



# Lập trình Android

## Bài 2: *Chu trình ứng dụng Android*

Phòng LT & Mạng

<http://csc.edu.vn/lap-trinh-va-cSDL>

2019





# Nội dung

---

1. Các thành phần chính của ứng dụng Android
2. Chu trình ứng dụng Android



# Các thành phần chính của ứng dụng Android

- Dựa trên các thành phần này để tạo ra các ứng dụng
- Mỗi thành phần có những tính năng riêng và chu trình sống riêng. Mỗi chu trình sống định nghĩa một thành phần được tạo ra, chuyển hóa và kết thúc như thế nào
- Có 4 loại thành phần chính
  - **Activity**
  - **Service**
  - **Broadcast Receiver**
  - **Content Provider**





# Activity class

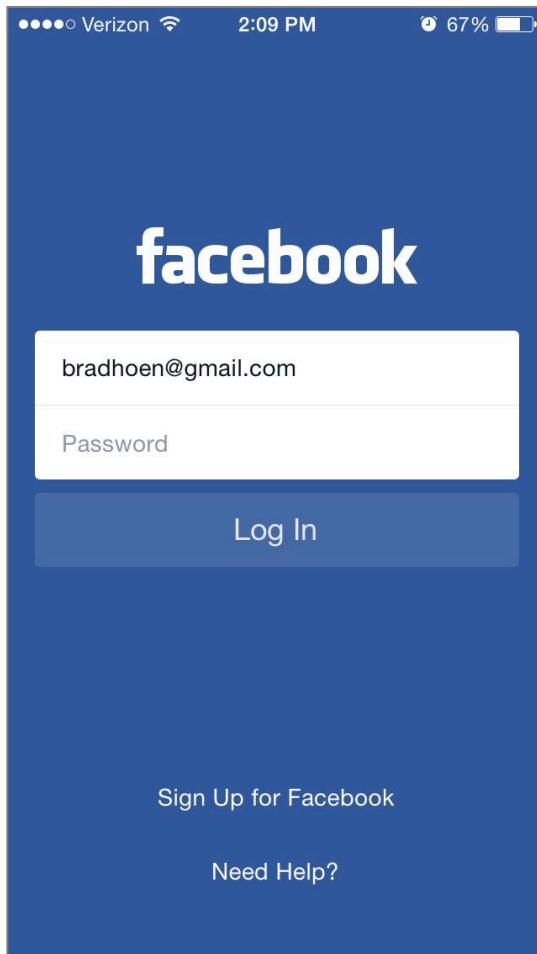
---

- **Activity** tương tự như WindowsForm, trình bày một giao diện đồ họa người dùng (GUI), giúp trình bày dữ liệu/thu thập dữ liệu
- Một ứng dụng Android gồm 1 hoặc nhiều Activity
- Một ứng dụng cần định nghĩa 1 Activity là tác vụ chính (**main task**) hay là đầu vào của ứng dụng (**entry point**). Activity này sẽ được thực thi đầu tiên khi mở ứng dụng
- Một Activity có thể truyền dữ liệu hoặc tín hiệu điều khiển sang 1 Activity khác thông qua cơ chế **Intent**
- Ví dụ: Login activity trình bày một màn hình cho phép nhập username & password. Sau khi click vào Sign In button, quá trình xử lý đăng nhập sẽ thao tác trên dữ liệu thu thập được. Và trước khi Login activity kết thúc (đóng), 1 activity khác sẽ được gọi bắt đầu (mở)

