Mobility Programming

Lecture 3: Android Activities

DR. RAMEZ ALKHATIB

The Activity Lifecycle

- Activities are managed by the Android runtime
- Activities have a "lifecycle" consisting of states
  - From creation till death
- Standard (lifecycle) methods on the activity are invoked at each state change (can test by rotating device)

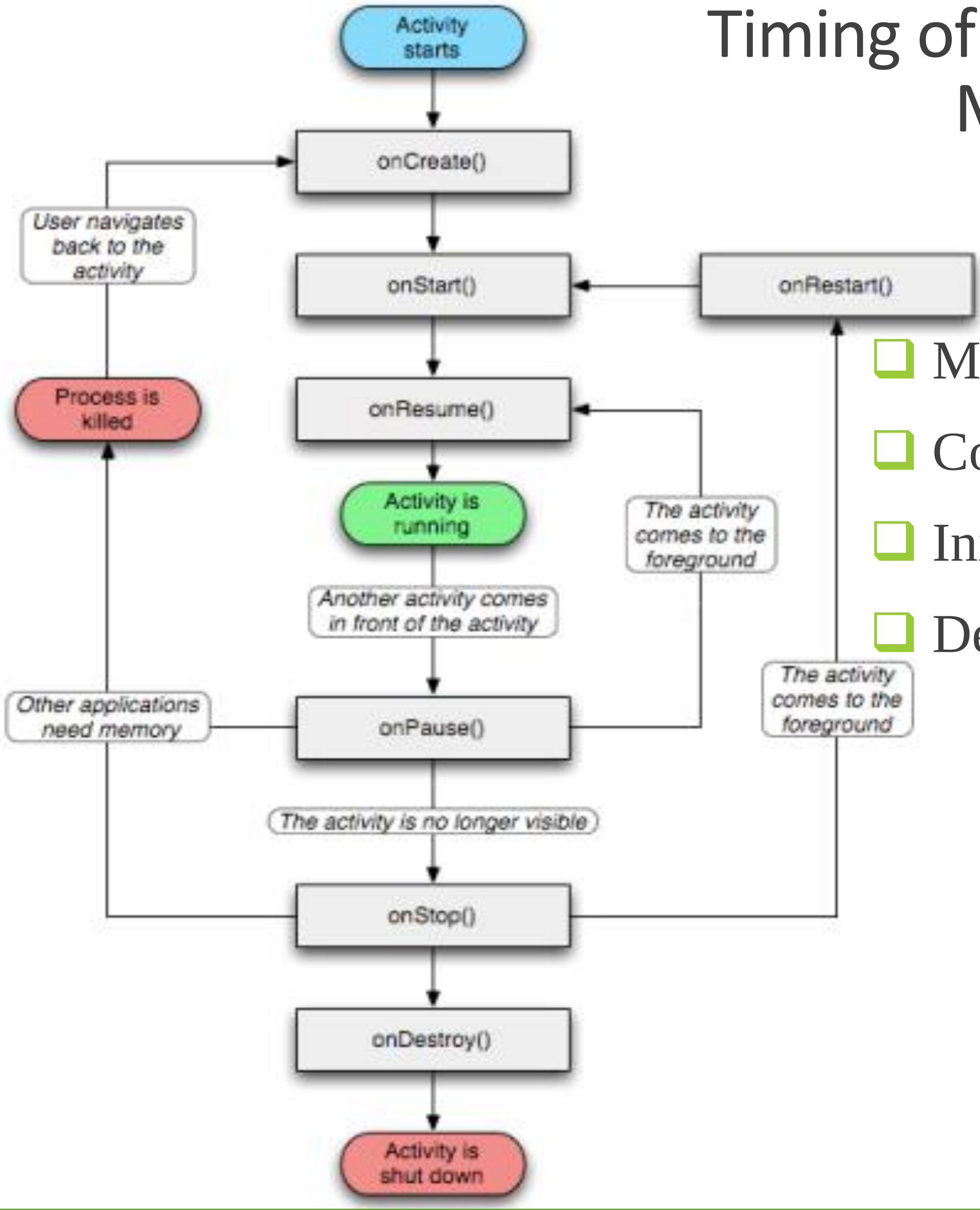
Activity States

- Created: Born to run
- Active: Working 9 to 5
- Paused: I’m about to break
- Resumed: Back to work
- Stopped: Obscured by clouds, vulnerable

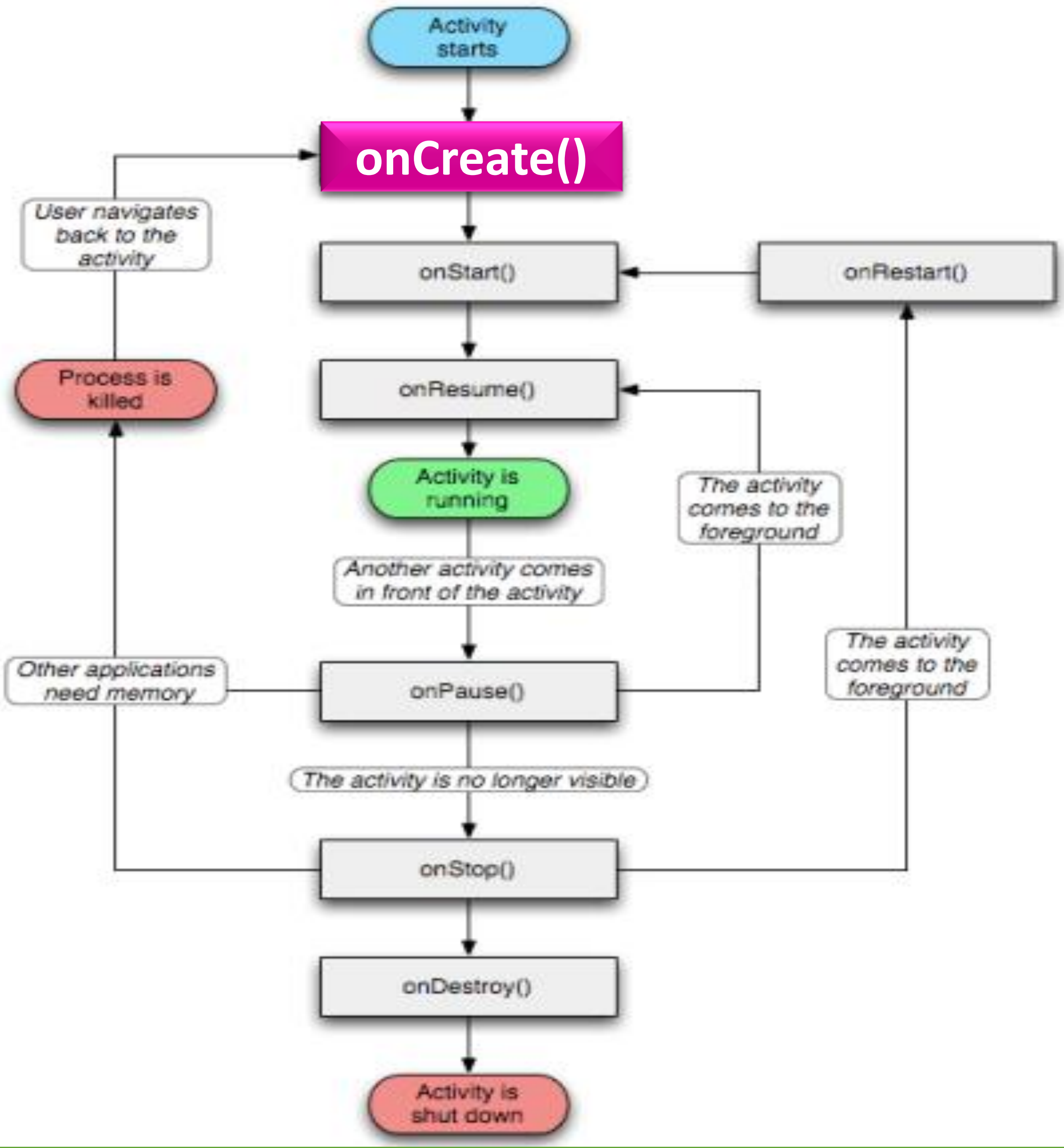
Activity Transitions

- Created $\Longrightarrow$ Active
- Active $\Longleftrightarrow$ Paused
- Paused $\Longrightarrow$ Stopped $\Longrightarrow$ Active
- Stopped $\Longrightarrow$ Killed

