

ACTIVITY 1.2.2

Today's Top 40

INTRODUCTION

In the last activity, you displayed a number of song titles, one at a time, using either a **for loop** or a **while loop**. Imagine that you wanted to store all of this information at one time in your program and that you had not 10 songs, but hundreds, perhaps thousands of songs. Maybe you do have hundreds or thousands of songs stored in a program somewhere! In this activity, you will learn a new construct to store many data items at once, and that construct is an **array**.

Materials

- Computer with BlueJ and Android™ Studio
- Android™ tablet and USB cable, or a device emulator

RESOURCES



Lesson 1.2 Reference Card for Strings
Resources available online

for loop

A way to iterate, especially when you know how many times you want to loop, commonly used to iterate over arrays and ArrayLists. Also known as a “standard for loop”.

while loop

A way to iterate using a conditional statement that evaluates to true or false, such as `(x >= 0)`.

array

A collection or list of similar data items, accessed through a single variable reference.

Procedure

Part I: Make an Array

To start off this activity you will populate an array with a list of values.

1



Learn about Arrays in Java. Be sure to answer the **Check for understanding** questions at the end.

2

When you write a statement such as

```
String[] names = {"Jamal", "Emily", "Destiny", "Mateo", "Sofia"};
```

you are using an **initialization list**. In one step, you declare and initialize the `String` array `names`. This statement can be rewritten in other ways, too:

```
String[] names = {"Jamal",  
                  "Emily",  
                  "Destiny",  
                  "Mateo",  
                  "Sofia" // notice no comma  
};
```

or

```
String[] names = {new String("Jamal"),  
                  new String("Emily"),  
                  new String("Destiny"),  
                  new String("Mateo"),  
                  new String("Sofia")  
};
```

Recall that the Java compiler ignores **whitespace** (the non-printable characters such as blanks, returns, or tabs) when it interprets code.

initialization list

A way to declare and initialize an array in one step.

whitespace

Blank or non-visible characters such as spaces, tabs and newlines.

- 3 Open your BlueJ MediaLib project and create a new class `ArrayMediaLib`, with a `main` method similar to your other MediaLib classes.
- 4 Using an initialization list, create a `String` array of at least five friends or family members with whom you would like to share your media items. Call the array `sharingFriends`.

Part II: Looping and Arrays

You can loop over the items in an array in much the same way you did with the characters in a string. In this part of the activity, you will explore looping over the items in an array and also what happens if you try to access an index in an array that does not exist.

- 5 Use a `for` loop to iterate over your friends' array using the following:

```
for (int i = 0; i < 5; i++)  
{  
    System.out.println(sharingFriends[i]);  
}
```

- 6 Test your program.

What do you think would happen if you had less than five people in your array? Find out.

- 7 Comment out one person from the initialization list using `//`. If you comment the last element, be sure to delete the comma (,) on the line above it to indicate it is the last item in the initialization list.
- 8 Run your program. What happens?

The problem is called a **runtime error** because it happened while your program was running. It is also an **exception** because something out of the ordinary happened in your code. In this case, you tried to iterate beyond the length of your array causing an "array index out of bounds" exception.


runtime error

An error that occurs while a program is executing.

exception


A type of runtime error that indicates something unexpected (exceptional) occurred.

- 9 At what line of code did that error occur?

- 10 To make your loop work with any number of songs, read  **Using a For Loop to Loop Through an Array**. Then, modify your for loop to fix the problem. Test your program and record how you fixed it.

Part III: Use a For-Each Loop

The for-each loop is a java construct that is very useful for iterating over the items in an array. Explore this construct in this part of the activity.

- 11 Because looping over arrays is such a common algorithm, Java provides a convenient construct called a **for-each** loop.  **Learn about the For-Each Loop**, but do not read past the second paragraph of text, as these are advanced topics.

- 12 In your media library, assume you have been tracking the number of days between song purchases. In `ArrayMediaLib`, create the following array:

```
1: int[] daysBtwnPurchase = {2, 5, 1, 2, 4, 2, 1, 3};
```

- 13 Use an integer `total` and a for-each loop to calculate the average purchase time in days. Like you did at the end of Part II, make the algorithm versatile; do not hard code the number of data items.
- 14 Display the result.

Part IV: Arrays of Objects


In addition to integers, doubles, and strings, you can create arrays of other types of data, including objects.

