ITW202: Mobile Application

Unit IV: Developing for Android

Ms. Sonam Wangmo

Gyalpozhing College of Information Technology
Royal University of Bhutan

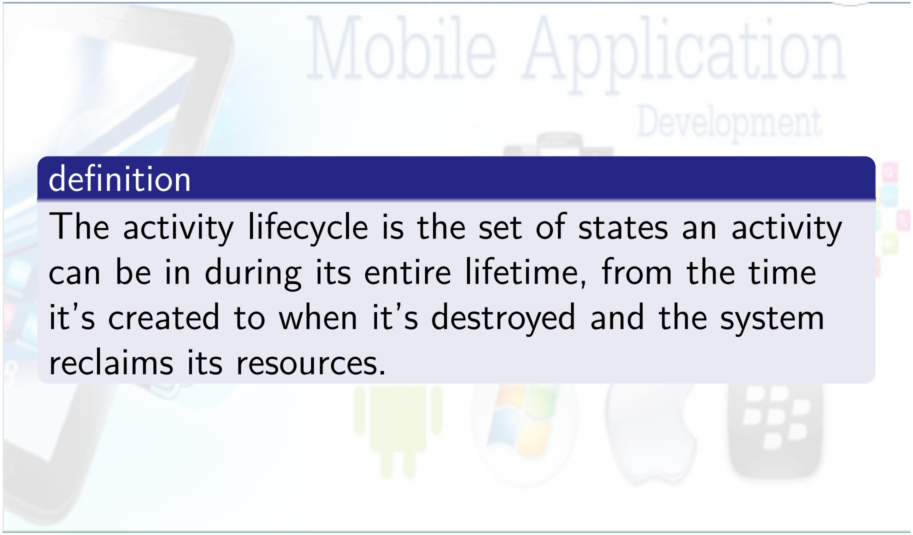
March 23, 2021

Activity lifecycle and state

Contents

- Activity lifecycle
- Activity lifecycle callbacks
- Activity instance state
- Saving and restoring Activity state

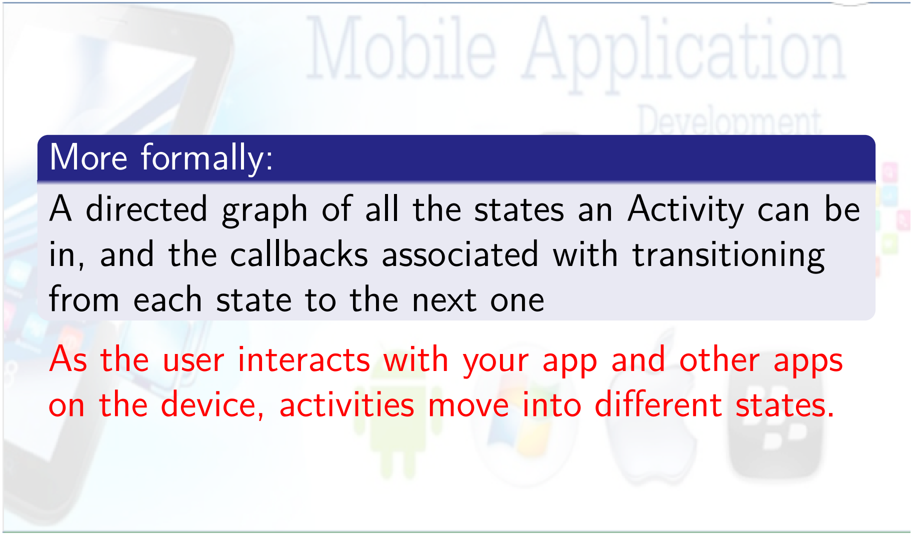
What is the Activity Lifecycle?



definition

The activity lifecycle is the set of states an activity can be in during its entire lifetime, from the time it's created to when it's destroyed and the system reclaims its resources.

What is the Activity Lifecycle?

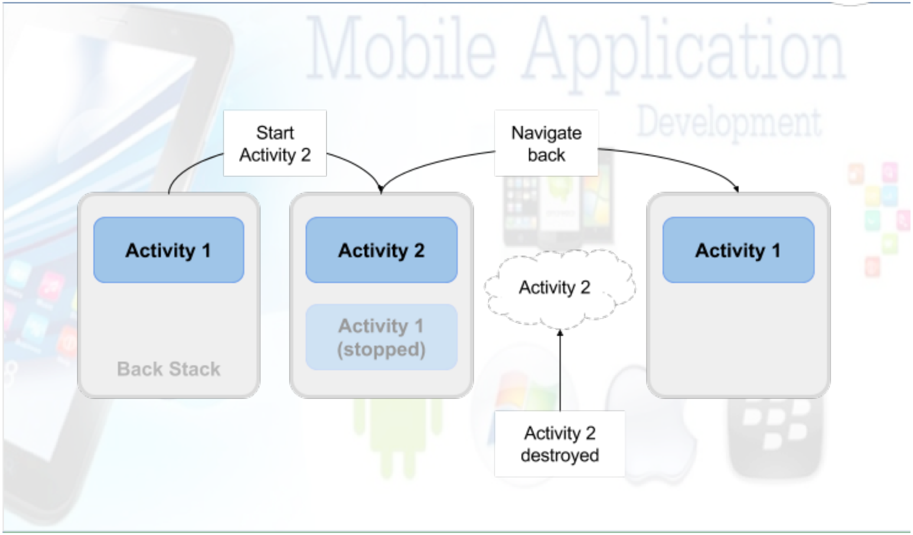


More formally:

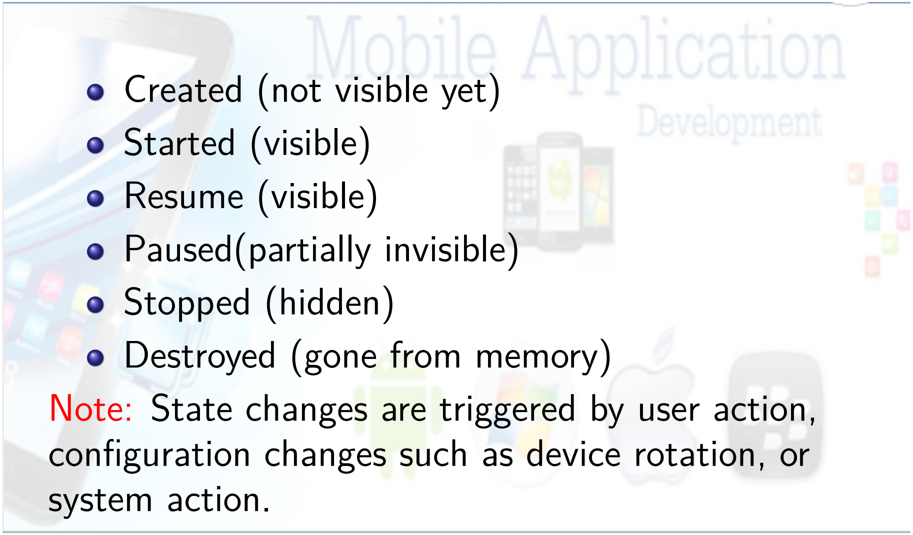
A directed graph of all the states an Activity can be in, and the callbacks associated with transitioning from each state to the next one

As the user interacts with your app and other apps on the device, activities move into different states.

What is the Activity Lifecycle?



Activity states and app visibility

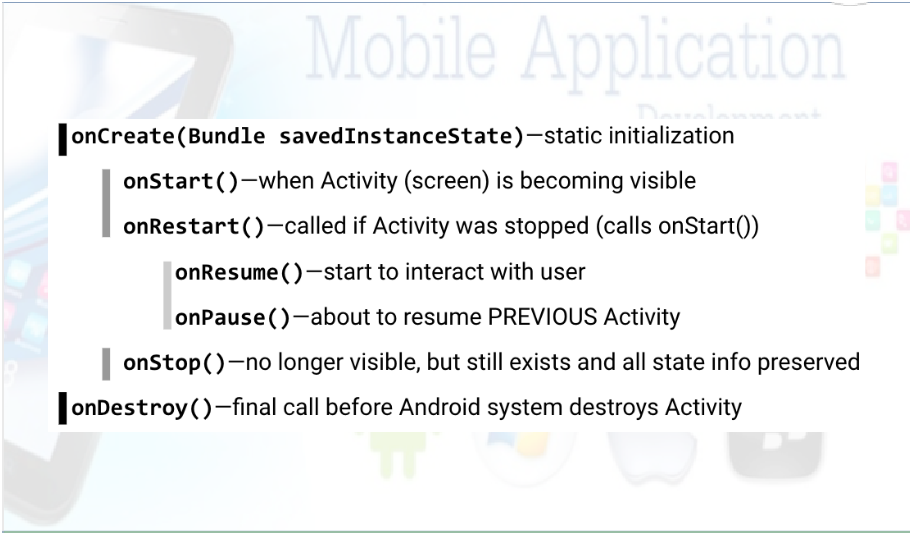


- Created (not visible yet)
- Started (visible)
- Resume (visible)
- Paused(partially invisible)
- Stopped (hidden)
- Destroyed (gone from memory)

Note: State changes are triggered by user action, configuration changes such as device rotation, or system action.