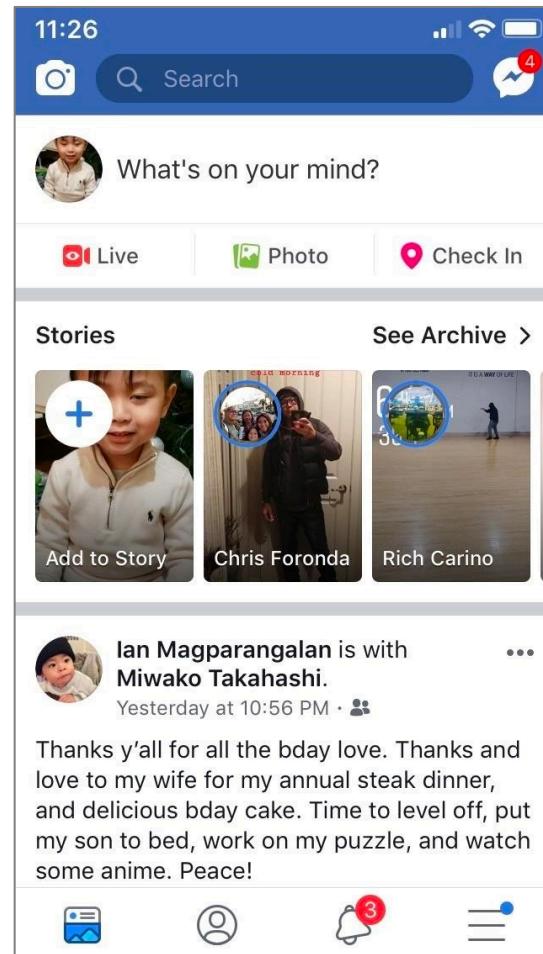


# Ví dụ về 1 app có chứa nhiều activity

Log In Activity



News Feed Activity



Profile Activity





# Service class

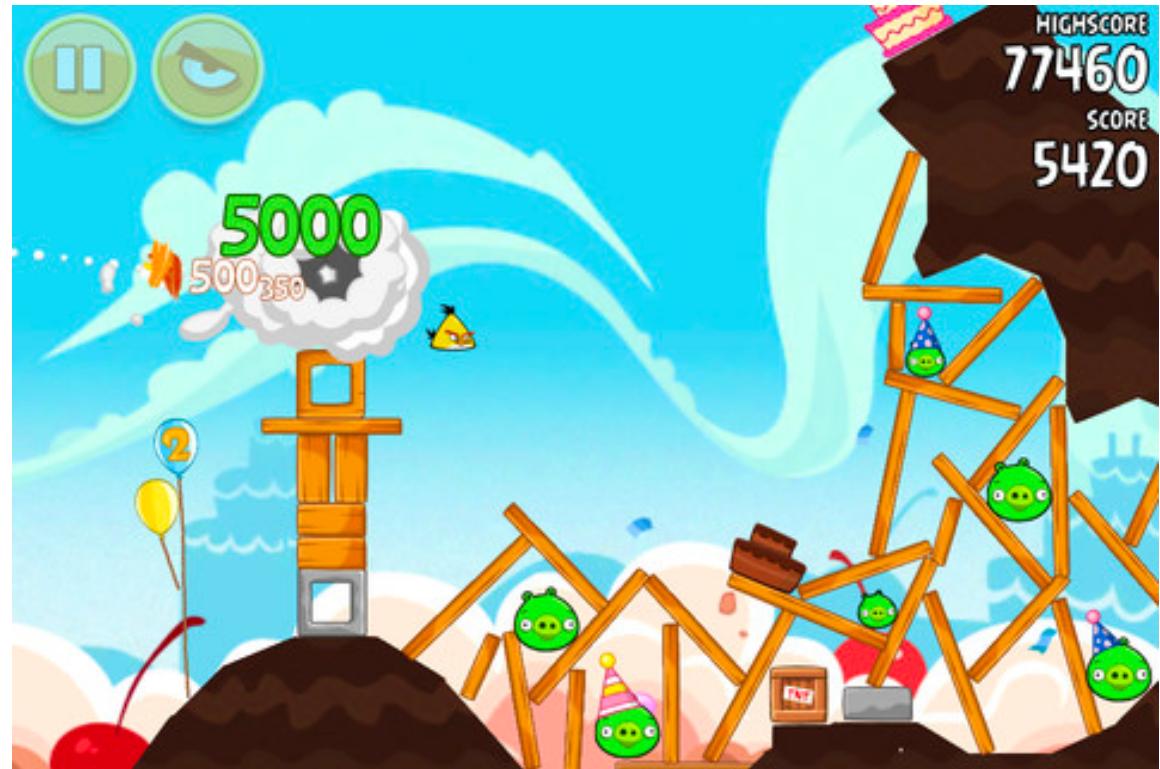
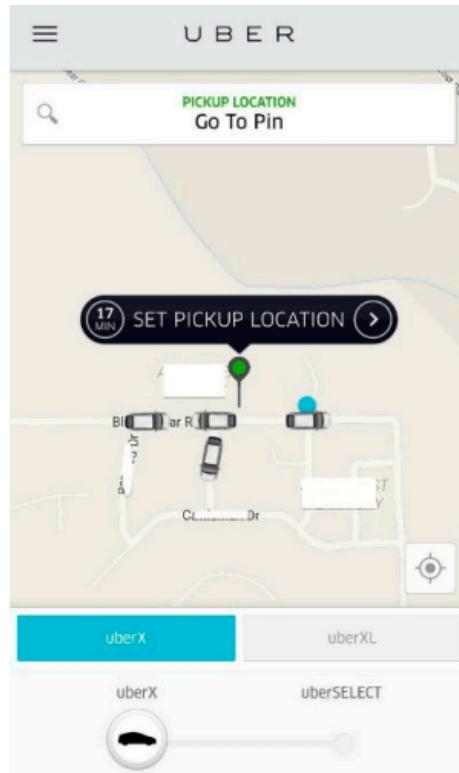
---

