Lab 5: String Comparison V

20 pts

Distribute on October 13, 2025
Due October 19, 2025 (Sunday) at 11:59 pm before 12:00 midnight

Learning Outcomes (CLO vs SO Mapping)

- Recognize the software and hardware components of a computer system (1)vs(6)
- 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(2)

Requirements
With this lab, you gain further experience with fundamental Java constructs, including variables and assignment statements, as well as comparing String objects, the Scanner class, and the JOptionPane class.

Preliminaries
0. Create an Eclipse Java project. The project's name must be **'lab05_Order3Strings_FirstNameLastName'**, where your Last Name follows the first character of your First Name. For example, my project would be named lab05_Order3Strings_PNg.

1. Create a class named **`Order3Strings`** added to the project

2. A comment block is required at the beginning of each Java class containing the following lines:

```
/*
 * <your name>
 * CS 16000-01 – 02/03, Fall Semester 2025
 *  (Note: Write either 02 or 03, depending on which section your section is.)
 * Lab 5
 *
 */
```

Exercise

In this exercise,

- solve Programming Challenges problem 7, Sorted Names of Chapter 3, p 238 of your textbook
- learn and practice the confirm dialog of the JOptionPane class
- learn and practice, and apply a Scanner object for the sake of splitting texts

Additional requirements are detailed as follows.

1. (2pt) Open a JOptionPane confirm dialog as shown in Figure 1 below. Follow the given template strictly. The syntax is similar to the other dialogs:

   ```
   JOptionPane.showConfirmDialog(null, <ask for a choice here>,
   <title line here>, JOptionPane.YES_NO_OPTION);
   ```

   Note that by clicking one of the buttons, this method returns an integer number, which is one of the two named constants,

   `JOptionPane.YES_OPTION` or **`JOptionPane.NO_OPTION`**

   Assign the return value to an integer variable. For example, the variable answer has a value returned by the showConfirmDialog.

   ```
   int answer = JOptionPane.showConfirmDialog( … )
   ```

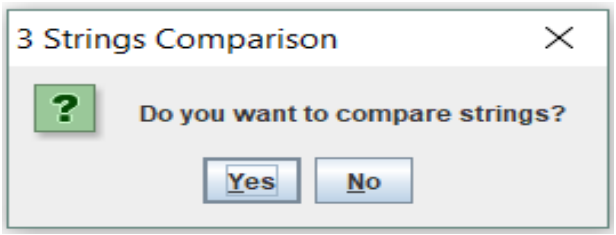

Figure 1
The title is "3 Strings Comparison"
The task is "Do you want to compare strings?"

2. (1pt) Using an if statement, check if the answer is the No option; in that case, send the message

   ```
   The program terminates!
   End of this program.
   ```

   to the message dialog box with JOptionPane.INFORMATION_MESSAGE and to console and terminate the program with the System.exit(0) statement. (See Figure 1a.)

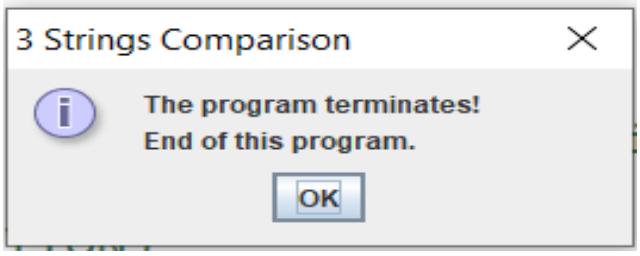

Figure 1a.

Otherwise, do as follows:

3. (1pt) Solicit three names written in a row to a JOptionPane input window, as shown in

Figure 2. Each name must be one word separated by spaces. Follow the template layout accurately, but feel free to choose alternative names. Save the returned value in a variable named **names**. Recall, the input window of the showInputDialog box returns a string.

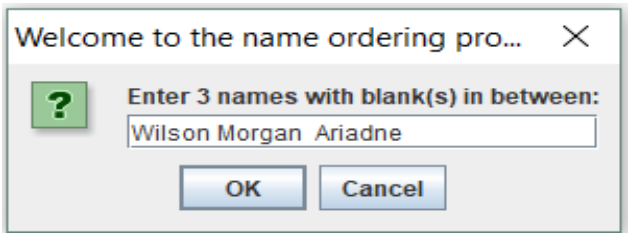

Figure 2

If you press Cancel in Figure 2, disregard whether any input or not (empty), send the message

"You pressed Cancel button!"

to the message dialog box with JOptionPane.INFORMATION_MESSAGE and to the console. Then prompt OK to terminate the program with the System.exit(0) statement. This sends the two-line message

"The program terminates!
End of this program."

to the message dialog box with JOptionPane.WARNING_MESSAGE and to the console also. (see Figures 2a and 2b).