Activity lifecycle callbacks

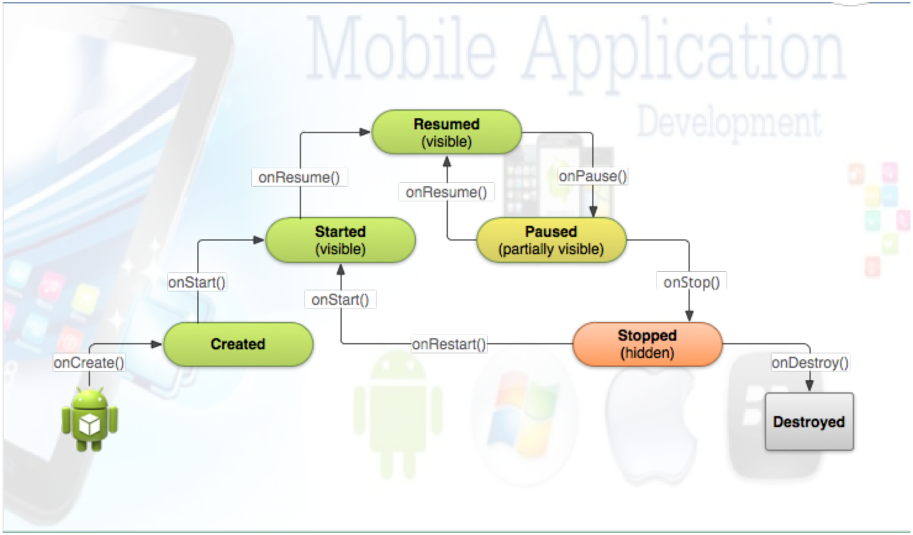
Callbacks and when they are called



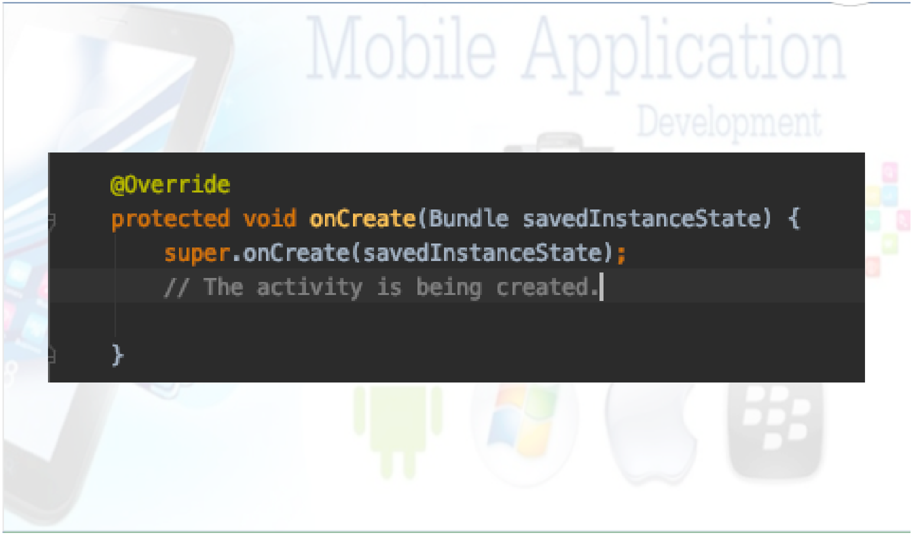
Mobile Application

- **`onCreate(Bundle savedInstanceState)`**—static initialization
    - **`onStart()`**—when Activity (screen) is becoming visible
    - **`onRestart()`**—called if Activity was stopped (calls onStart())
        - **`onResume()`**—start to interact with user
        - **`onPause()`**—about to resume PREVIOUS Activity
    - **`onStop()`**—no longer visible, but still exists and all state info preserved
- **`onDestroy()`**—final call before Android system destroys Activity

Activity states and callbacks graph



Activity created: the onCreate() method



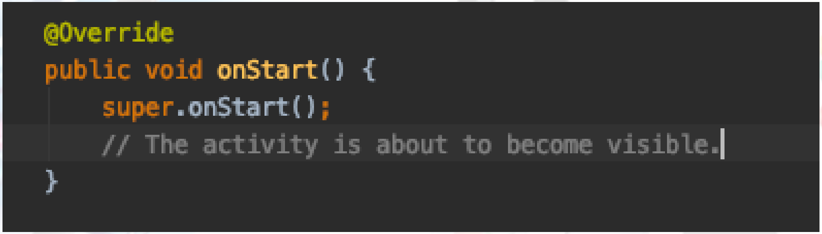
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // The activity is being created.

}
```

Activity created: the onCreate() method

- Called when the Activity is first created, for example when user taps launcher icon
- Does all static setup: such as setting up the user interface, assigning class-scope variables, or setting up background tasks.
- Only called once during an activity's lifetime
- Takes a Bundle with Activity's previously frozen state, if there was one
- Created state is always followed by onStart()

Activity started: the onStart() method

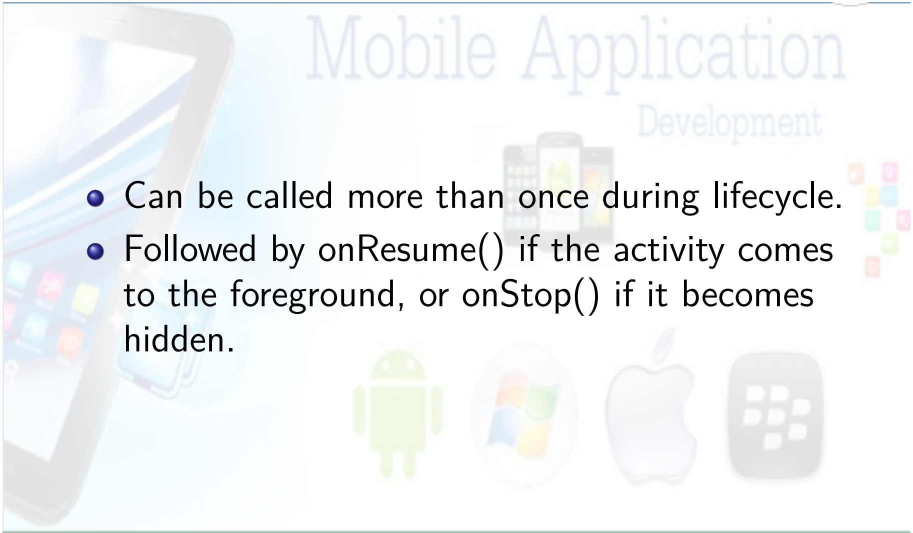


