LẬP TRÌNH ANDROID

Giảng Viên: Phùng Mạnh Dương

Trường ĐH Công nghệ, ĐHQGHN

CHƯƠNG 2
CĂN BẢN VỀ ỨNG DỤNG

4 khối căn bản tạo thành ứng dụng Android

- Ứng dụng được tạo thành từ nhiều thành phần kết hợp với nhau.
- Android khởi phát và thực thi chúng khi cần.
- Mỗi thành phần có mục đích và API riêng.

4 khối căn bản tạo thành ứng dụng Android

- Activity: Cung cấp giao diện người dùng
- Service: chạy ngầm và thực thi các tác vụ cần nhiều thời gian
- Broadcastreceiver: Lắng nghe và xử lý các sự kiện xảy ra trên thiết bị
- Content provider: cho phép nhiều ứng dụng lưu và chia sẻ dữ liệu

Activity

- Trong ứng dụng Android, Activity đóng vai trò là một màn hình, nơi người dùng có thể tương tác với ứng dụng, ví dụ: chụp hình, xem bản đồ, gửi mail…
- 1 activity thông thường tập trung vào 1 tác vụ đơn nhất mà người dùng có thể thực hiện.
- Một ứng dụng có thể có một hoặc nhiều Activity, Activity được khởi chạy đầu tiên khi ứng dụng hoạt động được gọi là "MainActivity".
- Activity có thể hiển thị ở chế toàn màn hình, hoặc ở dạng cửa sổ với một kích thước nhất định.
- Các Activity có thể gọi đến các Activity khác, Activity được gọi sẽ nhận được tương tác ở thời điểm đó.
- VD: The Answer

Service

- Service thực hiện ở chế độ ngầm và thường không cần giao diện hiển thị. Được sử dụng để:
  - Thực thi các tác vụ cần nhiều thời gian
  - Hỗ trợ tương tác với các tiến trình khác
- Service có thể được khởi chạy và hoạt động xuyên suốt ngay cả khi ứng dụng không hoạt động.
- Một số tác vụ cần thực hiện bằng Service:
  - Trình diễn các tập tin đa truyền thông như nhạc, phim…
  - Kết nối và thực hiện tải các nội dung thông qua Internet
  - Truy xuất đọc ghi tập tin
- VD: MusicPlayer

Broadcast Receiver

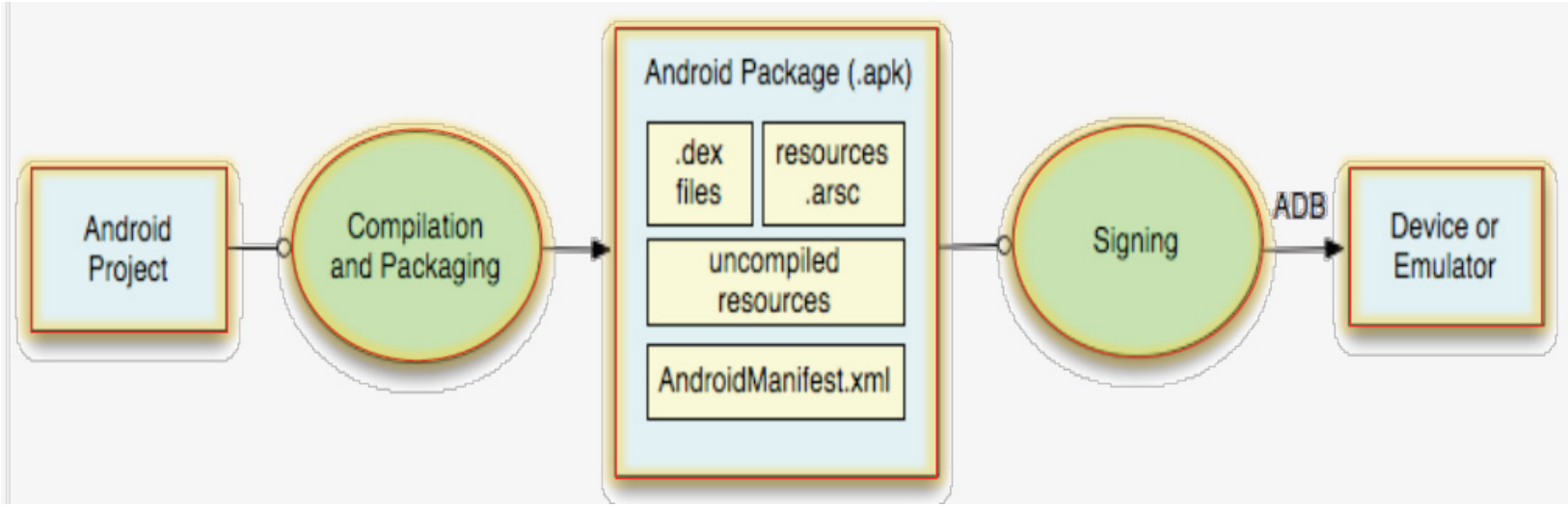
- Lắng nghe và trả lời các sự kiện.
  - Các sự kiện được biểu diễn bởi lớp Intent rồi sau được truyền quảng bá.
  - BroadcastReceiver nhận và trả lời các sự kiện mà chúng được khai báo để xử lý.
- Không có giao diện nhưng có thể thực hiện thông báo qua thanh trạng thái.
- Broadcast Receiver lắng nghe sự kiện ở hai dạng:
  - Hệ thống: các thông báo được truyền trực tiếp từ hệ thống như: tắt màn hình, pin yếu, thay đổi kết nối…
  - Ứng dụng: Truyền thông báo đến các thành phần trong ứng dụng như: khởi động Service, tải nội dung đến ứn dụng…
- VD: Messaging