Timing of Activity Lifecycle Methods

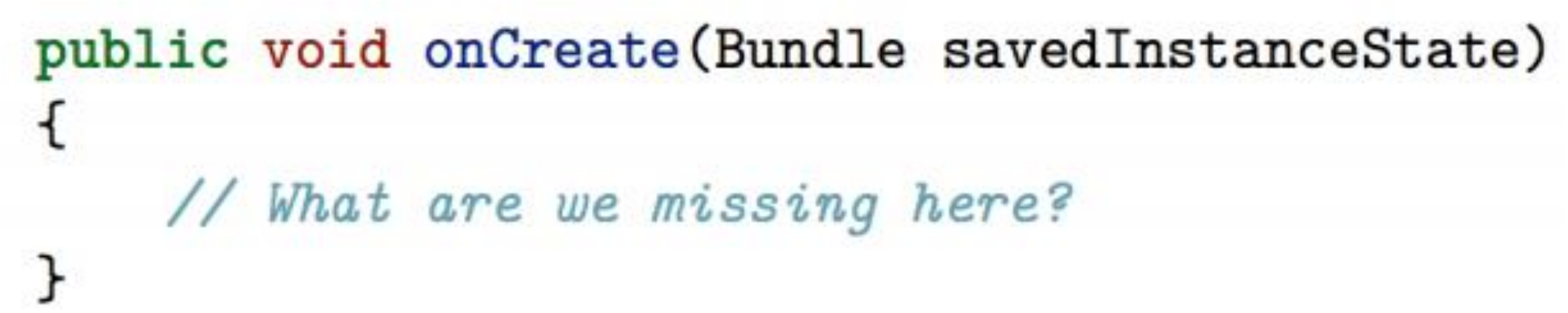

Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
The activity comes to the foreground
Other applications need memory
onPause()
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

- Most important component type
- Controls the application flow
- Initiates intents
- Delegates to other activities


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
The activity comes to the foreground
Other applications need memory
onPause()
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

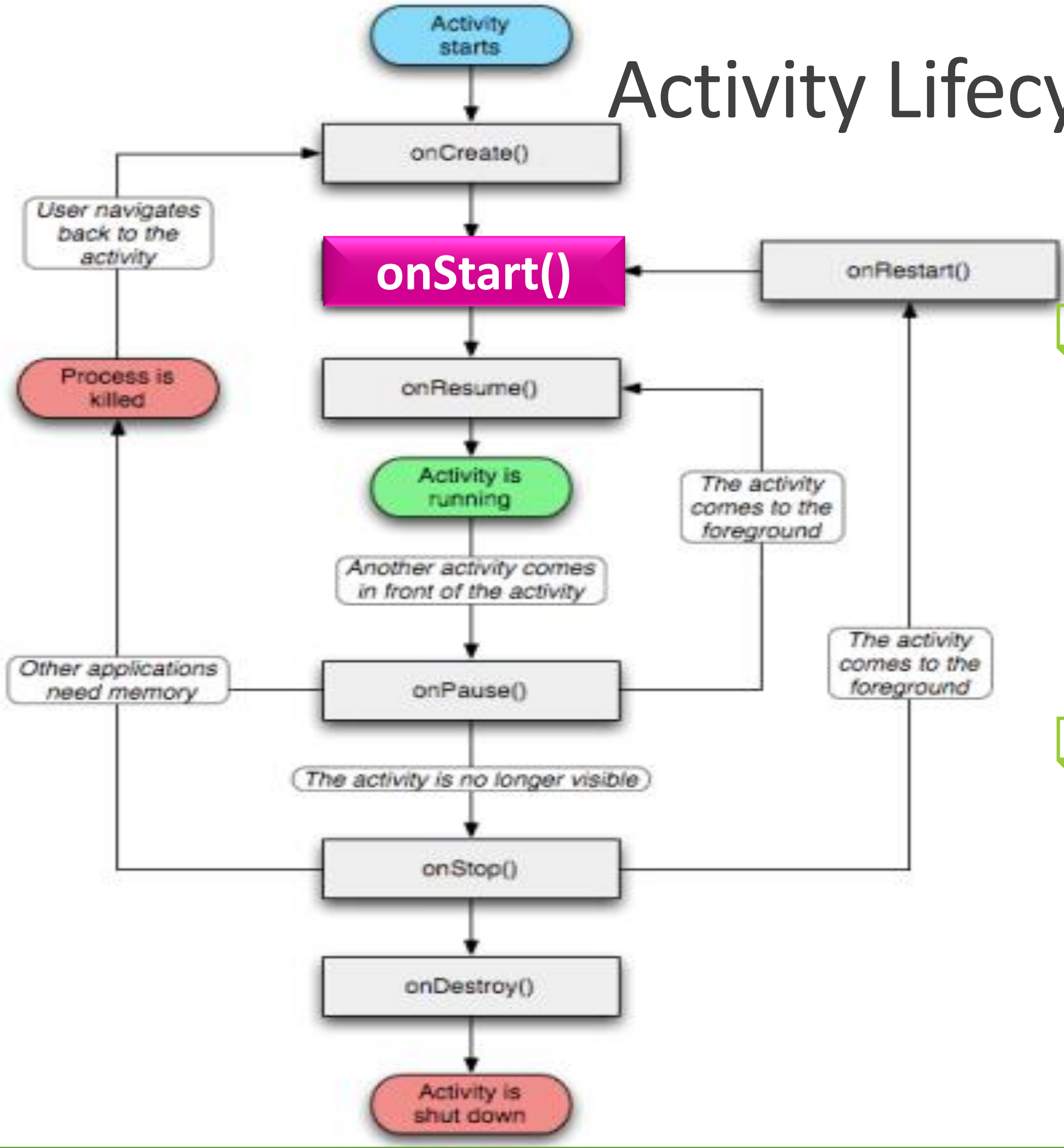
Activity Lifecycle : OnCreate()

- Activity on the foreground of the screen
- First thing called
- Called when screen is rotated
- Called when there is a language change



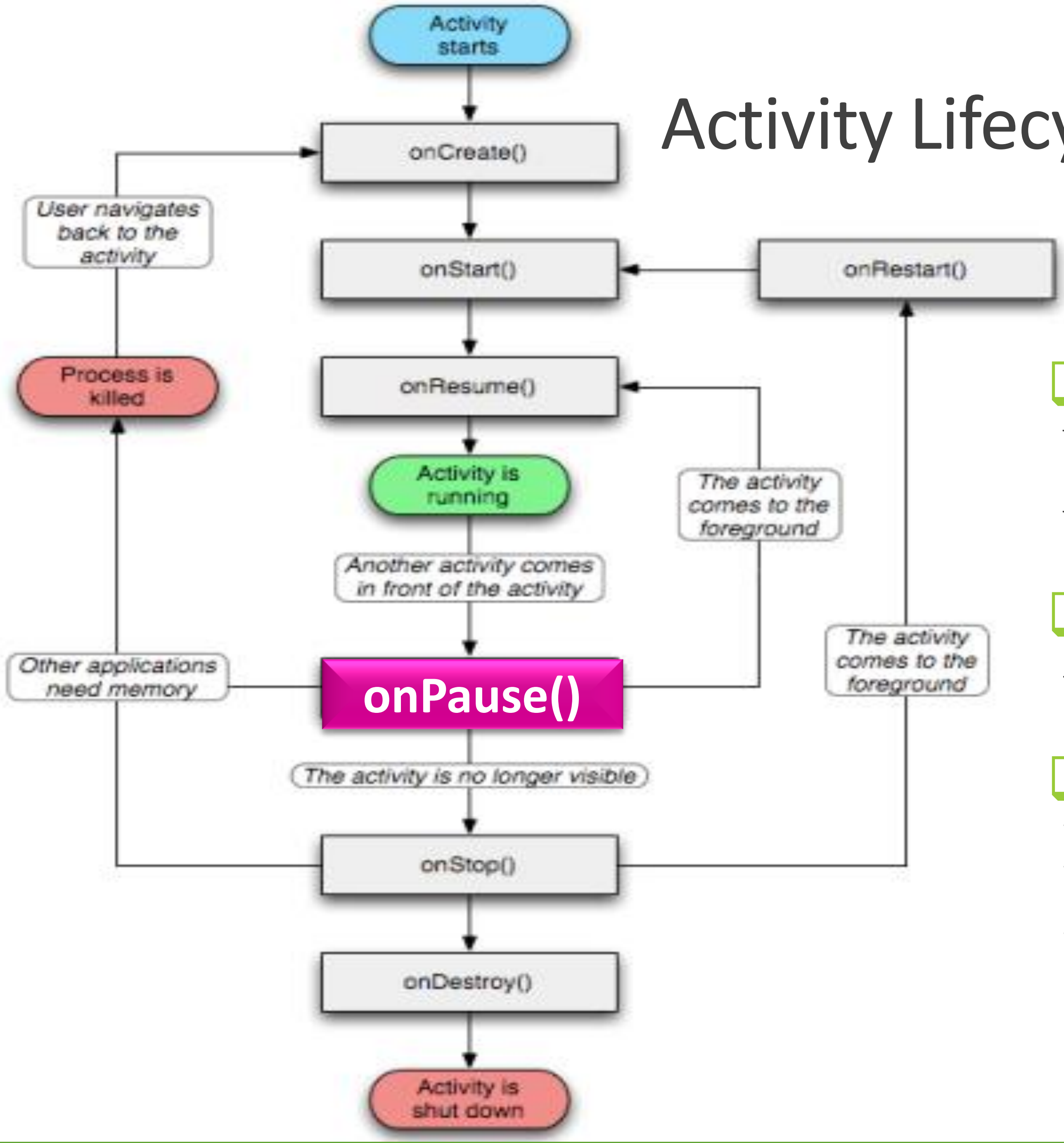
```
public void onCreate(Bundle savedInstanceState)
{
     // What are we missing here?
}
```

