ATLS 4120/5120: Mobile Application Development
Week 13: Activities and Intents

Android follows the model view controller (MVC) architecture

- Model: holds the data and classes
- View: all items for the user interface
- Controller: links the model and the view together. The backbone or brain of the app.
- These categories should never overlap.

Activities

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

- An activity is a single, specific task a user can do
- Each activity has its own window for its view The window typically fills the screen, but may be smaller than the screen and float on top of other windows
- An app can have as many activities as needed
- Each activity is listed in the AndroidManifest.xml file
- There is typically a 1:1 ration for activity and layout

To start an activity you need to define an intent and then use the startActivity(Intent) method to start a new activity.

Intents

https://developer.android.com/reference/android/content/Intent.html

An intent tells the app you're about to do something such as start a new activity

- Provides the binding between two activities

You build an Intent with two key pieces of information

- Action – what action should be performed
- Data – what data is involved

There are two types of intents

- An explicit intent tells the app to start a specific activity
  - Provide the class name
- An implicit intent tells Android what type of action you want to perform
  - the app will start any activity that can handle the action specified
  - Android uses intent resolution to see what activities can handle the intent An intent filter specifies what types of intents each component can receive
  - The intent filter also specifies a category
  - Each activity has intent filters defined in the AndroidManifest.xml file
  - Android compares the information given in the intent with the information given in the intent filters specified in the AndroidManifest.xml file.
  - An intent filter must include a category of android.intent.category.DEFAULT if it's to receive implicit intents
  - If an activity has no intent filter, or it doesn't include a category name of android.intent.category.DEFAULT, it means that the activity can't be started with an implicit intent. It can only be started with an explicit intent using the fully qualified component name.
  - Android first considers intent filters that include a category of android.intent.category.DEFAULT

- Android then matches the action and mime type with the intent filters
- Those intents are then presented to the user
- If just one activity can handle the intent, that activity is chosen
- If there is more than one activity that can handle the intent, the user is prompted to chose
- Android tells the activity to start even though it's in another app and passes it the intent

Data
https://developer.android.com/training/basics/firstapp/starting-activity.html
You can add extra information to your intent to pass data to the new activity using the putExtra() method
- The putExtra() method is overloaded so you can pass many possible types
- Call putExtra() as many times as needed for the data you're passing

When a new activity starts it needs to receive any data passed to it in the intent using the getExtra() methods.

Using intents Android knows the sequence in which activities are started. This means that when you click on the Back button on your device, Android knows exactly where to take you back to.

Coffee
Create a new project called Coffee
Minimum SDK: API 16
Empty Activity template
Activity name: FindCoffeeActivity
Check Generate Layout File
Layout Name: activity_find_coffee
Backwards Compatibility should be checked

User Interface
Use the textview provided as a heading that says "Coffee Shop Finder".
Make the text size larger and center it horizontally.
Also change its text property to use a string resource.

Add a spinner and a textview above it that will describe the spinner with text "Chose your crowd"
Add a button with the text "Find Coffee"
Add an imageview below the button. Copy an image into the res/drawable folder and use that as the src in the xml. You should also add a contentDescription using a string resource.

If you add these from top to bottom they will be added to the layout below each other.
Make sure you use string resources for any text.

Strings.xml
```
<resources>
   <string name="app_name">Coffee</string>
   <string name="title">Coffee Shop Finder</string>
   <string name="coffee_type">Chose your crowd</string>
   <string name="button">Find Coffee</string>
   <string-array name="crowd">
      <item>popluar</item>
      <item>cycling</item>
```

```
        <item>hipster</item>
        <item>tea</item>
        <item>hippie</item>
        <item>college</item>
    </string-array>
    <string name="coffee_image">coffee cup</string>
</resources>
```

Acitivity_find_coffee_.xml

```
<TextView
    android:text="@string/title"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:textSize="24sp"
    android:layout_marginTop="10dp"
    android:id="@+id/textView" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/coffee_type"
    android:id="@+id/textView2"
    android:layout_below="@+id/textView"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginTop="47dp" />

<Spinner
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/spinner"
    android:entries="@array/crowd"
    android:layout_below="@+id/textView2"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginTop="20dp" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button"
    android:id="@+id/button"
    android:layout_below="@+id/spinner"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="20dp" />

<ImageView
```

```
   android:layout_width="135dp"
   android:layout_height="137dp"
   android:id="@+id/imageView"
   android:src="@drawable/coffee"
   android:contentDescription="@string/coffee_image"
   android:layout_below="@id/button"
   android:layout_centerHorizontal="true"
   android:layout_marginTop="40dp" />
```

Java class

We're going to create a custom Java class for coffee shop info.
In the app/java folder select the coffee folder (not androidTest)
File | New | Java class (or right click)
Select …/app/src/main/java
Name: CoffeeShop
Kind: Class

We're going to create a CoffeeShop class with two data members to store the coffee shop name and URL.
We'll have getter and setter methods for both and a private utility method that choses the coffee shop.

```
public class CoffeeShop {
   private String coffeeShop;
   private String coffeeShopURL;

private void setCoffeeInfo(Integer coffeeCrowd){
   switch (coffeeCrowd){
      case 0: //popular
         coffeeShop="Starbucks";
         coffeeShopURL="https://www.starbucks.com";
         break;
      case 1: //cycling
         coffeeShop="Amante";
         coffeeShopURL="https://www.amantecoffee.com";
         break;
      case 2: //hipster
         coffeeShop="Ozo";
         coffeeShopURL="https://ozocoffee.com";
         break;
      case 3: //tea
         coffeeShop="Pekoe";
         coffeeShopURL="http://www.pekoesiphouse.com";
         break;
      case 4: //hippie
         coffeeShop="Trident";
         coffeeShopURL="http://www.tridentcafe.com";
         break;
      default:
         coffeeShop="none";
```

```
        coffeeShopURL="https://www.google.com/search?q=boulder+coffee+shops&ie=utf-
8&oe=utf-8";
    }
}

    public void setCoffeeShop(Integer coffeeCrowd){
        setCoffeeInfo(coffeeCrowd);
    }

    public void setCoffeeShopURL(Integer coffeeCrowd){
        setCoffeeInfo(coffeeCrowd);
    }

    public String getCoffeeShop(){
        return coffeeShop;
    }

    public String getCoffeeShopURL(){
        return coffeeShopURL;
    }
}
```

