CMSC436: Programming Handheld Systems

Fall 2017

The Activity Class

Today’s Topics

The Activity class

The Task Backstack

The Activity lifecycle

Starting an Activity

Handling configuration changes

The Activity Class

Provides a visual interface for user interaction

Each Activity typically supports one focused thing a user can do, such as

- Viewing an email message
- Showing a login screen

Activities and Application

Applications often comprise several Activities

Android’s Navigation Support

Tasks

The Task Backstack

Suspending and resuming Activities

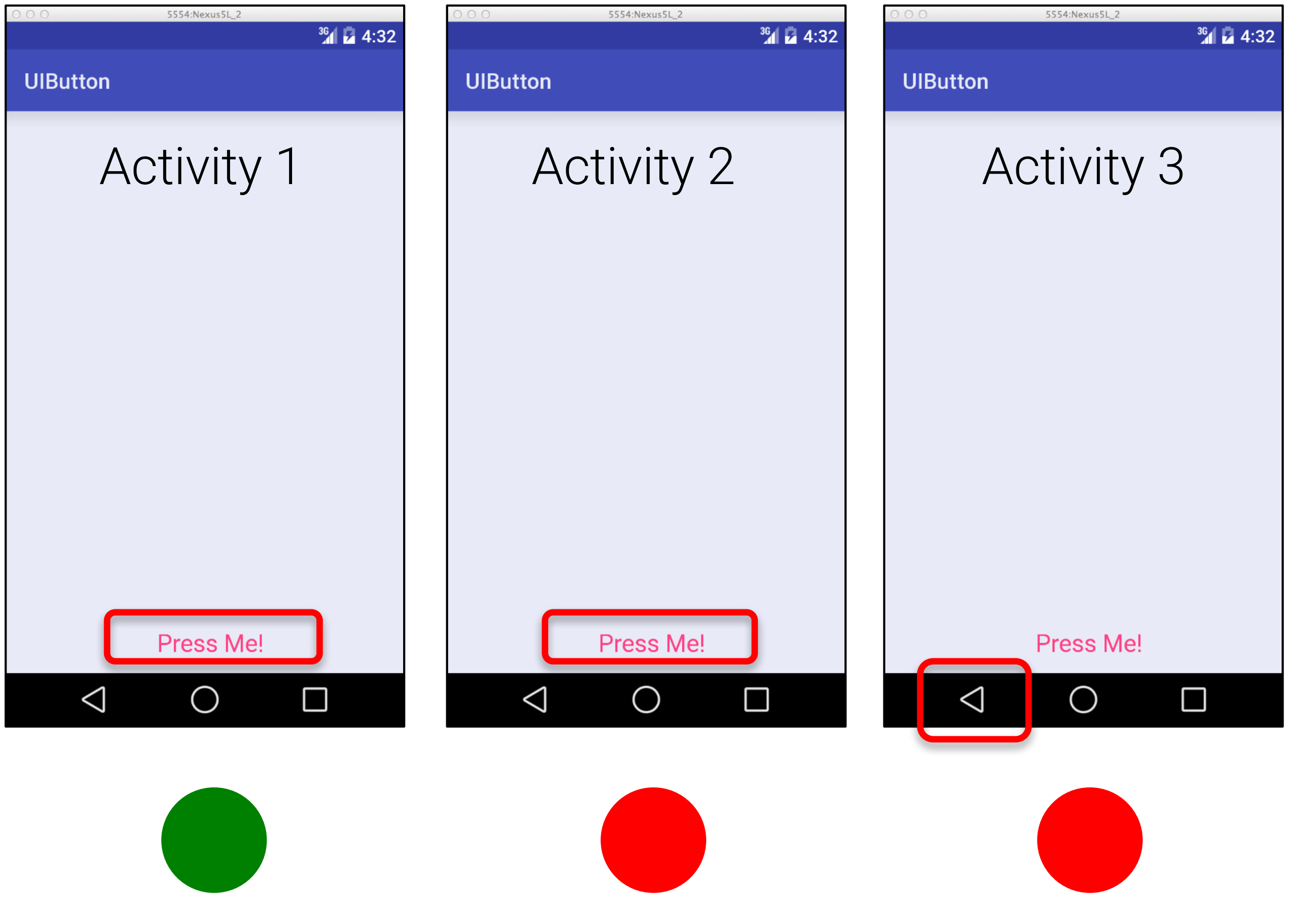
Tasks

- A set of related Activities
  - These Activities can be from different applications
- Most Tasks start at the home screen

Task Backstack

When an Activity is launched, it goes on top of the backstack

When the Activity is destroyed, it is popped off the backstack



Activity 3
Activity 2
Activity 1

Task Backstack

The Activity Lifecycle

- Activities are created, suspended, resumed and destroyed as necessary when an application executes
- Some of these actions depend on user behavior
- Some depend on Android
  - For example, Android can kill Activities when it needs their resources

Activity Lifecycle States

Resumed/Running—Visible, user interacting

Paused—Visible, user not interacting, can be terminated

Stopped—Not visible, can be terminated

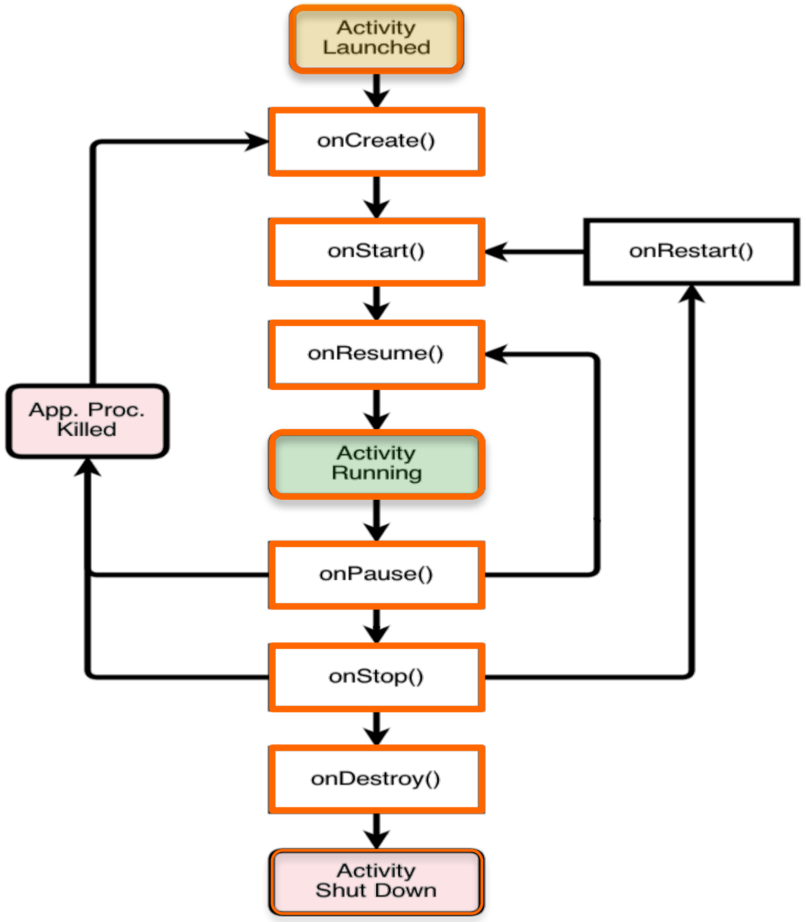
The Activity Lifecycle Methods

Android announces Activity lifecycle state changes to Activities by calling specific Activity methods

