Dashboard

[SP19 COMPSCI 400 001](#)

[p3b Performance Analysis](#)

[Skip To Content](#)

[Dashboard](#)

- VU PHAM Account
- [Dashboard](#)
- Courses
-

Calendar

-

37 unread messages.37
Inbox

- History
-

Help

Close

p3b Performance Analysis

Performance Analysis of Data Structures

Announcements | Overview | Specifications | Files | Steps | Submission

Announcements (See Piazza for answers to FAQs)

Corrections and clarifications regarding this programming assignment will be found below.
Search Assignment pages and then Piazza before asking for best results.

- **3/11 Program assigned.**
  - **https://docs.oracle.com/javase/8/docs/api/java/util/TreeMap.html** (Links to an external site.)

Overview:

For this project, you will be implementing a small program to analyze the performance of your hash table against Java's built-in TreeMap. TreeMap (Links to an external site.) is a known and in-built data structure of Java.

To analyze the performance, you need to write a small program that performs the same operations on both your custom **HashTable** class and Java's **TreeMap** class. You will be using **Java Flight recorder** to create a program profile, and Java Mission Control to analyze the profile information. Both are open source and packaged with Oracle JVM's (The lab machines have this installed).

For this assignment, you are required to:

1. Write **MyProfiler.java**, a program that will be profiled to compare the your **HashTable** and Java's **TreeMap**.
2. Use make and Makefiles (Links to an external site.) **(use our Makefile - no need to define your own)**
3. Run your program using Oracle Java Flight Recorder (Links to an external site.) to profile your **MyProfiler** program.
4. Use Oracle Java Mission Control (Links to an external site.) (jmc) to analyze the generated **my_profile.jfr** data.
5. Answer the questions in **conclusions.txt** (Links to an external site.) file.
6. Take screenshots of the relevant parts of your profile data as viewed from **Java Mission Control**
7. Submit your files to p3b Performance Analysis (Links to an external site.)

All files must be named correctly (case-sensitive). You may define and submit other package level (not public or private) classes as needed and you may add private members to your classes.

The goal for your **HashTable** was to build a searchable data structure that achieves constant time O(1) for lookup, insert, and delete operations with comparable performance to Java's built-in **TreeMap** type. This assignment attempts to determine if your hash table achieved that goal.

Ensure that your hash table implementation works correctly prior to analyzing its performance against Java's **TreeMap** class.

Files

Note: updates to these files are possible, and will be posted in announcements on this page.

In the p3b.zip (Links to an external site.) file:

- (Links to an external site.)**conclusions.txt (Links to an external site.) (do answer the questions after profiling and analyzing, do SUBMIT)**
- DataStructureADT.java (Links to an external site.) (provided interface - do NOT EDIT or SUBMIT)
- DuplicateKeyException.java (Links to an external site.) (provided class - do NOT EDIT or SUBMIT)
- HashTableADT.java (Links to an external site.) (provided interface - do NOT EDIT or SUBMIT)
- **HashTable.java (Links to an external site.) (starter class - do SUBMIT - you may improve from p3a)**
- HashTableTest.java (Links to an external site.) (starter for JUnit test class - do EDIT if you like, but you will not need to SUBMIT this file)
- heap_status.jfc (Links to an external site.) - (configuration file for Java Flight Recorder - do not submit)
- IllegalNullKeyException.java (Links to an external site.) (provided class - do NOT EDIT or SUBMIT) (Links to an external site.)
- junit-platform-console-standalone-1.3.2.jar (Links to an external site.) (for running JUnit5 tests -- do not submit)
- KeyNotFoundException.java (Links to an external site.) (provided class - do NOT EDIT or SUBMIT)
- Makefile (Links to an external site.) (a command-line build make utility file - you may use, edit, or ignore - do not SUBMIT)
- **MyProfiler.java** (Links to an external site.) **(starter class - do write your profiler code and do SUBMIT)**
- **(Links to an external site.)**SampleProfilerApplication.java (Links to an external site.) (provided to help you complete MyProfiler.java - do NOT SUBMIT) **(Links to an external site.)**

Other files you will need to create and submit with your code:

- **screenshot_001.png** (screenshot relevant parts of java mission control - DO SUBMIT)
- **screenshot_002.png** (screenshot relevant parts of java mission control - DO SUBMIT)

MyProfiler

This is the program that you will profile to determine relative performance between your HashTable and Java's TreeMap structures. The program must perform a "bunch" of inserts, lookups, and removes to the data structures.

