



# Programming with Android: **Activities**

**Luca Bedogni**

**Marco Di Felice**

**Dipartimento di Informatica: Scienza e Ingegneria**

**Università di Bologna**



# Activity

## ➤ Outline:

- What is started by the device
- It contains the application's informations
- Has methods to answer certain events
- An application could be composed of multiple activities

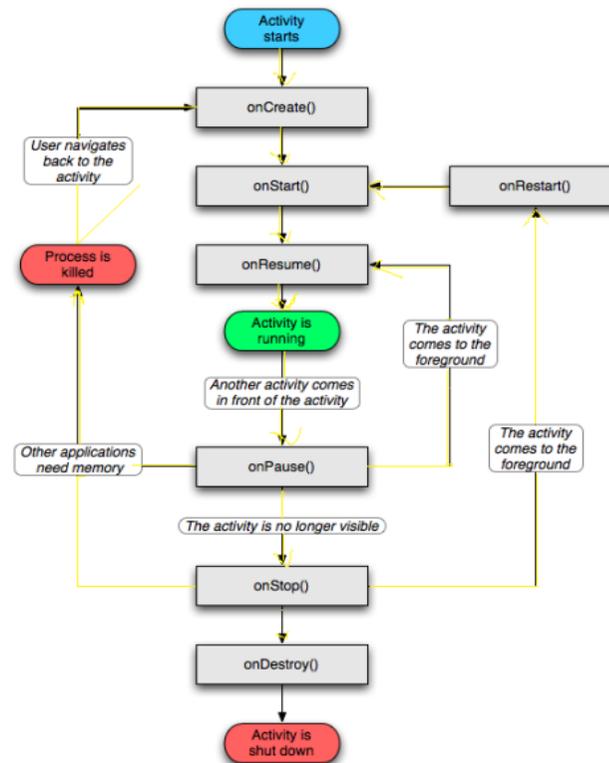


# Creating an activity

- Create a class that is a subclass of Activity
- Implement callback methods
  - `onCreate()`:
    - Initialize
    - `setContentView()`

