DEN 11 AUGUSTI 2017

Time Measurements

AYA KATHEM
1DV507

Content

1.1 Introduction

The aim of this report is to show results of different experiments which test the time measurements. In the first assignment will check repeating string concatenations in two different way. The first one is the plus operator and the second is by using StringBuilder. In the second assignment will check how many strings and integers can be sorted in 1 second using insertion sort.

2.1 String Concatenation

The main point of this assignment is to check which is faster plus operator or StringBuilder to repeat string concatenations in 1 second. In the first hand, is to check adding short strings concatenation from only one character. In the second hand, is to check adding long strings representing a row with 80 characters. The below table will show the number of concatenations, and the final string length in one second in different runs of the program

2.2 Procedure

Four methods have been created. All methods are a static void method, whilst no need to create an object and no need to return specific case. Every method tests a specific case, the first method is for testing one character using plus operator, the second methods for testing a long string of 80 characters using plus operator, the thread is for testing one char using StringBuilder and the last one is for testing long string of 80 characters by StringBuilder. Inside every method, I have used System.currentTimeMillis() to check how many string concatenations can be repeated in one second by using while loop. In the main method, there is a for loop run for five times to see approximate result. However, I got a problem in the fourth method when I build the long StringBuilder, that cause out of memory error. I solve this problem by allocating more memory to the JVM by updating Eclipse using Xms1024m

Short strings containing only one character

NUM	Time in millisecond	Number of Concatenations (using plus operator)	Length (using plus operator)	Number of Concatenations (Using StringBulider)	Length (Using StringBulider)
1	1000	55401	55401	81149	81149
2	1000	61161	61161	90592	90592
3	1000	60295	60295	90340	90340
4	1000	63100	63100	89862	89862
5	1000	63551	63551	91198	91198

Long strings representing a row with 80 characters

NUM	Time in millisecond	Number of Concatenations (using plus operator)	Length (using plus operator)
1	1000	4128	330240
2	1000	4894	391520
3	1000	5014	401120
4	1000	4957	396560
5	1000	4963	397040

Long strings representing a row with 80 characters

NUM	Time in millisecond	ToString Time in millisecond	Number of Concatenations (Using StringBulider)	Length (Using StringBulider)
1	1117	330	4299162	343932960
2	1040	165	2149581	171966480
3	1330	352	4299162	343932960
4	1043	155	2149581	171966480
5	1325	407	4299162	343932960

2.3 why StringBuilder is Faster

The final results in the tables show that StringBuilder is much faster than plus operator. The process of the plus operator is to make a copy each concatenation. This process consumes memory (new string instance) and requires time to copy the old string. In contrast, the StringBuilder use internally an array of chars. The process of the StringBuilder is to add the string at the end position and only make a copy in some cases. As a result, in this assignment when we use the StringBuilder the program only saves a copy when the data become too big.

3.1 Sorting algorithms

In this assignment will show how fast is when we use insertion Sort. The experiment proves how many integers or Stings can be stored in one second. Insertion method is a simple sorting algorithm, which returns an array in order. The insertion method is filled by a random generator of integer and string. In this assignment, I have also used System.currentTimeMillis() to check how much insertion method can store data in one second.

3.2 procedure

The program starts with a random generated integer. An array of length 10000 has been created, the array contains random numbers twice the length of it. Then initialized two long variables to calculate time. Start a while loop with the statement while the time is less than 998 stop the while loop. I choose 998 milliseconds because I will get result more nearly to one second. Inside the while loop, I start to fill in insertion method. Finally, when the time is about one second the while loop terminates and calculates the total time and array length. Then I test the string insertion sort in the same way with also random generated strings.

Table for integer insertion sort

num	Created array length	Time in Millisecond	Array length has been stored
1	10000	1001	980000
2	10000	1004	930000
3	10000	998	950000
4	10000	1000	960000
5	10000	1002	970000

Table for string insertion sort

num	Created array length	Time in Millisecond	Array length has been stored
1	2000	999	532000
2	2000	1000	544000
3	2000	1000	526000
4	2000	998	534000
5	2000	999	532000
6	2000	1000	540000