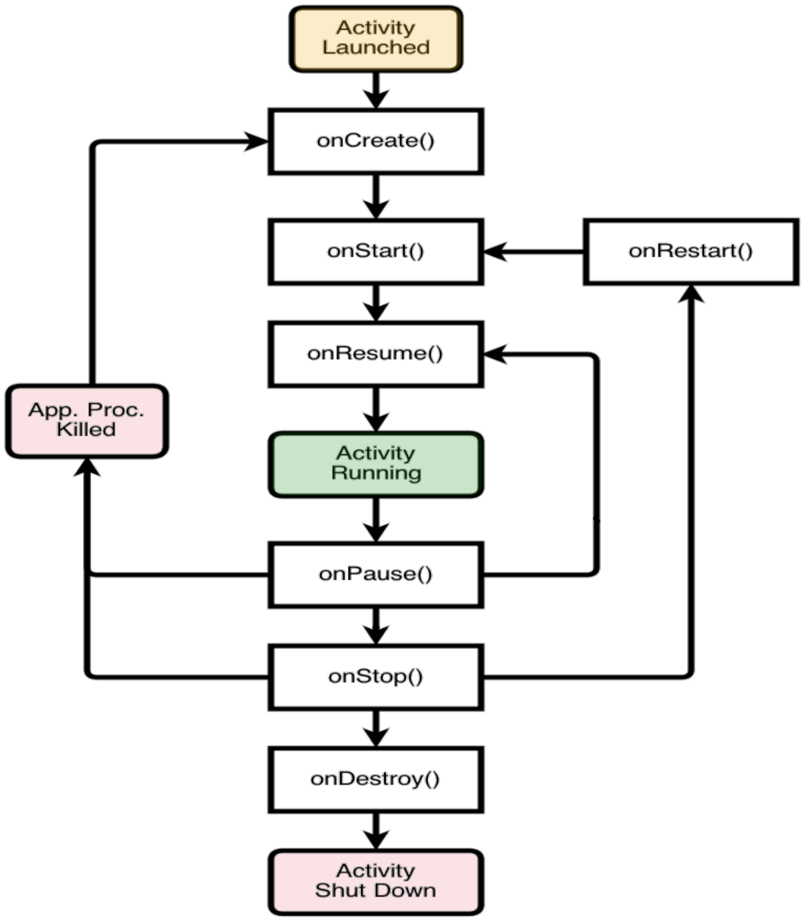
Some Activity Callback Methods

```
protected void onCreate(Bundle savedInstanceState)
protected void onStart()
protected void onResume()
protected void onPause()
protected void onRestart()
protected void onStop()
protected void onDestroy()
```