Keep in mind, you are trying to figure out which data structure performs best. Because modern computers are so fast, you will likely need to add a lot of items to see enough difference.

Given the complexity analysis of different lookups (single item vs range of values) for the two data structures is different, you will also want to experiment with different lookup operations. Consider to how to lookup many individual values, and many different ranges of values from each structure.

Be scientific and iterative

https://www.sciencebuddies.org/science-fair-projects/science-fair/steps-of-the-scientific-method (Links to an external site.)

1. Make an hypothesis.
2. Design an experiment (a.k.a. write code, a single test, or set of tests).
3. Run your experiment (code) and record the results.
4. Learn how to interpret your results.
5. Compare your results against your predictions.
6. How well do you understand your data? your results? Can you predict results?
7. Repeat above to collect more data and refine your hypotheses and come to some conclusions.
8. Record your conclusions.

Java Flight Recorder

Read Oracle Java Flight Recorder (Links to an external site.)

The commands for compiling and running Java Flight Recorder are long and error prone to type. UNIX has a utility called **make** that will execute the instructions found in a file. These files provide a convenient way for programmers to save commands for easy (and repeated) execution.

Read make and Makefiles (Links to an external site.)

CAUTION: the file format of a **Makefile** is quite exact and a little unique. You must use the "tab" character and not spaces on the indented lines. We have provided a **Makefile** for you for program p3b. You may edit this Makefile as you learn more about makefile formats and wish to make other targets.

Use make to run the sample application from the command line:

1. Navigate to your project directory
2. Your code must compile and you must have our provided files and our **Makefile** in the current working directory
3. Type this make command to run the sample_profiler program.

```
> make sample_profiler
```

4. Type this make command to run your MyProfiler program.

```
> make my_profiler
```

Java Mission Control

Now, you are ready to view and start analyzing the results of your program. Read about and use Oracle Java Mission Control (Links to an external site.) to do this.

Specifications (requirements)

- Write Professional Source Code (Links to an external site.)
- **Write the MyProfiler** class
  - The **MyProfiler** class contains methods for inserting and retrieving data from hash table and tree map
  - You are required to instantiate **MyProfiler** class in the main method with the appropriate data type
  - In the insert method, you will need to insert into both the hash table and tree map
  - Similarly in the retrieve method, you need to get from both the hash table and the tree map

- The profile class takes one argument **<num_elements>**, you need to insert and retrieve these many number of elements from both hash table and tree map
- Instructions to perform the profiling are provided here: Oracle Java Flight Recorder (Links to an external site.)
- Instructions to view and analyze results are provided here: Oracle Java Mission Control (Links to an external site.)
- Once, you complete the profiling, answer the questions asked in the file: **conclusions.txt**

Steps

1. Read the entire assignment.
2. Review grading rubric. Note: the rubric is subject to some changes.
   You may use the Java development environment of your choice. *However, all programs must compile and run using JUnit5 and Java 8 from the Linux workstations in the lab computers for grading*.
3. Create a new **p3b** project for your work.
4. Add the provided source files into your project folder and refresh.
   1. Download the **p3b.zip** file
   2. Extract all files from **p3b.zip** into your top level project folder.
   3. If the extracted files are in sub-directory **p3b**
      1. move the files "up" to the project level folder
      2. remove the now empty **p3b** folder
5. Refresh the Eclipse project
   1. Click on Project name in Project Manager
   2. Click File->Refresh
6. Get your project to compile
   1. Add unimplemented stub methods. Don't implement just yet. Just get it to compile first.
   2. Review the provided classes.
   3. Add JUnit5 Library JUnit compile and run JUnit Tests from the command-line on CS workstations (Links to an external site.)
7. Run the SampleProfilerApplication from the command line as instructed above
8. Now, you're ready to write the code and complete the assignment.
9. Review Specifications and assignment for more details.

Caution: If you are not using the lab computers to develop your program, make sure that you compile and run your program to ensure that it works on the Linux lab computers.

Submit your work to: p3b Hash Table Performance Analysis (Links to an external site.)