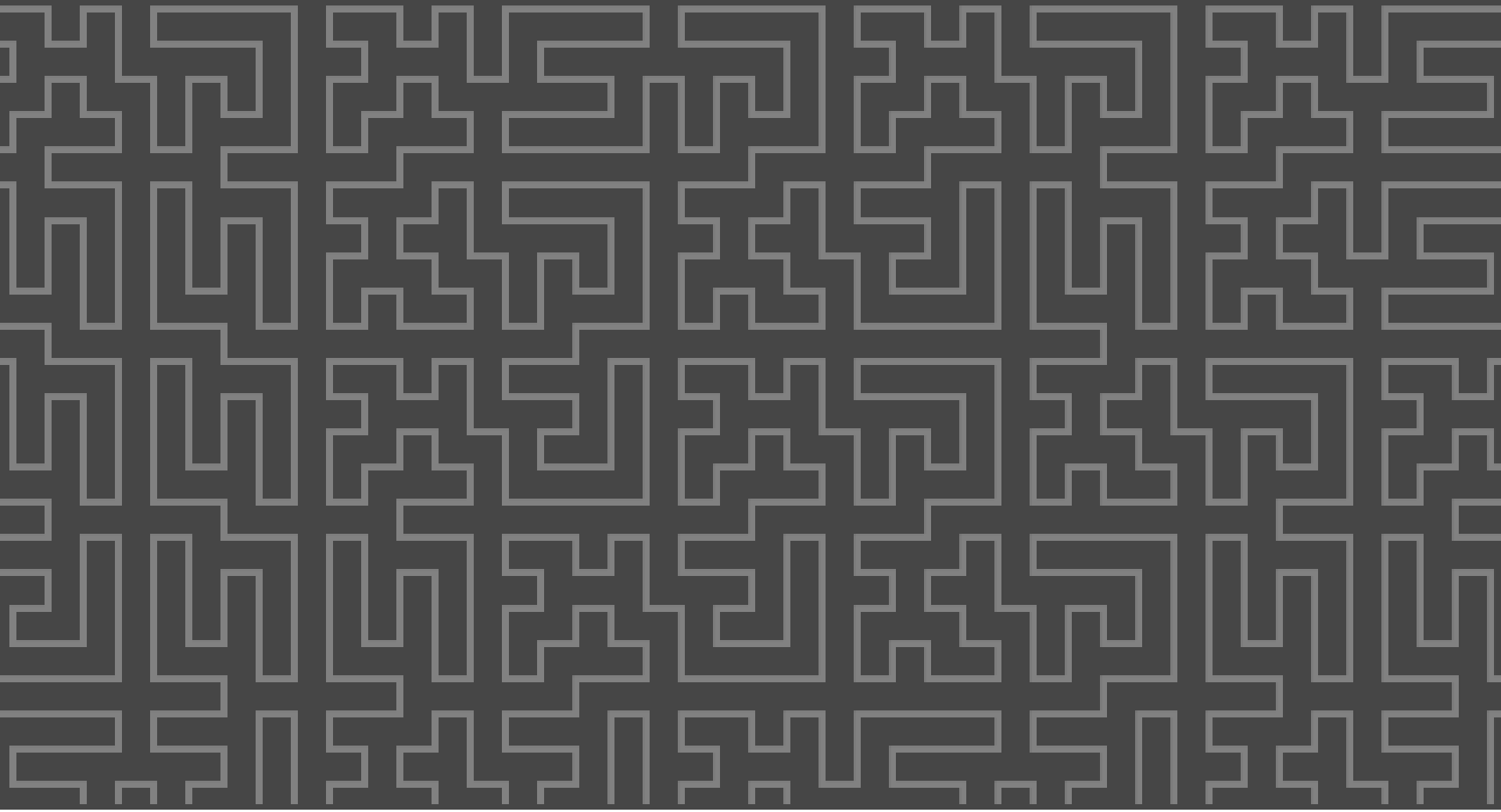
# MOBILE DEVELOPMENT



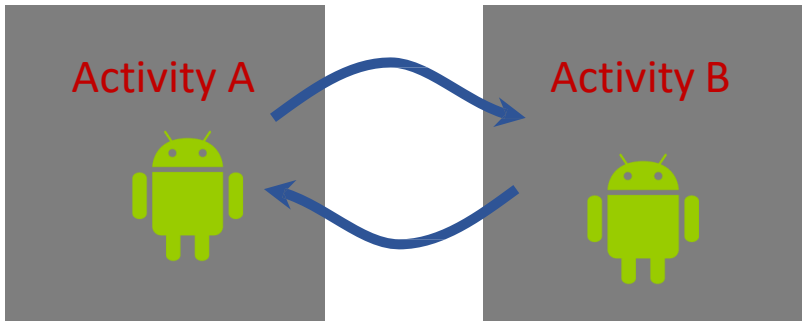
## Android Messages and Intents

# Content

- Creating new Activity classes
- Explicit Intents
- Implicit Intents
- Recycler View
- Readings

