Activity 1.2.2B Today's Top 40

Otakar Andrysek

Do 18 to 27...and then try going on, but I may have to change some of the instructions.

18. An important feature when using a for-each loop: you can change the data *in* the object, but you cannot change the object reference *itself*. An example will help make this clear.
    a. Using your array name for your songs, use the following code to change data *in* each Song object, specifically, each song's title:

```
        // change the array
        System.out.println("—BEFORE--"); //(This should go before 
the print out we just made. -- DW)
        for (Song changeSong : topTenSongs) {
            changeSong.setTitle("test");
        }
        // show the array
        System.out.println("—AFTER--");
        for (Song showSong : topTenSongs) {
            System.out.println(showSong.getTitle());
        }
```

    b. Run your program.
    The first loop changes the data in the object and all song titles should be "test".

    c. Replace the line of code in the first loop to change the *object reference* of the songs. Add a line of code to see the new title.

```
        for (Song changeSong: topTenSongs) {
             changeSong = new Song("test");
             System.out.println(changeSong.getTitle());
         }
```

    The second loop remains unchanged.
    d. Run your program.
    In the first loop, the songs in your array appear to change, but in the second loop, they have not changed and they contain their original titles.

When you assign a new Song to the changeSong variable, you assign the new song to the *temporary variable* changeSong and not the actual data in your array. At the end of the loop, this temporary variable goes away and the data along with it.

In the first example, the code song.setTitle("test"); uses the temporary variable changeSong to *point to the data in the object itself* and changes data *in the object*. Even though the song variable goes away, you have changed the data that each Song object points to in memory.

19. If you want to change the *object reference*, meaning the entire object itself and not just the data *in* it, you must use a standard for loop. In the body of the for loop, reassign the object reference for all of your songs using syntax similar to topTenSongs[i] = new Song("test");.

20. Once you have modified the array using a standard for loop, you can show the contents of the array using a for-each loop as before. Run your program and confirm correctness.

    Confirmed.

21. You probably do not want all of your song titles to be "test", so once you have confirmed that you can change object references, comment out that code so the original Song objects remain unchanged.

    Done.

22. Use a for-each loop to add a standard price to your songs. Remember changeSong.setPrice(1.29) changes the data *in* the Song object, so a for-each loop is a good construct to use.

    Done.

23. Modify your loop so that every third song has a discounted price of $.99. Use the modulus operator % to determine every third song.

    Done.

24. Use a second for-each loop to show titles and prices, and run your program to show correctness.

    Done.

Part V: Loop Out of Order

For this part of the activity, you will explore other ways to iterate over the items in an array: you can go backward through an array and you can also loop through just a section of an array.

25. Often, you want to loop through arrays in a different order, for example, **Looping From Back to Front**. Complete the online exercises.

    Why does the algorithm in getIndexLastSmaller work by looping back to front?

    The getIndexLastSmaller( ) starts at the end of an array which is defined by one less than the length, from there the algorithm increments all the way back to the start of the string.

26. Read through **Looping through Part of an Array**, completing the exercises. Write a new for loop to show the top five songs.

    Done.

    What is the danger of writing a loop in this way? What could you do to protect against this problem?

    If an array has less than five elements the looping method would be broken (OutOfBounds) error. One way to prevent this error is to make an if statement to detect if the index is less than the array length.

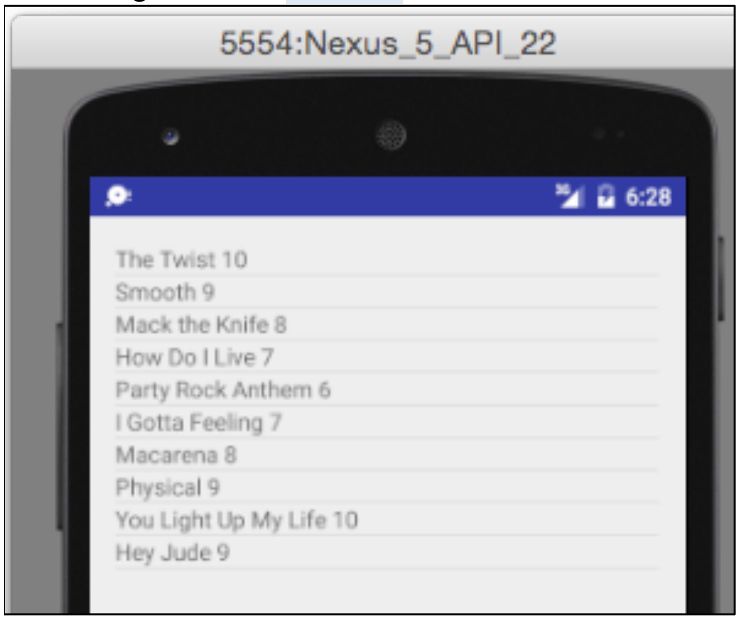
27. Write a for loop *and* a for-each loop to show every other song.