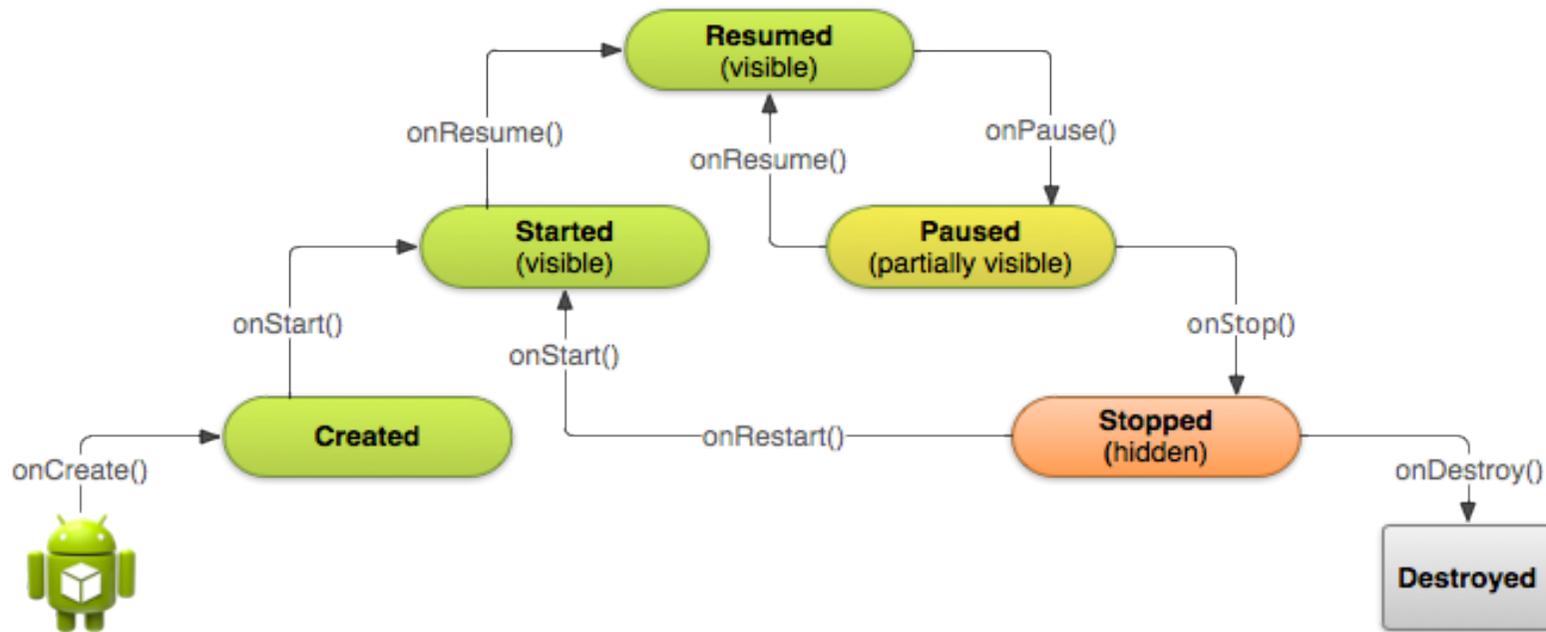


# Activity lifecycle





# Activity lifecycle





# Activities

- Need to implement every single method? No!
  - It depends on the application complexity
- Why it is important to understand the activity lifecycle?
  - So your application does not crash (or do funny things) while the user is running something else on the smartphone
  - So your application does not consume unnecessary resources
  - So the user can safely stop your application and return to it

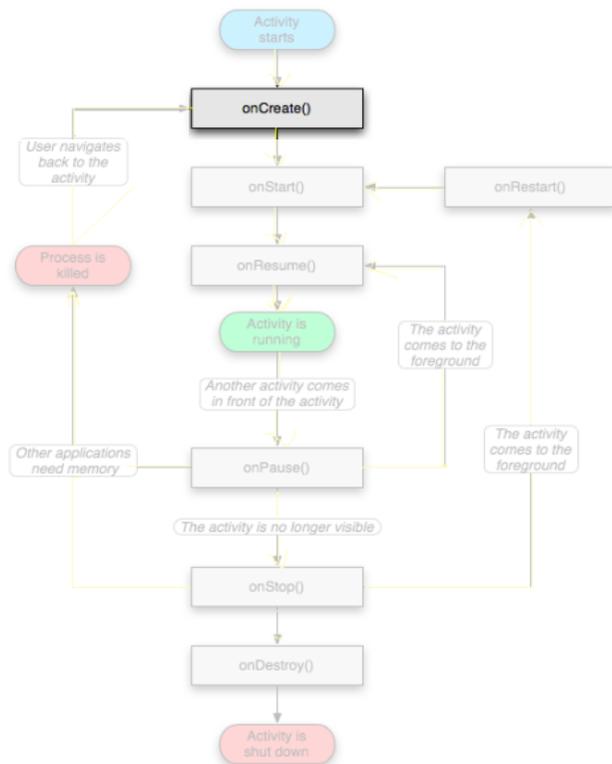


# Activities **states**

- **Resumed**
  - The activity is in the foreground, and the user can interact.
- **Paused**
  - The activity is partially overlaid by another activity. Cannot execute any code nor receive inputs.
- **Stopped**
  - Activity is hidden, in the background. It cannot execute any code.



