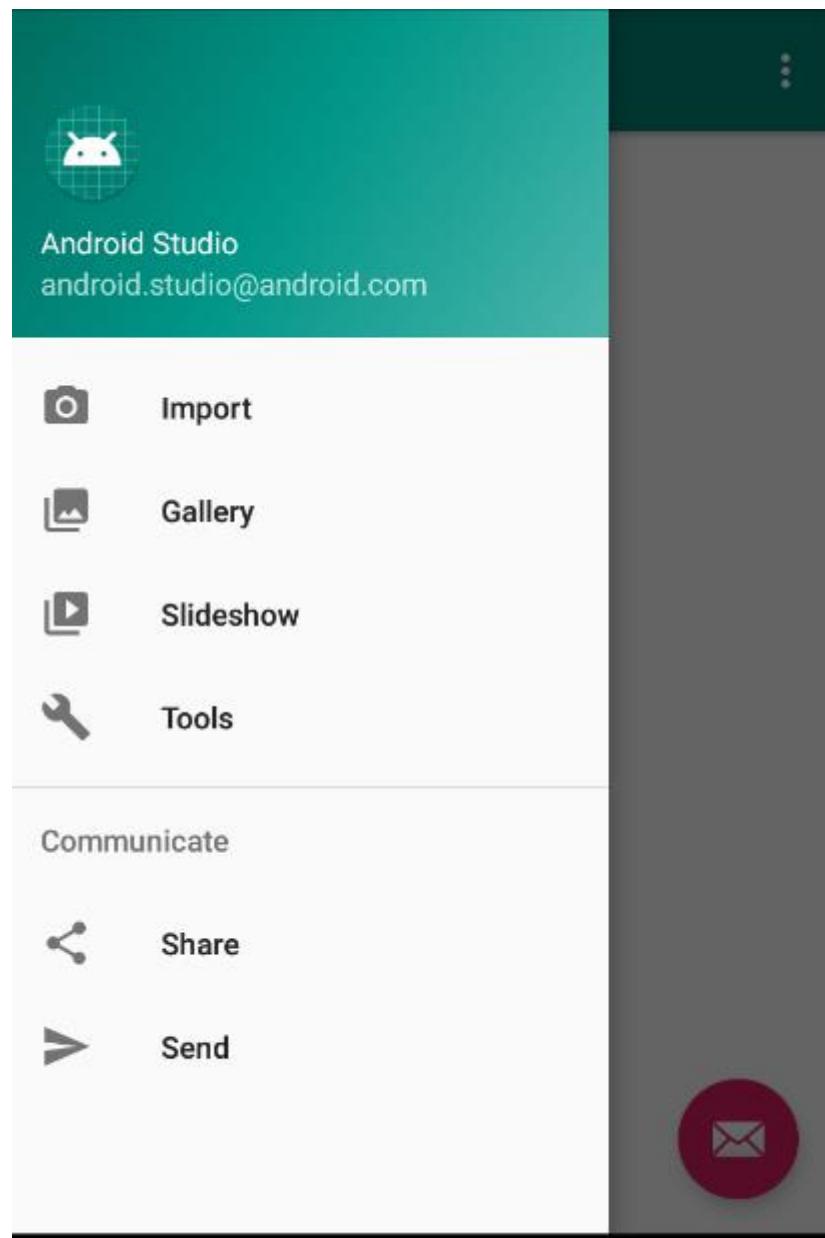
Hình 4-3: Minh họa Navigation Drawers

4.2.2 | Tạo Navigation Drawers

Tạo mới bất kỳ project Android nào, đến bước chọn một **template** → chọn **Navigation Drawer Activity**. Tùy chọn này giúp tạo một ứng dụng có cả **Navigation Drawer** lẫn **Floating Action Button**.



Hình 4-4: Navigation drawer chọn template



Hình 4-5: Giao diện Navigation Drawer

4.2.3 | Thao tác với Navigation Drawers

4.2.3.1 / Danh sách navigation.

↳ res/menu → activity_main_drawer.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"  
      xmlns:tools="http://schemas.android.com/tools"  
      tools:showIn="navigation_view">
```

```
<group android:checkableBehavior="single">  
    <item
```

```

        android:id="@+id/nav_camera"
        android:icon="@drawable/ic_menu_camera"
        android:title="Import" />
    <item
        android:id="@+id/nav_gallery"
        android:icon="@drawable/ic_menu_gallery"
        android:title="Gallery" />
    <item
        android:id="@+id/nav_slideshow"
        android:icon="@drawable/ic_menu_slideshow"
        android:title="Slideshow" />
    <item
        android:id="@+id/nav_manage"
        android:icon="@drawable/ic_menu_manage"
        android:title="Tools" />
</group>

<item android:title="Communicate">
    <menu>
        <item
            android:id="@+id/nav_share"
            android:icon="@drawable/ic_menu_share"
            android:title="Share" />
        <item
            android:id="@+id/nav_send"
            android:icon="@drawable/ic_menu_send"
            android:title="Send" />
    </menu>
</item>
</menu>

```

4.2.3.2 / Xây dựng layout hiển thị Navigationview.

↳ Res/layout → activity_main.xml

```

<android.support.v4.widget.DrawerLayout
    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:id="@+id/drawer_layout"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:fitsSystemWindows="true"
tools:openDrawer="start">>

<include
    layout="@layout/app_bar_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

<android.support.design.widget.NavigationView
    android:id="@+id/nav_view"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:fitsSystemWindows="true"
    app:headerLayout="@layout/nav_header_main"
    app:menu="@menu/activity_main_drawer" />

</android.support.v4.widget.DrawerLayout>

```

4.2.3.3 / Bắt sự kiện khi click vào menu của list trong Navigation Drawer.

