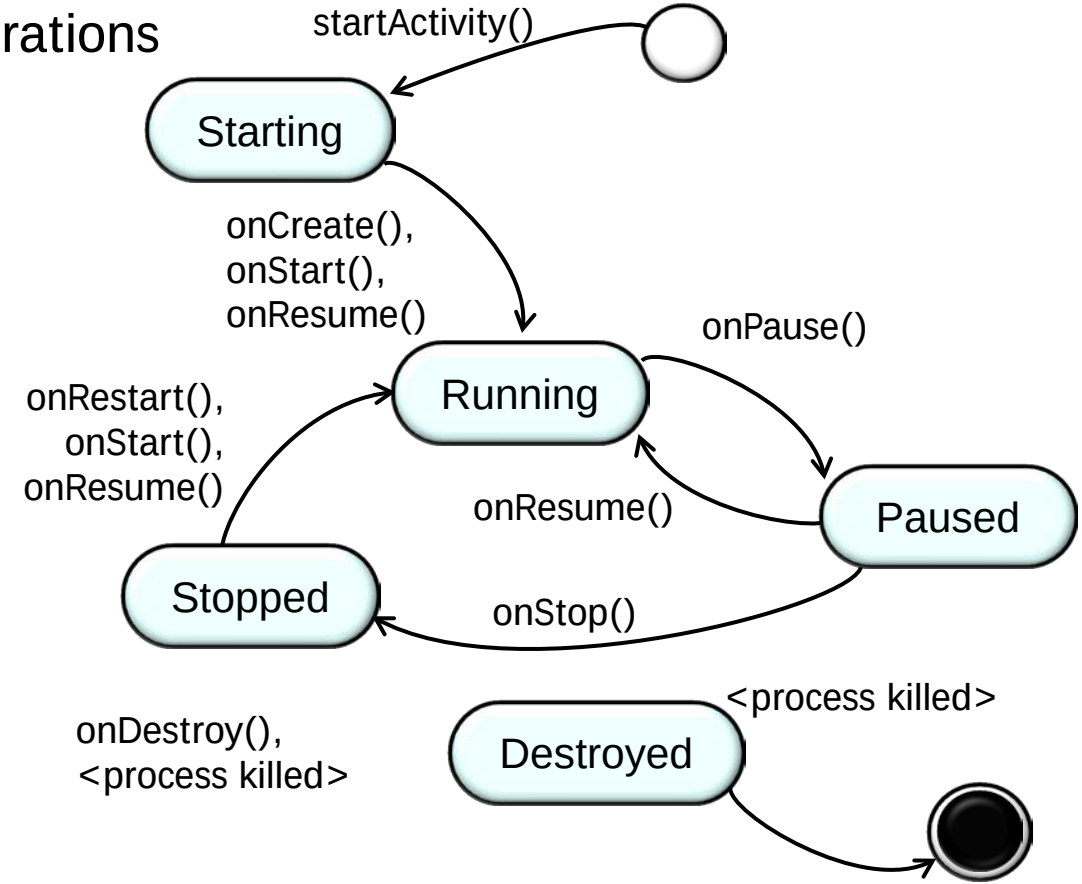
Activity Lifecycle Operations (Part 1)

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA

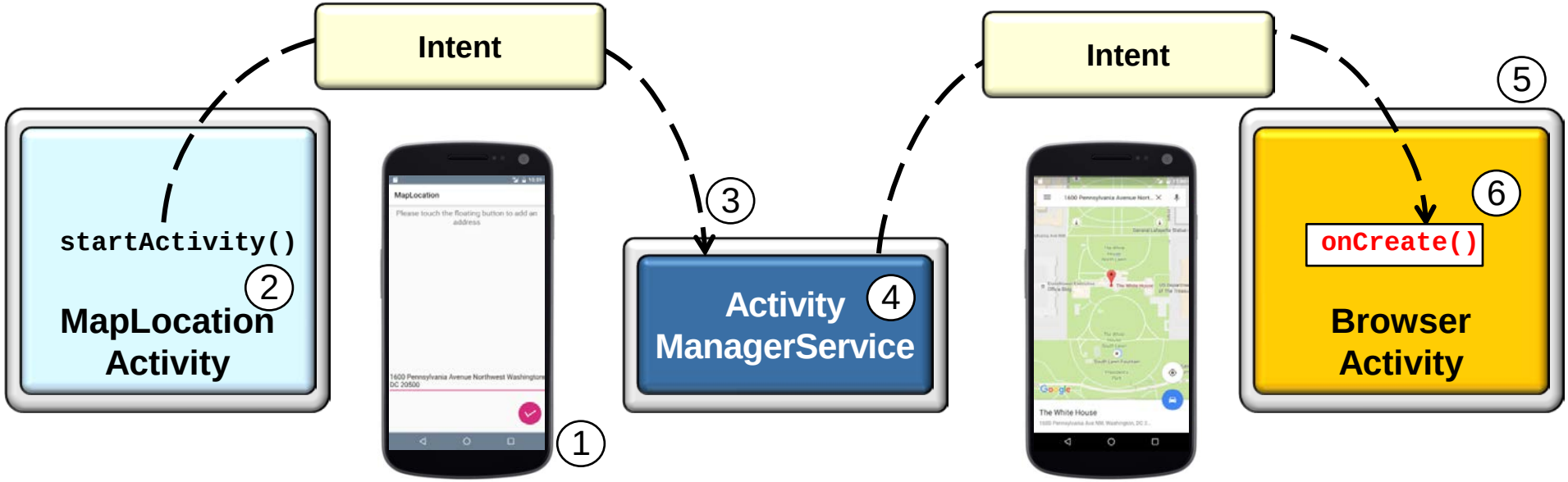
Learning Objectives in this Part of the Lesson

- Understand activity lifecycle operations



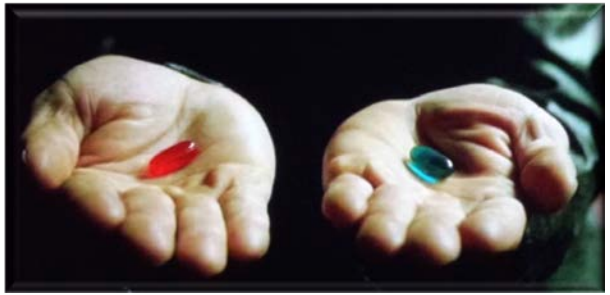
Learning Objectives in this Part of the Lesson

- Understand activity lifecycle operations
- Recognize the steps involved in starting an activity



Learning Objectives in this Part of the Lesson

- Understand activity lifecycle operations
- Recognize the steps involved in starting an activity
- Learn the methods used to start activities



void	startActivity(Intent intent) Launch a new activity.
void	startActivityForResult(Intent intent, int requestCode) Launch an activity for which you would like a result when it finished.

Learning Objectives in this Part of the Lesson

- Understand activity lifecycle operations
- Recognize the steps involved in starting an activity
- Learn the methods used to start activities
- Know how to finish an activity & return a result

Learning Objectives in this Part of the Lesson

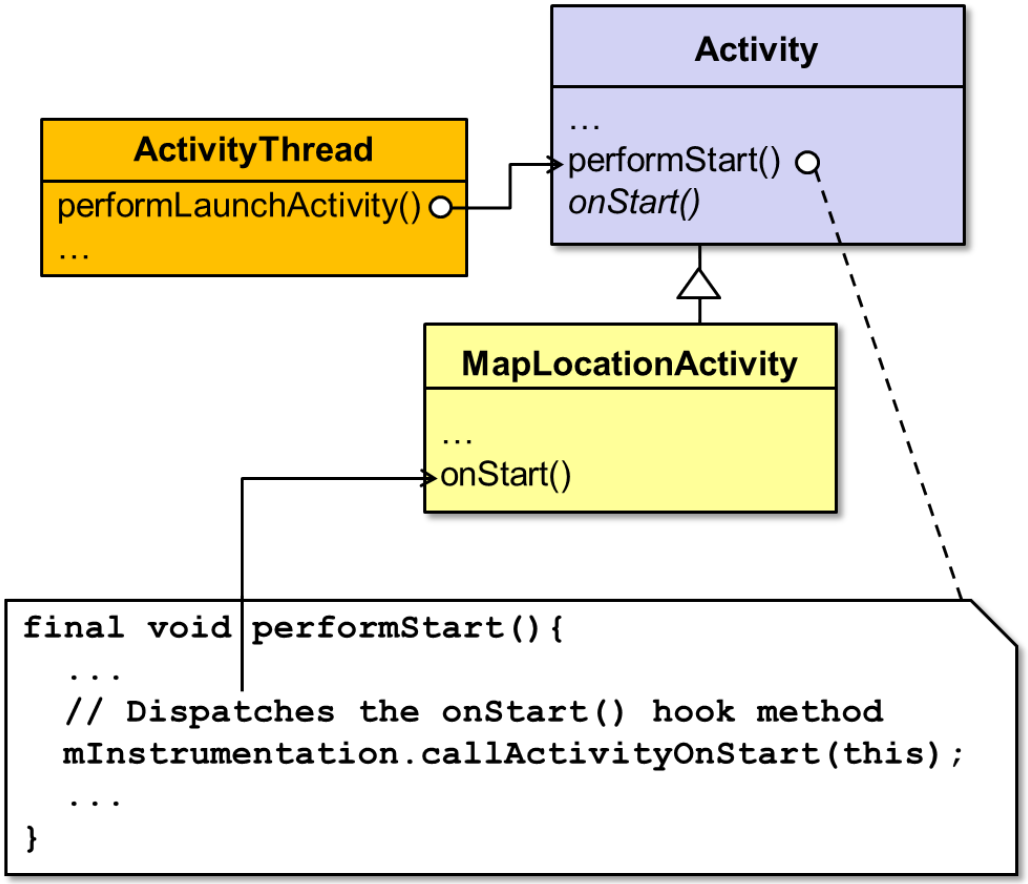
- Understand activity lifecycle operations
- Recognize the steps involved in starting an activity
- Learn the methods used to start activities
- Know how to finish an activity & return a result

Our case study apps are used as running examples, but the steps generalize

Overview of Activity Lifecycle Operations

Overview of Activity Lifecycle Operations

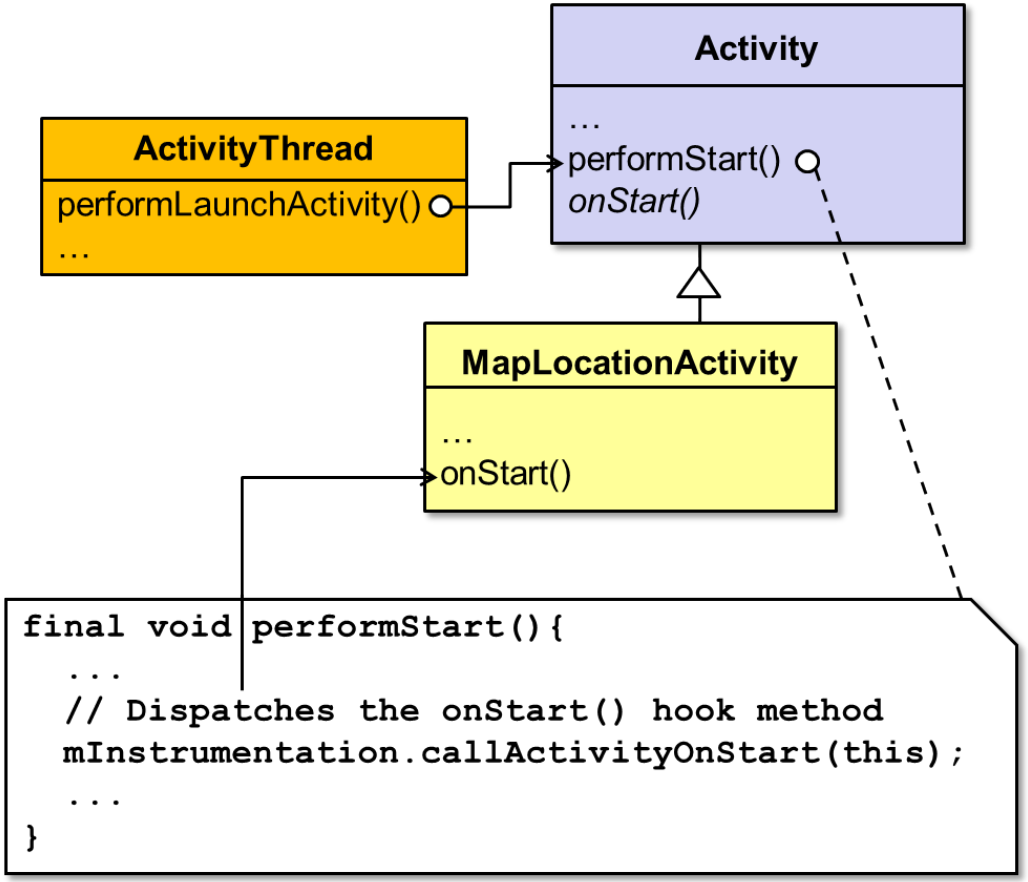
- Android activities are implemented as an object-oriented framework



