# Activities

## Contents


- 1. Definition
- 2. Creating an Activity
- 3. Starting and shutting down an Activity
- 4. Managing the Activity Lifecycle
- 5. Tasks and Back Stack
- 6. Practice

## Contents


- 1. Definition
- 2. Creating an Activity
- 3. Starting and shutting down an Activity
- 4. Managing the Activity Lifecycle
- 5. Tasks and Back Stack
- 6. Practice

## Definition

- Là thành phần cung cấp giao diện màn hình người dùng có thể tương tác.
- Mỗi Activity được cung cấp 1 cửa sổ để thể hiện giao diện người dùng.
- Một ứng dụng bao gồm nhiều Activity có quan hệ với nhau.
- Một ứng dụng có 1 “main” Activity – Là Activity hiển thị đầu tiên khi chạy ứng dụng.
- Mỗi Activity có thể gọi ra 1 Activity khác để thực hiện 1 action khác.

## Contents


- 1. Definition
- 2. Creating an Activity
- 3. Starting and shutting down an Activity
- 4. Managing the Activity Lifecycle
- 5. Tasks and Back Stack
- 6. Practice

## Creating an Activity

- Tạo class kế thừa “android.app.Activity”
- Implement các callback method mà system sẽ gọi khi Activity thay đổi trạng thái
  - 2 callback method quan trọng nhất: onCreate() và onPause()
- onCreate()
  - Bắt buộc phải implement.
  - Được gọi bởi system khi tạo Activity
  - Thực hiện khởi tạo cho các component
    - Quan trọng nhất là thiết đặt layout bằng cách gọi method setContentView()
- onPause()
  - Được gọi bởi system khi chuyển sang 1 Activity khác.
  - Tiến hành commit (lưu lại) các thay đổi trong session người dùng (do người dùng có thể quay lại Activity này)

## Implementing a user interface

- UI trong Android được cung cấp thông qua hệ thống các View như Text field, button, image, …
- Các View quản lý 1 vùng không gian trong window của Activity và có thể phản hồi lại các thao tác của user
- Android cung cấp sẵn 1 số View để Developer thiết kế và bố trí layout
  - "Widget“: Là view cung cấp các phần tử nhìn thấy được trên màn hình như button, text field, checkbox, image, …
  - "Layout“: Là view thiết đặt cách bố trí cho các view khác. Ví dụ: Linear layout, grid layout, relative layout, …
  - Developer có thể tạo ra các widget hay layout riêng của mình bằng cách kế thừa “android.view.View” hay “android.view.ViewGroup”.
- Cách phổ biến nhất để định nghĩa layout là thông qua file XML.
  - Ưu điểm: Tách biệt View và Controller/Model

## Declaring the activity in the manifest

- Cần khai báo Activity trong file manifest file để Activity có thể truy cập bởi system.
- Định nghĩa bằng phần tử <activity> trong manifest file

```
<manifest ... >
  <application ... >
      <activity android:name=".ExampleActivity" />
      ...
  </application ... >
  ...
</manifest >
```

- Một vài thuộc tính chỉ định cho <activity>
  - Name: Bắt buộc, chỉ định class name của Activity
  - Label, Icon, Theme and Style: Các chỉ định cho UI
  - Intent filter: Quy định các component khác của ứng dụng kích hoạt Activity này như thế nào

# Using intent filters

- Sử dụng phần tử <intent-filter> để quy định Intent filter.
- Khi tạo mới ứng dụng, Android SDK tự động thêm Intent filter mặc định sau

```
<activity android:name=".ExampleActivity" android:icon="@drawable/app_icon">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Main entry point của ứng dụng

Activity này được liệt vào application launcher của hệ thống (cho phép người dùng chạy Activity này)

- Trong 1 ứng dụng chỉ có 1 activity có action là "main" và category là "launcher"
- Không thiết đặt intent filter cho các Activity không public ra ngoài (cho các ứng dụng khác)

## Contents


- 1. Definition
- 2. Creating an Activity
- 3. Starting and shutting down an Activity
- 4. Managing the Activity Lifecycle
- 5. Tasks and Back Stack
- 6. Practice

# Starting an Activity

- Sử dụng method startActivity()
  - Truyền vào đối số là Intent object: Mô tả Activity muốn khởi động
- Ví dụ

```
Intent intent = new Intent(this, SignInActivity.class);
startActivity(intent);
```

Tên Activity muốn khởi động

➔ Tại sao phải truyền Intent làm đối số?

# Starting an Activity (cont.)

- Trong trường hợp:
  - Gọi ra Activity của ứng dụng khác
  - Truyền data khi gọi ra Activity khác

➔Sử dụng Intent để mô tả Activity sẽ được gọi.

Ví dụ

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.putExtra(Intent.EXTRA_EMAIL, recipientArray);
startActivity(intent);
```

Ứng dụng gửi mail sẽ hiển thịcác địa chỉ mail tại item "**TO**" trên màn hình gửi mail.