Activity Lifecycle : OnStart()


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
The activity comes to the foreground
Other applications need memory
onPause()
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

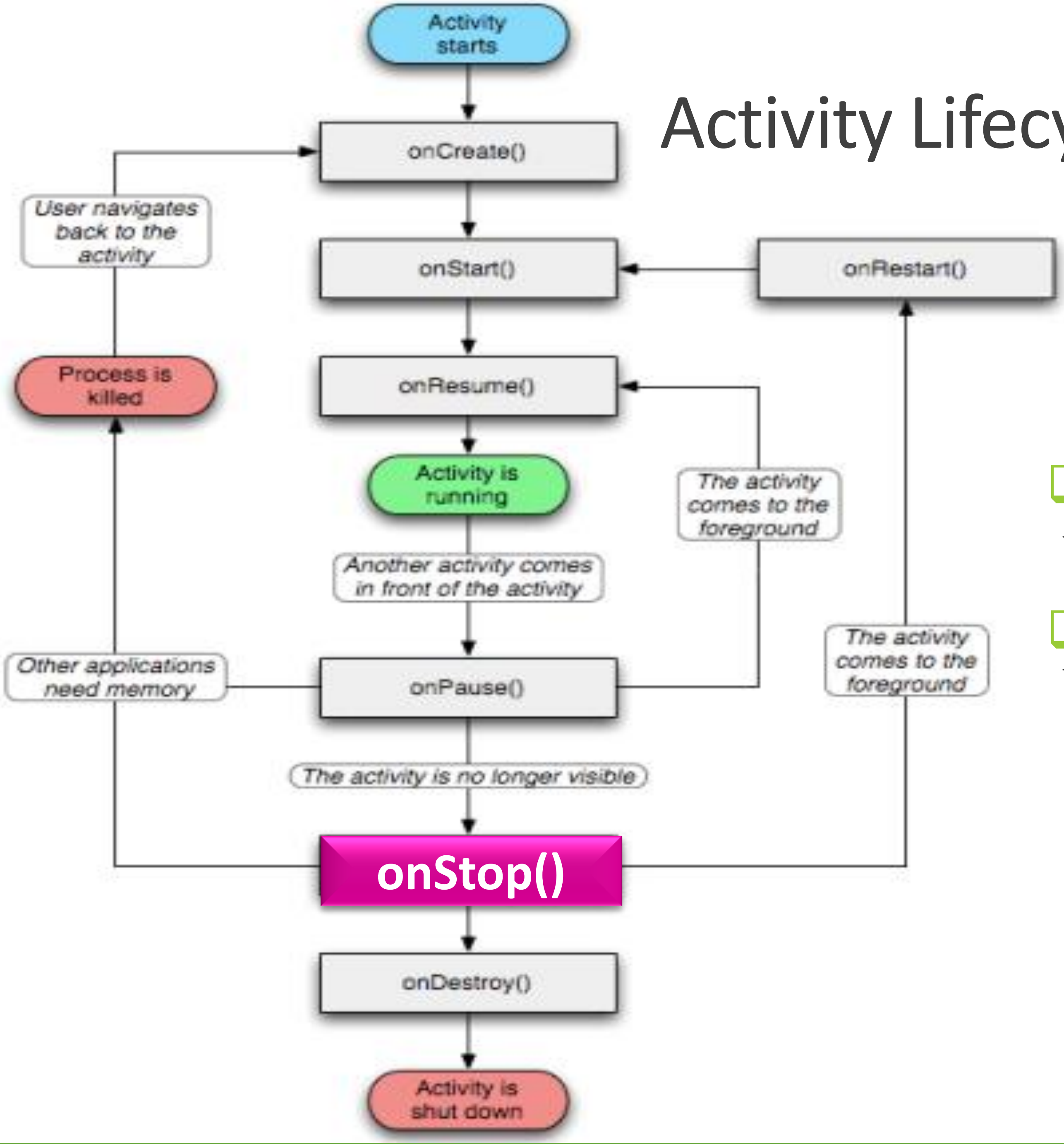
- Called after onCreate() and when user brings activity to the foreground
- When activity is brought to the foreground

Activity Lifecycle : OnPause()


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
The activity comes to the foreground
Other applications need memory
onPause()
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

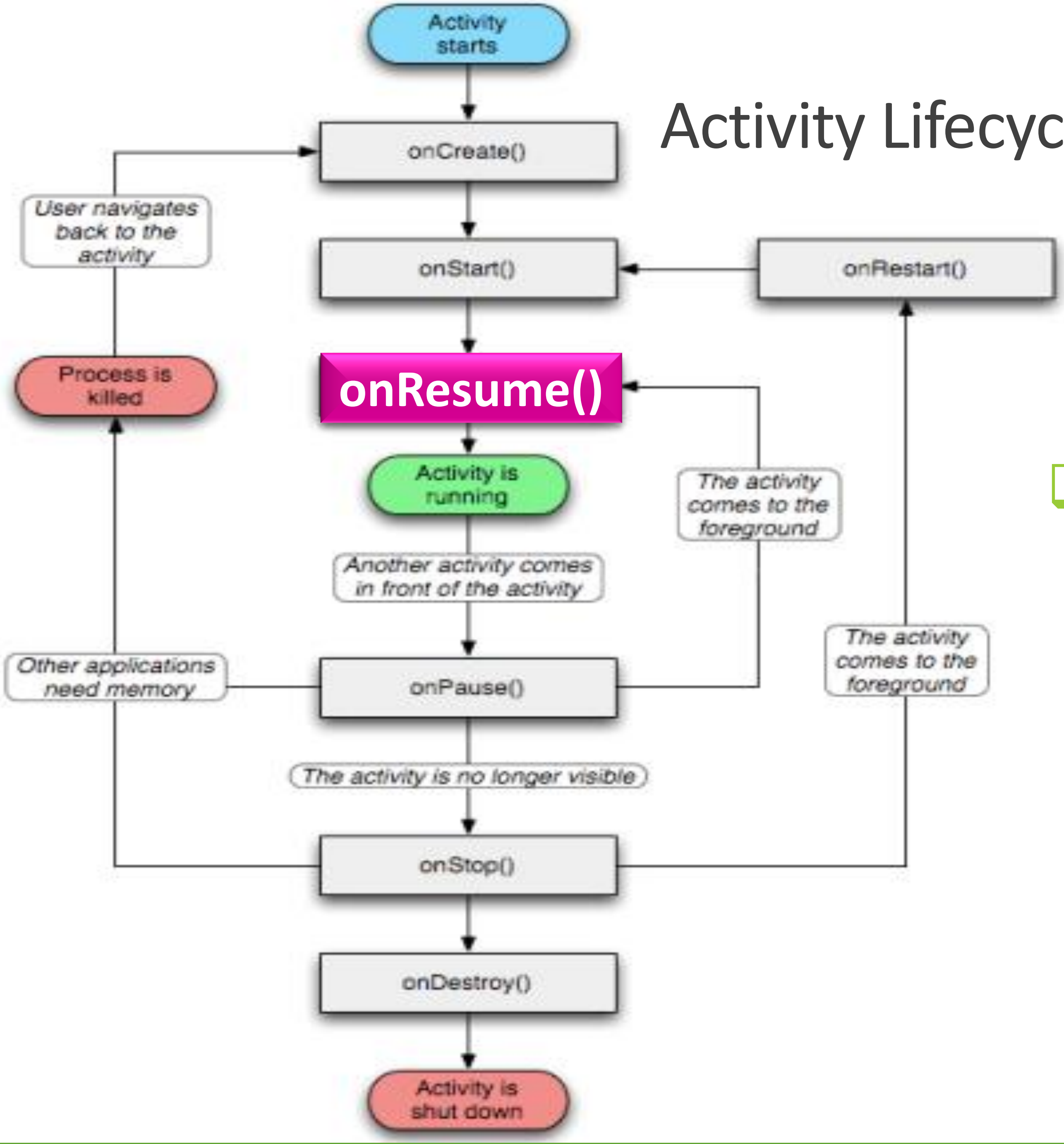
- Called when user brings another window up
- Application has to be visible
- State might be lost, if device low in memory