```

public boolean onNavigationItemSelected(MenuItem item) {
    // Handle navigation view item clicks here.
    int id = item.getItemId();

    if (id == R.id.nav_camera) {
        // Handle the camera action
    } else if (id == R.id.nav_gallery) {

```

```
        } else if (id == R.id.nav_slideshow) {  
        } else if (id == R.id.nav_manage) {  
        } else if (id == R.id.nav_share) {  
        } else if (id == R.id.nav_send) {  
    }  
  
DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);  
drawer.closeDrawer(GravityCompat.START);  
return true;  
}
```

4.3 | Bài tập Chương 4

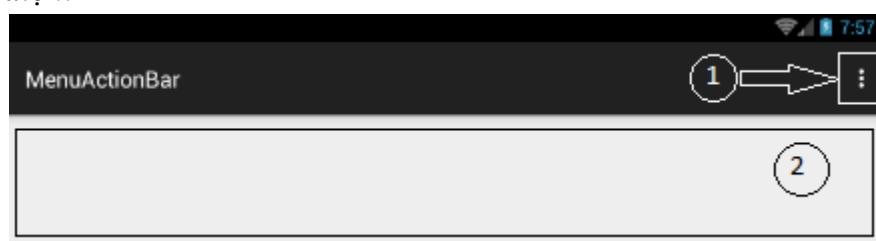
Bài tập 4.1. Tạo và sử dụng OptionMenu

Đề bài:

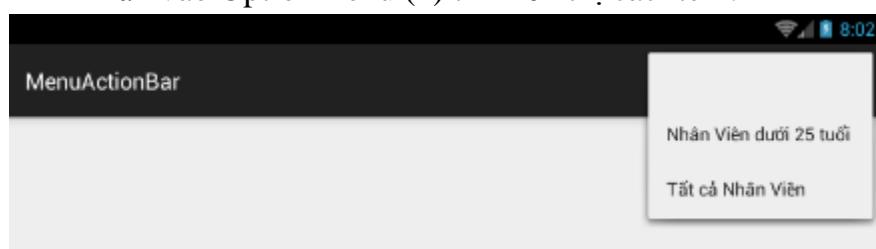
- Lọc những nhân viên có độ tuổi nhỏ hơn 25 và tất cả nhân viên.

Chú ý: Sử dụng OptionMenu.

Giao diện:



Khi nhấn vào OptionMenu (1) thì hiển thị các item:



Khi chọn vào item thì lọc dữ liệu hiển thị lên ListView (2).

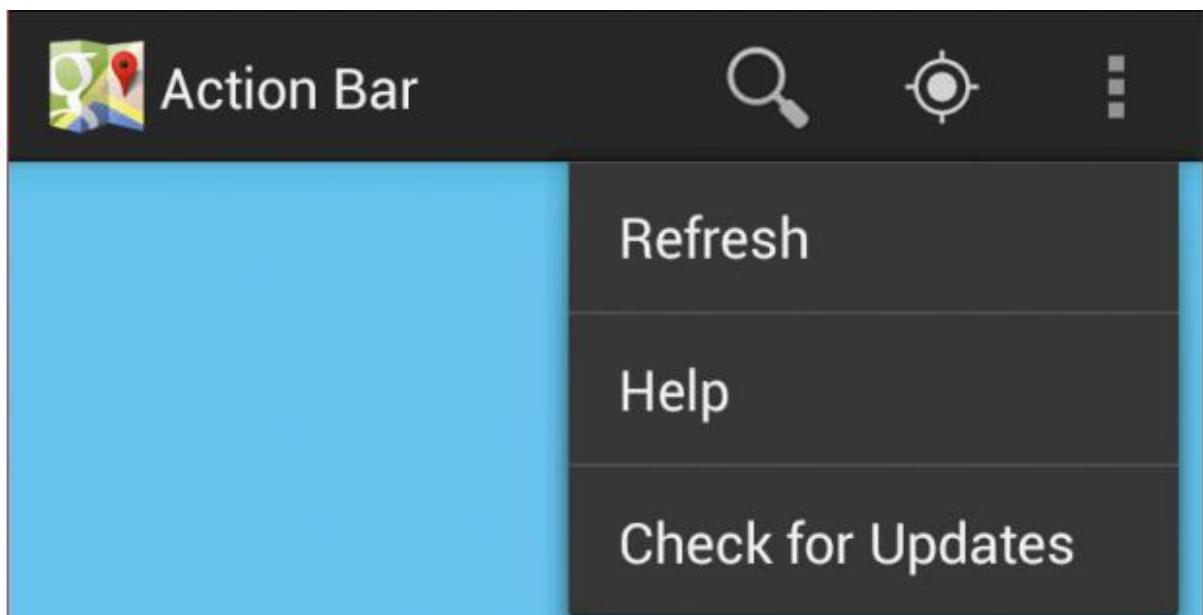
Gợi ý thực hiện:

- Tạo file “main.xml” trong thư mục “res/menu/main.xml”.
- Trong hàm “MainActivity” override lại 2 phương thức “onCreateOptionsMenu” và “onOptionsItemSelected” để tạo và bắt sự kiện khi chọn item menu.

Bài tập 4.2. Xây dựng chương trình sử dụng ActionBar



Bài tập 4.3. Xây dựng chương trình sử dụng ActionBar SearchView



CHƯƠNG 5. Lập trình cơ sở dữ liệu với SQLite

MỤC TIÊU THỰC HIỆN

- Trình bày được các khái niệm về SQLite
- Trình bày được các bước thực hiện database trong SQLite
- Tạo được database trong SQLite
- Thực hiện được các thao tác dữ liệu trong Sqlite

5.1 | Giới thiệu về SQLite

SQLite là cơ sở dữ liệu mở được viết dưới dạng thư viện tích hợp nhúng vào Android, hỗ trợ các đặc điểm về quan hệ chuẩn của cơ sở dữ liệu như cú pháp, transaction, các câu lệnh. SQLite được sử dụng rộng rãi trong các ứng dụng di động trên Android, iOS, ... Mozilla Firefox cũng sử dụng SQLite để lưu trữ các dữ liệu về cấu hình.

Với bộ thư viện được tích hợp sẵn SQLite sẽ giúp các lập trình viên có thể lưu trữ và phục hồi dữ liệu bất cứ lúc nào. Tuy chỉ là phiên bản rút gọn nhưng SQLite có thể đáp ứng được hầu hết các nhu cầu về xử lý dữ liệu. Tính linh động trong SQLite cũng giúp ta dễ kiểm soát được các thông tin dữ liệu. Sử dụng SQLite không yêu cầu về thiết lập bất cứ cơ sở dữ liệu hoặc đòi hỏi quyền admin

SQLite hỗ trợ các kiểu dữ liệu : **TEXT, INTEGER, REAL**.

Đường dẫn của cơ sở dữ liệu: **DATA/data//databases/FILENAME**

Mặc định mỗi ứng dụng sẽ được cấp phát một thư mục cho việc lưu trữ cơ sở dữ liệu và nó chỉ có thể được dùng bởi ứng dụng đó. Nếu muốn chia sẻ dữ liệu dùng chung giữa các ứng dụng ta có thể sử dụng **Content Provider**

5.2 | Sử dụng SQLite trong Android

5.2.1 | Tạo và xóa cơ sở dữ liệu

Tạo cơ sở dữ liệu thông qua lớp **SQLiteOpenHelper**.

- ↳ SQLiteOpenHelper là một lớp ảo, SQLiteOpenHelper giúp tạo các cơ sở dữ liệu dùng SQLite (vì SQLite không hỗ trợ các phương thức khởi tạo cơ sở dữ liệu). Vậy làm sao để sử dụng được SQLiteOpenHelper? Vì SQLiteOpenHelper là một lớp ảo nên ta cần khai báo một lớp khác kế thừa lớp này.
- ↳ SQLiteOpenHelper thực hiện các phương thức cần thiết cho phép khởi tạo, nâng cấp cơ sở dữ liệu. Tạo đối tượng để truy cập cơ sở dữ liệu (Read và Write).
- ↳ *SQLiteDatabase* cung cấp phương thức *insert()*, *update()*, *delete()*, hoặc *execSQL()* cho phép thực hiện truy xuất dữ liệu và Override phương thức *onCreate()* để tạo cơ sở dữ liệu. Override phương thức *onUpgrade()* để nâng cấp cơ sở dữ liệu.
- ☞ Override phương thức **onCreate()** để tạo cơ sở dữ liệu.
- ☞ Override phương thức **onUpgrade()** để nâng cấp cơ sở dữ liệu.
- ☞ Lớp SQLiteOpenHelper cung cấp 2 phương thức **getReadableDatabase()** và **getWritableDatabase()** để trả về đối tượng SQLiteDatabase.

Ví dụ: Tạo một CSDL có tên: COUNTRY_DB Trong đó:

- ☞ Tạo một Table có tên: COUNTRY.
- ☞ Table COUNTRY có 3 cột là: _id, enName và viName với _id là khóa chính và có giá trị tự động tăng.