# Starting an activity for a result

- Nhiều trường hợp Activity được gọi trả về kết quả xử lý ➔ Cách implement:
  1. Sử dụng method **startActivityForResult**() thay cho startActivity()
  2. Implement call back method: **onActivityResult**()
- Ví dụ

```
private void pickContact() {
    Intent intent = new Intent(Intent.ACTION_PICK, Contacts.CONTENT_URI);
    startActivityForResult(intent, PICK_CONTACT_REQUEST);
}
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == Activity.RESULT_OK && requestCode == PICK_CONTACT_REQUEST) {
        Cursor cursor = getContentResolver().query(data.getData(), new String[]
  { Contacts.DISPLAY_NAME }, null, null, null);
        if (cursor.moveToFirst()) {
            int columnIndex = cursor.getColumnIndex(Contacts.DISPLAY_NAME);
            String name = cursor.getString(columnIndex);
            // Do something with the selected contact's name...
        }
    }
}
```

Intent object dùng để gọi ra Activity lấy thông tin Contact

Hằng số private quy định request là lấy thông tin Contact

Request thành công hay ko

Truy vấn thông tin content provider của Contact

## Shutting down an Activity

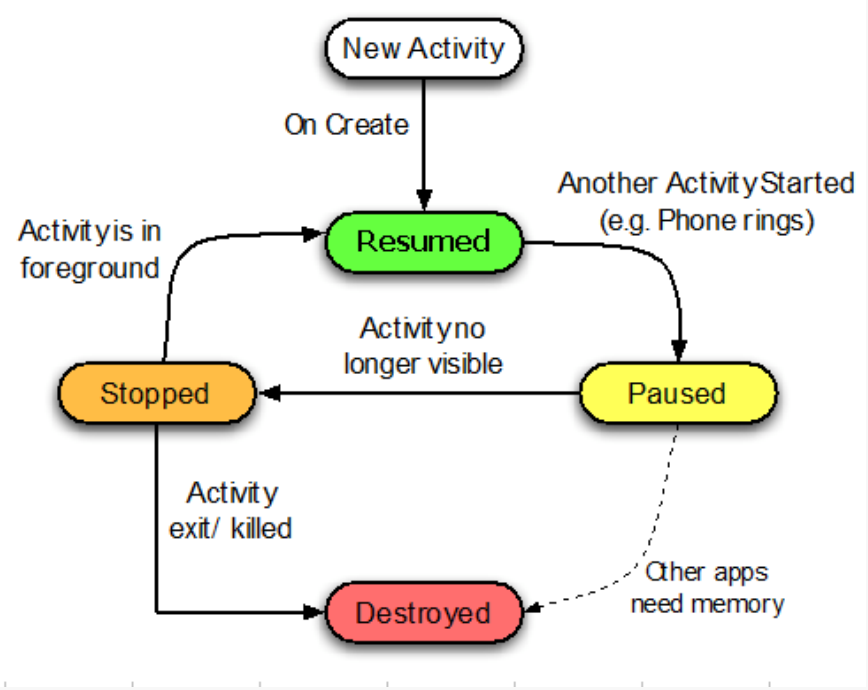
- Shut down Activity hiện tại: Sử dụng method **finish()**
- Shut down Activity khởi động trước đó: Sử dụng method **finishActivity()**
  - Đối số: Request code – Request code tương ứng với Activity mà Developer chỉ định khi khởi động Activity bằng method startActivityForResult()
- Chú ý
  - Về cơ bản sẽ không cần sử dụng các method này, do Android đã thực hiện quản lý life cycle cho Activity

