CHANDRAKANTH HV - 26-JUN-2025

Activity 1:

List three tasks you perform regularly that involve repetition. For each:

1. What is being repeated?
2. What determines when it stops?

Algorithm :

1. Start the program.
2. Create a Scanner object to accept user input.
3. Prompt the user: "Enter how many minutes you want to brush your teeth:"
4. Read the input value and store it in `totalMinutes`.
5. Initialize `minutesBrushed` to 0.
6. While `minutesBrushed` is less than `totalMinutes`:
7. Display: `"Brushing teeth... minute (minutesBrushed + 1)"`
8. Increment `minutesBrushed` by 1.
9. After the loop ends, display `"Done brushing!"`.
10. End the program

Pseudo code :

```
START
Prompt the user to enter the number of minutes to brush
Read the user input and store it as totalMinutes
Set minutesBrushed = 0
WHILE minutesBrushed < totalMinutes DO
 Display "Brushing teeth... minute (minutesBrushed + 1)"
 Increment minutesBrushed by 1
END WHILE
Display "Done brushing!"
END
```

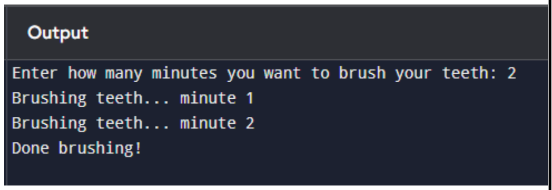
Code :

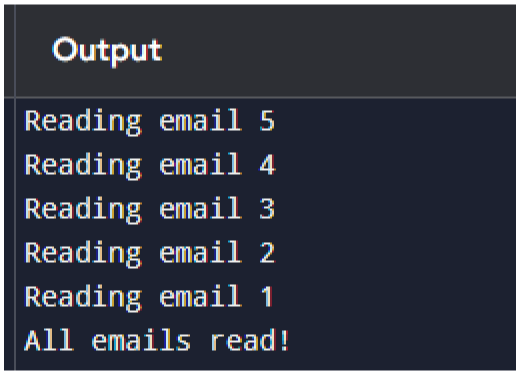
```
import java.util.Scanner;
```

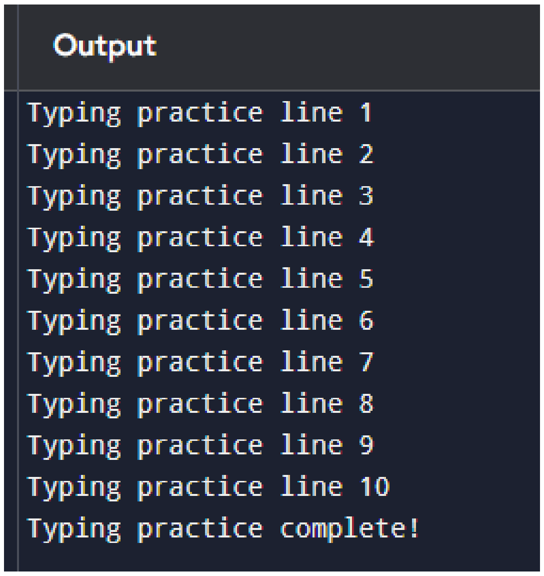
```
public class BrushingTeeth {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter how many minutes you want to brush your teeth: ");
        int totalMinutes = scanner.nextInt();

        int minutesBrushed = 0 ;

        while (minutesBrushed < totalMinutes) {
            System.out.println("Brushing teeth... minute " + (minutesBrushed + 1));
            minutesBrushed++;
        }

        System.out.println("Done brushing!");
    }
}
```

```
TC2 : public class TypingPractice {
    public static void main(String[] args) {
        int linesTyped = 0;
        int maxLines = 10;

        while (linesTyped < maxLines) {
            System.out.println("Typing practice line " + (linesTyped + 1));
            linesTyped++;
        }

        System.out.println("Typing practice complete!");
    }
}
```