Activity Lifecycle : OnStop()


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
The activity comes to the foreground
Other applications need memory
onPause()
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

- ❑ Activity no longer visible
- ❑ All state lost, must be persisted somewhere

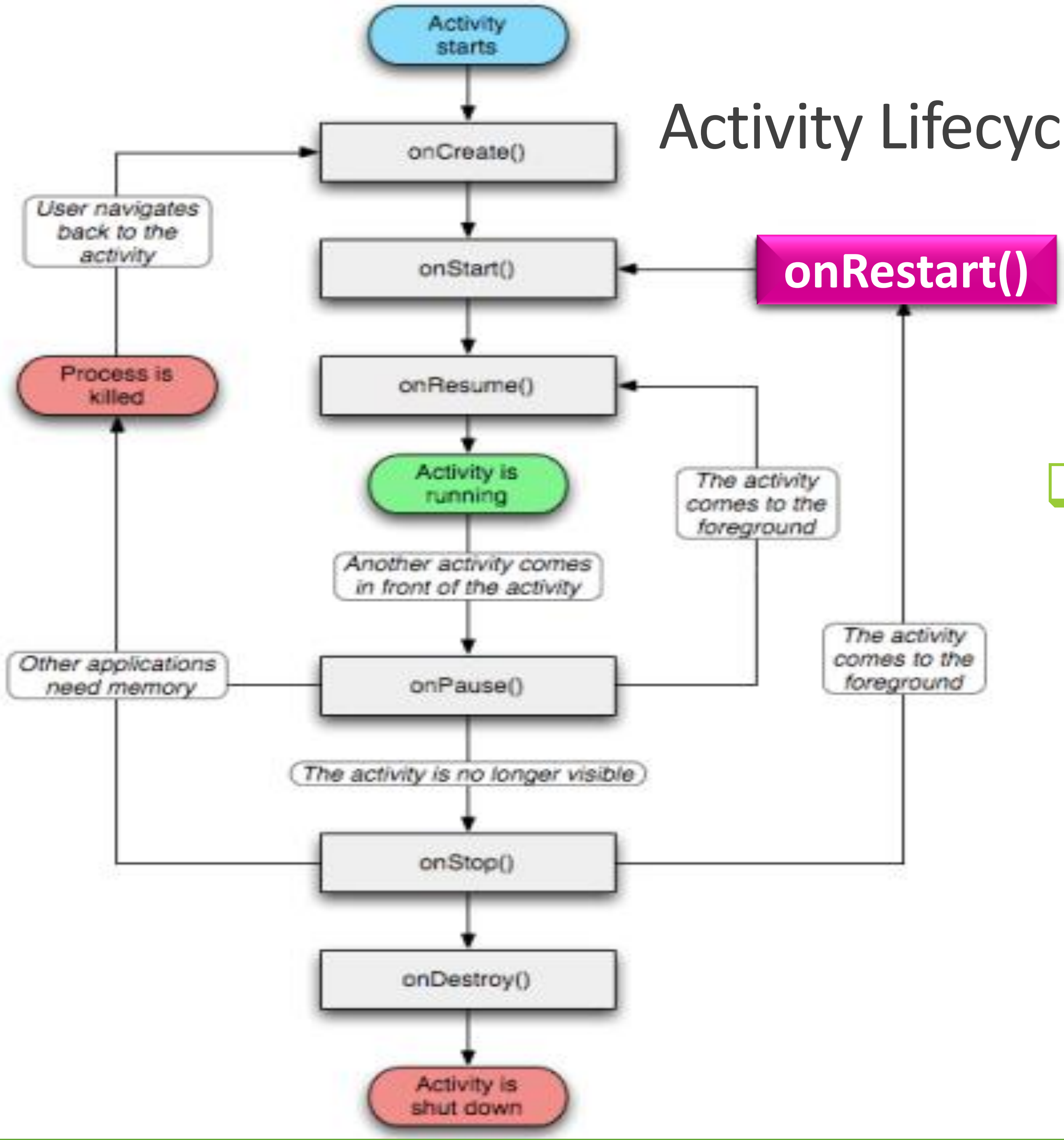
Activity Lifecycle : OnResume()



Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

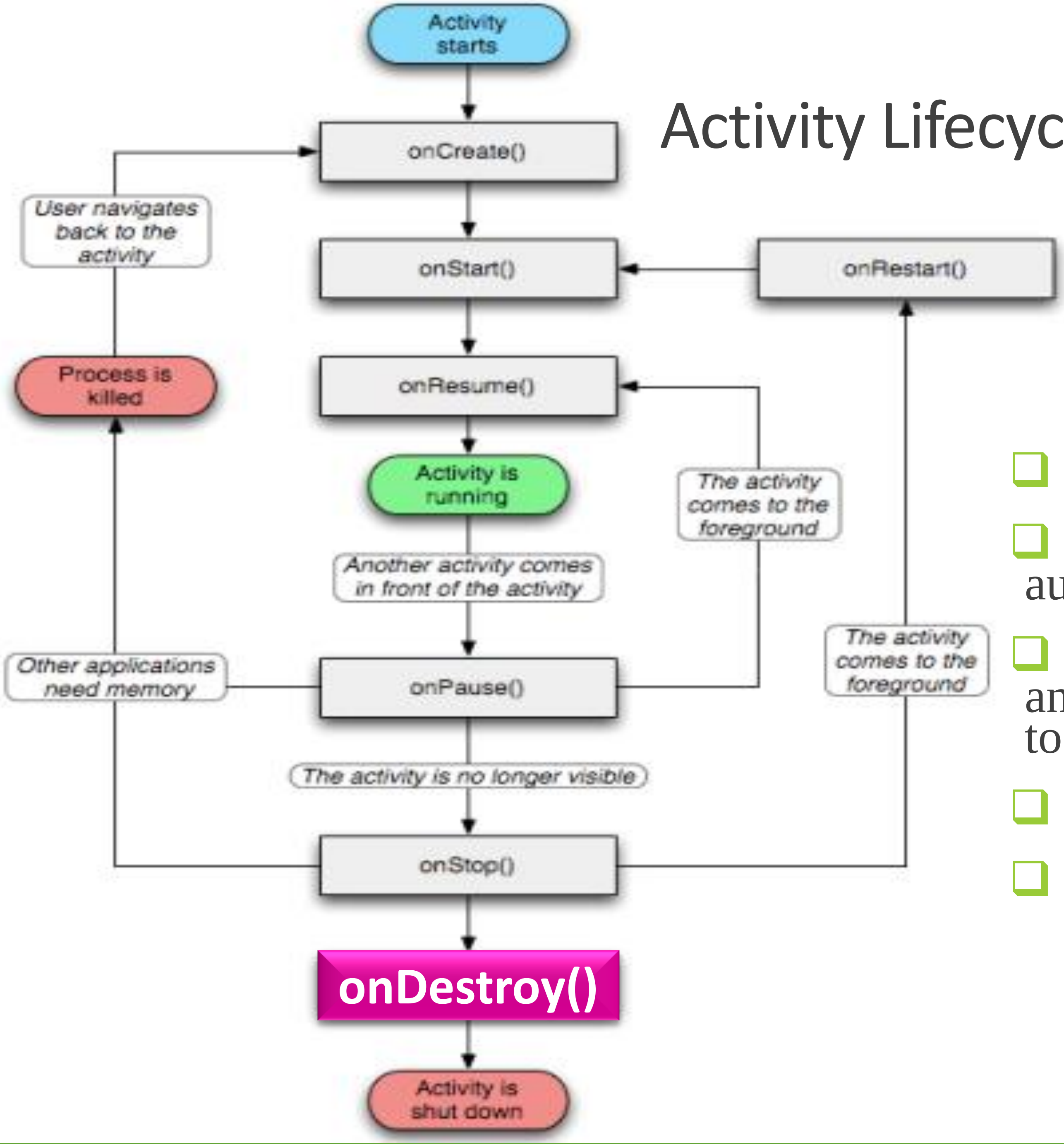
- The opposite of onPause()