## Contents

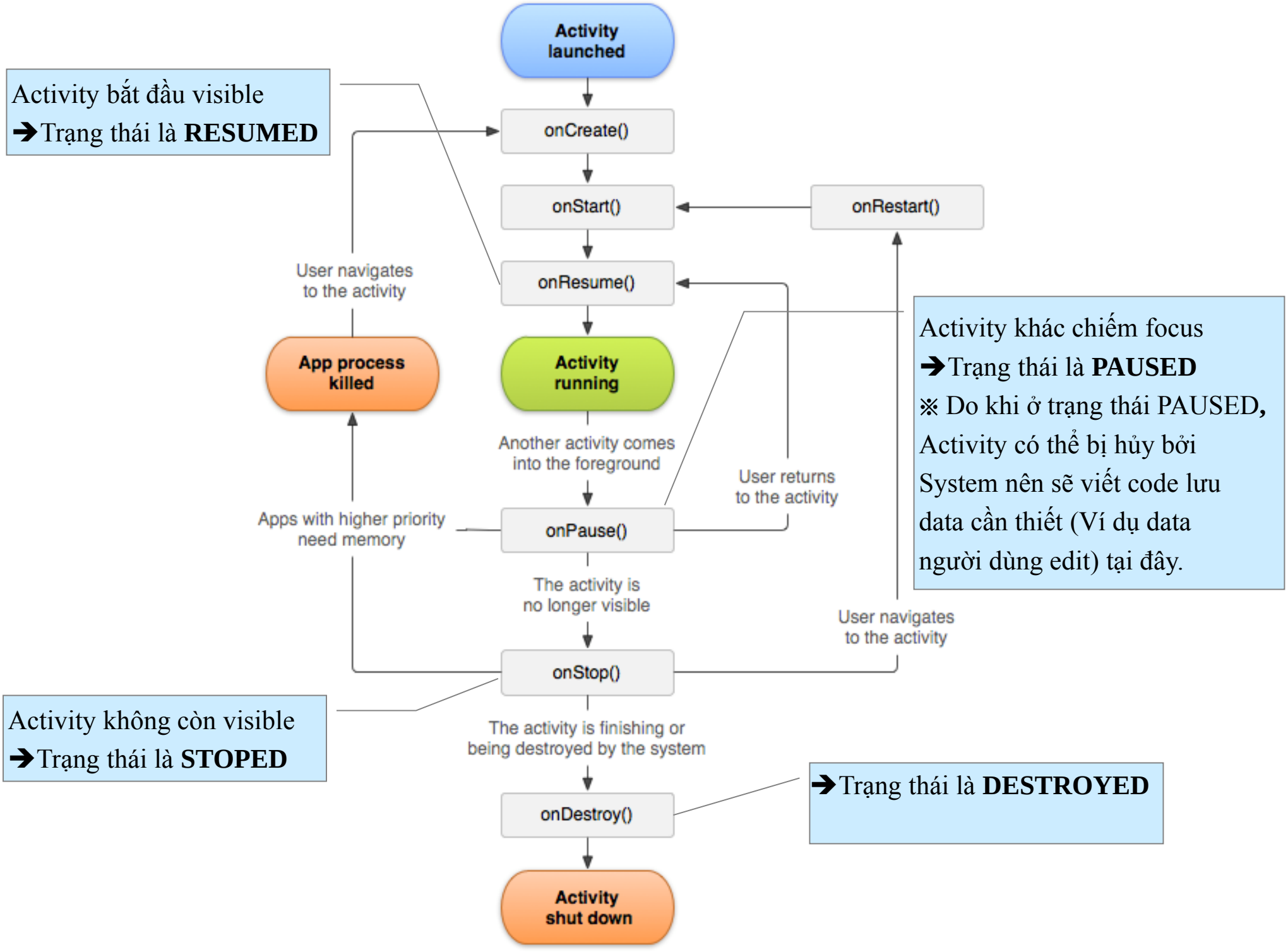

- 1. Definition
- 2. Creating an Activity
- 3. Starting and shutting down an Activity
- 4. Managing the Activity Lifecycle
- 5. Tasks and Back Stack
- 6. Practice

# Managing the Activity Lifecycle

- 3 trạng thái cơ bản của Activity
  - **Resumed (Running):** Activity ở foreground và được focus
  - **Paused:** Activity đang visible, nhưng 1 Activity khác đang hiển thị ở foreground và được focus
    - Activity object được lưu trong bộ nhớ và duy trì thông tin trạng thái
    - Vẫn được gắn với window manager
    - Có thể bị hủy bởi system nếu tình trạng bộ nhớ quá thấp
  - **Stopped:** Activity không còn visible
    - Có thể bị hủy bởi system khi cần bộ nhớ




➔**Quản lý lifecycle của Activity bằng cách implement các callback method**


Activity launched
onCreate()
onStart()
onRestart()
onResume()
Activity running
App process killed
User navigates to the activity
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down
Activity bắt đầu visible
➔Trạng thái là RESUMED
Activity khác chiếm focus
➔Trạng thái là PAUSED
※ Do khi ở trạng thái PAUSED, Activity có thể bị hủy bởi System nên sẽ viết code lưu data cần thiết (Ví dụ data người dùng edit) tại đây.
Activity không còn visible
➔Trạng thái là STOPED
➔Trạng thái là DESTROYED