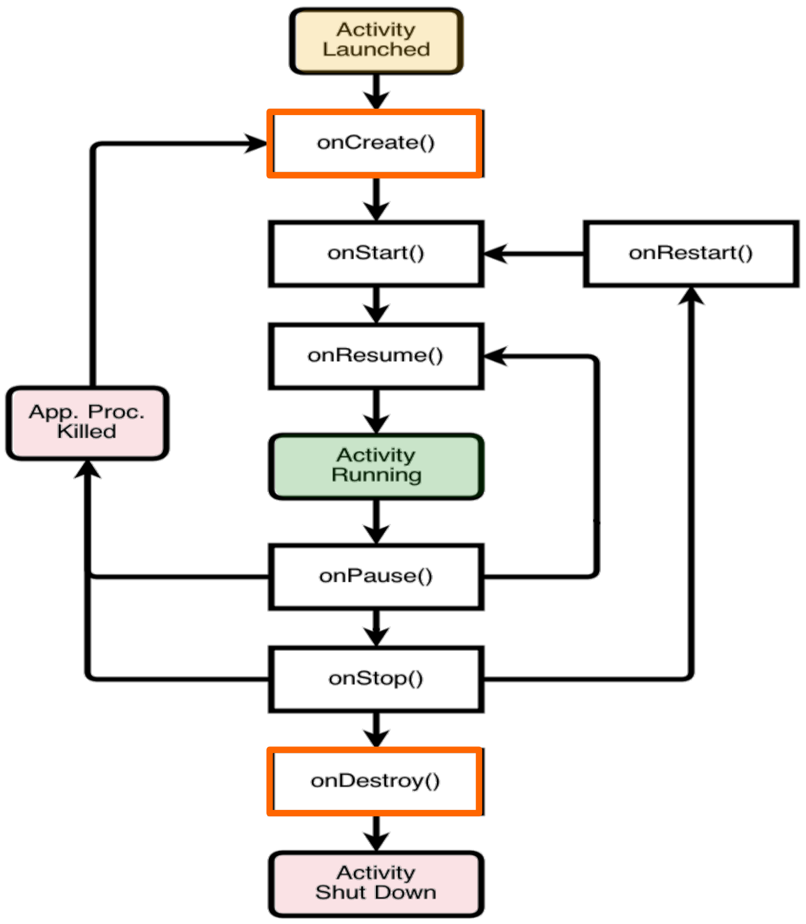
The Activity Lifecycle

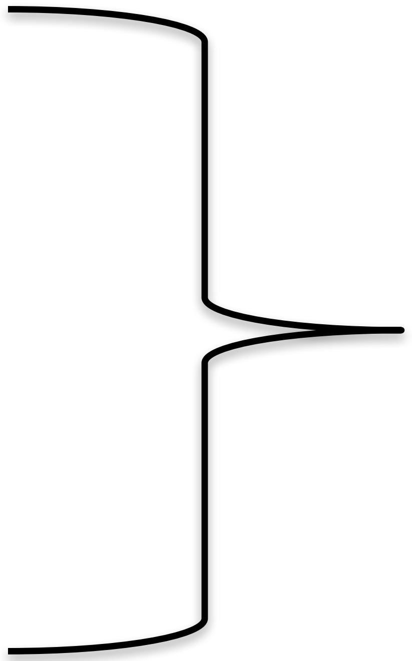


The Activity Lifecycle



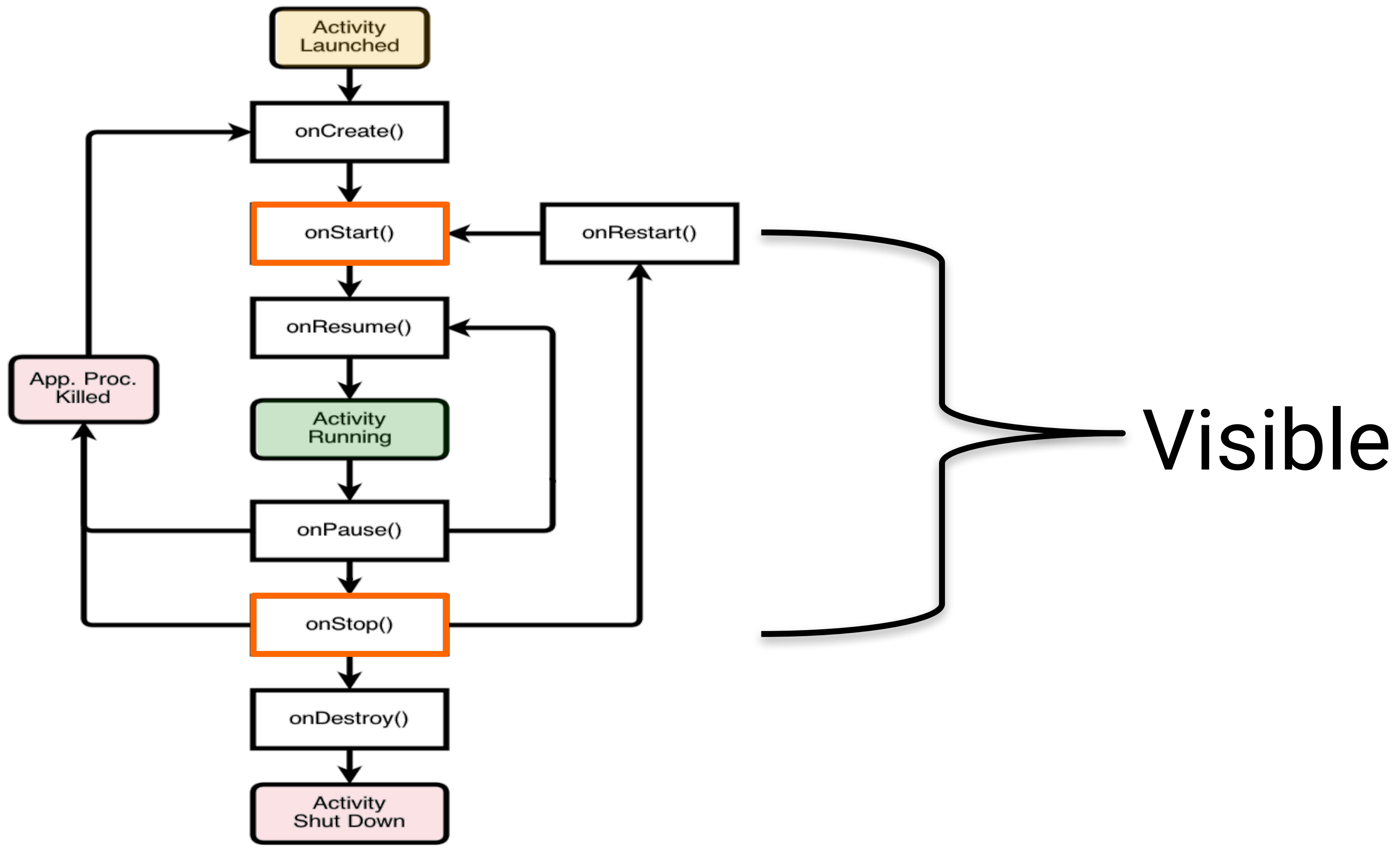
The Activity Lifecycle



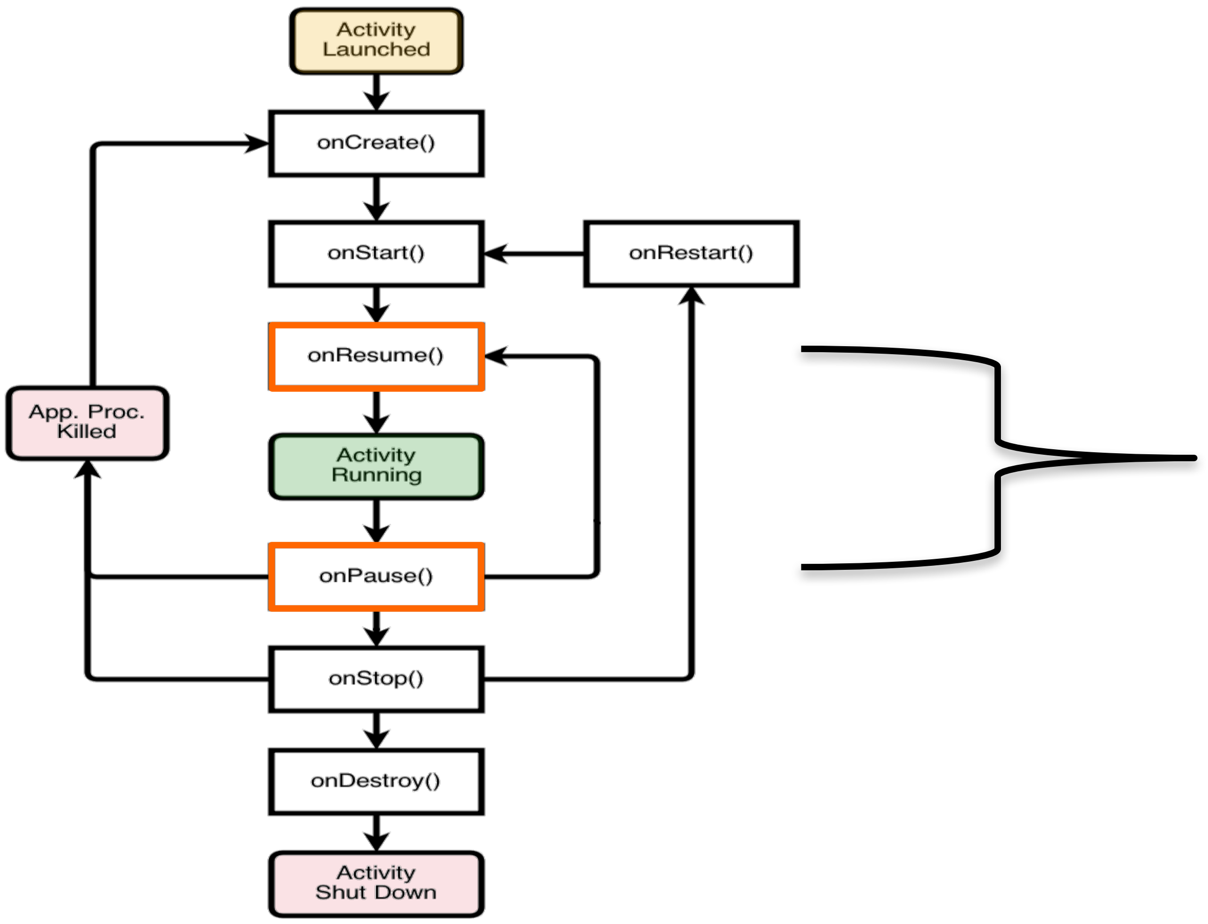


Entire Lifetime

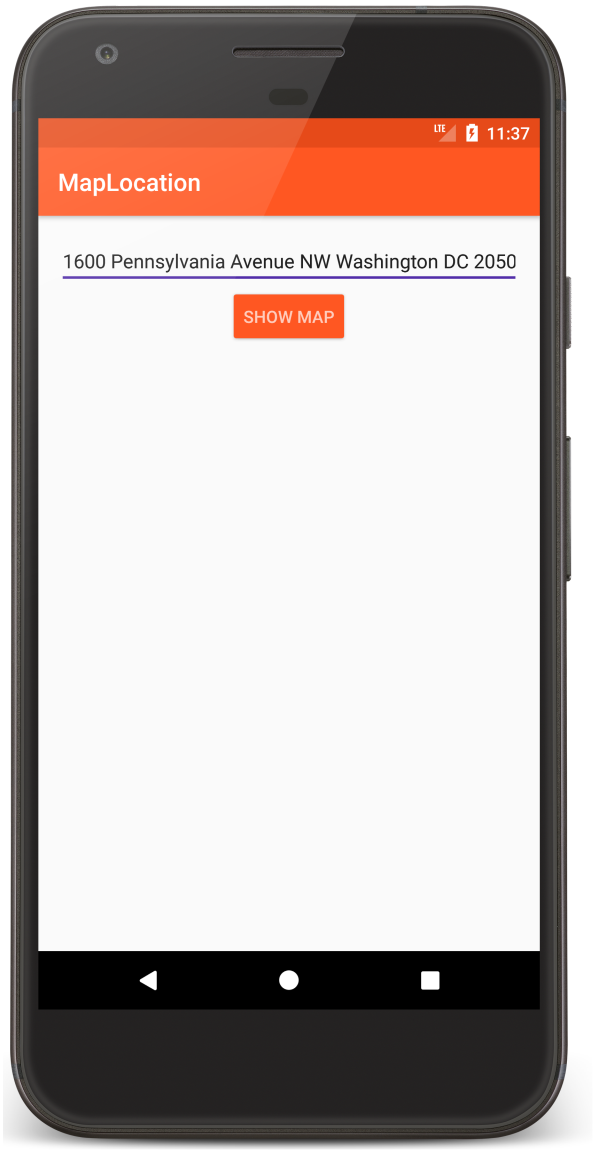
The Activity Lifecycle

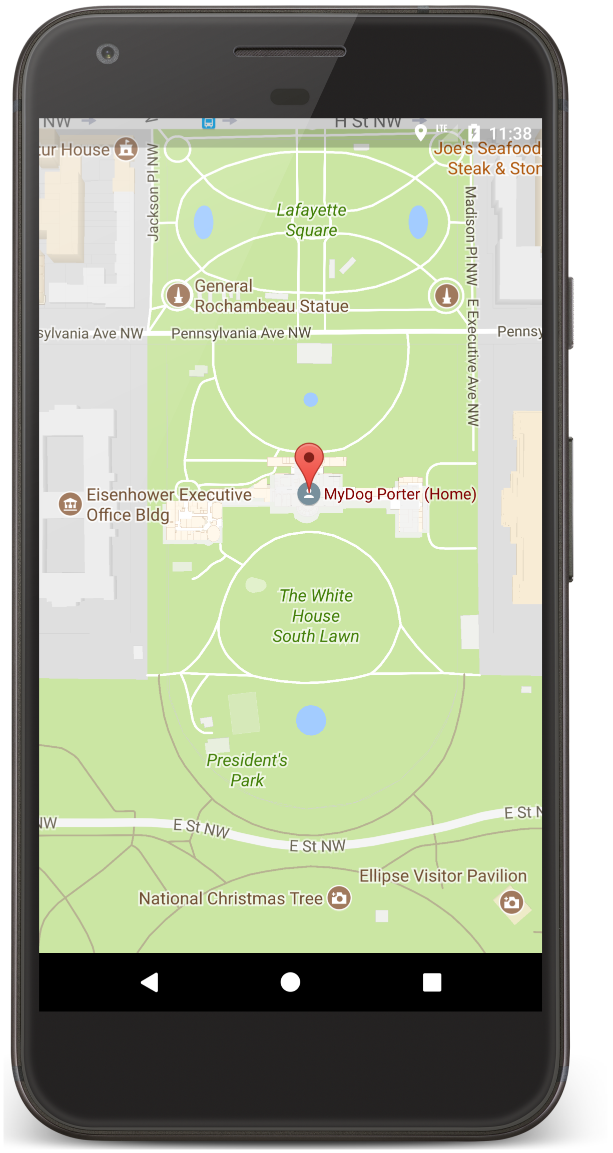


