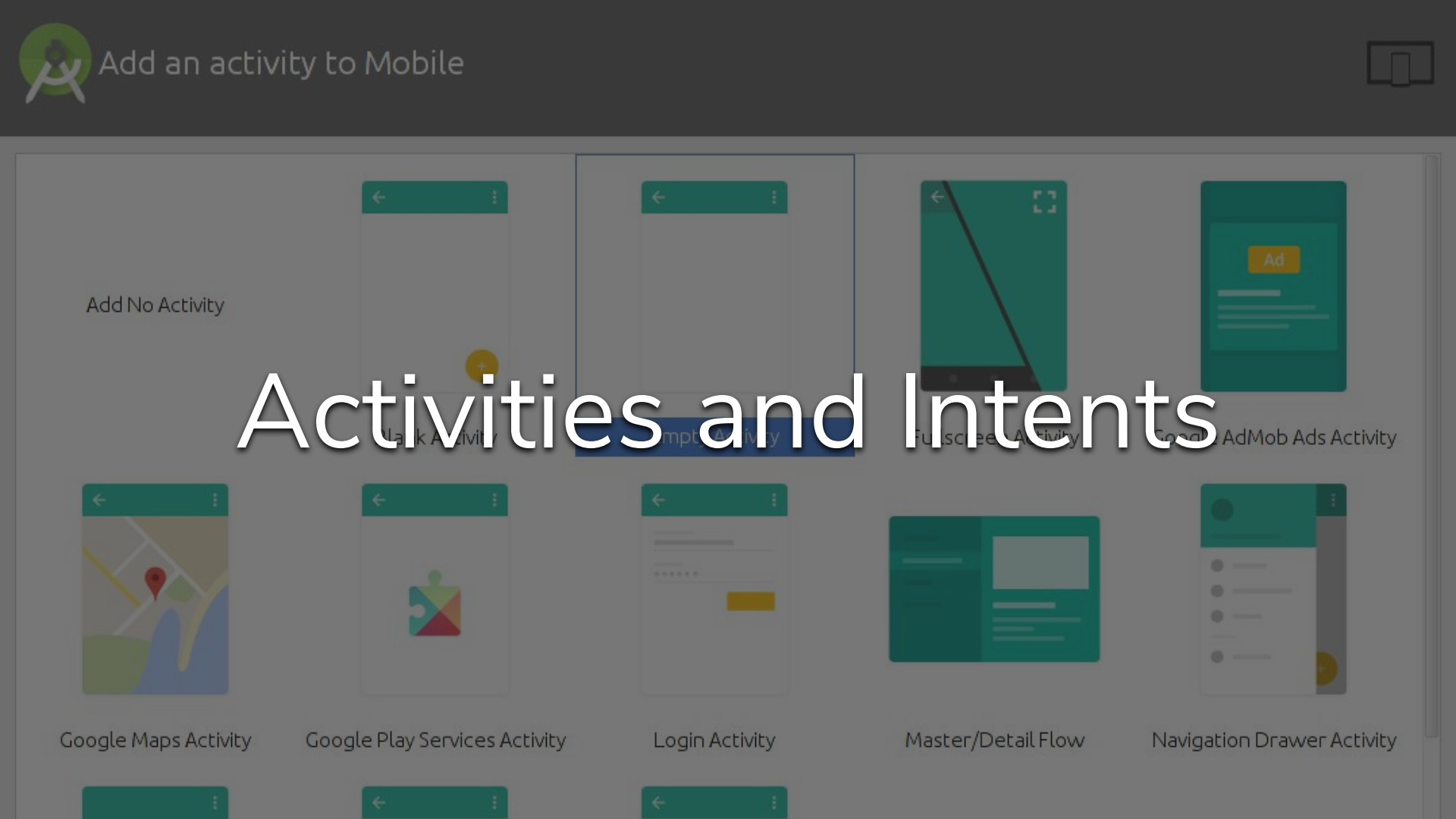