# Activity lifecycle

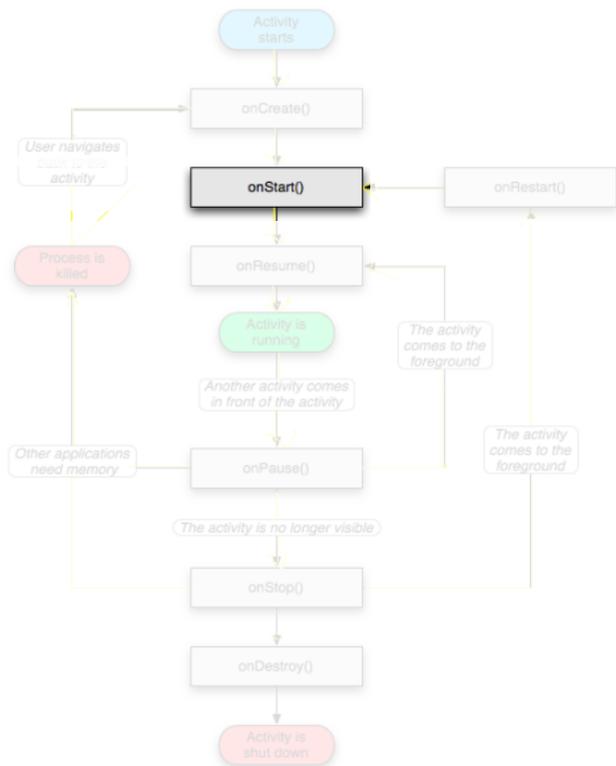


## ➤ onCreate()

- Called when the activity is created
- Should contain the initialization operations
- Has a Bundle parameter
- If `onCreate()` successful terminates, it calls `onStart()`



# Activity lifecycle

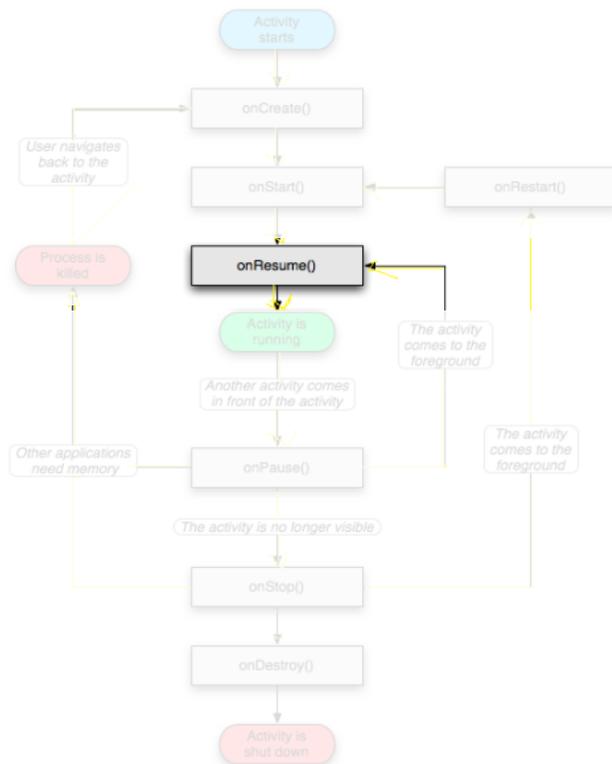


## ➤ onStart()

- Called when onCreate() terminates
- Called right before it is visible to user
- If it has the focus, then onResume() is called
- If not, onStop() is called



# Activity lifecycle

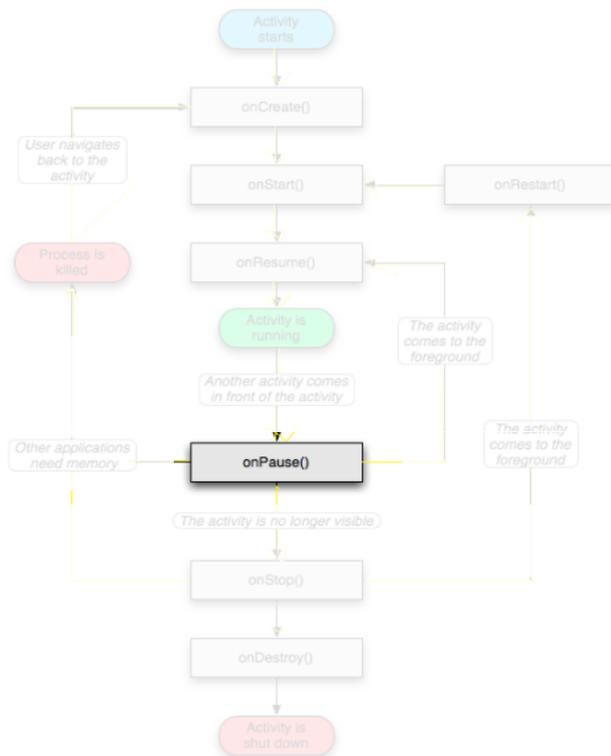


## ➤ OnResume()

- Called when the activity is ready to get input from users
- Called when the activity is resumed too
- If it successfully terminates, then the Activity is RUNNING



