Lab 4: Java Fundamentals, Part IV

20 pts

Distribute on October 2, 2023
Due before October 8, 2023 (Sunday) at 12:00 midnight

Learning Outcomes ((CLO) vs (SO) Mapping)

- Utilize Java syntax in fundamental programming algorithms (3)vs(1)
- Recognize and apply the various input and output devices in programming (4)vs(2)
- Recognize and apply the various control structures (5)vs(1)
- Recognize and apply the basic debugging strategies in programming (8)vs(1)

Requirements

With this lab, you gain experience with fundamental Java constructs including variables, assignment statements, the String, JOptionPane classes, and if-else selection structures.

Preliminaries

1. Create a Java project. The name of the project must be **lab04_<your FirstNameLastName>**. For example, my project would be named, lab04_PeterNg.
2. Add a class named **`RomanNumerals`**. Declare the class as your **main class** when the project is set up with Eclipse
3. Make sure you have the usual comment block at the beginning of your class:

```
/*
 * <your name>
 * CS 16000-01 – 02/03, Fall Semester 2023
 *  (Note: write either 02 or 03, depending on which is your section.)
 * Lab 4
 *
 */
```

Exercises (20 pts) **(Your work must be verified by the TA in the lab to get credit for this lab)**

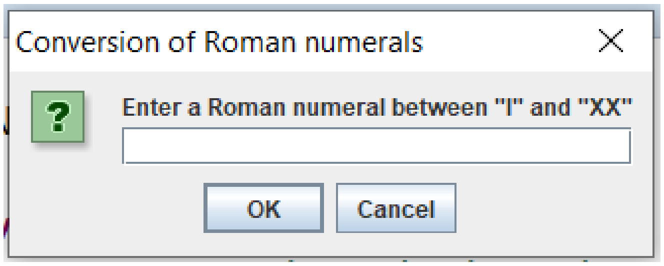
In this assignment, you solve a conversion problem similar to **Programming Challenges: problem 1 Roman Numerals** of Chapter 3 of your text, (page 181 in Edition 7). Given one of the Roman numerals **I, II, III, IV, V, VI, VII, VIII, IX, X, XI, XII, XIII, XIV, XV, XVI, XVII, XVIII, XIX, XX** as **input** your program must determine and display the corresponding decimal digit **1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16. 17, 18, 19, 20.**

Requirements

I. The solution must use a nested **if-else** construct built as follows:

1. The top-level if-else decision is made upon the first character of the Roman numeral, which is always `'I'` or `'V' or 'X'`. That is, divide the given group of Roman numerals into three subgroups according to their first character either `'I'` or `'V' or 'X'`.
2. Both the **if** and **else** blocks contain deeply nested if-else structures where the input string is compared to the Roman numerals I, II, III, IV, and IX in the first block and to V, VI, VII, VIII in the second block, and then to X, XI, XII, XIII, XIV, XV, XVI, XVII, XVIII, XIX, XX in the third block.

II. Input is solicited on a dialog window as shown in Figure 1.



Conversion of Roman numerals

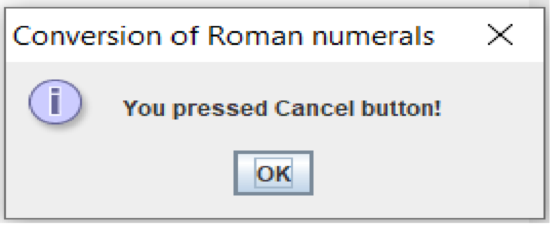
Enter a Roman numeral between "I" and "XX"

Figure 1

III. Declare a String variable **title** to store the title text such as “Conversion of Roman numerals”. Declare a String variable **task** to store the input solicitation text, such as “Enter a Roman numeral between “I” and “XX” ”. In the showInputDialog() method call that creates the window you must use the variables. For example, JOptionPane.showInputDialog(null, task, title, JOptionPane.QUESTION_MESSAGE). The input window must follow the template of Figure 1.

IV. Declare a String variable **roman** to save the input string returned by the window.

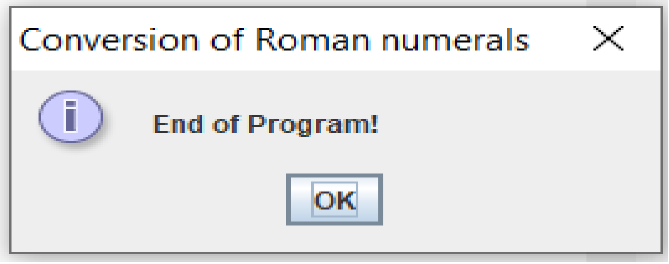
V. Use if (roman == null) for the following case: If press the Cancel button in the message box as shown in Figure 1, your program should display the following message “You pressed the Cancel button!”, as shown in Figure 2. Then after pressing the OK button of the message box in Figure 2, the message in Figure 3 is displayed through the message box as shown in Figure 3 before the program ends. before the program ends. (This is a way for solving error messages!)



Conversion of Roman numerals

You pressed Cancel button!

Figure 2

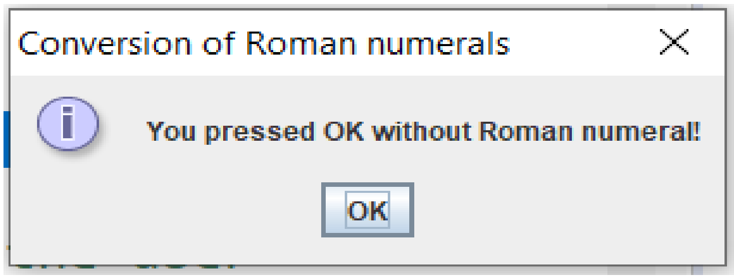


Conversion of Roman numerals

End of Program!

