

Distributed: Friday, Feb. 23, 2007

Due: At the **start** of class, Friday, March 2, 2007

Goal: To review timing from COSC 182, updated for Java 5, and with more care in your analysis.

Objective: To compare Java's speed in using Vector with its speed in using arrays.

This is an **individual** assignment, not a team assignment. Be sure to document any and all sources that you use in completing this assignment! This lab does not repeat style points or other things covered by class discussion, previous courses, or postmortems of previous labs, but expects that you will use what you have learned before.

Reading before you begin: You have read through page 140 of your text except for the sections that you were asked to omit (5.2.2 and 6.4), and all handouts (including this one). Although this assignment is based on Laboratory 5.5 from your textbook, it is different in several ways. Do this assignment; do not do the textbook version. But do read Section 5.5 of your text. It is part of your reading assignment, and might provide hints for some of the parts of this assignment.

The assignment:

1. Describe the machine that you are using. Use the same machine for all your experiments.
2. Write a short driver program to measure the time that elapses when an **empty** for loop counts to one million. Print out the elapsed time, as well as the per-iteration time. Adjust the number of iterations of the loop so that the (total) elapsed time falls between, say, one-hundredth of a second (10 milliseconds) and one-tenth of a second (100 milliseconds). Use `System.nanoTime()` from Java 5, which is documented at [http://java.sun.com/j2se/1.5.0/docs/api/java/lang/System.html#nanoTime\(\)](http://java.sun.com/j2se/1.5.0/docs/api/java/lang/System.html#nanoTime()) because it is more accurate than `System.currentTimeMillis()` which you likely used in COSC 182. The code on page 122 of your text gives a start at this driver code. You are not being graded on object-oriented programming in this assignment! You will only have a driver program; no other objects are needed other than `Vector`. Do not use `structure.Vector`; use one of the versions of `Vector` that supports generics: `structure5.Vector` from Bailey or `java.util.Vector` from Sun Microsystems.
3. Measure the time it takes to assign a `String` to an array. Take enough samples so that you can graph the result for different length arrays.
4. Measure the time it takes to assign the same number of `Strings` to a `Vector<String>`. Take enough samples so that you can graph the result on the same set of axes as the graph in #3. Do graph both times.
5. Answer the following five questions.
 - a. Why did I ask you to use the same machine for all your runs? Be as specific as possible, since two machines could be "identical," could they not?
 - b. What did you do to guarantee that your total experiment time lasts between 10 and 100 milliseconds?
 - c. Are there compiler and run-time switches that you could have used to make your results more predictable? [Compare p. 123, Question #1] For example, typing these four bulleted things from the command line will give you help on compile and run time options. Why might you want to run using `java -Xint?`
 - `javac -help`
 - `javac -X`
 - `java -help`
 - `java -X`
 - d. Does the graph of the time have a "kink" in it where the `Vector` may be resizing itself?
 - e. **Does it take longer in general to assign a `String` to a `Vector<String>` than to add the `String` to an array of `Strings`?**
6. Put your driver and your lab report on drive q:.