Advanced Mobile Application Development
Week 12: List Views and Adapters

Spring
Create a new project called Spring
Minimum SDK: API 16
Check Phone and Tablet, leave the rest unchecked
Empty Activity template
Activity name: BulbMainActivity
Check Generate Layout File
Layout Name: activity_bulb_main

Images
To add images go into the Project view.
Navigate to app/src/main/res
Drag images into the drawable folder (or copy/paste)
These are now available through the R class that Android automatically creates.
R.drawable.*imagename*

Bulb Layout
activity_bulb_main.xml
Change from RelativeLayout to LinearLayout with **android:orientation="vertical"**
Remove the textview
Add an ImageView with these properties
android:src="@drawable/bulbs"
android:contentDescription="@string/bulbs"
android:layout_width and **android:layout_height** are required. Don't use pixels, use dp instead.
In the ImageView the properties **android:layout_below** and **android:layout_centerHorizontal** are not valid so remove them.

Having a content description for an image is optional but makes your app more accessible.
In your strings.xml add the text for this string
```
<string name="bulbs">Bulbs</string>
```

Set up the values for the list view in strings.xml
```
<string-array name="bulb_types">
   <item>Tulips</item>
   <item>Daffodils</item>
   <item>Iris</item>
</string-array>
```

Add a ListView in activity_bulb_main.xml with the property

```
android:entries="@array/bulb_types"/>
```

You should be able to run it and see your values in the list view.

We bind data to the list view using the android:entries attribute in our layout XML because the data is static. If your data is not static you use an adapter to act as a bridge between the data source and the list view.

ListView Listener

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.
The ListView class isn't a subclass of Button, so using the android:onClick attribute won't work. That's why you have to implement your own listener.
In BulbMainActivity.java add to onCreate()

```
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_bulb_main);
   //create listener
   AdapterView.OnItemClickListener itemClickListener = new AdapterView.OnItemClickListener(){
      public void onItemClick(AdapterView<?> listView, View view, int position, long id){
         String bulbtype = (String) listView.getItemAtPosition(position);
         //create new intent
         Intent intent = new Intent(BulbMainActivity.this, BulbCategoryActivity.class);
         //add bulbtype to intent
         intent.putExtra("bulbtype", bulbtype);
         //start intent
         startActivity(intent);
      }
   };
   //get the list view
   ListView listview = (ListView) findViewById(R.id.listView);
   //add listener to the list view
   listview.setOnItemClickListener(itemClickListener);
}
```

BulbCategoryActivity will give you an error because we haven't created it yet, we'll do that next.

Java class

We're going to create a custom Java class for the bulb data we'll be using in the next category activity.
In the java folder select the spring folder (not androidTest)
File | New | Java class
Name: Bulb
Kind: Class

We're going to create a Bulb class with two data members to store the bulb name and image resource id.
We'll have getter and setter methods for both and a private utility method that choses the bulb.

```
public class Bulb {
   private String name;
   private int imageResourceID;

   //constructor
```

```
    private Bulb(String newname, int newID){
        this.name = newname;
        this.imageResourceID = newID;
    }

    public static final Bulb[] tulips = {
            new Bulb("Daydream", R.drawable.daydream),
            new Bulb("Apeldoorn Elite", R.drawable.apeldoorn),
            new Bulb("Banja Luka", R.drawable.banjaluka),
            new Bulb("Burning Heart", R.drawable.burningheart),
            new Bulb("Art Royal", R.drawable.artroyal)
    };

    public String getName(){
        return name;
    }

    public int getImageResourceID(){
        return imageResourceID;
    }

    //the string representation of a tulip is its name
    public String toString(){
        return this.name;
    }
}
```

You will eventually add two more arrays for daffodils and iris.

Tulip List
As our activity only needs to contain a single list view with no other GUI components, we can use a list activity, an activity that only contains a list. It's automatically bound to a default layout that contains a list view. So we don't need to create a layout or an event listener as the ListActivity class already implements one.

