Tafita Rakotozandry
CS150 Lab
09/05/2020

Lab 3 Report

INTRODUCTION

The goal of this lab is to familiarize with generics using ArrayList. It also initialises the basics of sorting algorithms: selection sort. As part of the requirement for this lab, a class called randomStringcontainer was created. This class contains the following method: AddToFront, addToBack, addSorted and SelectionSort. Each of these methods are be tested using Unit testing. Their execution duration will be analyzed through the ExperimentController class.

Unit Tests

The functions of each methods are:

- addToBack(): allows to add elements to the back of the container
- addToFront(): allows to insert an element at the beginning of the container
- addSorted(): allows new insert new element on a sorted list
- selectionSort(): allow to sorts the list using selection sort method.

Each one of these methods were tested one by one using unit testing. Unit testing is used to validate if the method's results executes the same way as it is expected to be.

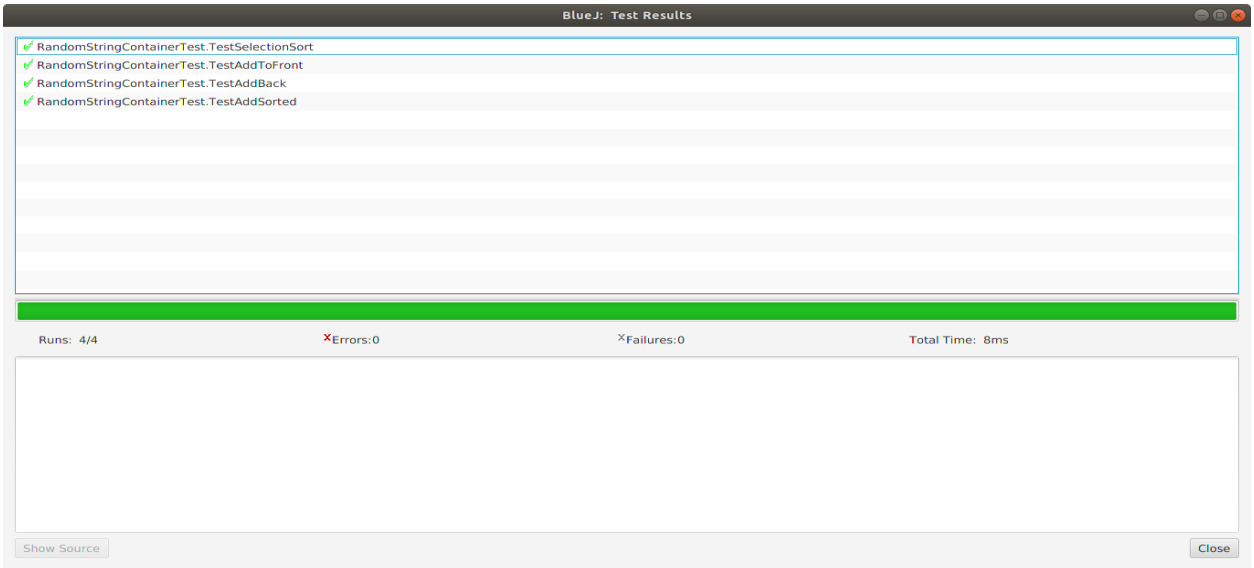

Figure 1: Screenshot Unit Testing Window

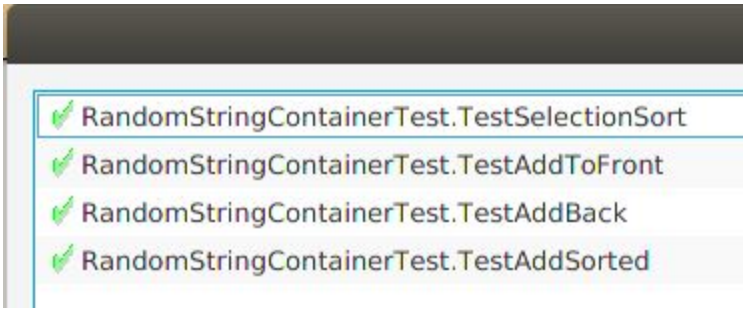

Figure 2: A zoom of the Unit Testing Window

Required Output

When measuring the execution time of addToBack(), addToSort(), addSorted() and selectionSort(), 4 combinations of seeds were used to generate the random numbers. The

different times obtained from these different combinations of seeds are averaged. These averages are the ones used to compare the execution time of the different methods.

Five initial strings are inserted inside the container before calling any method. The role of these initial strings is to make sure that each function is executing correctly as they are supposed to. For example, when using AddSorted, the function expects that the container is sorted first. However if there are no initial items in the container, it would just add an item similar to addToBack() or addToFront().

