Lab 2: Java Fundamentals **II.**
20 pts

Distribute on September 15, 2025
Due before September 21, 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)

Requirements
The focus of this lab is to
- gain experience with variables, assignment statements, and arithmetic operators;
- learn the fundamental Java output device: sending messages to the console;
- practice error-correcting routines.

Furthermore, this lab is also to gain experience
- creating a String object of the String class,
  - variables (called reference variables) holding the address of a String object with the value, which is a string of a star pattern,
  - the String class's length method, and
- applying the JOptionPane class' message dialog and input dialog.

Preliminaries

1. Create a NetBeans or Eclipse Java project. Make sure you know the folder's name (workspace) where you store your project. The name of the project must be **lab02_<your FirstNameLastName >**. For example, my project would be named lab02_PeterNg.

2. Add a Java class named **StarPattern** to your project.

3. Add the following comment block to the beginning of your Java class (necessary for getting a grade):

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

4. Add another comment block shortly describing what task your class is responsible for (see the description of the assignments below).

Assignments

Solve the programming challenge, the problem "Star Pattern" at the end of Chapter 2, "Programming Challenges," in the textbook, with added requirements as explained below.

1. Rather than the pattern of the book, you must create the modified star pattern as shown in Figure 1:

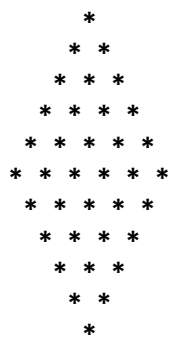


Figure 1

Define a method, constructStarsPattern(), to construct this modified star pattern and return the address where the star pattern is stored. The method is defined as follows:

```
public static String constructStarsPattern() {
      //The body of this method contains the solution of the
      //problem 2.
      . . .
      return definedPattern;
}//end of constructstartsPattern() method
```

2. In the constructStarsPattern() method, write definedPattern as a reference variable of the String class. The string value is a description of the star pattern given in 1. Write this description using the *, the **space** character, **\n** (newline character), and \t (tab character) as many times as needed as a string value.

```
public static String constructStarsPattern() {
      //declaration block
      …
      //The description of the defined starPattern is:
      definedPattern = new String("\n\t\t     *\n\t\t    * *"
                     + "\n\t\t   * * * . . ."
                     + . . .
                     + "\n\t\t    * *\n\t\t     *\n\t\t");
      //lengthOfDefinedPattern = definedPattern.length() ;
      //will yield 127 characters if you defined correctly.

      . . .
```

```
        return definedPattern;
    }//end of constructstartsPattern() method
```

3. Use the StarPattern class, where the above solution is used to construct a partial solution for the given problem. That is,

```
public class StarPattern {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        //declaration block.
        // …
        //a call statement to call constructStarsPattern().

    }//end of main() method

//Insert the solution of the problems 1 and 2 here.
//That means, insert the obtained constructStarsPattern()
//Method here.

}//end of class StarPattern
```

4. In the body of the main method, declare a single String class type variable named **`pattern`**. The **pattern** will hold the address of a String object with a string literal as its value, which is created in problem 2.

Write a call statement,

pattern = constructStarsPattern();

//Executing this statement, the reference variable, pattern, has the contents of //***definedPattern.***

5. In main(), in the declaration block, declare title as a reference variable containing the address where the string "Stars_Pattern" is stored. Then, write a call statement,

displayStarsPattern(title, pattern)

Define a method, displayStarsPattern(String, String), to display the information as shown in Figure 2, using println() and then printf() alternatively.

I. Use the String class' length method, length(), to print the length of the obtained **pattern** in 4 to the console. The length of this **pattern** is the total number of characters used in constructing this **pattern**. Use println() to write 5a. Use printf() to write 5b.