- **Service** là một activity đặc biệt, không có giao diện đồ họa người dùng (GUI). Một service có thể hoạt động mà người dùng không nhận thấy sự hiện diện của nó.
- Service tương tự như thread thứ 2, luôn thực thi một công việc ở background, trong một khoảng thời gian không xác định. Phân biệt với thread thứ 1, ở foreground, chính là Activity.
- Các ứng dụng có thể khởi động 1 service hoặc kết nối đến 1 service đã hoạt động.
- Ví dụ: GPS service thực thi ở background để xác định vị trí của bạn từ vệ tinh, trạm viễn thông hay wifi routers. Nó sẽ broadcast thông tin toạ độ có được đến tất cả các ứng dụng đang lắng nghe thông tin này. Một app có thể chọn liên kết với một GPS service đang chạy và sử dụng dữ liệu mà service này cung cấp.



# Ví dụ về Service

## Foreground



## Background

GPS service



Music service



Lập trình Android





# Broadcast Receiver class

---

- **Broadcast Receiver** chuyên lắng nghe và chờ đợi một thông báo kích hoạt toàn hệ thống để thực hiện một số công việc. Thông điệp có thể là: pin yếu, có kết nối wifi, hệ thống vừa khởi động, vừa kết nối xác pin...
- Broadcast Receiver không có giao diện đồ họa người dùng (GUI)
- Thường đăng ký với hệ thống một bộ lọc đóng vai trò là khoá (key). Khi broadcast message trùng với key, receiver này sẽ được kích hoạt.
- Một broad receiver có thể phản hồi bằng cách thực thi một activity hoặc phát một "notification" để gây chú ý đến người dùng



# Minh họa Broadcast Receiver

## Background services

## Broadcast Receiver

## Foreground Activity



Gởi tín hiệu màu CAM



Chờ đợi tin nhắn...

Chỉ chấp nhận tín hiệu  
màu CAM, bỏ qua tất  
cả các tín hiệu khác

Method()