# Activity lifecycle

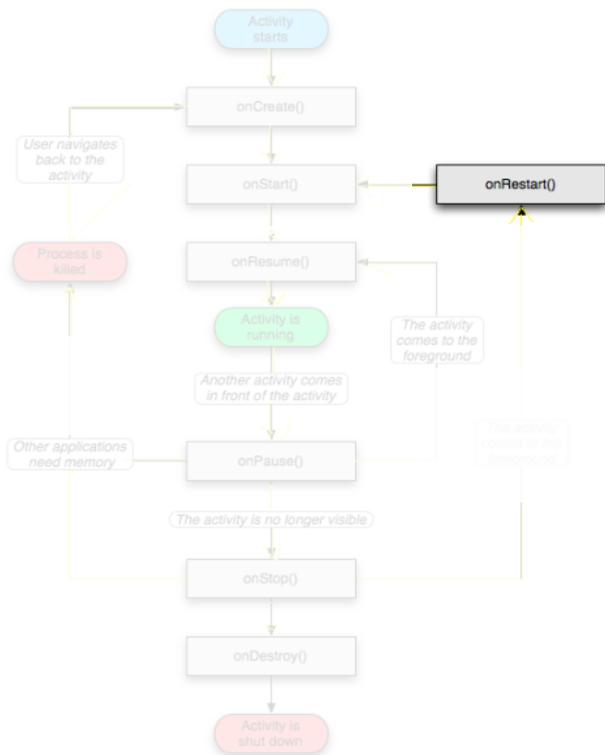


## ➤ onPause()

- Called when another activity comes to the foreground, or when someone presses back
- Commit unsaved changes to persistent data
- Stop cpu-consuming processes
- Make it fast



# Activity lifecycle



## ➤ OnRestart()

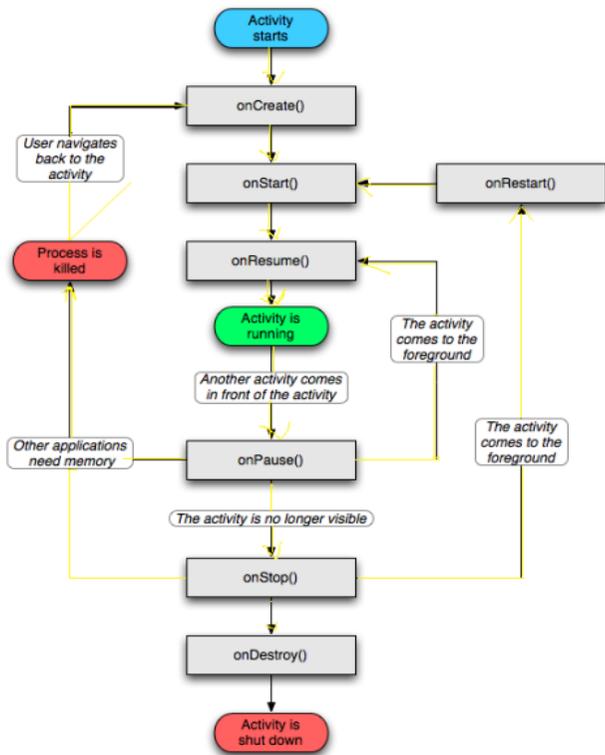
- Similar to `onCreate()`
- We have an activity that was previously stopped







# Activity loops



- Mainly 3 different loops
- **Entire lifetime**
  - Between `onCreate()` and `onDestroy()`.
  - Setup of global state in `onCreate()`
  - Release remaining resources in `onDestroy()`
- **Visible lifetime**
  - Between `onStart()` and `onStop()`.
  - Maintain resources that has to be shown to the user.
- **Foreground lifetime**
  - Between `onResume()` and `onPause()`.