# Multiple Activities

- Application normally have multiple activities
- Activity A can launch Activity B in response of an event
- A can pass data to B
- B can send back the data to A when it is done



# Adding a new Activity class

- Use File -> New -> Activity -> Basic Activity
- Android Studio will create:
  - a new activity (.java and .xml file)
  - entries in the manifest file

```
<activity android:name=".SimpleActivity"
  android:label="@string/app_name" >

  <intent-filter></intent-filter>
</activity>
```

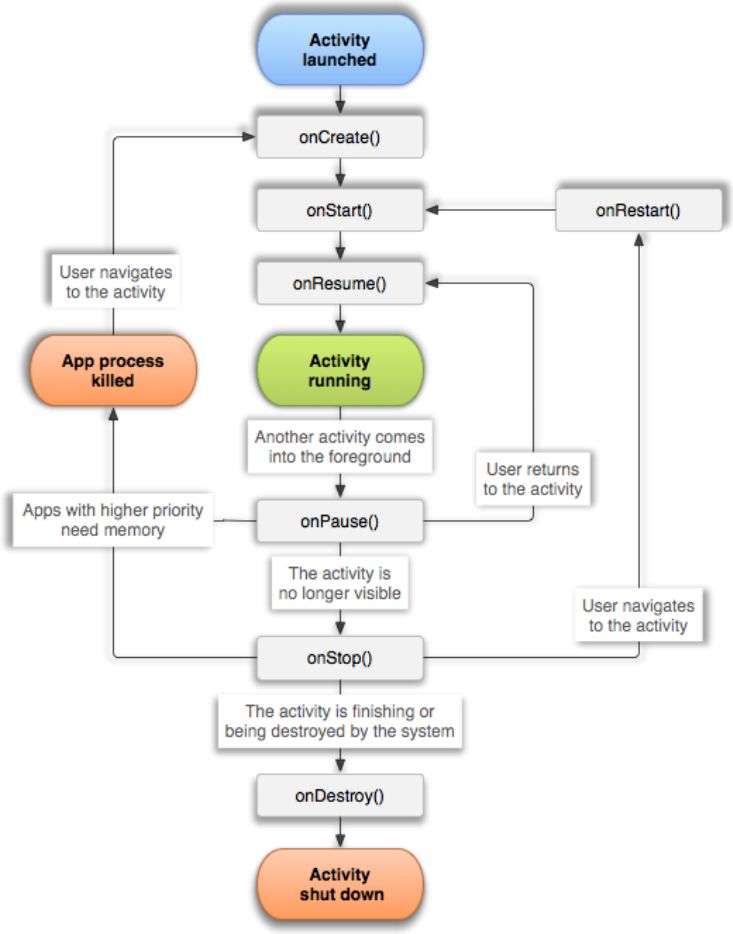
# Adding a new Activity class

- Create a new XML layout file with a unique file name in your res/layout folder
- Create a link in your code through the R class to your new layout file
- Android Studio does all of this for us now

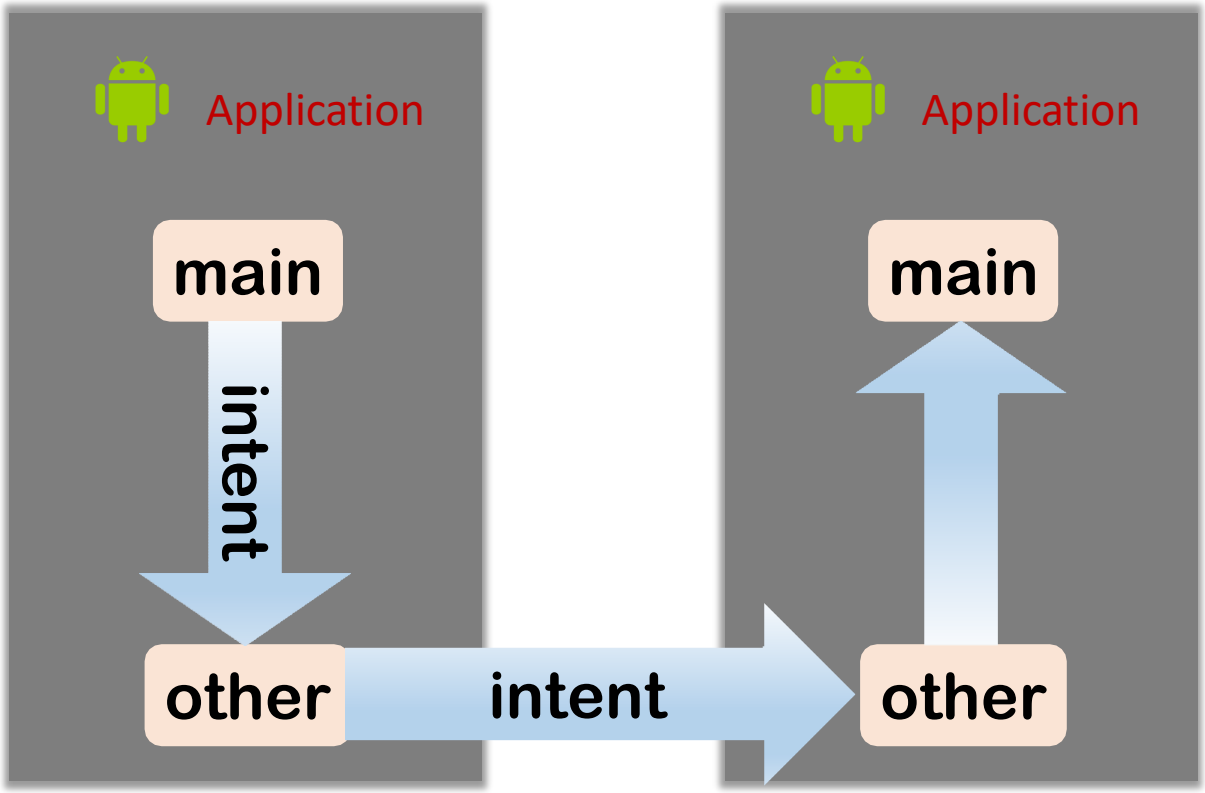
```
@Override
protected void onCreate(Bundle savedInstanceState)
{
   super.onCreate(savedInstanceState);
   setContentView(R.layout.simple_activity);
}
```

