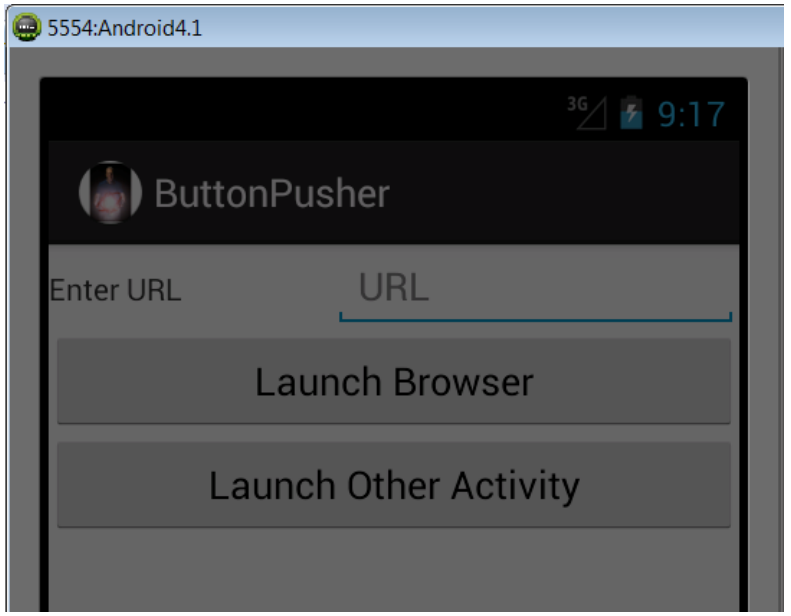
Activity

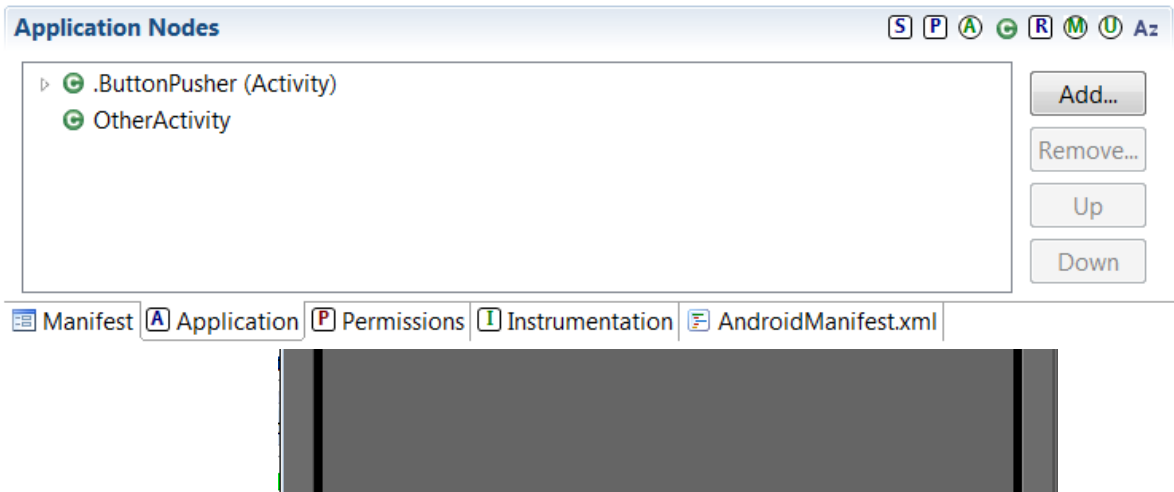
Programming the Android Platform

CS 282
Principles of Operating Systems II
Systems Programming for Android

Activity

- Provides a visual interface for user interaction
- Typically supports one thing a user can do
  - View an email message
  - Show a login screen
- Applications can include several activities