Thực hiện công việc  
sau khi nhận tin nhắn  
màu CAM



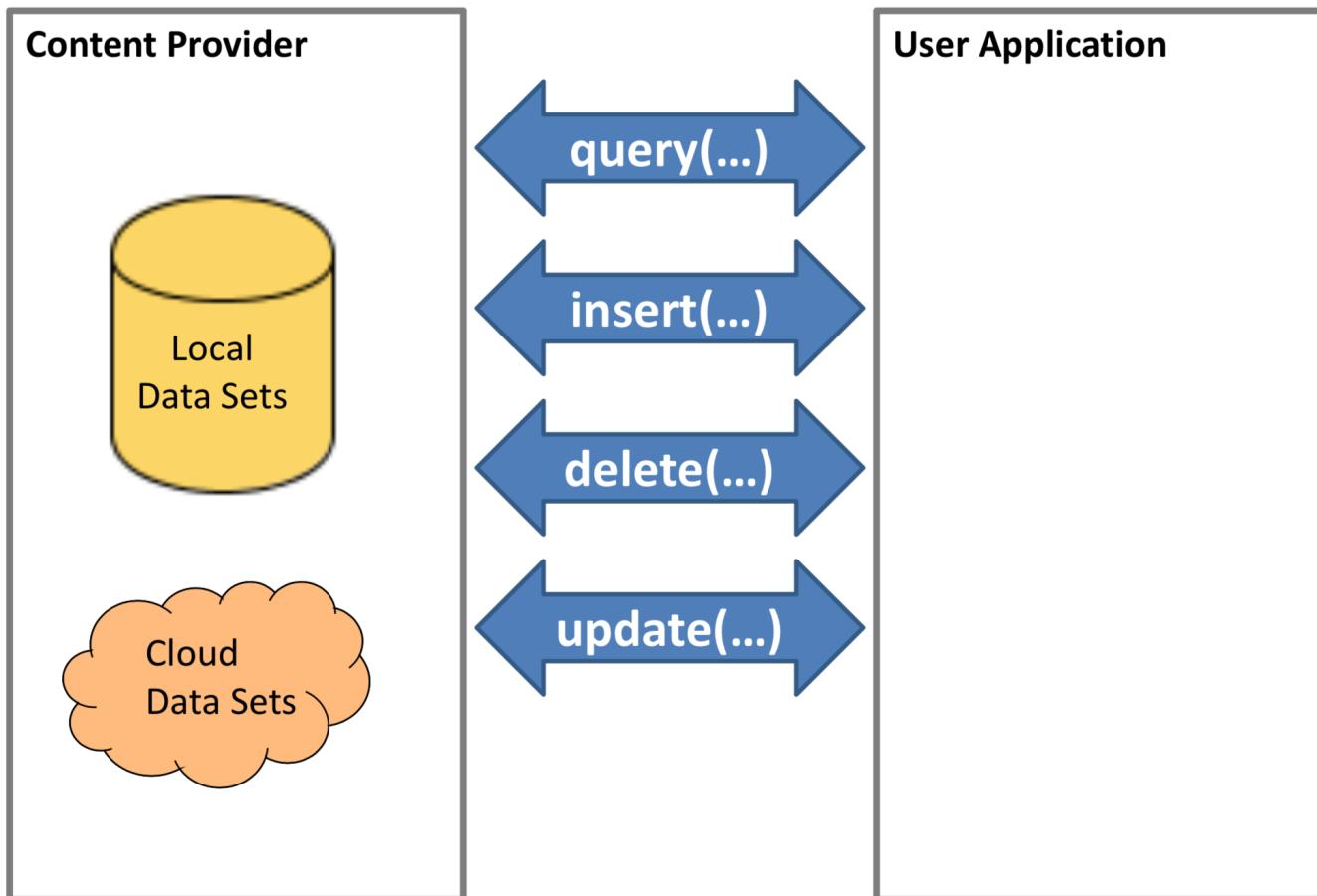
# Content Provider class

---

- **Content Provider** đóng vai trò như một dịch vụ dữ liệu trung tâm nhằm cung cấp dữ liệu đến bất kỳ ứng dụng nào
- Các dữ liệu toàn cục thường là: danh bạ, hình ảnh, tin nhắn, các file audio, email
- Các dữ liệu toàn cục thường được lưu trữ trong SQLite database (người lập trình không cần phải là chuyên gia về SQL)
- Content Provider cung cấp các phương thức chuẩn để các ứng dụng khác có thể lấy dữ liệu hoặc thêm, xoá, sửa dữ liệu