# Callback methods summary

- onCreate()
  - Được gọi khi Activity được tạo
  - Xử lý thường implement: Khởi tạo UI (Tạo View, bind data, ...)
  - Đối số: Đối tượng Bundle lưu giữ trạng thái trước đó của Activity nếu có (Trường hợp Activity được capture)
- onRestart()
  - Được gọi khi 1 Activity được start lại khi Activity đó đã bị stop trước đó
- onStart()
  - Được gọi trước khi Activity trở nên visible với người dùng
- onResume()
  - Được gọi ngay trước khi Activity bắt đầu tương tác với user
- onPause()
  - Được gọi khi system tiếp tục lại 1 Activity khác
  - Xử lý thường implement: Lưu lại các thay đổi, stop animation cũng như các thao tác tiêu tốn CPU
- onStop()
  - Được gọi khi Activity không còn visible
- onDestroy()
  - Được gọi khi Activity bị hủy

## Saving activity state

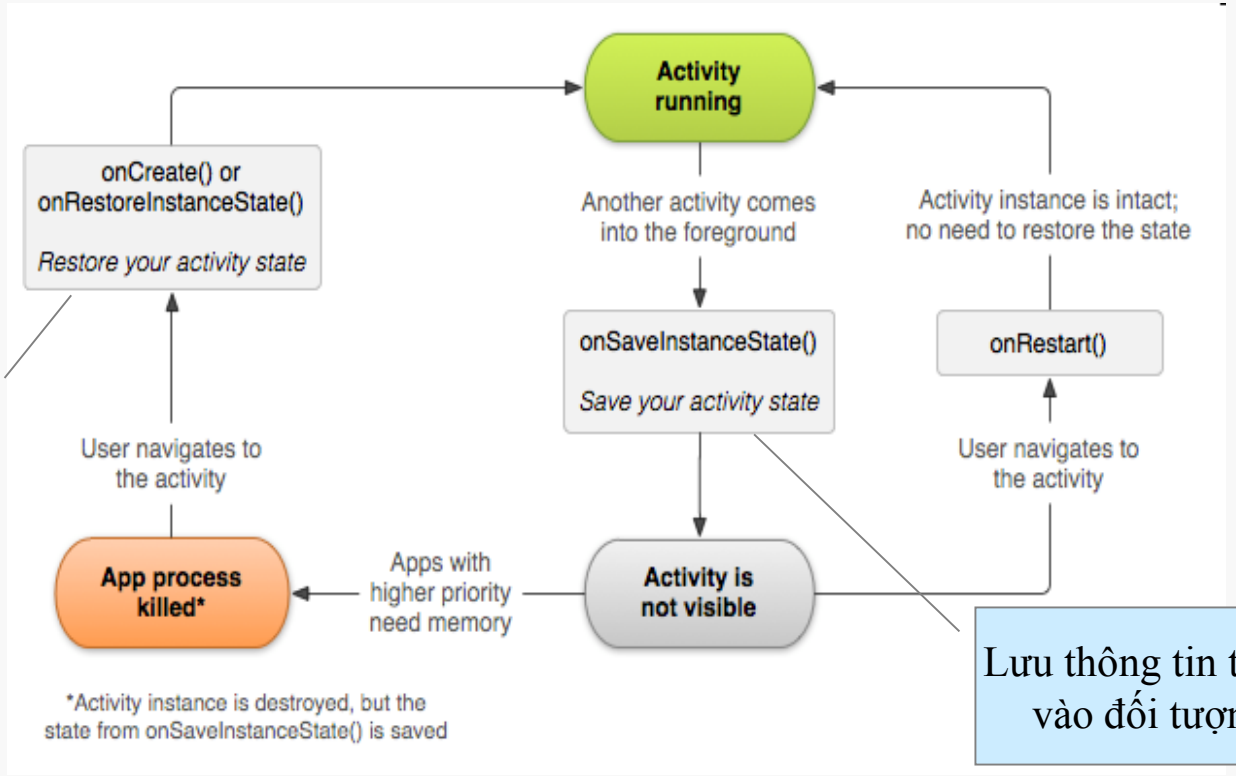
- Khi Activity bị pause hay bị stop, trạng thái (*1) của Activity vẫn được giữ lại
  - ➔ Khi Activity được focus trở lại các thay đổi trước đó sẽ được giữ nguyên
- Tuy nhiên 1 số trường hợp chẳng hạn khi system cần bộ nhớ, Activity sẽ bị hủy -> Activity sẽ ko giữ nguyên được trạng thái khi lấy lại focus.