Comparison between AddToFront() ,addToBack() and addSorted

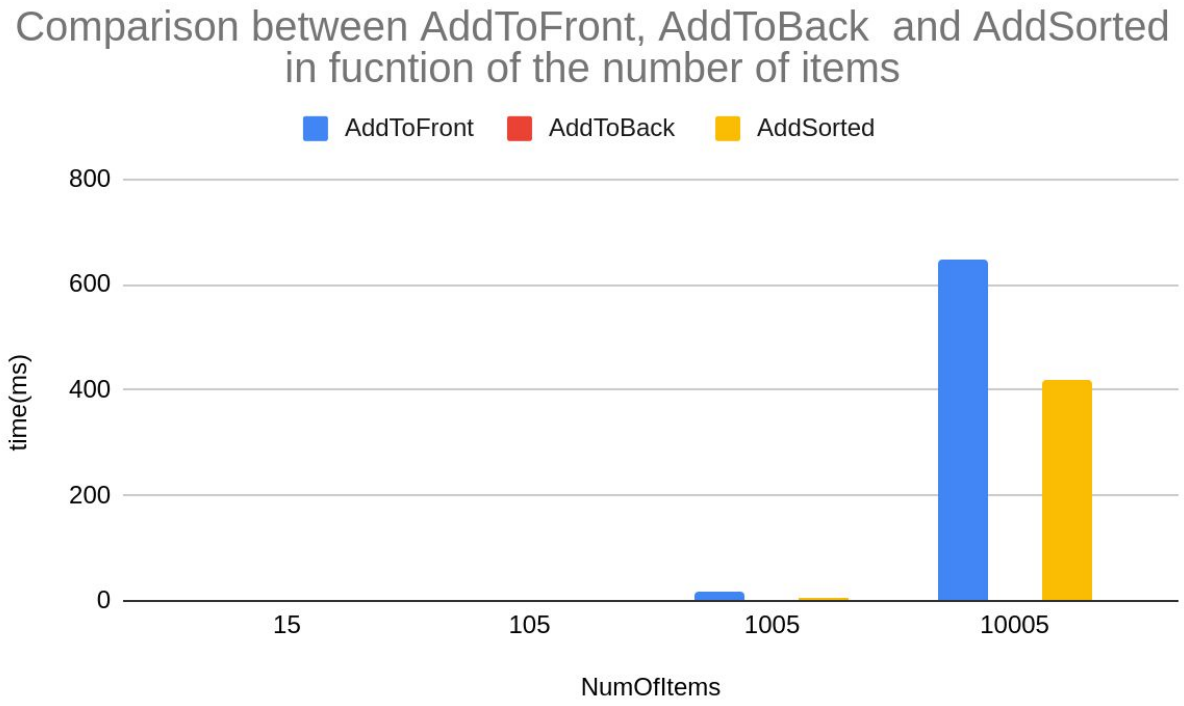

Figure 3: Comparison Execution time AddToFront ,AddToBack and addSorted method in function of the item number