Test cases	Input	Expected Out put

TC1	Enter how many minutes you want to brush your teeth: 2	 Output
		`Enter how many minutes you want to brush your teeth: 2`
		`Brushing teeth... minute 1`
		`Brushing teeth... minute 2`
		`Done brushing!`

TC2	Enter how many mails : 5	 Output
		`Reading email 5`
		`Reading email 4`
		`Reading email 3`
		`Reading email 2`
		`Reading email 1`
		`All emails read!`

TC3		 Output
		`Typing practice line 1`
		`Typing practice line 2`
		`Typing practice line 3`
		`Typing practice line 4`
		`Typing practice line 5`
		`Typing practice line 6`
		`Typing practice line 7`
		`Typing practice line 8`
		`Typing practice line 9`
		`Typing practice line 10`
		`Typing practice complete!`

Observation : This program simulates the real-life task of brushing teeth based on user-defined duration.It uses a `while` loop to repeat the brushing action for the number of minutes specified.The user can input any valid integer to customize the brushing duration.The output shows each minute of brushing and ends with a confirmation message.This demonstrates the use of loops, user input, and counters in Java programming.

Activity 2:

Write a Java program that prints the word `"Hello!"` ten times without using a loop.
After writing the code, reflect on:

What would happen if you had to print `"Hello!"` 1000 times instead?
Would you still write every line manually?

Algorithm :
Start the program.
Use `System.out.println()` to print `"Hello!"` ten times.
End the program

Pseudo code :

```
 START
Print "Hello!"
 END
```

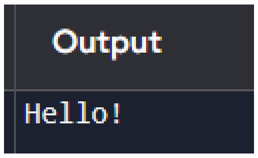
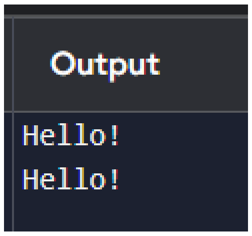
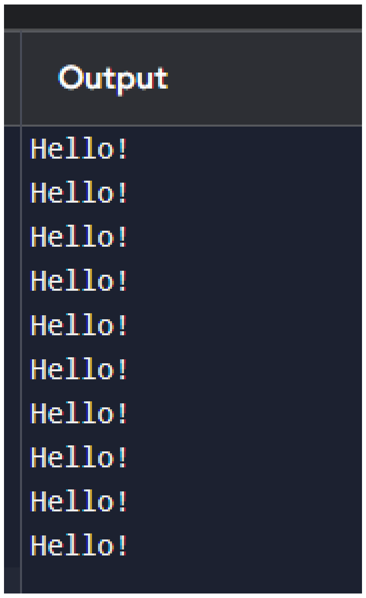
Code :

```
public class HelloPrinter {
    public static void main(String[] args) {
        System.out.println("Hello!");
}
}
```

For loop :

```
import java.util.Scanner;

public class HelloPrinter {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter how many times to print 'Hello!': ");
        int times = scanner.nextInt();
        if (times <= 0) {
            System.out.println("Please enter a positive number.");
        } else {
```

```
        for (int i = 1; i <= times; i++) {
            System.out.println("Hello!");
        }
    }

    scanner.close();  // Close scanner
  }
}
```

Test cases	Input	Expected Out put
TC1	1	 Output Hello!
TC2	2	 Output Hello! Hello!
TC3	For loop using : 10	 Output Hello! Hello! Hello! Hello! Hello! Hello! Hello! Hello! Hello! Hello!

Observation : The code successfully prints `"Hello!"` ten times without using loops.
However, the code becomes repetitive and long.
If the task was to print 100 or 1000 times, writing each line manually would be:
 Time-consuming
 Hard to read and maintain
 Not scalable
In real-world scenarios, we should use loops (`for`, `while`) to handle such repetition efficiently.This activity highlights why loops are important in programming to avoid code duplication.

SECTION 1

Problem Statement 1:
Print numbers from 10 to 1, then print “Blastoff!”

Algorithm :

Start the program.