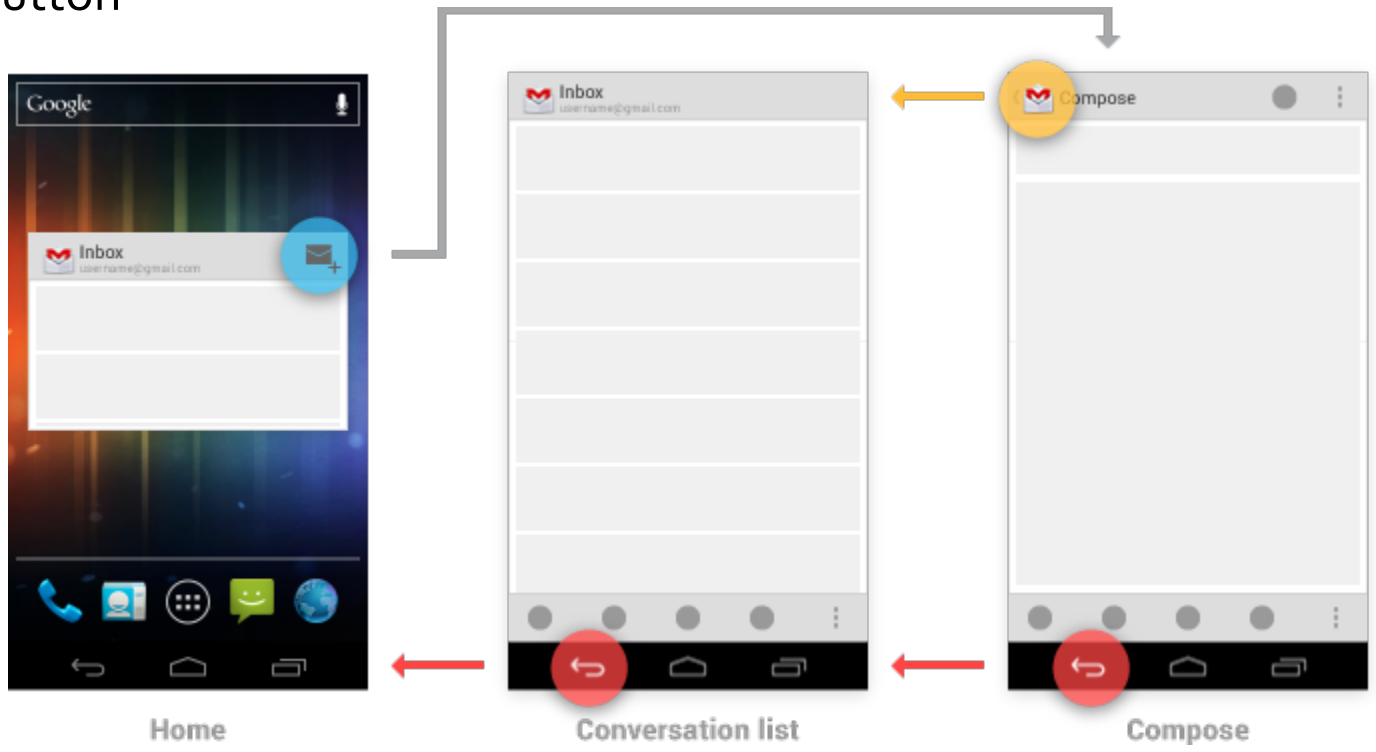


Tasks

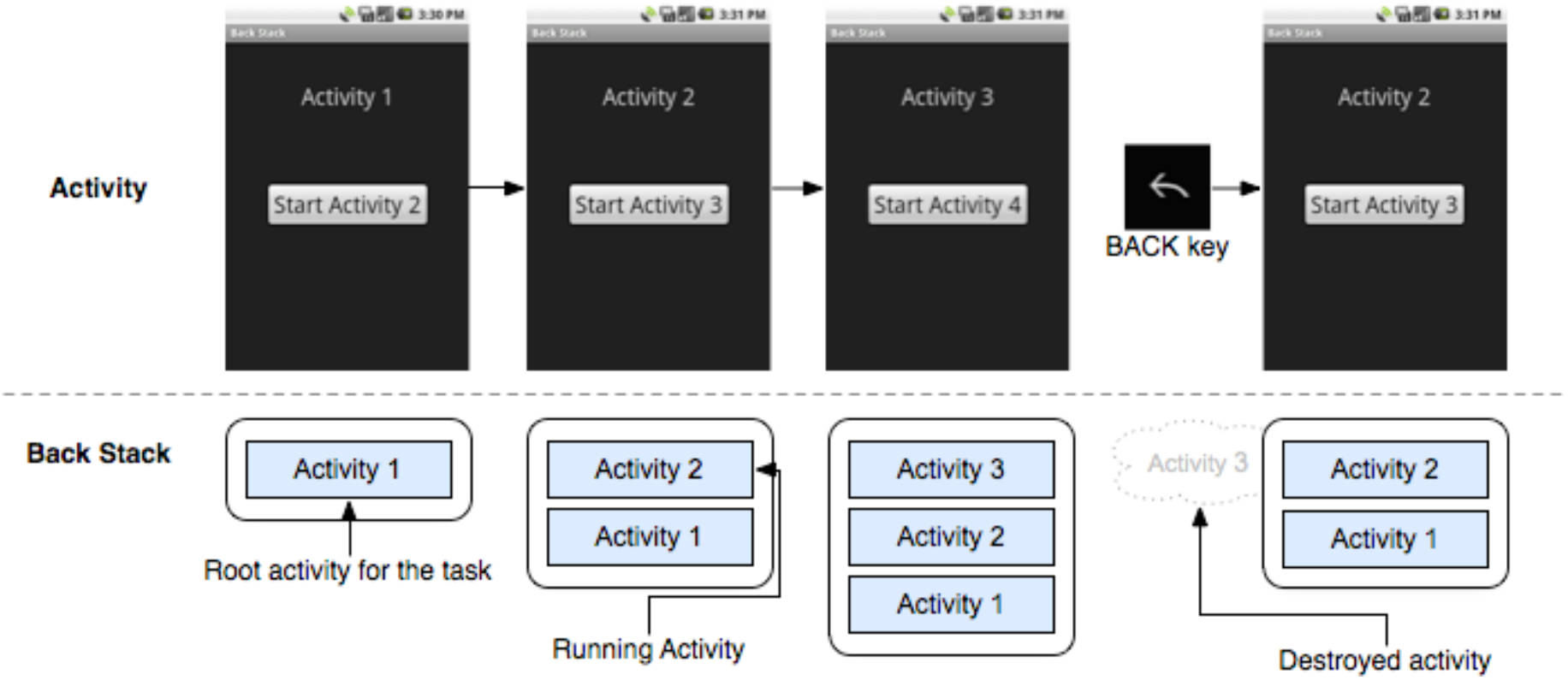
- A Task is a chain of related Activities
  - Task not necessarily provided by a single application
- Gives the illusion that multiple, unrelated Activities were developed as part of the same application

Tasks

- The task's Activity objects are stored on a "back stack" with the currently running Activity at the top
- At runtime
  - Launching an Activity places it on top of the stack
  - Hitting BACK button pops current activity off the stack



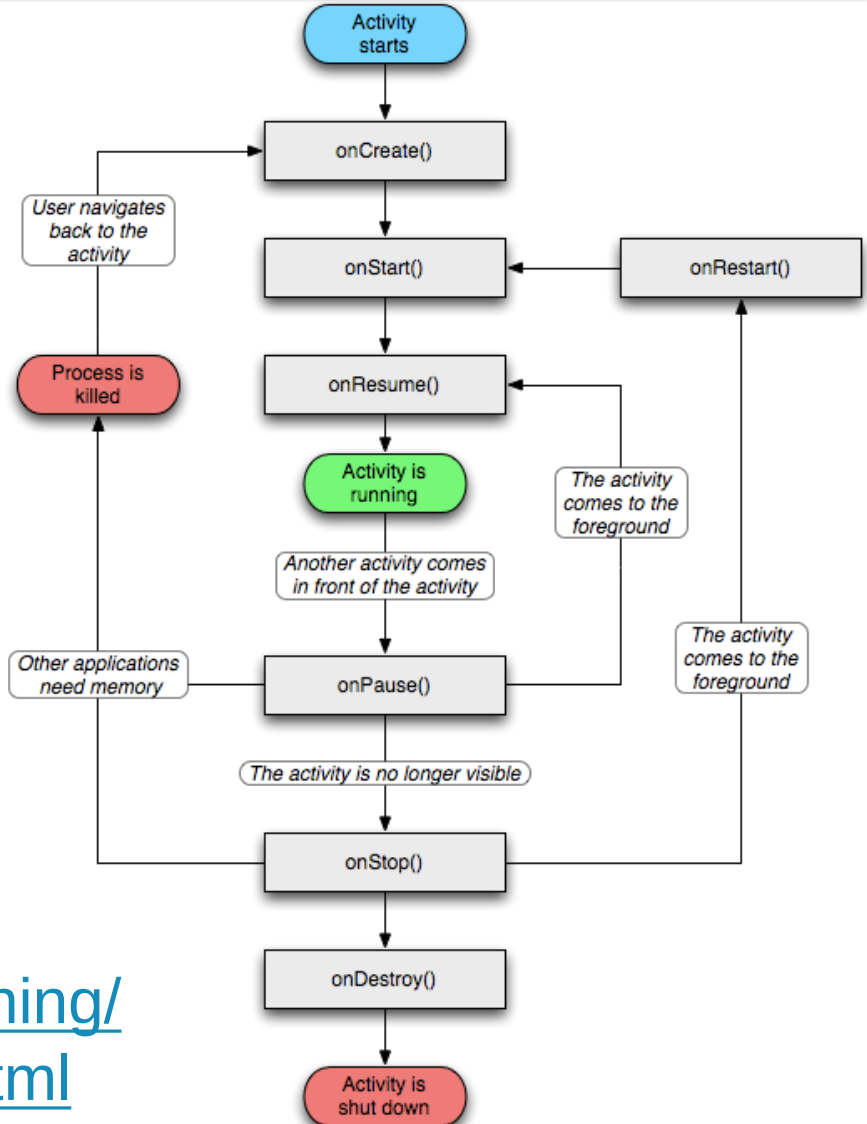
Task Stack



http://developer.android.com/guide/topics/fundamentals/tasks-and-back-stack.html

Activity States

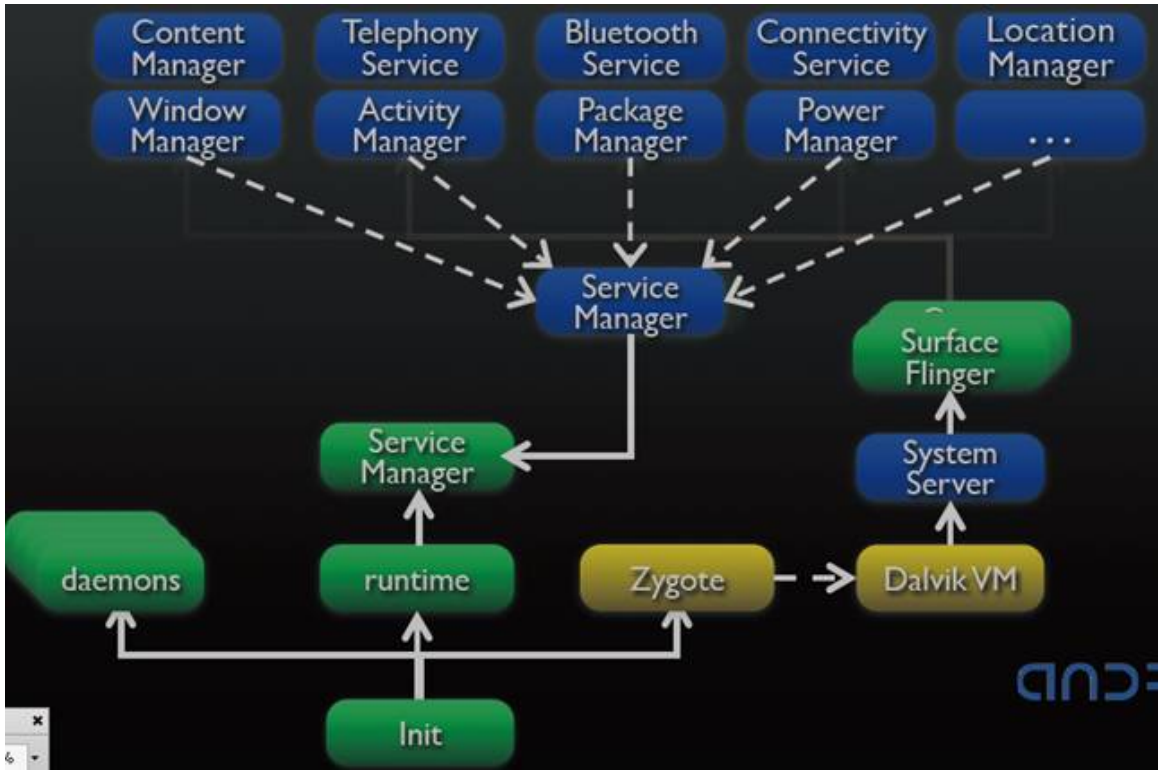
- Not started – not yet created
- Active
  - Resumed/Running - visible, has focus
  - Paused - visible, does not have focus, can be terminated
  - Stopped - not visible, does not have focus, can be terminated
- Finished – done