New | Activity | Empty
BulbCategoryActivity
Uncheck Generate Layout File

In BulbCategoryActivity.java change the superclass from AppCompatActivity to ListActivity.

Just as with normal activities, list activities need to be registered in the AndroidManifest.xml file. This is so they can be used within your app. When you create your activity, Android Studio does this for you.

We're going to use an array adapter to bind our tulips array to our new list activity. You can use an array adapter with any subclass of the AdapterView class, which means you can use it with both list views and spinners.

You use an array adapter by initializing the array adapter and attaching it to the list view.

```
import android.widget.ArrayAdapter;

private String bulbtype;

protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   Intent i = getIntent();
   String bulbtype = i.getStringExtra("bulbtype");
   //get the list view
   ListView listBulbs = getListView();
   //define an array adapter
   ArrayAdapter<Bulb> listAdapter;
   //initialize the array adapter with the right list of bulbs
   switch (bulbtype){
      case "Tulips":
         listAdapter = new ArrayAdapter<Bulb>(this, android.R.layout.simple_list_item_1, Bulb.tulips);
         break;
      default: listAdapter = new ArrayAdapter<Bulb>(this, android.R.layout.simple_list_item_1,
Bulb.tulips);
   }
   //set the array adapter on the list view
   listBulbs.setAdapter(listAdapter);
}
```

simple_list_item_1 is a built-in layout resource that tells the array adapter to display each item in the array in a single text view.

Behind the scenes, the array adapter takes each item in the array, converts it to a String using its toString() method and puts each result into a text view. It then displays each text view as a single row in the list view.

Now you should be able to click on tulips and it will launch your new activity and show you the list of tulips.

ListActivity Listener

In BulbMainActivity.java we created the OnItemClickListener event listener and implemented its onItemClick() method.

BulbCategoryActivity.java is a subclass of the ListActivity class which is a specific type of activity that's designed to work with list views. The ListActivity class already implements an on item click event listener so instead of creating your own event listener, you just need to implement the onListItemClick() method.
This is a significant difference when it comes to handling user clicks. Since the ListActivity class already implements an on item click event listener you don't need to create your own event listener.

```
import android.view.View;
```

```
@Override
public void onListItemClick(ListView listView, View view, int position, long id){
    Intent intent = new Intent(BulbCategoryActivity.this, BulbActivity.class);
    intent.putExtra("bulbid", (int) id);
    intent.putExtra("bulbtype", bulbtype);
    startActivity(intent);
}
```

BulbActivity will give you an error because we haven't created it yet, we'll do that next.
We use the id to pass which bulb was selected.

BulbActivity
Now let's create the BulbActivity activity.
New | Activity | Empty
Activity name: BulbActivity
Check Generate Layout File
Layout name: activity_bulb

Change it to a linear layout and add the orientation.
android:orientation="vertical"

Add a Large TextView for the name.
In the TextView remove the **android:text** property and change the id to
android:id="@+id/bulb_name"
To center the text add **android:layout_gravity="center_horizontal"**

Add an ImageView for the image.
In the ImageView the properties **android:layout_below** and **android:layout_centerHorizontal** are not valid so remove them.
Change the id to **android:id="@+id/bulbImageView"**

Now let's populate the view.

View Data
In BulbActivity.java we'll get the data sent from the intent to populate the view.

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

    //get bulb data from the intent
    int bulbnum = (Integer)getIntent().getExtras().get("bulbid");
    String type = (String)getIntent().getExtras().get("bulbtype");
    Bulb bulb = Bulb.tulips[bulbnum];

    //populate image
    ImageView bulbImage = (ImageView)findViewById(R.id.bulbImageView);
    bulbImage.setImageResource(bulb.getImageResourceID());

    //populate name
    TextView bulbName = (TextView)findViewById(R.id.bulb_name);
    bulbName.setText(bulb.getName());
}
```

Don’t forget launcher icons. Android Asset Studio creates all the sizes you need.