See en.wikipedia.org/wiki/Software_framework

Overview of Activity Lifecycle Operations

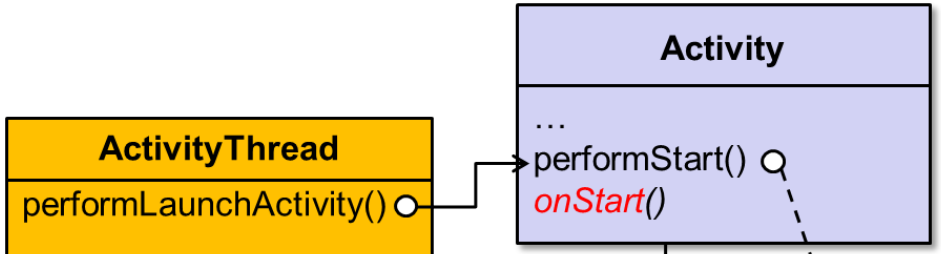
- Android activities are implemented as an object-oriented framework
  - e.g. “inversion of control” via callbacks on hook methods



See www.dre.vanderbilt.edu/~schmidt/hollywood-principle.txt

Overview of Activity Lifecycle Operations

- Android activities are implemented as an object-oriented framework
  - e.g. "inversion of control" via callbacks on hook methods



Hook methods customize reusable framework classes to run app-specific logic

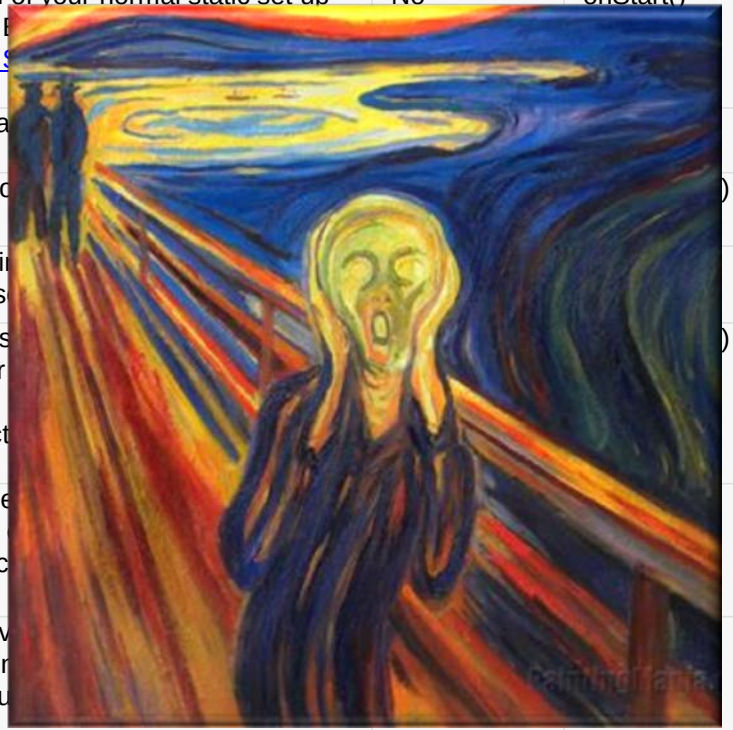
```
final void performStart(){
  ...
  // Dispatches the onStart() hook method
  mInstrumentation.callActivityOnStart(this);
  ...
}
```

See en.wikipedia.org/wiki/Template_method_pattern

Overview of Activity Lifecycle Operations

- There are many hook methods & interactions in Android's Activity framework

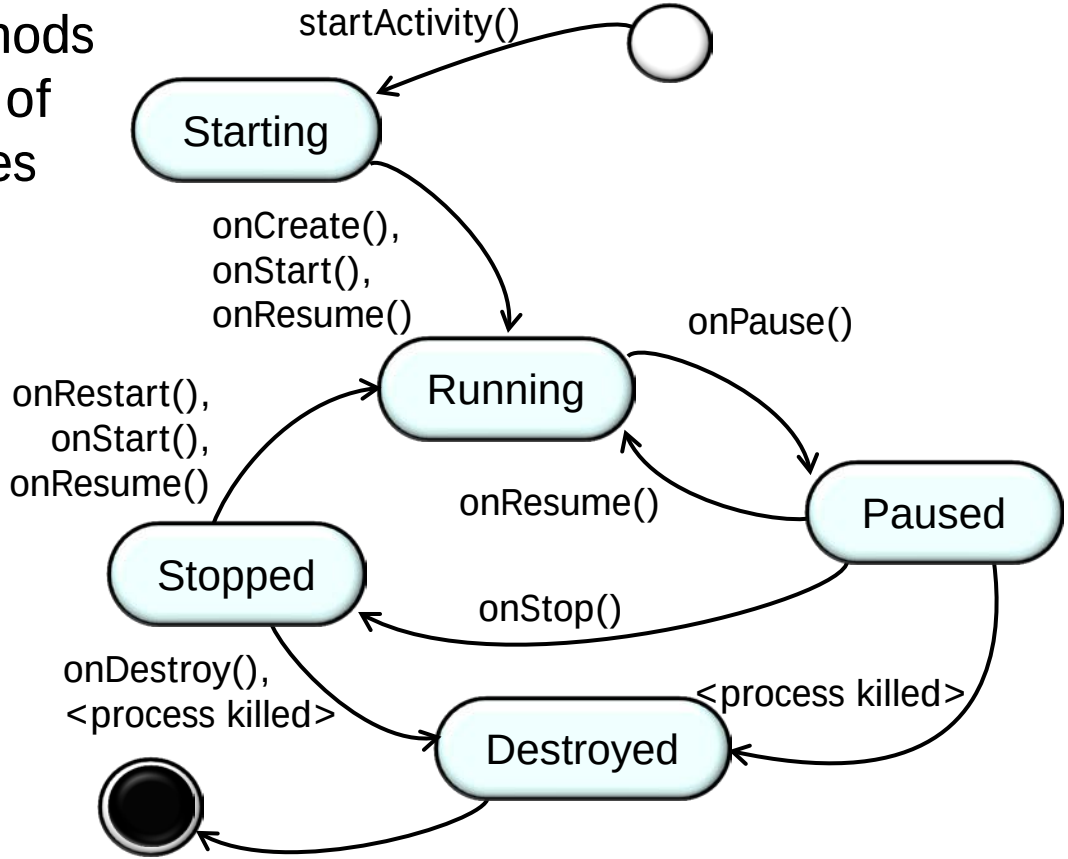
Method	Description	Killable after?	Next
onCreate()	Called when the activity is first created. This is where you should do all of your normal static set up — create views, bind data to lists, and so on. This method is passed a activity's previous state, if that state was captured (see Saving Activity by onStart().	No	onStart()
onRestart()	Called after the activity has been stopped, just prior to it being started a by onStart()		
onStart()	Called just before the activity becomes visible to the user.Followed by o comes to the foreground, or onStop() if it becomes hidden.		)
onResume()	Called just before the activity starts interacting with the user. At this poi the activity stack, with user input going to it.Always followed by onPaus		
onPause()	Called when the system is about to start resuming another activity. This commit unsaved changes to persistent data, stop animations and other consuming CPU, and so on. It should do whatever it does very quickly, not be resumed until it returns.Followed either by onResume() if the act or by onStop() if it becomes invisible to the user.		)
onStop()	Called when the activity is no longer visible to the user. This may happe destroyed, or because another activity (either an existing one or a new is covering it.Followed either by onRestart() if the activity is coming bac byonDestroy() if this activity is going away.		
onDestroy()	Called before the activity is destroyed. This is the final call that the activ called either because the activity is finishing (someone called finish() or is temporarily destroying this instance of the activity to save space. You these two scenarios with the isFinishing() method.		



See developer.android.com/guide/components/activities.html#ImplementingLifecycleCallbacks

Overview of Activity Lifecycle Operations

- The behavior of these hook methods can be expressed as a sequence of events occurring in multiple states

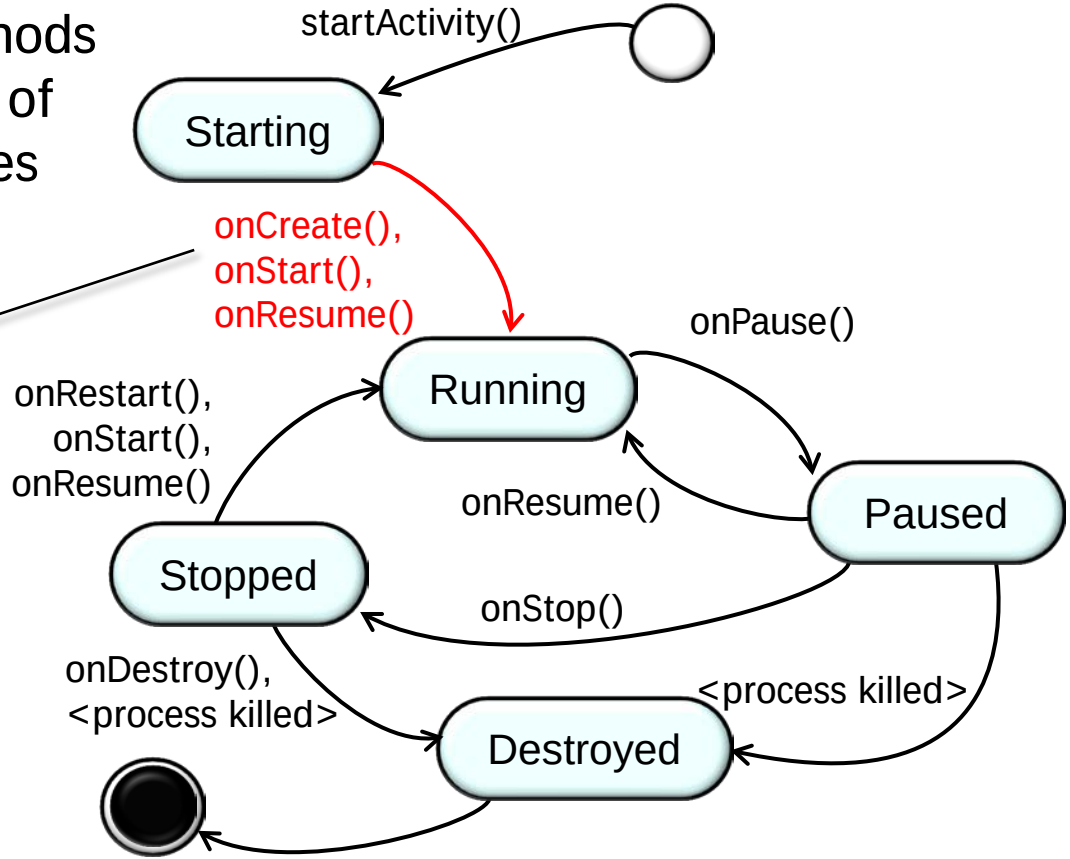


See en.wikipedia.org/wiki/UML_state_machine

Overview of Activity Lifecycle Operations

- The behavior of these hook methods can be expressed as a sequence of events occurring in multiple states