http://developer.android.com/training/basics/activity-lifecycle/index.html

The Activity Lifecycle

- Android communicates state changes to application by calling specific lifecycle methods
- The ActivityManager is the system service in Android that communicates these changes



http://developer.android.com/reference/android/app/ActivityManager.html

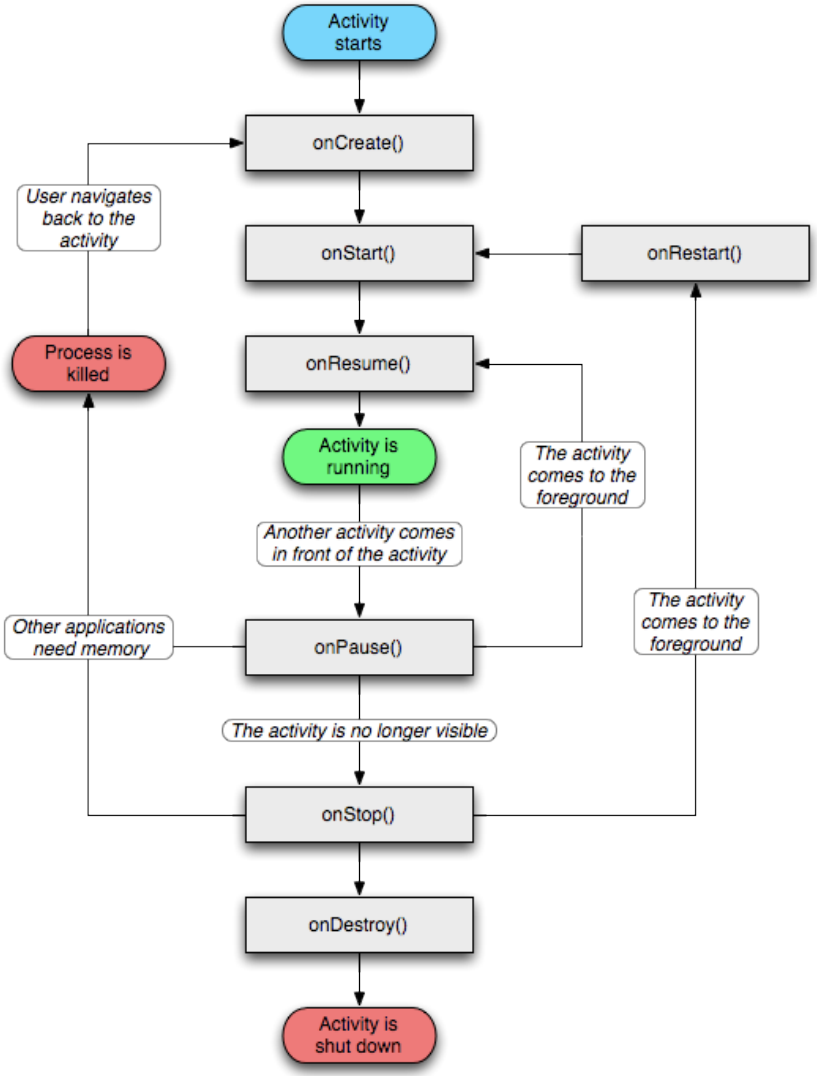
Activity Lifecycle Methods

- protected void onCreate()
- protected void onStart()
- protected void onResume()
- protected void onPause()
- protected void onRestart()
- protected void onStop()
- protected void onDestroy()

http://developer.android.com/reference/android/app/Activity.html

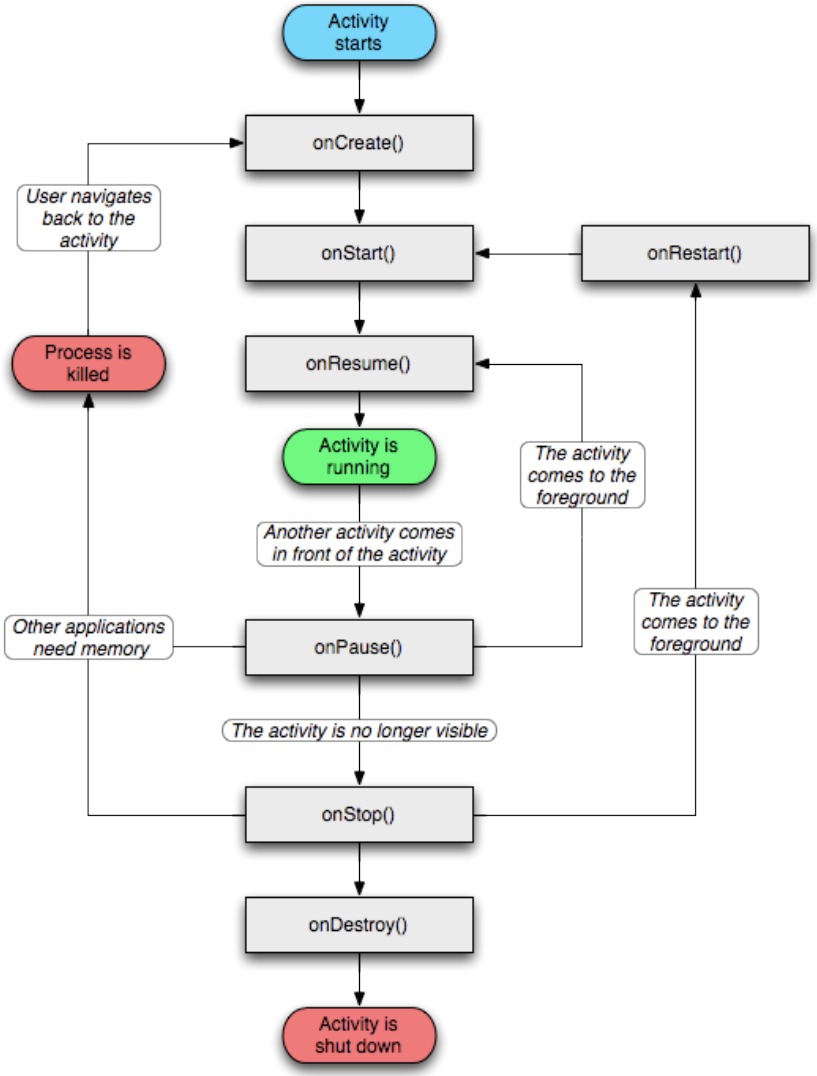
Activity Lifecycle Methods

- An Activity has several important methods that are called by the Android runtime to control its life-cycle:
  - **onCreate()** – this method is called when the Activity is first created. You will almost always override this method & provide setup code in this method
  - **onStop()** – this method is called when the user leaves your Activity for another Activity (your Activity is not visible)
  - **onPause()** – the user leaves your Activity but it is still visible in the background (e.g. transparent or partial foreground coverage)