Initialize a loop variable `i` with value `10`.

Repeat while `i` is greater than or equal to 1:

a. Print the value of `i`.
b. Decrease `i` by 1.

After the loop ends, print `"Blastoff!"`.

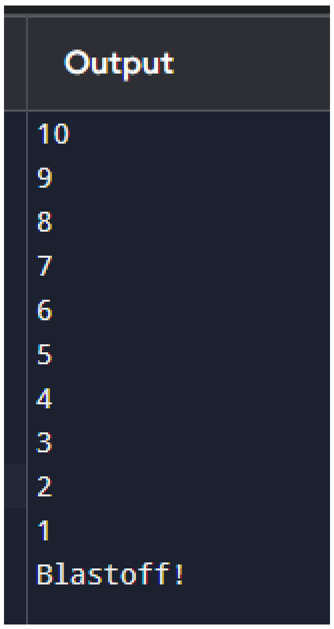
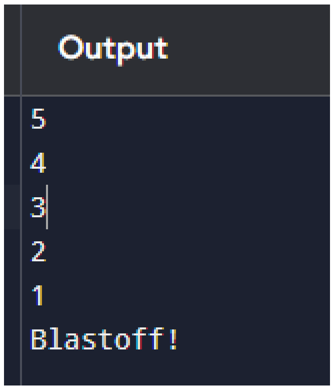
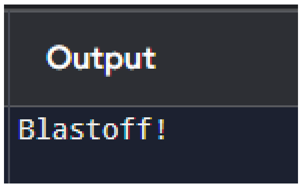
End the program.

Pseudo code :

```
START
FOR i from 10 down to 1 DO
PRINT i
END FOR
PRINT "Blastoff!"
END
```

Code :

```
public class Countdown {
   public static void main(String[] args) {
      for (int i = 10; i >= 1; i--) {
         System.out.println(i);
      }
      System.out.println("Blastoff!");
   }
}
```

Test cases	Input	Expected Out put
TC1	Input : from 10	 Output `10 9 8 7 6 5 4 3 2 1 Blastoff!`
TC2	Input : 5	 Output `5 4 3 2 1 Blastoff!`
TC3	Input : 0	 Output `Blastoff!`

Observation : The program uses a `for` loop to print numbers from 10 to 1 in descending order.After the countdown, it prints `"Blastoff!"` to simulate a rocket launch.This demonstrates control flow using loops, particularly counting in reverse.The program is simple and effective for teaching loop decrementing (`i--`).

Problem Statement 2 :
Ask the user to enter numbers repeatedly. Keep adding them until the user enters `0`. Then print the total sum.

Algorithm :

1. Start the program.
2. Declare variables `sum` and `number`.
3. Set `sum = 0`.
4. Use a `do-while` loop:
5. Prompt the user to enter a number.
6. Read the number into `number`.
7. Add the number to `sum`.
8. Repeat the loop until the number is 0.
9. After exiting the loop, display the value of `sum`.
10. End the program.

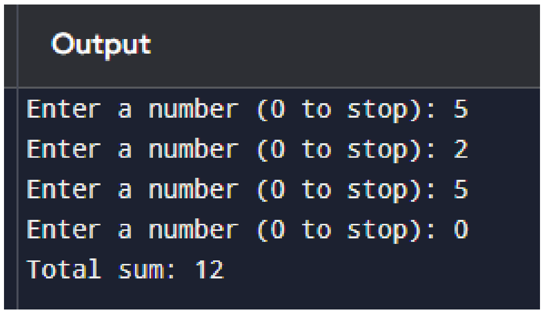
Pseudo code :

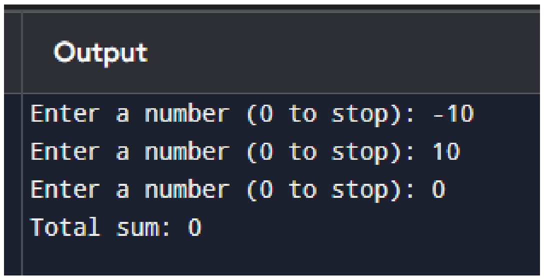
```
START
Initialize sum = 0
REPEAT
Prompt user to enter a number (0 to stop)
Read number
Add number to sum
UNTIL number is 0
Display total sum
END
```

