**Báo cáo tổng hợp kiến thức**

1. **Activity**

* Activity biểu diễn một màn hình với giao diện cho người dùng.
* Vòng đời của 1 activity:
* onCreate(): phương thức đầu tiên được gọi khi activity được tạo.
* onStart(): phương thức này được gọi khi activity trở nên nhìn thấy tới người dùng tuy nhiên vẫn chưa tương tác được.
* 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. Được gọi khi activity hiện tại đang được dừng và activity trước đó đang được phục hồi.
* onStop(): phương thức này được gọi trước khi activity không còn nhìn thấy được nữa.
* onDestroy(): phương thức này được gọi trước khi activity bị hủy bởi hệ thống.
* onRestart(): Được gọi khi activity tái khởi động.

1. **Service**

* Service là một thành phần chạy ngầm ở dưới ứng dụng, service có thể hoạt động mà không cần tương tác với người dùng và ngay cả khi ứng dụng bị hủy.
* Service có 2 trạng thái:
* Started: Một Service được bắt đầu khi một Component, chẳng hạn như activity gọi nó bằng phương thức *startService()*. Khi đã được bắt đầu, một Service có thể chạy ngầm định trong Background, ngay cả khi thành phần bắt đầu nó bị hủy.
* Bound: Một Service được gắn kết (bound) khi một Component kết nối tới nó bởi việc gọi *bindService()*. Một bound service cung cấp một giao diện Client-Server cho phép các thành phần để tương tác với Service đó, gửi yêu cầu, nhận kết quả.

1. **Broadcast Receiver**

* Broadcast Receiver phản hồi các thông báo phát ra từ các ứng dụng khác hoặc từ chính hệ thống.
* Ta cũng có thể tự tạo các tín hiệu broadcast theo ý của mình.

1. **Content Provider**

* Content Provider cung cấp dữ liệu từ một ứng dụng tới các ứng dụng khác theo yêu cầu.
* Content Provider vận hành khá giống một cơ sở dữ liệu, ta có thể truy vấn nó, sửa đổi nội dung, cũng như thêm hoặc xóa nội dung bởi sử dụng các phương thức insert(), update(), delete() và query().

1. **Fragment**

* Nếu hiểu activity là 1 màn hình thì ta có thể hiểu fragment là 1 mảnh màn hình, có thể thiết kế tùy ý để phù hợp với các loại màn hình (như khi xoay ngang hay màn hình cho máy tính bảng).
* Fragment có file layout riêng của nó.
* Vòng đời của fragment có điểm khác so với activity:



1. **Resource**

* Resource là các tài nguyên ví dụ như ảnh, âm thanh, file layout, các giá trị như string, file menu,... Tất cả được chứa trong thư mục **res/.**

1. **Intent - Filter**

* 1 intent có thể được sử dụng với startActivity() để chạy một Activity, broadcastIntent() để gửi tín hiệu tới bất kỳ thành phần BroadcastReceiver nào quan tâm đến, hay với startService(Intent) hoặc bindService(Intent, ServiceConnection, int) để giao tiếp với một Service ở Background.
* Các đối tượng intent là action và data.
* Android sử dụng các Filter để định vị tập hợp các Activity, Service và BroadcastReceiver mà có thể xử lý Intent với sự giúp đỡ của một tập đã cho của các Action, Category, Data Sheme gắn kết với một Intent.

1. **Layout**

* Linear Layout: là một view group mà căn chỉnh các view con theo chiều dọc hay chiều ngang.
* Relative Layout: là một view group mà hiển thị các view con với mối liên hệ về vị trí với nhau (ví dụ: nằm trên, nằm dưới, ở bên trái, ở bên phải).
* Table Layout: là một view mà nhóm tất cả các view vào trong các hàng và các cột (nhưng không hiện ra đường kẻ).
* Absolute Layout: cho phép xác định vị trí chính xác của các view con.
* Frame Layout: là một placeholder, thường dùng để gắn fragment vào.
* List View: là một view group mà hiển thị một danh sách item theo 1 cột.
* Grid View: là một view group mà hiển thị các item theo dạng lưới, có thể chia thành nhiều cột.
* Recycler View: là một view group cũng dùng để hiển thị danh sách các item, tuy nhiên có thể custom theo kiểu Grid Layout hay Staggered Layout. Ngoài ra cơ chế hoạt động của Recycler View là chỉ phải xác định “khung” của item 1 lần và gán dữ liệu vào nên tốc độ nhanh.

1. **UI Control**

* TextView: hiển thị dữ liệu dạng text.
* EditText : nhập dữ liệu dạng text.
* AutoCompleteTextView : giống như edit text nhưng có thêm khả năng gợi ý.
* Button: nút bấm với text ở trong
* ImageButton: tương tự như button nhưng có thể hiển thị bằng ảnh
* CheckBox: dùng để tích chọn
* ToggleButton: nút on/off.
* RadioButton: dùng để tích chọn, khi dùng với Radio Group thì chỉ có thể tích 1 lựa chọn trong tất cả.
* ProgressBar: vòng quay hay thanh chạy thường thấy khi load ứng dụng hay download, thường dùng khi cần báo hiệu cho người dùng chờ.
* Spinner: cung cấp lựa chọn cho người dùng theo dạng drop-down list.
* TimePicker: hiển thị và cho người dùng chọn giờ.
* DatePicker: hiển thị và cho người dùng chọn ngày tháng.

1. **Event Handling**

* Chỉ các hàm dùng để xử lí sự kiện như:
* onClickListener()
* onLongClickListener()
* onItemSelectedListener()
* addTextChangeListener()
* etc...

1. **Style and Theme**

* Thiết kế giao diện riêng cho các view hay kể cả layout.

1. **Custom Component**

* Tự tạo nên những component theo ý muốn để sử dụng bằng code java và xml.

1. **Mô hình MVP**

* M: model - là một interface xác định cách mà dữ liệu được hiển thị trong giao diện người dùng.
* V: view - là một giao diện người dùng thụ động hiển thị dữ liệu (model) và tiếp nhận tương tác người dùng và truyền tới để Presenter xử lý tương tác.
* P: Presenter - khi người dùng tương tác với View, Presenter tiếp nhận tương tác người dùng và update Model. Khi Model được update hay có thay đổi, Presenter lấy dữ liệu từ Model, định dạng và đưa tới View để hiển thị.