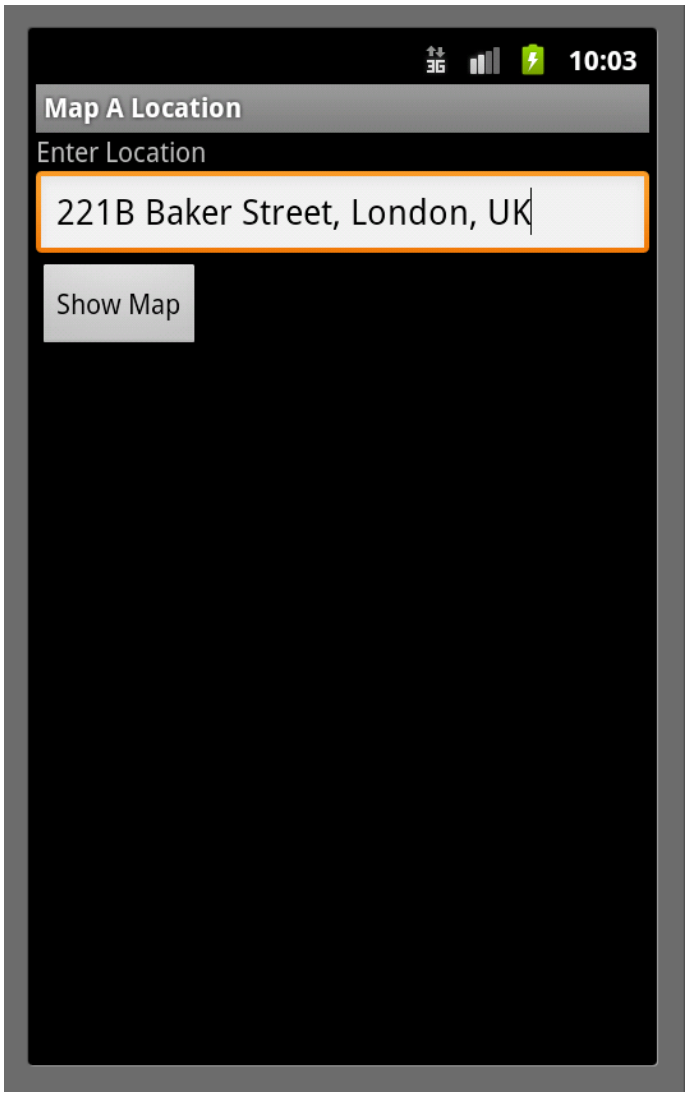


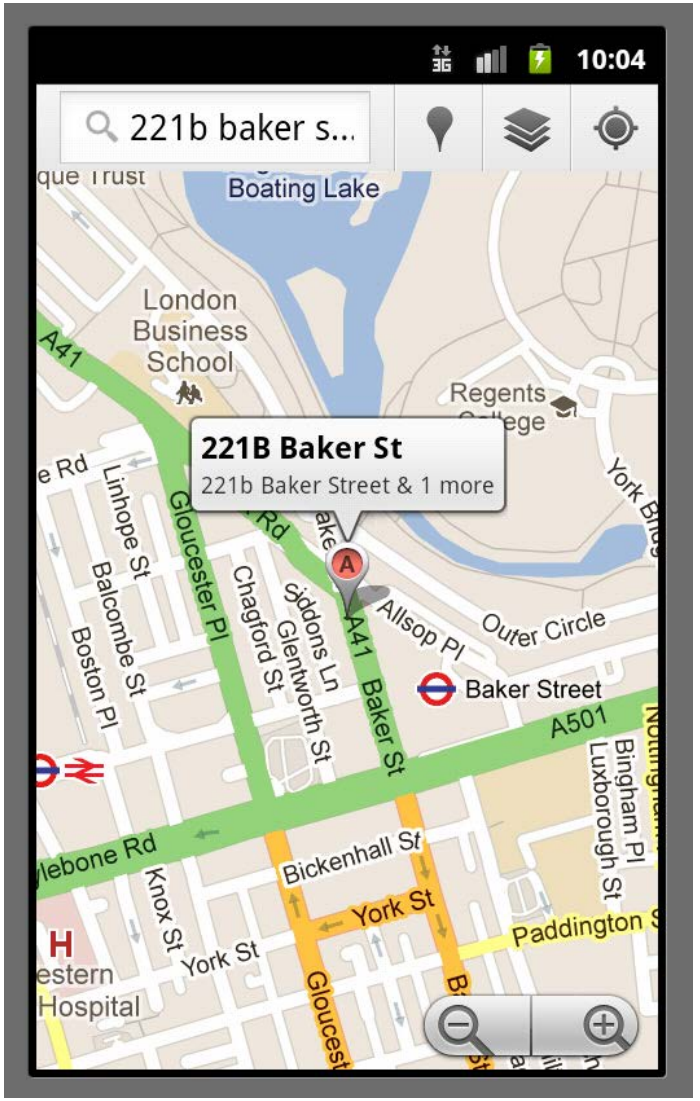
Activity Lifecycle Methods

- An Activity has several important methods that are called by the Android runtime to control its life-cycle:
  - **onResume()** – this method is called when the user returns to your Activity from another Activity
  - **onStart()** – this method is called after your Activity is created or stopped
  - **onDestroy()** – the Activity is being released & needs to clean up all resources