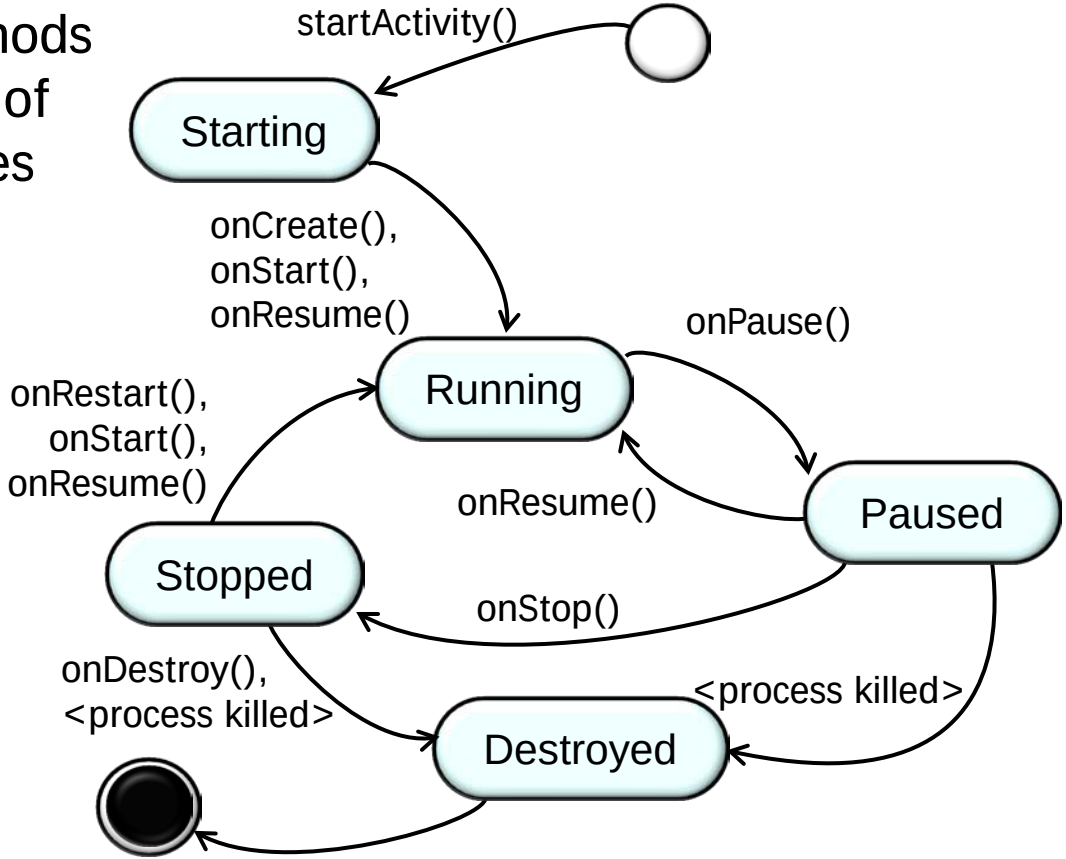
Transitions between each state are associated with various hook methods



See en.wikipedia.org/wiki/UML_state_machine

Overview of Activity Lifecycle Operations

- The behavior of these hook methods can be expressed as a sequence of events occurring in multiple states



State diagrams help enhance clarity & manage control flow in complex software

Overview of Activity Lifecycle Operations

- **Activity lifecycle states**

Activity Lifecycle

Activities in the system are managed as an *activity stack*. When a new activity is started, it is placed on the top of the stack and becomes the running activity -- the previous activity always remains below it in the stack, and will not come to the foreground again until the new activity exits.

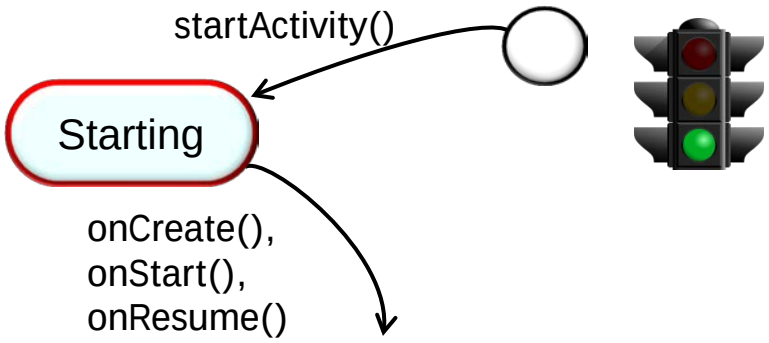
An activity has essentially four states:

- If an activity is in the foreground of the screen (at the top of the stack), it is *active* or *running*.
- If an activity has lost focus but is still visible (that is, a new non-full-sized or transparent activity has focus on top of your activity), it is *paused*. A paused activity is completely alive (it maintains all state and member information and remains attached to the window manager), but can be killed by the system in extreme low memory situations.
- If an activity is completely obscured by another activity, it is *stopped*. It still retains all state and member information, however, it is no longer visible to the user so its window is hidden and it will often be killed by the system when memory is needed elsewhere.
- If an activity is paused or stopped, the system can drop the activity from memory by either asking it to finish, or simply killing its process. When it is displayed again to the user, it must be completely restarted and restored to its previous state.

See developer.android.com/reference/android/app/Activity.html#ActivityLifecycle

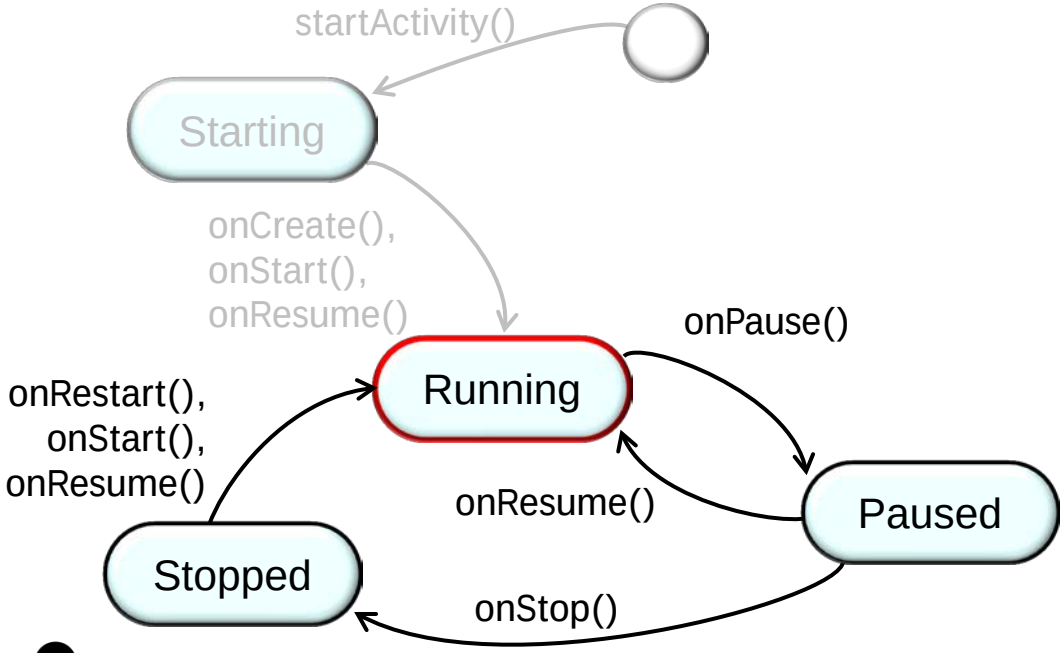
Overview of Activity Lifecycle Operations

- **Activity lifecycle states**
  - **Activity starting** – Initialization steps



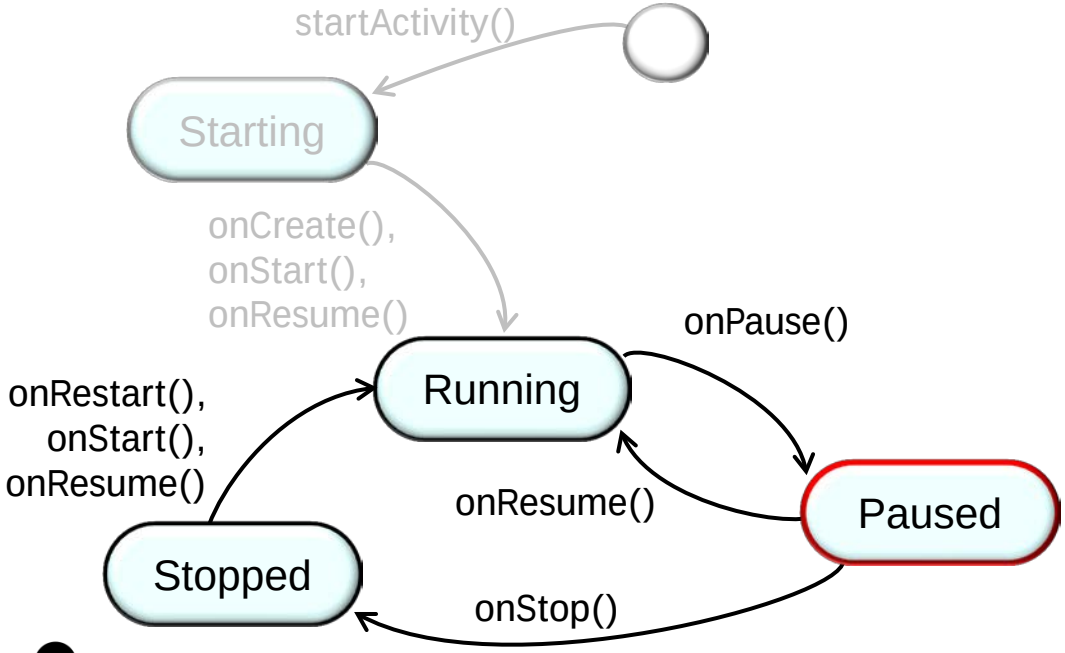
Overview of Activity Lifecycle Operations

- **Activity lifecycle states**
  - **Activity starting**
  - **Activity running**
    - *Running* – visible, has focus