# Activity Lifecycle

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



# Intent


Application
main
intent
other
intent
Application
main
other

# Explicit Intents

- Define message for specific Activity class
- Android infrastructure routes message behind the scenes to the right class
- Only require an <activity .../> code block
- No intent filter or action and category code blocks required
- Android Studio already defines the activity for us

```
<activity android:name=".SimpleActivity"
android:label="@string/app_name" />
```

# Explicit Intents

- Define an intent and set the from/to activity classes as below

```
Intent intent = new Intent();
intent.setClass(MainActivity.this,
SecondActivity.class);

startActivity(intent);
```

# Implicit Intents

- No specific target Activity
- Android matches Intent actions with Activities using *Intent filters*
- Simply set the action, and send the message
- Android will then send the message to the appropriate Activity

```
Intent message = new Intent();
message.setAction("com.jcasey.OPEN_SIMPLE_ACTIVITY");
startActivity(message);
```

# Implicit Intents

- Can define custom actions or use pre-defined actions from the android.content.Intent class
- Custom actions need to defined using your app's package name
- Need to define a filter-intent that matches the action called

```
<activity
 android:name=".SimpleActivity"
 android:label="@string/app_name" >

    <intent-filter>
    <action android:name="com.jcasey.OPEN_SIMPLE_ACTIVITY"/>
    <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

# Implicit Intents

- For built in Actions we do not need to create our own intent-filter
- If action matches more than one intent-filter then Android will ask you which application you would like to use to process the action

```
Intent message = new Intent();
message.setAction(Intent.ACTION_VIEW);
message.setDataAndType(Uri.parse("http://www.google.co.nz/"), "text/html");

startActivity(message);
```

# Message Passing

- Define message, set context and destination
- Set data payload using putExtra(key,value) method

```
Intent message = new Intent();

message.setClass(Week4Activity.this, SimpleActivity.class);

// add some key-value pairs to our message
message.putExtra("clientId", "jcasey");
message.putExtra("firstName","John");
message.putExtra("lastName","Casey");

startActivity(message);
```

# Message Passing

- Use getIntent() to get at the Intent which started your Activity
- Get the extra data and then use the various type casted get methods to get at your data

```
Bundle extras = getIntent().getExtras();
String clientId =extras.getString("clientId");
String firstName = extras.getString("firstName");
String lastName = extras.getString("lastName");
```

# Message Passing

- Use the finish() method to close current activity – returning to the previous activity in the stack
- If you want to return a value when you have finished calling the new Activity use the *startActivityForResult(message, REQ_CODE)* method

```
Intent intent = new Intent();
intent.putExtra("RESULT", "Return message");
setResult(Activity.RESULT_OK, intent);
finish();
```

# Message Passing

- Send another Intent message back
- Process message using the onActivityResult() method

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

      // also check the requestCode ...

      if(resultCode == Activity.RESULT_OK)
      {
          Bundle extras = message.getExtras();
          String result = extras.getString("RESULT");
      }
}
```

# References

- https://developer.android.com/training/camera/photobasics.html
- https://developer.android.com/guide/components/activities/intro-activities
- https://developer.android.com/reference/android/content/Intent