CS/COE 0445 Fall 2020 Assignment 2
Part Two

Note: Prior to attempting this, carefully read Assignment 2 Part One. This part of the assignment requires your MyStringBuilder class to be complete and working.

Online: Friday, September 18, 2020
Due: *See Assignment 2 Part One for due date and submission information*

Purpose: Once you have completed and tested the functionality of your MyStringBuilder class, you will do a rough comparative test to see how long some of the operations take relative to the predefined StringBuilder class and the predefined String class.

Details:
You will complete a simple simulation to compare the times required for three operations:
`append(char c)`, `insert(int loc, char c)` and `delete(int start, int stop)`
These methods are defined in the StringBuilder and MyStringBuilder classes, but you will have to approximate them with the String class, since it is immutable [part of the assignment is to figure out how to approximate these with a String – as a hint look at the substring() method in the String class).

You will test these operations in the following way:
- Write a program called `Assig2B.java`, which will be the *main program* for the rest of this description.
- Your main program will be invoked with a command line argument, which is the number of characters that you will be appending / inserting into your string objects. Call this value N. Since the value of the character does not matter here, you can just use all character 'A's for your operations.
  For example, a run with 10000 characters would be:
  `java Assig2B 10000`
- In your main program, you will iterate 3 times, once for each of the three test classes (StringBuilder, MyStringBuilder and String). For each iteration in your main program, you will do the following:
  - Create a new, empty object of the class that is being tested
  - Time the operations as described below. For more details on the procedure for timing, see below.
  - **Time** append(): Use the append() method (or String approximation thereof) to add each of the N characters, one at a time, to the end of the current testing object.
  - **Time** delete(0,1): Remove the first character of the current testing object repeatedly until the object contains no more characters (a total of N characters should be removed).
  - **Time** insert(): Process the N characters again. However, this time you will insert() each character into the middle of the testing object. The "middle" of the object can be determined by dividing the length by 2. **Be careful to test for special cases (make sure the "middle" index is valid).**
  - For each of your **Time** operations above you should proceed in the following way:
    - Mark the start "time" of your test with a call to System.nanoTime()
    - Perform ALL N of the operations on the current testing object (i.e. append / insert all N characters into or remove all N characters from the object)
    - Mark the stop "time" of your test with a call to System.nanoTime()
    - Calculate the elapsed time for all of the operations by subtracting the start time from the stop time
    - Calculate the average time per operation by dividing the elapsed time by the number of characters processed (either appended / inserted or deleted)
    - Output your results to the display, both for the total time and the time per operation

Run your program with each of the following input values for N: 10000, 20000, 40000, 80000, 160000

Write a very short report (i.e. a few paragraphs) with the following information:
- Which implementation was best for append() and was it "close" or was there a clear winner?

- Which implementation was best for delete(0,1) and was it "close" or was there a clear winner?
- Which implementation was best for insert() and was it "close" or was there a clear winner?
- Overall which implementation do you think would be best from a run-time point of view?
- To support your report, include a table for each operation showing each of the algorithms and **run-times per operation** for the different numbers of characters. For example, the table for your append() operation would look like the table below. You will need 3 tables total in your report.

Append Operation					
Implementation	10000	20000	40000	80000	160000
StringBuilder					
MyStringBuilder					
String					

Your paper should be in some standard format (.txt, .pdf, .doc, .docx, .html) and be submitted in its own file. **Call your paper `Writeup` (with which ever extension you use). Make sure you include this file in your submission .zip file.**

For more details on submission procedures and dates, see Assignment 2 Part One