```
private static final String DB_NAME= "Country_db";
public static final String TABLE_NAME = "Country";
public static final String COL_ID = "_id";
public static final String COL_EN_NAME = "enName";
public static final String COL_VI_NAME = "viName";
private static final String CREATE_TABLE = """
    + "create table " + TABLE_NAME + " ( "
    + COL_ID + " integer primary key autoincrement ,"
    + COL_EN_NAME + " text not null , "
    + COL_VI_NAME + " text not null );";
```

```
public MySqlliteHelper(Context context) {
```

```
super(context, TABLE_NAME, null, 1);  
}
```

Hàm khởi tạo có các thông số sau:

- ☞ **Context**: Biến ngữ cảnh.
- ☞ **DB_NAME**: Tên của cơ sở dữ liệu.
- ☞ **CursorFactory**: Đôi lúc chúng ta có thể extend lớp cursor để kế thừa một số phương thức và truy vấn. Trong trường hợp đó, ta dùng một instance của CursorFactory để tham chiếu đến lớp chúng ta tạo thay cho mặc định. Khi dùng mặc định thì ta để nó null.
- ☞ **Version**: Version của cơ sở dữ liệu.

Tiến hành tạo CSDL trong hàm onCreate():

```
public void onCreate(SQLiteDatabase db) { db.execSQL(CREATE_TABLE);  
}  
@Override  
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
  
}
```

5.2.2 | Mở Cơ sở dữ liệu

Dùng cú pháp **getReadableDatabase()** và **getWritableDatabase()** để trả về đối tượng SQLiteDatabase cần sử dụng.

SQLiteDatabase là đối tượng để truy cập cơ sở dữ liệu (Read và Write).

SQLiteDatabase cung cấp phương thức **insert()**, **update()**, **delete()**, hoặc **execSQL()** cho phép thực hiện truy xuất dữ liệu.

```
public class MyCounTryDB {  
    SQLiteDatabase database;  
    MySqliteHelper myHelper;  
    public MyCounTryDB(Context context) {
```

```

myHelper = new MySqliteHelper(context);
try {
    database = myHelper.getWritableDatabase();
} catch (SQLiteException ex) {
    database = myHelper.getReadableDatabase();
}

}
}

```

Nếu cơ sở dữ liệu đã tồn tại thì lớp SQLiteOpenHelper sẽ không tạo thêm mà chỉ mở kết nối đến cơ sở dữ liệu đó.

5.2.3 | Đóng Cơ sở dữ liệu

```

public void close()
{
    myHelper.close();
}

```

5.2.4 | Truy vấn dữ liệu trong các bảng

Truy vấn (Query): Sử dụng phương thức `rawQuery()` của lớp SQLiteDatabase. Phương thức `rawQuery()` nhận vào một giá trị chuỗi là câu lệnh SQL dùng để truy vấn dữ liệu và trả ra một đối tượng Cursor.