Figure 3

VI. If you pressed OK without entering a Roman numeral in the message box as shown in Figure 1, your program should display the following message “You pressed OK without a Roman numeral!” through a message box as shown in Figure 4. Then after pressing the OK button of the message box in Figure 4, the message in Figure 3 is displayed through the message box as shown in Figure 3 before the program ends. (This also is a way for solving the error message!)



Conversion of Roman numerals

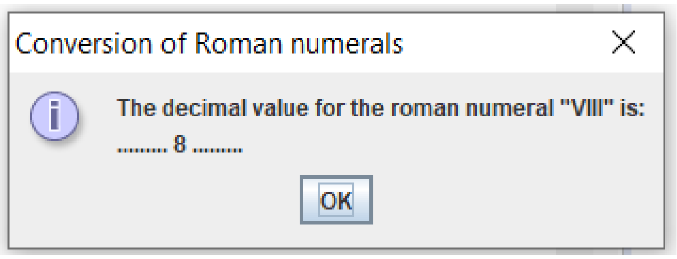
You pressed OK without Roman numeral!

Figure 4

VII. Declare an int variable **decimal** to store the decimal digit equivalent of the roman numeral and initialize to 0.

VIII. Allow using lowercase or uppercase, even mixed characters as input Roman numerals. The output Roman numerals must be in upper case. Use toUpperCase() for the conversions (Read Chapter 2: 2.9 The String Class, see toUpperCase, page 75, 7th edition).

IX. The output must be displayed on a message dialog as shown in Figure 5 and the output window must follow the template, take note of the title line and the output arrangement. Use the String.format() method for the output arrangement, such as “The decimal value for the Roman numeral “VIII” is: ……. 8 ……” in two lines. (Read Chapter 03: Displaying Formatted Output with System.out.printf and String.format, See the String.format on pages 172-173).



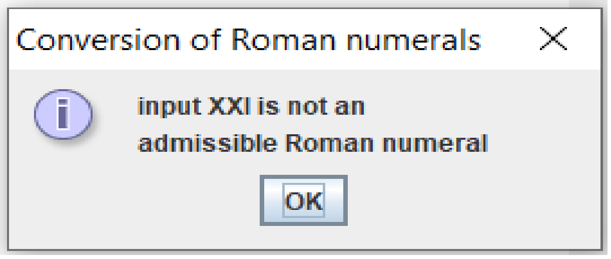
Conversion of Roman numerals

The decimal value for the roman numeral "VIII" is:
.... 8 ....

Figure 5

X. We assume that the input string is not empty and not **null** (each triggers a runtime error)

XI. If the input string is none of the ***twenty*** admissible Roman numerals, the message as shown in Figure 6 is displayed


Conversion of Roman numerals

input XXI is not an
admissible Roman numeral

Figure 6

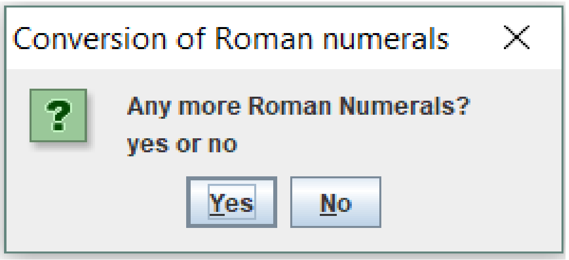
XII. When you check the first character, use the **==** operator for equality, and do not forget the character operator on the literal, put **`roman.charAt(0) == ‘I’`** into the topmost if header. (hints)

XIII. Use the equals method for string comparison (Read Chapter 3: 3.6 Comparing String Objects, pp 142-147, 7th edition). A nested if header shall look like. (hints)

```
if(roman.equals(“VII”)){
```

XIV. The output text (see Figure 5) contains the “ ” symbols. Use the escape sequence **\”** within the text. (hints)

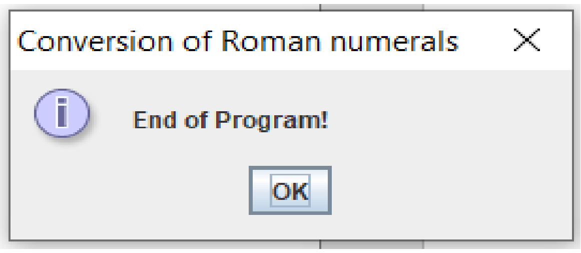
XV. Use do-while loop and the showConfirmDialog() method, such as “int yesNo = JOptionPane.showConfirmDialog(null, "Any more Roman Numerals? \nyes or no", title, JOptionPane.YES_NO_OPTION);” asking whether the process be continued as shown in Figure 7.


Conversion of Roman numerals

Any more Roman Numerals?
yes or no

Figure 7

XVI. Finally, before the program ends, a message is displayed through the message box as shown in Figure 3.



End of Program!

Figure 3

XVII. For JOptionPane class, you need to use the following two statements in a proper location. Do not excessively state System.exit(0); before the end of the main method. The import statement must be placed well top before the “public class RomanNumerals{…}”.

```
import javax.swing.JOptionPane;     //needed for JOptionPane class.
```

and

```
System.exit(0);     //required JOptionPane class.
```

What to Submit

- upload your zipped project folder at the class website on Bridgespace