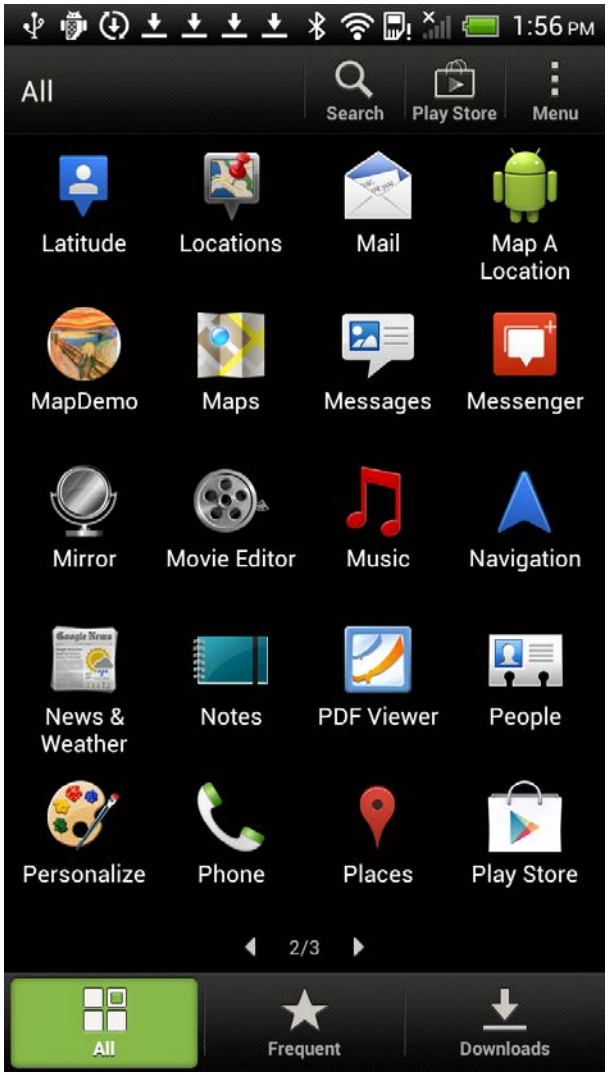
MapLocation App Example





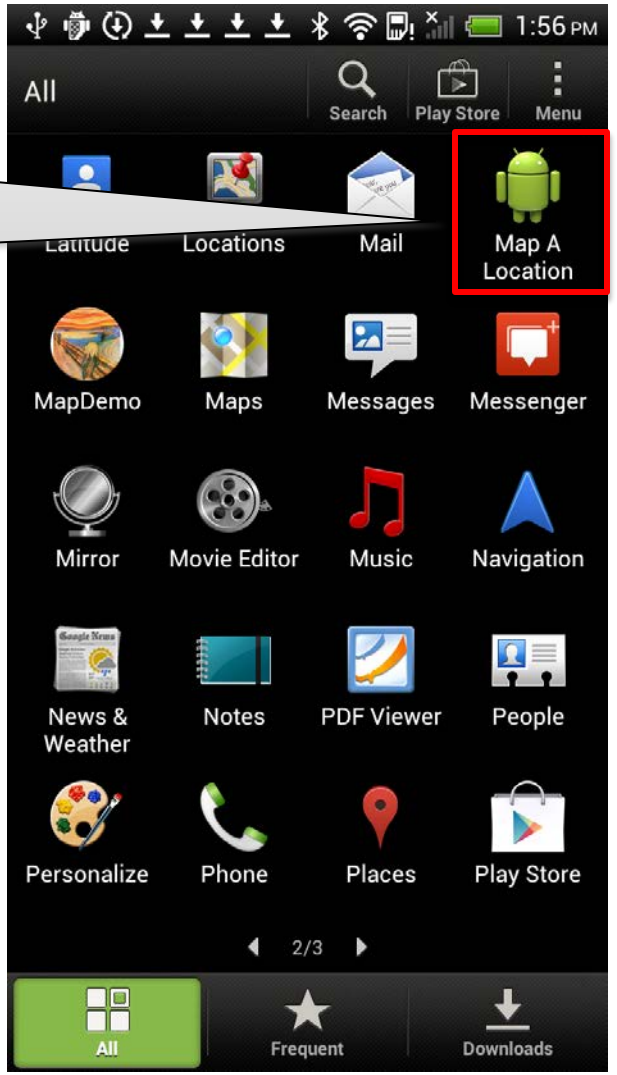
Calling onCreate() in Map App

- Called when Activity is first being created
- Setup global state
  - Call super.onCreate()
  - Inflate UI views
  - Configure views as necessary
  - Set the Activity's content view



Calling onCreate() in Map App

When the main Activity for your app shows on screen the onStart() method is called

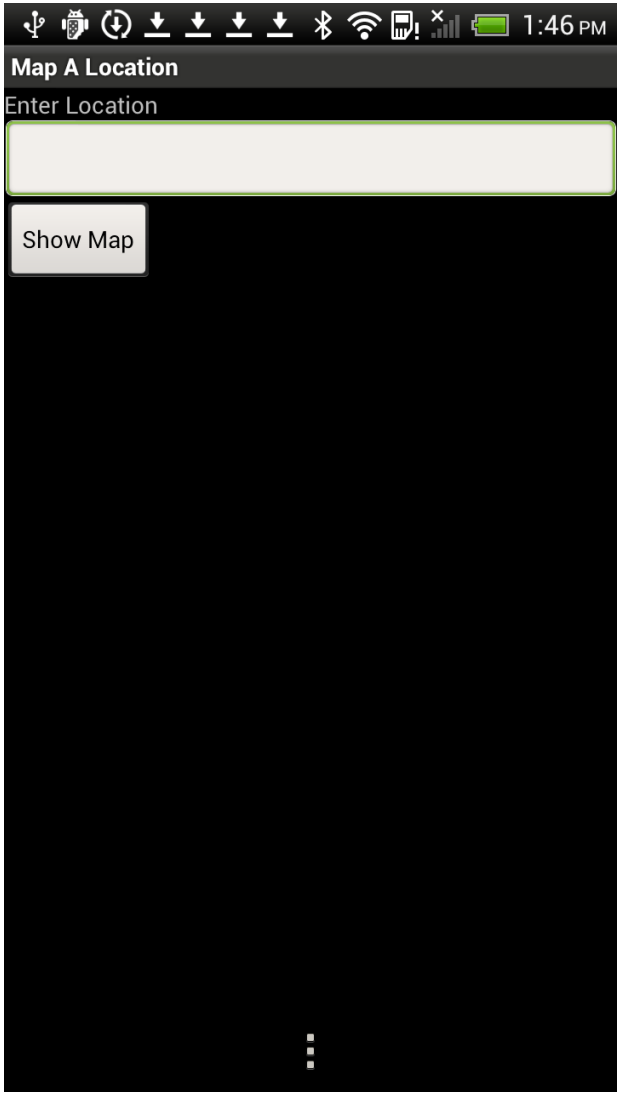


MapLocation.onCreate()