```

public Cursor Select_ALL() {
    String sql="Select * From CounTry where _id = ? and enName =?";
    Cursor cursor = database.rawQuery(sql,new String[]{"1","VietNam"});
    return cursor;
}

```

↳ Cú pháp truy xuất dữ liệu trong câu lệnh SQL:

❖ Phát biểu truy vấn SQL có dạng:

```
SELECT {Tên trường cần truy vấn, * nếu lấy tất cả các trường}  
FROM {Tên Table}  
WHERE {Biểu thức điều kiện}
```

- ☞ **SELECT** dùng để đọc thông tin từ cơ sở dữ liệu theo trường hợp quy định hay những biểu thức cho trường hợp đó.
 - ☞ **FROM** chỉ ra tên một bảng hay những bảng có liên quan cần truy vấn thông tin.
 - ☞ **WHERE** để tạo nên điều kiện cần lọc mẫu tin theo tiêu chuẩn được định nghĩa. Thông thường, WHERE dùng cột (trường) để so sánh với giá trị cột khác, hay biểu thức chứa cột (trường) bất kỳ có trong bảng (table).
- ↳ Truy vấn với cú pháp hàm
- ☞ Sử dụng phương thức `query()` của lớp `SQLiteDatabase`.
 - ☞ Hỗ trợ lập trình viên truy xuất dữ liệu từ CSDL mà không cần biết cú pháp các câu lệnh SQL.

Phương thức `query()` cũng trả ra một đối tượng Cursor. Để lấy dữ liệu từ Cursor, trước hết cần chuyển đến vị trí xác định với các phương thức: `moveToFirst()`, `moveToNext()`, `moveToPosition(int position)`

- ↳ Cursor:

Đối tượng dữ liệu được trả về khi thực hiện truy vấn dữ liệu trong cơ sở dữ liệu của SQLite hoặc Content Provider.

- ☞ Thể hiện dữ liệu ở dạng bảng quan hệ :
- ☞ Cột: thể hiện trường dữ liệu (hoặc thuộc tính dữ liệu).
- ☞ Dòng: một thể hiện dữ liệu

```
Cursor cursor = database.query(String table,
                                String[] columns,
                                String selection,
                                String[] selectionArgs,
                                String groupBy,
                                String having,
                                String orderBy);
```

- ☞ String table: tên của bảng cần truy vấn.
- ☞ String[] columns: danh sách các cột sẽ trả về dữ liệu.
- ☞ String selection: chứa các điều kiện truy vấn.
- ☞ String[] selectionArgs: danh sách các tham số phụ cho câu điều kiện.
- ☞ String[] groupBy: gom nhóm các cột kết quả.
- ☞ String[] having: bộ lọc theo điều kiện.
- ☞ String[] orderBy: sắp xếp theo mảng cột được chỉ định.

```
public ArrayList<Country> Select_query() {
    ArrayList<Country> data = new ArrayList<>();
    String[] select_Column = {"id", "enName", "viName"};
    Cursor cursor = database.query("Country", select_Column, null, null, null, null);
    if (cursor.moveToFirst()) {
        data.clear();
        while (!cursor.isAfterLast()) {
            int id =
```

```

    Integer.parseInt(cursor.getString(cursor.getColumnIndex(select_Column[0])));

    String enName =
    cursor.getString(cursor.getColumnIndex(select_Column[1]));

    String viName =
    cursor.getString(cursor.getColumnIndex(select_Column[2]));

    CounTry counTry = new CounTry(id, enName, viName);
    data.add(counTry);

}

return data;
}

```

Lớp SQLiteDatabase cung cấp các phương thức như: insert(), update(), delete() hoặc execSQL() cho phép người dùng có thể quản lý và truy xuất dữ liệu.

☞ Insert

```

public long Insert(CounTry counTry) {

    ContentValues values = new ContentValues();
    values.put("enName",counTry.getEnName());
    values.put("viName",counTry.getViName());
    return database.insert("CounTry", null, values);
}

```

☞ Update

```

public long update(CounTry counTry) {

    ContentValues values = new ContentValues();
    values.put("enName",counTry.getEnName());
    values.put("viName",counTry.getViName());
    return database.update("CounTry", values,

```

```
"_id = " + counTry.getId(), null);  
}
```

❖ Delete

Sử dụng câu truy vấn delete truyền vào tên bảng và chỉ số của hàng cần xóa trong mệnh đề WHERE.

Nếu mệnh đề WHERE không có sẽ thực hiện xóa tất cả các dòng:

```
public long Delete(CounTry counTry) {  
  
    return database.delete("Country", "id = " + counTry.getId(), null);  
}
```

5.2.5 | Sắp xếp dữ liệu

❖ Truy vấn sắp xếp

Thông thường trong khi truy vấn mẫu tin từ bảng dữ liệu, kết quả hiển thị sắp xếp theo chiều tăng hay giảm dựa trên ký tự ALPHABET. Nhưng cũng có thể sắp xếp theo một tiêu chuẩn bất kỳ.

Cú pháp cho mệnh đề ORDER BY cùng với trạng thái tăng dần (ASCENDING) hoặc giảm dần (DESCENDING):

❖ Tăng dần (ASCENDING)

Truy vấn với câu lệnh SQL:

```
SELECT * FROM COUNTRY ORDER BY enName ASC
```

Truy vấn với cú pháp hàm:

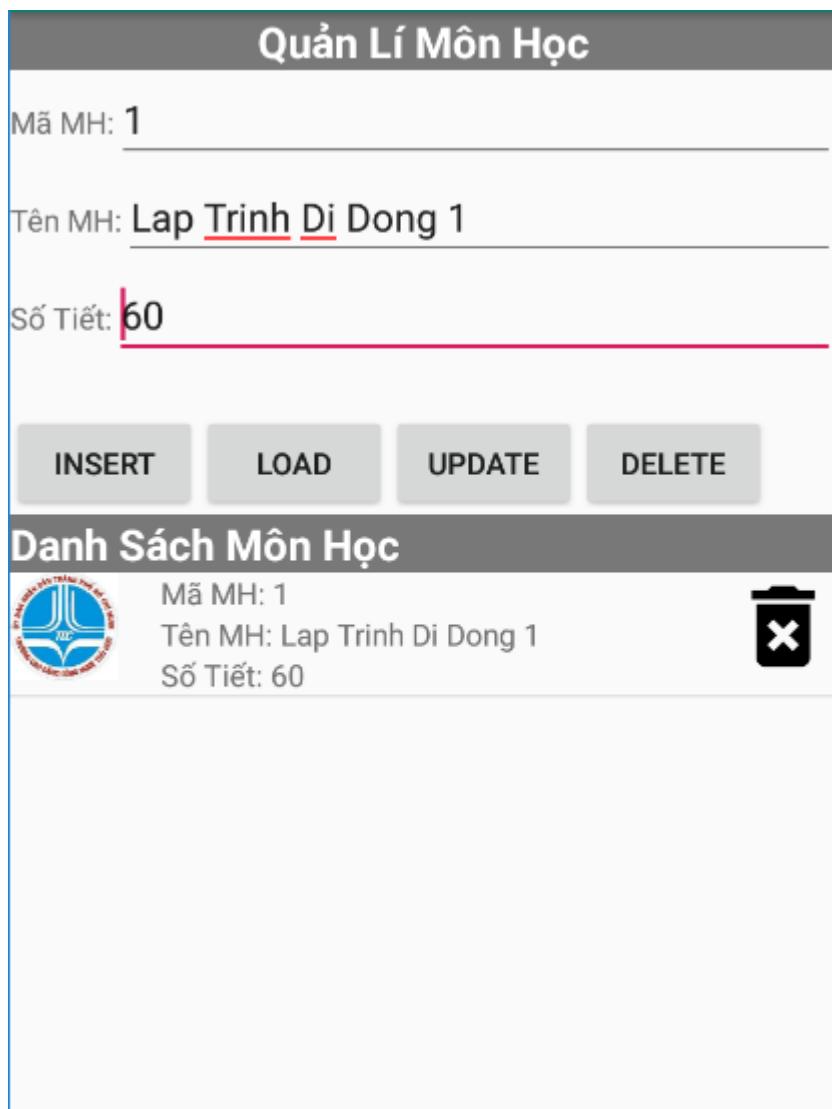
```
String orderBy = myHelper.COL_EN_NAME + " ASC";  
database.query(TABLE_NAME, null, null, null, null, orderBy);
```

❖ Giảm dần (DESCENDING)

```
ORDER BY columNname DESC  
ORDER BY columNname1 + columnName2 DESC
```

```
database.query(TABLE_NAME, null, null, null, null, orderBy);
```

Ví dụ 1. Xây dựng chương trình quản lý môn học



Bước 1: Tạo một project tên là QLMonhoc: **File → New → Android Application Project** điền các thông tin → **Next →Finish**.

Bước 2: Vào thư mục **res/values** bô sung **string.xml**

```
<resources>
    <string name="app_name">Test</string>
    <string name="qlmh">Quản Lý Môn Học</string>
    <string name="dsmh">Danh Sách Môn Học</string>
    <string name="mamonhoc">Mã MH:</string>
    <string name="tenmonhoc">Tên MH:</string>
    <string name="sotiet">Số Tiết:</string>
    <string name="nhapmonhoc">Insert</string>
```

```
<string name="loadmonhoc">Load</string>
<string name="update">Update</string>
<string name="delete">Delete</string>

</resources>
```

Bước 3: Mở res → layout

☞ Tạo file listview_item.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginRight="30dp"
        android:layout_weight="1"
        android:orientation="vertical">

        <TextView
            android:id="@+id/tvTenSv"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"

            />

        <TextView
            android:id="@+id/tvLop"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            />
    
```

```
    />

</LinearLayout>

<ImageView
    android:id="@+id/imDelete"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:src="@drawable/ic_delete_forever_black_24dp"
    />

</LinearLayout>
```

☞ Tạo file main_activity.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MonHocActivity">
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="2"
    android:orientation="vertical">
```

```
<TextView
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#787878"
    android:gravity="center"
    android:text="@string/qlmh"
    android:textColor="#ffffff"
    android:textSize="20sp"
    android:textStyle="bold" />
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
```

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

```
<EditText
    android:id="@+id/mamonhoc"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
</LinearLayout>
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
```

```
<TextView
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/tenmonhoc" />

<EditText
    android:id="@+id/tenmonhoc"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
</LinearLayout>

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

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

<EditText
    android:id="@+id/sotiethoc"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
</LinearLayout>

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

```
        android:orientation="horizontal">

<Button
    android:id="@+id/btnNhapMH"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/nhapmonhoc" />

<Button
    android:id="@+id	btnLoad"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/loadmonhoc" />

<Button
    android:id="@+id	btnUpdate"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/update" />

<Button
    android:id="@+id	btnDelete"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/delete" />

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
```

```

    android:layout_weight="3"
    android:orientation="vertical">

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#787878"
    android:text="@string/dsmh"
    android:textColor="#ffffff"
    android:textSize="20sp"
    android:textStyle="bold" />

<ListView
    android:id="@+id/listView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</ListView>
</LinearLayout>

</LinearLayout>

```

Bước 4: Mở app → src

❖ Tạo file MonHoc.java

```

public class MonHoc {
    public MonHoc() {
    }

    public MonHoc(int img, String ten, String ma, String sotiet) {
        this.img = img;
        this.ten = ten;
        this.ma = ma;
    }
}

```

```
this.sotiet = sotiet;
}

public int getImg() {
    return img;
}

public void setImg(int img) {
    this.img = img;
}

public String getTen() {
    return ten;
}

public void setTen(String ten) {
    this.ten = ten;
}

public String getMa() {
    return ma;
}

public void setMa(String ma) {
    this.ma = ma;
}

public String getSotiet() {
    return sotiet;
}
```

```

public void setSotiet(String sotiet) {
    this.sotiet = sotiet;
}

@Override
public String toString() {
    return "MonHoc{" +
        "img=" + img +
        ", ten='" + ten + '\'' +
        ", ma='" + ma + '\'' +
        ", sotiet='" + sotiet + '\'' +
        '}';
}

private int img;
private String ten, ma, sotiet;
}