The Activity Lifecycle

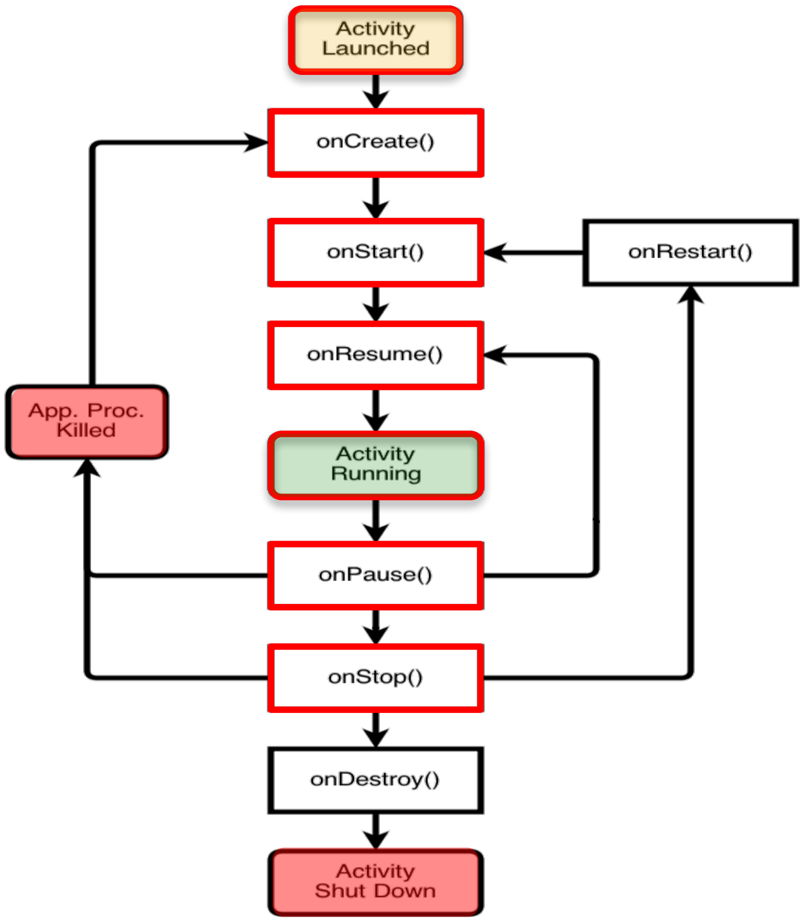


Visible & in Foreground

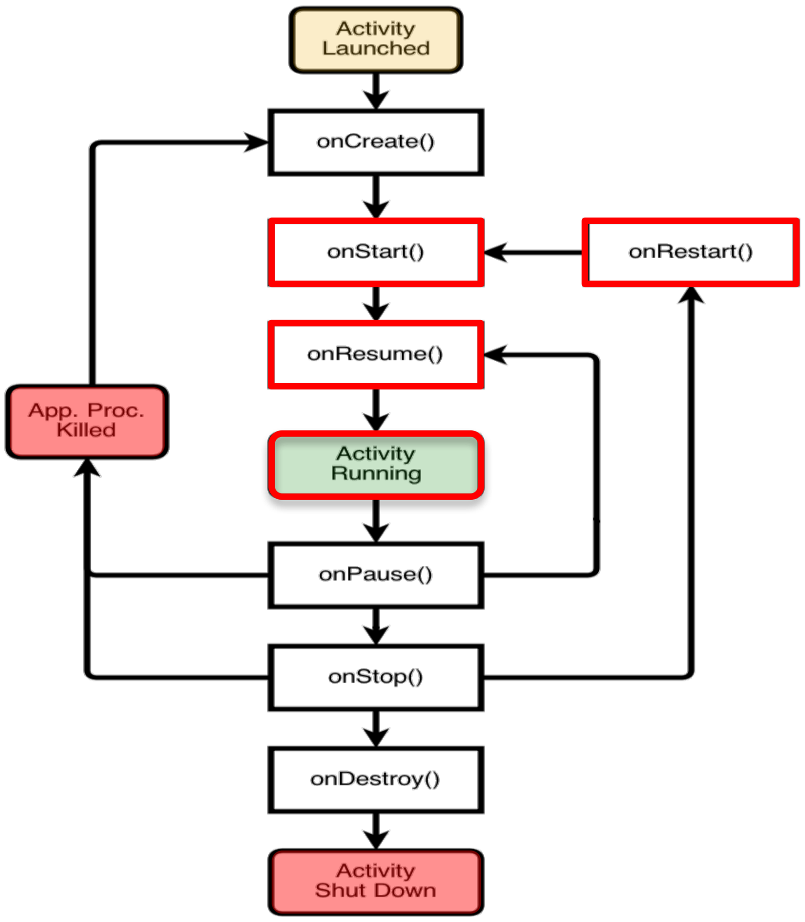

11:37
MapLocation
1600 Pennsylvania Avenue NW Washington DC 2050
SHOW MAP


11:38
H St NW
Joe's Seafood Steak & Ston
Lafayette Square
General Rochambeau Statue
Jackson Pl NW
Madison Pl NW
Pennsylvania Ave NW
