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.

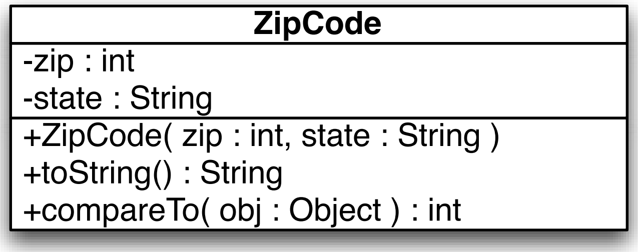
Make sure you read this handout and look at all of the source code posted on the class website for this lab exercise before you begin working.

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

Your task is to design and build an inventory system for zip code information that will allow the user to add, remove, and search for zip codes in the collection. In this exercise, you will complete and test an initial implementation of the zip code class that will be the heart of this inventory system.

1. Download the Lab3.zip project archive from the class website. This archive contains the ZipCode.java and Driver.java files. You will modify these files as a part of this lab exercise and submit the project archive 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 zip code class. Redraw the diagram including any missing components. Make sure to highlight any changes you make. Note that you do not need to include getters and setters.


ZipCode
-zip : int
-state : String
+ZipCode(zip : int, state : String)
+toString() : String
+compareTo(obj : Object) : int

3. 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 item is less than, equal to, or greater than another item? This is called the "Natural" ordering for the class. Describe your method below in English (you will write code for the method in a few moments).

4. As a group, complete the implementation of the zip code class. Make sure you fill in the class and method header comments where information is missing. First, read the entire ZipCode.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?

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 (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.

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

6. Complete the implementation of the *main()* method in the Driver.java test program in order to test your zip code class. Follow the steps specified below to finish the *main()* method.

   (a) Read through the entire listing of Driver.java.
   (b) Analyze the code to create eight ZipCode objects, initialize them, and add them to the list *zipcodes*. 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. 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.

7. Submit the project archive following the steps given in the Submission Instructions by 9:00pm.

8. Turn in this lab handout (with completed answers) to your lab instructor.