CMPSC 112
Introduction to Computer Science II
Fall 2016

Laboratory Assignment Eleven: Performance Evaluation of Java Collections

Introduction

The Java Collections Framework provides an implementation of a wide variety of data containers, such as the `ArrayList`, `LinkedList`, and the `Vector`. Each of these different containers are implemented in a different way and thus exhibit different trade-offs in their performance. For instance, the `ArrayList` is "backed" by an array while the `LinkedList` uses nodes that are connected together by references. How do you think that these different implementation choices will influence their performance in different usage scenarios? To get started on this assignment, you should review the content in Sections 7.1 through 7.5 of your textbook, with a focus on the content in Section 7.5. To learn more about the classes provided by the Java Collections Framework, you should also read the JavaDoc documentation of all the relevant classes and find online sites that explain their performance trade-offs. What ideas do you have for evaluating the performance of these containers?

Using the Benchmarking Framework

To start this laboratory assignment, you should return to the `cs112F2016-share` Git repository and type the "`git pull`" command in the terminal window. Now, you should have a `lab10/` directory that you can explore further. Once again, please make sure that you can find the source code in this new directory and you understand why the directories in the assignment are structured the way that they are. Next, you should use GVim to study the source code in the `build.xml` file. As in the past assignments, when editing a Java program you can type "`ant compile`" in your terminal and it will compile all of the Java classes and save the bytecode in the correct subdirectories inside of the `bin/` directory. Please see the instructor if you cannot get this to work.

One aspect of a container's performance is how efficiently it supports adding data to the front of the list after the list has already been populated with data. After you have used the build system to compile the `ArrayListBenchmarks` and the `LinkedListBenchmarks` classes, you can run them in the terminal window through the build system. What command do you have to type to run these benchmarks? How did you learn which command to type? These two benchmarks should output a few lines of empirical results. What is the meaning of this output? Once you understand the output better, please carefully run the benchmarks multiple times, recording data values and noting important trends as you continue. Why are these performance results evident in your output?

Adding New Performance Evaluation Benchmarks

If you review the source code and the build system, you will notice that a benchmark for the `Vector` class is currently missing. As such, your next task for this assignment is to create a new class, called `VectorBenchmarks` that follows the same format as the two provided benchmarks. Additionally, you should add a new rule to the `build.xml` file so that you can easily run the benchmark by typing "`ant VectorBenchmarks`" in the terminal window. Does your new benchmark work correctly?

Of course, adding data to the front of an already populated container is only one operation that is likely to occur in practice. For the next phase of this assignment, you should again review your notes about the Java collections framework and conceptually design a completely new benchmark. Your benchmark should have a clear purpose and should be added, as one or more methods, to the existing Java classes for the first benchmark. As you are implementing your new benchmark, please make sure that you have a way to evaluate performance for all three of the Java containers that we are studying in this laboratory assignment (e.g., the `ArrayList`, `LinkedList`, and `Vector`). Now, you should repeat this entire process again so that your final benchmarking framework contains three separate benchmarks. Please see the course instructor if you have questions about this process.

Experimentally Evaluating the Efficiency of Java Collections

Once you have finished implementing your benchmarks and testing them to ensure that they work correctly, you must plan out and conduct a series of experiments to evaluate the efficiency of the Java Collections Framework. One aspect on this experimentation effort is identifying all of the parameters of your benchmarks. For instance, the `ArrayListBenchmarks` class has two configurable parameters: `initialListSize` and `addToFrontNumber`. After you have listed all of the parameters, you should determine what values you will assign to them, perhaps following the doubling strategy to effectively pick your values. Now, go ahead and conduct all of your experiments, producing tables of data and/or graphs to highlight your results. Finally, you should write a detailed experiment report that explains the purpose of each benchmark, your hypothesis for the results, a justification for each stated hypothesis, the analysis of your empirical results, and suggestions for future work.

Summary of the Required Deliverables

This assignment invites you to submit a signed and printed version of the following deliverables:

1. A complete description of the features provided by the Java Collections Framework.
2. The properly commented version of every benchmark that you have implemented.
3. The properly commented version of the build system that runs every benchmark.
4. The output from several separate runs of your benchmarks, demonstrating all key features.
5. A written report that provides a response to all of the questions posed in this assignment.
6. A written report that follows all of the guidelines outlined in the previous section.
7. A reflective commentary on the challenges that you faced when implementing the benchmarks.
8. A reflective commentary on the challenges that you faced when doing the empirical analysis.

Along with turning in a printed version of these deliverables, you should ensure that everything is also available in the repository that is named according to the convention `cs112F2016-<your user name>`. Please note that students in the class are responsible for completing and submitting their own version of this assignment. While it is acceptable for members of this class to have high-level conversations, you should not share source code or full command lines with your classmates. Deliverables that are nearly identical to the work of others will be taken as evidence of violating the Honor Code. Please see the instructor if you have questions about the policies for this assignment.