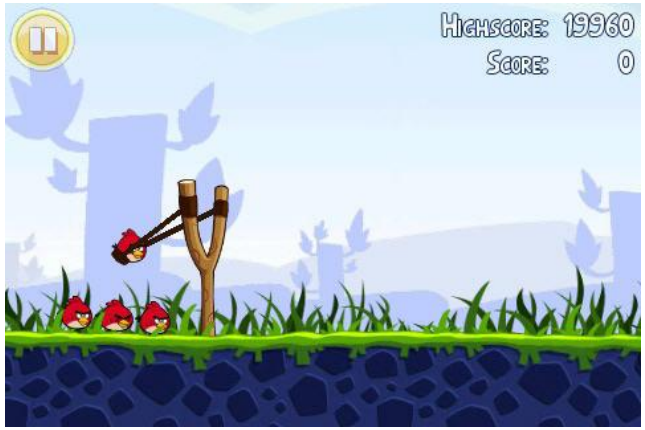
# Application Components

Dependable and Secure Systems

# Activity

- An Activity is a single screen with a user interface of an application.
- Generally an application is composed by several activities. Each activity represent a single application screen.

# Creating an activity

- First Step?
- The activity is an application component, so it has to be declared in the **Manifest** file.
- Look at the code of the sample application…

# Creating an Activity (2)

- Graphic Layout.
- Layout of activities are described statically through XML files stored in **res** folder.
- Text box, buttons, loading bars and so on can be added and customized.
- GUI. Simple with drag and drop functionality.
  - Hint: Insert the items in the activity screen using the GUI, then customize them statically from XML code.

# Creating an Activity (3)

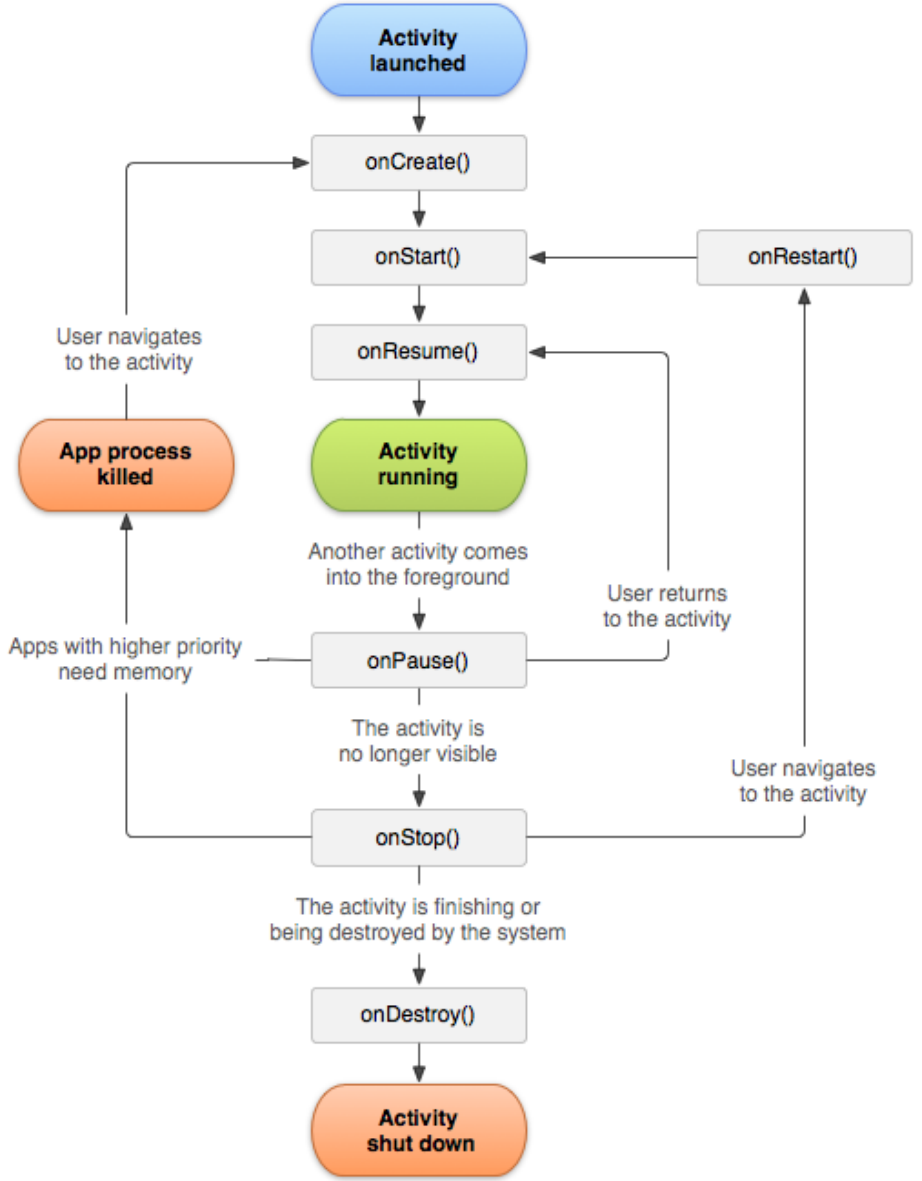
- Java class.
- Each activity should be a java class in your project.