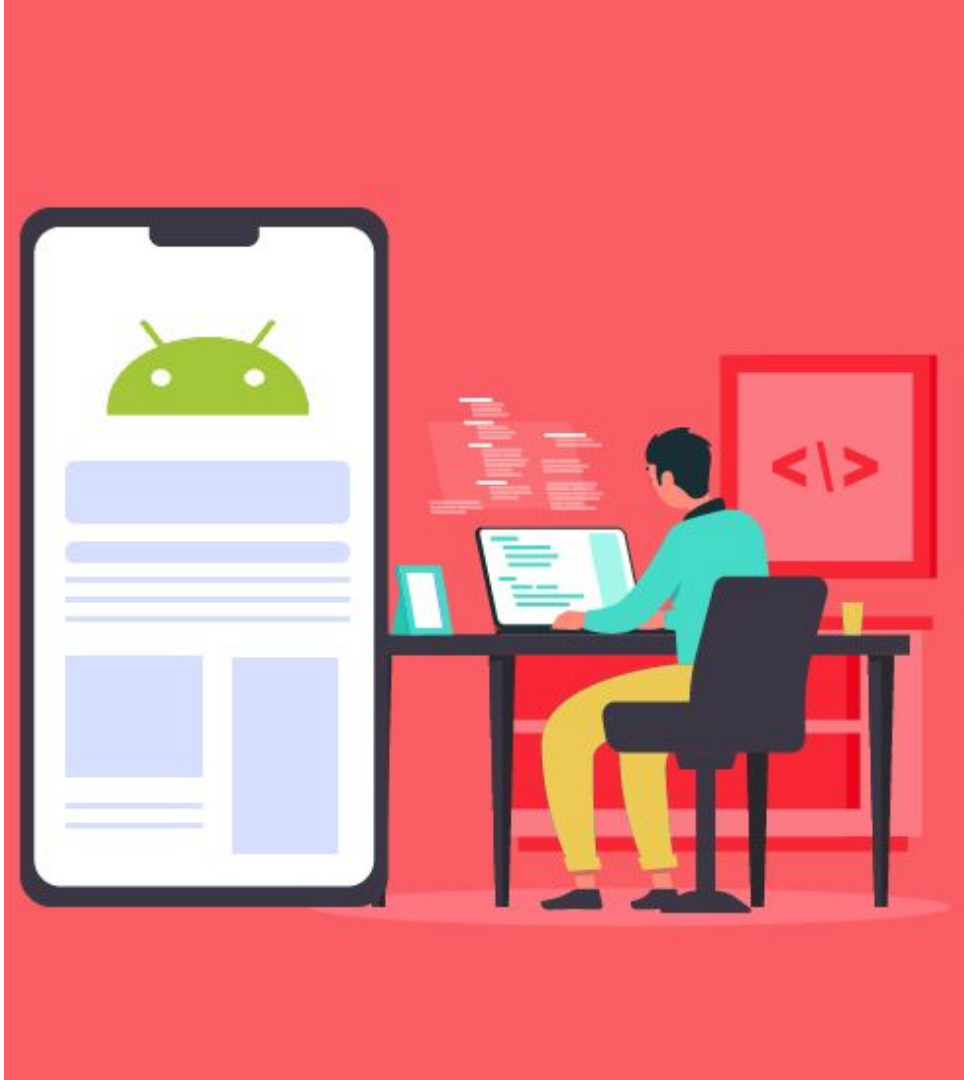