# Minh họa Content Provider

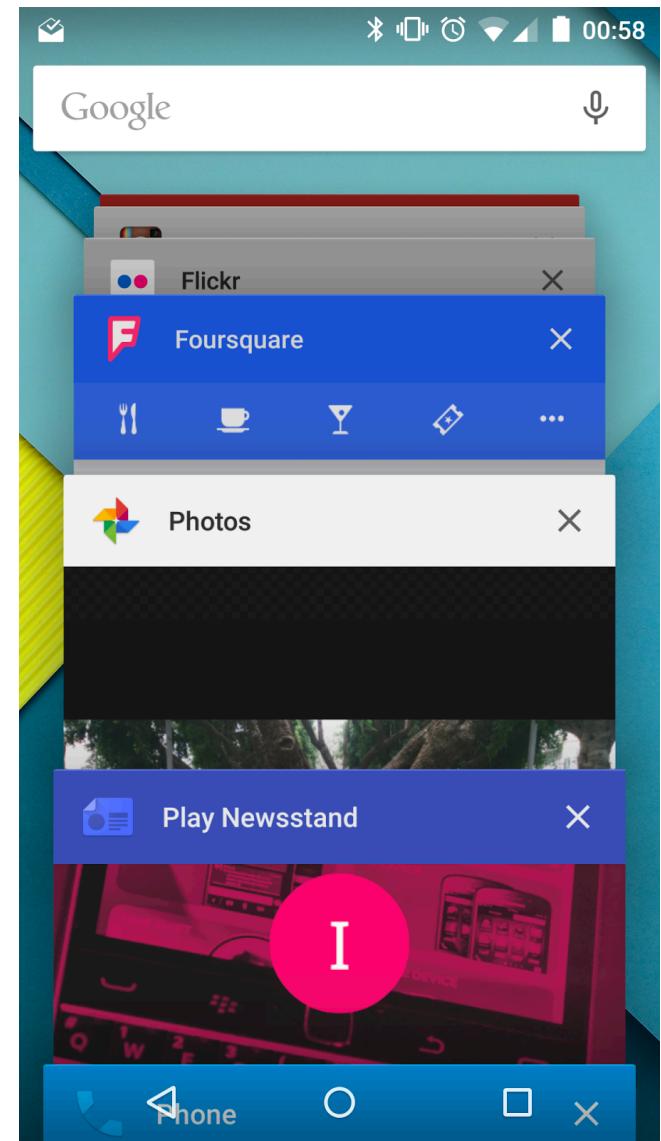


- Content Provider bao bọc/che giấu dữ liệu thực tế, người dùng gián tiếp tương tác với dữ liệu thông qua các công cụ do Content Provider cung cấp