Content Provider

- Lưu trữ và chia sẻ dữ liệu giữa các ứng dụng
  - Lưu trữ: ở nhiều dạng như SQLite, tập tin, tài nguyên Web hoặc bất kì thư mục lưu trữ nào.
  - Xử lý các giao tiếp ở mức thấp giữa các tiến trình giúp các ứng dụng chạy ở các tiến trình khác nhau có thể giao tiếp và trao đổi dữ liệu một cách an toàn.
- Có thể sử dụng Content Provider để xây dựng các ứng dụng sử dụng chung nguồn tài nguyên hoặc sử dụng riêng.
- Trong Android, một số Content Provider được xây dựng sẵn:
  - Danh bạ
  - Tài nguyên đa truyền thông
- VD: Webrowser

Giới thiệu
MapLocation App

Xây dựng 1 ứng dụng



Tham khảo:
http://developer.android.com/guide/developing/building

Tạo 1 ứng dụng

1. Xác định các tài nguyên
2. Cài đặt các lớp ứng dụng
3. Đóng gói ứng dụng
4. Cài đặt và thực thi ứng dụng (cho debug và test)

1. Xác định tài nguyên

- Tài nguyên là các nguồn không chứa mã nguồn như Layout, Strings, Images, Menus, & animations
- Cho phép ứng dụng tùy biến với các thiết bị và người dùng khác nhau mà không cần biên dịch lại mã nguồn.
- Tham khảo: http://developer.android.com/guide/topics/resources

1. Tài nguyên: String

- Types: String, String Array
- Được lưu tại res/values/*.xml
- Được trình bày theo cú pháp XML, VD

  <string name="hello">Hello World!</string>
- Truy cập bởi các tài nguyên khác: @string/string_name
- Truy cập bởi Java: R.string.string_name
- VD trên Emulator: MapLocation
  - String bằng tiếng Anh và Ý.

1. Tài nguyên: Layout file

- Layout giao diện người dùng định nghĩa trong 1 file XML.
- Lưu trữ tại: res/layout/*.xml
- Truy cập bằng Java: R.layout.layout_name
- Truy cập bởi các tài nguyên khác: @layout/layout_name
- Có thể định nghĩa layout file riêng cho hướng màn hình, kích thước màn hình …
- VD: MapLocation
  - Quay hướng màn hình

1. Tài nguyên: R.java

- Tại thời điểm biên dịch, các tài nguyên được sử dụng để tạo lớp R.java
- Mã nguồn Java sử dụng lớp R để truy cập các tài nguyên
- VD: MapLocation

app\build\generated\source\r\debug\course\examples\MapLocation

```
public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int ic_launcher=0x7f020000;
    }
    public static final class id {
        public static final int RelativeLayout1=0x7f050000;
        public static final int location=0x7f050001;
        public static final int mapButton=0x7f050002;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int location_string=0x7f040001;
        public static final int show_map_string=0x7f040000;
    }
}
```

