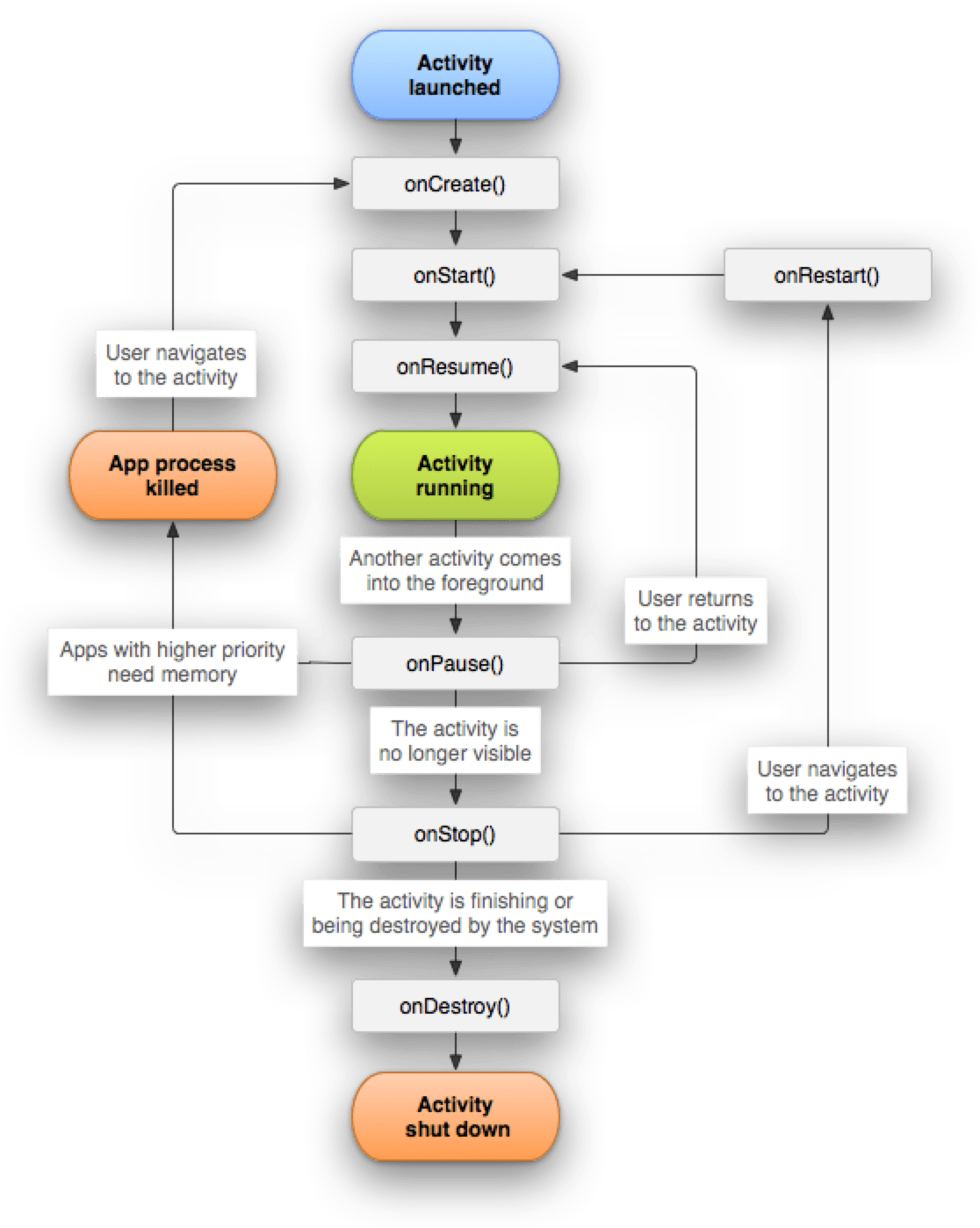
**1. Activity & life cycle Activity & Activity Stack**

Lớp **Activity** là thành phần quan trọng nhất của ứng dụng Android, cách mà chúng hoạt động tạo thành nền tảng cơ bản của mô hình lập trình ứng dụng.

**Android khởi chạy một ứng dụng thông thường bằng kích hoạt một Activity tương ứng với vòng đời cụ thể của nó trong quá trình hoạt động**.

