CMPSC 112
Introduction to Computer Science II
Fall 2016

Laboratory Assignment Two: Performance Evaluation of Array and Iteration Use

Introduction

The Java programming language provides facilities for you to store many values of the same type and to "iterate" through those values using iteration constructs like a `for` loop. In this laboratory assignment, you will review the use of arrays and iteration constructs in the Java programming language. You will also practice the debugging of programs that use arrays and `for` loops. Finally, you will continue to conduct experiments to evaluate the performance of Java programs.

Reading Assignment

To start your review of the key features of the Java programming language, please study the material in Chapter 1 of the textbook, paying particularly close attention to the content about iteration and input and output in Sections 1.5 and 1.6. Please also review the slides that we have discussed during our recent class sessions. If you have questions about this reading assignment or the material that was presented in class, then please see the instructor or a teaching assistant. If done appropriately, you may post your question to the `#laboratory` channel of our Slack team.

Compiling, Running, and Repairing a Java Program

This laboratory assignment is inspired by programming project P-1.26 on page 57 of your textbook. Instead of requiring you to start this assignment "from scratch" I have begun an implementation and placed it in the "share" repository for our couurse. To start this assignment, please use your terminal window to navigate to the share repository and then run the "`git pull`" command. Now, you should see that you have three source code files, with a `main` method located in the class called `SentencesReverser`. Please use the Java compiler and virtual machine to run this program in your terminal window. At this point, you should see that it produces the following output:

```
Strings, Wrappers, Arrays, and Enum Types
Increment and Decrement Operators
Type Conversions
An Example Program
Exception in thread "main" java.lang.NullPointerException
        at Sentences.printSentences(Sentences.java:33)
        at SentencesReverser.main(SentencesReverser.java:12)
```

This output clearly indicates that there is a defect in at least one of the three Java class files that I have provided to you! Before going further with this assignment, please find and fix this defect. The output provided above should draw your attention to line 33 of the `printSentences` method in the `Sentences` class. Once you have repaired the Java program, you should carefully study the source code of the three Java classes so that you better understand how the program works and how the three classes help to solve programming project P-1.26. You should also try to conceptualize what an instance of the `Sentences` class looks like in the computer's memory. Once you better understand the design and implementation of the `SentencesReverser` program, then you are ready to extend it with the feature that is requested on page 57 of your textbook.

As it is currently implemented, `SentencesReverser` inputs a list of sentences from a text file called `sentences.txt` and then displays them to the terminal window. However, the description of the programming project requires you to "write them to standard output in reverse order" with the further explanation that "each line is output in the correct order, but the ordering of the lines is reversed". This means that first line in the `sentences.txt` file should now be the last line output by the `SentencesReverser` program. If you look into the source code of the `Sentences` class you will see that there is a `reverseSentences()` method that is not yet implemented. Please add your own implementation of this method, testing it with different inputs to ensure that it is working correctly. You may talk with a teaching assistant or the course instructor if you have questions.

Evaluating the Performance of Iteration

The final version of the `SentencesReverser` program operates in several distinct phases: first, it inputs sentences from the `sentences.txt` file; second, it reverses all of the sentences stored in the `private Sentence[] sentences` instance variable; third it outputs all of the sentences by calling the `printSentences` method. For the last part of this laboratory assignment, you should modify your program so that it collects timing information for each of these three phases. Please note that, since these phases occur "inside of" the Java program, we cannot employ the timing method that you learned in class. That is, running the program with `/usr/bin/time` in front of the call to the Java virtual machine will not allow us to collect timings at the desired level of granularity. Instead, you will need to surround calls to the methods that you want to profile with calls to the `currentTimeMillis()` method in the `java.lang.System` class, as shown in this code segment.

```
long beforeRead = System.currentTimeMillis();
sentences.readSentencesFromFile();
long afterRead = System.currentTimeMillis();
System.out.println("Read time: " + (afterRead - beforeRead));
```

At this point in the assignment, you should follow the pattern in the previous code segment to time the three phases of the `SentencesReverser` program. To accomplish this task, you may want to declare a total of six variables that will respectively track the time before and after the execution of the methods subject to performance evaluation. For instance, to time the `readSentencesFromFile()` method the previous code segment declared the variables called `beforeRead` and `afterRead` and then used the equation "`afterRead - beforeRead`" to calculated the time that elapsed when running the `readSentencesFromFile()` method.

Once you have added all of the source code needed to record the execution time of the three phases of the `SentencesReverser` program, you are ready to conduct an experiment to evaluate the performance of iteration in a Java program. As we have learned during our classroom discussions, you must carefully design your experiment so as to ensure that you are measuring what you desire to study. For instance, you should conduct your experiment by varying the number of sentences in the `sentences.txt` file and then running multiple trials of the `SentencesReverser`. Then, you will need to record the program's output into a data table organized according to the number of sentences in the text file and the timings for the three phases (i.e., read, reverse, and print).

Carefully Review the Honor Code

The Academic Honor Program that governs the entire academic program at Allegheny College is described in the Allegheny Academic Bulletin. The Honor Program applies to all work that is submitted for academic credit or to meet non-credit requirements for graduation at Allegheny College. This includes all work assigned for this class (e.g., examinations, laboratory assignments, and the final project). All students who have enrolled in the College will work under the Honor Program. Each student who has matriculated at the College has acknowledged the following pledge:

> I hereby recognize and pledge to fulfill my responsibilities, as defined in the Honor Code, and to maintain the integrity of both myself and the College community as a whole.

It is understood that an important part of the learning process in any course, and particularly one in computer science, derives from thoughtful discussions with teachers and fellow students. Such dialogue is encouraged. However, it is necessary to distinguish carefully between the student who discusses the principles underlying a problem with others and the student who produces assignments that are identical to, or merely variations on, someone else's work. While it is acceptable for students in this class to discuss their programs, data sets, and reports with their classmates, deliverables that are nearly identical to the work of others will be taken as evidence of violating the Honor Code.

Summary of the Required Deliverables

This assignment invites you to submit both a printed and an electronic version of these deliverables:

1. Using the Javadoc standard, a commented version of the three Java classes.
2. The output from running the final version of the `SentencesReverser` program.
3. A one-page report that responds to these prompts about the provided Java classes:
   (a) What is the mistake that you found and fixed in the three provided Java classes?
   (b) What is the relationship between the three Java classes used to solve project P-1.26?
   (c) What does an instance of the `Sentence` class look like in the computer's memory?
   (d) How "expensive" is the strategy that you adopted for reversing an array?
4. A one-page experiment report that comments on the timing experiments that you conducted. Your experiment report should include, at minimum, the following deliverables:
   (a) An overview of how you collected timing information for the phases in the `SentencesReverser`.
   (b) A description of the steps that you took to conduct the experiments and collect timings.
   (c) Tables of data that record the number of sentences subject to reversal and the timings.
   (d) A list of the challenges you faced when creating the program or running experiments.

Before you turn in this assignment, you also must ensure that the course instructor has read access to your personal Bitbucket repository that is named according to the convention `cs112F2016-<your user name>`. Please see the course instructor if you have questions about this assignment.