2. Cài đặt các lớp

- Thường liên quan tới ít nhất 1 activity
- Code khởi tạo Activity thực thi trong onCreate()
- Quá trình thực thi trong onCreate():
  - Khôi phục trạng thái đã lưu
  - Thiết lập các thành phần nội dung
  - Khởi phát các thành phần giao diện
  - Liên kết các thành phần giao diện tới các xử lý.
- VD: MapLocation

```
@Override
protected void onCreate(Bundle savedInstanceState) {

    // Required call through to Activity.onCreate()
    // Restore any saved instance state
    super.onCreate(savedInstanceState);

    // Set content view
    setContentView(R.layout.main);

    // Initialize UI elements
    final EditText addrText = (EditText) findViewById(R.id.location);
    final Button button = (Button) findViewById(R.id.mapButton);
```

3. Đóng gói ứng dụng

- Hệ thống đóng gói các thành phần ứng dụng và tài nguyên vào 1 file .apk
- Nhà phát triển chỉ định các thông tin cần thiết cho ứng dụng trong 1 file gọi là AndroidManifest.xml
- Thông tin bao gồm:
  - Tên ứng dụng
  - Các thành phần
  - Thông tin khác:
    - Quyền truy cập
    - Tính năng ứng dụng
    - Min API level
- VD: Maplocation

4. Cài đặt và thực thi

- Sử dụng Android Studio

CHƯƠNG 3
Lớp Activity

Activity

- Cung cấp UI cho người dùng tương tác
- 1 activity thông thường **tập trung vào 1 tác vụ đơn nhất** mà người dùng có thể thực hiện:
  - Đọc 1 email
  - Hiển thị 1 màn hình login
- Ứng dụng thường bao gồm một vài activity.

Di chuyển qua các Activity

- Android hỗ trợ 1 số cách thức:
  - Qua ứng dụng
  - Nút back
  - Nút home

Activity LifeCycle

- Khi 1 ứng dụng thực thi, Activity có thể được khởi tạo, tạm dừng, tiếp tục, hoặc huy khi cần.
- Một số hoạt động phụ thuộc hành vi người dùng
- Một số hoạt động phụ thuộc Android
  - VD: Android có thể kill activity khi cần tài nguyên.