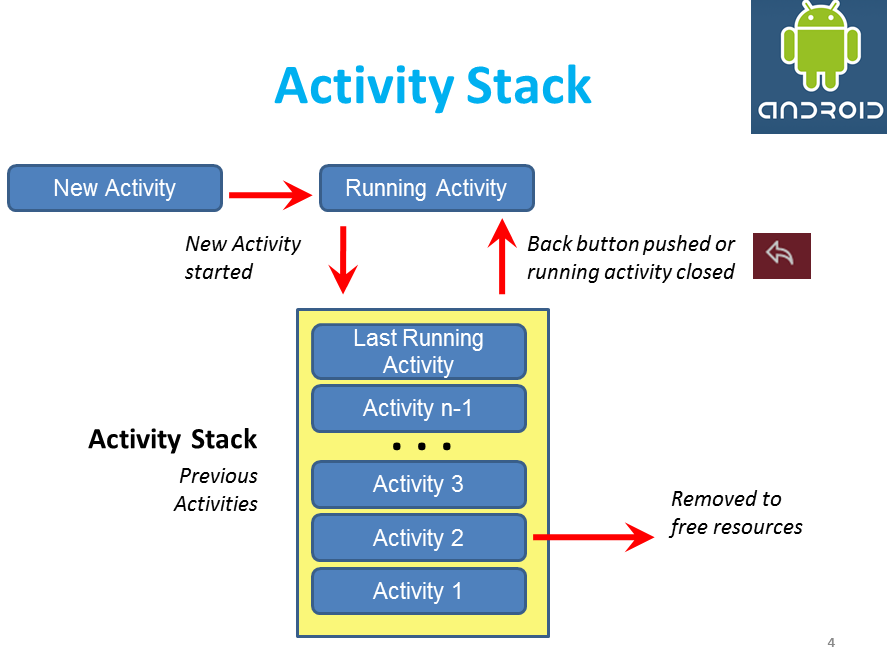
Thường **một Activity cung cấp một của sổ, ở đó ứng dụng sẽ dựng các thành phần UI (User Interface - giao diện người dùng).** Mặc định cửa sổ này là đầy màn hình thiết bị, nhưng có một vài trường hợp riêng sẽ nhỏ hơn và nổi phía trên cửa sổ khác.

Hầu hết các ứng dụng đều sử dụng nhiều màn hình khác nhau, có nghĩa nó sẽ phải có nhiều Activity khác nhau**. Khi một Activity chỉ định là Activity chính, nó sẽ là màn hình đầu tiên khi khởi chạy ứng dụng. Một Activity này lại có thể gọi và kích hoạt một Activity khác.** Ví dụ một Activity hiện thị danh sách các ghi chú, nó gọi một Activity khác để xem nội dung chi tiết của ghi chú.

**Vòng đời Activity**

* **onCreate(Bundle savedInstanceState)**: Được gọi khi hoạt động mới được tạo, tại đây khởi tạo các biến, nạp giao diện layout ..., phương thức này cũng nhận dữ liệu lưu lại trạng thái hoạt động trước đó (với mục đích để phục hồi - savedInstanceState). Sau sự kiện này bao giờ cũng gọi ngay lập tức **onStart()**.
* **onStart()**: Được gọi ngay trước khi Activity hiển thị trên màn hình.
* **onResume()**: Được gọi ngay khi Activity bắt đầu có thể tương tác với người dùng, và Activity nằm trên cùng trong danh sách các Activity của hệ thống. Sau phương thức này là các chức năng của Activity hoạt động dựa trên tương tác của người dùng ..., cho đến khi có một nguyên nhân nào đó mà phương thức OnPause được gọi.
* **onPause()**: Được gọi khi hệ thống sắp kích hoạt một Activity khác, nếu bạn quá tải phương thức này, thường để lưu lại dự liệu thật nhanh để hệ thống còn kích hoạt Activity khác. Ngay sau phương thức này nó sẽ gọi **onResume()** nếu Activity được kích hoạt lại ngay, hoặc gọi **OnStop()** nếu Activity bị ẩn đi.
* **onStop()**: Được gọi khi nó bị ẩn đi. Sau phương thức này, Activity có thể gọi **onRestart()** nếu nó được người dùng kích hoạt lại hoặc gọi **onDestroy()** để hết thúc.
* **onDestroy()**: gọi khi Activity bị hủy hoàn toàn (ví dụ gọi finish(), hoặc người dùng kill Activity)

**Activity Stack**



* Hoạt động theo cơ chế LIFO (LAST IN FIRST OUT)
* Mỗi Activitymới mở lên nó sẽ ở bên trên Activity cũ
* Khi mở activity mới thì activity cũ sẽ xảy ra sự kiện onPause và onStop -Nếu mở lên và chỉ làm che khuất một phần activity cũ thì xảy ra sự kiện onPause -/khi gọi lại activity cũ thì nó phải gọi hàm onResume