This yields the following in the console:

```
5a. The number of characters in the pattern is 127
5b. The number of characters in the pattern is 127.
```

II. Print the star pattern from the string value constructed in 4. Use a single print statement that prints a **`pattern`** to the console; you are not allowed to repeatedly use System.out.println to print the given star pattern line by line. The result of this print statement will display a star pattern, as shown in 1. Use println() to write 5c. Use printf() to write 5d.

This yields the following

```
5c. The stars' pattern is:

                *
               * *
              * * *
             * * * *
            * * * * *
           * * * * * *
            * * * * *
             * * * *
              * * *
               * *
                *

5d. The stars' pattern is:

                *
               * *
              * * *
             * * * *
            * * * * *
           * * * * * *
            * * * * *
             * * * *
              * * *
               * *
                *
```

Figure 2

III. Use the JOptionPane class' showMessageDialog method to complete the same tasks as problem 5 above. Display the length of the **pattern** and the star pattern, as shown in Figure 3. To do these tasks using the showMessageDialog method.
In the main(), write a call statement,

displayStarsPatternInMessageBox(title, pattern);

Define a method, displayStarsPatternInMessageBox(String, String), to display the information shown in Figure 3.

When you use the JOptionPane class, you must have the following import for the JOptionPane class to be placed before comment block 3 in the ***Preliminaries*** section.

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

Before the end of the main method, you are required to have

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

An example of an outline of the code is as follows:

```
import javax.swing.JOptionPane;  //required for JOptionPane class
public class StarPattern {

   public static void main(String[] args) {
        // TODO Auto-generated method stub
        //declaration block.
        // …
        //a call statement to call constructStarsPattern().
        // problem 4.
        pattern = constructStarsPattern();

        // problem 5, Part I.
        displayStarsPattern(title + " Pbm 5, Part I", pattern);
        // problem 5, Part II
        displayStarsPatternInMessageBox(title +
                                        " Pbm 5, Part II",
                                        pattern);


        System.exit(0);//required for JOptionPane class.
   }//end of main() method

   public static String constructStarsPattern(){
        . . .
      return definedPattern;
   } //end of constructStarsPattern()

   public static void displayStarsPattern(String title,
                                          String starsAddress){
      . . .
   } //end of displayStarsPattern(…)
```

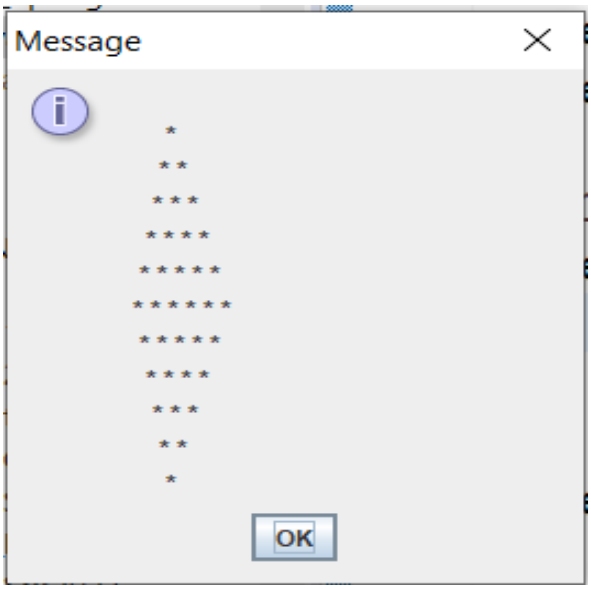
```
    public static void displayStarsPatternInMessageBox(String title,
                                    String strPattern){
          String title = "Stars Pattern";
          JOptionPane.showMessageDialog(null, strPattern);
          JOptionPane.showMessageDialog(null, strPattern, "5e: "
                                + title,
                               JOptionPane.INFORMATION_MESSAGE);

          . . .

    }//end of displayStarsPatternInMessageBox(…)


}//end of class StarPattern
```

6. Run your program to see if the pattern appears correctly. Remove errors and make corrections as necessary in 4. That means you must have the correct number of occurrences for the *'s with spaces, \n's, and \t's in a string literal. Repeat this until the star pattern is correctly displayed to both the console and the message dialog boxes.



