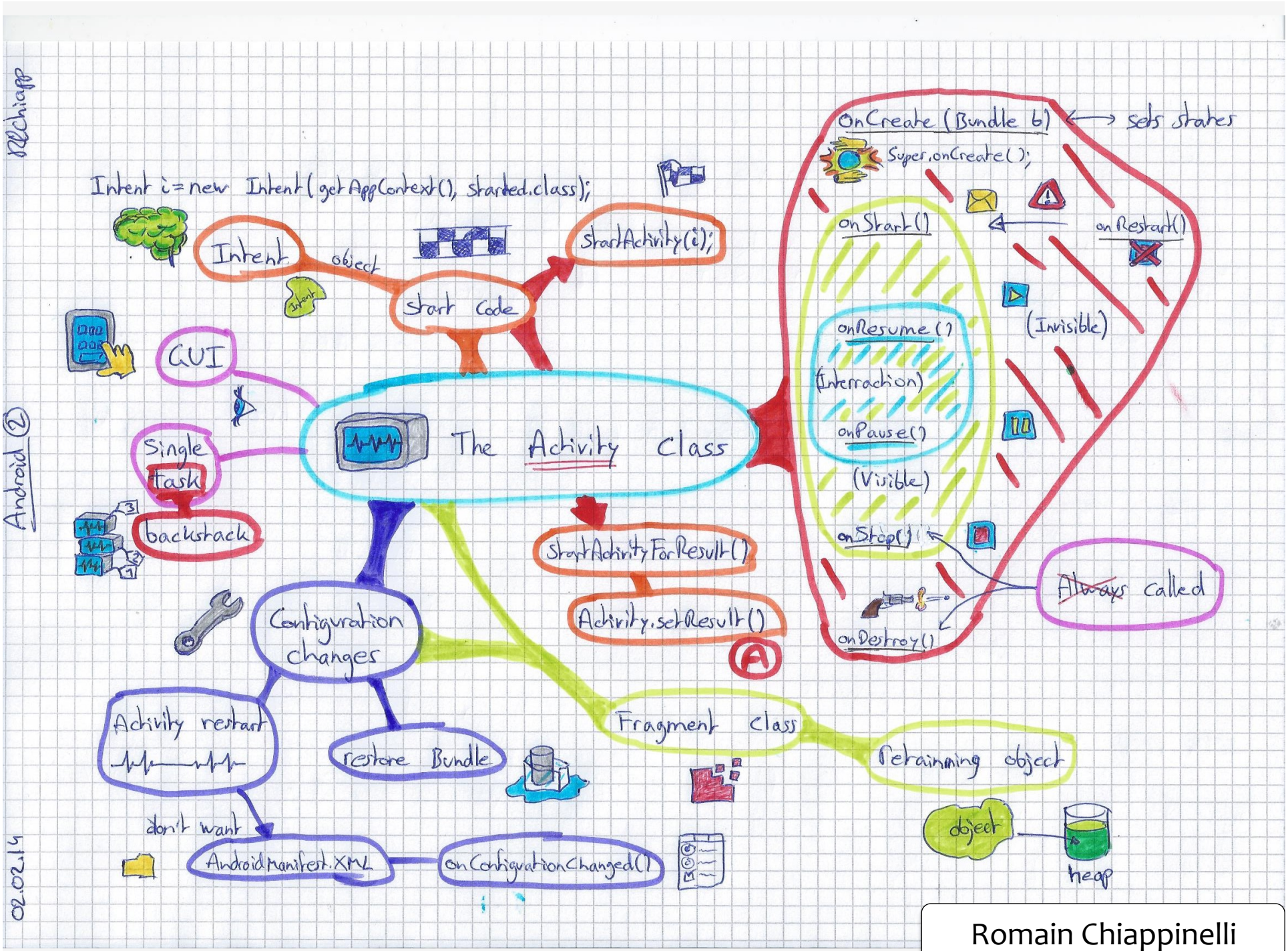


Romain Chiappinelli

Programming Handheld Systems

Adam Porter

The Activity Class

Today's topics

The activity class

The task backstack

The activity lifecycle

Starting activities

Handling configuration changes

Activity

Provides a visual interface for user interaction

Activity

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

- Viewing an email message
- Showing a login screen

Activity

Applications often comprise several activities

Navigation through Activities

Android supports navigation in several ways:

Tasks

The task backstack

Suspending & resuming activities

Tasks

A Task is a set of related Activities

These related activities don't have to be part of the same application

Most tasks start at the home screen

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

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

Task Backstack

The Activity lifecycle

Activities are created, suspended, resumed & destroyed as necessary when an application executes

The Activity lifecycle

- Some of these actions depend on user behavior
- Some depend on Android
  - e.g., 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 activity by calling specific activity methods

Some Activity Callback Methods

protected void onCreate (Bundle savedInstanceState)

protected void onStart()

protected void onResume()