- 15 In `ArrayMediaLib`, declare and initialize an array of `Song` objects:

```
1: Song[] topTenSongs = {new Song(),
2:                       new Song(),
3:                       new Song()
4: };
```

Compare this syntax with syntax you used to create your array of friends. Notice, in particular, the `Song[]` syntax in place of `String[]`. This indicates the data type of the array. For `topTenSongs`, the data type is `Song`.

You *could* use 10 calls to `setTitle(...)` to set all of your song titles, but an initialization list requires less code.

- 16 Modify your array of songs to declare and initialize 10 `Song` objects. Use the top 10 songs in  **Billboard's top 100 Rock-n-Roll songs of all time**. You will not have a price or rating for these songs, so create and use a new `Song` constructor with only a title as its parameter.
- 17 Display the song titles using a for-each loop.

```
1: for (Song s: topTenSongs)
2: {
3:   System.out.println(s.getTitle());
4: }
```

In the body of the for loop, notice `s.getTitle()`. Since `s` is a `Song`, you can access all of the public methods with `s.` syntax.

- 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.
- Using your array name for your songs, use the following code to change data *in* each Song object, specifically, each song's title:

```
1: // change the array
2: System.out.println("-BEFORE--");
3: for (Song changeSong : topTenSongs) {
4:     changeSong.setTitle("test");
5: }
6: // show the array
7: System.out.println("-AFTER--");
8: for (Song showSong : topTenSongs) {
9:     System.out.println(showSong.getTitle());
10: }
```

- Run your program.

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

- 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.

```
1: for (Song changeSong: topTenSongs) {
2:     changeSong= new Song("test");
3:     System.out.println(changeSong.getTitle());
4: }
```

The second loop remains unchanged.

- 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.
- 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.
- 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.
- 23 Modify your loop so that every third song has a discounted price of \$.99. Use the modulus operator `%` to determine every third song.
- 24 Use a second for-each loop to show titles and prices, and run your program to show correctness.

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?

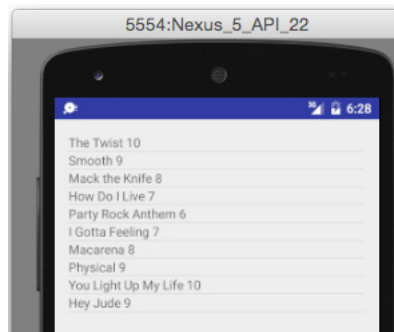
- 26 Read through  **Looping through Part of an Array**, completing the exercises. Write a new for loop to show the top five songs.

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

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

Part VI: Android List Views

`ListView`s 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`.



- 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:  
3:     @Override  
4:     protected void onCreate(Bundle savedInstanceState) {  
5:         super.onCreate(savedInstanceState);  
6:         setContentView(R.layout.activity_media_lib);  
7:     }  
8: }
```

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:

```
1: import android.app.ListActivity;
```

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

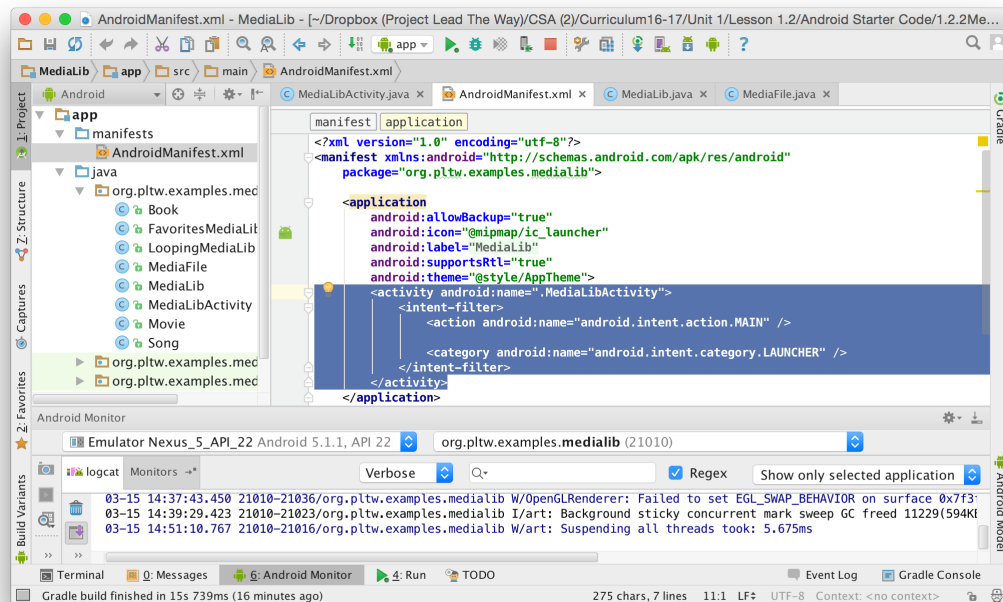
```
1: public class MediaLibActivity extends AppCompatActivity {  
2: private Song[] songs;  
3: private ListView songListView;
```

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.



- 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.