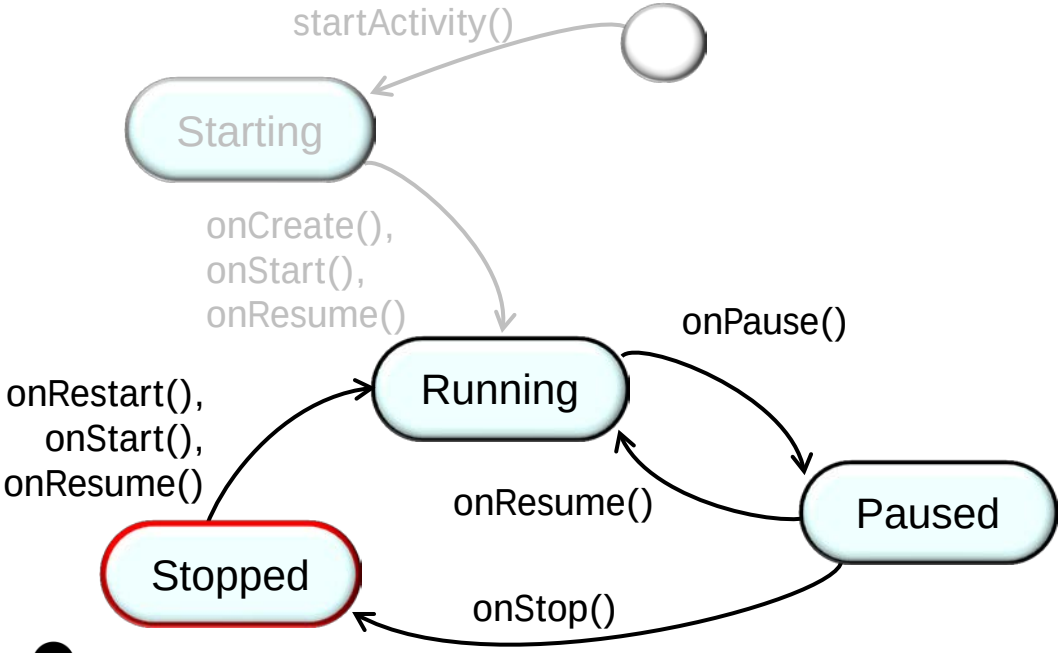
Overview of Activity Lifecycle Operations

- **Activity lifecycle states**
  - **Activity starting**
  - **Activity running**
    - *Running* – visible, has focus
    - *Paused* – visible, does not have focus, can be terminated



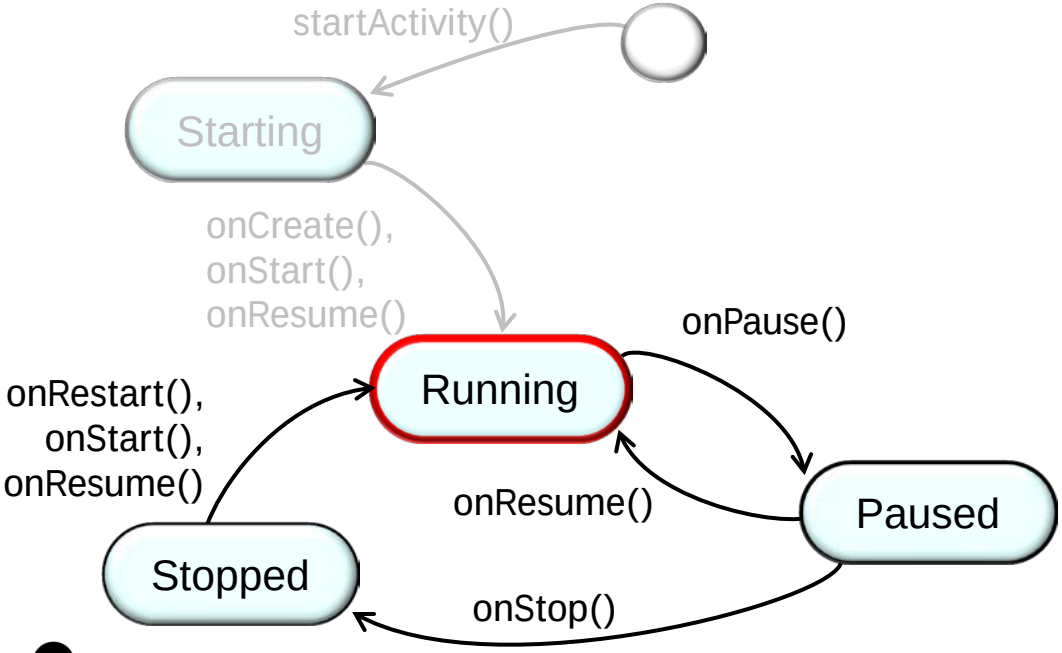
Overview of Activity Lifecycle Operations

- **Activity lifecycle states**
  - **Activity starting**
  - **Activity running**
    - *Running* – visible, has focus
    - *Paused* – visible, does not have focus, can be terminated
    - *Stopped* – not visible, does not have focus, can be terminated



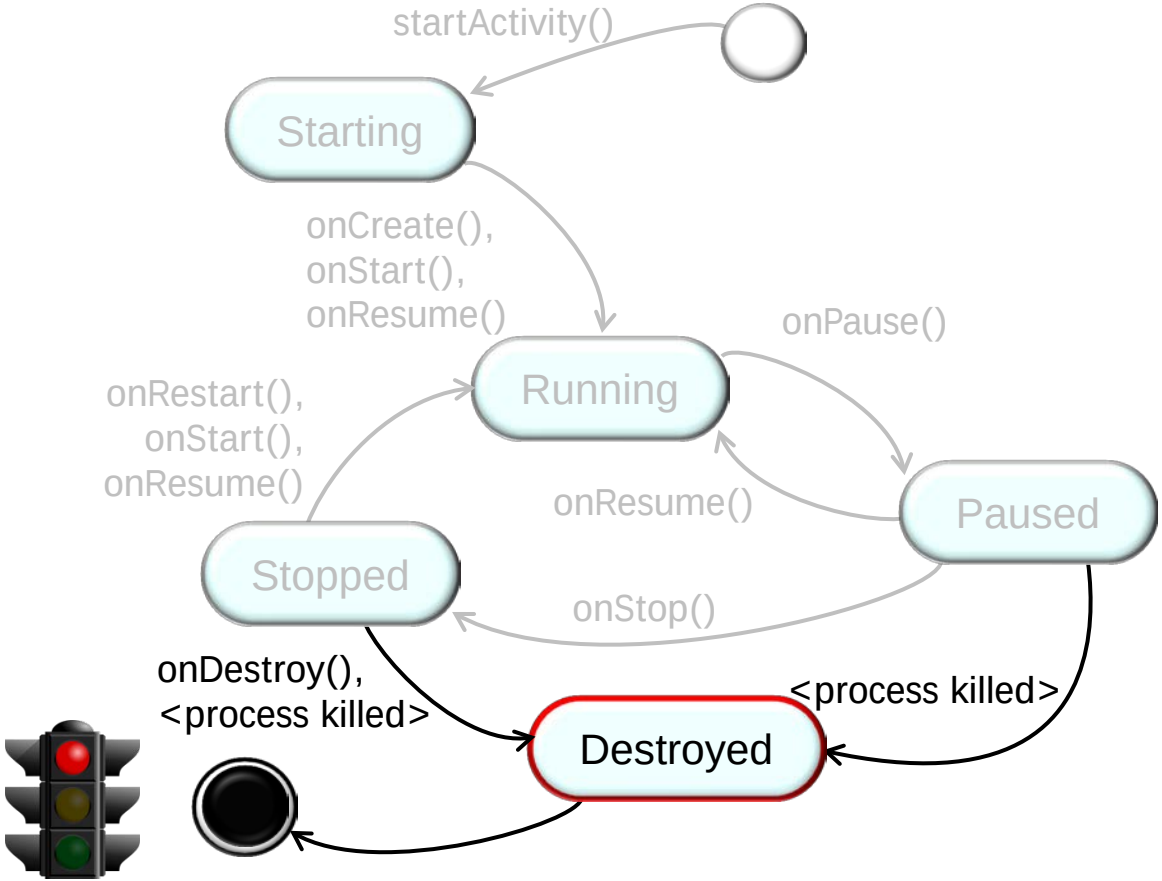
Overview of Activity Lifecycle Operations

- **Activity lifecycle states**
  - **Activity starting**
  - **Activity running**
    - *Running* – visible, has focus
    - *Paused* – visible, does not have focus, can be terminated
    - *Stopped* – not visible, does not have focus, can be terminated



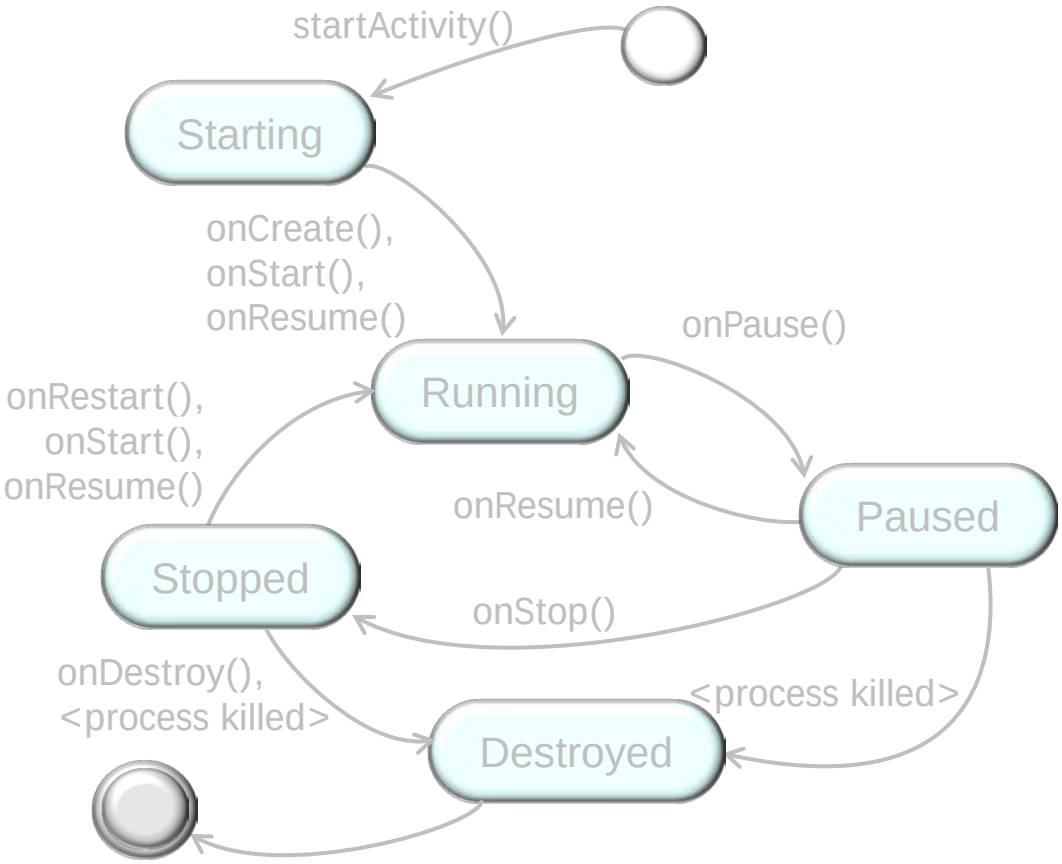
Overview of Activity Lifecycle Operations

- **Activity lifecycle states**
  - Activity starting
  - Activity running
  - **Activity shut down** – Voluntarily finished or involuntarily killed by the system



Overview of Activity Lifecycle Operations

- Activity lifecycle states
  - Activity starting
  - Activity running
  - Activity shut down