➔Cần implement callback method để lưu thông tin trạng thái của Activity:

- onPause(): Lưu persistent state data phạm vi ứng dụng
  - Xử lý khôi phục trạng thái thực hiện ở method *onResume()*
- onSaveInstanceState(Bundle outState): Lưu trạng thái của Activity (transient state data)
  - Xử lý khôi phục trạng thái thực hiện ở method *onCreate() hoặc onRestoreInstanceState()*
  - Trường hợp khởi động lại ứng dụng sẽ mất các thông tin trạng thái

# onSaveInstanceState() callback method

- Được gọi trước khi system hủy Activity
- Đối số là Bundle instance dùng để lưu thông tin trạng thái Activity dưới dạng các cặp key-value.




Đối số là đối tượng Bundle đã thiết đặt thông tin ở method onSaveInstanceState()

Lưu thông tin trạng thái vào đối tượng Bundle

# Sample implementation

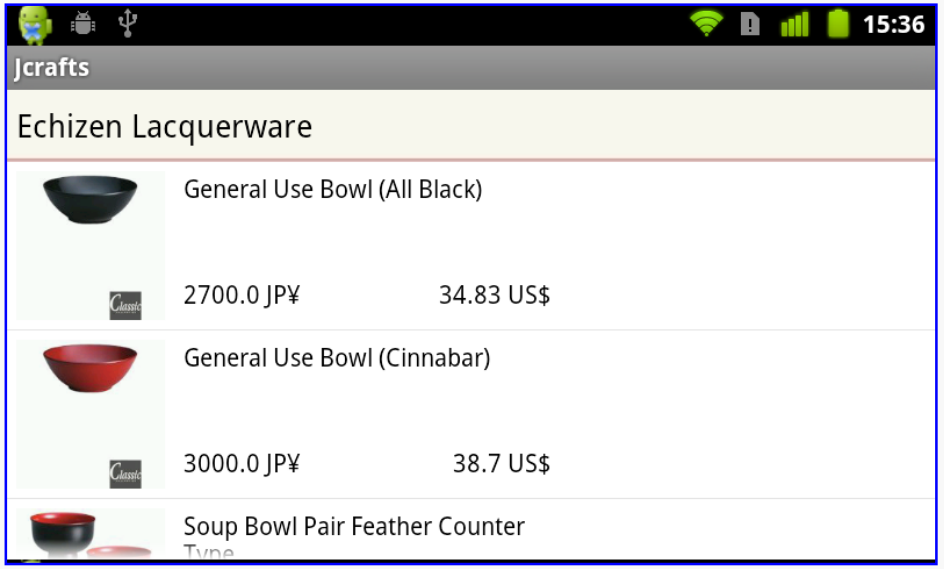
```
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
  super.onSaveInstanceState(savedInstanceState);

  // Save UI state changes to the savedInstanceState.
  // This bundle will be passed to onCreate if the process is killed and
  restarted.
  savedInstanceState.putBoolean("MyBoolean", true);
  savedInstanceState.putDouble("myDouble", 1.9);
  savedInstanceState.putInt("MyInt", 1);
  savedInstanceState.putString("MyString", "Welcome back to Android");
  // etc.
}
@Override
public void onRestoreInstanceState(Bundle savedInstanceState) {
  super.onRestoreInstanceState(savedInstanceState);

  // Restore UI state from the savedInstanceState.
  // This bundle has also been passed to onCreate.
  boolean myBoolean = savedInstanceState.getBoolean("MyBoolean");
  double myDouble = savedInstanceState.getDouble("myDouble");
  int myInt = savedInstanceState.getInt("MyInt");
  String myString = savedInstanceState.getString("MyString");
}
```

## Default implementation

- Cho dù Developer không implement onSaveInstanceState(), thì Activity vẫn sẽ được lưu trạng thái bởi implementation mặc định của framework
- Default implementation của method onSaveInstanceState() gọi ra default implementation của từng View trong Layout
  - ➔Từng View sẽ quyết định xem sẽ lưu thông tin gì
  - Hầu hết các Widget trong Android đã implement các xử lý lưu trạng thái thích hợp cho method này vì vậy mọi thay đổi ở UI sẽ tự động được lưu lại
    - **Điều kiện:** Developer phải quy định ID (thuộc tính android:id) duy nhất cho mỗi Widget
- Trường hợp override để lưu thông tin bổ sung thì cần gọi ra implementation của super class *(super.onSaveInstanceState())*