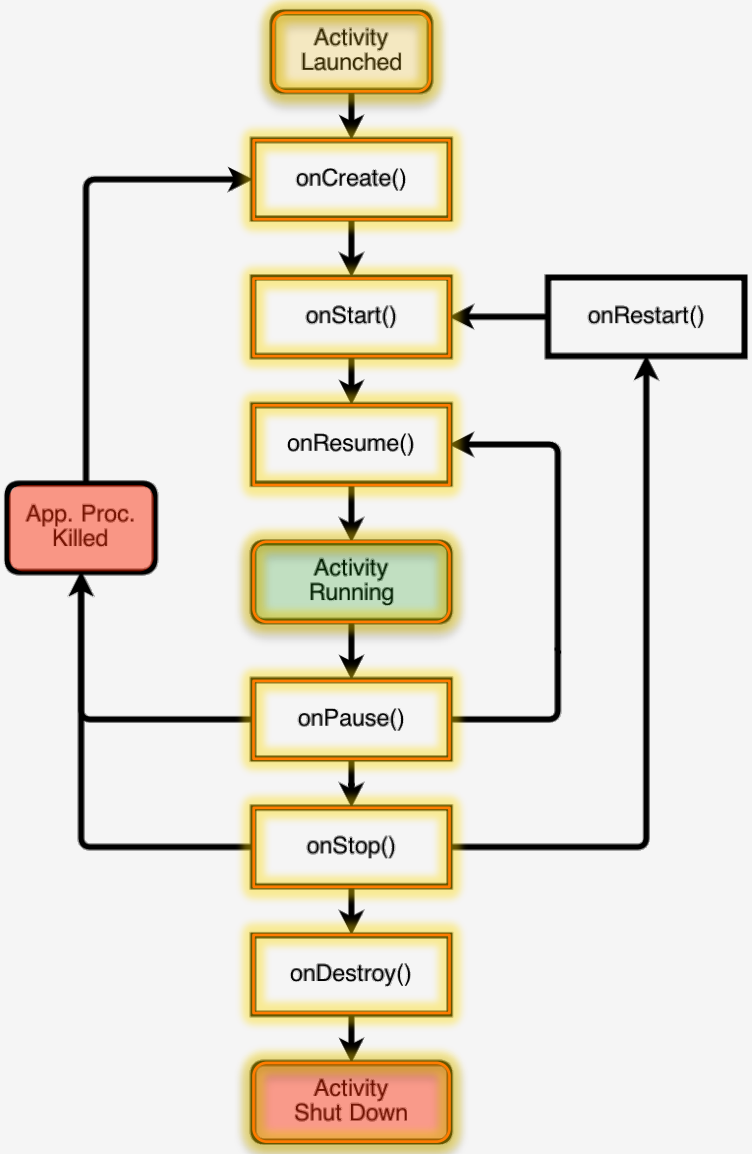
protected void onPause()

protected void onRestart()

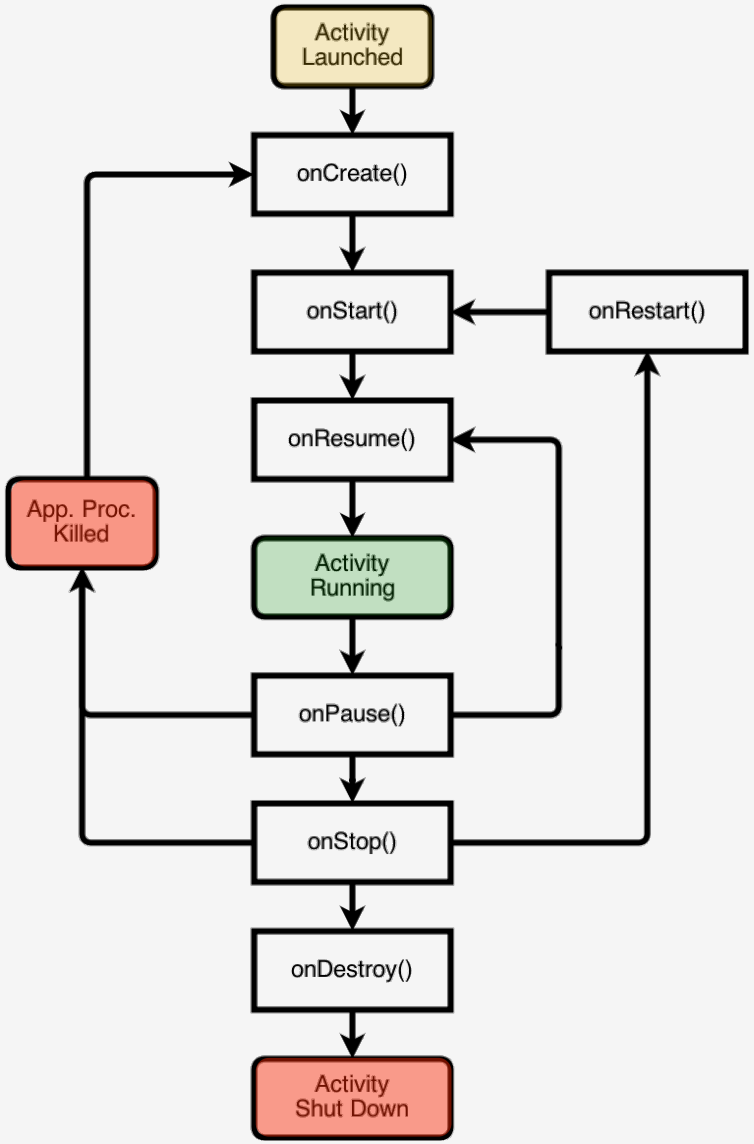
protected void onStop()

protected void onDestroy()