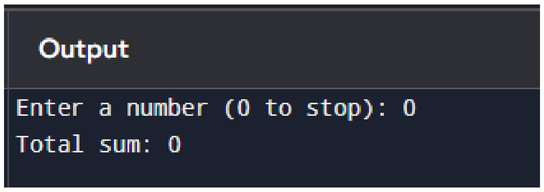
Code :

```
import java.util.Scanner;

public class SumUntilZero {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int number, sum = 0;
        do {
          System.out.print("Enter a number (0 to stop): ");
          number = scanner.nextInt();
          sum += number;
        } while (number != 0);

        System.out.println("Total sum: " + sum);
    }
}
```

Test cases	Input	Expected Out put

TC1	Enter a number (0 to stop): 5
	Enter a number (0 to stop): 2
	Enter a number (0 to stop): 5
	Enter a number (0 to stop): 0
		 Output
		`Enter a number (0 to stop): 5`
		`Enter a number (0 to stop): 2`
		`Enter a number (0 to stop): 5`
		`Enter a number (0 to stop): 0`
		`Total sum: 12`

TC2	Enter a number (0 to stop): -10
	Enter a number (0 to stop): 10
	Enter a number (0 to stop): 0
		 Output
		`Enter a number (0 to stop): -10`
		`Enter a number (0 to stop): 10`
		`Enter a number (0 to stop): 0`
		`Total sum: 0`

TC3	Enter a number (0 to stop): 0	 Output
		`Enter a number (0 to stop): 0`
		`Total sum: 0`

Observation :

The program accepts numbers from the user continuouslyThe loop runs at least once, which is why `do-while` is appropriate.When the user enters `0`, the loop stops.The program then displays the sum of all numbers entered (excluding the zero).This demonstrates:Input handling,Use of loops,Basic accumulator pattern

Problem Statement 3 :
Generate a random number between 1 and 10. Ask user to guess until correct.

Algorithm :

Start the program.

Create a `Scanner` object to read user input.

Ask User 1 to enter a secret number between 1 and 10

Display a message to pass the game to User 2.

Start a loop:

- Ask User 2 to guess the number.
- If the guess is lower than the secret number, print "Too low!"
- If the guess is higher than the secret number, print "Too high!"
- If the guess is equal, print "Correct!"

Repeat until the correct number is guessed

End the program.

Pseudo code :

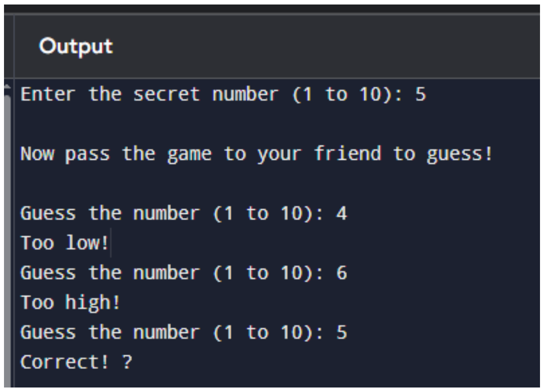
```
START
Create scanner
Prompt user to enter secret number
Store it in secretNumber
PRINT "Pass to your friend"
REPEAT
   Prompt user to guess number
   Read guess
   IF guess < secretNumber
   PRINT "Too low!"
   ELSE IF guess > secretNumber
   PRINT "Too high!"
   ELSE
```