```

☞ Tạo file MonHocAdapter.java

```

public class MonHocAdapter extends ArrayAdapter<MonHoc> {
    Context context;
    int layoutResourceId;
    ArrayList<MonHoc> data = null;

    public MonHocAdapter(Context context, int layoutResourceId,
    ArrayList<MonHoc> data) {
        super(context, layoutResourceId, data);
        this.context = context;
        this.layoutResourceId = layoutResourceId;
        this.data = data;
    }
}

```

```

static class MonHocHolder {
    ImageView img,imgDelete;
    TextView txtMaMH, txtTenMH, txtSoTiet;
}

public View getView(final int position, View convertView, ViewGroup parent) {
    View row = convertView;
    MonHocHolder holder = null;

    if(row != null)
    {
        holder = (MonHocHolder) row.getTag();
    }
    else
    {
        holder = new MonHocHolder();
        LayoutInflater inflater = ((Activity)context).getLayoutInflater();
        row = inflater.inflate(R.layout.list_item_row, parent, false);

        holder.img = (ImageView) row.findViewById(R.id.tdc);
        holder.txtMaMH = (TextView) row.findViewById(R.id.mamh);
        holder.txtTenMH = (TextView) row.findViewById(R.id.tenmh);
        holder.txtSoTiet = (TextView) row.findViewById(R.id.sotiet);
        holder.imgDelete = (ImageView) row.findViewById(R.id.imgDelete);

        row.setTag(holder);
    }
    final MonHoc mh = data.get(position);

    holder.img.setImageResource(R.drawable.tdc);
}

```

```

holder.txtMaMh.setText("Mã MH: " + mh.getMa());
holder.txtTenMH.setText("Tên MH: " + mh.getTen());
holder.txtSoTiet.setText("Số Tiết: " + mh.getSoTiet());
holder.imgDelete.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        MonHocDatabase db = new MonHocDatabase(context);
        db.deleteMH(mh.getMa());
        Toast.makeText(context, "Đã xóa mã" + mh.getMa(),
        Toast.LENGTH_SHORT).show();
    }
});

return row;
}
}

```

Tạo file MonHocDataBae.java

```

public class MonHocDatabase extends SQLiteOpenHelper {
    private static String DB_NAME = "dbMonHoc.db";
    private static int DB_VERSION = 1;

    //Define table Monhoc
    private static final String TB_MONHOCS = "tbMonHoc";
    private static final String COL_MONHOC_ID = "monhoc_id";
    private static final String COL_MONHOC_MA = "monhoc_ma";
    private static final String COL_MONHOC_TEN = "monhoc_ten";
    private static final String COL_MONHOC_SOTIET = "monhoc_sotiet";

    public MonHocDatabase(Context context)
    {
        super(context, DB_NAME, null, DB_VERSION);
    }
}

```

```
}
```

@Override

```
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
    db.execSQL("DROP TABLE IF EXISTS " + TB_MONHOCS);  
    onCreate(db);  
}
```

@Override

```
public void onCreate(SQLiteDatabase db) {  
    String scriptTBMonHocs = "CREATE TABLE " + TB_MONHOCS + "(" +  
        COL_MONHOC_ID + " INTEGER PRIMARY KEY  
        AUTOINCREMENT NOT NULL, " +  
        COL_MONHOC_MA + " TEXT, " +  
        COL_MONHOC_TEN + " TEXT, " +  
        COL_MONHOC_SOTIET + " TEXT) ";  
  
    //Execute script  
    db.execSQL(scriptTBMonHocs);  
}
```

```
public void getMonHoc(ArrayList<MonHoc> monHocs)  
{  
    SQLiteDatabase db = getWritableDatabase();  
    Cursor cursor = db.query(TB_MONHOCS, new  
    String[]{COL_MONHOC_MA, COL_MONHOC_TEN,  
    COL_MONHOC_SOTIET}, null,null,null, null, null);  
    if(cursor.moveToFirst()) {  
        do {  
            MonHoc mh = new MonHoc();  
            mh.setMa(cursor.getString(cursor.getColumnIndex(COL_MONHOC_MA)));  
            mh.setTen(cursor.getString(cursor.getColumnIndex(COL_MONHOC_TEN)));  
            mh.setSotiet(cursor.getInt(cursor.getColumnIndex(COL_MONHOC_SOTIET)));  
            monHocs.add(mh);  
        } while(cursor.moveToNext());  
    }  
}
```

```

mh.setTen(cursor.getString(cursor.getColumnIndex(COL_MONHOC_TEN)));

mh.setSotiet(cursor.getString(cursor.getColumnIndex(COL_MONHOC_SOTIET))
);

monHocs.add(mh);
} while (cursor.moveToNext());
}

}

public void getAllData(ArrayList<MonHoc> monHocs) {
SQLiteDatabase db = this.getWritableDatabase();
String query = "select * from "+TB_MONHOCS;
Cursor cursor = db.rawQuery(query,null);
if(cursor.moveToFirst()) {
do {
MonHoc mh = new MonHoc();

mh.setMa(cursor.getString(cursor.getColumnIndex(COL_MONHOC_MA)));
mh.setTen(cursor.getString(cursor.getColumnIndex(COL_MONHOC_TEN)));
mh.setSotiet(cursor.getString(cursor.getColumnIndex(COL_MONHOC_SOTIET)))
};

monHocs.add(mh);
} while (cursor.moveToNext());
}
}

```

```

public void saveMH(MonHoc mh)
{
    SQLiteDatabase db = getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COL_MONHOC_MA, mh.getMa());
    values.put(COL_MONHOC_TEN, mh.getTen());
    values.put(COL_MONHOC_SOTIET, mh.getSotiet());
    db.insert(TB_MONHOCS, null, values);
    db.close();
}

public void deleteMH(String maHH)
{
    SQLiteDatabase db = this.getWritableDatabase();
    String query = "DELETE FROM tbMonHoc WHERE monhoc_ma=" +
maHH;
    db.execSQL(query);
}

public void updateMH(MonHoc mh)
{
    SQLiteDatabase db = this.getWritableDatabase();
    String sql = "Update "+ TB_MONHOCS +" set ";
    sql += COL_MONHOC_TEN +" = '"+ mh.getTen()+"', ";
    sql += COL_MONHOC_SOTIET +" = '"+ mh.getSotiet()+"' ";
    sql += " WHERE "+ COL_MONHOC_MA +" = '"+ mh.getMa()+"';

    db.execSQL(sql);
}