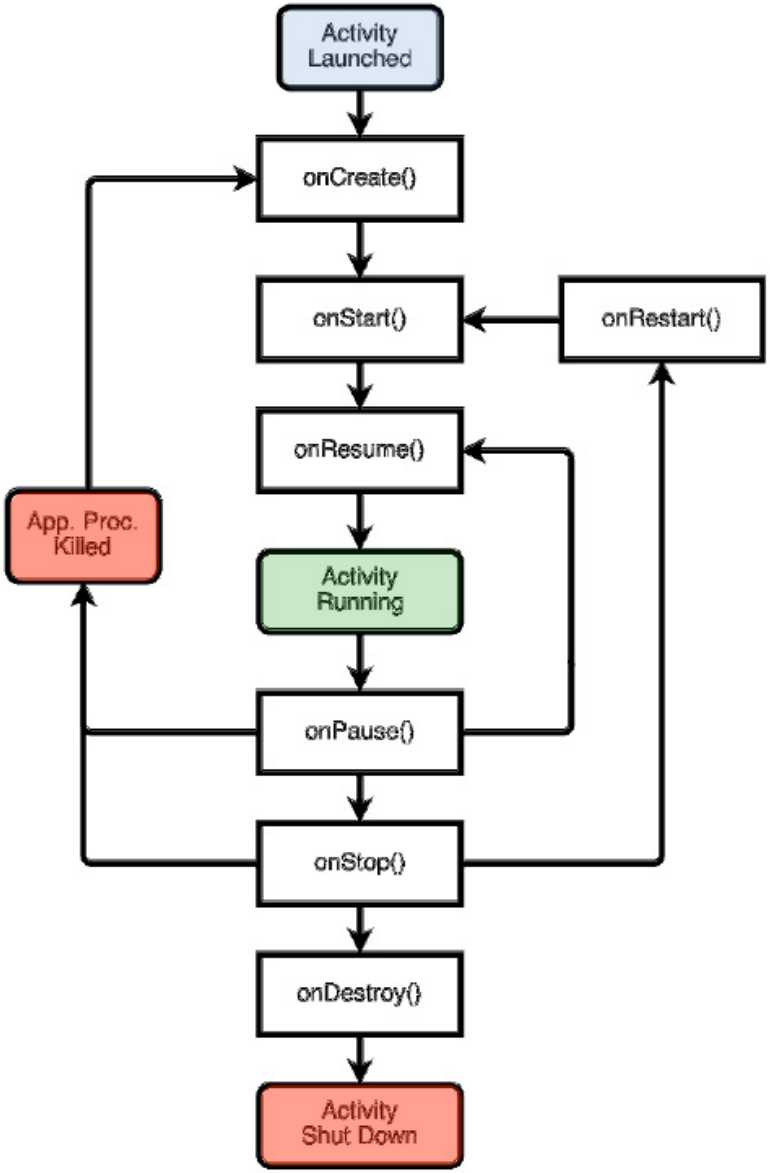
Các trạng thái của Activity

- Resumed/Running: Hiển thị với người dùng để tương tác
- Paused: Tạm dừng, vẫn hiển thị nhưng người dùng không thể tương tác
- Stopped: Không hiển thị