```
@Override
public void onStart() {
    super.onStart();
    // The activity is about to become visible.
}
```

Activity started: the onStart() method

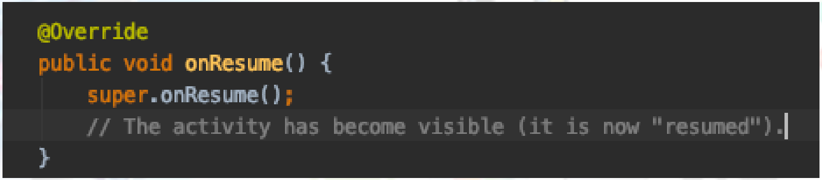
- After your Activity is initialized with onCreate(), the system calls the onStart() method, and the Activity is in the started state
- The onStart() method is also called if a stopped Activity returns to the foreground, such as when the user clicks the Back button or the Up button to navigate to the previous screen

Activity started: the onStart() method



- Can be called more than once during lifecycle.
- Followed by onResume() if the activity comes to the foreground, or onStop() if it becomes hidden.

Activity resumed/running: the onResume() method

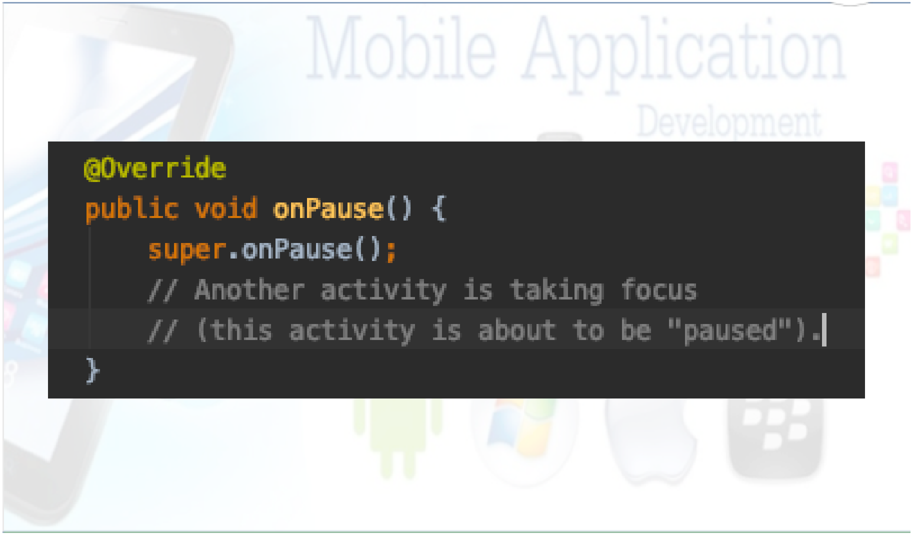


```
@Override
public void onResume() {
    super.onResume();
    // The activity has become visible (it is now "resumed").
}
```

Activity resumed/running: the onResume() method

- Your Activity is in the resumed state when it is initialized, visible on screen, and ready to use.
- The resumed state is often called the running state, because it is in this state that the user is actually interacting with your app.
- Always followed by onPause()

Activity paused: the onPause() method



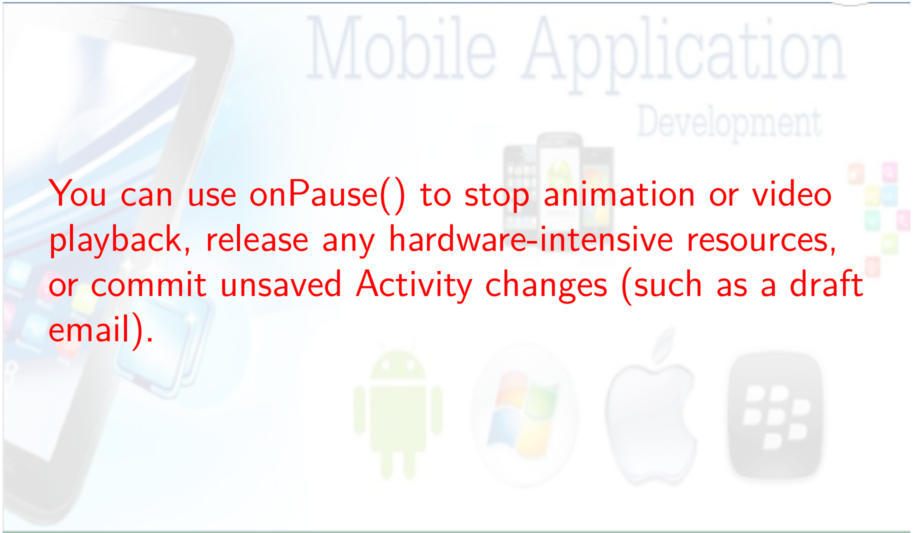
```
@Override
public void onPause() {
    super.onPause();
    // Another activity is taking focus
    // (this activity is about to be "paused").
}
```

Activity paused: the onPause() method

The paused state can occur in several situations:

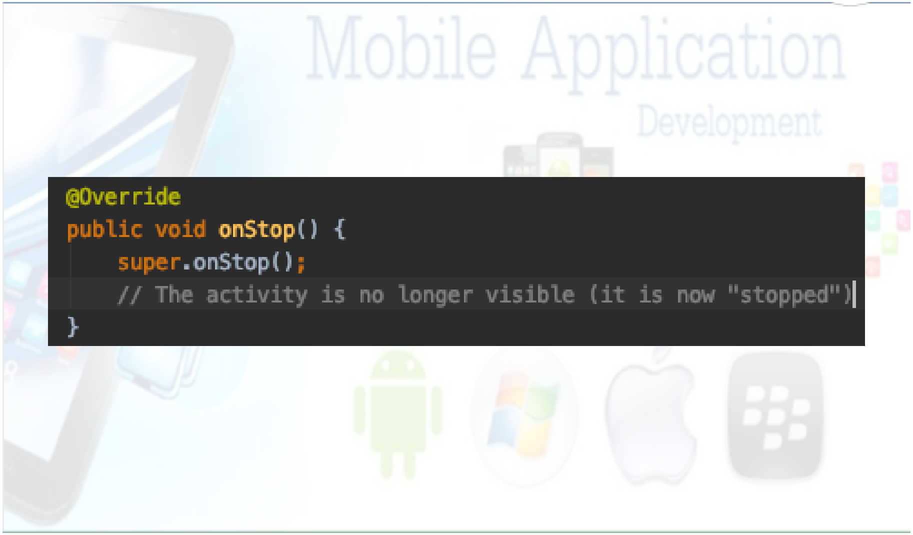
- The Activity is going into the background, but has not yet been fully stopped. This is the first indication that the user is leaving your Activity.
- The Activity is only partially visible on the screen, because a dialog or other transparent Activity is overlaid on top of it.
- Followed by either onResume() if the Activity returns back to the front, or onStop() if it becomes invisible to the user.

Activity paused: the onPause() method



You can use onPause() to stop animation or video playback, release any hardware-intensive resources, or commit unsaved Activity changes (such as a draft email).

Activity stopped: the onStop() method



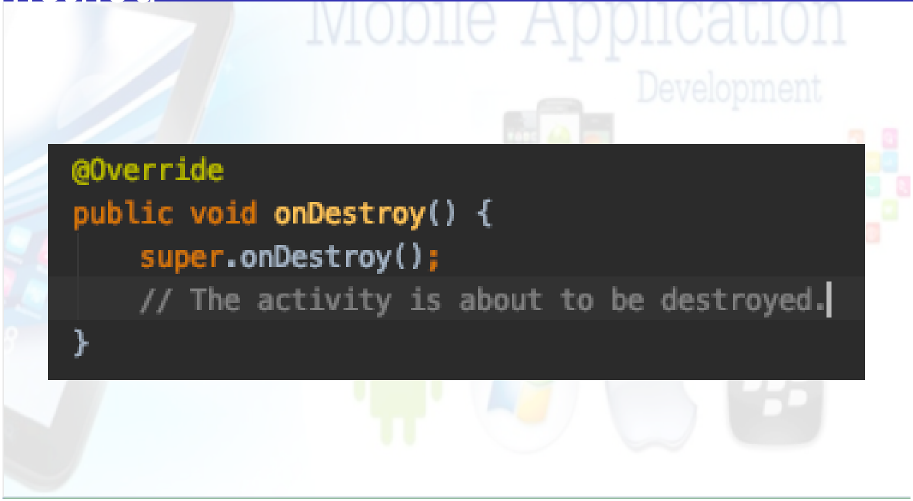
Mobile Application Development

