Computer Science 501
Data Structures & Algorithms
The College of Saint Rose
Fall 2014

Lab 6: Sorting and Comparators

Due: 6:00 PM, Tuesday, October 14, 2014

This week's lab focuses on sorting. You will take the next step toward our upcoming empirical analysis of sorting algorithms in a practice program, then get an introduction (or reintroduction) the power of Java's `Comparator` interface as it applies to writing a generalized sorting method. You will extend the functionality of an existing class using inheritance, you will implement a simple sorting procedure within this extension, and you will learn about `Comparator`s, which provide a more flexible mechanism for ordering objects than the `Comparable`s we have seen in class.

You may work alone or in a group of two or three on this lab. Of course, collaboration with your partner is unrestricted. You may discuss the lab with your classmates and give and receive some help, but your submission must be your own work (or that of you and your teammates, if you choose to form a group).

Getting Set Up

To get your BlueJ environment set up for this week's lab assignment, start BlueJ and choose "New Project" from the "Project" menu. Navigate to your folder for this course and choose the name "Lab6" (no spaces) for the project.

Create a document where you will record your answers to the lecture assignment and lab questions. If you use plain text, call it "`lab6.txt`". If it's a Word document, you can call it whatever you'd like, but when you submit, be sure you convert it to a PDF document "`lab6.pdf`" before you submit it.

Lecture Assignment Questions

We will usually discuss these questions at the start of class on the lab due date, so no credit can be earned for late submissions of lecture assignment questions.

? LA Question 1:

Bailey Exercise 6.13, p. 146. (2 points)

? LA Question 2:

Bailey Exercise 6.18, p. 146. (4 points)

Practice Programs

For this week's practice program(s), you will be taking another step toward an empirical analysis study of sorting algorithms.

Your program should be able to gather timings for sorting algorithms operating on arrays of `int`. It should have options to set the array size, the number of trials (to improve timing accuracy), the ability to generate initial data that is sorted, nearly sorted, completely random, and reverse sorted. Design your program to make it easy to implement a variety of sorting algorithms. Include the ability to count basic operations (number of comparisons and/or number of swaps) as well as to generate timings. You will need the ability to generate and report tabular data to show how your sorting algorithms perform on different sizes and distributions of data.

For this week, you should implement just three "naive" sorting algorithms in your program(s):

- bubble sort
- selection sort
- insertion sort

Tips, Tricks, Precautions, and Suggestions

- Use command line parameters rather than prompts, as this makes it much easier when running many (likely hundreds or thousands) of trials to generate timing results. `args[]` has what you need! If you don't know how to run with command-line parameters inside your IDE, run your Java program at the command line. That's what you'll want to do when generating timing results anyway.
- Have one big program rather than lots of little ones. This will help you avoid repeated code as you implement each of the sorting algorithms within the same framework.
- Be careful that you don't reuse an array of values for multiple runs, since all but the first could end up having already-sorted data as input.
- A simple tabular format of output will help you manage the creation of tables and/or graphs that you'll need later. Something like

  ```
  10000 bubble random .034693
  ```

  might indicate for an input size of 10,000, using a bubble sort on random input took .034693 seconds.

Programming Assignment

We will do the laboratory at the end of Chapter 6 in Bailey.

Please note the following clarifications, modifications, and explanations relating to the lab procedure outlined in the text:

- In step 1, you are asked to create an extension of `structure.Vector` called `MyVector`. Since we are using the generic version of the structure package, the class header for `MyVector` should look something like this:

  ```
  public class MyVector<T> extends structure5.Vector<T>
  ```

  Keep in mind that as an extension of `structure5.Vector`, methods of `MyVector` will have access to instance variables and methods declared as `protected` in the `structure5.Vector` implementation. Make good use of this fact!

  Important Note: The `elementData` array in `structure5.Vector` is declared as `private` rather than `protected` for type safety reasons. This means, unfortunately, that `MyVector` will not be able to access the array directly. Fortunately, `Vector` has mutator and accessor methods that are (almost) as good as direct access to the array.

  Another Important Note: You do not need to copy or rewrite `Vector.java`! When you extend the existing `Vector` code, it will inherit all of the constructors and methods. You're just adding one method.

- In step 2, you are to write a `sort` method. The structure of the code will be very similar to what you have seen in our class and text examples, but you will need to modify it to use `Comparator`s instead of base types or `Comparable`s and to operate on the contents of your instance of `MyVector`.

- Be sure to test your `sort` method in `MyVector` thoroughly before going on to part 3 of the lab assignment.

- For part 3 of the lab procedure, write two applications. They should each work on a different data file and each should perform more than one "interesting" sort process. Both will use the same `MyVector` class, but each application will call its own class for encapsulating the data objects you are working with, and a number of `Comparator`s to sort the data in different ways. You should have a total of at least 5 unique `Comparator`s in your submission.

  You may choose from the data files I have provided in the "`labs/comparators`" directory in the class shared area. See the `README` file there for more information. You may also use some other data that you find interesting.

? Question 1:
Answer Thought Question 1 on p. 147-148 of Bailey. (3 points)

? Question 2:
Answer Thought Question 2 on p. 148 of Bailey. (3 points)

Submitting

Before 6:00 PM, Tuesday, October 14, 2014, submit your lab for grading. There are two things you need to do to complete the submission: (i) Copy your file with the answers to the lecture assignment and lab questions into your project directory. Be sure to use the correct file name. If you prepared your answers in Word, export to a PDF file and submit that. (ii) Upload a copy of your lab (a `.7z` or `.zip` file containing your project directory) using Submission Box at `http://sb.teresco.org` under assignment "Lab6".

Grading

This assignment is worth 60 points, which are distributed as follows:

Feature	Value	Score
LA Question 1 (6.13)	2	
LA Question 2 (6.18)	4	
General sorting algorithm test framework	5	
Java code for specific sorting algorithms	12	
`MyVector` correctness	12	
`Comparator`s correctness	5	
`MyVector` and `Comparators` design and style	5	
`MyVector` and `Comparators` documentation	6	
Sorting applications	3	
Question 1: Thought Question 1	3	
Question 2: Thought Question 2	3	
Total	60	