```

❖ Tạo file MainActivity.java

```
public class MonHocActivity extends AppCompatActivity {  
    private ListView lvDanhSach;
```

```
    ArrayList<MonHoc> data = new ArrayList<>();
```

```
    MonHocAdapter adapter = null;
```

```
    Button btnNhap, btnLoad, btnUpdate, btnDelete;
```

```
    EditText txtMaMH, txtTenMH, sotiet;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

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

```
    setControl();
```

```
    setEvent();
```

```
}
```

```
public void setControl() {
```

```
    lvDanhSach = (ListView) findViewById(R.id.listView1);
```

```
    btnNhap = (Button) findViewById(R.id.btnNhapMH);
```

```
    txtMaMH = (EditText) findViewById(R.id.mamonhoc);
```

```
    txtTenMH = (EditText) findViewById(R.id.tenmonhoc);
```

```
    sotiet = (EditText) findViewById(R.id.sotieuthoc);
```

```
    btnLoad = (Button) findViewById(R.id.btnLoad);
```

```
    btnUpdate = (Button) findViewById(R.id.btnUpdate);
```

```
    btnDelete = (Button) findViewById(R.id.btnDelete);
```

```
}
```

```
public void setEvent() {
```

```
adapter = new MonHocAdapter(this, R.layout.list_item_row, data);
lvDanhSach.setAdapter(adapter);
```

```
khoiTao();
adapter.notifyDataSetChanged();
```

```
btnNhap.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
```

```
    public void onClick(View v) {
```

```
        Nhap();
```

```
    }
```

```
});
```

```
btnLoad.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
```

```
    public void onClick(View v) {
```

```
        LoadData();
```

```
    }
```

```
});
```

```
btnUpdate.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
```

```
    public void onClick(View v) {
```

```
        MonHoc monHoc = getMonHoc();
```

```
        UpdateMH(monHoc);
```

```
        LoadData();
```

```
    }
```

```
});
```

```
btnDelete.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
```

```

public void onClick(View v) {
    MonHoc monHoc = getMonHoc();
    DeleteMH(monHoc.getMa());
    LoadData();
}

lvDanhSach.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
        MonHoc mh = data.get(position);
        txtMaMH.setText(mh.getMa());
        txtTenMH.setText(mh.getTen());
        sotiet.setText(mh.getSotiet());
        Toast.makeText(MonHocActivity.this, mh.toString(),
Toast.LENGTH_SHORT).show();
    }
});
}

lvDanhSach.setOnItemLongClickListener(new
AdapterView.OnItemLongClickListener() {
    @Override
    public boolean onItemLongClick(AdapterView<?> parent, View view, int
position, long id) {
        MonHoc mh = data.get(position);
        DeleteMH(mh.getMa());
        return false;
    }
});

```

```

    });
}

public void khoiTao() {
    data.add(new MonHoc(R.drawable.tdc, "Lap Trinh Di Dong 1", "1", "60"));
}

public void nhapMH() {
    data.add(new MonHoc(R.drawable.tdc, txtTenMH.getText().toString(),
    txtMaMH.getText().toString(), sotiet.getText().toString()));
}

public void Nhap() {
    MonHocDatabase db = new MonHocDatabase(this);
    MonHoc monHoc = getMonHoc();
    db.saveMH(monHoc);
}

private MonHoc getMonHoc() {
    MonHoc monHoc = new MonHoc();
    monHoc.setMa(txtMaMH.getText().toString());
    monHoc.setTen(txtTenMH.getText().toString());
    monHoc.setSotiet(sotiet.getText().toString());
    return monHoc;
}

public void LoadData() {
    MonHocDatabase db = new MonHocDatabase(this);
    data.clear();
    db.getAllData(data);
    adapter.notifyDataSetChanged();
}

```