```
@Override
public void onStop() {
    super.onStop();
    // The activity is no longer visible (it is now "stopped")
}
```

Activity stopped: the onStop() method

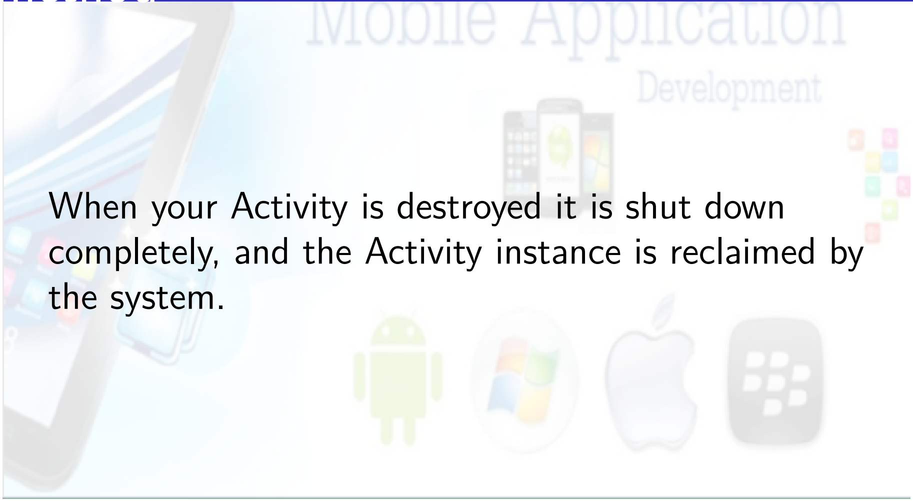
- An Activity is in the stopped state when it's no longer visible on the screen. This is usually because the user started another activity or returned to the home screen.
- Implement the onStop() method to save persistent data and release resources that you didn't already release in onPause().
- Followed by either onRestart() if Activity is coming back to interact with user, or onDestroy() if Activity is going away.

Activity destroyed: the onDestroy() method