We'll next cover the steps associated with each of these states in more depth

Starting an Android Activity

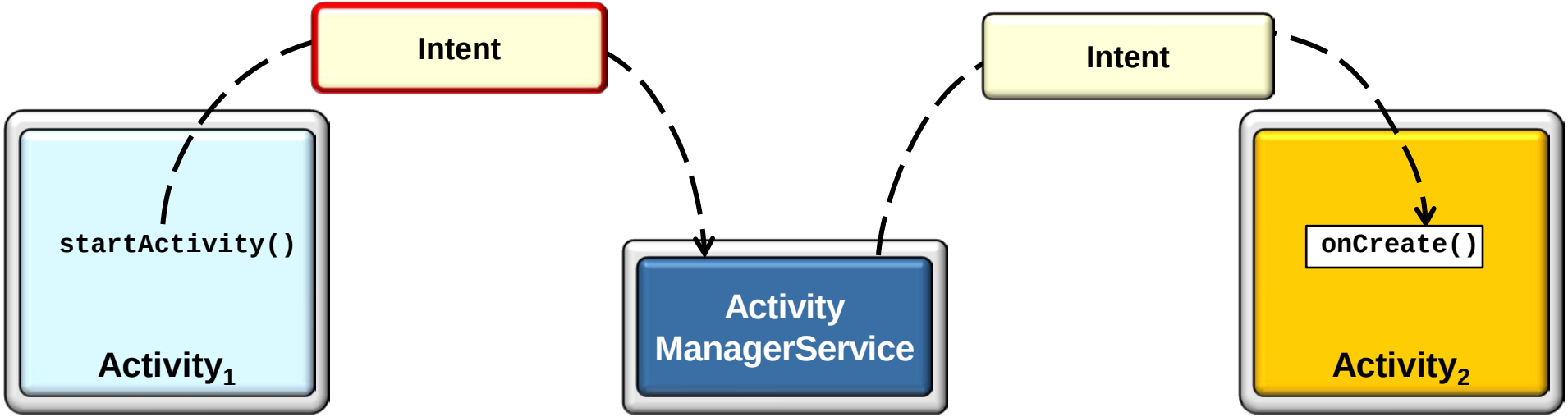
Starting an Android Activity

- An Activity can be launched on-demand

See developer.android.com/training/basics/firstapp/starting-activity.html

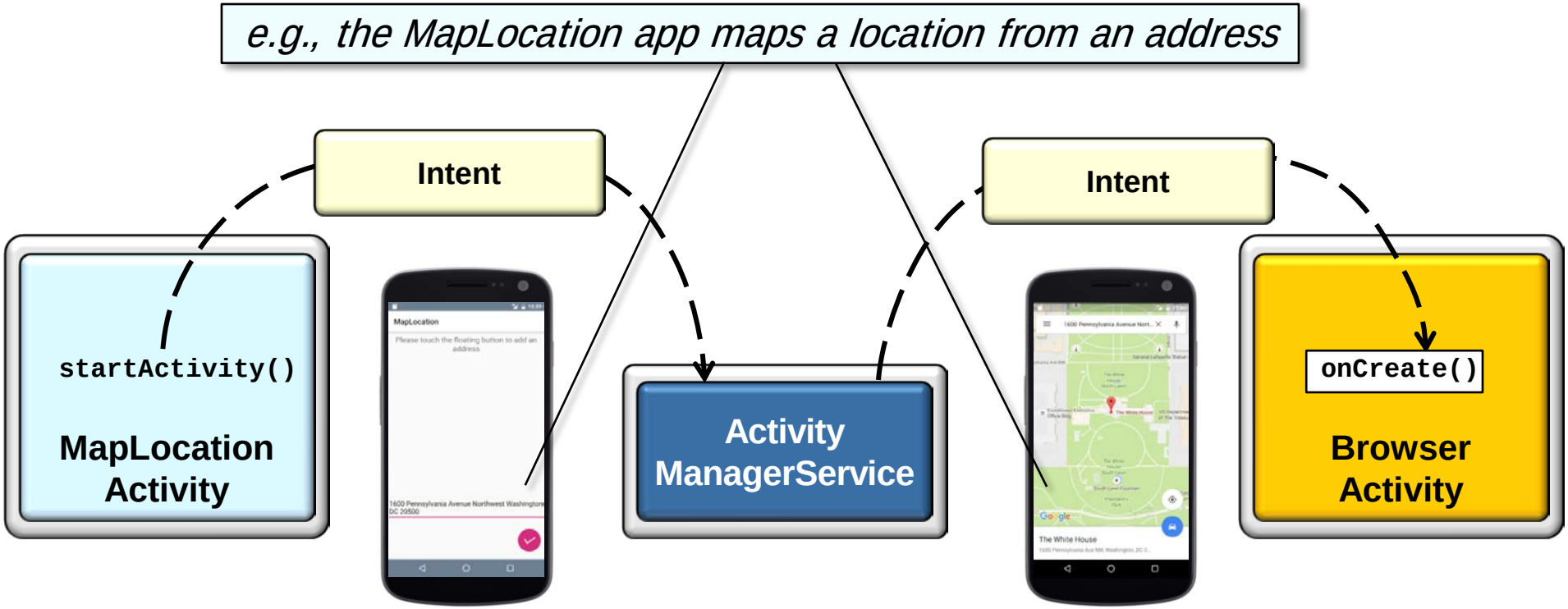
Starting an Android Activity

- An Activity can be launched on-demand via an intent passed to startActivity()



Starting an Android Activity

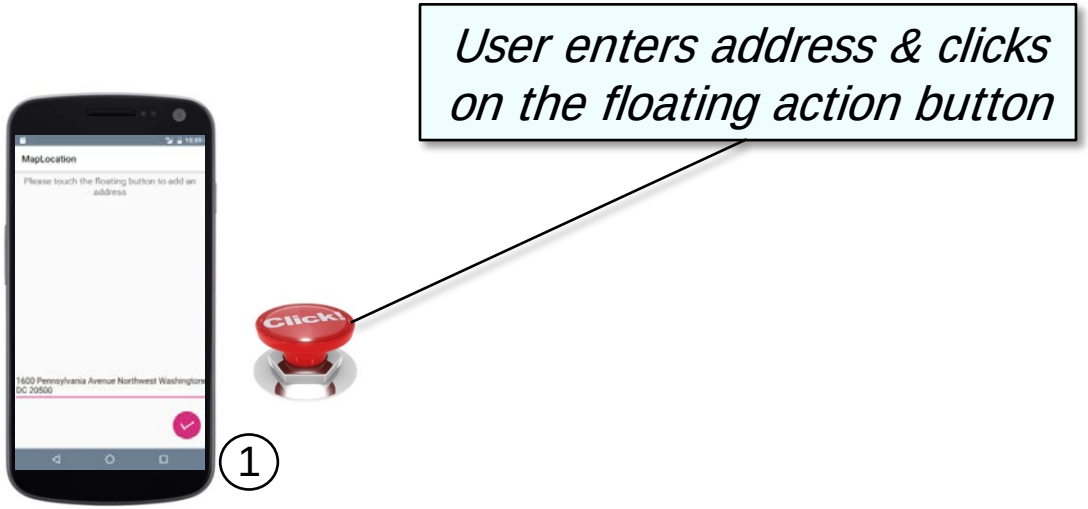
- An Activity can be launched on-demand via an intent passed to startActivity()



See gitlab.com/vandy-aad-2/MapLocation

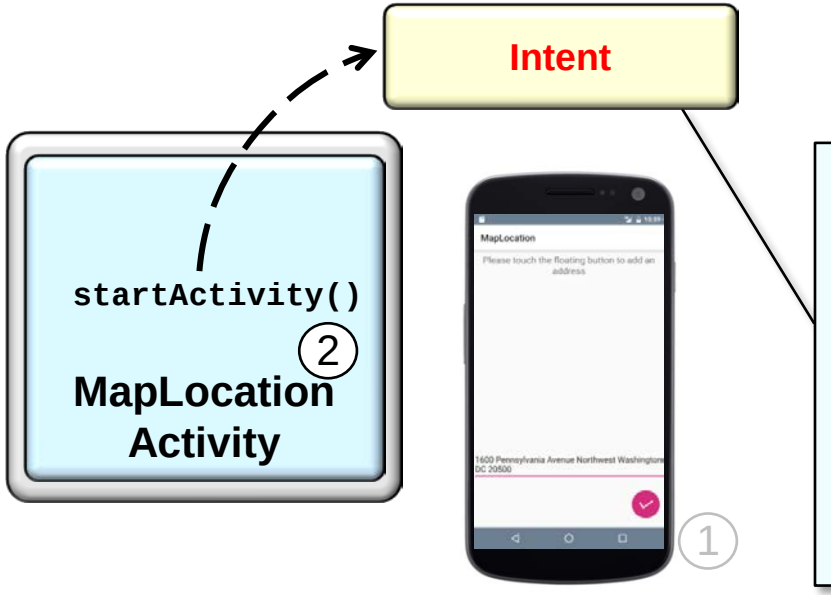
Starting an Android Activity

- An Activity can be launched on-demand via an intent passed to startActivity()



Starting an Android Activity

- An Activity can be launched on-demand via an intent passed to startActivity()

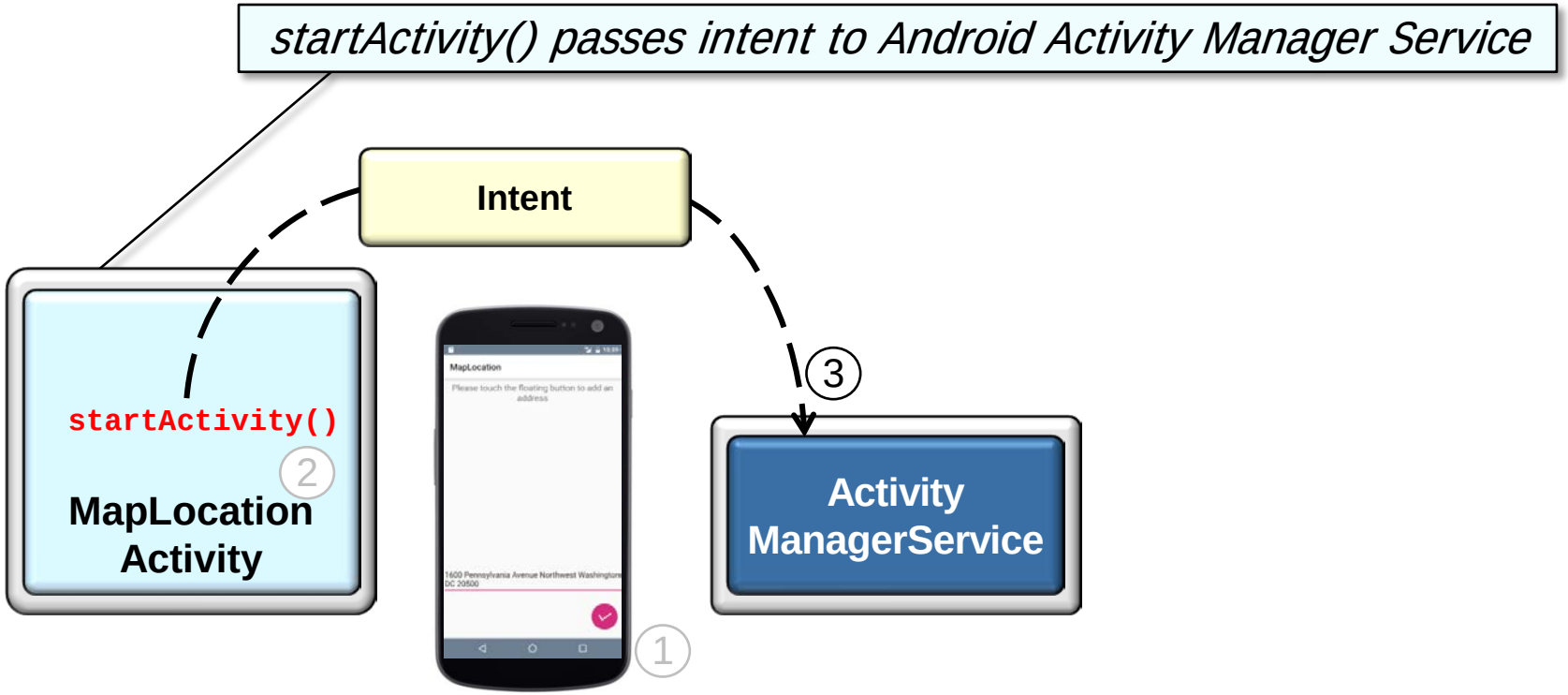


MapLocationActivity creates an implicit intent with ACTION_VIEW action & URL data:

```
new Intent
   (Intent.ACTION_VIEW,
    Uri.parse
      ("http://maps.google.com/?q="
       + address))
```