# Activities in the manifest

Declare them before running them

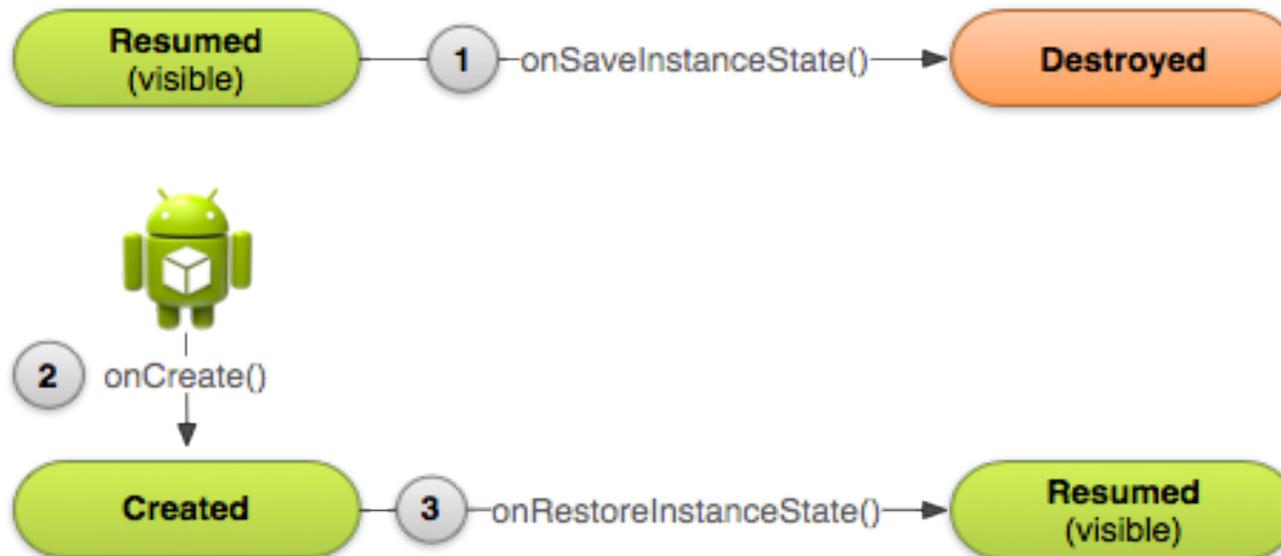
```
<activity android:name=".MainActivity" android:label="@string/  
app_name">  
    <intent-filter>  
        <action android:name="android.intent.action.MAIN" />  
        <category android:name="android.intent.category.LAUNCHER" />  
    </intent-filter>  
</activity>
```

Why “MAIN” and “LAUNCHER”?

To show the application in the menu



# Recreating Activities





# Recreating Activities

- Android keeps the state of each view
  - Remember to assign unique Ids to them
  - So, no code is needed for the “basic” behavior
- What if I want to save more data?
  - Override `onSaveInstanceState()` and `onRestoreInstanceState()`

```
static final String STATE_SCORE = "playerScore";  
  
@Override  
public void onSaveInstanceState(Bundle savedInstanceState) {  
    savedInstanceState.putInt(STATE_SCORE, mCurrentScore);  
}
```



# Recreating Activities

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState); // Always call the superclass
    first
    if (savedInstanceState != null) {
        // Restore value of members from saved state
        mCurrentScore = savedInstanceState.getInt(STATE_SCORE);
    } else {
        // Probably initialize members with default values for a new
        instance
    }
}

public void onRestoreInstanceState(Bundle savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);
}
```



## Activity: Conclusions

- Activities should be declared in the Manifest
- Extend the Activity class
- Code wisely
  - Put your code in the right place
  - Optimize it
  - Test even on low-end devices