```
1: <uses-permission android:name="android.permission.READ_  
EXTERNAL_STORAGE" />  
2: <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:

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

with this code:

```
1: <ListView
2:     android:layout_width="match_parent"
3:     android:layout_height="match_parent"
4:     android:id="@android:id/list"></ListView>
```

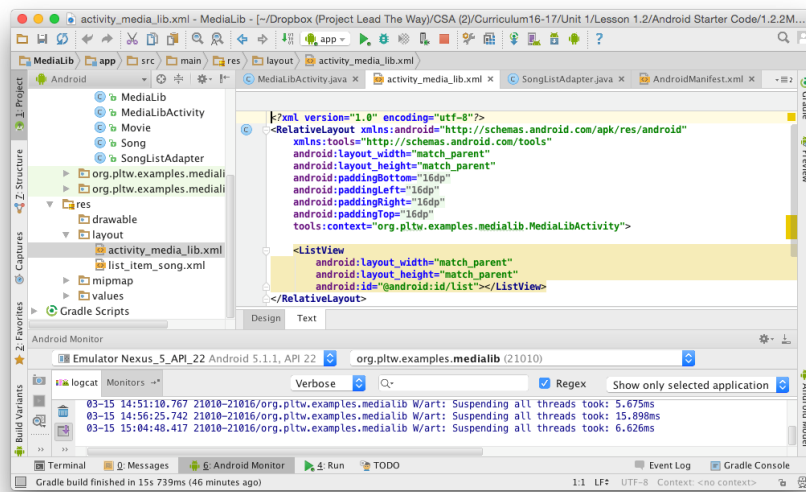
The previous step created space in the layout of your app for a `ListView`. Note that the name of this file `activity_media_lib` appears in `MediaLibActivity` within the `setContentView` method.

- 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.

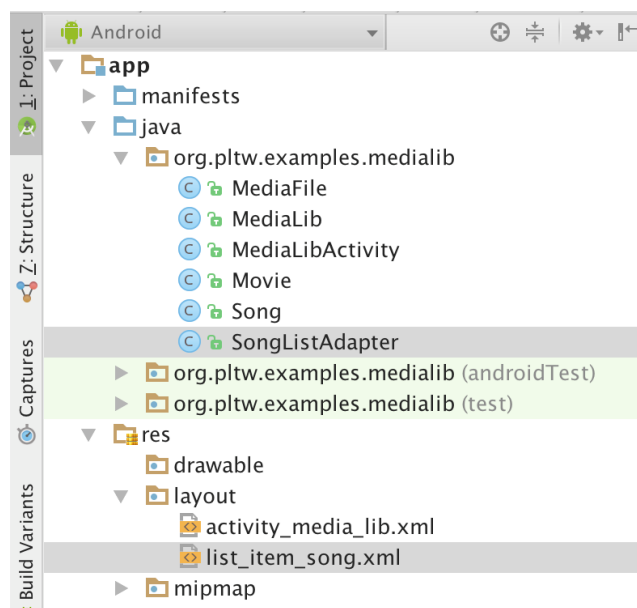
The first couple of lines of a typical `onCreate` method are:

```
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.



- 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`.

ArrayList

A class for managing a list of objects, similar to an array.

- 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!

```
1: songLibrary = new ArrayList<Song>();
2: for (int i = 0; i < numSongs; i++) {
3:     songLibrary.add(songs[i]);
4: }
5: SongListAdapter adapter = new SongListAdapter(this,
    songLibrary);
6: 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";
}
```
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.
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.