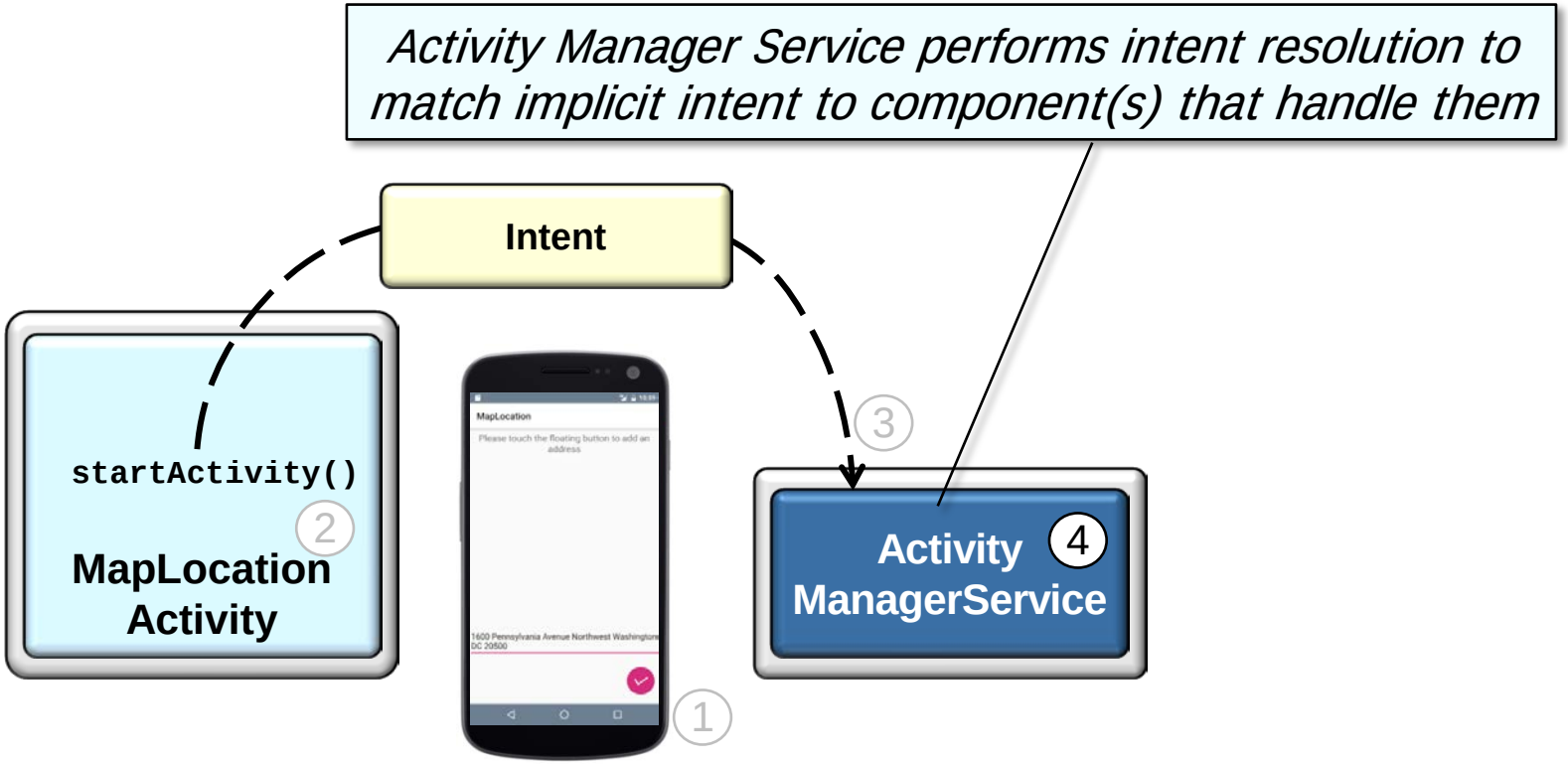
Starting an Android Activity

- An Activity can be launched on-demand via an intent passed to startActivity()



Starting an Android Activity

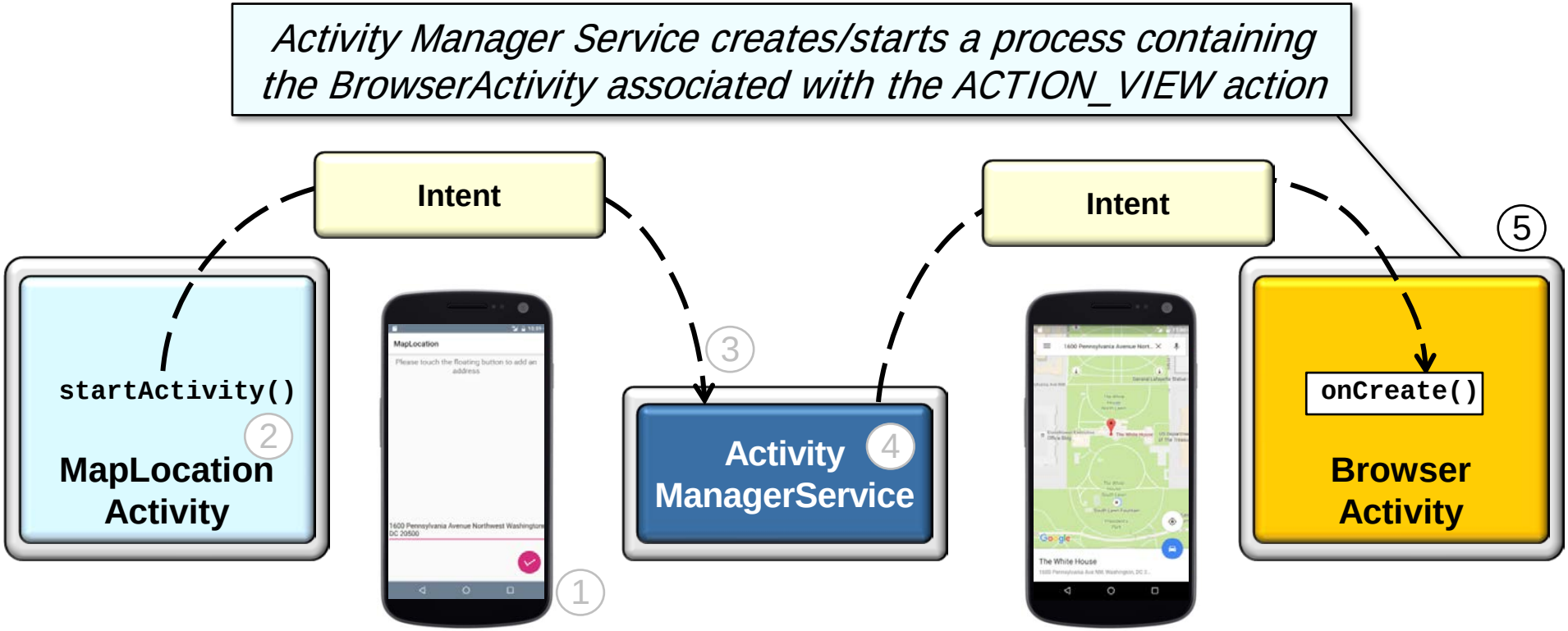
- An Activity can be launched on-demand via an intent passed to startActivity()



See developer.android.com/guide/components/intents-filters.html#Resolution

Starting an Android Activity

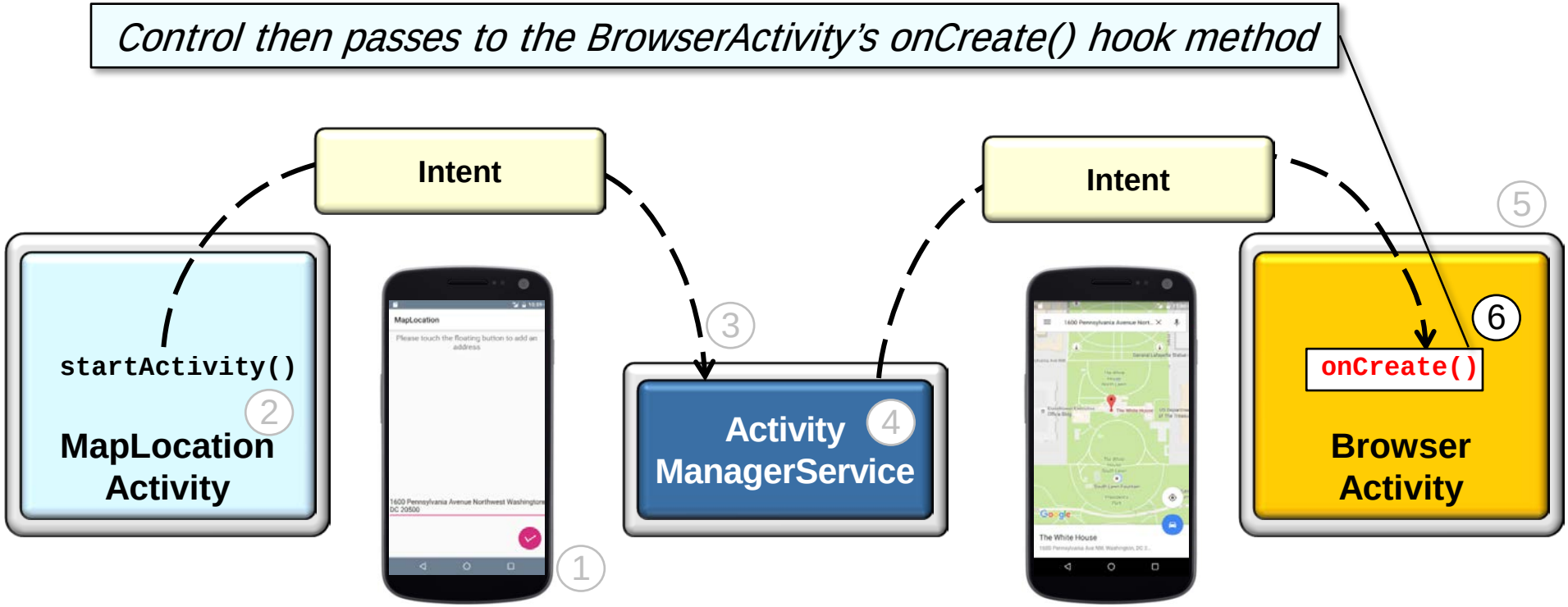
- An Activity can be launched on-demand via an intent passed to startActivity()



See coltf.blogspot.com/p/android-os-processes-and-zygote.html

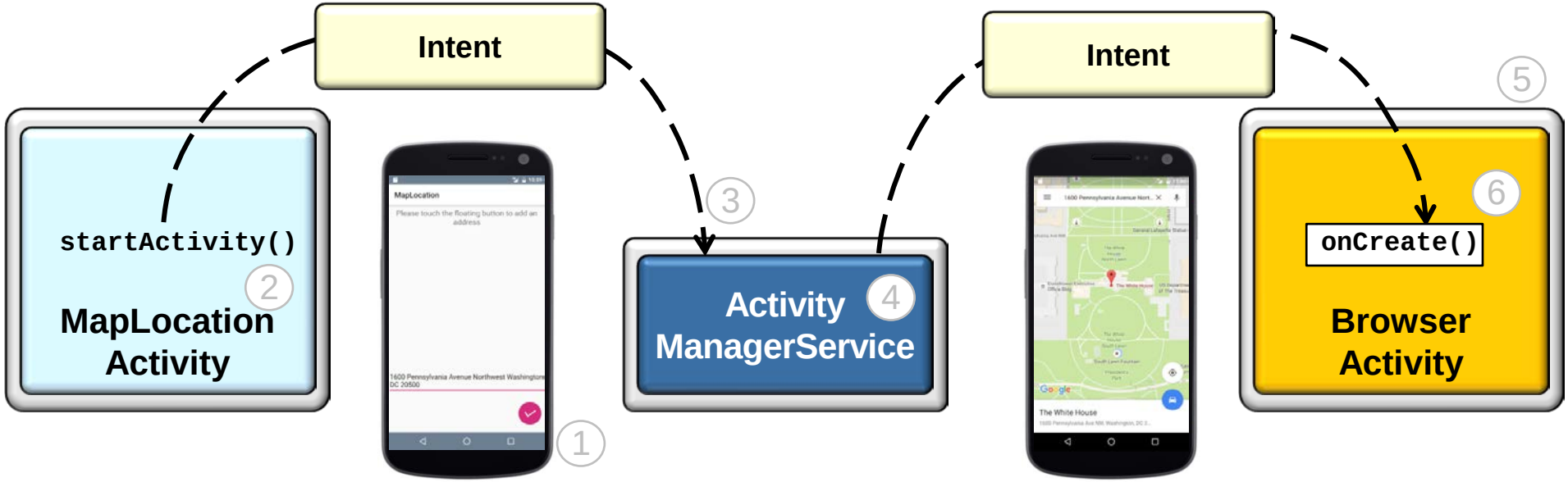
Starting an Android Activity

- An Activity can be launched on-demand via an intent passed to startActivity()