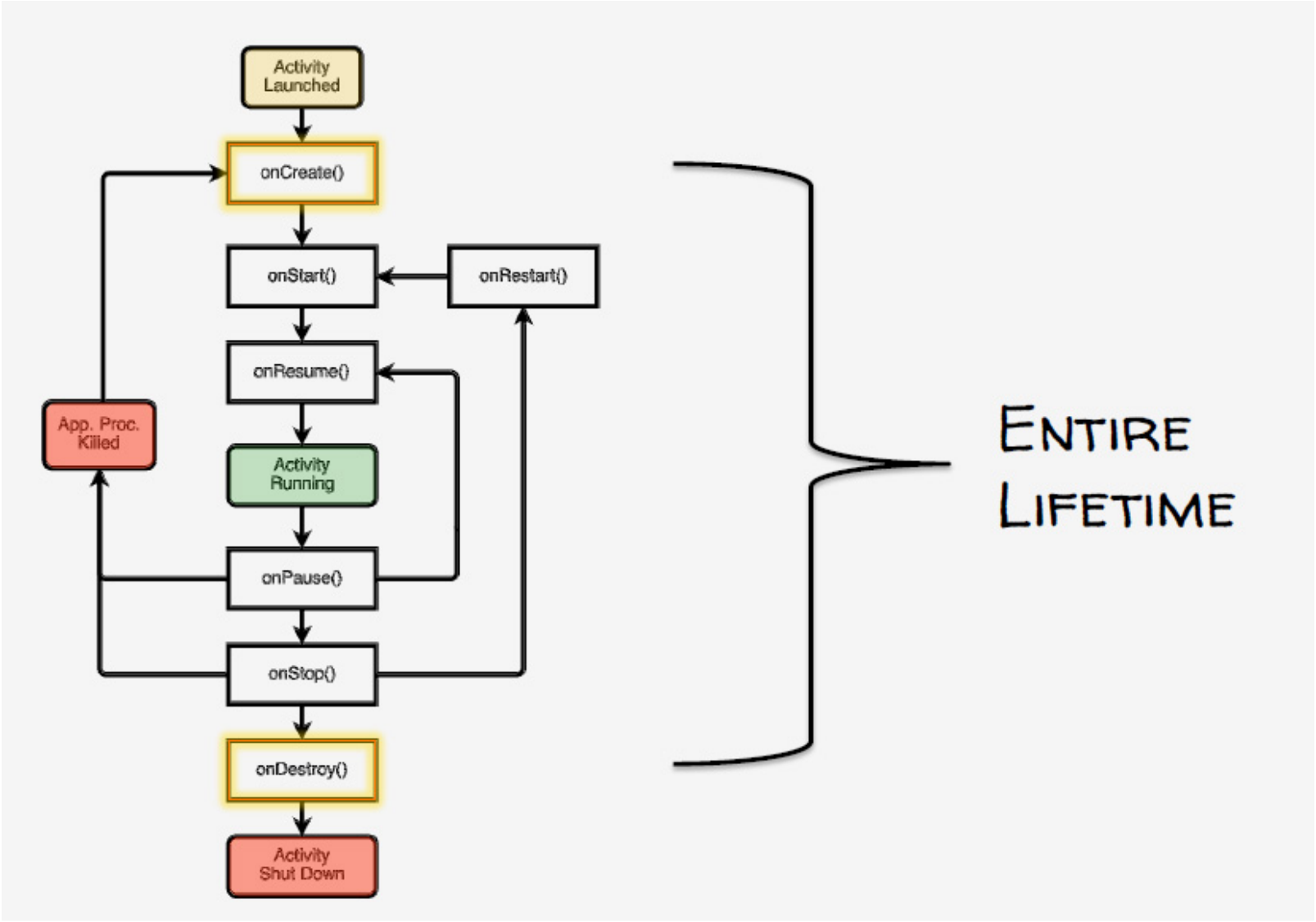
Các phương thức lifecycle của Activity

- Android thông báo những thay đổi về trạng thái của ứng dụng bằng cách gọi các phương thức đặc trưng cho Activity
- Một số phương thức điển hình:
  - protected void onCreate (Bundle savedInstanceState)
  - protected void onStart()
  - protected void onResume()
  - protected void onPause()
  - protected void onRestart()
  - protected void onStop()
  - protected void onDestroy()

