CMPU 102 Assignment 1: Collatz sequence

Due on Thursday, February 9th, by midnight.

Problem Specification:

For this assignment, you will write a program to compute the Collatz Sequence. The algorithm for this program is shown below.

1. Prompt for and read any positive integer n from user.
2. While (n > 1)
3. If n is even, set n = n / 2.
4. If n is odd, set n = 3n + 1.
5. Display the Collatz sequence for the given number n.

In 1937, Lothar Collatz proposed that no matter what number you begin with, the sequence eventually reaches 1. This is widely believed to be true, but has never been formally proved.

Write a program that reads a positive whole number n (a natural number) from the user and then displays the Collatz Sequence starting at n. Stop when n reaches 1.

You are expected to write this code on your own. Do not copy from the internet or other sources.

Part I

To get started, create a new folder called **walter-assign1** (**with your last name first, not mine**). For example, if your name is Jill Tracy, call the folder tracy-assign1. Then download the starter file Collatz.java from the course web page into the directory you just created.

Code Specifications:

Open the Collatz.java program in DrJava. This program is incomplete, but it should run with no errors. Try running it to ensure it has no errors.

START CODING BY TYPING YOUR NAME AT THE TOP OF THE FILE.

The main method should prompt for and read a positive integer (≥ 1) from the user. Choose between System.out and JOptionPane built-in classes to prompt the user and either Scanner or JOptionPane to read the number.

A method called getCollatzSequence is already present in the starter file, but it does not function properly. Read on for instructions on how to code this method so that it implements the algorithm given at the top of this page.

The execution of your program should have output that looks like examples I-1 through I-4, in which the user input is shown in boldface type:

Example I-1: Running the program with input of 1.

```
Please enter a whole number greater than or equal to 1 and I will generate and
display all the numbers in the Collatz sequence:  1

The Collatz sequence for starting number 1 is:
1
```

Example I-2: Running the program with input of 6.

```
Please enter a whole number greater than or equal to 1 and I will generate and
display all the numbers in the Collatz sequence:  6

The Collatz sequence for starting number 6 is:
6 3 10 5 16 8 4 2 1
```

Example I-3: Running the program with input of 5.

```
Please enter a whole number greater than or equal to 1 and I will generate and
display all the numbers in the Collatz sequence:  5

The Collatz sequence for starting number 5 is:
5 16 8 4 2 1
```

Example I-4: Running the program with input of 3.

```
Please enter a whole number greater than or equal to 1 and I will generate and
display all the numbers in the Collatz sequence:  3

The Collatz sequence for starting number 3 is:
3 10 5 16 8 4 2 1
```

Hint: Declare a String variable (e.g., `str`) prior to a while loop. The string `str` should initially contain the empty String. Each time the loop iterates, concatenate each new number in the sequence with a space or two after each number.

Do not write all the code in the main method. Instead, call the method with signature:

```
public static String getCollatzSequence(int n)
```

from the main method. The getCollatzSequence method should implement the algorithm shown at the top of the page and return the sequence to be printed in a String.

In Java, to determine if a number is even or odd, use the Modulus operator %. If x % 2 == 0, the number is even and if x % 2 == 1, the number is odd.

You should make sure (via testing) that your program works for all integers less than 10 before going on to Part 2.

Part II

For this part, you should start with the code you already have written in the Collatz.java file, as specified above. *Save the Collatz.java file and then give it another name– CollatzExtra.java and be sure to change the class name inside the file to match the file name.* Modify the code from Collatz.java so that it tallies how many numbers are generated and reports the largest number in the the output sequence. These changes require 2 more variables to use in the while loop, one to keep track of the number of iterations and the other to hold the maximum value in each sequence. Your output for this part should look like that shown in the examples below (user input is shown in bold face type):

Example II-1: Running the program with input of 6.

```
Please enter a whole number greater than or equal to 1 and I will generate and
display all the numbers in the Collatz sequence:  6

The Collatz sequence for starting number 6 is:
6 3 10 5 16 8 4 2 1
Terminated after 9 steps.
The highest number in the sequence is 16.
```

Example II-2: Running the program with input of 11.

```
Please enter a whole number greater than or equal to 1 and I will generate and
display all the numbers in the Collatz sequence:  11

The Collatz sequence for starting number 11 is:
11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
Terminated after 15 steps.
The highest number in the sequence is 52.
```

Example II-3: Running the program with input of 27.

```
Please enter a whole number greater than or equal to 1 and I will generate and
display all the numbers in the Collatz sequence:  27

The Collatz sequence for starting number 27 is:
  27   82     41   124    62    31    94    47   142    71
 214  107    322   161   484   242   121   364   182    91
 274  137    412   206   103   310   155   466   233   700
 350  175    526   263   790   395  1186   593  1780   890
 445 1336    668   334   167   502   251   754   377  1132
 566  283    850   425  1276   638   319   958   479  1438
 719 2158   1079  3238  1619  4858  2429  7288  3644  1822
 911 2734   1367  4102  2051  6154  3077  9232  4616  2308
1154  577   1732   866   433  1300   650   325   976   488
 244  122     61   184    92    46    23    70    35   106
  53  160     80    40    20    10     5    16     8     4
   2    1

Terminated after 112 steps.
The highest number in the sequence is 9232.
```

Don't worry about getting the sequence numbers to line up perfectly but they should be separated by at least one space.

IMPORTANT: Be sure to remove all code that is not used in the files and all comments that do not apply and make sure your code is indented properly with no lines that are longer than 80 columns before you submit it.

When you are finished with this project, submit the zipped **yourname-assign1** folder on the CMPU102 Moodle page. If compressing the file is problematic, submit the 2 java files separately. Any file with a after the name is a previous copy of the algorithm. Do not submit any files with a after the name.