Starting an Android Activity

- An Activity can be launched on-demand via an intent passed to startActivity()



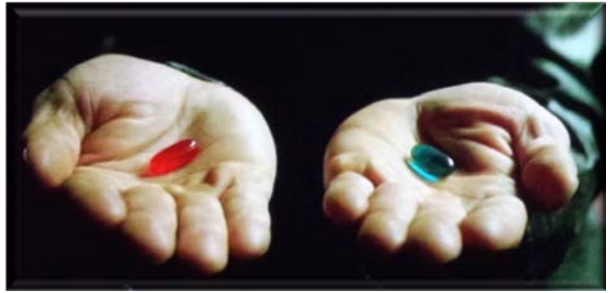
Most of these steps are invisible to app developers!

Methods Used to Start an Android Activity

Methods Used to Start an Android Activity

- Intent can be passed using one of two methods

void	startActivity(Intent intent) Launch a new activity.
void	startActivityForResult(Intent intent, int requestCode) Launch an activity for which you would like a result when it finished.

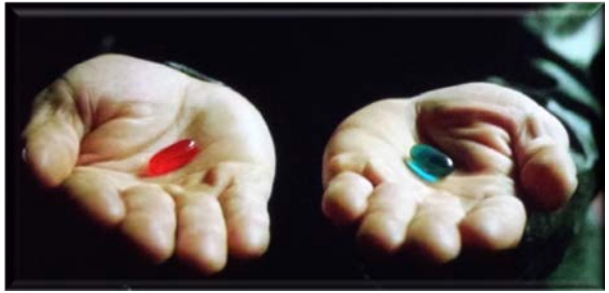


See developer.android.com/reference/android/app/Activity.html

Methods Used to Start an Android Activity

- Intent can be passed using one of two methods

void	startActivity(Intent intent) Launch a new activity.
void	startActivityForResult(Intent intent, int requestCode) Launch an activity for which you would like a result when it finished.

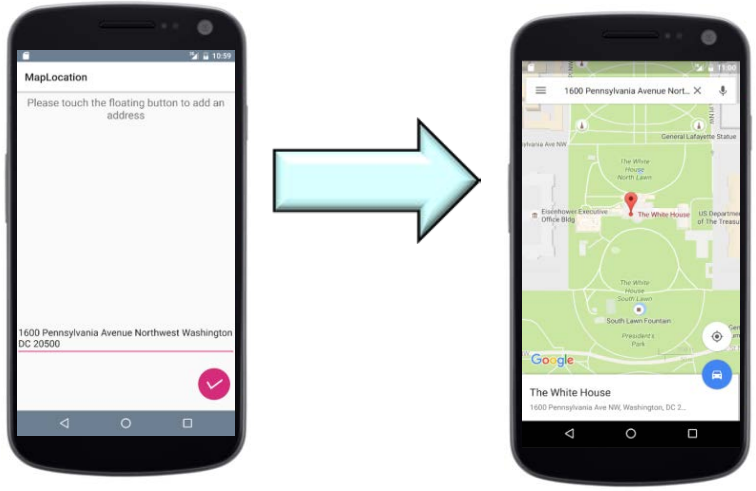


Which method to choose depends on whether or not a return result is needed from activity that is started

See developer.android.com/training/basics/intents/result.html

Methods Used to Start an Android Activity

- startActivity()
  - Activity$_1$ starts activity$_2$ with no result needed



activity$_1$ *activity$_2$*

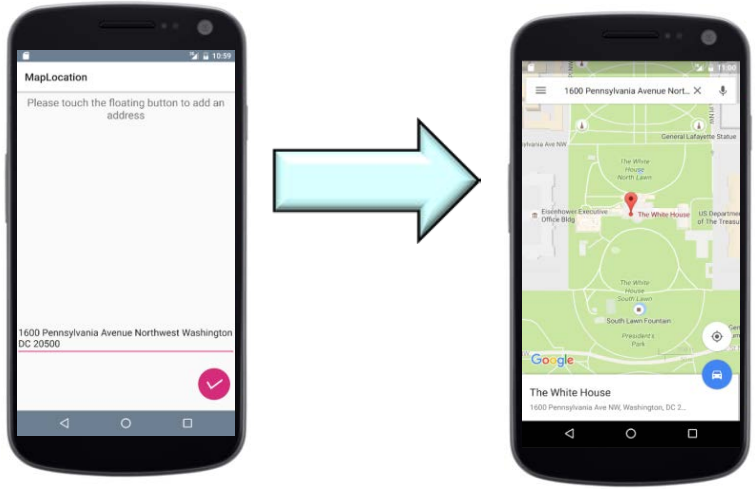
```
public void startMap() {
  ...
 final Intent geoIntent =
   makeGeoIntent(address);
 ...
 startActivity(intent);

...
```

See developer.android.com/reference/android/app/Activity.html#startActivity(android.content.Intent)

Methods Used to Start an Android Activity

- startActivity()
  - Activity$_1$ starts activity$_2$ with no result needed



activity$_1$ *activity$_2$*

```
public void startMap() {
  ...
 final Intent geoIntent =
   makeGeoIntent(address);
 ...
 startActivity(intent);

...
```

This method is asynchronous & one-way, i.e., it returns immediately & the caller continues executing its next operation(s)

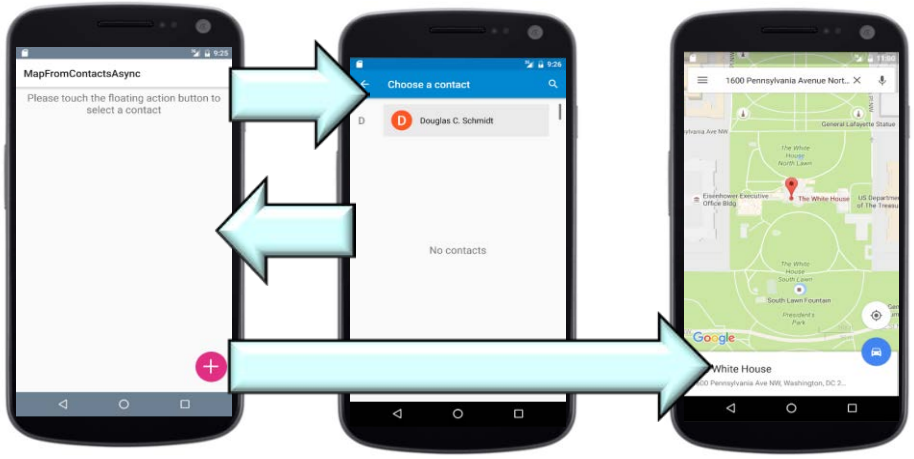
See en.wikipedia.org/wiki/Asynchronous_operation

Methods Used to Start an Android Activity

- startActivityForResult()
  - Activity$_1$ starts activity$_2$ & needs result

```
void startContactPicker() {
  Intent intent = new
    Intent(Intent.ACTION_PICK,
           ContactsContract.
             Contacts.
             CONTENT_URI);

  startActivityForResult
    (intent,
     PICK_CONTACT_REQUEST);
  ...
```



activity$_1$ *activity$_2$* *activity$_3$*

See developer.android.com/reference/android/app/Activity.html#startActivityForResult(android.content.Intent, int)

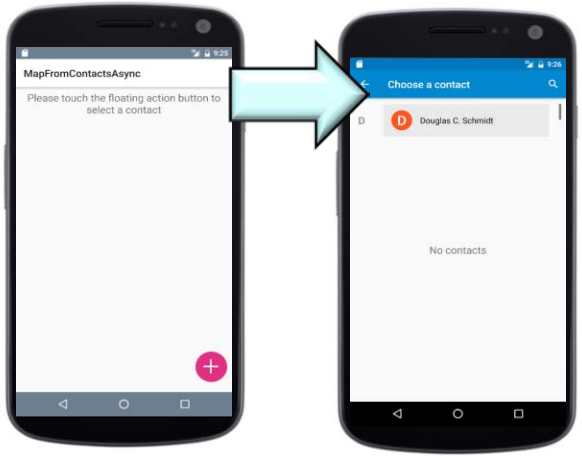
Methods Used to Start an Android Activity

- startActivityForResult()
  - Activity$_1$ starts activity$_2$ & needs result

This method is asynchronous, but two-way, i.e., a result is later returned via a callback

```
void startContactPicker() {
  Intent intent = new
    Intent(Intent.ACTION_PICK,
           ContactsContract.
             Contacts.
             CONTENT_URI);

startActivityForResult
  (intent,
   PICK_CONTACT_REQUEST);
...
```



activity$_1$ *activity$_2$*

See en.wikipedia.org/wiki/Callback_(computer_programming)

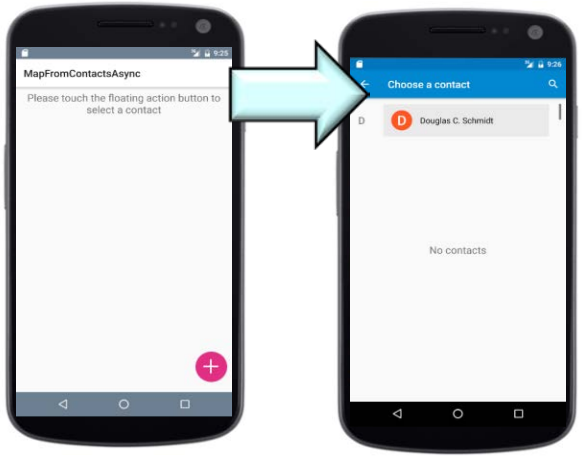
Methods Used to Start an Android Activity

- startActivityForResult()
  - Activity$_1$ starts activity$_2$ & needs result

```
void startContactPicker() {
  Intent intent = new
    Intent(Intent.ACTION_PICK,
           ContactsContract.
             Contacts.
             CONTENT_URI);

  startActivityForResult
    (intent,
     PICK_CONTACT_REQUEST);
  ...
```

This int value identifies request to activity$_2$ so activity$_1$ can handle the result properly



activity$_1$ *activity$_2$*

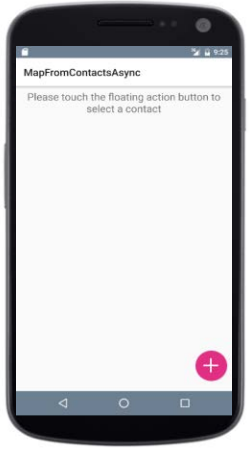
See developer.android.com/training/basics/intents/result.html#StartActivity