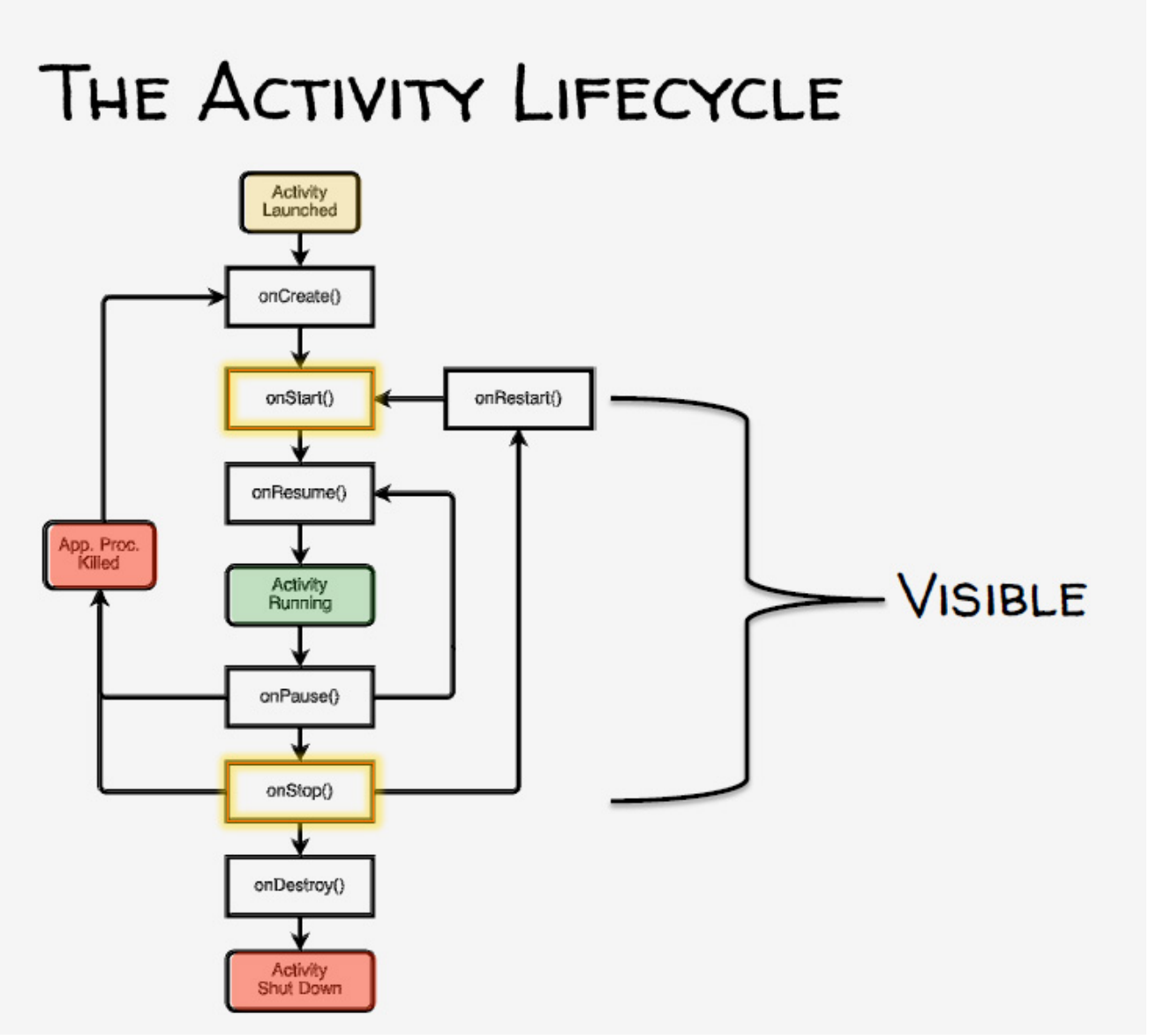
Activity lifecycle



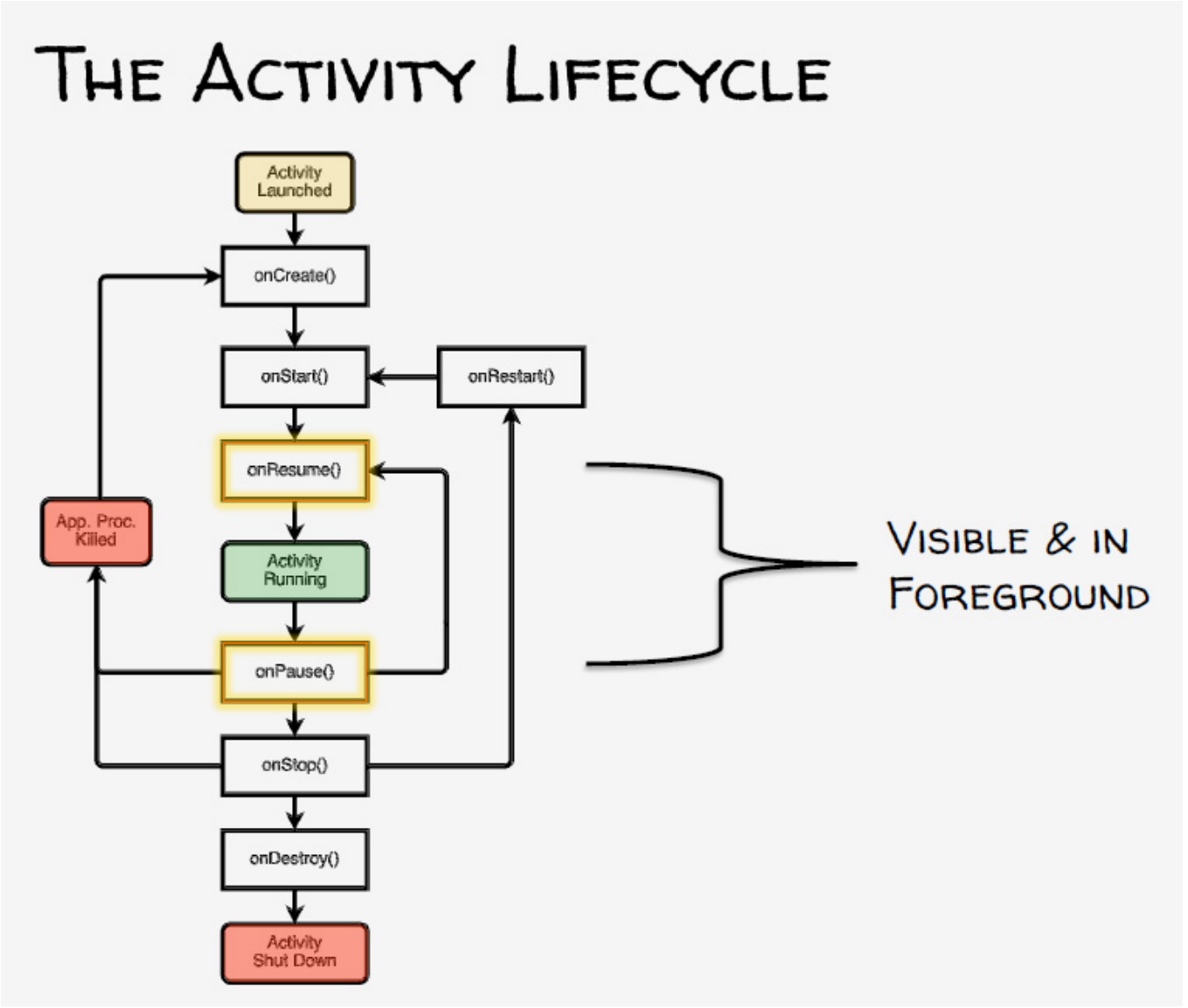
Activity lifecycle



Activity lifecycle



Activity lifecycle



onCreate()

- Được gọi khi khởi tạo Activity
- Thiết lập trạng thái ban đầu
  - Gọi super.onCreate()
  - Thiết lập content view
  - Lấy các tham chiếu các thành phần giao diện nếu cần
  - Cấu hình các thành phần giao diện nếu cần
- VD:

```
16  public class MapLocation extends Activity {
17
18      private final String TAG = "MapLocation";
19
20      @Override
21      protected void onCreate(Bundle savedInstanceState) {
22
23          // Required call through to Activity.onCreate()
24          // Restore any saved instance state
25          super.onCreate(savedInstanceState);
26
27          // Set content view
28          setContentView(R.layout.main);
29
30          // Initialize UI elements
31          final EditText addrText = (EditText) findViewById(R.id.location);
32          final Button button = (Button) findViewById(R.id.mapButton);
33
```

onRestart()

- Được gọi nếu Activity đã bị dừng và chuẩn bị chạy lại.
- Các hành động thường thực thi:
  - Những xử lý đặc biệt cần thiết sau khi ứng dụng đã bị dừng

onStart()

- Activity chuẩn bị hiển thị với người dùng.
- Các hành động thường thực thi:
  - Khởi phát các hành động cần thiết cho tương tác người dùng.
  - Nạp trạng thái ứng dụng.

onPause()

- Tập trung vào chuyển sang 1 activity khác
- Các hành động thường thực thi:
  - Dừng các hành động hiển thị với người dùng
  - Lưu trạng thái

onStop()

- Ứng dụng không còn hiển thị với người dùng nữa (nhưng vẫn có thể khởi động lại)
- Các hành động thường thực thi:
  - Cache state

