Lab 5: String Comparison V

20 pts

Distribute on October 9, 2023
Due before October 15, 2023 (Sunday) at 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, assignment statements, comparing String objects, Scanner class, and JOptionPane class.

Preliminaries
0. Create an Eclipse Java project. The name of the project must be **lab05_<your FirstNameLastName>**. For example, my project would be named, lab05_PeterNg.

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

2. Make sure you have a comment block at the beginning of each Java class containing the following lines:

```
/*
 * <your name>
 * CS 16000-01 – 02/03, Fall Semester 2023
 * (Note: write either 02 or 03, depending on which is your section.)
 * Lab 5
 *
 */
```

Exercise

In this exercise you
solve Programming Challenges problem 7 Sorted Names of Chapter 3, p 182 (7th Edition) of your textbook
learn and practice the confirm dialog of the JOptionPane class
learn and practice and additional use of 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 template exactly. 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 like

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

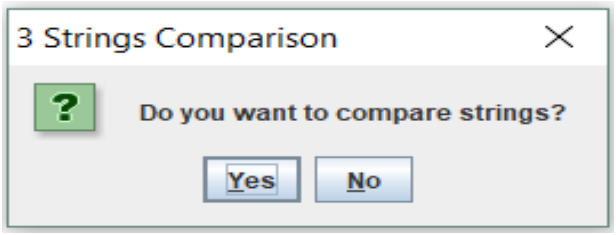

Figure 1

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.)

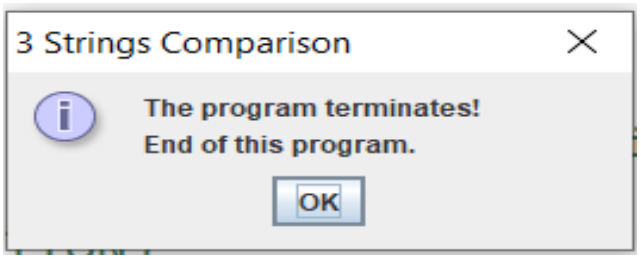

Figure 1a.

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 you can choose other names. Save the return value in a variable named **names**. Remember, the input window returns a string.

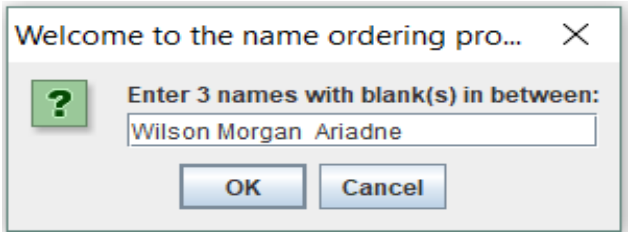

Figure 2

If 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).

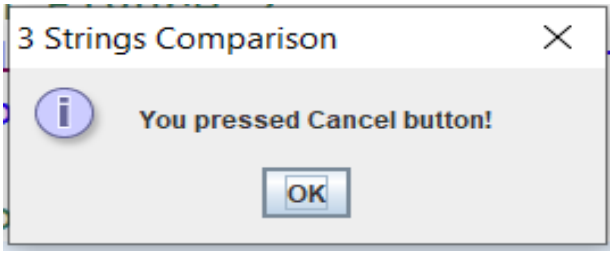

Figure 2a

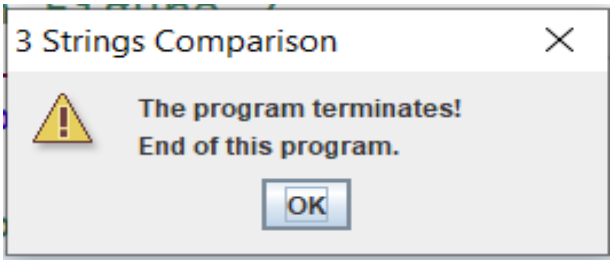

Figure 2b

4. (4pts) Validate the input: build a Boolean expression for the two cases **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: Must use the equals() method to check for the empty string and the **==** operator to check the **null** value. Assume that the value of the variable

names is null or empty when null is entered or without entering any string.) (See 2c through 2e).