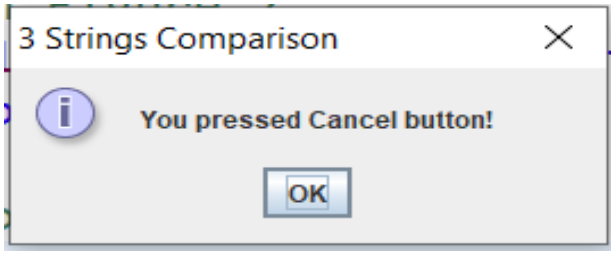

Figure 2a

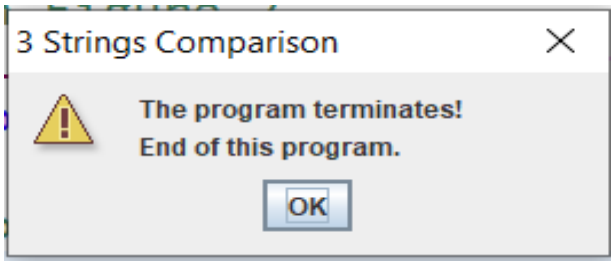

Figure 2b

4. (4pts) Validate the input: build a Boolean expression for the ***two*** cases with **names** that are empty or null, using an if statement with the Boolean to send the message

   "This program terminates for invalid input."

to the message dialog box with JOptionPane.WARNING_MESSAGE and to the console and terminate the program. (Note: The equals() method must be used to check for the empty string, and use the **==** operator to check the **null** value.) Let the value of the variable **names** be null when null is entered, or empty when no string is entered. (See 2c through 2e).

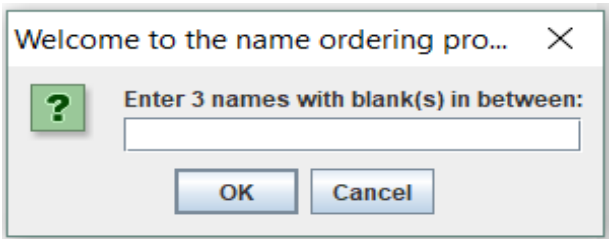

Figure 2c: empty and click on the button OK

Case 1: Build a boolean expression for the empty case **names**. (Note: You must use the equals() method to check for the empty string.) Let the value of the variable **names** be empty when no string is entered. If you press the OK button without entering any name(s) (i.e., an empty line), send the following message

"You pressed OK without three names!"

to the message dialog box with JOptionPane.INFORMATION_MESSAGE (shown in Figure 2d) and to the console. Then prompt OK to terminate the program with the System.exit(0) statement. This sends the two-line message

"The program terminates!
End of this program."

to the message dialog box with JOptionPane.WARNING_MESSAGE and to the console also. (shown in Figure 2b).

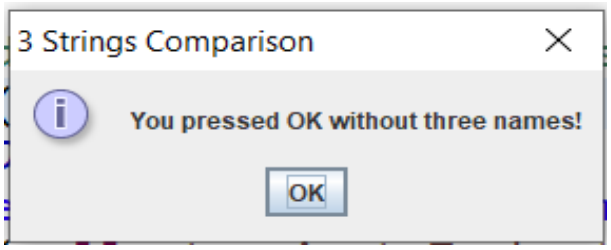

Figure 2d

Case 2: Build a Boolean expression for the null case **names**. If you press the OK button after entering null, use an if statement with the Boolean to send the following message

"This program terminates for invalid input."

to the message dialog box with JOptionPane.WARNING_MESSAGE and to the console and terminate the program. (Note: You must use the **==** operator to check the **null** value.) Let the value of the variable **names** be null when null is entered. (See 2e through 2f).