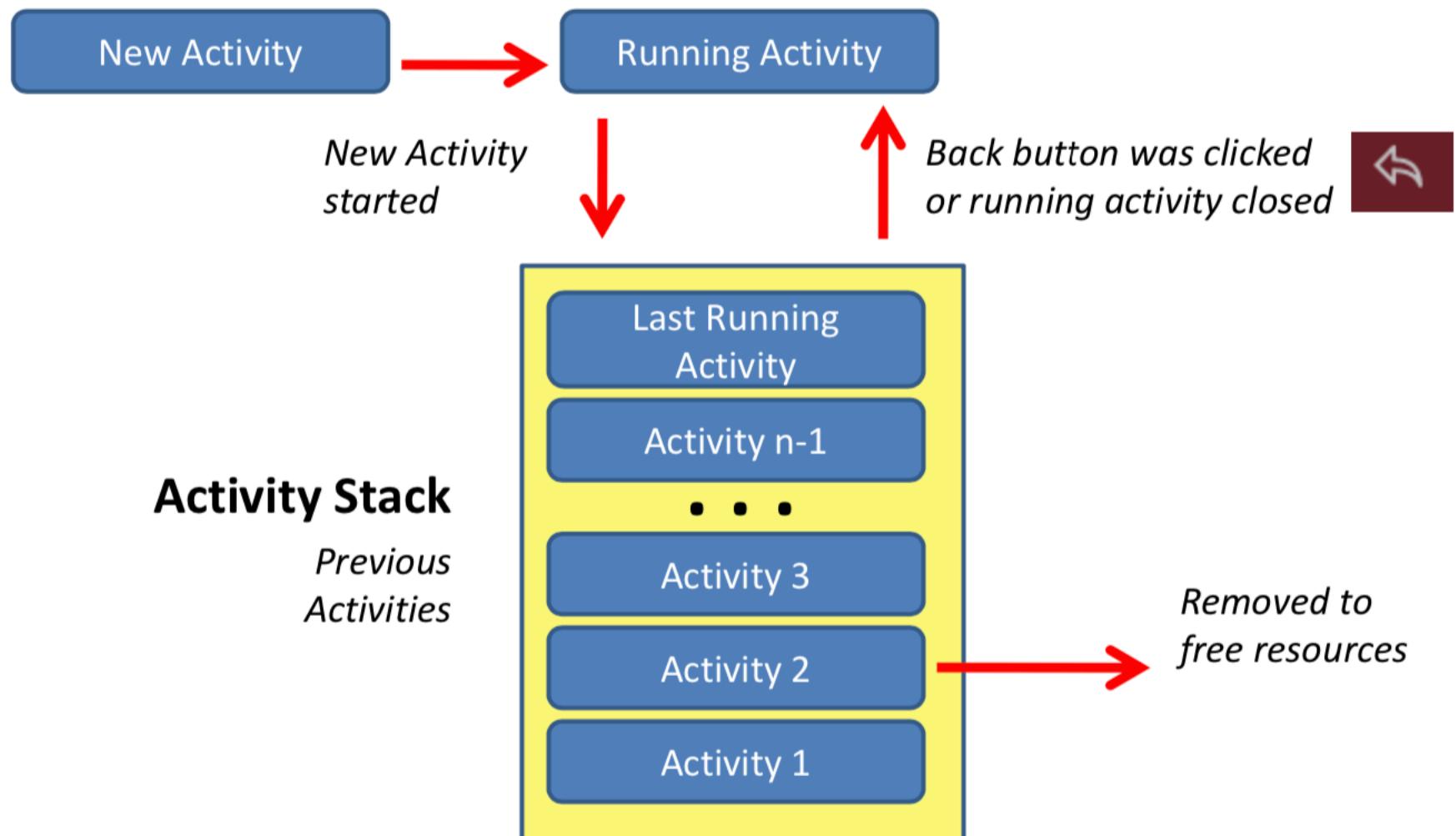
# Activity stack

- Các activity trong hệ thống được lập lịch bằng activity stack
- Khi 1 activity được khởi động, nó nằm trên cùng (top) của stack và trở thành "running activity"
- Người dùng thường nhấn Back button của thiết bị → activity hiện tại sẽ bị huỷ → activity trước đó trong stack sẽ trở thành "active"
- Từ Android 4.0, cung cấp thêm "Recent apps" để người dùng có thể chọn 1 activity trong stack trở thành "active"





# Activity stack (2)





# Các hàm callback

- Một thành phần của ứng dụng Android (activity, service ...) hoạt động theo nguyên tắc chung sau:
  - Có 1 bắt đầu (begin)
  - Có 1 kết thúc (end)
  - Có 1 chuỗi các trạng thái (state) giữa begin và end
  - Các thành phần này có thể dịch chuyển từ “active” sang “inactive” và ngược lại, hoặc đối với activity là “visible” và “invisible”



- Khi xử lý chuyển từ trạng thái này sang trạng thái khác, HĐH thông báo đến ứng dụng nhờ các hàm callback

```
void onCreate()  
void onStart()  
void onRestart()  
void onResume()
```

```
void onPause()  
void onStop()  
void onDestroy()
```



# Life cycle callbacks

Most of your code  
goes here

```
public class ExampleActivity extends Activity {  
    @Override  
    public void onCreate (Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        // The activity is being created.  
    }  
    @Override  
    protected void onStart() {  
        super.onStart();  
        // The activity is about to become visible.  
    }  
    @Override  
    protected void onResume() {  
        super.onResume();  
        // The activity has become visible (it is now "resumed").  
    }  
    @Override  
    protected void onPause() {  
        super.onPause();  
        // Another activity is taking focus (this activity is about to be "paused").  
    }  
    @Override  
    protected void onStop() {  
        super.onStop();  
        // The activity is no longer visible (it is now "stopped")  
    }  
    @Override  
    protected void onDestroy() {  
        super.onDestroy();  
        // The activity is about to be destroyed.  
    }  
}
```