E Executive Ave NW
Eisenhower Executive Office Bldg
MyDog Porter (Home)
The White House South Lawn
President's Park
E St NW
Ellipse Visitor Pavilion
National Christmas Tree

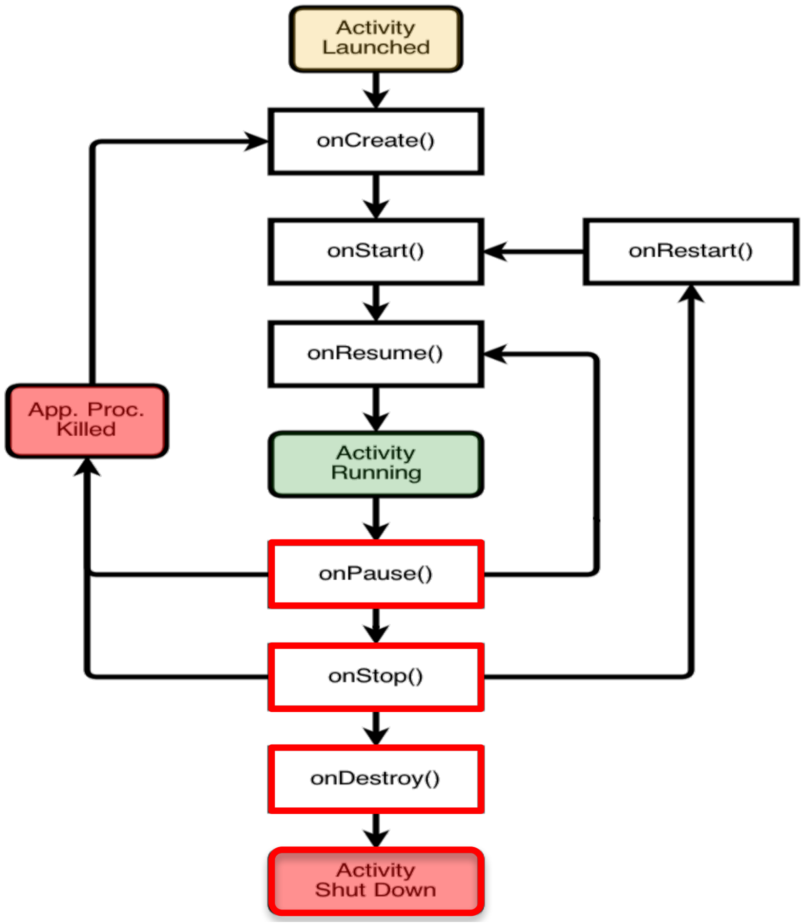
The Activity Lifecycle: MapLocation



The Activity Lifecycle: MapLocation



The Activity Lifecycle: MapLocation



onCreate()