```java
@Override
public void onDestroy() {
    super.onDestroy();
    // The activity is about to be destroyed.
}
```

Activity destroyed: the onDestroy() method



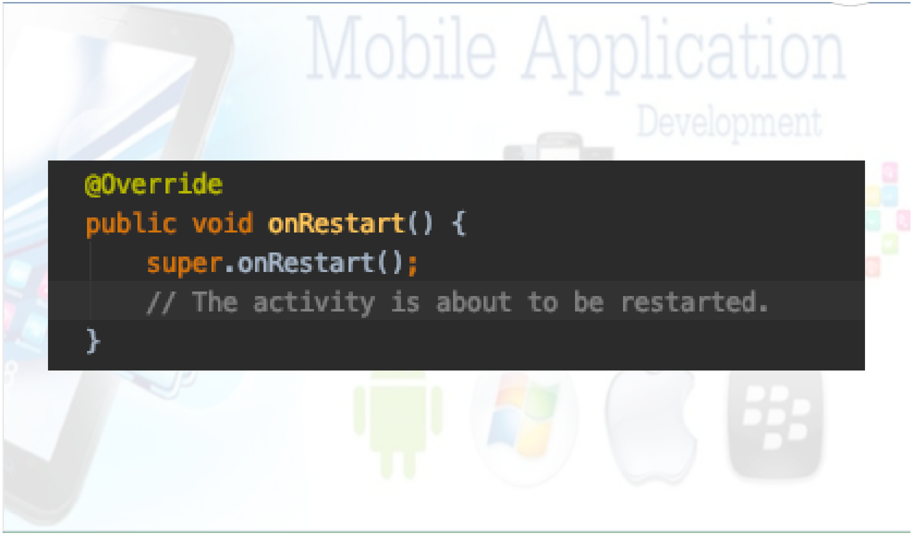
When your Activity is destroyed it is shut down completely, and the Activity instance is reclaimed by the system.

Activity destroyed: the onDestroy() method

This can happen in several cases:

- You call finish() in your Activity to manually shut it down.
- The user navigates back to the previous Activity.
- The device is in a low memory situation where the system reclaims any stopped Activity to free more resources.
- A device configuration change occurs.

Activity restarted: the onRestart() method

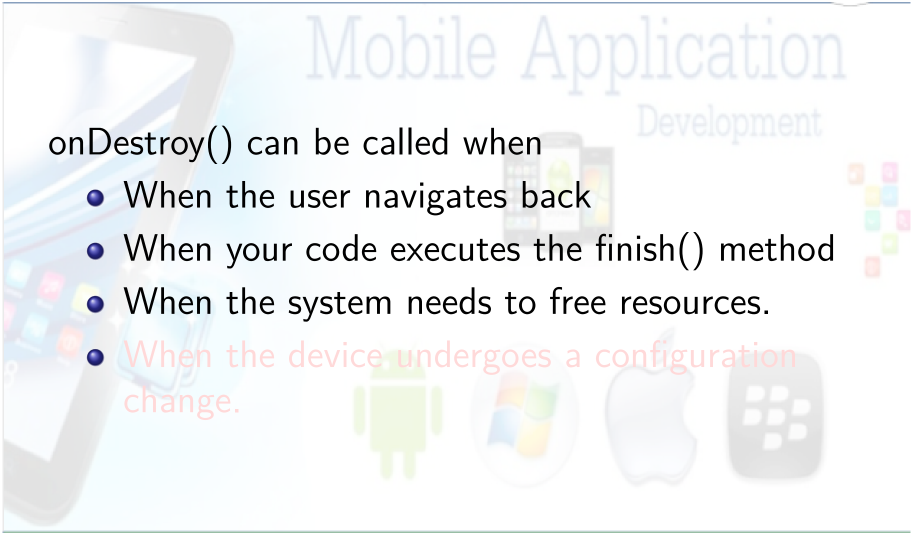