FindCoffeeActivity.java

First we need to create an object of our new CoffeeShop class and then implement our findCoffee() method.

```
private CoffeeShop myCoffeeShop = new CoffeeShop();

public void findCoffee(View view){
//get spinner
Spinner crowdSpinner = (Spinner)findViewById(R.id.spinner);
//get spinner item array postion
Integer crowd = crowdSpinner.getSelectedItemPosition();
//set the coffee shop
myCoffeeShop.setCoffeeShop(crowd);
//get suggested coffee shop
String suggestedCoffeeShop = myCoffeeShop.getCoffeeShop();
//get URL of suggested coffee shop
String suggestedCoffeeShopURL = myCoffeeShop.getCoffeeShopURL();
System.out.println(suggestedCoffeeShop);
System.out.println(suggestedCoffeeShopURL);
}
```

For now we're using System.out.println() just to make sure it's working.

Button

We could add the onClick event to our button like we did last time, but we're going to use an event listener instead.

You can only use the android:onClick attribute in activity layouts for buttons, or any views that are subclasses of Button such as CheckBoxes and RadioButtons.

So it's good to understand how to set up event listeners.

Update the onCreate() method.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_find_coffee);
   //get button
   final Button button = (Button) findViewById(R.id.button);
   //create listener
   View.OnClickListener onclick = new View.OnClickListener(){
      public void onClick(View view){
         findCoffee(view);
      }
   };
   //add listener to the button
   button.setOnClickListener(onclick);
}
```

New Activity
File | New |Activity
Either Gallery or Empty Activity
Activity name: ReceiveCoffeeActivity
Check Generate Layout File
Layout Name: activity_receive_coffee
This creates a new layout xml file and java file for our new activity.
It also updates the AndroidManifest.xml file with a new activity.

Our layout will simply consist of a textView where we'll suggest a coffee shop.
Add a text view and give its text a string resource and a descriptive id of coffeeShopTextView
(you can remove the text once your layout is set)

```
<string name="suggested_coffee">This is your suggested coffee shop</string>

<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/suggested_coffee"
   android:textSize="24sp"
   android:id="@+id/ coffeeShopTextView"
   android:layout_centerHorizontal="true"
   android:layout_marginTop="68dp" />
```

Explicit Intent
Now let's get the button in the first activity to call the second activity.
In FindCoffeeActivity.java update findCoffee() to create and start an intent.

```
//create an intent
Intent intent = new Intent(this, ReceiveCoffeeActivity.class);
```

```
//start the intent
startActivity(intent);
```

(press option + return on Mac (Alt + Enter on Windows) to import missing classes.)

Passing Data

Now let's pass the coffee shop name and URL to the second activity.
In FindCoffeeActivity.java BEFORE you start the new activity, add the data to the intent.

```
//create an intent
Intent intent = new Intent(this, ReceiveCoffeeActivity.class);

//pass data
intent.putExtra("coffeeShopName", suggestedCoffeeShop);
intent.putExtra("coffeeShopURL", suggestedCoffeeShopURL);

//start the intent
startActivity(intent);
```

Receiving Data

Now let's update ReceiveCoffeeActivity.java to get the data sent in the intent. Create two private strings in the class

```
private String coffeeShop;
private String coffeeShopURL;
```

The onCreate() method is called as soon as the activity is created so that's where we'll get the intent.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_receive_coffee);

    //get intent
    Intent intent = getIntent();
    coffeeShop = intent.getStringExtra("coffeeShopName");
    coffeeShopURL = intent.getStringExtra("coffeeShopURL");
    System.out.println(coffeeShop);
    System.out.println(coffeeShopURL);

    //update text view
    TextView messageView = (TextView) findViewById(R.id.coffeeShopTextView);
    messageView.setText("You should check out " + coffeeShop);
}
```

Implicit Intent

Let's add a button in our second activity to open up the coffee shop's web site in an external app.
Add an image button in the bottom right corner of the layout.
Add an image resource into the drawable folder and use that for the background.

```
<ImageButton
    android:layout_width="60dp"
```

```
    android:layout_height="60dp"
    android:id="@+id/imageButton"
    android:background="@drawable/earth"
    android:layout_alignParentBottom="true"
    android:layout_alignRight="@+id/coffeeShopTextView"
    android:layout_alignEnd="@+id/coffeeShopTextView" />
```

We want to start an implicit intent when the button is clicked so add an onClick event

```
android:onClick="loadWebSite"
```

In ReceiveCoffeeActivity.java implement this method

```
public void loadWebSite(View view){
    Intent intent = new Intent(Intent.ACTION_VIEW);
    intent.setData(Uri.parse(coffeeShopURL));
    startActivity(intent);
}
```

Use the back arrow to see how it takes you to the last activity you were in.

Launcher icons

Add your own launcher icon to your app by creating a new Image Asset.
Select the res folder right-click and select New > Image Asset
Or File > New > Image Asset.
Use shape none or bevel to keep the transparent background.