  *public class myActivity* ***extends*** *Activity {*

  …

  *}*
- There is no main in an activity class. Everything starts from the method **OnCreate().**

# Activity Lifecycle



# Activity Status

- Running Foreground.
  - The activity is visible and in the foreground.
- Running Background (Paused).
  - The activity is visible but another app has the focus.
- Stopped.
  - The activity is not visible.
- Destroyed.
  - The activity has been killed, all variables have been destroyed.

# Zygote

- User does not decide when to close an application.
- The user starts an Android application (and thus one or more activities), then the application runs until the process manager **Zygote** does not assert that it is not useful anymore.
- What about variables stored in memory?

# Variables

- If Zygote destroy an application, all variables will be deleted!

# Bundle

- The Bundle is a data container. Is a structure where it is possible to push values of variables.
- Bundles are stored in memory and survive when an application is destroyed.
- Save data to bundle in *OnSaveInstanceState()* method, restore data from bundle in *OnCreate()*.

# Warning!!!

- Do **not** save data in *onStop()* or *onDestroy()* methods. These methods could be not invoked in cases of extremely low memory.
- Use them only for clean-ups.

# Starting an Activity (1)

- When an application is started the main activity is launched.
- The main activity is declared in the manifest file through an **intent filter** for the application launcher.
- If the main activity is not declared the application will not start.

# Starting an Activity (2)

```
public void callActivity(){
Intent intent = new
Intent(this,CalledActivity.class);
startActivity(intent);
}
```

# Intent

- **Intents** are used to send messages and data between applications or application components.
- Every Intent has a *sender* and a *receiver*.
- There are two types of Intents:
  - Explicit Intents: the sender specifies the intent receiver.
  - Implicit Intents: the sender specifies a class of possible receivers.

# Intent Components

- **Action**: The action that should be performed by the receiver. Used to assess who should be the receiver. Example: ACTION_CALL, ACTION_MAIN
- **Data**: information sent from the sender to the receiver.
- **Category**: Additional info on the type of operation requestested.

# Explicit Intent

- **Intent(this,CalledActivity.class);**

- The receiver is explicitly specified. The Intent has to be delivered to the CalledActivity class.
- The action can be considered implicit, since it is specified by the method: **startActivity(intent);**

# Implicit Intents

```
Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.com"));
startActivity(i);
```

- For implicit intents the receiver is not specified. With this intent the sender asks that the web page at www.google.com is displayed.
- The OS looks for all the applications or components that are able to satisfy such a request.
- Applications specifies that they are able to satisfy a request declaring an Intent Filter.

# Intent Filters

- Intent Filters are declared in manifest file.

```
<activity android:name=".BrowserActivitiy"
android:label="@string/app_name">
<intent-filter>
 <action android:name="android.intent.action.VIEW" />
<category android:name="android.intent.category.DEFAULT" />
<data android:scheme="http"/>
 </intent-filter>
</activity>
```

# Exercise

- An easy **Money converter** with two activities.
- The first activity allows conversion.
- The second activity allows the definition of the conversion rate.

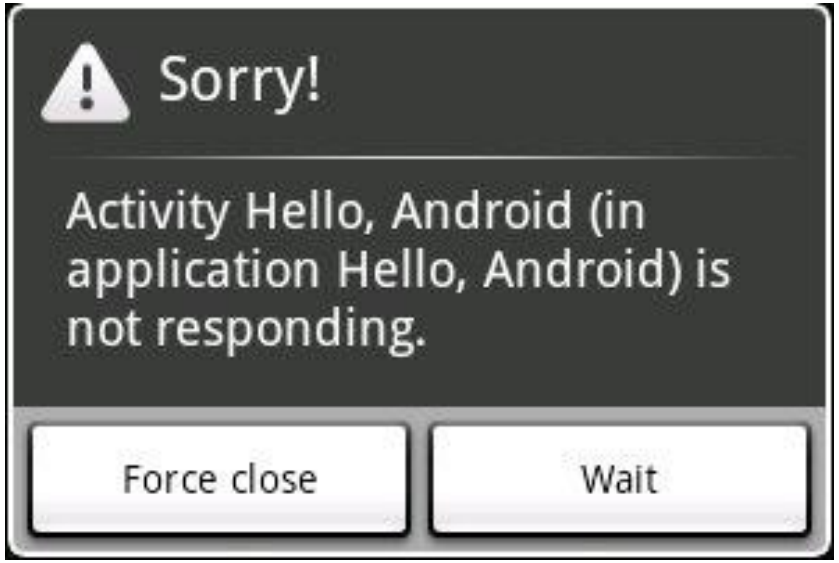
# Service

- A service is an application component that perform long and heavy background operations.
- Services do not provide a user interface.
- A service is called by an activity or another service and generally run for a long period of time.

