Lab Exercise #3
Sorting and Searching Lists
Computer Science 2334

Group #: _____ Section #: _____

Members: _____

_____

_____

_____

_____

Objectives:

- To understand the use of Lists, how to create them, sort them, and search them.
- To learn how to use the *sort()* and *binarySearch()* methods of the *Collections* class.
- To demonstrate this knowledge by completing a series of exercises.

Instructions:

This lab exercise requires a laptop with an Internet connection. Once you have completed the exercises in this document, your group will submit it for grading. All group members should legibly write their names at the top of this lab handout.

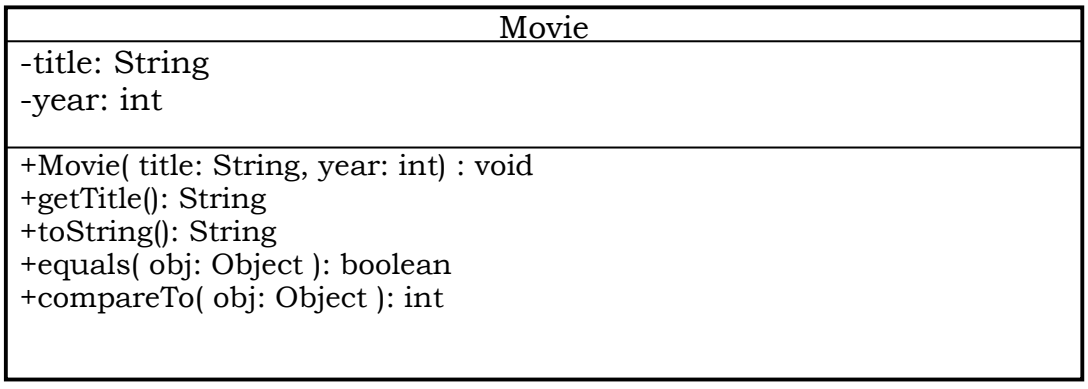
Make sure you read this handout and look at all of the source code posted on the class website for this lab exercise before you begin working.

The *sort()* and *binarySearch()* methods in the *Collections* class may be useful for completing the objectives in Project 2. The *List* interface and *Collections* class will be used extensively in later projects as well.

You have been hired to an inventory system for movies that will allow the user to add, remove, and search for movies in the collection. In this exercise, you will complete and test an initial implementation of the Movie ADT that will be the heart of this inventory system.

1. Download the Movie.java and Test.java files from the class website. You will modify these files as a part of this lab exercise and submit them when you are finished. But, before you start modifying these files, first answer the questions listed below.

2. Below is a UML diagram for the movie ADT. What is missing from this diagram? Draw in the missing components on the diagram.



3. What combination of the qualities of a movie would be a good way to uniquely identify a movie?

4. We have discussed the *Comparable* interface in class and we have seen some methods in the *Collection* interface and *Collections* class that use the *compareTo()* method. What would be a good method for determining whether one movie is less than, equal to, or greater than another movie? This is called the "Natural" ordering for the movie ADT. Describe your method below in English (you will write code for the method in a few moments).

5. As a group, complete the implementation of the movie class. Make sure you fill in the class and method header comments and declarations where information is missing. First, read the entire Movie.java file. After reading the file, add code to complete the implementation of the *toString()*, *equals()*, and *compareTo()* methods.
   Before you can search a *List* for an item, you must sort the *List* by calling the *sort()* method of the

Collections class. This method will call the *compareTo()* method of each item that is present in the *List*. Sample code that uses *sort()* to sort a list is given below.

```
List myList = new ArrayList();
myList.add( new Person( ... ) );
...
Collectons.sort( myList );
```

In order to search a *List* to find a particular object you must call the *binarySearch()* method of the *Collectons* class. This method takes as a parameter an object (called the key) that represents the object we are searching for. If *binarySearch()* finds the key in the list, it will return the index to the item in the list that matches the key, otherwise it will return a negative integer (we will talk about how this negative integer is computed in class). Sample code that uses *binarySearch()* to search for an item in a list is given below.

```
Person key = new Person( ... );
Person result;
int index;

index = Collections.binarySearch( myList, key );

if( index > -1 )
{
   result = List.get( index );
   System.out.println( result.toString() + " resides at index " +
      index + " in the list." );
}
else
{
   System.out.println( key.toString() + " was not found in the list!" );
}
```

6. Complete the implementation of the *main()* method in the Test.java test program in order to test your movie ADT. Follow the steps specified below to finish the *main()* method.

   (a) Read through the entire listing Test.java.
   (b) Analyze the code to create eight objects of type Movie, initialize them, and add them to the list *movies*. This code has already been provided, however you are expected to understand how it works.
   (c) Analyze the code that will print out the list of movies. You need to understand how this works.
   (d) Add the code necessary to sort the list of movies.
   (e) Provide the code that will print out the sorted list of movies.
   (f) Analyze the code necessary to search the list of movies for the movie named "key" that is created in the base code. You need to understand how this works.
   (g) Add code that prints out the results of this search.
   (h) Complete the code that tests the *equals()* method of the Movie class by adding in the missing code to the main method, as specified in the base code.

7. Submit this lab electronically by transferring the Movie.java and Test.java files to codd.cs.ou.edu and

save them in a folder named lab3. Then ssh into codd.cs.ou.edu and use the following commands to submit your lab.

```
cd lab3
/opt/cs2334/bin/submit cs2334-<section #> lab3 Movie.java Test.java
```

where <*section #*> is replaced by the section number for your lab section (011, 012, or 013).

8. Turn in this lab handout to your lab instructor.