Activity Lifecycle : OnRestart()


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
The activity comes to the foreground
Other applications need memory
onPause()
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

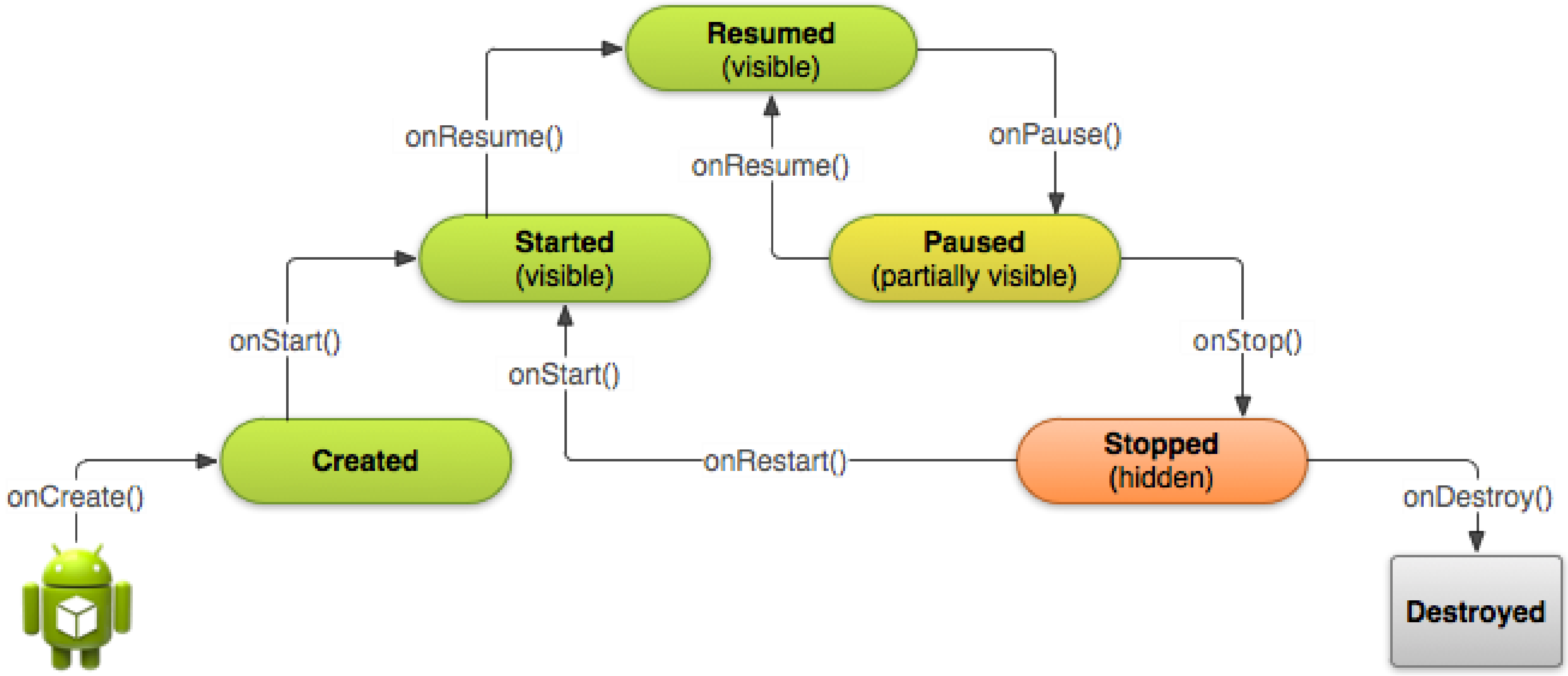
- Calls onStart()

Activity Lifecycle : onDestroy()


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
The activity comes to the foreground
Other applications need memory
onPause()
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

- ❑ Final exit
- ❑ Clean up happens automatically
- ❑ But if you have spawned any threads, you might have to kill them
- ❑ Might not be called at all !
- ❑ Don’t save state here

Activity State Transitions and Methods



App components

Four different kinds of components

- **Activities**
  - ✓ **Single Screen**
- Services
  - ✓ Background process
- Broadcast receivers
  - ✓ Route, present to status bar
- Content providers
  - ✓ Databases

Intents

- With the exception of content providers, all components exchange messages
  - These messages are called ***intents***
  - Think of them as asynchronous method calls

Manifest file

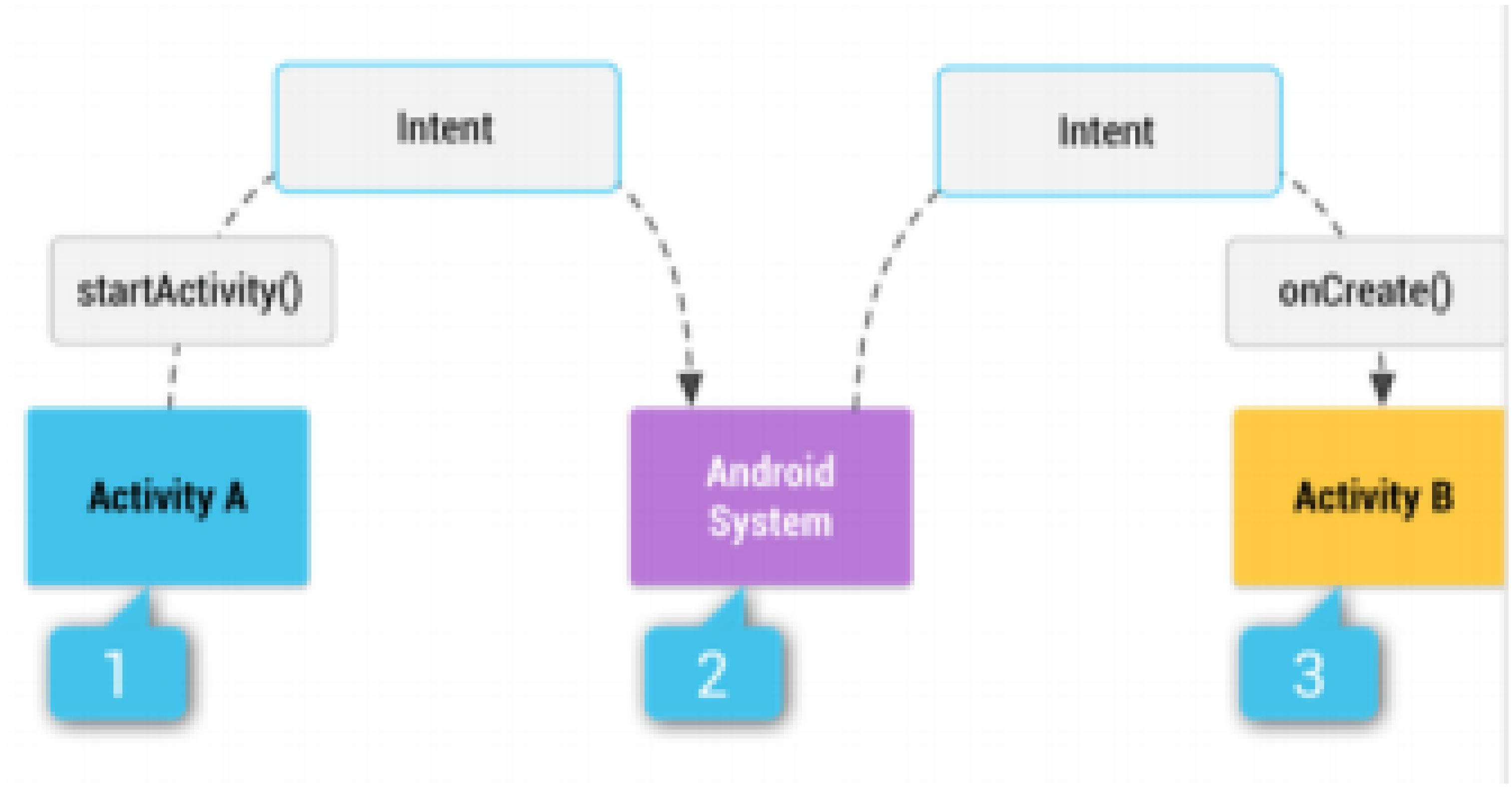
- AndroidManifest.xml
- All components have to be registered there
- http://developer.android.com/guide/topics/manifest/manifest-intro.html
- Android also picks up component information from here
- Other apps can make use of our components