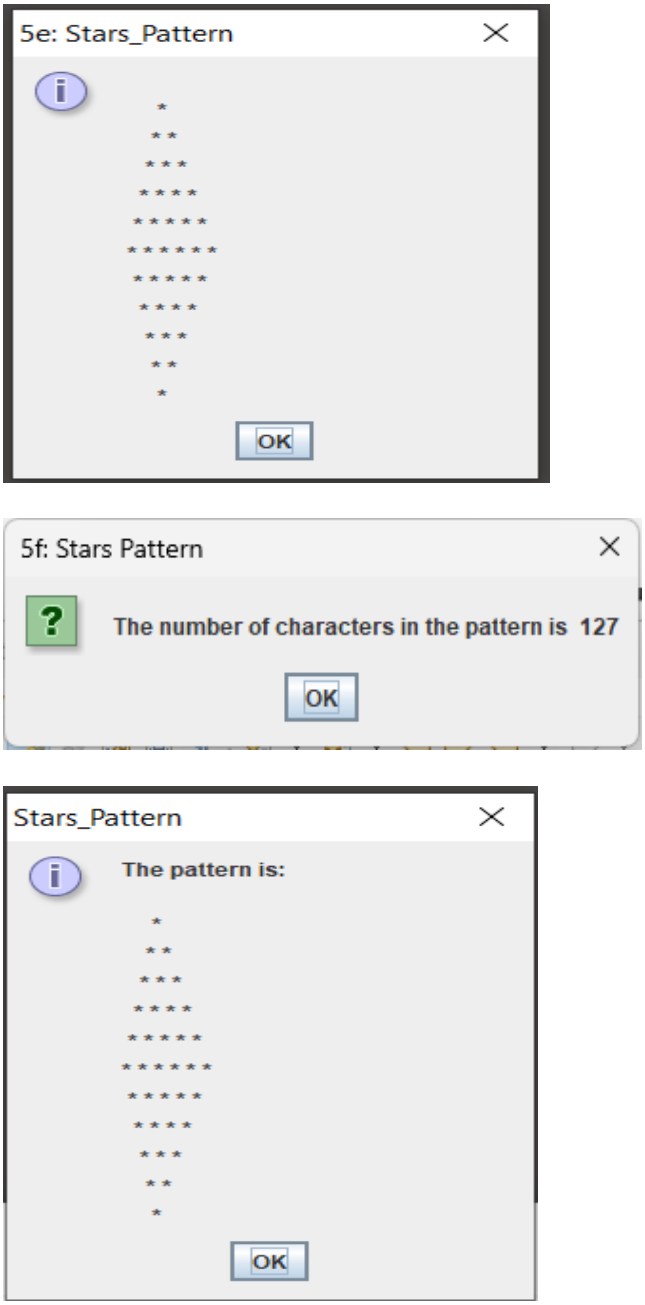


Figure 3.

Continue the program code as follows:

7. Declare an **`int`** type variable named **`asterisks`**

8. Write a formula that adds the number of asterisks in the individual lines of the pattern and assign the **`asterisks`** variable the formula. [Hint: The total number of asterisks depends upon the number of columns of stars! This is a simple formula.]

9. Separated by two blank lines from the above pattern printing, print the following added information to the console, arranged exactly as shown in Figure 4:

```
The number of * symbols in the lines of the pattern is
1,  2,  3,  4,  5,  6,  5,  4,  3,  2,  1
The total number of * symbols in the pattern is: 36
```

Figure 4

Likewise, use the JOptionPane class to display the result in Figure 4 to form the following message dialog boxes in Figure 5.