# Testing ability to restore state

- Xoay thiết bị để thay đổi hướng màn hình
  - ➔System sẽ hủy và tạo lại Activity
- Xác nhận thông tin trạng thái được giữ nguyên

## Handling configuration changes

- Các cấu hình của thiết bị có thể thay đổi khi chạy ứng dụng, (ví dụ hướng màn hình, ngôn ngữ, …)

➔Android sẽ tạo lại Activity để ứng load lại ứng dụng với các thông tin resource phù hợp với cấu hình mới (layout, string, …)

➔Khi thiết kế ứng dụng cần tính đến việc xử lý cho các thay đổi này

- Cách tốt nhất để lưu và khôi phục trạng thái là sử dụng các callback method:
  - onSaveInstanceState()
  - onRestoreInstanceState() (hoặc onCreate())

## Coordinating activities

- Khi 1 Activity gọi ra Activity khác, cả 2 Activity đều thay đổi về trạng thái (Resumed, Paused, Stopped)
- Trong trường hợp cần chia sẻ dữ liệu giữa các Activity thì cần phải hiểu  đúng đắn về timing thay đổi trạng thái của Activity để implement xử lý thích hợp
- Thứ tự gọi các callback method khi Activity A gọi ra Activity B
  1. Method onPause() của Activity A được execute
  2. Các method onCreate(), onStart(), và onResume() của Activity B được execute ➔ Activity B được focus
  3. Nếu Activity A không visible nữa, method onStop() được execute

➔ Trường hợp muốn ghi thông tin vào Database khi Activity A stop và đọc thông tin từ Database khi Activity B start thì nên implement xử lý ghi thông tin vào Database ở method onPause() của Activity A.

## Contents


- 1. Definition
- 2. Creating an Activity
- 3. Starting and shutting down an Activity
- 4. Managing the Activity Lifecycle
- 5. Tasks and Back Stack
  - 5.1. Overview
  - 5.2. Defining launch modes
  - 5.3. Handling affinities
  - 5.4. Clearing the back stack
  - 5.5. Starting a task
- 6. Practice

## Overview

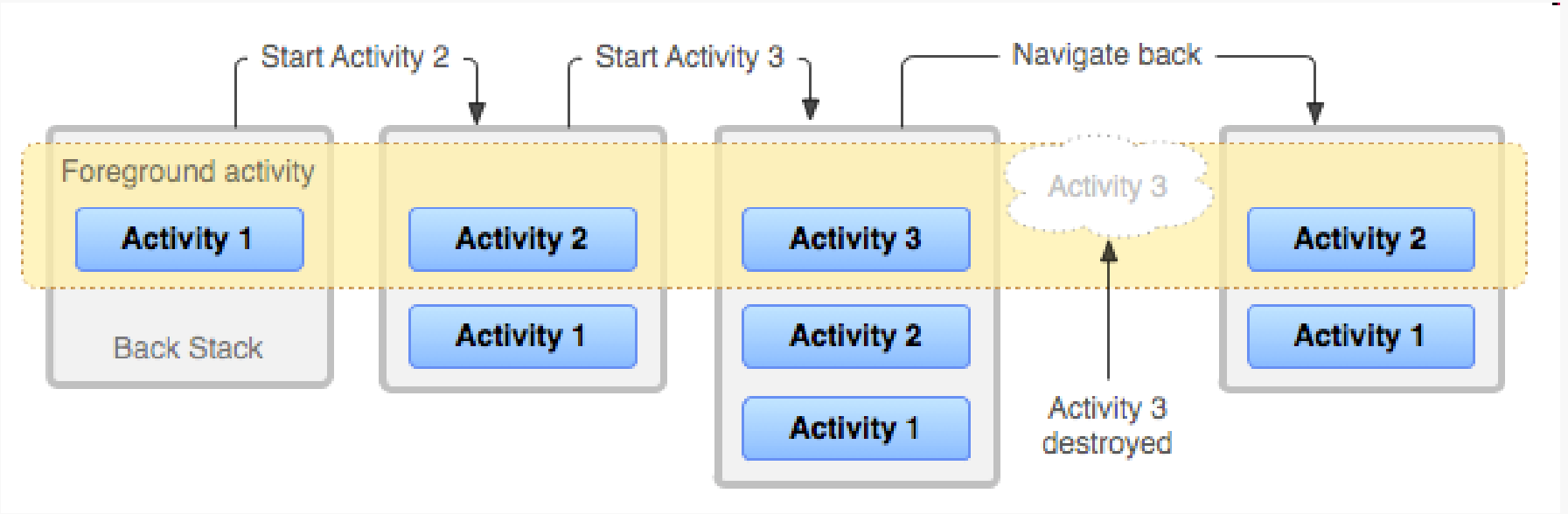
- Một ứng dụng thông thường chứa nhiều Activity
- Một Activity được thiết kế để người dùng có thể thực hiện 1 action nào đó. Ví dụ ứng dụng Shopping như Amazon, Ebay có các Activity sau:
  - Activity hiển thị category hàng hóa
  - Activity hiển thị danh sách hàng hóa
  - Activity tìm kiếm hàng hóa
- Một Activity có thể gọi ra Activity của ứng dụng khác. Ví dụ việc gọi ra Activity gửi mail như sau:
  - Start Activity thông qua đối tượng Intent có action là SEND, kèm theo data như địa chỉ email, nội dung mail, …
  - Activity Compose của ứng dụng gửi mail sẽ được khởi động
    - Activity (của ứng dụng khác) khai báo xử lý Intent loại này sẽ được gọi ra
    - Trường hợp nhiều Activity hỗ trợ Intent thì system sẽ đưa ra danh sách để người dùng lựa chọn