```
@Override
public void onRestart() {
    super.onRestart();
    // The activity is about to be restarted.
}
```

Activity restarted: the onRestart() method

- The restarted state is a transient state that only occurs if a stopped Activity is started again.
- In this case the onRestart() method is called between onStop() and onStart().

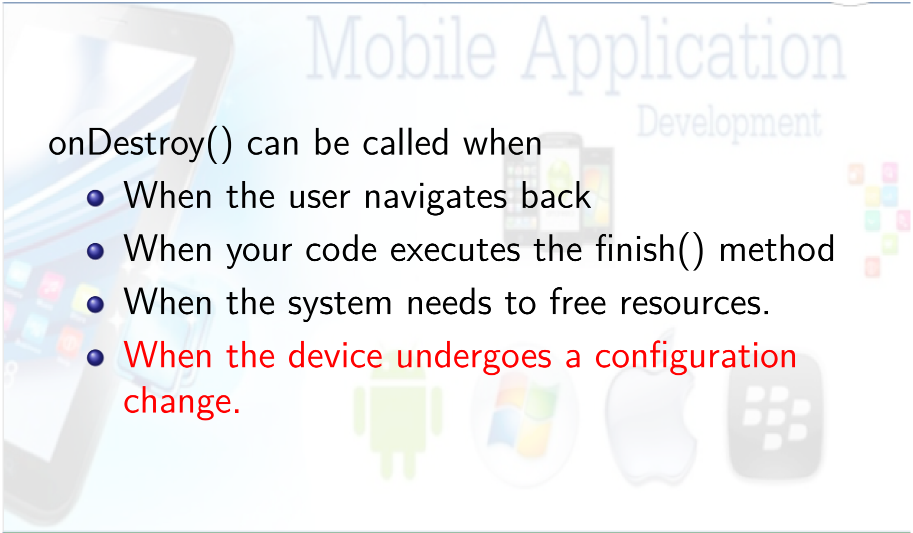
Configuration changes and Activity state



onDestroy() can be called when

- When the user navigates back
- When your code executes the finish() method
- When the system needs to free resources.
- When the device undergoes a configuration change.

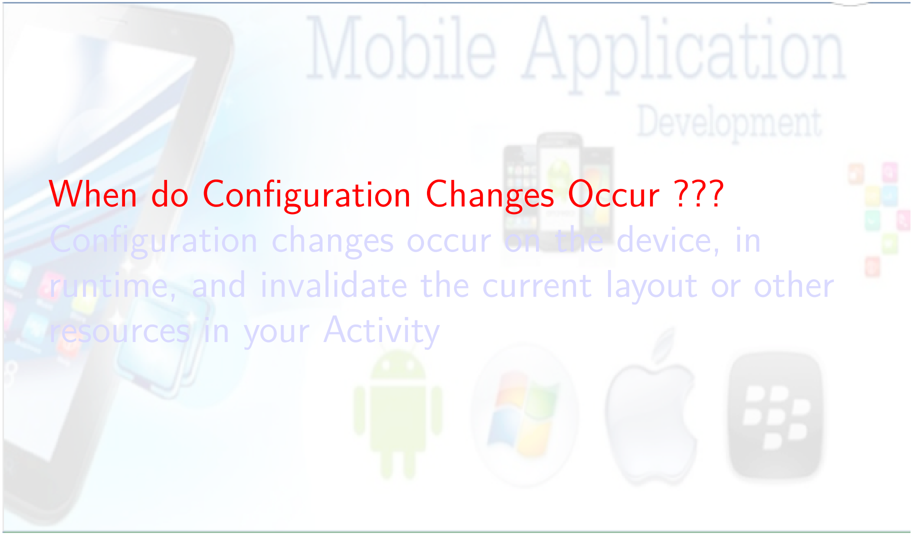
Configuration changes and Activity state



onDestroy() can be called when

- When the user navigates back
- When your code executes the finish() method
- When the system needs to free resources.
- When the device undergoes a configuration change.

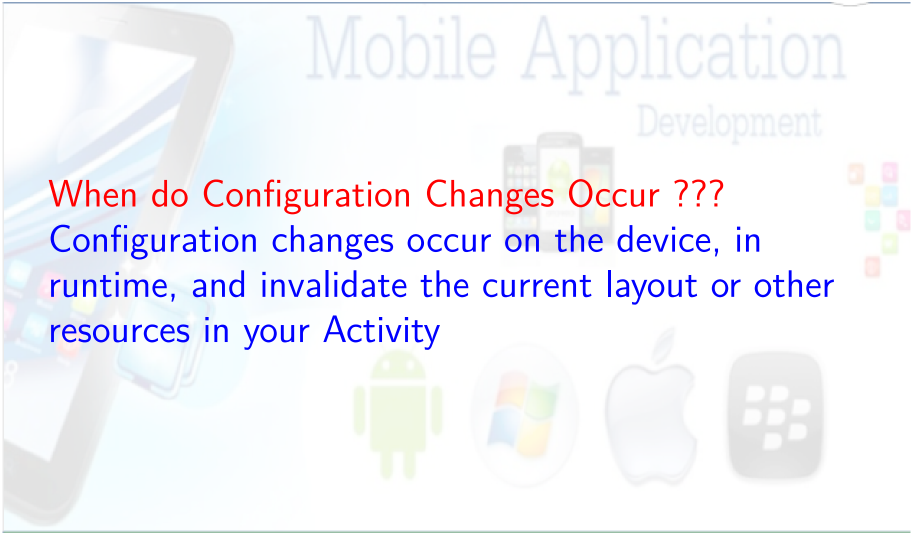
Configuration changes and Activity state



When do Configuration Changes Occur ???

Configuration changes occur on the device, in runtime, and invalidate the current layout or other resources in your Activity

Configuration changes and Activity state



When do Configuration Changes Occur ???

Configuration changes occur on the device, in runtime, and invalidate the current layout or other resources in your Activity

Configuration changes and Activity state

The most common form of a configuration change

- When the device is rotated.
- Chooses different system language, so locale changes
- Enters multi-window mode (from Android 7)

What happens on config change?

On configuration change, Android:

1. Shuts down Activity by calling:
   - onPause()
   - onStop()
   - onDestroy()

What happens on config change?

2. Starts Activity over again by calling:
    - onCreate()
    - onStart()
    - onResume()

Activity instance state

- State information is created while the Activity is running, such as a counter, user text, animation progression
- State is lost when device is rotated, language changes, back-button is pressed, or the system clears memory

Saving and restoring Activity state

What the system saves

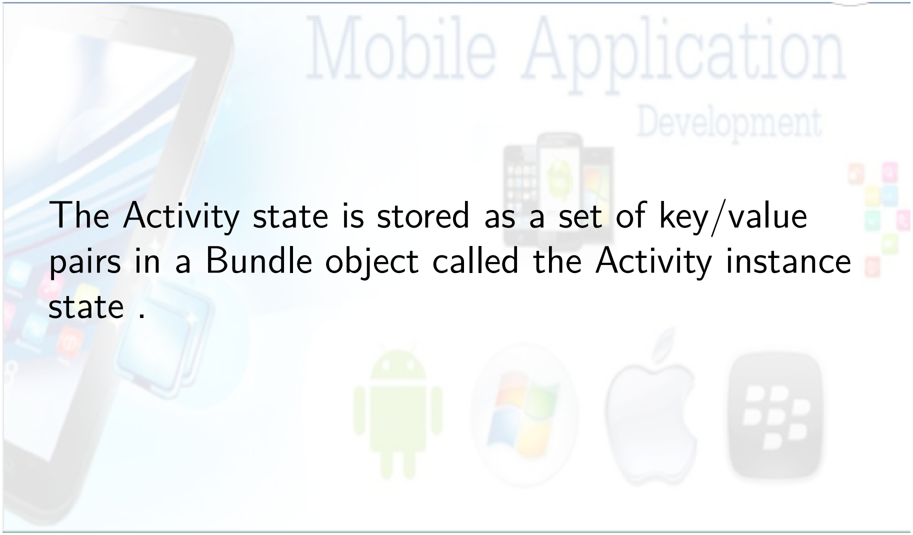
- System saves only:

  State of views with unique ID (android:id) such as text entered into EditText, and

  Intent that started activity and data in its extras

Note: You are responsible for saving other activity and user progress data

