Android Boot Camp for Developers Using Java, 3E

Chapter 5: Investigate! Android Lists, Arrays, and Web Browsers

Objectives

In this chapter, you learn to:

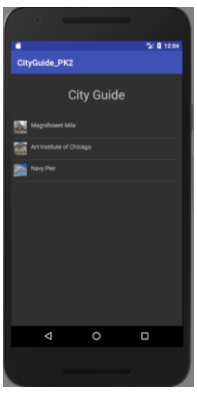
- Create an Android project using a list
- Develop a user interface that uses ListView
- Extend the AppCompatActivity class
- Use an array to create a list
- Create a custom adapter class to populate the listview
- Code a setAdapter to display an array
- Design a custom ListView layout with XML code
- Display images with the ListView control
- Change the default title bar text

Objectives (continued)

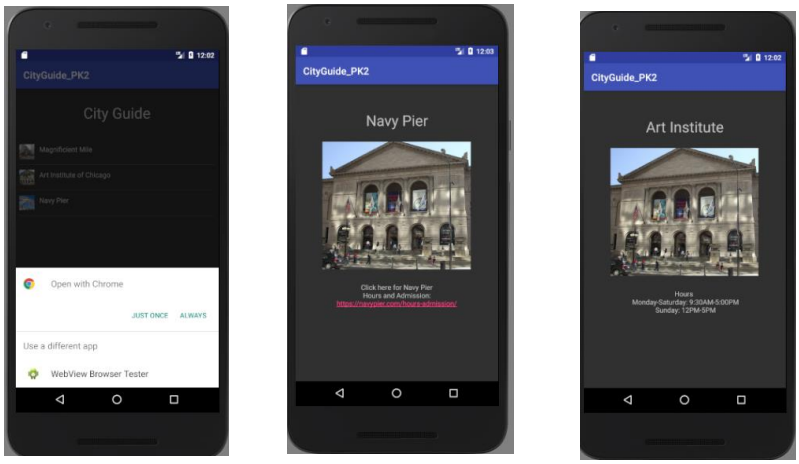
- Code a custom setListAdapter for a custom layout
- Call the onListItemClick method when a list item is selected
- Write code using the Switch decision structure
- Call an intent to work with an outside app
- Open an Android Web browser
- Launch a Web site through the use of a URI using an Android browser
- Test an application with multiple decisions

Creating a List

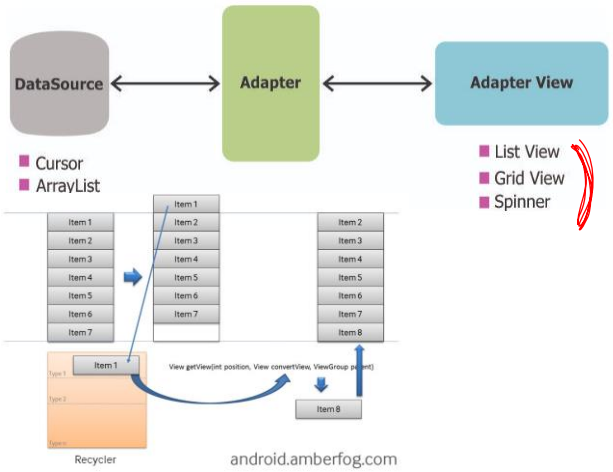
- Lists are one of the most common designs in mobile apps
  - Scrollable
  - Selectable
  - Programmable to bring up the next Activity (screen)



Creating a List (continued)



Creating a ListView



Creating a List

- Steps to complete the app:
  1. Create the ListView in the main activity layout
  2. Create a second layout containing the TextView
  3. Create your own adapter that will inherit from BaseAdapter
  4. In the MainActivity.java
     1. Define an array to establish the items of the list
     2. Define an array to establish pictures for the list
  5. Instantiate the Listv=View
  6. Call the SetAdapter with an instance of your own adapter on the listview instance

Creating a List (continued)

- Opening screen contains a vertical list of attractions
- List is automatically scrollable if it exceeds the window size
- ListView is better than TableLayout View because each row can be selected for further action
- Selecting an item opens up a related Web page or an image of the attraction
  - Call setOnItemClickListener on the ListView
  - Create alternate structures using switch