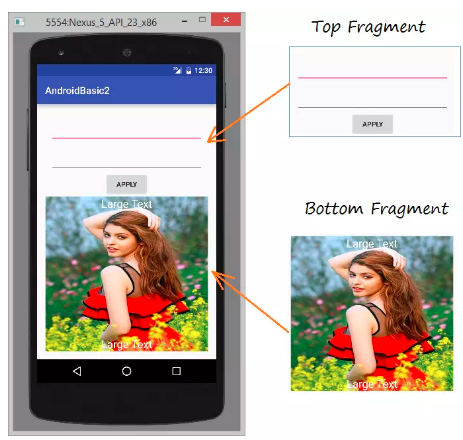
**2. Fragment & life cycle Fragment**

**Fragment là một công cụ** tuyệt vời **xử lý nhiều vấn đề khi chỉ có các lớp Activity.**

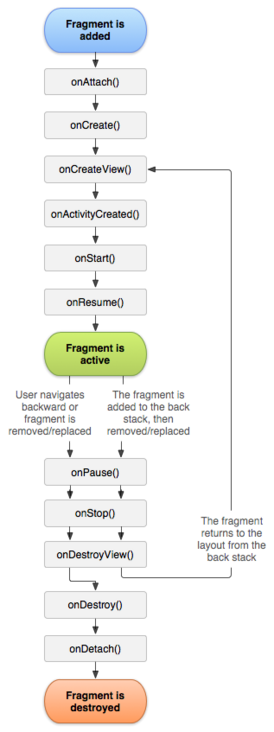
**Fragment cho phép tổ chức các thành phần giao diện người dùng** của dự án cho các thiết bị khác nhau **bằng cách cho bạn khả năng hiển thị nhiều phần giao diện người dùng trên một màn hình lớn** hơn

**Fragment cho phép ứng dụng dễ dàng điều hướng** **và** cung cấp một cách đơn giản để **giao tiếp qua lại giữa các phần khác nhau của ứng dụng.**

**Fragment là một phần giao diện người dùng hoặc hành vi của một ứng dụng**. Fragment **có thể được đặt trong Activity,** nó có thể **cho phép thiết kế activity với nhiều mô-đun**.



**Fragment Lifecycle**



Sơ đồ bắt đầu khi Fragment được gắn vào Activity. Khi đó các *callback* ***onAttach()***, ***onCreate()***, ***onCreateView()***, ***onActivityCreated()***, ***onStart()*** và ***onResume()*** lần lượt được gọi.

Sau khi các *callback* trên được gọi, thì Fragment lúc bấy giờ mới chính thức được xem là đang chạy.

Sau đó, nếu người dùng nhấn nút ***Back***, hay có bất kỳ thao tác gỡ/thay thế (remove/replace) Fragment ra khỏi Activity nào, thì các *callback* ***onPause()***, ***onStop()***, ***onDestroyView()***, ***onDestroy()*** và ***onDetach()*** sẽ được gọi.

Nhưng có một cái hay là, nếu Fragment được đưa vào *Back Stack* kèm với lệnh gỡ/thay thế, thì ***onDestroy()*** và ***onDetach()*** sẽ ***chưa được gọi ngay***. Để khi rơi vào trường hợp sau đó khi Fragment này được hiển thị lại trong *Back Stack*, thì ***onCreateView()*** sẽ được gọi lại.

Các Trạng Thái Chính Trong Vòng Đời Fragment

Bạn sẽ thấy các trạng thái chính này cũng không khác gì với Activity cả, nhưng hãy xem các tình huống cụ thể liên quan với nhau giữa Fragment và Activity như thế nào nhé.

*Hoạt Động (Active Hay Resume)*

Khi Fragment được gắn vào Activity, được nhìn thấy và có thể tương tác được.

*Tạm Dừng (Pause)*

Cũng khá giống với trạng thái [*tạm dừng của Activity*](https://yellowcodebooks.com/2017/10/19/android-bai-28-vong-doi-cua-mot-activity/#Tam_Dung_Pause). Tức là nếu Activity có chứa Fragment này bị che lấp bởi Activity khác (nhưng không bị che hoàn toàn, người dùng vẫn nhìn thấy được Activity bị che lấp, chỉ là không tương tác được), thì cả Activity và Fragment đó đều vào trạng thái tạm dừng.

*Dừng (Stop)*

Cũng giống với [*Activity*](https://yellowcodebooks.com/2017/10/19/android-bai-28-vong-doi-cua-mot-activity/#Dung_Stop), Fragment bị dừng khi bị thành phần nào đó che khuất hoàn toàn. Hay bị gỡ ra khỏi Activity.