Saving instance state



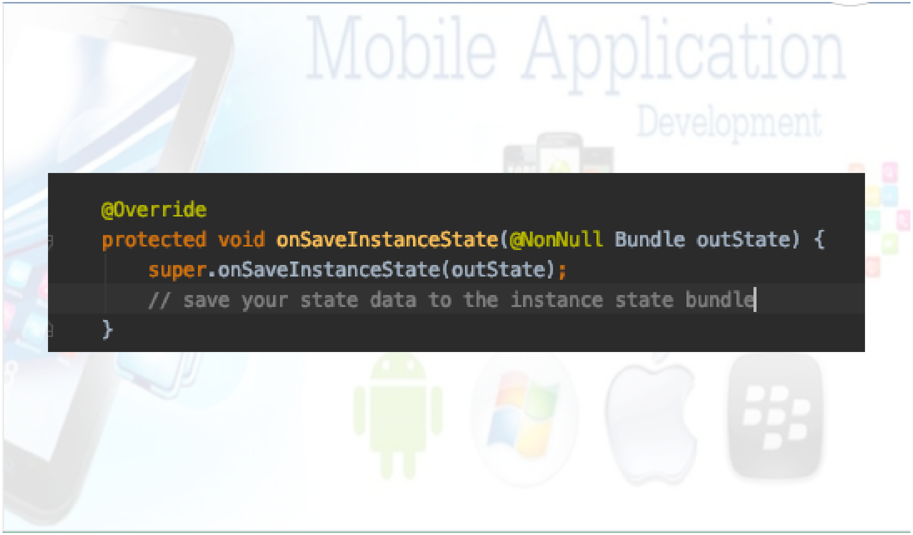
The Activity state is stored as a set of key/value pairs in a Bundle object called the Activity instance state .

Saving instance state

Implement onSaveInstanceState() in your Activity

- Called by Android runtime when there is a possibility the Activity may be destroyed.
- Saves data only for this instance of the Activity during current session.

Saving instance state



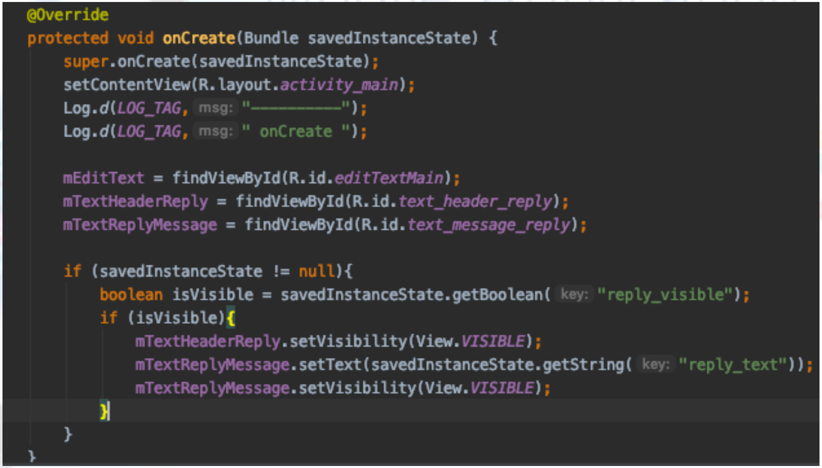
```
@Override
protected void onSaveInstanceState(@NonNull Bundle outState) {
    super.onSaveInstanceState(outState);
    // save your state data to the instance state bundle
}
```

Restoring instance state

Once you've saved the Activity instance state, you also need to restore it when the Activity is recreated. You can do this one of two places:

- The onCreate() callback method, which is called with the instance state Bundle when the Activity is created.
- The onRestoreInstanceState() callback, which is called after onStart() after the Activity is created.

Restoring instance state

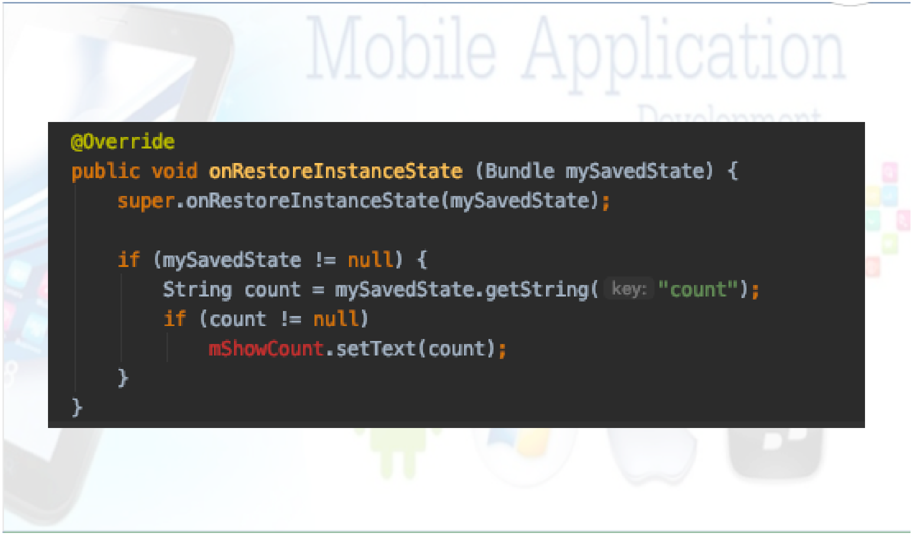


```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Log.d(LOG_TAG, msg: "----------");
    Log.d(LOG_TAG, msg: " onCreate ");

    mEditText = findViewById(R.id.editTextMain);
    mTextHeaderReply = findViewById(R.id.text_header_reply);
    mTextReplyMessage = findViewById(R.id.text_message_reply);

    if (savedInstanceState != null){
        boolean isVisible = savedInstanceState.getBoolean( key: "reply_visible");
        if (isVisible){
            mTextHeaderReply.setVisibility(View.VISIBLE);
            mTextReplyMessage.setText(savedInstanceState.getString( key: "reply_text"));
            mTextReplyMessage.setVisibility(View.VISIBLE);
        }
    }
}
```

Restoring instance state



```
@Override
public void onRestoreInstanceState (Bundle mySavedState) {
    super.onRestoreInstanceState(mySavedState);

    if (mySavedState != null) {
        String count = mySavedState.getString( key: "count");
        if (count != null)
            mShowCount.setText(count);
    }
}
```

Instance state and app restart

- When you stop and restart a new app session, the Activity instance states are lost and your activities will revert to their default appearance.
- If you need to save user data between app sessions, use shared preferences or a database.

THANK YOU