Linnaeus University

1DV507 - Report on Assignment 4 - Exercise 4

Student: Dennis STEINHILBER
Student ID: ds222rr@student.lnu.se

Contents

1 Introduction

This report contains the results and discussion of two experiments. The first experiments is about the efficiency of concatenations of Strings on one hand and on the other hand, the efficiency of the append-method of StringBuilders. The second experiment is about the efficiency of Insertion and merge sorting algorithms regarding random integers and Strings. In both experiments, it was measured for every method how many operations could be performed in one second. This experiments was carried out on a MicroSoft Surface Book, Intel Core i5 (6. Generation), 128 GB Flash, 8 GB RAM using Windows 10 and Eclipse Oxygen.

2 Excercise 2

The program 'Time' was written in Java Eclipse to determine the amount of concatenations and appendices that can be performed in one second. The main method contains essentially four method calls, one for every case. The first case is concatenations of strings with just a single character and the second case is concatenation of strings with 80 characters. The third case is appendices of single character strings to a StringBuilder and the fourth case is appendices of strings having 80 characters. Two methods were created, one to determine the amount of concatenations and the other one to determine the amount of appendices in one second. Every case will be executed several times (depends on the value of 'b', number of iterations of the for-loop). In every case, the start time is measured at the beginning of an iteration and the end time is measured in a do-while-iteration after one concatenation/appendix. If the time is not less than one second, the do-while iteration ends and the next for-iteration starts. Finally, the average result is printed out.
For the method to count the appendices, there is an additional condition: The time to execute the StringBuilder.toString-method is measured as well, also iterated by a for-loop several times and divided by the number of iterations to get the average value. This value is then divided by the average time to append one single character. The result itself is then divided by the length of the string which was appended, which can only have the value of 1 or 80. Finally, the result is first subtracted from the total number of appendices and then the result is multiplied with 80 and subtracted from the total string-length. This is done to subtract the time of the toString-method from the overall result of the appendices and length of the StringBuilder.
The results show that the shorter the concatenated/appended String is, more concatenations/appendices are possible. On the other hand, the longer the concatenated/appended String is, the longer will be the length of the final string.

Method	Period	Con/App	Length
String short	1	54073	54073
	2	57459	57459
	3	53484	53484
	4	55528	55528
	5	54354	54354
String long	1	4194	335552
	2	4465	357256
	3	4708	376656
	4	4583	366688
	5	4704	376328
StringBuilder short	1	13189765	13189765
	2	12593766	12593766
	3	14382567	14382567
	4	15623259	15623259
	5	16324790	16324790
StringBuilder long	1	2526071	202085728
	2	2904223	232337912
	3	2769395	221551664
	4	2913401	233072144
	5	2716064	217285144

Figure 1: Results of Exercise 2

3 Excercise 3

the program 'TimeSort' was written in Java Eclipse to determine the number of integers and strings that can be sorted by integer- and mergesort algorithms in one second. Also here, the main method contains essentially four method calls. The start time is measured before a method is called. The methods are the same as from the SortingAlgorithms-exercise in assignment 3. In all methods, the end time is measured after an element has been sorted. The difference between the insertionsort and the mergesort is that in the insertionsort methods, if one second has passed, the iteration is stopped by a break-command, the method ends and the result is printed. In the mergesort methods, if one second has passed, no more elements are added to the overall result. After the method ends, the result is printed as well.

Method/Period	1	2	3	4	5
Insertion Integer	38009	37977	38404	37758	39330
Insertion String	16579	16369	15719	15747	16680
Merge Integer	1879758	2109370	1992183	1808156	1787388
Merge String	937494	865574	919915	964011	962395

Figure 2: Results of Exercise 2

4 Conclusion

Concatenations of strings is slower than appendices in StringBuilder. The reason for this is that during a process of a concatenation, a new string object is declared, which needs more computing power. This is not the case when using the StringBuilder class. In contradiction to a string, a StringBuilder object is changeable. The purpose of the StringBuilder class is to build strings and finalze them in the end to avoid the problem of creating new string objects after a change.
The smaller an array to be sorted is, the more effective is the insertion sort. On the other hand, the larger an array to be sorted is, the more effective is the merge sort.