- Called when Activity is created
- Sets up Initial state
  - Call super.onCreate()
  - Set the Activity’s content view
  - Retain references to UI views as necessary
  - Configure views as necessary

```java
package course.examples.maplocation;
…
public class MapLocation extends Activity {
   …
   @Override
   protected void onCreate(Bundle savedInstanceState) {

      // Required call through to Activity.onCreate()
      // Restore any saved instance state
      super.onCreate(savedInstanceState);

      // Set content view
      setContentView(R.layout.main);


      // Initialize UI elements
      final EditText addrText = findViewById(R.id.location);
      final Button button = findViewById(R.id.mapButton);

      …
```

```
…
// Link UI elements to actions in code
button.setOnClickListener(new OnClickListener() {
   // Called when user clicks the Show Map button
   public void onClick(View v) {
       try {
          // Process text for network transmission
         String address = addrText.getText().toString();
         address = address.replace(' ', '+');

          // Create Intent object for starting Google Maps application
         Intent geoIntent = new Intent(android.content.Intent.ACTION_VIEW,
                                               Uri .parse("geo:0,0?q=" + address));

          if (getPackageManager().resolveActivity(geoIntent, 0) != null) {
             // Use the Intent to start Google Maps application using Activity.startActivity()
            startActivity(geoIntent);
          }
   ...
```

onStart()

Activity is about to become visible

Typical actions

- Start when visible-only behaviors
- Loading persistent application state

onResume()

Activity is visible and about to start interacting with user

Typical actions

- Start foreground-only behaviors

onPause()

- Focus about to switch to another Activity
- Typical actions
    - Shutdown foreground-only behaviors

onStop()

- Activity is no longer visible to user
  - may be restarted later
- Typical actions
  - Save persistent state
  - Do CPU-intensive save procedures
- Note: Pre-Honeycomb - this method may not be called if Android kills your application

onRestart()

Called if the Activity has been stopped and is about to be started again

Typical actions

- Special processing needed only after having been stopped

onDestroy()

Activity is about to be destroyed

Typical actions

- Release Activity resources

Note: may not be called if Android kills your application

```java
@Override
protected void onStart() {
   super.onStart();
   Log.i(TAG, "The activity is visible and about to be started.");
}


@Override
protected void onRestart() {
   super.onRestart();
   Log.i(TAG, "The activity is visible and about to be restarted.");
}

@Override
protected void onResume() {
   super.onResume();
   Log.i(TAG, "The activity is and has focus (it is now \"resumed\")");
}
```

```
@Override
protected void onPause() {
   super.onPause();
   Log.i(TAG,
         "Another activity is taking focus (this activity is about to be \"paused\")");
}

@Override
protected void onStop() {
   super.onStop();
   Log.i(TAG, "The activity is no longer visible (it is now \"stopped\")");
}

@Override
protected void onDestroy() {
   super.onDestroy();
   Log.i(TAG, "The activity is about to be destroyed.");
}
```

Starting Activities

Create an Intent object matching the Activity to start

Starting Activities

Pass newly created Intent to methods, such as:

startActivity()

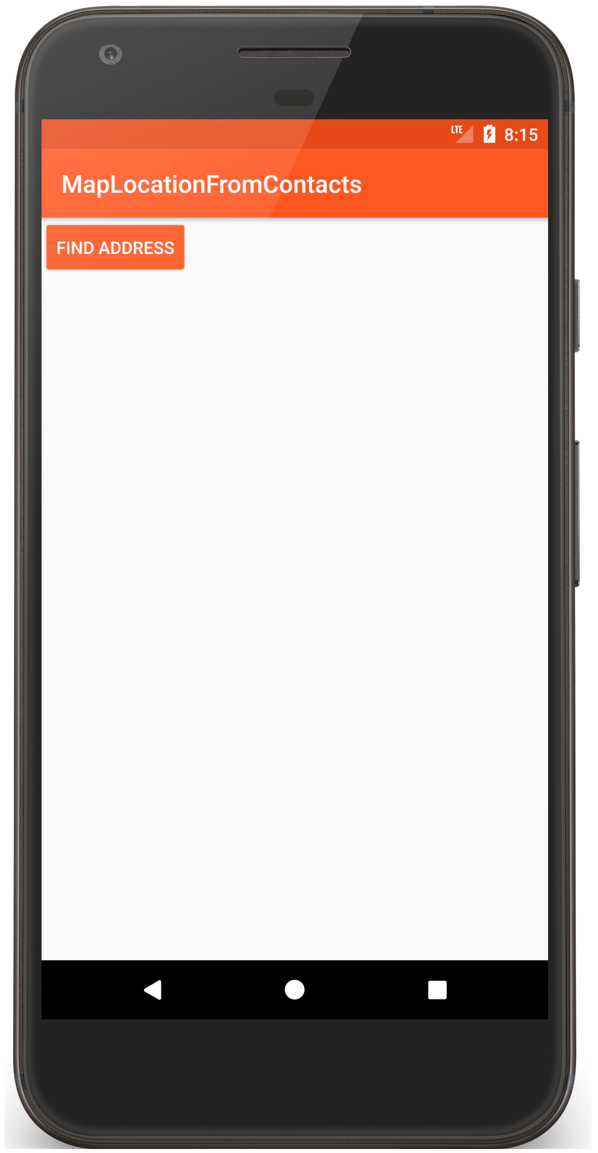
startActivityForResult()