Activities and Intents

Overview

- Android Components
  - Activities
  - Services
  - Broadcast Receivers
  - Content Providers
- Intents
  - Explicit
  - Implicit
  - Filters



Android Components

Android Components - Overview

Activities

- An activity is the entry point for interacting with the user
- Ex. anything with a UI, like checking your grades in the canvas app

Services

- A service is a general-purpose entry point for keeping an app running in the background for all kinds of reasons
- Ex. Spotify music playing without having the app ui open

Broadcast Receivers

- A broadcast receiver is a component that enables the system to deliver events to the app outside of a regular user flow, allowing the app to respond to system-wide broadcast announcements
- Ex. Notification Messages

Content Providers

- A content provider manages a shared set of app data that you can store in the file system, in a SQLite database, on the web, or on any other persistent storage location that your app can access
- Ex. Retrieving Receipt from Doordash

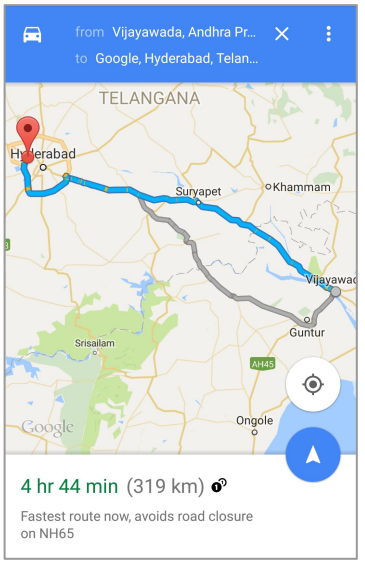
What is an Activity?

```
class MainActivity : AppCompatActivity() {
    ...
}
```
MainActivity.kt

- An **activity** is a single focused thing your user can do. If you chain multiple activities together to do something more complex, it's called a **task**.
- Activities are arranged in a **stack** (LIFO)
- Activity class creates a window for its UI
- Has a life cycle

Examples

- Getting Directions
- User Interactions, such as button clicks, can start other activities in the same or other apps



Multiple Screens

- Apps can be more than one screen, which can mean multiple activities.
- Tasks are broken down into Activities to make UX organized

Common examples

- Single item details (ie: Amazon shopping cart - Item description & price)
- New item creation (ie: Reddit - Post creation)
- App settings (ie: Default Settings App - Toggling/viewing settings)
- Access to services in other apps (ie: TikTok - Access to camera)

What is a Service?

```
class MyService : Service() {
    ...
}
```
MyService.kt

- Application component performing long-running operations in background
- Does not provide a user interface
- To create a service, you must create a subclass of Service

Three Types

- Foreground Service
  - Performs some operation that is noticeable to the user
- Background Service
  - Performs an operation that isn't directly noticed by the user
- Bound Service
  - Offers client-server interface allowing components to interact with the service, send requests, receive results, and do so across interprocess communication (IPC) processes

What is a Broadcast Receiver?

```
class MyReceiver : BroadcastReceiver() {
       override fun onReceive(context: Context,
       intent: Intent) {
       ...
       }
}
```

MyReceiver.kt

- Apps can receive broadcasts in two ways:
  - Manifest-declared receivers
  - Context-registered receivers
- The state of BroadcastReceiver (whether it is running or not) affects the state of its containing process, and can affect the likelihood of the system killing it.
- Android provide three ways for apps to send broadcast:
  - **sendOrderedBroadcast(Intent, String)** method sends broadcasts to one receiver at a time
  - **sendBroadcast(Intent)** method sends broadcasts to all receivers in an undefined order
  - **LocalBroadcastManager.sendBroadcast** method sends broadcasts to receivers that are in the same app as the sender

What is a Content Provider?

```
class MyContentProvider : ContentProvider() {
     override fun onCreate(): Boolean {
     ...
     }
}
```
MyContentProvider.kt

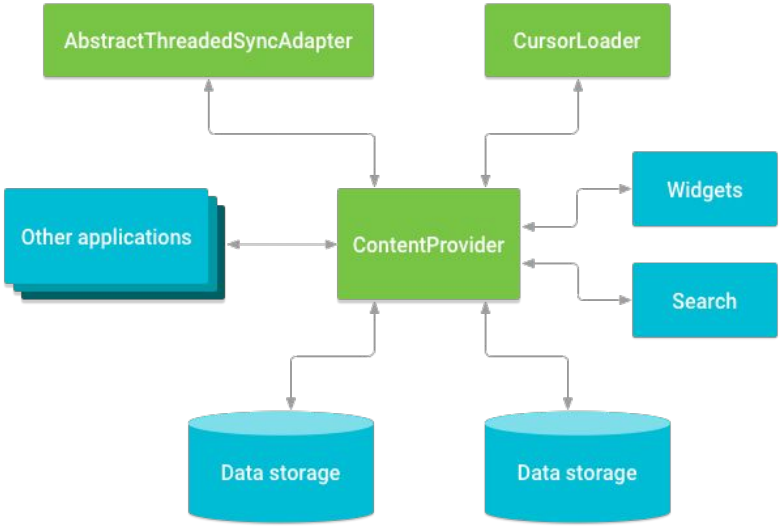
- Presents data to external applications as one or more tables that are similar to the tables found in a relational database
- Manages access to a central repository of data
- Coordinates access to the data storage layer in your app
    - Sharing access to your app data with other apps
    - Sending data to a widget
    - Recieve search suggestions through a search framework
    - Loading data in your UI