    Done.

To be completed in class.

Part VI: Android ListViews

ListViews are a common way to display data in apps as shown below. The goal of this part of the activity is to create an app that displays all the song data you've been working with in a ListView.



The Twist 10
Smooth 9
Mack the Knife 8
How Do I Live 7
Party Rock Anthem 6
I Gotta Feeling 7
Macarena 8
Physical 9
You Light Up My Life 10
Hey Jude 9

28. Open your Android Studio MediaLib project from the previous activity. If you have not launched Android Studio yet and created a MediaLib project, complete the following:
    a. In *Activity 1.1.1 Introduction to Android Studio*; Part III.
    b. In *Activity 1.1.2 Your First Class*; Parts V, VI, and VII.
    c. In *Activity 1.1.3 Making Objects*; Parts IV and V.

29. To make use of a ListView, you will need to make some modifications to the code created by Android Studio. On line 1 of MediaLibActivity below, replace AppCompatActivity with ListActivity. Remember your line numbering will be different.

```
1  public class MediaLibActivity extends AppCompatActivity {

2      @Override
       protected void onCreate(Bundle savedInstanceState) {
3          super.onCreate(savedInstanceState);
           setContentView(R.layout.activity_media_lib);
4      }
   }
5

6

7
```

The previous step may create an error because your project doesn't know what a ListActivity is. Whenever this kind of error occurs, you can correct it by importing the appropriate library.

30. To do this in Android Studio, hold **Alt** and press **Enter**. In this particular case, you could also type the following line near the top of the file with the other import statements:

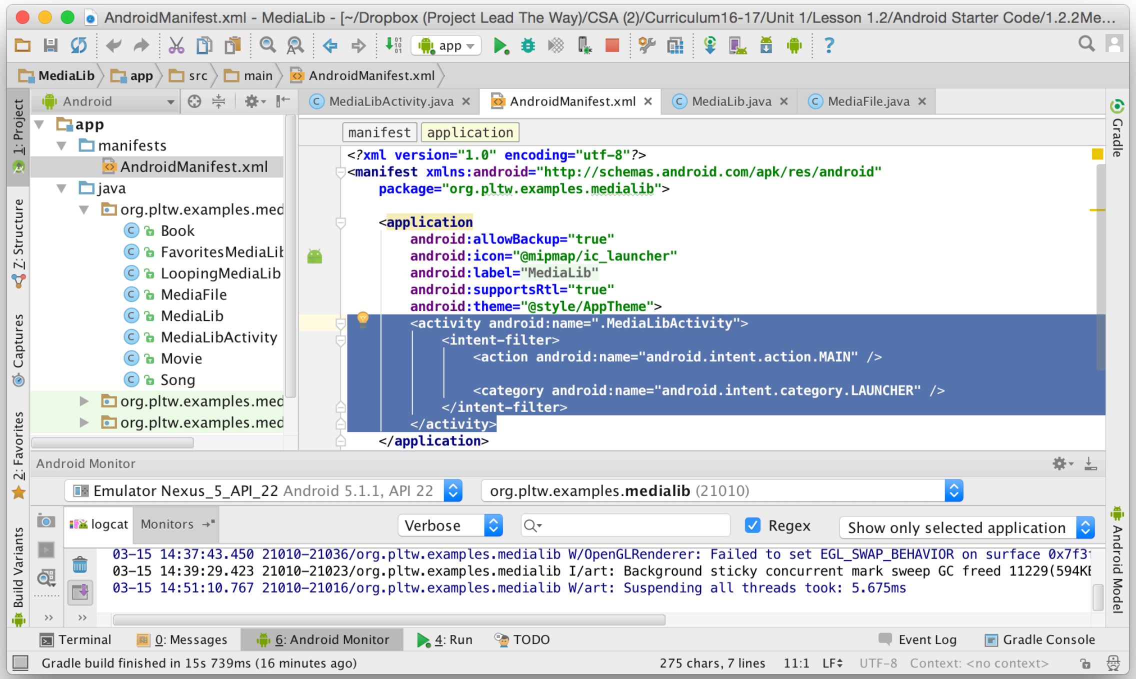
```
        import android.app.ListActivity;
```

31. Add the instance variables on lines 2–3 below to your MediaLibActivity class.

```
1  public class MediaLibActivity extends AppCompatActivity
   {     private Song[] songs;
2      private ListView songListView;

3
```

In a moment, you will manually place Song in the project.
MediaLibActivity is the class that gets created when your app launches. If you want to change which class gets created on launch, you would do so in your *AndroidManifest.xml* file for the project. In the following figure, the location of this file is shown in the Project panel, and the highlighted code in the file tells the Android OS that MediaLibActivity is part of this app. The line that reads:
<action android:name="android.intent.action.MAIN" />
tells the Android OS to launch MediaLibActivity as the main activity for the app.