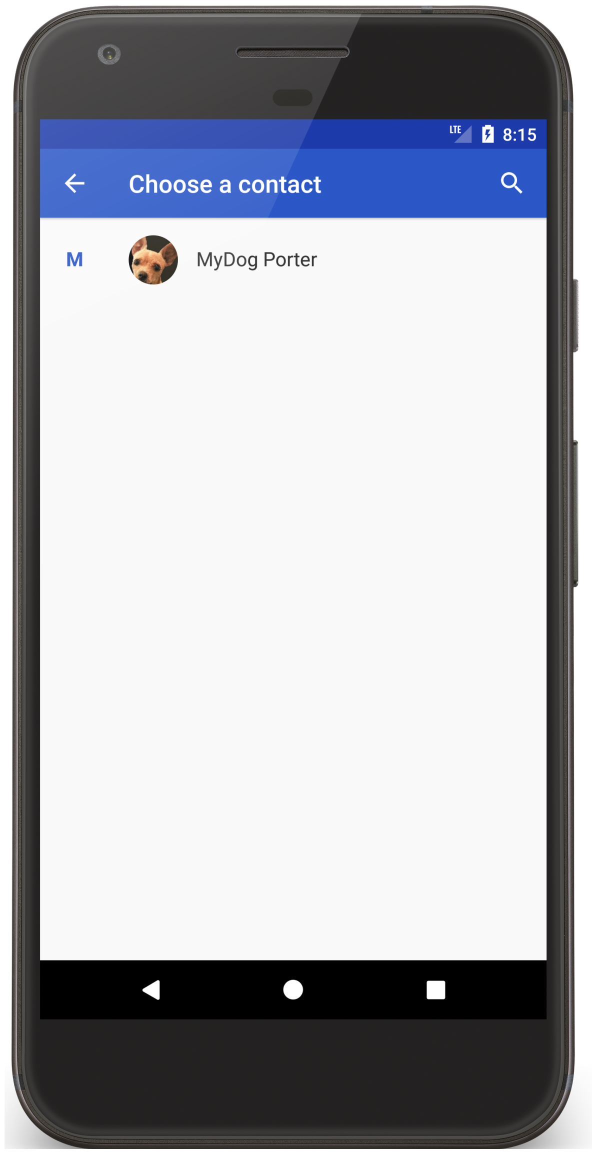
- Invokes a callback method when the called Activity finishes to return a result to the calling Activity

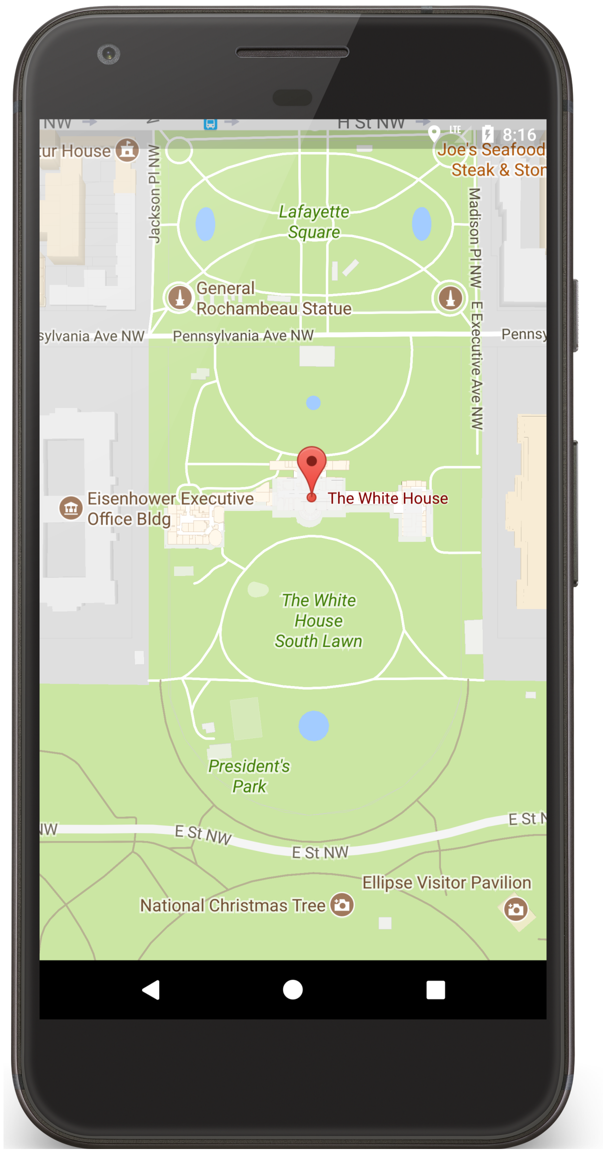
```
…
// Link UI elements to actions in code
button.setOnClickListener(new OnClickListener() {
   // Called when user clicks the Show Map button
   public void onClick(View v) {
      try {
         // Process text for network transmission
         String address = addrText.getText().toString();
         address = address.replace(' ', '+');

         // Create Intent object for starting Google Maps application
         Intent geoIntent = new Intent(android.content.Intent.ACTION_VIEW,
                                                  Uri .parse("geo:0,0?q=" + address));

         if (getPackageManager().resolveActivity(geoIntent, 0) != null) {
            // Use the Intent to start Google Maps application using Activity.startActivity()
           startActivity(geoIntent);
         }
   …
```

MapLocationFromContacts

Similar to MapLocation, but gets address from contacts database


8:15
MapLocationFromContacts
FIND ADDRESS


8:15
Choose a contact
M
MyDog Porter


8:16
H St NW
Joe's Seafood Steak & Ston
Jackson Pl NW
Lafayette Square
Madison Pl NW
General Rochambeau Statue
Pennsylvania Ave NW
E Executive Ave NW
Eisenhower Executive Office Bldg
The White House
The White House South Lawn
President's Park
E St NW
Ellipse Visitor Pavilion
National Christmas Tree

```
private void startContactsApp() {

   // Create Intent object for picking data from Contacts database
   Intent intent = new Intent(Intent.ACTION_PICK, CONTACTS_CONTENT_URI);

   if (getPackageManager().resolveActivity(intent,0) != null) {
      // Use Intent to start Contacts application
      // Variable PICK_CONTACT_REQUEST identifies this operation
      startActivityForResult(intent, PICK_CONTACT_REQUEST);
   }
}
```

Activity.setResult()

Started Activity can set its result by calling Activity.setResult()

public final void setResult (int resultCode)

public final void setResult (int resultCode, Intent data)

Activity.setResult()

- resultCode (an int)
    - RESULT_CANCELED
    - RESULT_OK
    - RESULT_FIRST_USER
        - Custom resultCodes can be added

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

   // Ensure that this call is the result of a successful PICK_CONTACT_REQUEST request
   if (resultCode == Activity.RESULT_OK
         && requestCode == PICK_CONTACT_REQUEST) {

      // get address from selected contact ….
            if (null != formattedAddress) {
              …
              // Create Intent object for starting Google Maps application
             Intent geoIntent = new Intent(android.content.Intent.ACTION_VIEW,
                                                    Uri.parse("geo:0,0?q=" + formattedAddress));