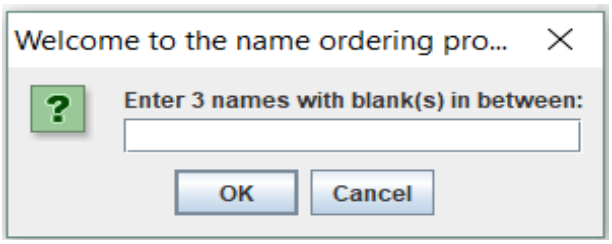

Figure 2c: empty and click on the button OK

Build a boolean expression for the empty case **names**: (Note: Must use the equals() method to check for the empty string. Assume that the value of the variable **names** is empty when without entering any string). If press the OK button without entering any name(s) (i.e, empty line), send the 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).

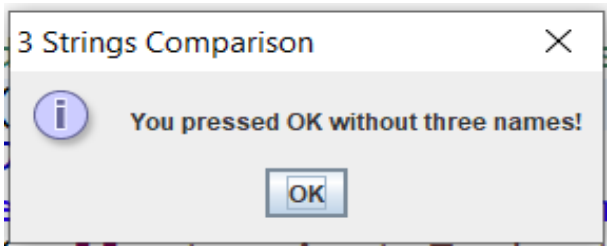

Figure 2d

Build a Boolean expression for the null case **names**. If press the OK button after entering null, use 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: Must use the **==** operator to check the **null** value. Assume that the value of the variable **names** is null when null.) (See 2e through 2f).

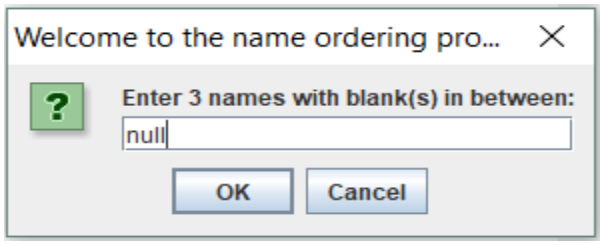

Figure 2e: enter null and press the OK button
this will yield Figure 2f.

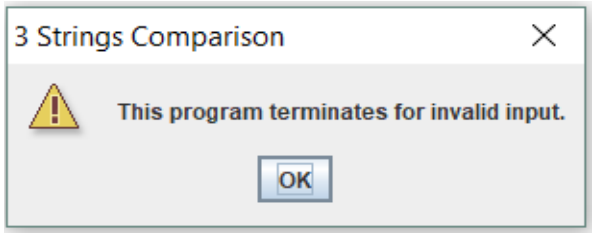

Figure 2f

For the third case, check whether the number of name strings entered for the **names** is at least 3 name strings, such as the number of name strings of "Wilson Morgan Ariadne" is 3. This is required if the method **`splitter.next( )`** calls three times. Furthermore, the names must be distinct.

For example, as shown in Figure 2g. after entering two name strings, and then click on OK. This will yield Figure 2f. If the number of input names is less than THREE, it will give a WARNING_MESSAGE and end the program.

`"This program terminates for invalid input."`

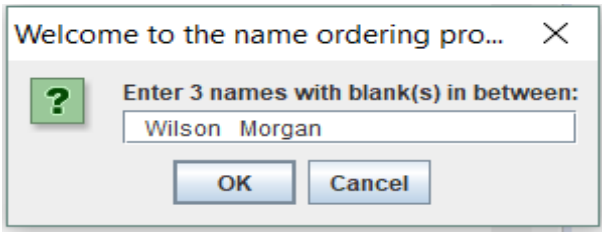

Figure 2g

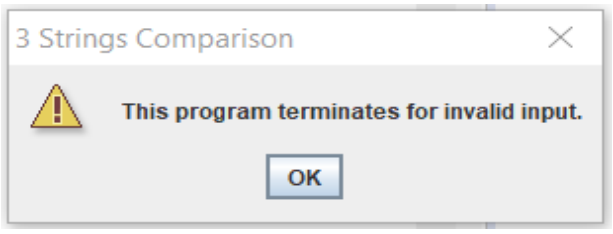

Figure 2f

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 serve as storage for 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 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