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    final EditText addressfield = (EditText) findViewById(R.id.location);
    final Button button = (Button) findViewById(R.id.mapButton);
    button.setOnClickListener(new Button.OnClickListener() {
        public void onClick(View v) {
            try {
              String address = addressfield.getText().toString();
              address = address.replace(' ', '+');
              Intent geoIntent = new Intent(android.content.Intent.ACTION_VIEW,
                                            Uri.parse("geo:0,0?q=" + address));
              startActivity(geoIntent);
             } catch (Exception e) {}
        }
      });
  }
```

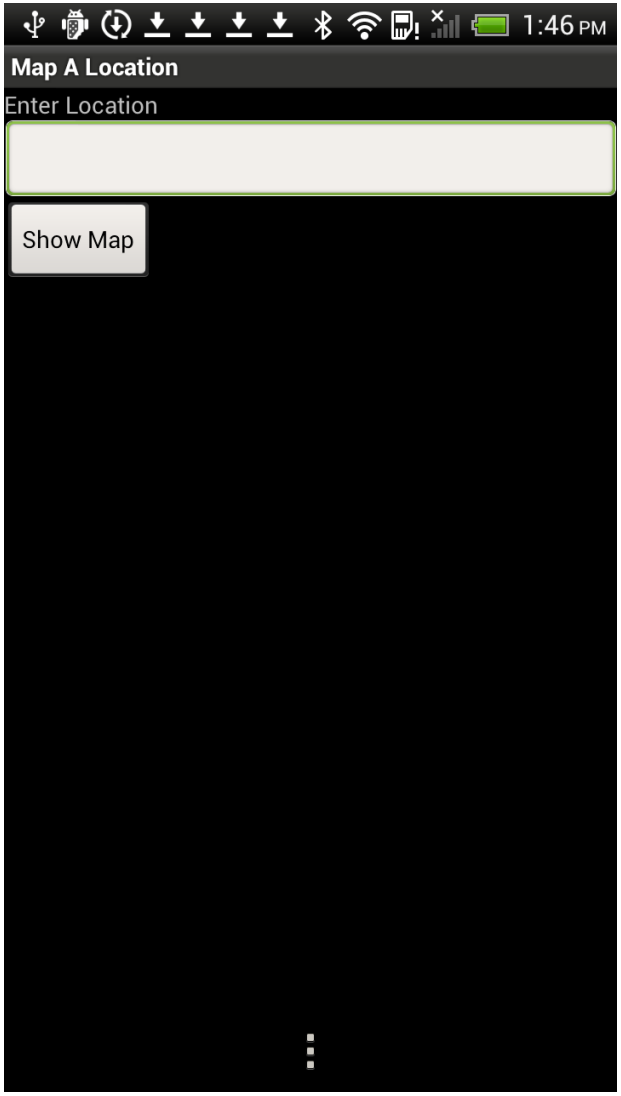
Calling onStart() in Map App

- Activity is about to become visible
- Typical actions
  - Reset application



Calling onResume() in Map App

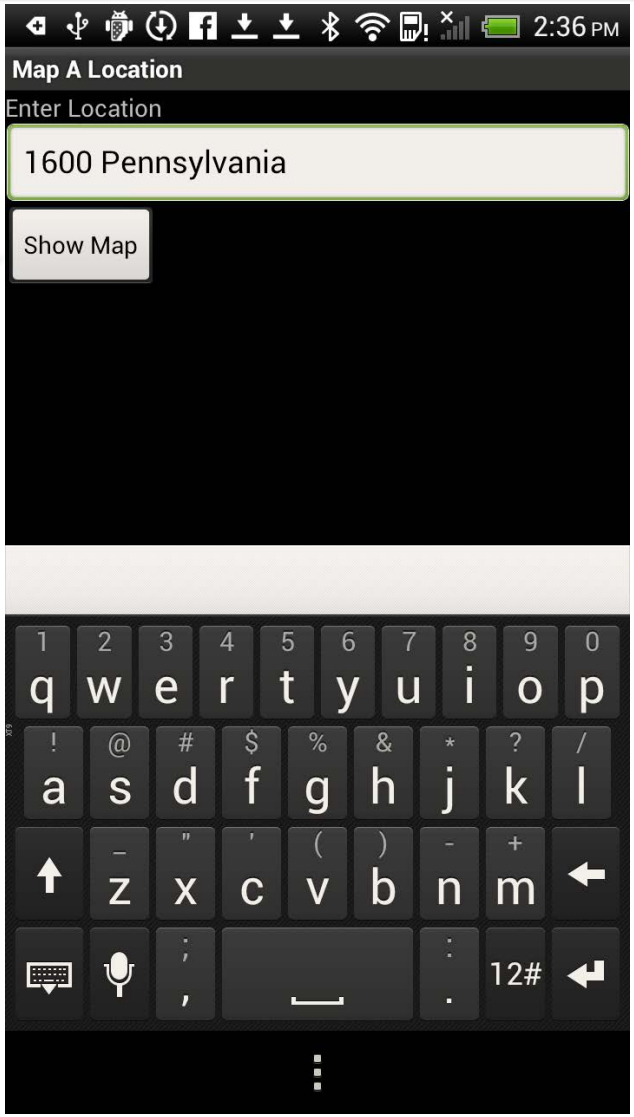
- About to start interacting with user
- Typical actions
  - Start foreground-only behaviors



Entering Text & Launch Map Activity

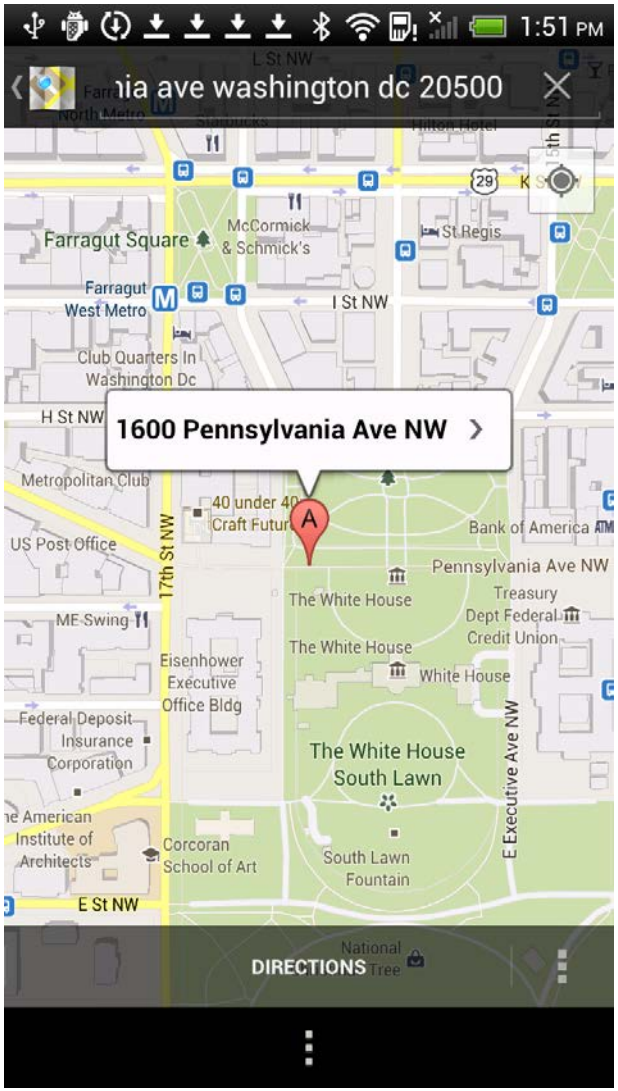
- Note that entering text via the virtual keyboard doesn't change the focus on the UI nor does it generate any lifecycle events

Clicking on the "Show Map" button will open a new Activity to display the map



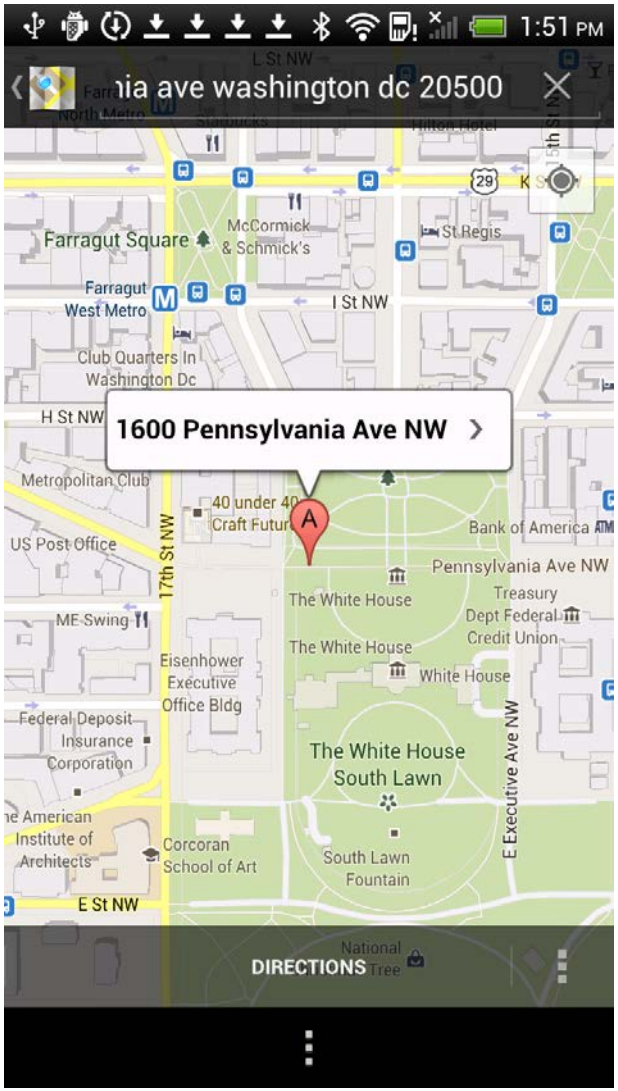
Calling onPause() in Map App

- Focus about to switch to another Activity
  - Could also be a "toast"
- Typical actions
  - Shutdown foreground-only behaviors