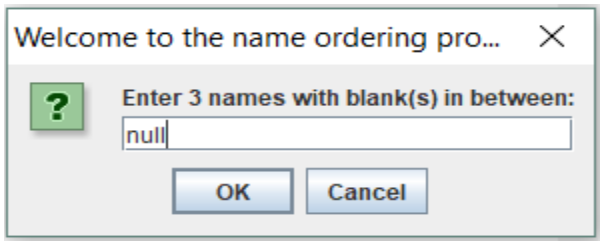

Figure 2e: Enter null and press the OK button
This will yield Figure 2f.

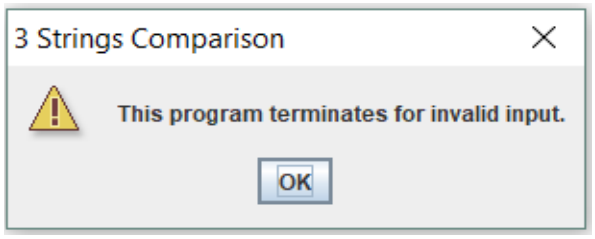

Figure 2f

Case 3: Check whether the number of name strings entered for the **names** is at least three (3) name strings. For example, the number of name strings in "Wilson Morgan Ariadne" is 3. This requires calling the method **`splitter.next( )`** three times. Furthermore, the names do not have to be distinct from each other.

If the number of input names is less than three (3), it will display a WARNING_MESSAGE and terminate the program. For example, as shown in Figure 2g, click OK after entering fewer than three name strings. This yields Figure 2f.

`"This program terminates for invalid input."`

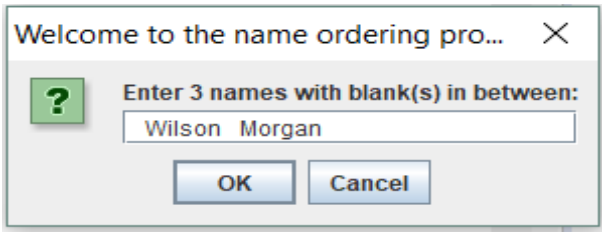

Figure 2g

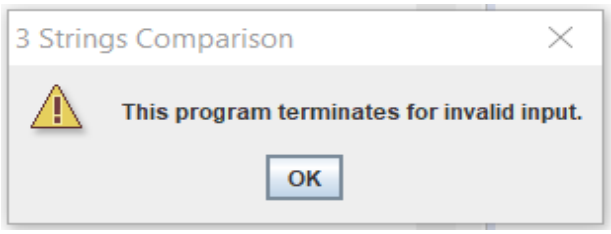

Figure 2f

Otherwise, do as follows:

5. Input containing more than three words shall be accepted, but only the first three items will be processed.

6. (1pt) Declare three String variables **`name1, name2,`** and **`name3`** to store the individual names from the input. Note that all variables used in the main method are to be declared right after the

```
public static void main(String[], args) {
       //variables declaration
       int yes/No;
          …
}
```

Do not declare any variables in the location between class and main, as shown below:

```
public class Order3Strings{
     //no variable should be declared here.
     public static void main(String[], args) {
       …
     }
}
```

7. (1pt) Declare and instantiate a Scanner variable **splitter** as before, but in the constructor, write the variable **names** as a parameter; that is, use the following code:

```
Scanner splitter = new Scanner(names);
```

8. (1pts) Apply the method call to the **`splitter.next( )`** three times, and save the return values one after each other in the variables **`name1,name2,`** and **`name3,`** respectively. Then use the following statement to call the sortWords() method, for example,

namesOrdered = *sortWords*(name1, name2, name3);

9. (6 pts) Write a sortWords() method, which sorts the three names, **`name1,name2,`** and **`name3,`** in lexicographic order. Create an **`if-else`** logic to determine the lexicographic order of the names. For this purpose, use the compareTo() method of the String class with the 'ignore case' option; declare and use additional helper variables if needed. There are six possible orders for three names; your selection logic must cover all of them.

10. (1pts) Collect and save the names in the correct order in a single String variable **`namesOrdered.`** The sortWords() method also returns to the main() the orderedNames

    Example:
    public static String sortWords(String w1, String w2, String w3) {

    //(9) sort the three names, **`name1, name2,`** and **`name3,`** in lexicographic
    //order.

    //(10) Concatenate the three name strings in lexicographic order. Then return the
    //string to the main() method.

    return orderedNames;

    } //end of sortWords()

11. (1pt) Display the variable **`namesOrdered`** on a message dialog, as shown in Figure 3, and the console.

