Exercise 2

I've created a function for each test im doing, so a total of 4 functions that prints out a String with two values. The amount of times it repeated the concatenation, and the length of the final String. I also have a static int[] with the size of 2 that fits the total amount of repeats,and String length for each test. I do each test 5 times which means to calculate the average result i just have to divide the total sum in my array by 5.

The Strings im using for the differents tests are simple. For the single character im just using the character 'a'. With the 80 character long String i just generated a 80 character long Lorem ipsum sentence from a website.

The data below represents the average data of five tests. Repeats represents the amount of concatenation the program was able to do, and the String length is as the name the final length of the String. Since i'm only adding one character each time on two of the tests, the amount of repeats and String length will be the same.

Variable Type	Character amount	Repeats	String Length
String	Single Character	50547	50547
String	80 Characters	4207	336576
StringBuilder	Single Character	36215624	36215624
StringBuilder	80 Characters	4299162	343932960

I'm happy with this data as it shows very clearly that the StringBuilder is a lot faster than the standard String variable when it comes to doing a lot of concatenation.

The reason for this is that when using the String += "some text" it's creating a whole new instance of the variable which allocates memory. Where StringBuilder is built of char arrays which only needs to check for free space and put in the .append("text").

Exercise 3

I’ve created a total of 5 methods for this exercise. 2 which are the sorting methods, and then two which are the time tests themself. The last method is the one generating a String with ten random characters. I do this by having an char array with the alphabet and then just using random numbers to pick 10 out of the letters to but in the String.

I run each of the time tests five times as the exercise above to get a more precise number. The tests themself are quite alike the ones from exercise , the only difference is that this time i'm only measuring the amount of sorts instead of the length of something at the end as well. Since i already calculated the amount of repeats from the last exercise i only needed to do the same here.

The results below represents the average result of five tests. The results clearly shows that sorting the Integers is around 8.5 times faster than sorting the Strings One thing that may impact this is my function to generate the random String since generating the random Integers is a lot simpler. Another impact may be the fact that comparing two Integers is simpler than comparing two Strings. When comparing two Integers you can just use the standard expression >,<,= etc.While comparing the Strings im using a comparator which probably allocates more memory than the Integer comparison.

Variable Typ	Repeats
Integer	4722182
String	552878