Starting a new activity

- Define a class that sub-classes Activity
- Add some GUI control to invoke it from the parent activity
- Listen for the relevant event, then launch a new Intent
- This will indirectly call the new Activity's method :
  - onCreate(Bundle savedInstance)
- The new activity will start and enter then Resumed state via the call graph shown previously

Pretty pictures

- Looks like this
- Using messages



Intents

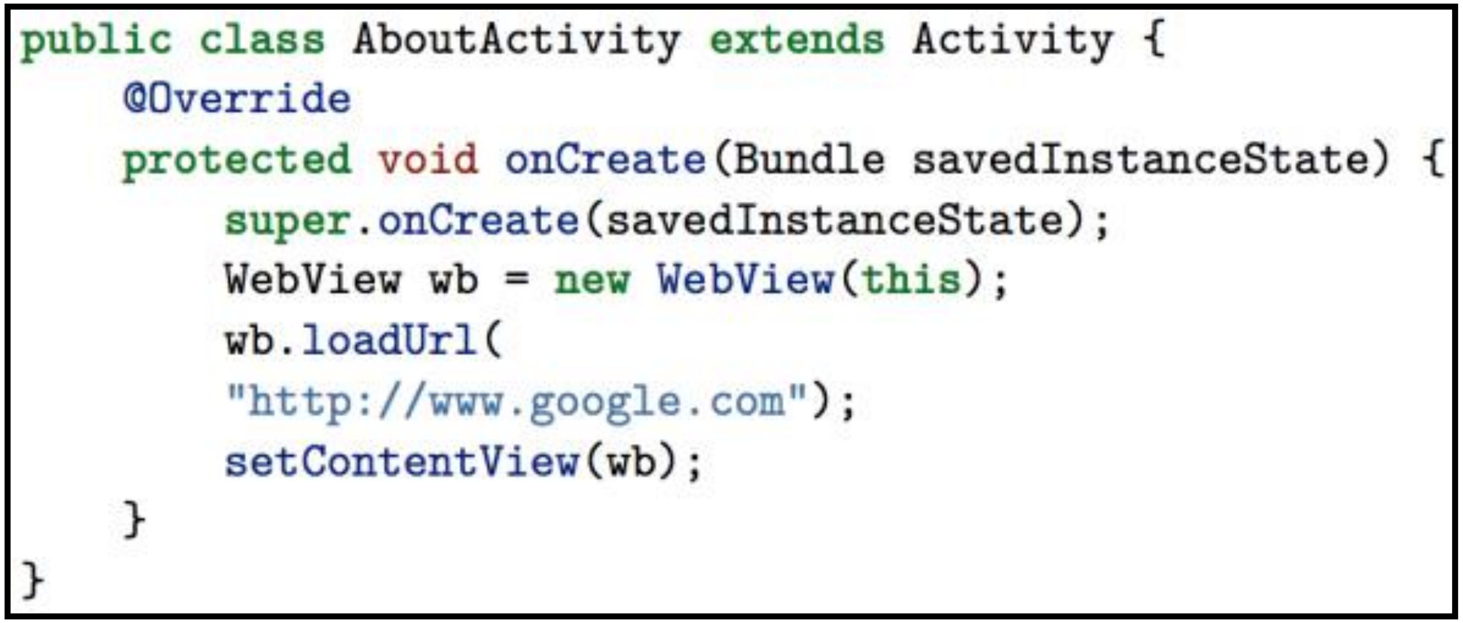
- “An intent is an abstract description of an operation to be performed.” (developer.android.com)
- A bit like a method call
- Two flavours : explicit and implicit
- An explicit Intent specifies exactly which Activity should be started
- An implicit Intent is more declarative : it explains what the Activity should do
- The system will then search for Activities that match by checking the Intent filters
- Example : opening a Web Page (more on this later)

Example

- The following example adds an Activity to provide information about an App
- A menu item called “About” is added to the options menu
- We listen for onOptionItemSelected events within the main activity
- Create an Intent, then call startActivity with the Intent as an argument
- When the user has finished reading the HTML page, the back button can be used to return to the main app
- This behavior is automatic use of the “back stack” ; no need to program it

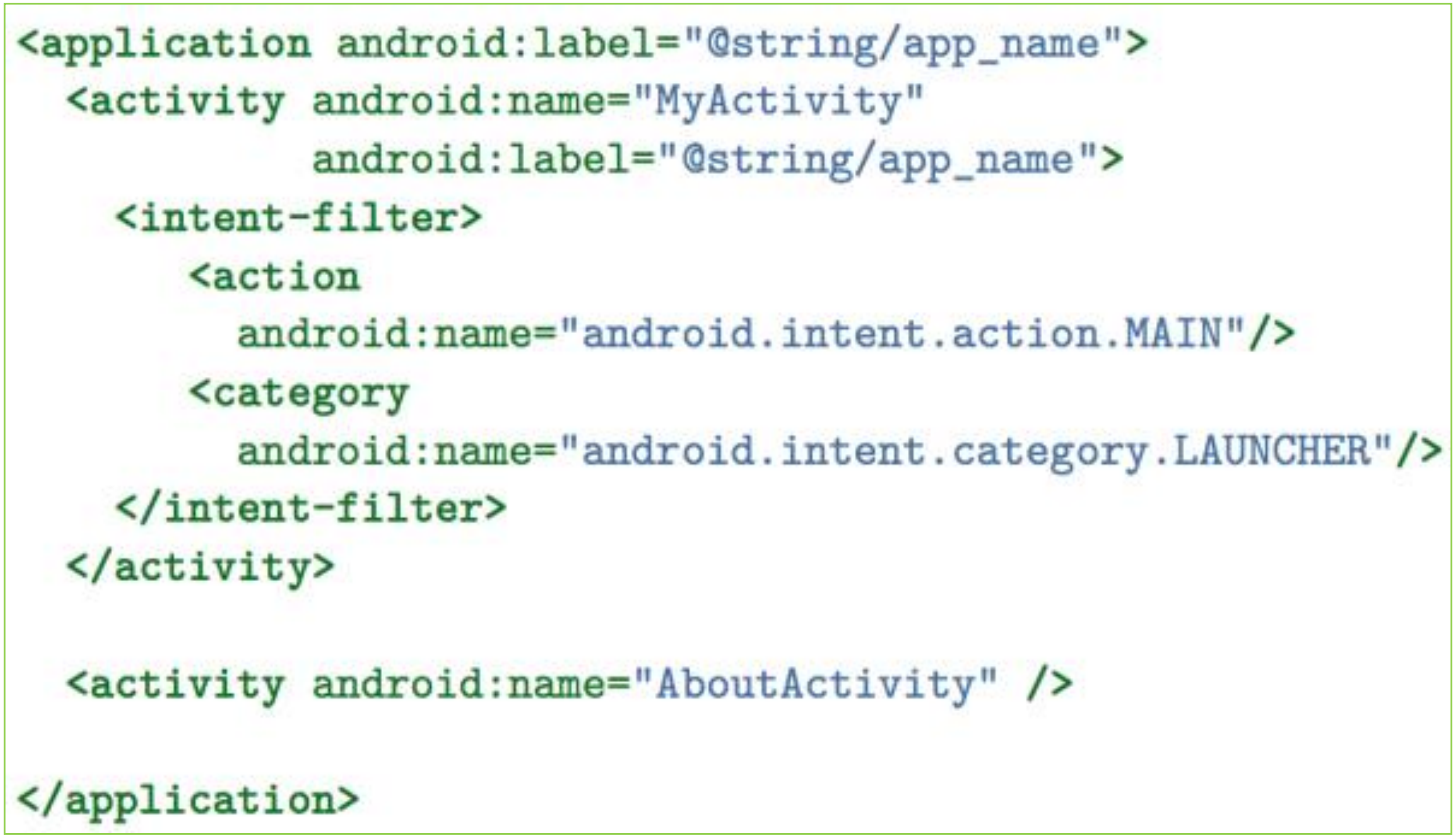
AboutActivity

- Simple example uses a hard-coded HTML file name ; import statements are omitted
- Uses a WebView to display an HTML page specified in loadUrl method)