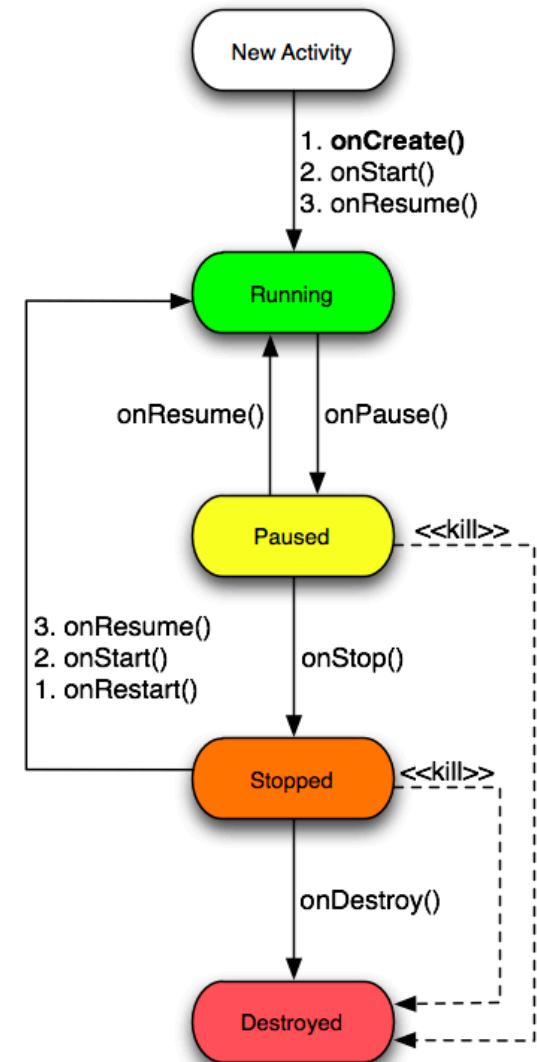
Save your  
important data  
here

# Các trạng thái của Activity và các hàm callback tương ứng



- Một activity có 3 phase cơ bản
  - active/running
    - activity đang ở foreground, ở top của activity stack
    - đang tương tác với người dùng (đang focus)
  - paused
    - mất focus, nhưng vẫn visible với người dùng
    - bị che một phần (hoặc không che phủ toàn bộ màn hình)
  - stopped
    - hoàn toàn bị che lấp bởi một activity khác
    - invisible với người dùng
- Di chuyển từ state này sang state khác sẽ phát sinh những hàm callback tương ứng (xem lược đồ)

