# Lab #5 – Arrays & Array Lists

Total Possible Points: 5

DUE: Wednesday, November 7th @ 11:59PM

LATE POLICY: -1 point off per day; no submissions accepted after Mon., Nov 12th @ 11:59PM

## What to Submit

* An Eclipse project in a ZIP folder named **Lab5.zip** or a RAR archive named **Lab5.rar**.

## Goals

* To write programs that use arrays and Java's ArrayList<> class
* To write programs that use loops to manipulate arrays
* To refactor a program to eliminate parallel arrays
* To write a simple class that validates user data

## Part 1 – Parallel Arrays (2 points)

In the first part of Lab 5, you will write a program that uses a pair of arrays to hold the titles and IMDb ratings of movies. One array stores the titles of the movies; the other stores the ratings.

All code will go inside a public static void main() method.

**TO-DO**: Open Eclipse and create a new Java project named **Lab 5**.

**TO-DO**: Create a new Java class named MovieArrays in a package named movies. Check the box to add a public static void main() method.

1.) (***0.5 point***) Create a String array variable named movieTitles. Use a *list initializer* to create an array with five (5) movie titles:

There Will Be Blood

Amadeus

This Is Spinal Tap

Dude, Where's My Car?

The Princess Bride

2.) (***0.5 point***) Next, create an array of five (5) integers, and assign it to an int array variable named movieRatings. Use the new keyword when creating the array.

Then, using *array subscript (index) notation* *in separate statements*, assign the following scores to the element slots of the array (remember, the index of the first array slot is “0”):

8.1

8.3

8.0

5.5

8.1

3.) (***0.5 point***) Use a traditional for() loop to **print out each movie's title and rating to the console**. Use the length property of the movieTitles array in the range test portion (i.e., the *loop* condition). The format of your output should look like this:

Dude, Where's My Car has an IMDb rating of 5.5

4.) (***0.5 point***) Using an enhanced loop, calculate the **average rating** of the movies. You’ll need to add a new local variable to keep track of the running sum of ratings. After the enhanced loop ends…

* Calculate the average and store it in a double variable. Use the length of the movieTitles array in your calculation. (DON’T JUST USE A NUMERIC LITERAL LIKE 5.0 OR YOU WILL LOSE POINTS!)
* Print out the average to the console:

The average movie rating is 7.7

|  |  |
| --- | --- |
| Forbidden | You wrote a program with two **parallel arrays** holding information about movies. That is, the first items in both arrays relate to the movie “There Will Be Blood”; the second items relate to the movie “Amadeus”, etc.  While this worked, it is **very fragile code prone to errors later down the line**. Someone could easily change the size of one of the arrays, breaking the for() loops. One finishes before the other, causing an *index out-of-bounds error*.  A less-error prone solution would have been to create a Movie class that encapsulates the movie title and rating. Instead of two parallel arrays, you would simply have an array of Movie objects.  Let’s do that in the second part of the lab. |

## Part 2 – Array Lists and Classes (3 points)

In this part of the lab, will you create a Movie class to encapsulate the properties of a movie. You will also use an ArrayList<> in a separate tester class to hold a collection of movies.

**TO-DO**: Create a second package named lists.

1.) (***0.5 point***) Add two new classes named Movie and MovieTester. Add a public static void main() method to MovieTester, but NOT to Movie.

2.) (***1 point***) Open the Movie class and add the following:

* Two private instance variables: title and rating. Choose appropriate *types*! (**HINT**: what did types of data did the arrays in Part 1 store?)
* A constructor with two (2) parameter variables, also named title and rating.
  + Use the parameters to initialize the title and rating *instance variables.*
  + Add logic to test if the passed-in rating is between 0 and 10. If it is not, it's a bad rating - set the rating to 0.
* Add public accessor methods that return the values of title and rating.

**Save your work.**

3.) (***0.5 point***) Open the MovieTester class. Add the following to the main() method:

* A new ArrayList<Movie>, assigned to a variable named movies.
* Adds five (5) new Movie objects to the movies ArrayList. Use the movie titles and ratings from Part 1. (**HINT**: See Part 1.)

*However*, change the rating of "Dude, Where's My Car?" to 100.0. Shibby!

* Using an enhanced for() loop, calculate the average rating of the movies. After the loop completes, print out the average rating to the console:

The average movie rating is 6.6

|  |  |
| --- | --- |
| Warning | Is your average movie rating different from the one you got in Part 1?  **It should be**. If the average is the same, check your validation logic in the Movie class's constructor. It should have changed the 100.0 rating to a 0. |

4.) (***1 point***) Almost done. Add the following to the main() method:

* Using a loop algorithm, **find the lowest rated movie** in movies.
  + See the Chapter 7 slides, slide 29 for inspiration.

|  |  |
| --- | --- |
| Lightbulb | Because you're dealing with objects, you can't directly compare one Movie to another. You need to get and compare their *ratings*.  **How do you get the ratings**? Now that they're encapsulated within the Movie objects, you make the Movie objects give them to you! Call the method on your Movie objects that returns a rating, then compare the ratings to one another. |

* Remove the lowest rated movie, then print what movie you removed:

"Dude, Where's My Car?" was removed from the list

* Using an enhanced for() loop, print the remaining movie titles and their ratings to the console.

The remaining movies are:

There Will Be Blood (9.0)

Amadeus (8.3)

This Is Spinal Tap (8.0)

The Princess Bride (7.7)

**DONE. Upload your Lab 5 to Moodle.**