Calling onStop() in Map App

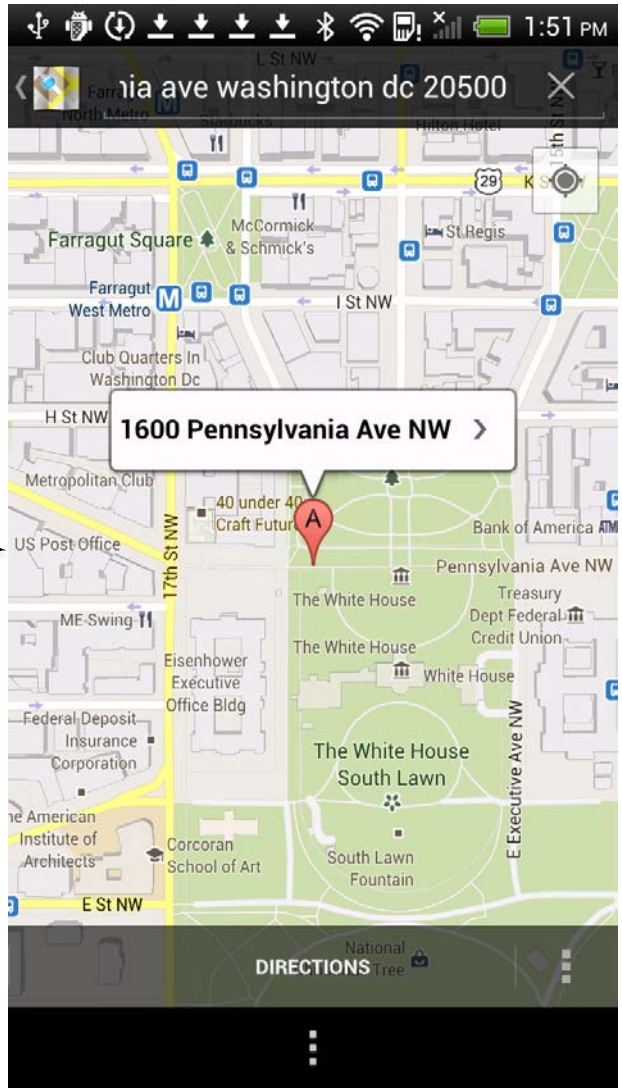
- Activity is no longer visible to user
  - But may be restarted later
- Typical actions
  - Cache state



onPause()/onStop() in Map App

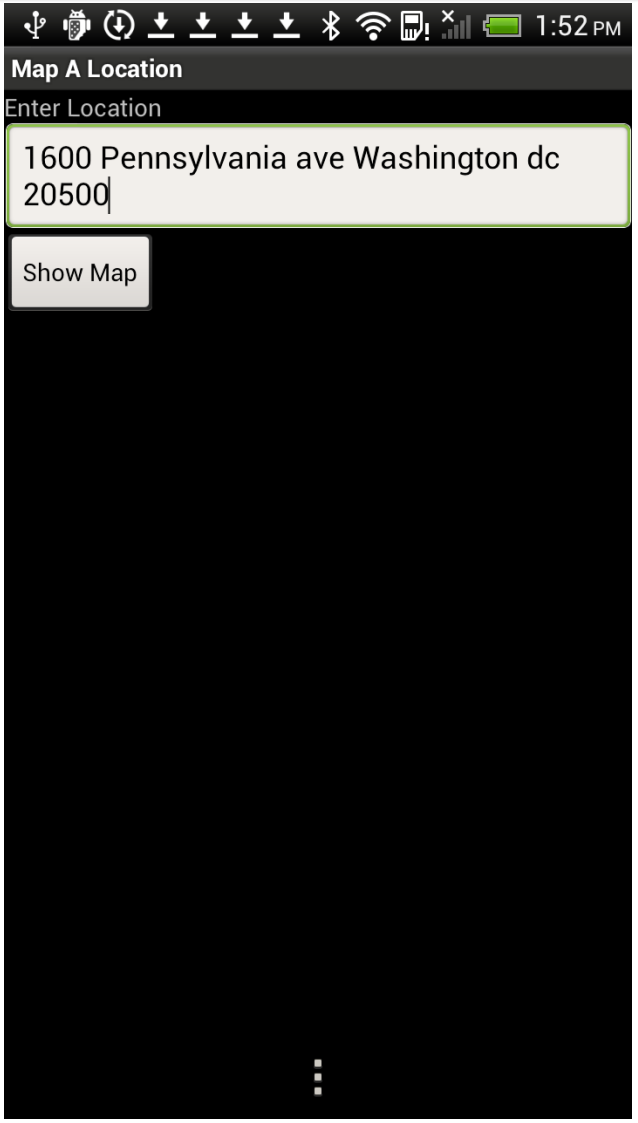
When the google map Activity is launched, its onCreate() & onStart() methods are called

The prior Activity's onPause() & onStop() methods are called



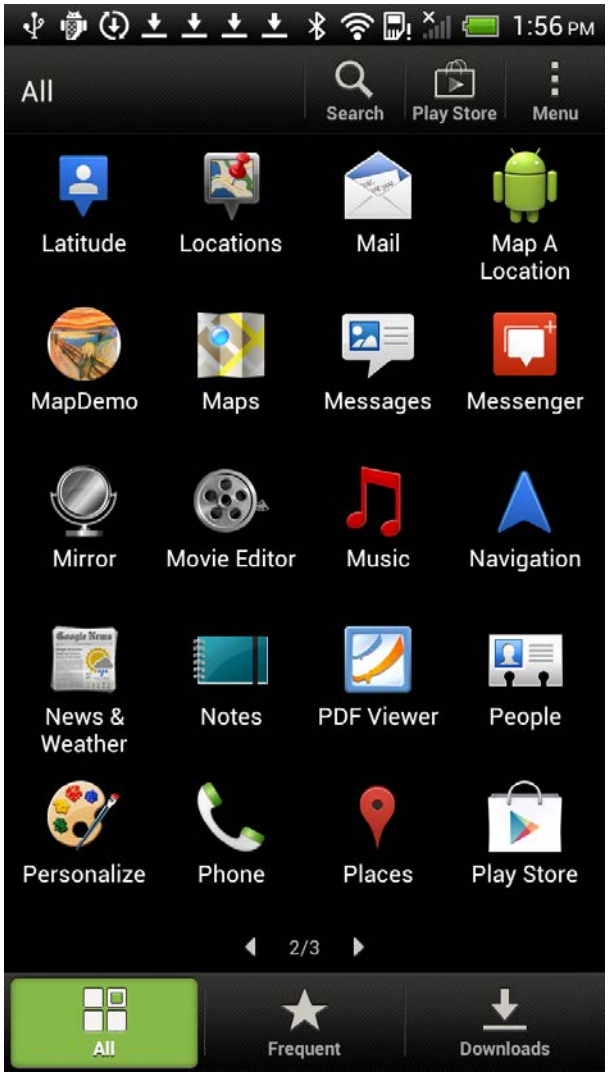
Calling onRestart() in Map App

- Called if the Activity has been stopped & is about to be started again
  - e.g., returning back to a previously launched Activity
- Typical actions
  - Read cached state



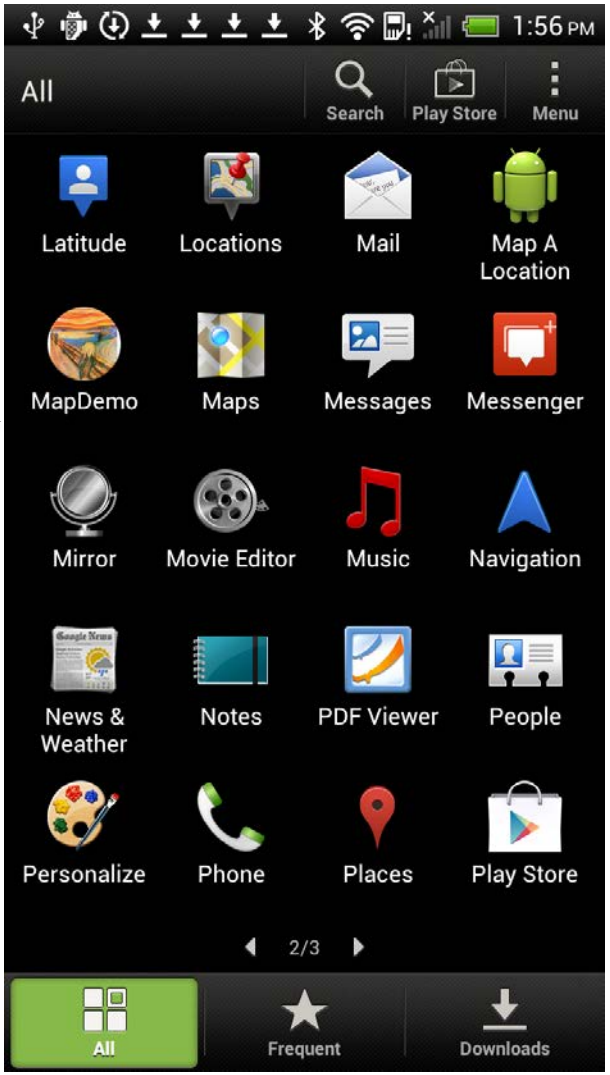
Calling onDestroy() in the Map App

- Activity is about to be destroyed
  - e.g., when the user presses the "back" button
- Typical actions
  - Save persistent state



Calling onDestroy() in the Map App

When the user completely exits the app, the original default Activity's onDestroy() method is called



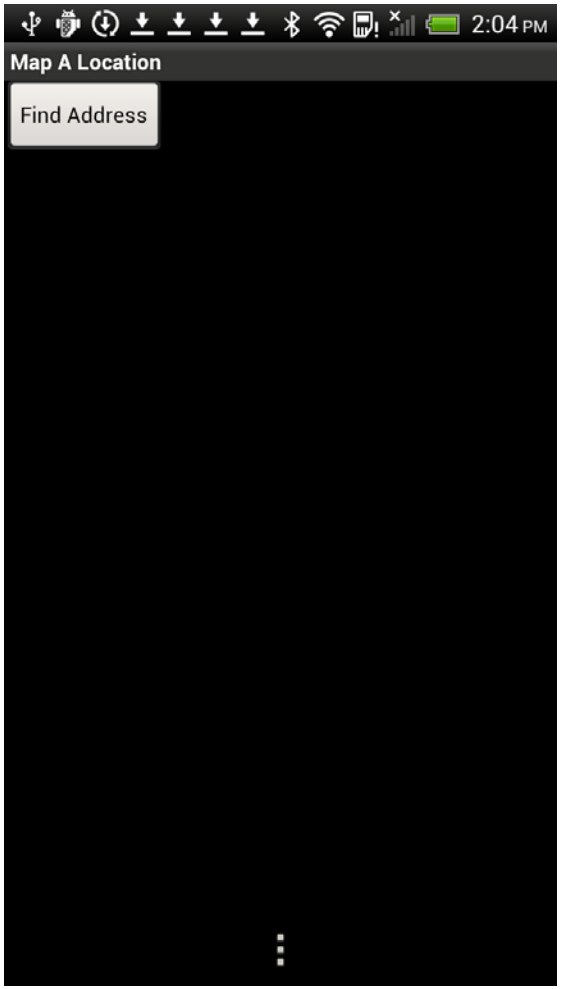
Starting Activities

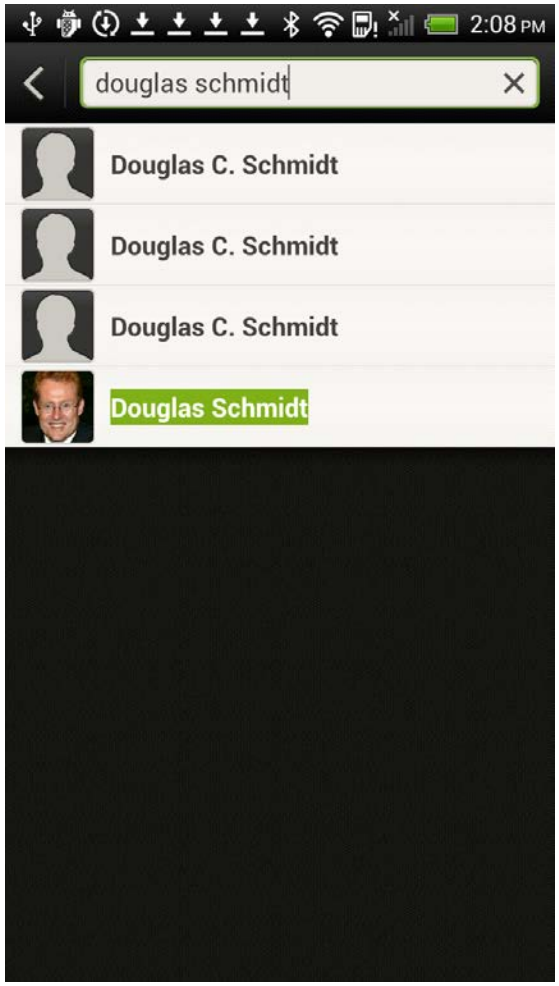
- Create an Intent object specifying the Activity to start
  - We'll discuss Intents in detail in later lectures
- Pass newly created Intent to one of the following methods
  - startActivity()
  - StartActivityForResult()
    - Callback to return result when called Activity finishes

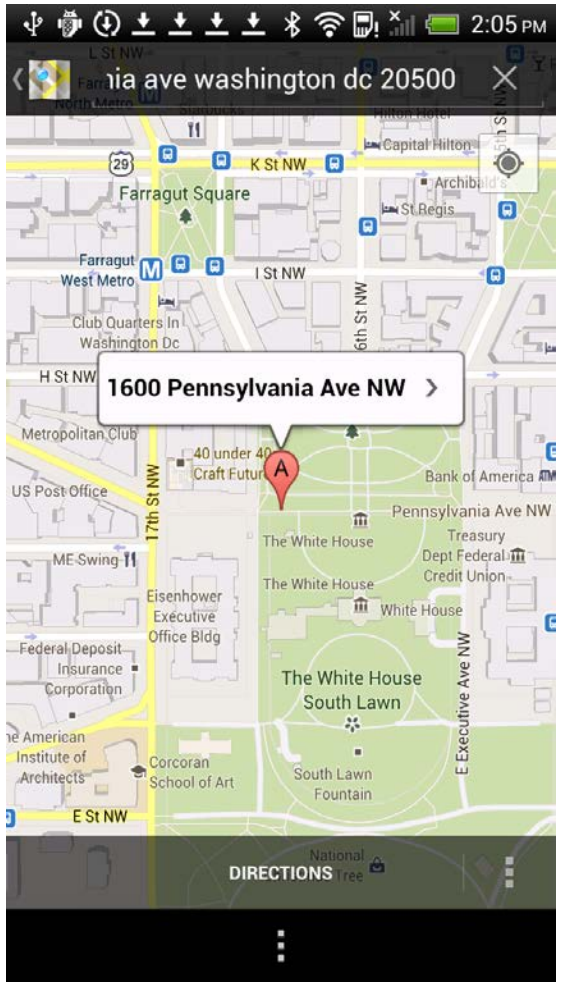
Using startActivity() in Map App