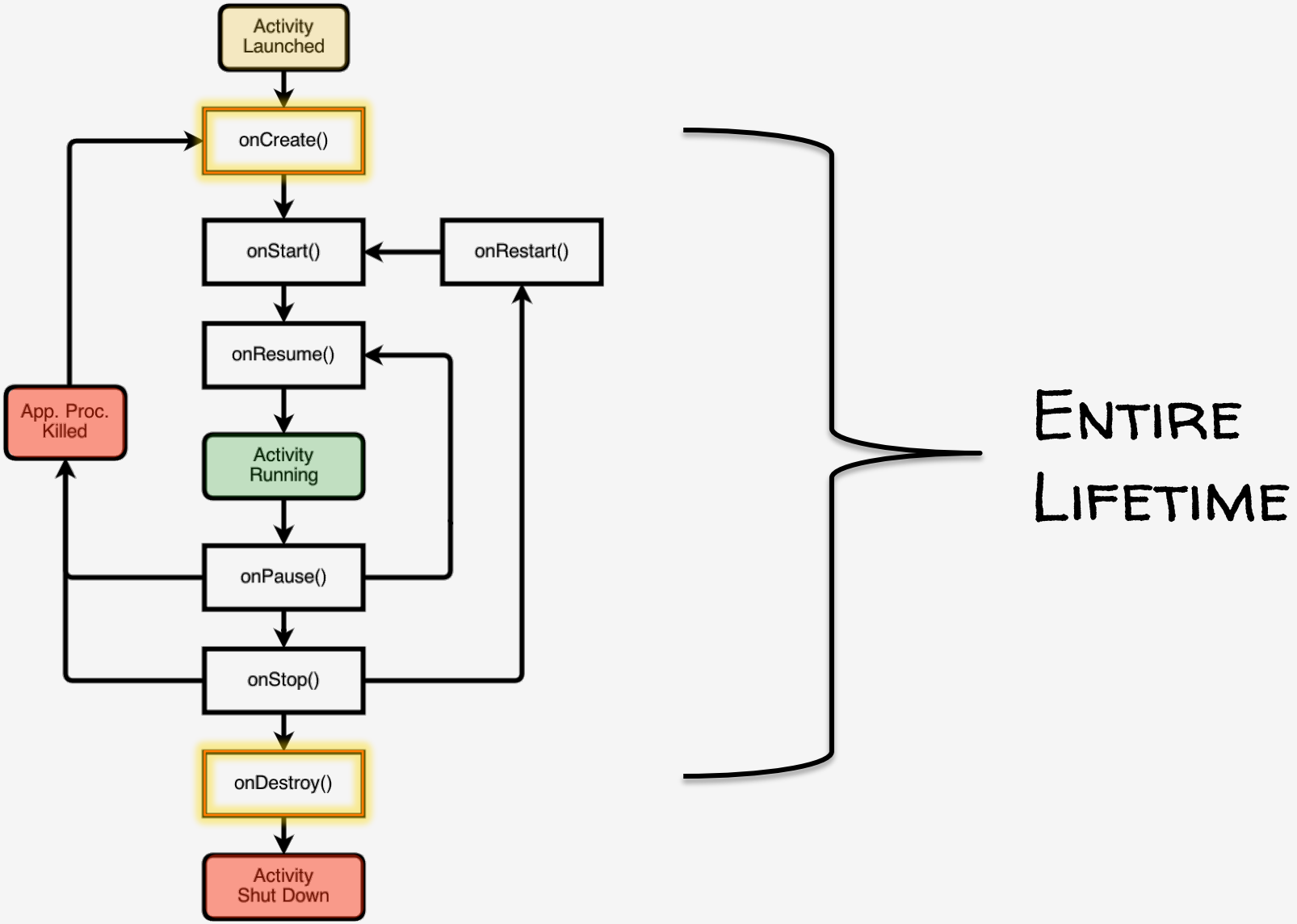
The Activity Lifecycle



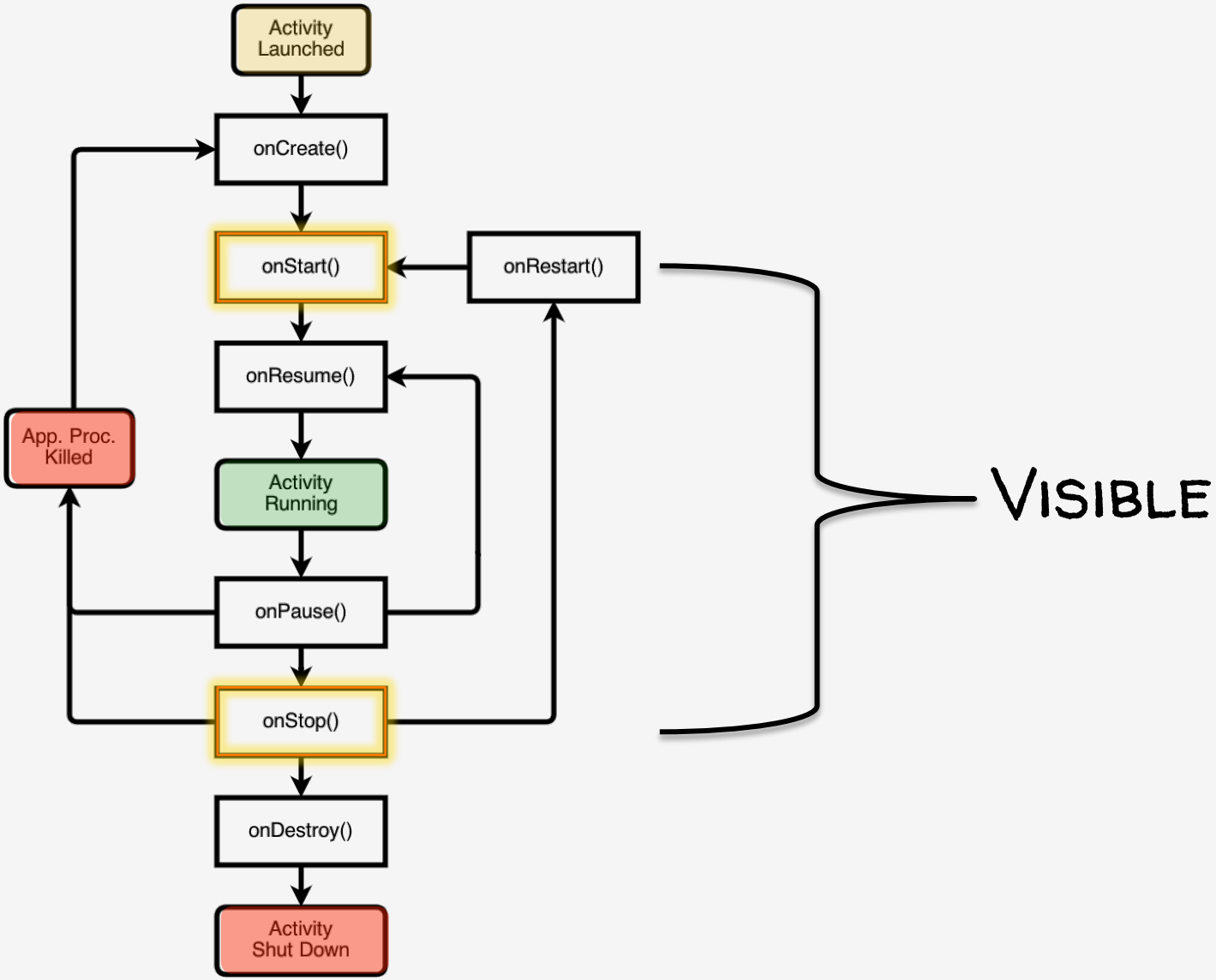
The Activity Lifecycle



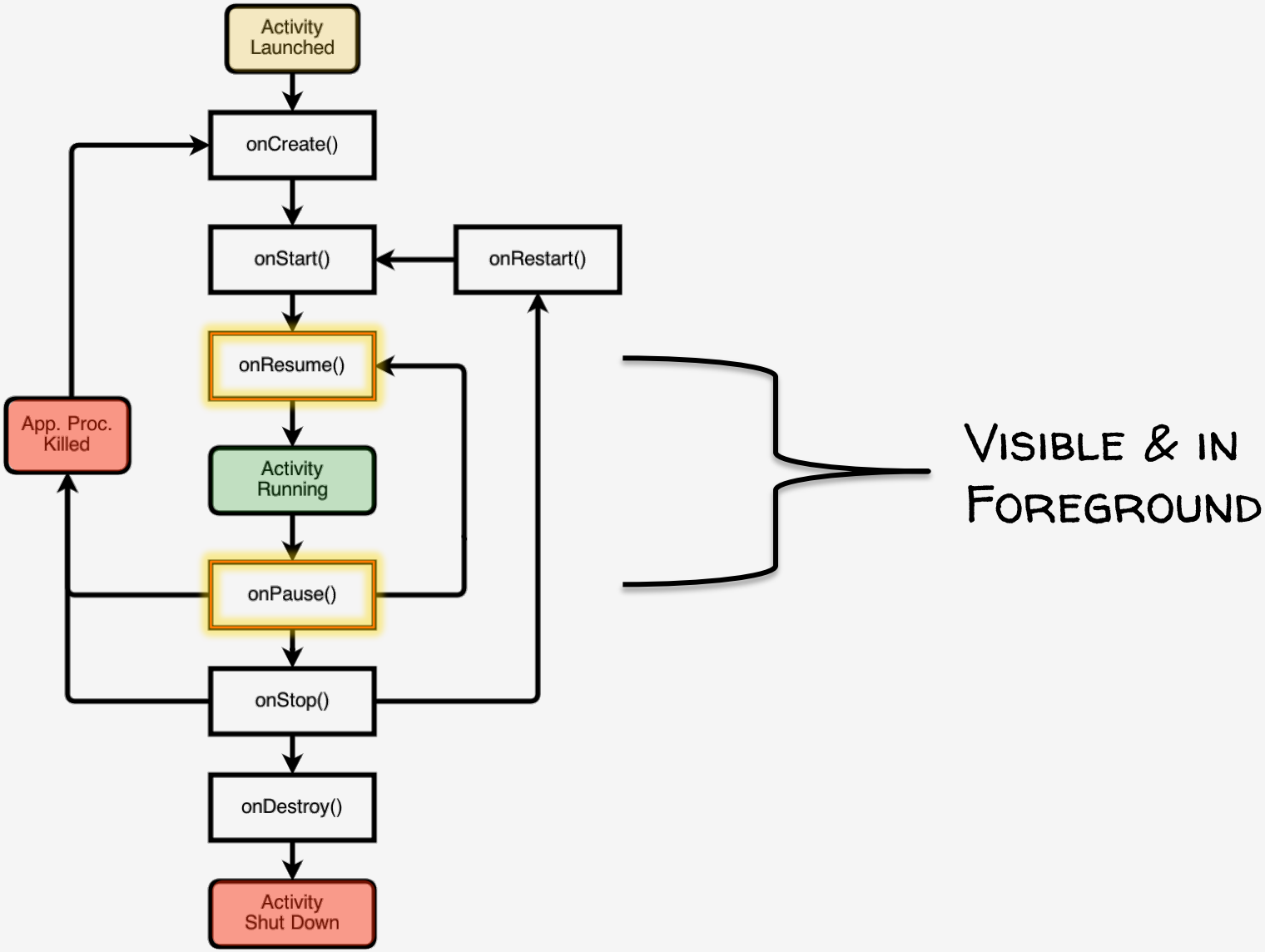
The Activity Lifecycle



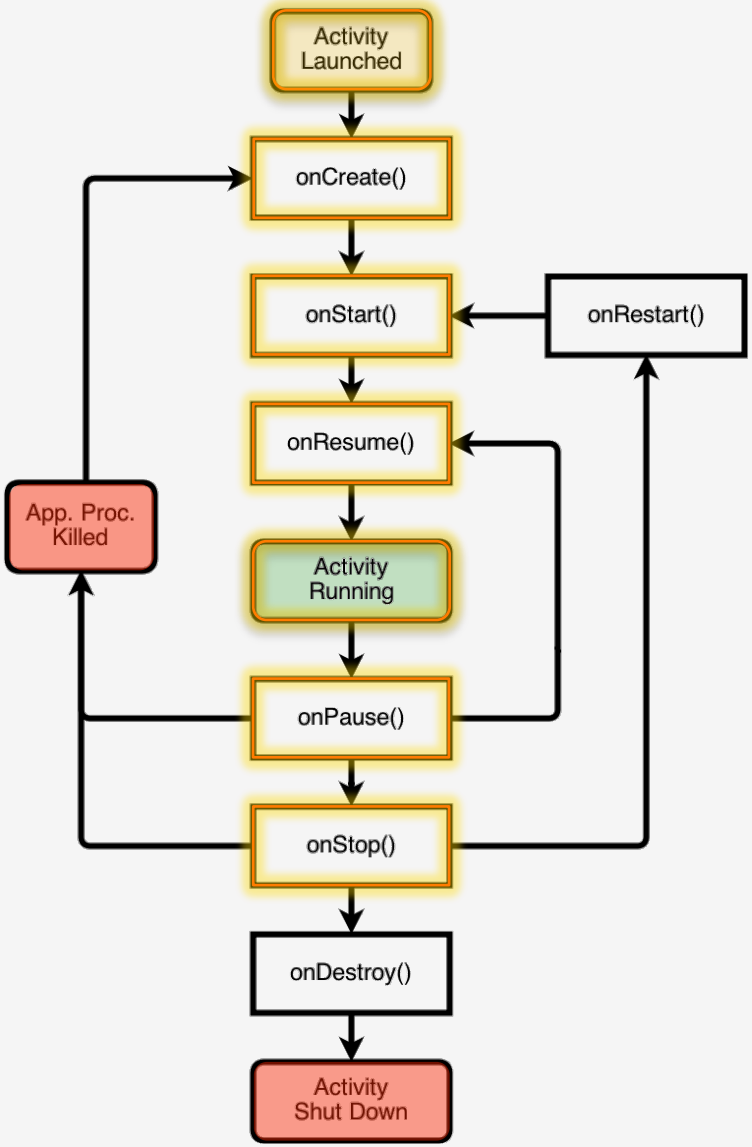
The Activity Lifecycle



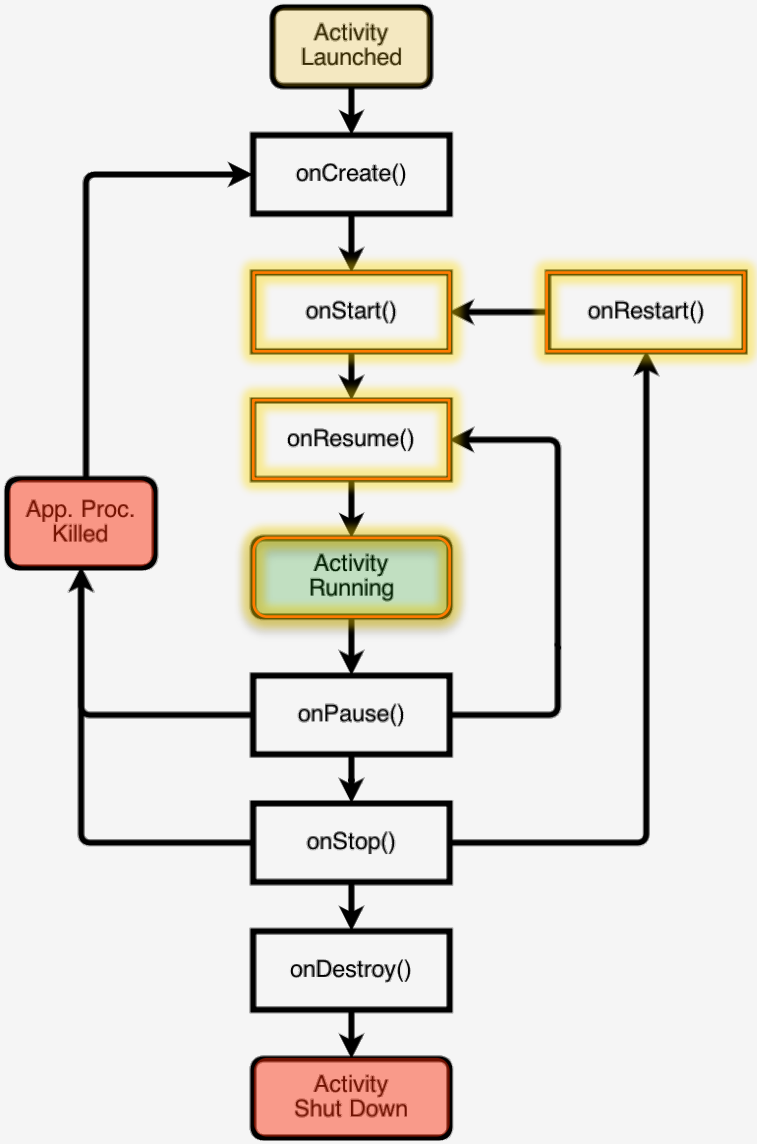
The Activity Lifecycle



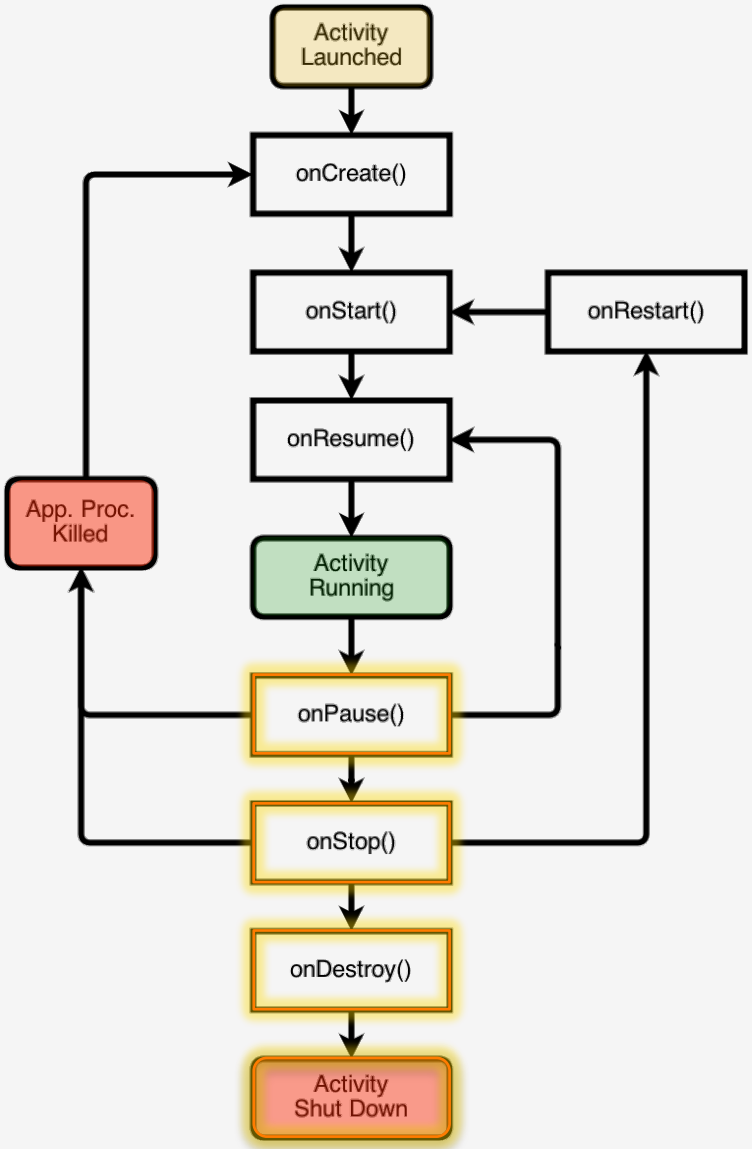