AndroidManifest.xml - MediaLib - [~/Dropbox (Project Lead The Way)/CSA (2)/Curriculum16-17/Unit 1/Lesson 1.2/Android Starter Code/1.2.2Me...

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.pltw.examples.medialib">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="MediaLib"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MediaLibActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
```

Emulator Nexus_5_API_22 Android 5.1.1, API 22 org.pltw.examples.medialib (21010) Verbose ✓ Regex Show only selected application

```
03-15 14:37:43.450 21010-21036/org.pltw.examples.medialib W/OpenGLRenderer: Failed to set EGL_SWAP_BEHAVIOR on surface 0x7f3
03-15 14:39:29.423 21010-21023/org.pltw.examples.medialib I/art: Background sticky concurrent mark sweep GC freed 11229(594KE
03-15 14:51:10.767 21010-21016/org.pltw.examples.medialib W/art: Suspending all threads took: 5.675ms
```

Gradle build finished in 15s 739ms (16 minutes ago) 275 chars, 7 lines 11:1 LF UTF-8 Context: <no context>

32. Between the line beginning with package and the line beginning with <application, enter the following code, which tells the Android OS that your app will need access to external storage such as an SD (Secure Digital) card.

```
<uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

The Android OS knows about MediaLibActivity because of the *AndroidManifest.xml* file, but what happens when the OS tries to launch this app? It calls the onCreate method of the main class—in this case, the onCreate method of MediaLibActivity. For this reason, the onCreate method of an Activity is similar to the main method of a plain old Java object like the MediaLib class you created earlier in this lesson.

33. Open *activity_media_lib.xml* and replace the following code:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!" />
```

with this code:

```
<ListView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@android:id/list"></ListView>
```

The previous step created space in the layout of your app for a ListView. Note that the name of this file activity_media_lib appears in MediaLibActivity within the setContentView method.

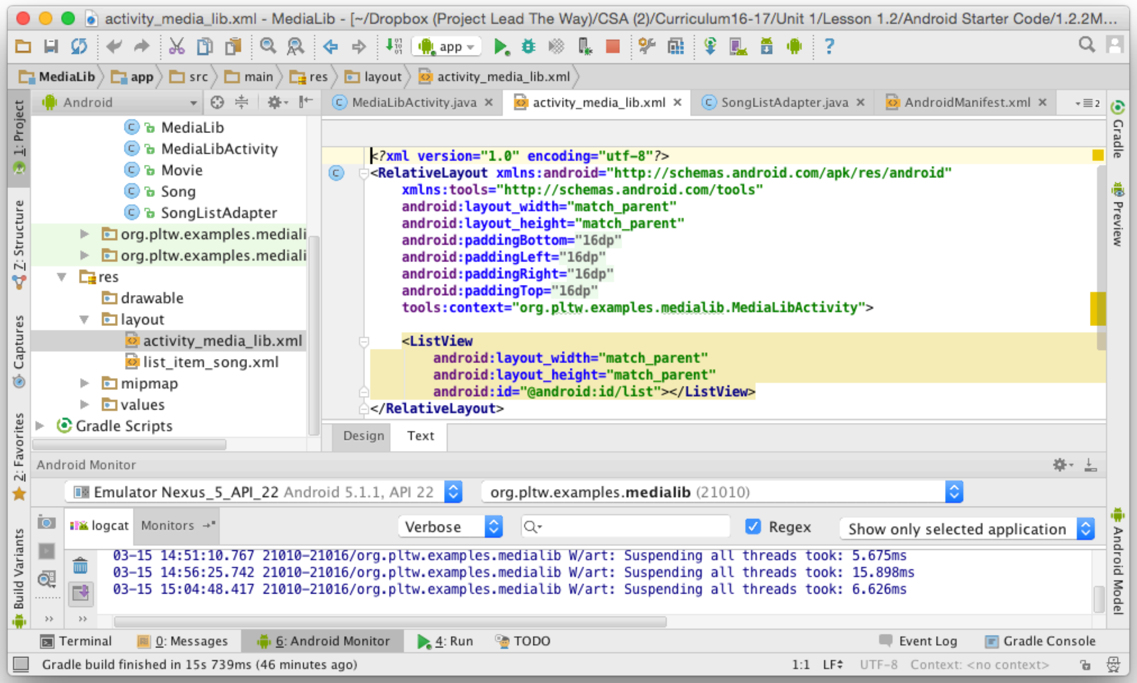
34. Use the official Android Developer site to find out what the setContentView(int layoutResID) method does and summarize it here.

35. Drag your Song.java, MediaFile.java, and Movie.java files into the directory containing MediaLibActivity. The introduction of the *Song.java* file to the project should resolve the error in MediaLibActivity introduced in Step 31. The other two files will come into play in later steps.

The first couple of lines of a typical onCreate method are:

```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_media_lib);
```

The first has to do with saving the state of the app in case the app is closed or quit. The second attaches this Activity to the xml file shown in the figure below.