More on Content Provider

One of the built-in providers in the Android platform is the user dictionary, which stores the spellings of non-standard words that the user wants to keep

word	app id	frequency	locale	_ID
mapreduce	user1	100	en_US	1
precompiler	user14	200	fr_FR	2
applet	user2	225	fr_CA	3
const	user1	255	pt_BR	4
int	user5	100	en_UK	5

A content provider coordinates access to the data storage layer in your application for a number of different APIs and components



Activities in Relation to Apps and Layouts

Apps

- Activities are loosely tied together to make up an app
- First Activity user sees is typically called “main activity”

Android Manifest

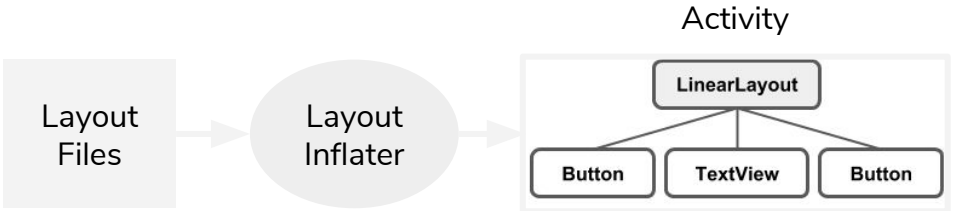
- Activities can be organized in parent-child relationships in the Android Manifest to aid navigation
- All activities are connected to the Android Manifest

Layouts

- An Activity typically has a UI layout
- A Layout is usually defined in one or more XML files
- Activity “inflates” layout as part of being created

Layout Inflater

- Parses layout XML into a View Hierarchy that the Activity understands



How to Create an Activity

1) Define Activity (Kotlin) and Layout (XML)
   a) Connect Layout to Activity
2) Declare Activity in Android Manifest

Activity	Layout	Manifest
NewActivity.kt	activity_new.xml	AndroidManifest.xml

How to Create an Activity

1) **Define Activity (Kotlin) and Layout (XML)**
   a) **Connect Layout to Activity**
2) Declare Activity in Android Manifest

```kotlin
class NewActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_new)
    }
}
```

NewActivity.kt

```xml
<?xml version="1.0" encoding="utf-8"?>
<ConstraintLayout ... >

    <!-- other views -->

</ConstraintLayout>
```