Methods Used to Start an Android Activity

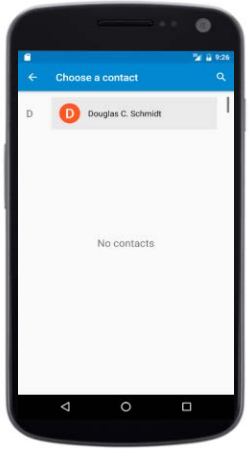
- startActivityForResult()
  - Activity$_1$ starts activity$_2$ & needs result

```
class ContactSelectionActivity
      extends ContactsActivity {
  ...
```

Activity$_2$ then runs until it finishes



activity$_1$



activity$_2$

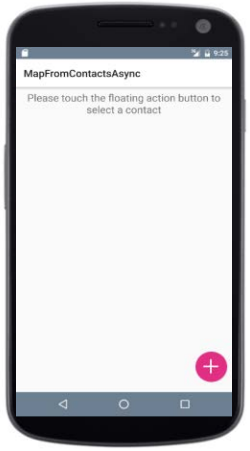
See packages/apps/Contacts/src/com/android/contacts/activities/ContactSelectionActivity.java

Methods Used to Start an Android Activity

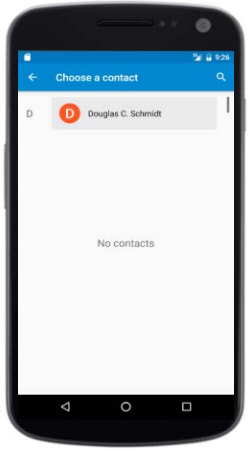
- startActivityForResult()
  - Activity$_1$ starts activity$_2$ & needs result

```
class ContactSelectionActivity
      extends ContactsActivity {
  ...
  public void returnPickerResult
      (Intent data) {
    ...
    setResult(RESULT_OK, data);
    ...
```

Set result code & data that activity$_2$ returns to its caller



activity$_1$



activity$_2$

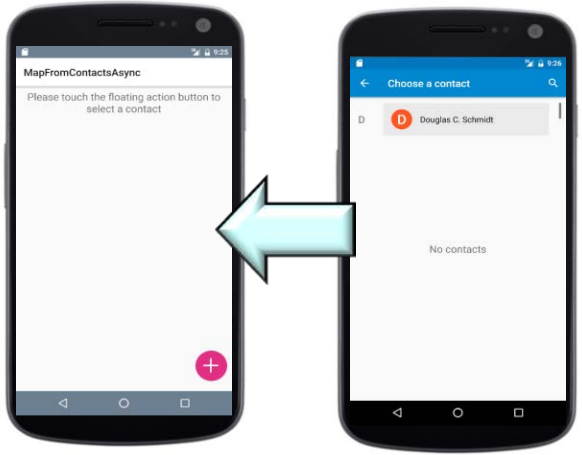
We'll discuss more about how to set the result of an activity shortly

Methods Used to Start an Android Activity

- startActivityForResult()
  - Activity$_1$ starts activity$_2$ & needs result
  - Result returned when activity$_2$'s done

This method is called back on activity$_1$

```
void onActivityResult
        (int requestCode,
         int resultCode,
         Intent data) {
  if (resultCode ==
      Activity.RESULT_OK
      && requestCode ==
      PICK_CONTACT_REQUEST)
    displayMap(data);
}  ...
```



activity$_1$ *activity$_2$*

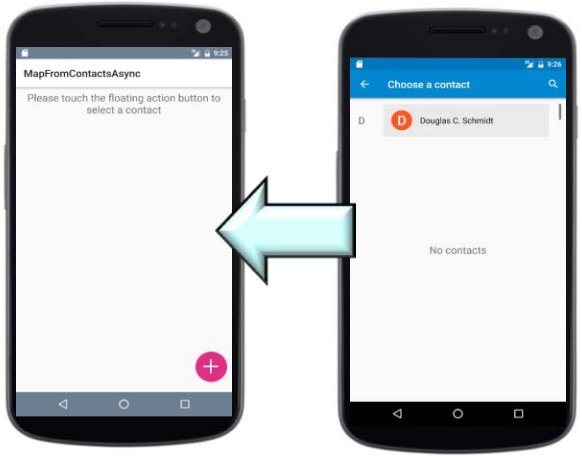
See developer.android.com/reference/android/app/Activity.html#onActivityResult(int, int, android.content.Intent)

Methods Used to Start an Android Activity

- startActivityForResult()
  - Activity$_1$ starts activity$_2$ & needs result
  - Result returned when activity$_2$'s done

```
void onActivityResult
                (int requestCode,
                 int resultCode,
                 Intent data) {
  if (resultCode ==
      Activity.RESULT_OK
      && requestCode ==
      PICK_CONTACT_REQUEST)
    displayMap(data);
}  ...
```

The request code matches the request with the result



activity$_1$ *activity$_2$*

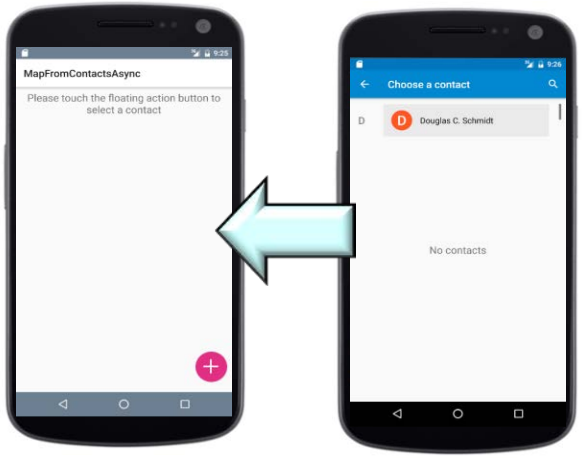
See developer.android.com/training/basics/intents/result.html#ReceiveResult

Methods Used to Start an Android Activity

- startActivityForResult()
  - Activity$_1$ starts activity$_2$ & needs result
  - Result returned when activity$_2$'s done

```
void onActivityResult
                (int requestCode,
                 int resultCode,
                 Intent data) {
  if (resultCode ==
      Activity.RESULT_OK
      && requestCode ==
      PICK_CONTACT_REQUEST)
    displayMap(data);
  }  ...
```

The result code & data convey what happened in activity$_2$



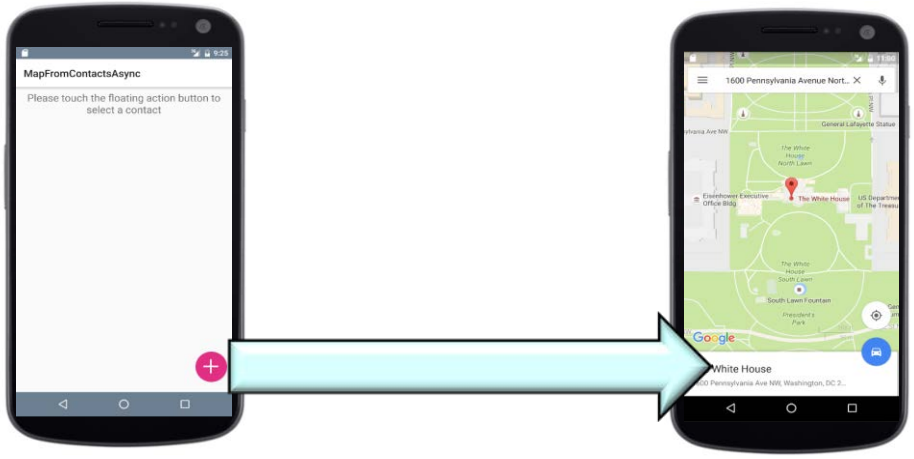
activity$_1$ *activity$_2$*

See developer.android.com/reference/android/app/Activity.html#onActivityResult(int, int, android.content.Intent)

Methods Used to Start an Android Activity

- startActivityForResult()
  - Activity$_1$ starts activity$_2$ & needs result
  - Result returned when activity$_2$'s done
  - The result data can then be used to start activity$_3$

```
void onActivityResult
                (int requestCode,
                 int resultCode,
                 Intent data) {
  if (resultCode ==
      Activity.RESULT_OK
      && requestCode ==
      PICK_CONTACT_REQUEST)
    displayMap(data);
}  ...
```



activity$_1$ *activity$_3$*

This method starts activity$_3$:

```
startActivity(geoIntent);
```

or

```
startActivity(browserIntent);
```

The selected intent/activity depends on the configuration of the Android device

Finishing an Activity & Returning a Result

Finishing an Activity & Returning a Result

- When an activity is done it can set its result by calling setResult()

```
class ContactSelectionActivity
      extends ContactsActivity {
  ...
  public void returnPickerResult
      (Uri data) {
    returnPickerResult(new
      Intent().setData(data);
  }

  public void returnPickerResult
      (Intent in) {
    ...
    setResult(RESULT_OK, in);
    finish();
  ...
```

See packages/apps/Contacts/src/com/android/contacts/activities/ContactSelectionActivity.java

Finishing an Activity & Returning a Result

- When an activity is done it can set its result by calling setResult()

```
class ContactSelectionActivity
      extends ContactsActivity {
  ...
  public void returnPickerResult
      (Uri data) {
    returnPickerResult(new
      Intent().setData(data);
  }

  public void returnPickerResult
      (Intent in) {
    ...
    setResult(RESULT_OK, in);
    finish();
  ...
```

Set result that an activity will return to its caller

See developer.android.com/reference/android/app/Activity.html#setResult(int, android.content.Intent)

Finishing an Activity & Returning a Result

- When an activity is done it can set its result by calling setResult()
  - resultCode can be
    - RESULT_OK
      - Indicates the operation succeeded

```
class ContactSelectionActivity
      extends ContactsActivity {
  ...
  public void returnPickerResult
      (Uri data) {
    returnPickerResult(new
      Intent().setData(data);
  }

  public void returnPickerResult
      (Intent in) {
    ...
    setResult(RESULT_OK, in);
    finish();
  ...
```

Finishing an Activity & Returning a Result

- When an activity is done it can set its result by calling setResult()
  - resultCode can be
    - RESULT_OK
    - RESULT_CANCELED
      - Indicates the operation was cancelled

```
class ContactSelectionActivity
      extends ContactsActivity {
  ...
  void onCreate(Bundle bundle) {
    ...
    mRequest = mIntentResolver.
      resolveIntent(getIntent());
    if (!mRequest.isValid()) {
      setResult
         (RESULT_CANCELED);
      ...
      return;
    }
    ...
```

Finishing an Activity & Returning a Result

- When an activity is done it can set its result by calling setResult()
  - resultCode can be
    - RESULT_OK
    - RESULT_CANCELED
    - RESULT_FIRST_USER
      - Indicates start of user-defined result codes

```
class RecognizerIntent {
  ...
  public static final int
    RESULT_NO_MATCH =
    Activity.RESULT_FIRST_USER;
  public static final int
    RESULT_CLIENT_ERROR =
    Activity.RESULT_FIRST_USER + 1;
  public static final int
    RESULT_SERVER_ERROR =
    Activity.RESULT_FIRST_USER + 2;
  public static final int
    RESULT_NETWORK_ERROR =
    Activity.RESULT_FIRST_USER + 3;
  ...
```

See frameworks/base/core/java/android/speech/RecognizerIntent.java

Finishing an Activity & Returning a Result

- Call the finish() method when an activity is done to close it down

```
class ContactSelectionActivity
      extends ContactsActivity {
  ...
  public void returnPickerResult
      (Uri data) {
    returnPickerResult(new
      Intent().setData(data);
  }

  public void returnPickerResult
      (Intent in) {
    ...
    setResult(RESULT_OK, in);
    finish();
  ...
```

Propagate the result back to launcher of this activity

See developer.android.com/reference/android/app/Activity.html#finish()

End of Activity Lifecycle Operations (Part 1)