9. (6 pts) 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`**

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

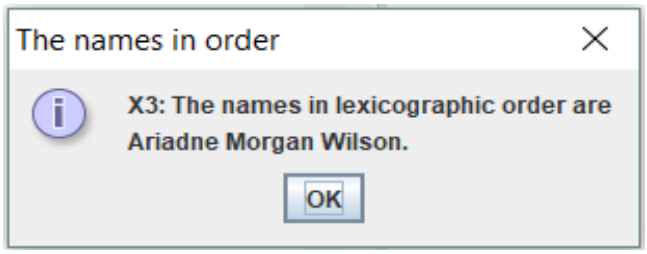

Figure 3

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

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

```
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
```

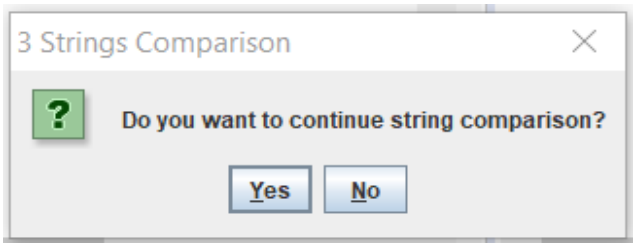

Figure 4.

In Figure 4, if click on Yes, it will go to Figure 2c, allowing for another round of string comparison, by entering the input dialog box for other **names** as shown in Figure 5.

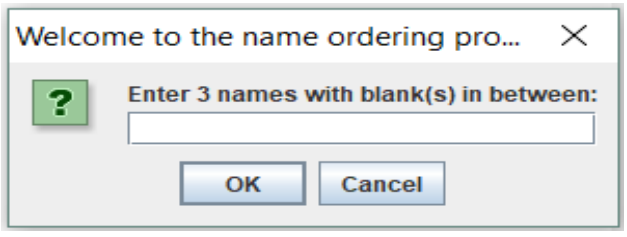

Figure 2c

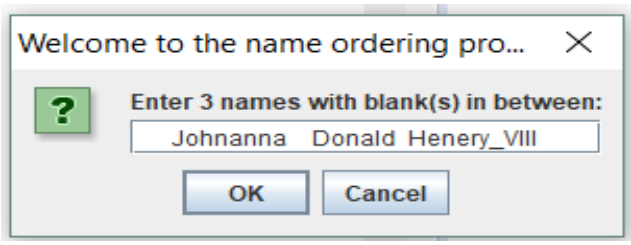

Figure 5.

From Figure 4, if click on No, it will go to Figure 1a, for ending this program. Therefore, for each run of this program, many sets of names can be sorted in lexicographic order without restarting the program.

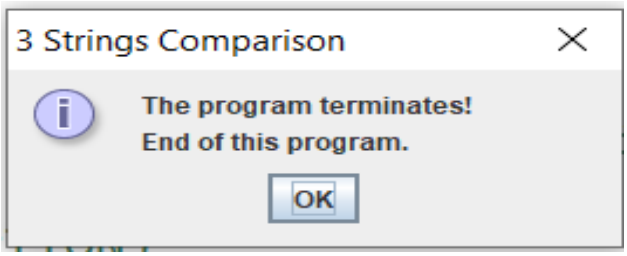

Figure 1a.

Ask your TA to verify your presence and work in the lab.

What to Submit

- Submit on Bridgespace your zipped project folder containing the source code and others.