              // Use the Intent to start Google Maps application using Activity.startActivity()
             startActivity(geoIntent);
            }
   …
```

Configuration Changes

Keyboard, orientation, locale, etc.

Device configuration can change at runtime

On configuration changes, Android usually kills the current Activity & then restarts it

Configuration Changes

Activity restarting should be fast

Options

- Save Activity state in Bundle
- Retain an separate Object
- Manually handle the configuration change

Saving Activity State

System saves some information such as View state in a Bundle

You must save other state yourself

Saving Activity State

Android calls onSaveInstanceState(Bundle) after onPause() and before onStop()

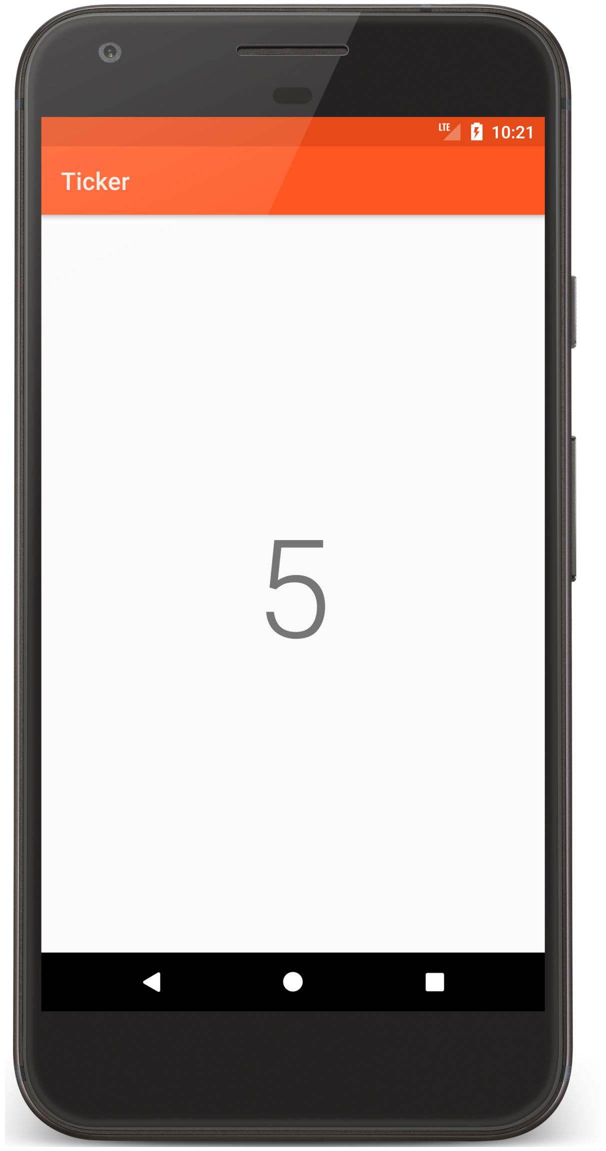
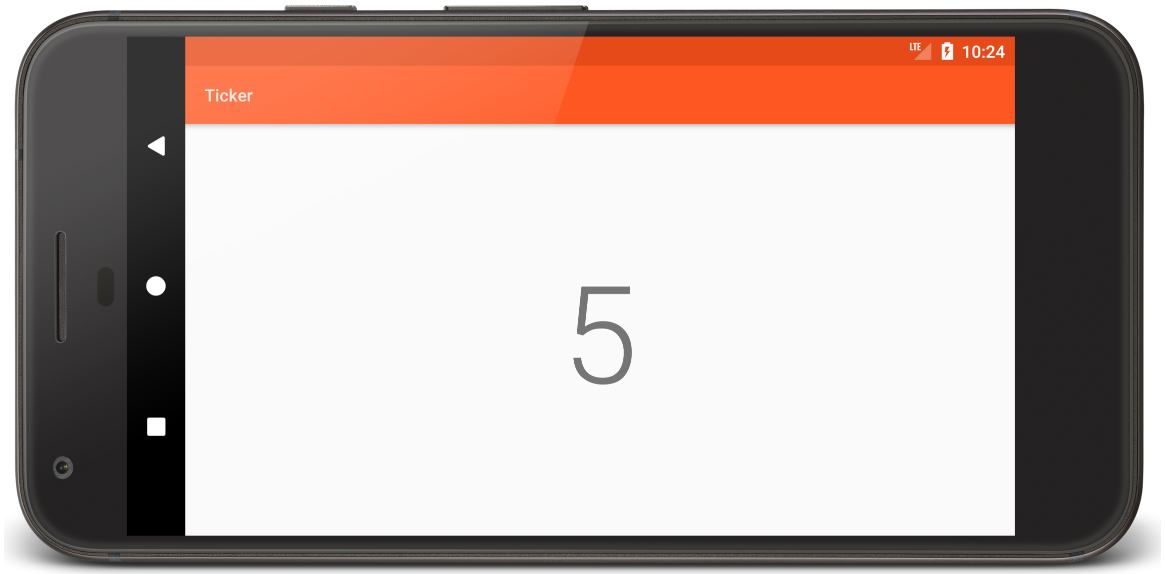
Save Activity instance state to system-provided Bundle

Saving Activity State

When Activity is restarted, you can restore this state from a system-provided Bundle

In onCreate(Bundle)

In onRestoreInstanceState(Bundle), which is called after onStart()


10:21
Ticker
5

10:24
Ticker
5

```
public class TickerDisplayActivity extends Activity {
   private static final String COUNTER_KEY = "COUNTER_KEY";
   private int mCounter = 0;
…
   @Override
   protected void onCreate(Bundle savedInstanceState) {
…
      // Comment out this step and the counter will reset on restarts
      if (null != savedInstanceState) {
         mCounter = savedInstanceState.getInt(COUNTER_KEY);
      }
      // Runnable that updates the counter once every second
      update = new Runnable() {
         @Override
         public void run() {
            mCounterView.setText(String.valueOf(++mCounter));
            mHandler.postDelayed(this, delay);
         }
      };
   }
```

```
// Save instance state
@Override
public void onSaveInstanceState(Bundle bundle) {

   // Save mCounter value
   bundle.putInt(COUNTER_KEY, mCounter);

   // call superclass to save any view hierarchy
   super.onSaveInstanceState(bundle);
}
```

Retaining an Object

Hard to recompute data can be cached to speed up handling of configuration changes

Current recommendation is to store state in a Fragment

We’ll come back to this in a later lesson

Manual Reconfiguration

Can prevent system from restarting Activity

Declare the configuration changes your Activity handles in AndroidManifest.xml file, e.g.,

```
<activity android:name=".MyActivity“
    android:configChanges=
        "orientation|screensize|keyboardHidden”…>
```

Manual Reconfiguration

When configuration changes,

Activity’ s onConfigurationChanged() method is called

Passed a Configuration object specifying the new device configuration

Manual Reconfiguration Caveat

Should generally avoid manual approach

- Hard to get right
- Fragile to system changes

Next

The Intent Class