## Overview (cont.)

- Một task là 1 tập hợp các Activity mà người dùng tương tác để thực hiện 1 công việc nào đó.
  - Các Activity có thể thuộc cùng 1 ứng dụng hay nhiều ứng dụng khác nhau
  - Các Activity này được sắp xếp trong 1 stack (Back stack) theo thư tự Activity đó được open
- Hầu hết các task được khởi động từ màn hình Home khi người dùng click vào icon chạy ứng dụng
  - Trường hợp chưa tồn tại task cho ứng dụng (Ứng dụng ko được sử dụng gần đây) ➔Task mới sẽ được tạo và “main” Activity của ứng dụng sẽ được open và đóng vai trò như root Activity

# Activity stack

- Khi 1 Activity mới được start
  - Activity trước đó sẽ bị stop và được hệ thống duy trì trong back stack
  - Activity mới nhận được focus và được đưa vào back stack
- Khi người dùng kết thúc thao tác với Activity hiện tại và nhấn Back button
  - Activity hiện tại bị pop ra khỏi stack và bị destroy
  - Activity trước đó được khôi phục



## Activity stack (cont.)

- Khi 1 Activity mới được start, Activity cũ sẽ được thông báo về việc thay đổi trạng thái thông qua các callback method
  - Implement callback method nhằm mục đích cung cấp xử lý thích hợp khi thay đổi trạng thái. Ví dụ:
    - Release kết nối mạng và kết nối database khi trạng thái là stop
    - Tiến hành kết nối lại khi trạng thái là resume

## Managing Tasks

- Android sử dụng cơ chế stack ("Last in, first out") để quản lý các Activity của Task
- Hầu hết trường hợp Developer ko cần quan tâm Activity liên kết với Task hay bị đưa ra khỏi back stack thế nào
- Tuy nhiên Android cho phép Developer thay đổi cơ chế stack thông thường, ví dụ:
  - Thiết đặt để khi start 1 Activity thì 1 task mới được start thay vì đặt Activity vào task hiện tại
  - Thiết đặt để khi start 1 Activity thì instance trước đó của Activity (nếu có) sẽ được focus thay vì tạo mới 1 Instance cho Activity và push vào đầu của back stack
  - Thiết đặt để remove toàn bộ Activity, ngoại trừ root Activity khi người dùng rời khỏi Task (Nhấn nút “Home” của device)

# Managing Tasks (cont.)

- Thay đổi cơ chế stack thông thường bằng cách
  - Thiết đặt Flag cho đối tượng Intent khi khởi động Activity bằng method startActivity()
  - Thiết đặt thuộc tính cho phần tử <activity> trong manifest file (AndroidManifest.xml)
- Các intent flag chính
  - FLAG_ACTIVITY_NEW_TASK
    - Tạo ra 1 task mới
    - Activity được khởi động là root Activity trong back stack của task mới
    - Trường hợp đã tồn tại Task đang chạy Activity rồi thì Activity mới ko được tạo
  - FLAG_ACTIVITY_CLEAR_TOP
    - Trường hợp Activity được khởi động đã tồn tại đang chạy trong task hiện tại rồi thì sẽ pop toàn bộ Activity ở trên Activity đó khỏi back stack
    - Ví dụ: Trường hợp back stack đang chứa các Activity A, B, C, D. Nếu D gọi ra B thì C và D sẽ kết thúc. Back stack còn lại A, B
  - FLAG_ACTIVITY_SINGLE_TOP
    - Trường hợp Activity được khởi động đang ở top của back stack thì ko tạo mới instance của Activity