Extending a AppCompatActivity

- A **List Activity** can be inherited from AppCompatActivity

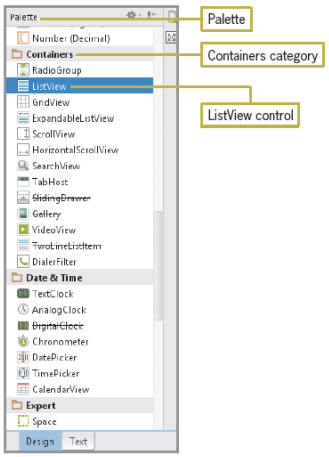


Figure 5-4 ListView control on the Palette

ListView

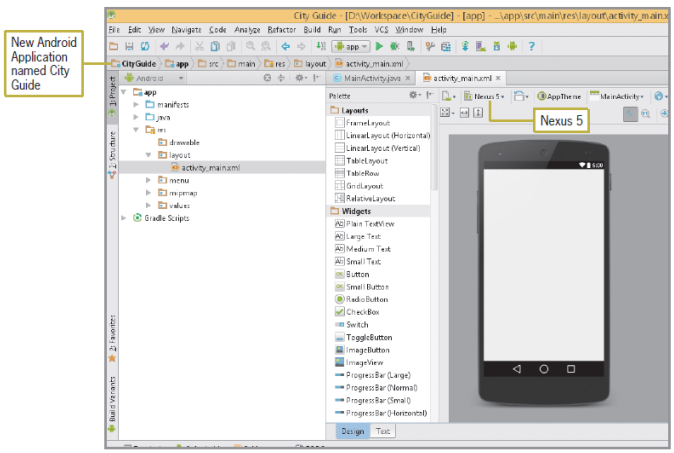


Figure 5-5 Application information for the new Android project

Creating an Array

- **Array variables** can store more than one value
- Different from other data types that can hold only one value
- Each individual item in an array is called an **element**
- Refer to each element using an index in the array

Element	Value
Attraction[0]	Art Institute of Chicago
Attraction[1]	Magnificent Mile
Attraction[2]	Willis Tower
Attraction[3]	Navy Pier
Attraction[4]	Water Tower

Table 5-1 Attraction array with index values

Declaring an Array

- Square brackets [ ] are used to define an array
- Curley braces { } contain the list of items in the array

Code Syntax

```
String[] attraction={"Art Institute of Chicago", "Magnificent Mile",
"Willis Tower", "Navy Pier", "Water Tower"};
```

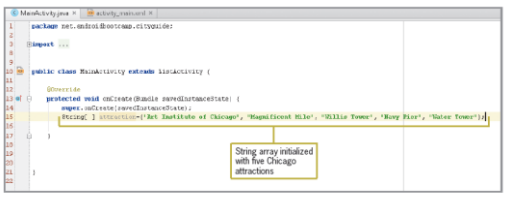


Figure 5-8 String array initialized with attractions

Using a setAdapter and Array Adapter

- An **adapter** provides a model for the layout and converts the data into a list
- The **setAdapter** connects the list items to the images/text or Web pages they represent
- An **array adapter** supplies the array data to the ListView
  - You need custom adapter if you need to pass both text and image list to the list view

Using a setAdapter and Own Array Adapter

The theme is updated in styles.xml to display a dark gray background and a black title bar when the application is executed (Figure 5-11).

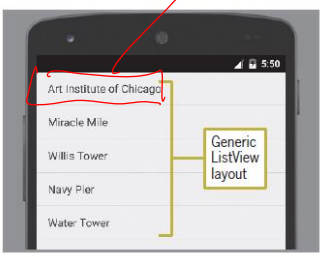


Figure 5-10 ListView built-in layout

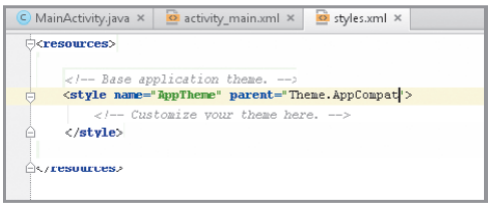


Figure 5-11 styles.xml with theme changed

Adding the Images to the Resources Folder

- Images must be located in the drawable-hdpi folder
- Remember that image may be subject to copyright laws

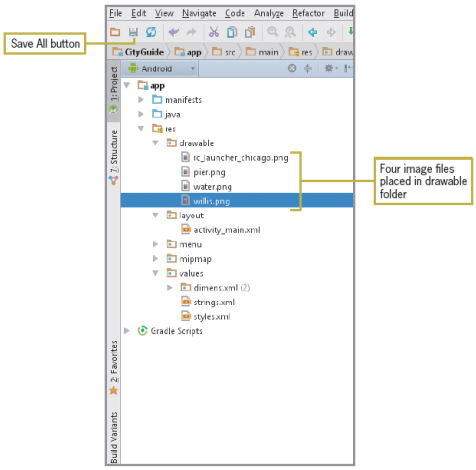


Figure 5-12 Images copied

Adding the String Table

The image description keys are entered in the Translations Editor (Figure 5-13).