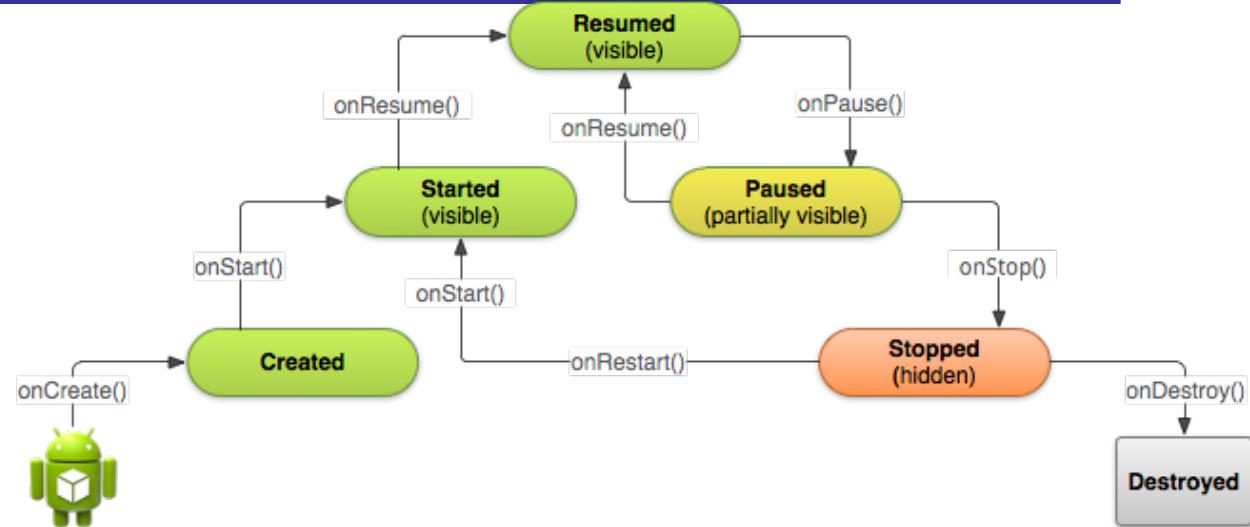
Activity Lifecycle



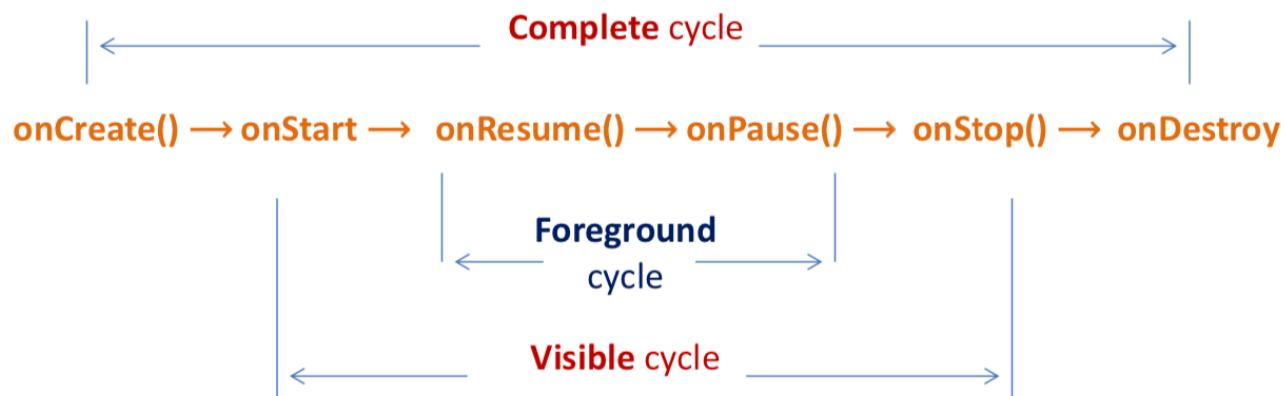


# Activity Life Cycle

- Activity Life Cycle



- Foreground Lifetime





# Các hàm callback

---

- **onCreate()**
  - **Bắt buộc** phải implement cho mỗi activity
  - Được thực thi 1 lần duy nhất trong activity's lifetime
  - Hầu hết source code được viết ở đây
  - Thường khởi tạo các cấu trúc dữ liệu, các đối tượng UI (button, text box, list...)
  - Có thể nhận dữ liệu từ activity trước đó
  - Các hàm đi theo sau: onStart() → onResume() ...
  - Activity đã được khởi động nhưng chưa visible với người dùng
- **onStart()**
  - Activity đã visible với người dùng, nhưng người dùng chưa thể tương tác với activity
  - Thường dùng để đăng ký BroadcastReceiver hay thay đổi UI



# Các hàm callback

---

- **onResume()**
  - Activity đã ở foreground và sẵn sàng nhận tương tác từ người dùng
  - Có thể bắt đầu các animation hay truy cập vào các thành phần thiết bị như camera
- **onPause()**
  - Tương ứng với **onResume()**
  - Được gọi khi hệ thống chuyển điều khiển sang 1 activity khác
  - Nên được dùng để lưu các dữ liệu hoặc dừng các công việc/tác vụ đang dang dở
  - Activity tiếp theo phải chờ trạng thái này hoàn tất
  - Theo sau là hàm `onResume()` nếu activity trở lại foreground, hoặc sau là hàm `onStop()` nếu activity là invisible với người dùng
  - Một paused activity có thể bị kill bởi hệ thống



# Các hàm callback

---

- **onStop()**
  - Tương ứng với onStart()
  - Thường dùng để huỷ bỏ các đối tượng đã khai báo trong onStart(), ví dụ: trong onStart() đăng ký 1 BroadcastReceiver → trong onStop() nên huỷ đăng ký BroadcastReceiver tương ứng
- **onDestroy()**
  - Tương ứng với onCreate()
  - Hàm này được gọi khi lệnh finish() được gọi từ activity hoặc từ hệ thống khi cần dọn dẹp bộ nhớ
  - Thường dùng để “dọn dẹp”. Ví dụ: activity có 1 background thread dùng để download 1 file từ network, thread này có thể được tạo ra trong onCreate() và cần được huỷ bỏ trong onDestroy()
- **onRestart()**
  - Được gọi khi activity đã stopped, sau đó được khởi động trở lại
  - Thường không cần implement phương thức này



# Killable states

---

- HĐH Android có thể tắt (kill) một ứng dụng (killable app) khi cần tài nguyên cho một ứng dụng có độ ưu tiên cao hơn
- Một app gọi là killable khi activity của nó đang ở các trạng thái **onPause()**, **onStop()**, **onDestroy()**
- **onPause()** là trạng thái duy nhất đảm bảo có thể hoàn tất các tác vụ trước khi activity bị huỷ
- Nên dùng **onPause()** để lưu các dữ liệu trước khi activity bị huỷ



## Q&A

---