The Activity Lifecycle

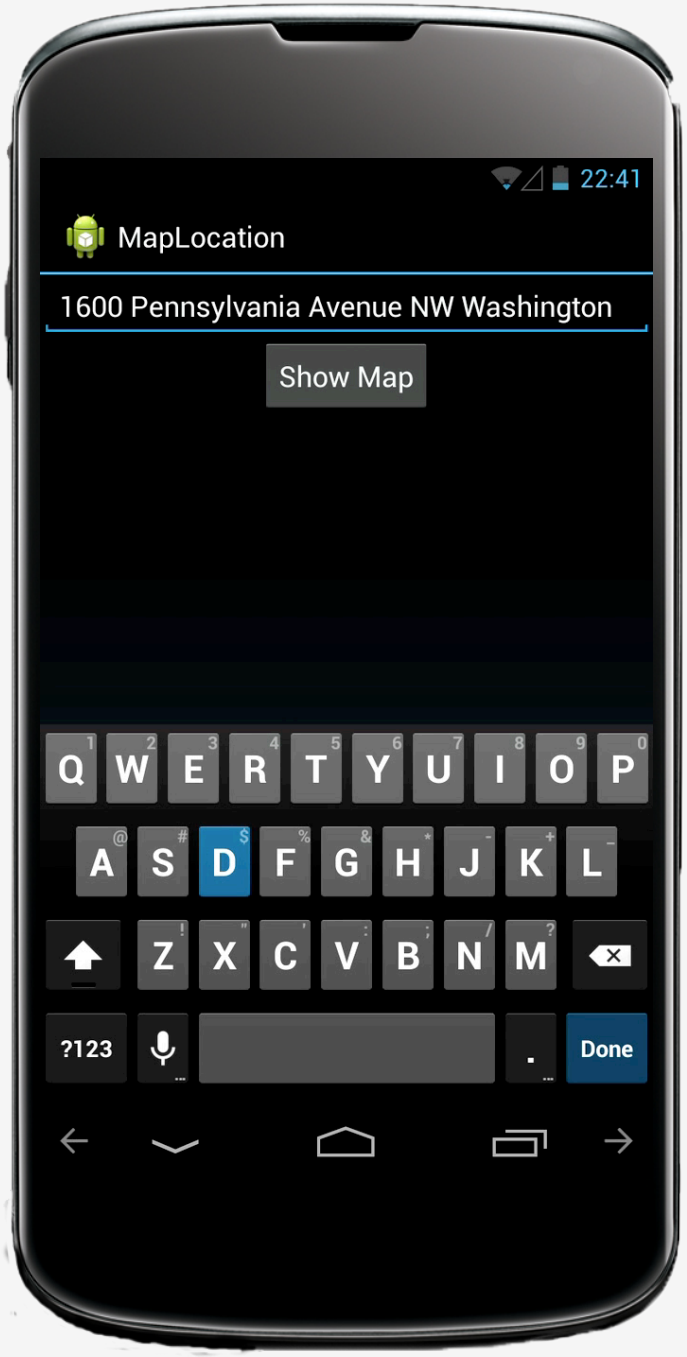


The Activity Lifecycle



The Activity Lifecycle




22:41
MapLocation
1600 Pennsylvania Avenue NW Washington
Show Map
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
?123
Done

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

MapLocation

```
16  public class MapLocation extends Activity {
17
18      private final String TAG = "MapLocation";
19
20      @Override
21      protected void onCreate(Bundle savedInstanceState) {
22
23          // Required call through to Activity.onCreate()
24          // Restore any saved instance state
25          super.onCreate(savedInstanceState);
26
27          // Set content view
28          setContentView(R.layout.main);
29
30          // Initialize UI elements
31          final EditText addrText = (EditText) findViewById(R.id.location);
32          final Button button = (Button) findViewById(R.id.mapButton);
33
```

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

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
- Save persistent state

onStop()

Activity is no longer visible to user

- may be restarted later

Typical actions

- Cache state

Note: may not be called if Android kills your application

onDestroy()

Activity is about to be destroyed

Typical actions

- Release Activity resources

Note: may not be called if Android kills your application

MapLocation

```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\")");
}
```