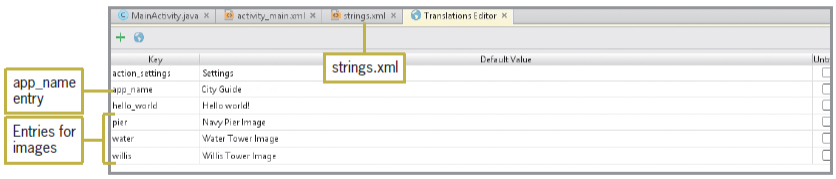


Figure 5-13 String table

Adding the string array

- Add string and integer list for names and pictures

```
List<String> Attractions = Arrays.asList("Magnificient Mile", "Navy Pier", "Art Institute");
List<Integer> AttrPics = Arrays.asList(R.drawable.magmile, R.drawable.pier, R.drawable.artinst);
```


Coding a setAdapter with a simple_list_item_1

- Built-in layout is called `simple_list_item_1`

```
ListView listViewAttr = (ListView)findViewById(R.id.listViewAttr);
listViewAttr.setAdapter(new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1,attractions));
```

Creating a Custom XML Layout for a TextView

Add this to new XML layout file: layout_txt

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/places"
        android:paddingBottom="10dp"
        android:paddingTop="10dp"
        android:drawableLeft="@drawable/ic_launcher_chicago"
        android:drawablePadding="10dp"
        />

</LinearLayout>
```


Adding My Own Adapter Class

)

- New Java Class: create a new class inherit BaseAdapter
- Has three fields:
  - List/Array of strings containing a list of attractions
  - List/Array of integers containing a list of pictures
  - Context: to pass application environment

Constructor: Your own adapter

- Constructor to create the adapter instance

```
MyCustomAdapter(List<String> anyItems, List<Integer> anyPics, Context anyContext){
    super();
    myItems = anyItems;
    myPics = anyPics;
    context = anyContext;
}
```


Your Own Adapter Class Needs the following methods

```
@Override
public int getCount() {
    return myItems.size();
}

@Override
public Object getItem(int position) {
    return myItems.get(position);
}

@Override
public long getItemId(int position) {
    return position;
}

@Override
public View getView(int position, View convertView, ViewGroup parent) {...}
```

Overriding the getView Method

- getView method is responsible for populating the contents, and is called when you set the adapter
  - For each item you pass to your adapter

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    View view;
    if (convertView == null){
        LayoutInflater layoutInflater = LayoutInflater.from(context);
        view = layoutInflater.inflate(R.layout.layout_item, parent, attachToRoot: false);
    } else {
        view = convertView;
    }
    TextView txtViewItem = (TextView)view.findViewById(R.id.txtViewItem);
    txtViewItem.setText(myItems.get(position));
    txtViewItem.setGravity(Gravity.CENTER_VERTICAL);
    Drawable img = parent.getResources().getDrawable(myPics.get(position));
    img.setBounds( left: 0,  top: 0,  right: 60,  bottom: 60);
    txtViewItem.setCompoundDrawables(img, top: null,  right: null,  bottom: null);
    txtViewItem.setCompoundDrawablePadding(8);
    return view;
}
```


Using the onListItemClick method

- **onItemClickListener()** is called when an item from the list is selected
- The item's **position** in the list is captured so the app knows which of the items was selected
- The position represents the number of the item in the list

Calling the set adapter with your own adapter instance

- Call setAdapter with MyOwnAdapter instance with the constructor call

```
listViewAttr.setAdapter(new MyOwnAdapter(this, attractions, pictures));
```


Using the onItemClickListener method (continued)

- Call the setOnItemClickListener with OnItemClickListener definition stub

```
listViewAttr.setOnItemClickListener((adapterView, view, i, l) -> {

});
```

Decision Structure – Switch Statement

- If statements are also decision structures
- The **switch** statement is used when there are many list items to be evaluated
- Can only evaluate integers or single characters
- The keyword **case** is used to test each item
- The keyword **break** is used to exit the switch decision structure

Adding code inside the OnItemClickListener

- Using switch for position of the item on the list

```
listViewAttr.setOnItemClickListener((adapterView, view, i, l) → {
        switch (i) {
            case 0:
                break;
            case 1:
                break;
            case 2:
                break;
        }
});
```

Android Intents

- Android intents send and receive activities and services including:
  - Opening a Web page in a browser
  - Calling a phone number
  - Locating a GPS position on a map
  - Posting notes to a note-taking program
  - Opening your contacts list
  - Sending a photo
  - Posting to a social network

Launching the Browser from an Android Device

- The intent sends the browser a **URI** (Uniform Resource Identifier)
- URI is similar to **URL** (Uniform Resource Locator)
- URI has additional information necessary for gaining access to the resources required for posting the page
- The action called **ACTION_VIEW** (must be in CAPS) is what actually displays the page in the browser
- ACTION_VIEW is the most common action performed on data

Launching the Browser from an Android Device (continued)

- Adding code to open magnificent mile URL in the browser

```
case 0:
    startActivity(new Intent(Intent.ACTION_VIEW,
            Uri.parse("https://www.themagnificentmile.com/?url=/")));
    break;
```


Adding Multiple Class Files

- **Class Files are needed to display images on the screen when the user selects options**
  - An onCreate method requests that the user interface opens to display an image of the attraction

A new class named Willis that creates activity_willis.xml appears in New Android Activity dialog box with the title Willis Tower (Figure 5-24).