# Managing Tasks (cont.)

- Các thuộc tính chính của phần tử <activity> trong manifest file (AndroidManifest.xml)
  - android:taskAffinity
    - Khai báo tên mối quan hệ (Affinity)
    - Các Activity có cùng Affinity sẽ thuộc về cùng 1 Task
    - Trường hợp thuộc tính android:taskAffinity không được thiết đặt thì Affinity của Activity sẽ kế thừa giá trị đã thiết đặt cho Application
    - Mặc định mọi Activity trong ứng dụng có cùng Affinity
    - Giá trị Affinity mặc định thiết đặt cho Application là package name khai báo trong file manifest
  - android:launchMode
    - Chỉ dẫn về việc Activity sẽ được start như thế nào
    - Có 4 mode:
      - "**standard**“, "**singleTop**"
        - Có thể khởi tạo nhiều instance. Mỗi instance có thể thuộc về bất cứ task nào
        - Mode "**singleTop**“: Trường hợp tồn tại Activity ở top của back stack thì ko tạo mới instance
      - "**singleTask**“, "**singleInstance**“
        - Tạo mới task, activity nằm ở root của task
        - Mode "**singleTask**“: Trường hợp 1 instance của Activity đã tồn tại thì ko tạo mới.
        - Mode "**singleInstance**“: Ko cho phép Activity khác ở cùng Task -> Trường hợp “singleInstance” gọi ra 1 Activity khác thì Activity khác sẽ được phân cho 1 Task khác.

# Managing Tasks (cont.)

- android:allowTaskReparenting
  - Thiết đặt liệu Activity có thể chuyển từ task khởi động nó tới task có cùng affinity khi task có cùng affinity được focus hay không.
  - Giá trị mặc định là false
  - Ví dụ: Trường hợp click vào link trong email message. Activity hiển thị trang Web được khởi động. Activity hiển thị trang web được định nghĩa bởi ứng dụng browser, nhưng được khởi động như 1 phần của e-mail task. Nếu Activity này được reparent (thiết đặt lại cha) tới browser task, nó sẽ được hiển thị khi browser focus và sẽ mất đi khi email task focus.
  - Vì các Activity có launchMode là "singleTask“, "singleInstance“, luôn ở root của back stack (ko có Activity cha) ➔ ko áp dụng được thuộc tính này.
- android:clearTaskOnLaunch
  - Flag thiết đặt việc có remove toàn bộ Activity của Task trừ root Activity khi Activity được khởi động lại từ home screen hay không
  - Ví dụ start Activity P từ home screen. Từ P gọi ra activity Q. Sau đó bấm Home và start lại P. Bình thường sẽ phải nhìn thấy Q. Tuy nhiên, trường hợp thiết đặt flag là true thì toàn bộ Activity trên P sẽ bị remove khỏi back stack nên chỉ nhìn thấy P.
  - Trường hợp thuộc tính clearTaskOnLaunch và allowTaskReparenting đều được thiết đặt là "true" thì các Activity có thể re-parent sẽ được chuyển tới task có cùng Affinity

# Managing Tasks (cont.)

- android:alwaysRetainTaskState
  - Thiết đặt liệu state của task chứa Activity luôn được duy trì bởi system hay không
  - Trường hợp thiết đặt là false, system sẽ cho phép reset task về state khởi tạo
  - Thuộc tính này chỉ có ý nghĩa đối với root activity của task
  - Thông thường system sẽ xóa 1 task (xóa toàn bộ Activity nằm trên root Activity tại back stack) trong 1 số tình huống khi use lựa chọn lại task từ màn hình home
    - Ví dụ: trường hợp người dùng ko sử dụng task trong khoảng 30‘
  - Trường hợp thuộc tính "android:alwaysRetainTaskState" được thiết đặt là true, thì task ko bị xóa
    - Ví dụ trong ứng dụng web browser, người dùng muốn giữ lại trạng thái các tab ở lần truy cập tiếp theo
- android:finishOnTaskLaunch
  - Thiết đặt xem có shutdown 1 instance đang tồn tại của Activity hay không khi người dùng chạy lại task của nó (lựa chọn task ở màn hình home)
  - Trường hợp thuộc tính này và thuộc tính "android:allowTaskReparenting" đều được thiết đặt là true thì thuộc tính này được ưu tiên hơn, tức là Activity bị shutdown (destroy) chứ không re-parent đến task khác

## Contents


- 1. Definition
- 2. Creating an Activity
- 3. Starting and shutting down an Activity
- 4. Managing the Activity Lifecycle
- 5. Tasks and Back Stack
- 6. Practice

# THANK YOU!