activity_new.xml

How to Create an Activity

1) Define Activity (Kotlin) and Layout (XML)
   a) Connect Layout to Activity
2) **Declare Activity in Android Manifest**

```
<manifest ... >
  <application ...>
    <activity android:name=".NewActivity"></activity>
    <activity android:name=”.MainActivity”>
      <intent-filter>
        <action android:name=”android.intent.action.MAIN” />
        <action android:name=”android.intent.category.LAUNCHER” />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

AndroidManifest.xml

Intents

What are Intents?

Intents request actions from other app components like other Activities

- Typically used for transitioning to other activities
  - Ex . startActivity()

Two Core Pieces

- The action to perform
  - ACTION_VIEW
  - ACTION_EDIT
  - ACTION_MAIN
  - more!
- The data to operate on
  - Ex. Record of friends in SnapChat

Other use cases

- Start the service
- Launch an activity
- Display a web page
- Display a list of contacts
- Broadcast a message
- Dial a phone call etc.

Two types

- Implicit
- Explicit

Explicit vs Implicit Intents

Explicit Intent

- Specify the **exact Android Component** you want to run
- Navigates internally to an Activity in App or specific third party app

Examples

- I **only want** to go to Lazi Cow for black milk tea with boba, 50% sweet, no ice
- Main Activity starts the ShoppingCart Activity

Implicit Intent

- Generic action done **without caring which activity** does it as long as can match request criteria
- Done by connecting data type & action to known existing components

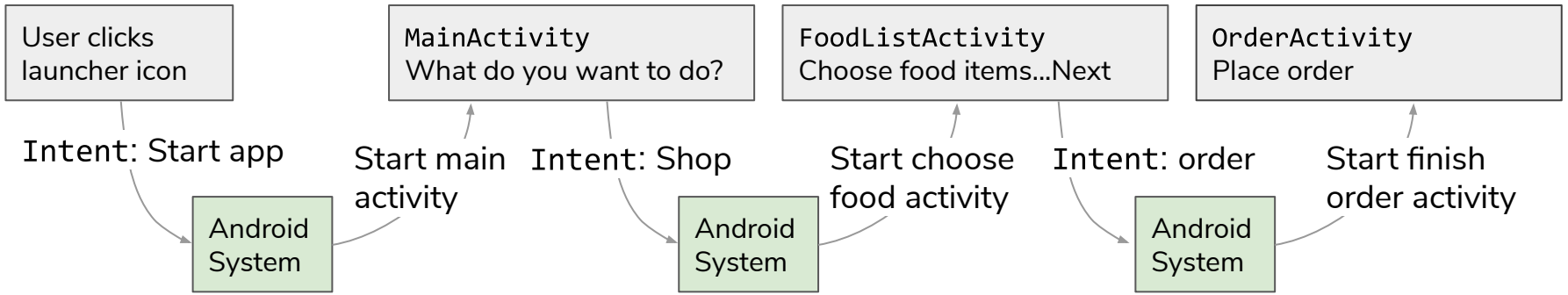
Examples

- I want to open up a Map, but **I don't care** if it is Google Maps or Apple Maps
- Clicking Share opens a chooser with a list of apps

Explicit Intents and how they work

Launch an Activity

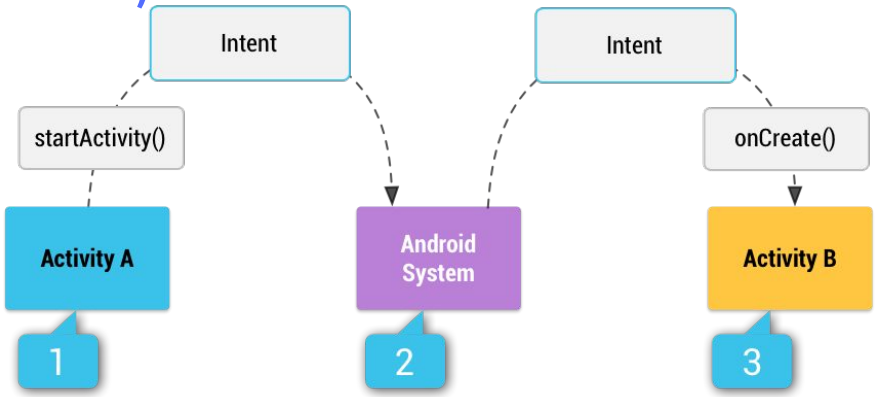
- Create an intent to launch another Activity
- The intent is sent to Android Runtime to launch the new activity
  - All Activities are managed by Android Runtime

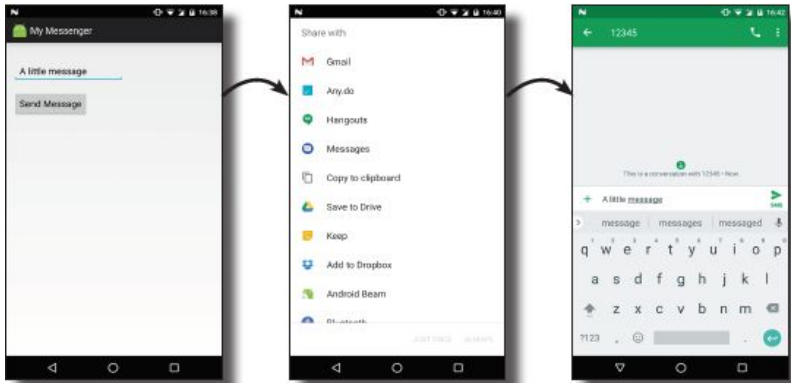


Implicit Intents and how they work

1) The Android Runtime keeps a list of registered Apps
2) Apps have to register via AndroidManifest.xml
3) Runtime receives the request and looks for matches
4) Android runtime uses Intent filters for matching
5) If more than one match, shows a list of possible matches and lets the user choose one
6) Android runtime starts the requested activity

Example (Implicit)





Coding Intents - Explicit vs Implicit

Explicit Intent

Navigate between activities in your app