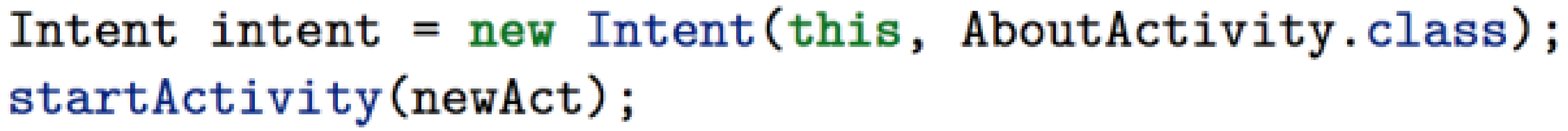
```
public class AboutActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        WebView wb = new WebView(this);
        wb.loadUrl(
        "http://www.google.com");
        setContentView(wb);
    }
}
```

Updating the AndroidManifest.xml



```xml
<application android:label="@string/app_name">
  <activity android:name="MyActivity"
            android:label="@string/app_name">
    <intent-filter>
       <action
         android:name="android.intent.action.MAIN"/>
       <category
         android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>
  </activity>

  <activity android:name="AboutActivity" />

</application>
```

Explicit calling



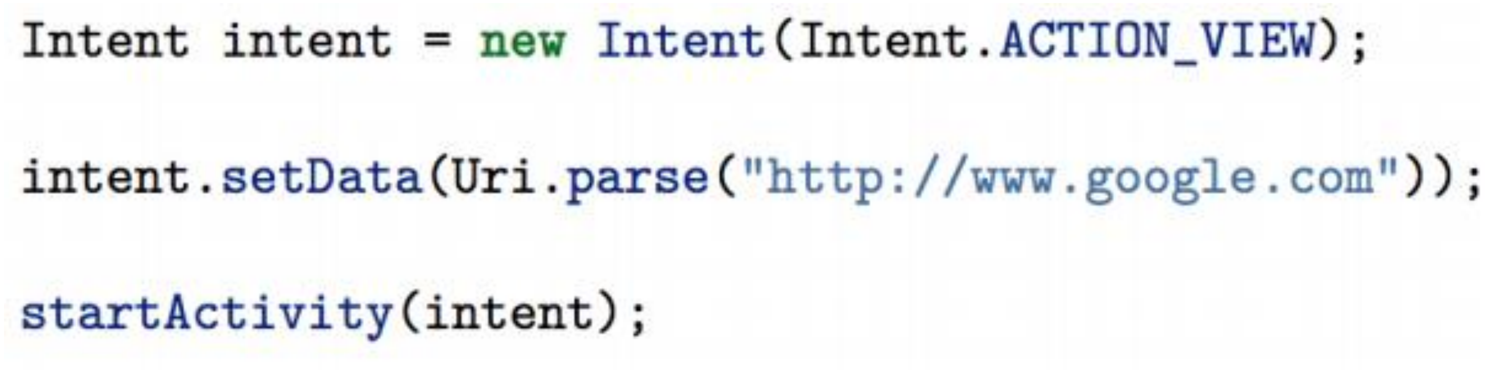
```
Intent intent = new Intent(this, AboutActivity.class);
startActivity(newAct);
```

Add the menu / launching Intent

```java
public boolean onCreateOptionsMenu(Menu menu) {
    menu.add("About");
    return true;
}

public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getTitle().equals("About")) {
        Intent intent =
            new Intent(this, AboutActivity.class);
        startActivity(intent);
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```

Implicit intent ?

- Instead of specifying exactly which Activity class should handle the intent, can instead specify an action e.g. via a URL



```
Intent intent = new Intent(Intent.ACTION_VIEW);

intent.setData(Uri.parse("http://www.google.com"));

startActivity(intent);
```

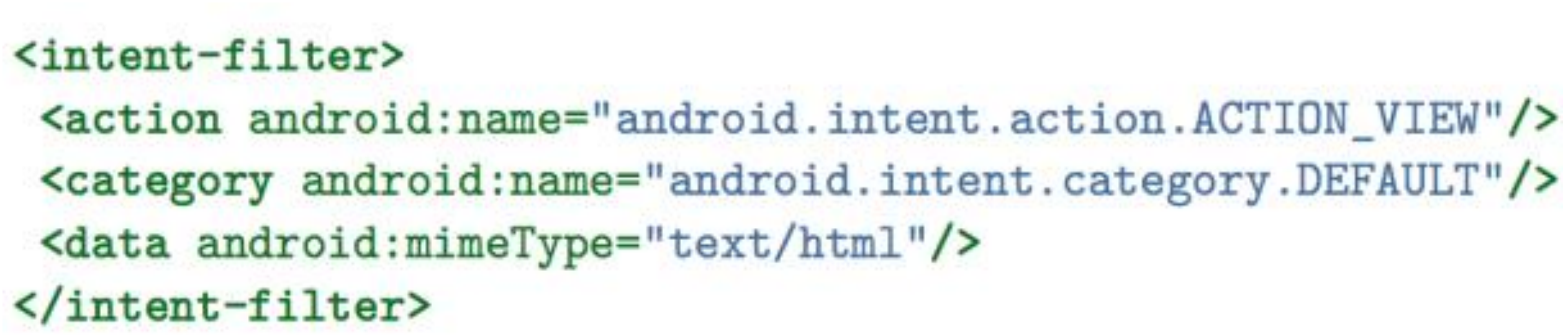
Another example, google maps

- Instead of specifying exactly which Activity class should handle the intent, can instead specify an action e.g. via a URL

```
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("geo:" + 42.516845 +
    "," + -70.898503));
startActivity(intent);
```

Intent filters

- Each activity can declare filters



```
<intent-filter>
 <action android:name="android.intent.action.ACTION_VIEW"/>
 <category android:name="android.intent.category.DEFAULT"/>
 <data android:mimeType="text/html"/>
</intent-filter>
```

- ✓ **How can we call our activity implicitly ?**
- ✓ **Where should we add this filter in our case ?**

Questions?

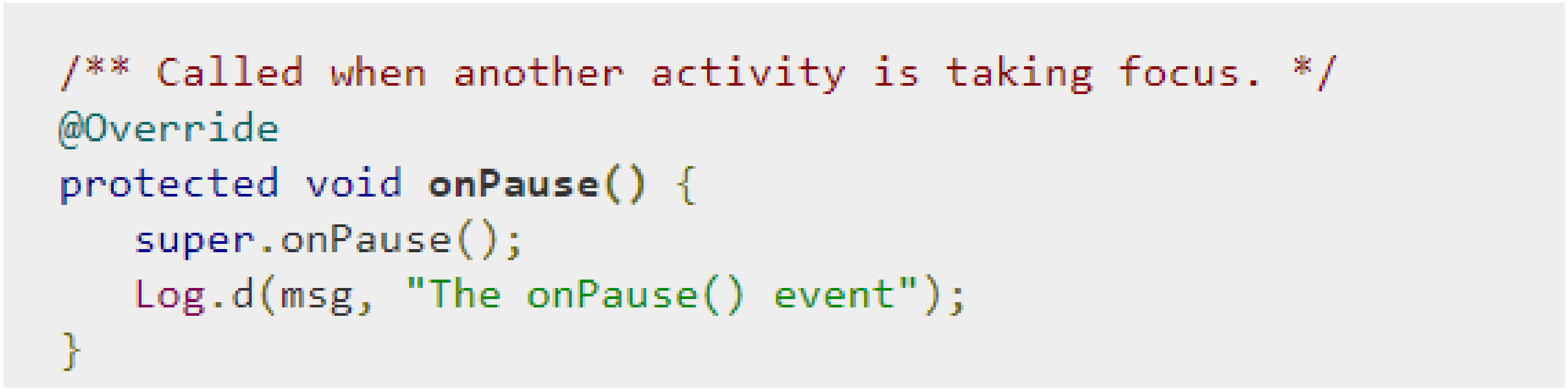
EXAMPLE

