07. Seventh Assignment – Arabic to Roman Numerals

In this assignment, your task is to write a program the converts Arabic numerals (the kind we usually use) to Roman numerals. Your program should be able to handle integers between 1 (the Romans had no way to represent zero, negative, or decimal numbers) and 3999. You should display an error message for numbers outside of this range. Allow the user to continue converting values until the enter -1 to quit. The code for doing the conversion should be contained within a method with the following signature:

```
public static String arabicToRoman(int arabic);
```

Part of the task of writing good code that is often overlooked in school is testing your code in a logical way. When you test code, it is important to have tests that cover each of the following situations:

- *normal case* – a standard instance of the problem that tests the basic operation on normal inputs
- *invalid case* – tests that the program responds appropriately to various types of invalid inputs
- *boundary case* – a tricky instance of the problem that tests the operation on inputs that have to be handled in a special way

At the bottom of your code in this assignment, include a comment that explains your test plan – the inputs you chose for each case, why you chose them, and the results produced by your program. Do NOT write a description of your development or debugging process. Instead, provide a list of normal, invalid, and boundary test cases and the expected results.

Roman Numerals

| | |
|---|---|
| 1 | I |
| 5 | V |
| 10 | X |
| 50 | L |
| 100 | C |
| 500 | D |
| 1000 | M |

The Romans thought it would be easier to read if they did not have the same numeral four times in a row, so instead of IIII, the number 4 is written IV. Similarly, 9 is written IX instead of VIIII. 99 is XCIX. Please see Wikipedia if you need more information about Roman numerals. Hint: If you can get your program to produce the Roman numerals, but you are having trouble with the rule about not having four of the same character in a row, you might want to consider using the replaceAll method of the String class to

search for the invalid patterns and replace them with the correct pattern.

Sample output:

```
Enter an integer (-1 to quit): 14
XIV

Enter an integer (-1 to quit): -2
The Romans did not have a way to represent negative numbers or zero.

Enter an integer (-1 to quit): -1
Goodbye!
```

You will be graded according to the following rubric (each item is worth one point):

- The user can continue converting values until they choose -1 to quit
- The program recognizes invalid input (negative numbers, zero, or numbers above 3999), displays an appropriate error message to the user, and allows them to try again
- Arabic numerals are converted into Roman numeral strings
- The Roman numerals produced are correct (including not containing four of the same character in a row)
- The code to do the conversion is contained within a method with the requested method signature
- Your test plan is logical
- Your program produces correct output for all inputs
- Your program compiles
- Your program runs
- You follow standard coding conventions (e.g. variable names, indentation, comments, etc.)
- ***Note:*** *If your program does not compile, you will receive a score of 0 on the entire assignment*
- ***Note:*** *If you program compiles but does not run, you will receive a score of 0 on the entire assignment*
- ***Note:*** *If your Eclipse project is not exported and uploaded to the eLearn drop box correctly, you will receive a score of 0 on the entire assignment*
- ***Note:*** *If you do not submit code that solves the problem for this particular assignment, you will not receive any points for the program's compiling, the program's running, or following standard coding conventions.*