```
fun viewNoteDetail() {
   val intent = Intent(this, NoteDetailActivity::class.java)
   intent.putExtra(NOTE_ID, note.id)
   startActivity(intent)
}
```

Navigate to a specific external app

```
fun openExternalApp() {
   val intent = Intent("com.example.workapp.FILE_OPEN")
   if (intent.resolveActivity(getPackageManager()) != null) {
       startActivity(intent)
   }
}
```

Coding Intents - Explicit vs Implicit

Implicit Intent

```
fun sendEmail() {
    val intent = Intent(Intent.ACTION_SEND)
    intent.setType("text/plain")
    intent.putExtra(Intent.EXTRA_EMAIL, emailAddresses)
    intent.putExtra(Intent.EXTRA_TEXT, "How are you?")

    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent)
    }
}
```

Using Intents to Send and Receive Data Forward

2 Ways of Sending Data

- **Data:** one piece of information whose data location can be represented by an URI
- **Extras:** one or more pieces of information as a collection of key-value pairs in a bundle

Sender Activity

- Create the Intent object
- Put data or extras into that Intent
- Start the new Activity with startActivity()

```
val intent = Intent(this, ReceiverActivity::class.java)
intent.putExtra("courseName", "ECS 198F")
startActivity(intent)
```

Receiver Activity

- Get the Intent Object that started the Activity
- Retrieve the data or extras from the Intent object

```
var courseName: String = intent.getStringExtra("courseName")
```

Sending Objects with Parcelable

- By default putExtra cannot send objects, but often we need to
- Java can serialize objects to send them in putExtra, but its slow
- Parcelable is an Android specific serialization to send objects across activities

Implement Parcelable

```
@Parcelize
data class Name(
    val first: String,
    val last: String
) : Parcelable
```

Start Intent in First Activity

```
val intent = Intent(this, SecondActivity::class.java)
intent.putExtra("Name", name)
startActivity(intent)
```

Collective Values in Second Activity

```
val name : Name = intent.getParcelableExtra<Name>("Name")!!
```

Using Intents to Send Data Backward

- Starting Activity: Switch to another activity
  - Specify that you are starting another activity expecting a return value
  - **startActivityForResult(intent, requestCode);**

- Second Activity: Send data back to Starting Activity
  - Create an Intent Object
  - Put return value into the intent using extras
  - Set result to Activity.RESULT_OK or RESULT_CANCELED, if the user cancelled out
  - Close the activity by calling finish()

- Starting Activity: Handling returned data
  - Implement onActivityResult() in first Activity

Intent Filters

- Advertise which implicit intents your app can receive
- Declared and specified in each component in the manifest file of an app

```
...
<activity android:name="ShareActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
</activity>
...
```

AndroidManifest.xml

Elements that Describe Intent Filters

You can create a filter that includes more than one instance of `<action>`, `<data>`, or `<category>`. If you do, you need to be certain that the component can handle any and all combinations of those filter elements.

`<action>`	Declares the intent action accepted, in the `name` attribute. The value must be the literal string value of an action, not the class constant.
`<data>`	Declares the type of data accepted, using one or more attributes that specify various aspects of the data URI (`scheme` , `host` , `port` , `path`) and MIME type.
`<category>`	Declares the intent category accepted, in the `name` attribute. The value must be the literal string value of an action, not the class constant.

Example

```
<activity android:name="MainActivity">
    <!-- This activity is the main entry, should appear in app launcher -->
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

<activity android:name="ShareActivity">
    <!-- This activity handles "SEND" actions with text data -->
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
    <!-- This activity also handles "SEND" and "SEND_MULTIPLE" with media data -->
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <action android:name="android.intent.action.SEND_MULTIPLE"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="application/vnd.google.panorama360+jpg"/>
        <data android:mimeType="image/*"/>
        <data android:mimeType="video/*"/>
    </intent-filter>
</activity>
```