```
protected void onCreate(Bundle savedInstanceState) {
…
  public void onClick(View v) {
  …
    Intent geoIntent = new Intent(android.content.Intent.ACTION_VIEW,
                                        Uri.parse("geo:0,0?q=" + address));
    startActivity(geoIntent);
  …
  }
…
}
```

MapLocationFromContacts







*Not really my address ☺

Using startActivityForResult()

```
private static final int PICK_CONTACT_REQUEST = 0;
...
protected void onCreate(Bundle savedInstanceState) {
  ...
    public void onClick(View v) {
       try {
         Intent intent = new Intent(Intent.ACTION_PICK,
                                    ContactsContract.Contacts.CONTENT_URI);
         startActivityForResult(intent, PICK_CONTACT_REQUEST);
       } catch (Exception e) {}
    }
  });
}
```

startActivityForResult() (cont.)

- Started Activity sets result by calling Activity.setResult()
  - public final void setResult (int resultCode)
  - public final void setResult (int resultCode, Intent data)
- resultCode (an int)
  - RESULT_CANCELED
  - RESULT_OK
  - RESULT_FIRST_USER
    - Custom resultCodes can be added after this

startActivityForResult() (cont.)

```
protected void onActivityResult(int requestCode, int resultCode, Intent data)
    {
 if (resultCode == Activity.RESULT_OK &&
    requestCode == PICK_CONTACT_REQUEST) {
  ...
  String address =  / * extract address from data */
  Intent geoIntent = new Intent(android.content.Intent.ACTION_VIEW,
                                        Uri.parse("geo:0,0?q=" + address));
  startActivity(geoIntent);
 }
}
```

Configuration Changes

- Device configuration can change at runtime
  - Keyboard, orientation, locale, etc
- On configuration changes, Android usually kills & restarts the current Activity
- Activity restarting should be fast. If necessary you can:
  - Retain an Object during a configuration change
  - Manually handle the configuration change

Retaining an Object

- Hard to recompute data can be cached to speed up handling of configuration changes
- Override onRetainNonConfigurationInstance() to build & return configuration Object
  - Will be called between onStop() & onDestroy()
- Call getLastNonConfigurationInstance() during onCreate() to recover retained Object
- **Note:** These methods have been deprecated in favor of methods in the Fragment class (will discuss at a later date).

Manual Reconfiguration

- Can prevent system from restarting Activity
- Declare the configuration changes the Activity handles in AndroidManifest.xml file, e.g.,

```
<activity android:name=".MyActivity
  android:configChanges="orientation keyboardHidden"
  ...>
```

- When configuration changes, Activity's onConfigurationChanged() method is called & passed a Configuration object specifying the new device configuration

Source Code Examples

- MapLocationFromContacts