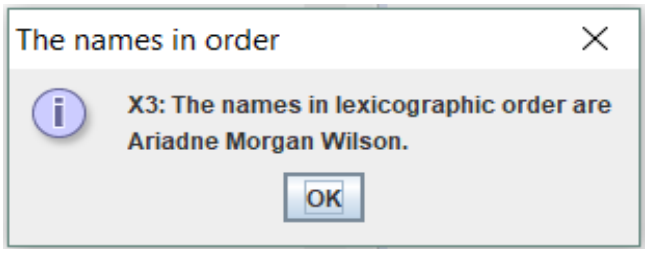

Figure 3

    Write a displayInConsole() method that displays the showMessageDialog and prints the message to the console.

12. (1pts) Test your program for each of the six (6) possible arrangements of three input words. You may use short words for testing purposes, like "A", "B", and "C". Note that the six (6) possible arrangements of three input words can be achieved only if the three input words must be distinct.

13. Use a while-loop statement for continuing string comparison. For example,

```
//initialization of the variable of integer data type, yesNo.
yesNo = JOptionPane.showConfirmDialog(null, task,
                title, JOptionPane.YES_NO_OPTION);

while (yesNo == JOptionPane.YES_OPTION)
 {
      . . .

    //Continue string comparison (See Figure 4)
     task = "Do you want to continue string comparison?";
     yesNo = JOptionPane.showConfirmDialog(null, task, title,
                                JOptionPane.YES_NO_OPTION);
 } //end while-loop
```

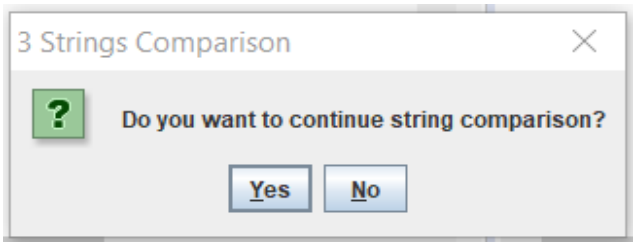

Figure 4.

In Figure 4, clicking "Yes" will proceed to Figure 2c, allowing for another round of string comparison by entering the input dialog box for other names, as shown in Figure 5.

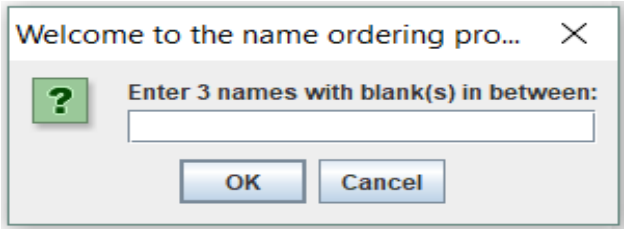

Figure 2c

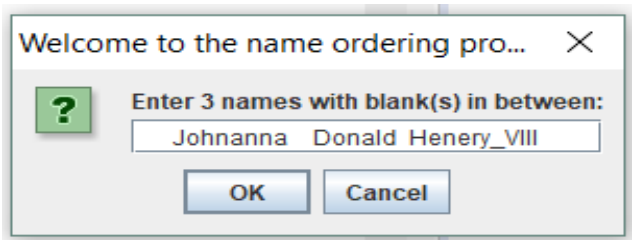

Figure 5.

From Figure 4, clicking 'No' will direct you to Figure 1a to end this program. Therefore, for each run of this program, many sets of names can be sorted in lexicographic order without restarting the program.

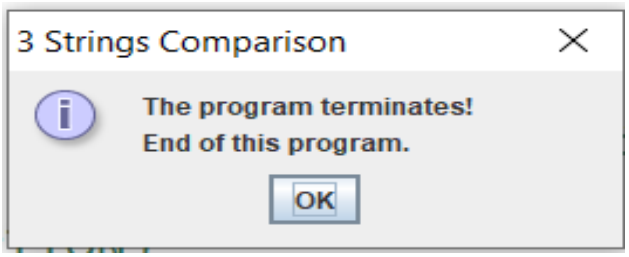

Figure 1a.

Ask your GTA to verify your presence and confirm that you are working in the lab.

What to Submit

- Submit your zipped project folder containing the source code and others on Purdue.Brightspace.com.