```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="org.pltw.examples.medialib.MediaLibActivity">

    <ListView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@android:id/list"></ListView>
</RelativeLayout>
```

Emulator Nexus_5_API_22 Android 5.1.1, API 22 org.pltw.examples.medialib (21010)

Verbose ✓ Regex Show only selected application

```
03-15 14:51:10.767 21010-21016/org.pltw.examples.medialib W/art: Suspending all threads took: 5.675ms
03-15 14:56:25.742 21010-21016/org.pltw.examples.medialib W/art: Suspending all threads took: 15.898ms
03-15 15:04:48.417 21010-21016/org.pltw.examples.medialib W/art: Suspending all threads took: 6.626ms
```

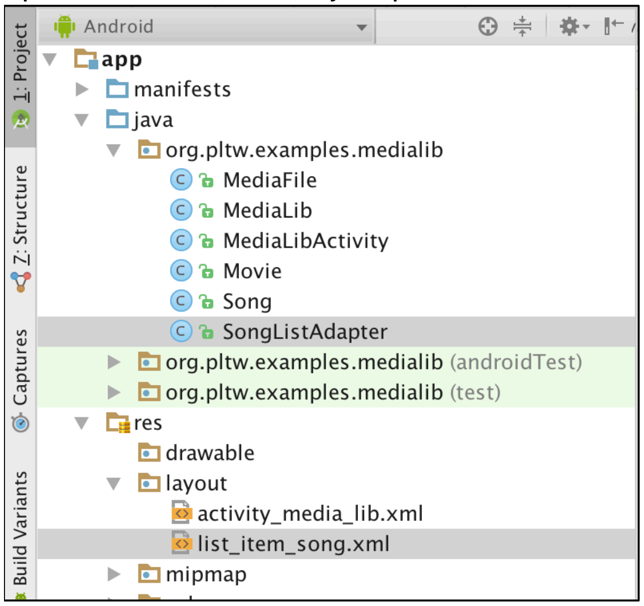
Gradle build finished in 15s 739ms (46 minutes ago) 1:1 LF UTF-8 Context: <no context>

36. Beneath both of those lines, but still within the onCreate method of MediaLibActivity, add the code from *MediaLib.java* that you used to create Song objects and write them to *mymedia.txt*.

37. Get a copy of *1.2.2MediaLibApp_StarterCode* from your teacher and copy or extract the individual files to a location that is convenient, but *not* in your project folder. Your Desktop is an example of a good location.
    a. Copy *SongListAdapter.java* to your project's *java* folder, such as *MediaLib/app/src/main/java/org/pltw/examples/medialib*

    b. Copy the *list_item_song.xml* to your project's *layout* folder, such as *MediaLib/app/src/main/res/layout*

38. Once these files are in place, your project in Android Studio will automatically update them in the Project panel:



These files will help the Android OS convert your Song data into something usable within a ListView.

39. Using what you've learned about parsing Strings, reading data from files, and manipulating data in arrays, store the data about songs from *mymedia.txt* back into Song objects. Try doing this in BlueJ first, because the MediaLib Android Studio project will not yet run. Once you have this code working, it will go in the onCreate method of MediaLibActivity underneath the code from Step 29.

An **ArrayList** is a data structure much like an array, but it has many convenience methods implemented for common list operations. The <Song> indicates that this particular ArrayList can store elements of type Song.

40. Find the official documentation for ArrayLists. You will learn more about these later in the course, but for now summarize what any two methods of the ArrayList class do.

41. Add the following to the onCreate method of the MediaLibActivity class, after your existing code, and it will be ready to run!

```
songLibrary = new ArrayList<Song>();
for (int i = 0; i < numSongs; i++) {
    songLibrary.add(songs[i]);
}
SongListAdapter adapter = new SongListAdapter(this, songLibrary);
setListAdapter(adapter);
```

Conclusion

1. Explain why the following code will not change the values in the names array to "Default Name".

```
String names = {"Adam", "Barbara", "Carlos", "Dianne"};

for (String n : names) {
    n = "Default Name";
}
```

   Lorem Ipsum.

2. Which looping construct is best to display every fifth element of an array? Why? Be sure to address all iteration constructs that you know.

   Lorem Ipsum.

3. Use the Internet to help you find out what JavaScript Object Notation (JSON) is. Explain how it could be useful in writing an Android app like the one that you worked on in this activity.

   Lorem Ipsum.