```
}
```

```
public void DeleteMH(String sMa) {  
    MonHocDatabase db = new MonHocDatabase(this);  
    data.clear();  
    db.deleteMH(sMa);  
    db.getMonHoc(data);  
    adapter.notifyDataSetChanged();  
}
```

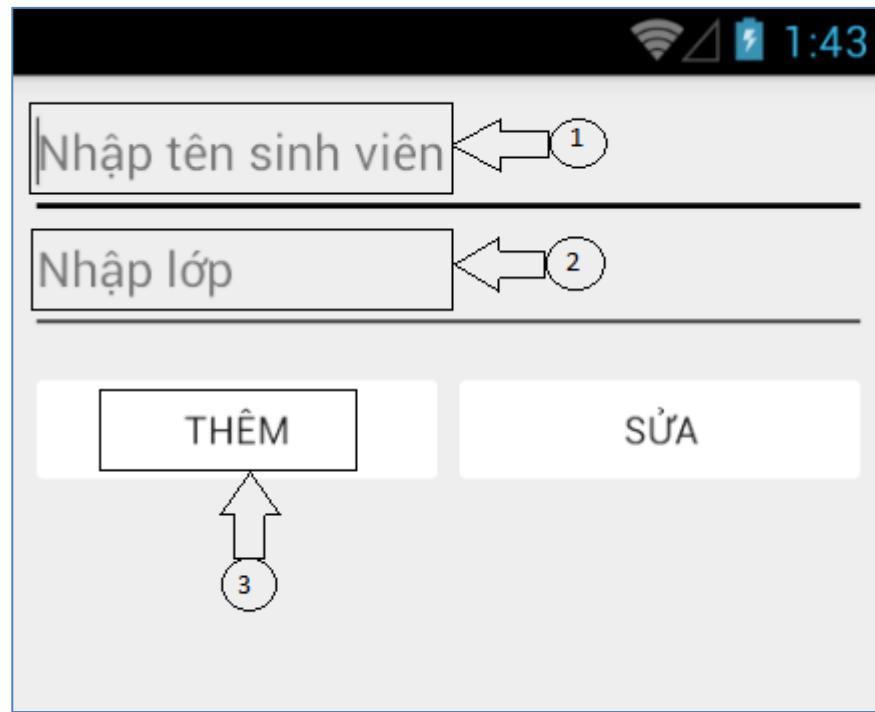
```
public void UpdateMH(MonHoc mh) {  
    MonHocDatabase db = new MonHocDatabase(this);  
    db.updateMH(mh);  
    LoadData();  
}  
}
```

5.3 | Bài tập Chương 5

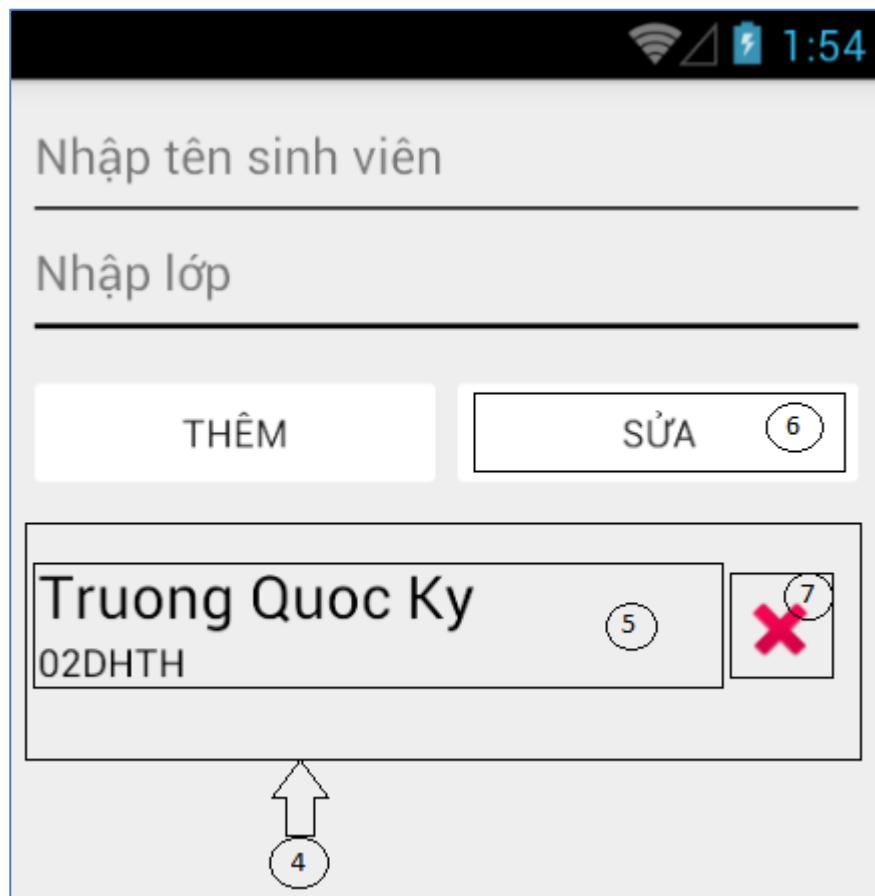
**Bài tập 5.1. Xây dựng ứng dụng quản lý sinh viên, dữ liệu SinhVien gồm:
Mã sinh viên, Tên sinh viên, Lớp.**

- Tạo cơ sở dữ liệu chứa bảng SinhVien.
- Thực hiện thêm, xóa, sửa.
- Lấy tất cả dữ liệu, lấy dữ liệu theo mã sinh viên, sắp xếp theo tên, theo lớp và xây dựng chức năng tìm kiếm sinh viên.

Với giao diện màn hình như sau:



- Sau khi nhấn thêm (3) thì thêm vào ListView (4) như sau:



- Khi nhấn vào item (5) thì hiển thị tên và lớp của item này lên 2 EditText tương ứng. Sau đó, có thể chỉnh sửa lại thông tin trên 2 EditText rồi ấn nút sửa (6) để cập nhật lại dữ liệu và hiển thị lên ListView (4).
- Khi nhấn vào ImageView (7) dòng nào thì xóa dòng đó trên ListView.

Bài tập 5.2. Xây dựng ứng dụng quản lý nhân viên

- Tạo một cơ sở dữ liệu SQLite gồm:
 - o Bảng Nhân Viên: Mã NV, Họ Tên, Ngày Sinh, Giới Tính, Địa Chỉ, Điện Thoại, Ghi Chú.
 - o Bảng Khách Hàng: Mã KH, Họ Tên, Ngày Sinh, Giới Tính, Email, Điện Thoại. Bảng Hóa Đơn: Mã HD, Ngày Bán, Mã NV, Mã KH, Tổng Tiền.
 - o Bảng Chi Tiết Hóa Đơn: Mã HD, Mã Sản Phẩm, Giá Bán, Số Lượng, Tổng Tiền.
 - o Bảng Sản Phẩm: Mã SP, Tên SP, Ngày SX, Giá Bán, Số Lượng.
- Tạo cơ sở dữ liệu SQLite có một bảng Nhân Viên. Thực hiện thêm, xóa, sửa, sắp xếp, lấy tất cả dữ liệu, lấy theo mã, lấy theo tên.
- Thêm 3 bảng còn lại vào cơ sở dữ liệu vừa tạo. Ràng buộc khóa chính, khóa ngoại. Xây dựng chức năng: tìm kiếm (Nhân Viên, Sản Phẩm, Khách Hàng), thực hiện mua/ bán hàng, thống kê số sản phẩm bán được, doanh thu theo thời gian nhập (ngày bắt đầu và ngày kết thúc), thống kê sản phẩm bán nhiều nhất trong tháng.

Giao diện: Tự thiết kế.

Giáo trình chính

[1] Dawn Griffiths and David Griffiths, **Head First Android Development**,, 2015

Tài liệu tham khảo

[1] Website: <http://o7planning.org/en/11007/android>

[2] Website <http://laptrinandroid.edu.vn>

[3] Website <http://hiepsit.com/khoa-hoc/android>