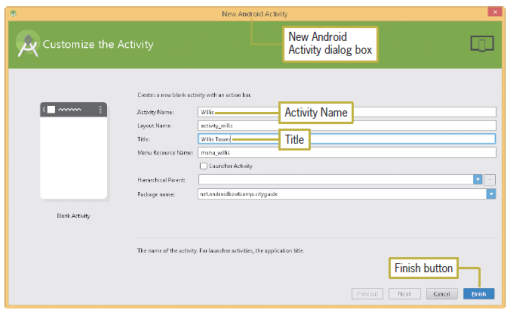
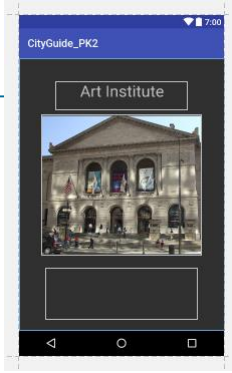


Figure 5-24 Creating the Willis.java class file

Adding a new activity



- Add ArtInstitute as an Empty Activity
- Add TextView, ImageView and another empty TextView for hours
- Set Text for Hours programmatically

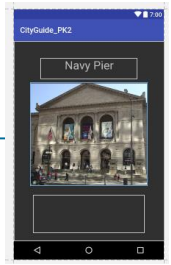
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_art_institute);
    TextView txtArtInstHours = (TextView)findViewById(R.id.txtHours);
    txtArtInstHours.setText("Hours \n  Monday-Saturday: 9:30AM-5:00PM " +
            "\n Sunday: 12PM-5PM");
}
```


Opening the Class Files

- A startActivity method opens the next Activity, which in turn launches the appropriate XML layout displaying an image of the attraction

```
case 1:
    startActivity(new Intent(MainActivity.this,
            ArtInstitute.class));
    break;
```

Adding another new activity



- Add NavyPier as an Empty Activity
- Add TextView, ImageView and another empty TextView for hours
- Set Text for Hours programmatically – as a link to an external link

XML

```
android:autoLink="web"
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_navy_pier);
    TextView txtNavyPierHours = (TextView)findViewById(R.id.txtNavyPierHours);
    String NavyPierHourMsg = "Click here for Navy " +
            "Pier Hours and Admission: https://navypier.com/hours-admission/";
    txtNavyPierHours.setText(NavyPierHourMsg);
    LinkifyCompat.addLinks(txtNavyPierHours, Linkify.WEB_URLS);
}
```


Opening the Class Files

- A startActivity method opens the next Activity, which in turn launches the appropriate XML layout displaying an image of the attraction

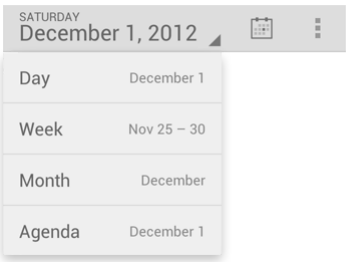
```
case 2:
    startActivity(new Intent(MainActivity.this,NavyPier.class));
    break;
```

Running and Testing the Application

- Make sure all the fields can gracefully handle any tap or click or any value entered in any Android app
- Testing an Android app is called usability testing
  - For more details, refer to *www.section508.gov*

Adapters: Beyond ListViews

- Can use adapter implementation with different layouts for each item
  - Less common as it affects consistency
- Adapter implementation can be used with Spinners as well
  - Similar to listview
  - Can set spinner items programmatically
  - Can use different layouts for each spinner items as well



Summary

- The Java View class creates a list and makes it scrollable; use a ListView control to select each row for further action
- Extend the AppCompatActivity class in Main.java to instantiate ListView control (already created in the layout)
- Declare list items (text, images) in an array variable
- Array indexes provide access to each element in the list (the list begins with 0)
- To declare an array, specify the data type followed by the values

Summary (continued)

- Add code to the main.xml file to get a custom layout
- Create a custom adapter that inherits from BaseAdapter
- Use an adapter to display the text and images from the arrays
- A ListView control is a container for the list items and the adapter binds the elements of the array to the ListView
- Drag controls from the palette to the emulator for a simple design

Summary (continued)

- setAdapter called on the new Adapter instance
- Adapter instance creation needs three parameters
  - *this* class - context
  - String array for the texts
  - Integer array for the images
- Code the onItemClickListener method to respond to the event of the user's selection
- Use the switch decision structure with a list or menu

Summary (continued)

- Android intents send and receive activities and services
- Test every possible combination of clicks, including incorrect user entries before publishing the app