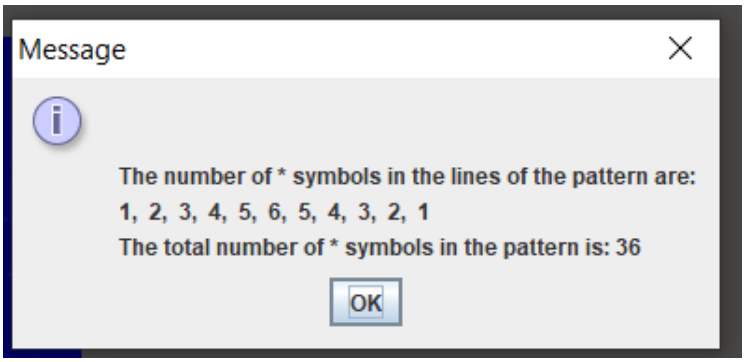
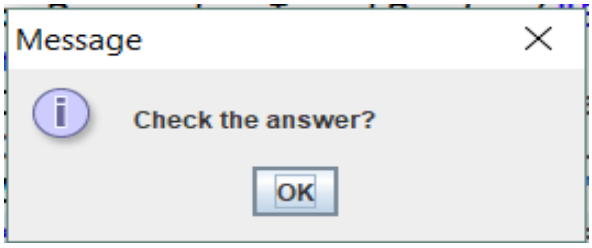
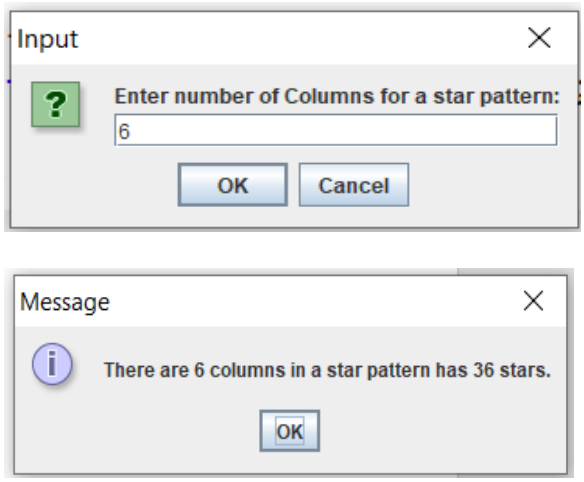


Figure 5.

10. Run your program and see if the second message prints correctly.

11. In the main() method, use the JOptionPane class's showInputDialog method to enter the number of columns for any star pattern (as shown in the top two dialog boxes). Then, write a method call statement, which passes the value of the number of columns to the called method for computing the number of asterisks and displaying "There are six (6) columns in a star pattern that has 36 stars" (as shown in the third dialog box). Use the showMessageDialog to output the following information, as shown in Figure 6.





If you enter 5 to the showInputDialog, the called method computes and displays "There are 6 columns in a star pattern that has 25 stars"

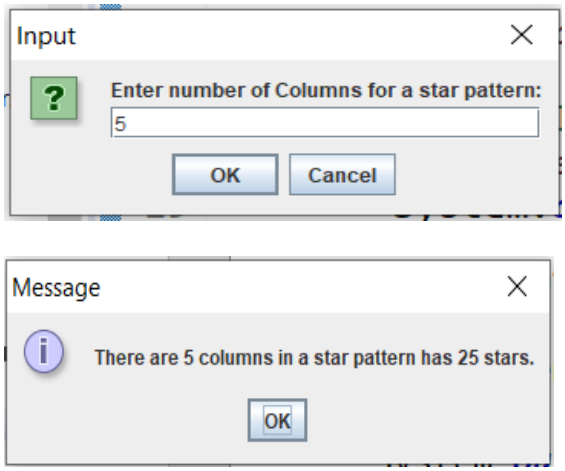


Figure 6

12. Ask the GTA or the instructor to verify your active presence in the lab. **Verification is necessary to grade your assignment.**

Submit: Zip your project folder and upload the zip file to Brightspace at the designated place.