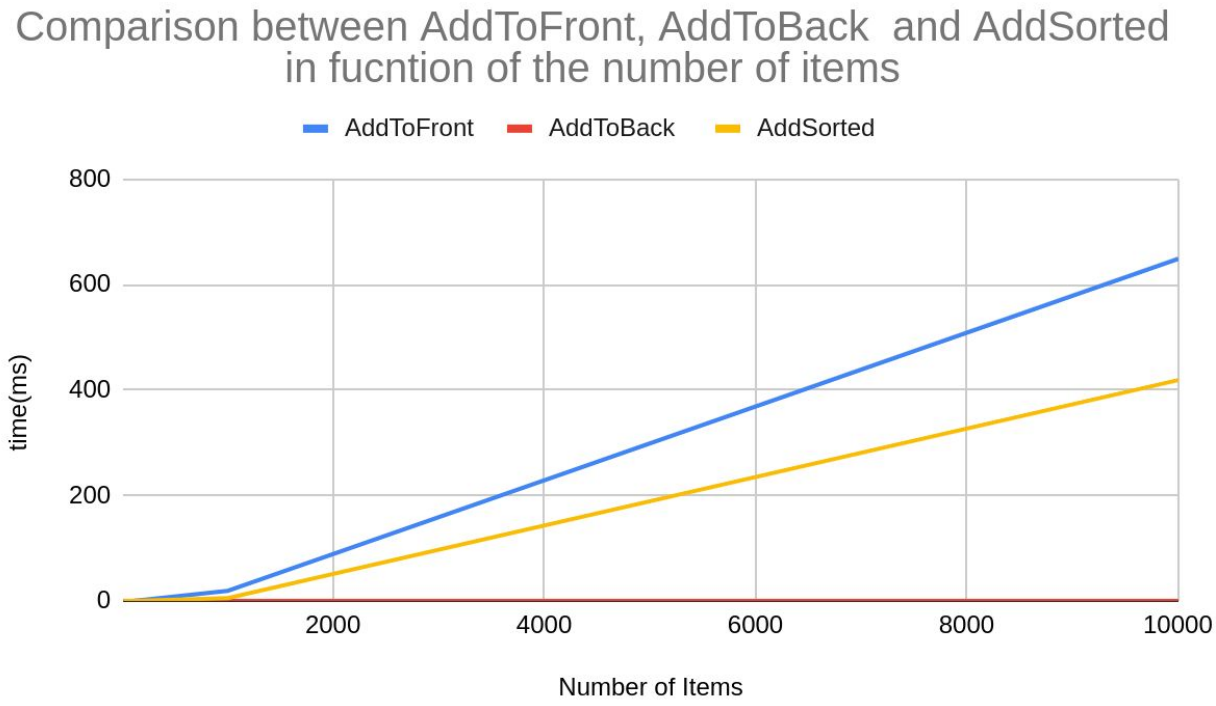

Figure 4: Comparison Execution time AddToFront ,AddToBack and addSorted method in function of the item number

When addToFront, addToBack and addSorted were measured, the result demonstrated that addFront took much more time than addSorted and addToBack when the number of items is high. This can be explained by the fact that addToFront method copies all the elements on another container before inserting the new values. The time it takes to do the copy and to re-append the elements takes several times as the number of items increases.

On the hand addToBack() takes almost the same time during the whole process.

Comparison between SortofUnsorted and SortOfSorted

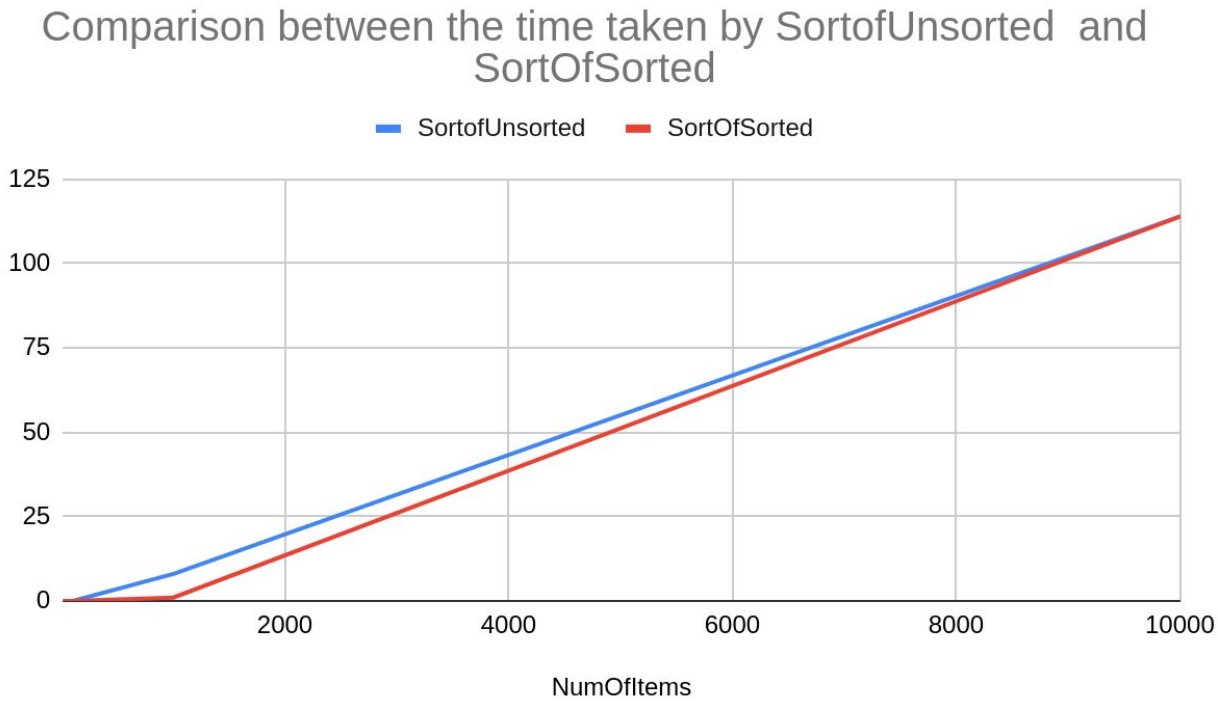

Figure 5: Execution time AddAtt method in function of the item number

There is almost no difference between the time taken by sorting an unsorted container and sorting a sorted container. It makes sense because the big O of a selection sort is O(n^2) for the worst, average and best case. The algorithm will always go from the initial index to the last index to check the smallest value and swap the value.

Trouble Report

N/A

References

Admin. (2020, July 28). Selection sort java ---- Flower Brackets ---- code here. Retrieved September 05, 2020, from https://www.flowerbrackets.com/selection-sort-java/

Oracle (2019). ArrayList Documentation. Oracle.Com. https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

Michael Kölling (2015). U0nit Testing in BlueJ https://www.bluej.org/tutorial/testing-tutorial.pdf