```java
package com.example.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.util.Log;

public class MainActivity extends Activity {
   String msg = "Android : ";

   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      Log.d(msg, "The onCreate() event");
   }

   /** Called when the activity is about to become visible. */
   @Override
   protected void onStart() {
      super.onStart();
      Log.d(msg, "The onStart() event");
   }

   /** Called when the activity has become visible. */
   @Override
   protected void onResume() {
      super.onResume();
      Log.d(msg, "The onResume() event");
   }

   /** Called when another activity is taking focus. */
   @Override
   protected void onPause() {
      super.onPause();
      Log.d(msg, "The onPause() event");
   }

   /** Called when the activity is no longer visible. */
   @Override
   protected void onStop() {
      super.onStop();
      Log.d(msg, "The onStop() event");
   }

   /** Called just before the activity is destroyed. */
   @Override
   public void onDestroy() {
      super.onDestroy();
      Log.d(msg, "The onDestroy() event");
   }
}
```

```java
package com.example.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.util.Log;

public class MainActivity extends Activity {
   String msg = "Android : ";

   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      Log.d(msg, "The onCreate() event");
   }
```

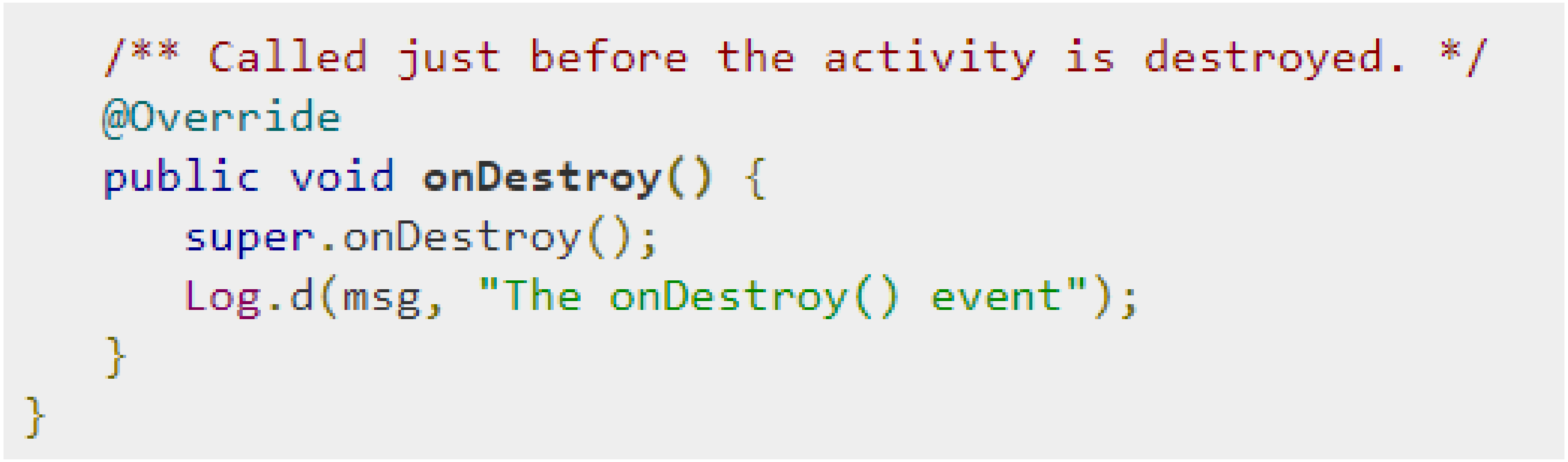
```java
/** Called when the activity is about to become visible. */
@Override
protected void onStart() {
   super.onStart();
   Log.d(msg, "The onStart() event");
}
```

```java
/** Called when the activity has become visible. */
@Override
protected void onResume() {
   super.onResume();
   Log.d(msg, "The onResume() event");
}
```



```java
/** Called when another activity is taking focus. */
@Override
protected void onPause() {
   super.onPause();
   Log.d(msg, "The onPause() event");
}
```

```java
/** Called when the activity is no longer visible. */
@Override
protected void onStop() {
   super.onStop();
   Log.d(msg, "The onStop() event");
}
```



```
    /** Called just before the activity is destroyed. */
    @Override
    public void onDestroy() {
       super.onDestroy();
       Log.d(msg, "The onDestroy() event");
    }
}
```

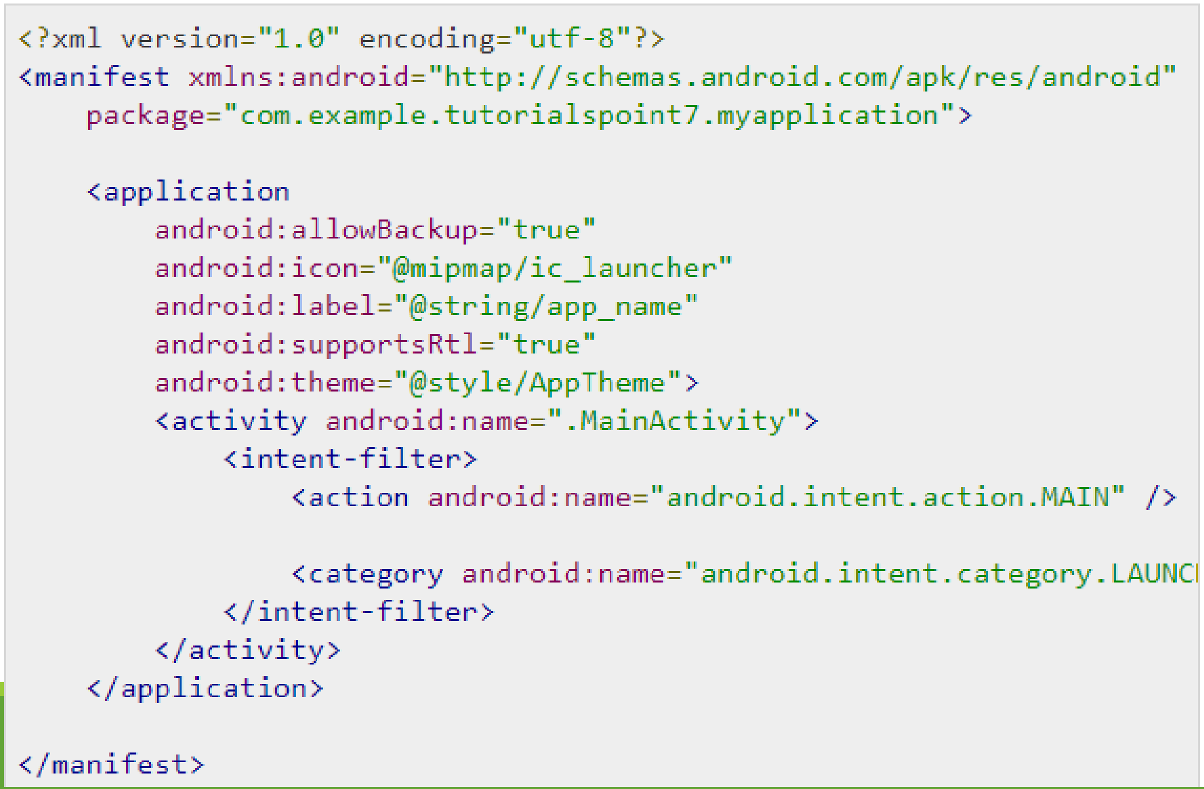
EXAMPLE

An activity class loads all the UI component using the XML file available in *res/layout* folder of the project. Following statement loads UI components from *res/layout/activity_main.xml file*:

```
setContentView(R.layout.activity_main);
```

EXAMPLE

- An application can have one or more activities without any restrictions.
- Every activity you define for your application must be declared in your *AndroidManifest.xml* file
- the main activity for your app must be declared in the manifest with an <intent-filter> that includes the MAIN action and LAUNCHER category as follows:



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.tutorialspoint7.myapplication">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNC
            </intent-filter>
        </activity>
    </application>

</manifest>
```