onDestroy()

- Activity chuẩn bị hủy
- Các hành động thường thực thi:
  - Giải phóng tài nguyên

Khởi phát Activity

- Tạo một đối tượng Intent chỉ định Activity cần khởi phát
- Truyền Intent đó vào các phương thức:
  - startActivity()
  - startActivityForResult(): thực thi 1 hàm callback khi ứng dụng được gọi kết thúc và trả về kết quả
- VD:

```
// Initialize UI elements
final EditText addrText = (EditText) findViewById(R.id.location);
final Button button = (Button) findViewById(R.id.mapButton);

// Link UI elements to actions in code
button.setOnClickListener(new OnClickListener() {

    // Called when user clicks the Show Map button
    public void onClick(View v) {
        try {

            // Process text for network transmission
            String address = addrText.getText().toString();
            address = address.replace(' ', '+');

            // Create Intent object for starting Google Maps application
            Intent geoIntent = new Intent(
                    android.content.Intent.ACTION_VIEW, Uri
                            .parse("geo:0,0?q=" + address));

            // Use the Intent to start Google Maps application using Activity.startActivity()
            startActivity(geoIntent);

        } catch (Exception e) {
            // Log any error messages to LogCat using Log.e()
            Log.e(TAG, e.toString());
        }
    }
});
```

Thay đổi cấu hình

- Cấu hình thiết bị có thể thay đổi khi đang hoạt động: hướng màn hình, ngôn ngữ...
- Khi cấu hình thay đổi, Android thường hủy (kill) activity hiện tại và restart nó.
- Việc restart thường nhanh, vì vậy nếu cần:
  - onSaveInstanceState()
  - onRestoreInstanceState()