Dừng chưa phải là chấm hết cho đời sống của Fragment. Cụ thể là các trạng thái của nó vẫn còn được lưu trữ, để phòng trường hợp Fragment này được trở lại hiển thị cho người dùng.

*Chết (Dead)*

Nếu Fragment bị gỡ ra khỏi Activity, nhưng không được đưa vào *Back Stack* trước đó, thì nó sẽ kết thúc vòng đời. Hoặc khi Activity chứa Fragment này bị gỡ khỏi *Back Stack*, Fragment cũng sẽ chết theo.

**3. Layout and widgets**

## Layouts

**Layout là nơi xác định cấu trúc cho giao diện người dùng trong ứng dụng** của bạn, ví dụ như là Activity.

**Tất cả các thành phần trong Layouts được xây dựng bằng cách sử dụng phân cấp các đối tượng View và ViewGroup**.

**View thường là các đối tượng mà người dùng có thể nhìn thấy và tương tác, còn Viewgroup là một container xác định cấu trúc layout cho View và các đối tượng ViewGroup khác**

**Các đối tượng View thường được gọi là Widget và có thể là một trongcác subclasses như Button, TextView hoặc CheckedTextView,...**

Các đối tượng ViewGroup thường được gọi là layouts có thể là một trong các kiểu cấu trúc layout khác nhau như RelativeLayout hoặc ConstraintLayout,...

### Example: ConstraintLayout, RelativeLayout, LinearLayout, TableLayout, FrameLayout, AbsoluteLayout, GridLayout

**4. Handle events (click, long click..)**

**5. Menus and preferences**

**Menus**

**Menu** trong **android là** một trong những thành phần giao diện quan trọng, cung cấp cho người dùng một vài lựa chọn cho những hành động cụ thể. Trong bài học này, chúng tôi sẽ trình bày hai loại **menu là** options **menu** và context **menu**

Để định nghĩa menu, chúng ta tạo tập tin XML trong thư mục menu (nếu thư mục này chưa tồn tại, bạn phải tạo nó trong thư mục res) và xây dựng menu với các thành phần sau:

**<menu>**: Định nghĩa một Menu, cái mà chứa menu items. Thành phần <menu> phải là root của tập tin XML và có thể chứa một hoặc nhiều <item> và <group>.

**<item>**: Tạo một MenuItem, cái mà sẽ hiển thị trong menu. Thành phần này có thể chứa <menu> bên trong để tạo ra submenu.

**Ví dụ định nghĩa một menu tên menu\_game.xml**

<?xml version="1.0" encoding="utf-8"?>

<menu xmlns:android="http://schemas.android.com/apk/res/android">

<item android:id="@+id/new\_game"

android:icon="@drawable/ic\_new\_game"

android:title= "New Game"

<item android:id="@+id/help"

android:icon="@drawable/ic\_help"

android:title= " Help" />

</menu>

## ****Options menu****

Options menu là nơi bạn nên bao gồm các hành động và các tùy chọn khác có liên quan đến ngữ cảnh hoạt động hiện tại, chẳng hạn như “Tìm kiếm”, “Soạn email” và “Cài đặt”.

**Graphical user interface

Description automatically generated with medium confidence**Vị trí mà options menu hiển thị trên màn hình thiết bị phụ thuộc vào phiên bản của hệ điều hành android đang chạy trên thiết bị ngườ dùng.

**preferences**

**Shared Preferences** là đối tượng**Android** cung cấp cho việc lưu trữ và truy xuất các dữ liệu có kiểu cơ bản như Boolean, string, float, long, and integer trong lập trình các ứng dụng đơn giản.   
Ví dụ như ứng dụng của bạn cho phép người dùng cài đặt các tùy chọn liên quan đến màu chữ hay màu nền của các màn hình. Trong trường hợp này, bạn có thể sử dụng đối tượng **Shared Preferences** để lưu lại các thông tin tùy chọn người dùng thiết lập theo từng cặp key/value và truy xuất sử dụng cho mỗi lần mở lại ứng dụng.

**Shared Preference** sẽ được lưu lại thông qua việc sử dụng chỉ định khóa cho từng giá trị dữ liệu (theo từng cặp key/value), những giá trị key.value này sẽ được tự động ghi vào một tập tin XML được chứa bên trong thư mục của ứng dụng. Những cặp giá trị này sẽ tồn tại và sẵn sàng cho việc sự dụng (đọc và cập nhật) trong suốt phiên làm việc của ứng dụng và được chia sẻ bên trong các thành phần cúa ứng dụng (như activity, service,…). Các giá trị shared preference này không thể truy xuất và sử dụng từ những ứng dụng khác.