MapLocation

```
@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 specifying the Activity to start

Starting Activities

Pass newly created Intent to methods, such as:

startActivity()

startActivityForResult()

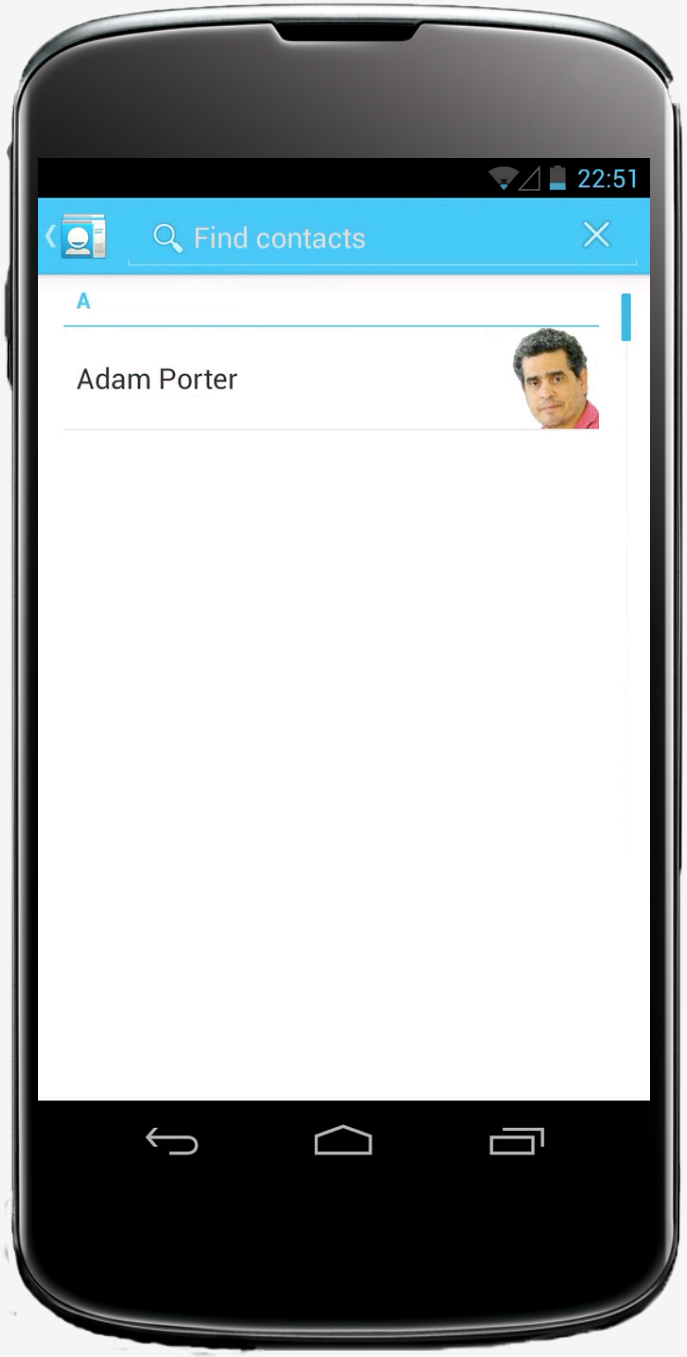
- Invokes a Callback method when the called Activity finishes to return a result

MapLocation

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

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

        } catch (Exception e) {
            // Log any error messages to LogCat using Log.e()
            Log.e(TAG, e.toString());
        }
    }
});
```

MapLocationFromContacts

Similar to MapLocation, but gets address from contacts database


22:51
Find contacts
A
Adam Porter

MapLocationFromContacts

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    final Button button = (Button) findViewById(R.id.mapButton);
    button.setOnClickListener(new Button.OnClickListener() {

        // Called when user clicks the Show Map button
        @Override
        public void onClick(View v) {
            try {
                // Create Intent object for picking data from Contacts database
                Intent intent = new Intent(Intent.ACTION_PICK,
                        CONTACTS_CONTENT_URI);

                // Use intent to start Contacts application
                // Variable PICK_CONTACT_REQUEST identifies this operation
                startActivityForResult(intent, PICK_CONTACT_REQUEST);

            } catch (Exception e) {
                // Log any error messages to LogCat using Log.e()
                Log.e(TAG, e.toString());
            }
        }
    });
}
```

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

MapLocationFromContacts

```
@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) {

        // These details are covered in the lesson on ContentProviders
        ContentResolver cr = getContentResolver();
        Cursor cursor = cr.query(data.getData(), null, null, null, null);

        if (null != cursor && cursor.moveToFirst()) {
            String id = cursor
                    .getString(cursor.getColumnIndex(CONTACTS_ID));
            String where = DATA_CONTACT_ID + " = ? AND " + DATA_MIMETYPE
                    + " = ?";
            String[] whereParameters = new String[] { id,
                    STRUCTURED_POSTAL_CONTENT_ITEM_TYPE };
            Cursor addrCur = cr.query(DATA_CONTENT_URI, null, where,
                    whereParameters, null);
```

MapLocationFromContacts

```
            if (null != addrCur && addrCur.moveToFirst()) {
                String formattedAddress = addrCur
                        .getString(addrCur
                                .getColumnIndex(STRUCTURED_POSTAL_FORMATTED_ADDRESS));

                if (null != formattedAddress) {

                    // Process text for network transmission
                    formattedAddress = formattedAddress.replace(' ', '+');

                    // 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);
                }
            }
            if (null != addrCur)
                addrCur.close();
        }
        if (null != cursor)
            cursor.close();
    }
}
```

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

If necessary you can:

- Retain an Object containing important state information 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() and onDestroy()

Retaining an Object

Call getLastNonConfigurationInstance() during onCreate() to recover retained Object

Note: These methods have been deprecated in favor of methods in the Fragment class (discussed in later classes)

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

Next Time

The Intent Class