Instructions for Lab Exercise 3
Sorting and Searching Lists
Computer Science 2334
Due by: Friday, February 6, 2015, 4:00 pm CST
This lab is a group exercise. Students must complete this assignment with at least one partner.

Learning Objectives:

1. To understand the use of **ArrayList**s, how to create them, sort them, and search them.
2. To learn how to use the `sort()` and `binarySearch()` methods of the **Collections** class.
3. To demonstrate this knowledge by completing a series of exercises.

The `sort()` and `binarySearch()` methods in the **Collections** class will be useful for completing the objectives in course projects. The **List** interface and other methods of the **Collections** class will be used extensively in later projects as well.

Instructions:

This lab exercise *requires your group to have a laptop with an Internet connection*. Your group will work collaboratively to complete the exercises described in these instructions. Some exercises will require you to fill in the associated answer sheet with answers while others will require you to work on a Java project in Eclipse. You will submit a PDF copy of the completed answer sheet as well as an Eclipse archive of the completed Java project.

1. Download and save the answer sheet (`Lab3-CS2334-answerSheet.odt`) and fill in the names of all group members where indicated at the top of that document.

2. Download the `Lab3-eclipse.zip` project archive from the class website and import it into Eclipse. This archive contains the `Character.java` and `Lab3Driver.java` files. You will modify these files as a part of this lab exercise and submit the project archive when you are finished.

3. On the answer sheet, place a UML diagram for the **Character** class. (You may create this diagram using specialized UML software or general drawing tools, draw it by hand and scan it in, etc., so long as it is clearly legible on the submitted PDF answer sheet.)

4. Generics are a way to use general classes to deal with objects of specific types. In the **Character** class, the names of the personality traits of each character are stored in an **ArrayList**. **ArrayList**s are general in that they can hold objects of any type. However, we should specify which types of objects *this particular* **ArrayList** will hold. What is an appropriate type for an **ArrayList** of character traits? Replace the "raw type" **ArrayList**s in the **Character** class with generic **ArrayList**s by putting this type between less than and greater than symbols as if they were parentheses. That is, if you said that an appropriate type for a personality trait is **SnickerDoodle**, then you would replace each instance of `ArrayList` with `ArrayList<SnickerDoodle>`. (Note that **SnickerDoodle** is not actually an appropriate type for personality trait.)

5. We have briefly discussed the **Comparable** interface in the class and the associated `compareTo()` method. What would be a good way of determining whether one character should be listed before or after another character? This is called the "Natural" ordering for the class. On the answer sheet, describe your method in English (you will write code for the method in a few moments). Make sure that your method for comparing takes into consideration all aspects of the character relevant to ordering.

6. As a group, complete the implementation of the **Character** class. Make sure you fill in the class and method header comments where information is missing. First, read the entire `Character.java` file. After reading the file, add code to complete the implementation of the `toString()` and `compareTo()` methods. Why are getters and setters not required for this particular class? (Put your answer on your answer sheet.)

Note:
Before you can search a **List** using `binarySearch()`, 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()` is given below.

```
Collections.sort(list);
```

In order to search a **List** to find a particular object you must call the `binarySearch()` method of the **Collections** 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. (In class at a later date, we will talk about how this negative integer is computed.) Sample code that uses `binarySearch()` to search for an item in a list is given below.

```
int index = Collections.binarySearch(list, key);
```

Yep, that is it. One line.

Instructions, continued:

7. Complete the implementation of the `main()` method in the `Lab3Driver.java` test program in order to try out your **Character** class. Follow the steps specified below to finish the `main()` method.

a) Read through the entire listing of `Lab3Driver.java`.

b) Add code to create seven additional **Character** objects, initialize them, and add them to the list `characters`. The code to create and add one **Character** has already been provided. You need to add code for seven more, for a total of eight.

c) Analyze the code that will print out the list. You need to understand how this works.

d) Add the code necessary to sort the list.

e) Provide the code that will print out the sorted list.

f) Analyze the code necessary to search the list for the "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.

8. Ensure that there are no warnings generated for your code. **Do not suppress warnings.** Fix your code so that warnings are not necessary.

9. For bonus points, you may add code to the end of `main` to print the list again but this time use a `foreach` loop rather than a list iterator.

10. Export your completed answer sheet to PDF and submit it to the D2L dropbox for Lab 3 on or before **4:00 pm, Friday, February 6, 2015**.

11. Export the project archive following the steps given in the Submission Instructions and submit it to the D2L dropbox on or before **4:00 pm, Friday, February 6, 2015**.