# Example of Services

- GPS location service.
- Timers.
- Watchdogs
- Streaming managers.
- Loggers.
- …

# Why do we need Services?



# Long Operations in Activities

- Each time an **activity** takes more than 30 seconds to perform an operation an alert is raised.
- This is considered a programming error.

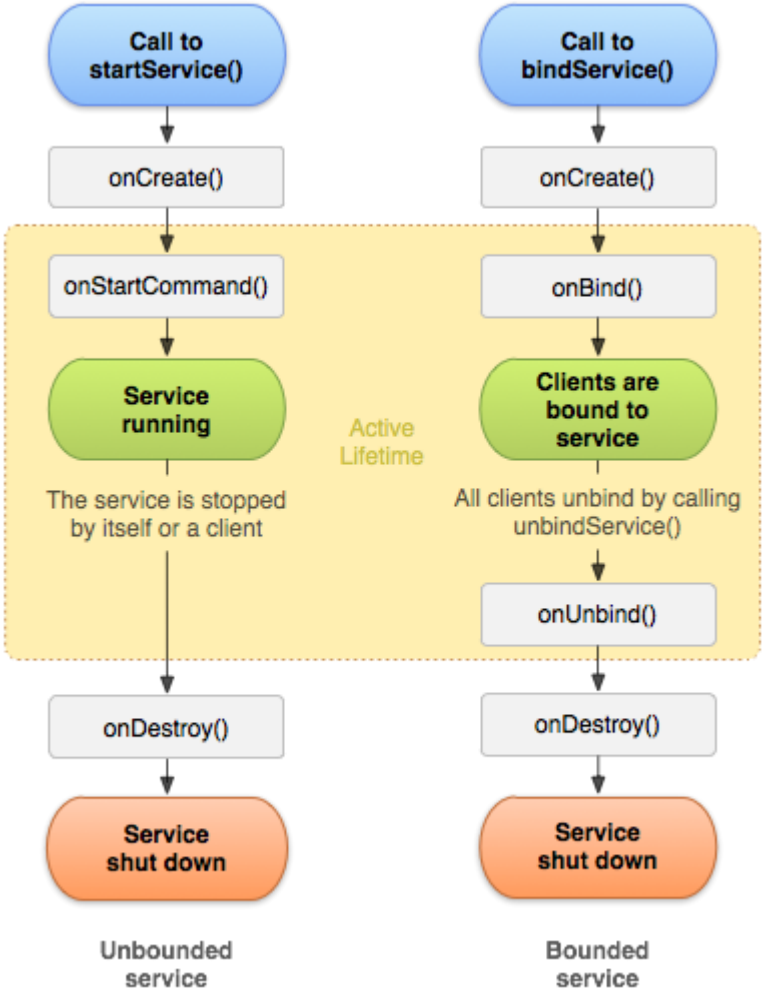
# Long Operations in Activities

- When an Activity has to perform a long operation should start a service and go on with the execution.
- The service executes its task in background and eventually return results to the calling activity.
- An activity start a service similarly to other activities: using Intents.

# Starting a Service

- There are two methods to start a service from an activity
  - *startService(Intent)*: Non-bounded call. The new service starts and live indipendently from the calling application.
  - BindService(Intent): Bounded call. The service is linked to the calling application and there is an interface of communication between the service and the application. The service dies when the application is called.

# Service Lifecycle



# Terminating a Process

- Differently from activities, services can be terminated programmatically using the *stopService()* or *stopSelf()* methods.
- If not terminated explicitely, the service will be stopped and destroyed by Zygote when the system needs memory.
- A service can be closed when it is still necessary.

# Sticky Start

- The method *startService(intent, flags)* can be used to specify what happens when a service is destroyed by Zygote.

1. *START_NOT_STICKY:* After destruction the service is not re-created.
2. *START_STICKY:* The service starts again after destruction as soon as the system has enough memory.

# Content Providers

- Content providers are data structures that allow to save and access data as in a relational database.
- Data are stored in tables.
- Tables are accessed by SQL like queries.