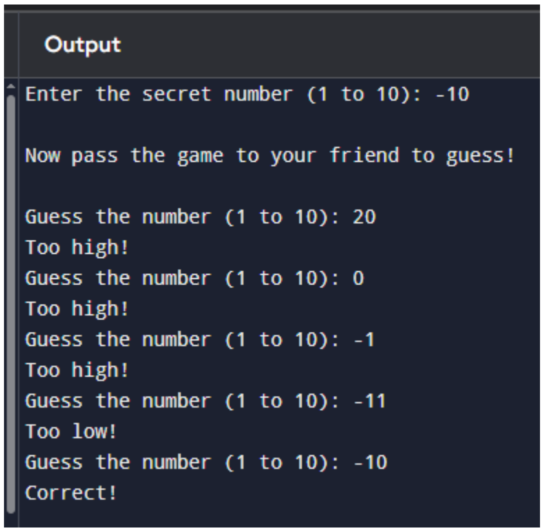
```
 PRINT "Correct!"
UNTIL guess == secretNumber
END
```

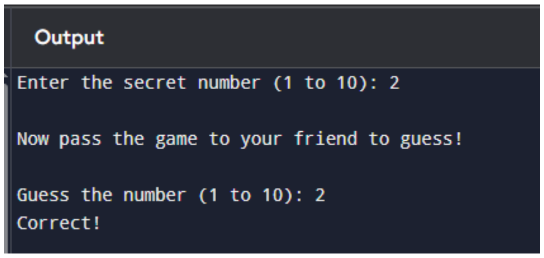
Code :

```
import java.util.Scanner;

public class GuessNumber {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the secret number (1 to 10): ");
        int secretNumber = scanner.nextInt();

        System.out.println("\nNow pass the game to your friend to guess!\n");
        int userGuess;
        do {
            System.out.print("Guess the number (1 to 10): ");
            userGuess = scanner.nextInt();

            if (userGuess < secretNumber) {
                System.out.println("Too low!");
            } else if (userGuess > secretNumber) {
                System.out.println("Too high!");
            } else {
                System.out.println("Correct! ");
            }

        } while (userGuess != secretNumber);
    }
}
```

Test cases	Input	Expected Out put

TC1	Enter a number (0 to stop): 5
	Enter a number (0 to stop): 2
	Enter a number (0 to stop): 5
	Enter a number (0 to stop): 0
		
		Output
		`Enter the secret number (1 to 10): 5`
		
		`Now pass the game to your friend to guess!`
		
		`Guess the number (1 to 10): 4`
		`Too low!`
		`Guess the number (1 to 10): 6`
		`Too high!`
		`Guess the number (1 to 10): 5`
		`Correct! ?`

TC2	Enter a number (0 to stop): -10
	Enter a number (0 to stop): 10
	Enter a number (0 to stop): 0
		
		Output
		`Enter the secret number (1 to 10): -10`
		
		`Now pass the game to your friend to guess!`
		
		`Guess the number (1 to 10): 20`
		`Too high!`
		`Guess the number (1 to 10): 0`
		`Too high!`
		`Guess the number (1 to 10): -1`
		`Too high!`
		`Guess the number (1 to 10): -11`
		`Too low!`
		`Guess the number (1 to 10): -10`
		`Correct!`

TC3	Enter a number (0 to stop): 2	
		Output
		`Enter the secret number (1 to 10): 2`
		
		`Now pass the game to your friend to guess!`
		
		`Guess the number (1 to 10): 2`
		`Correct!`

Observation :
This is a two-player guessing game:Player 1 sets the number.Player 2 tries to guess it.The loop continues until the guess is correct.It uses a `do-while` loop because at least one guess is required.This version is great for practicing:Input/output,Loops,Conditional statements

Problem Statement 4 :
Infinite loop debugging : analyze and fix

```
Int count = 0;
While (count < 5)
{
system.out.print ln("Hello");
}
```

Algorithm :

- Start the program.
- Create a Scanner object to read input.
- Ask the user: “How many times do you want to print 'Hello'?”
- Store the input in variable `times`
- Initialize `counter` to 0
- While `counter` is less than `times`:
  Print `"Hello"`
  Increase `counter` by 1
  After loop ends, close the Scanner.
- End the program.

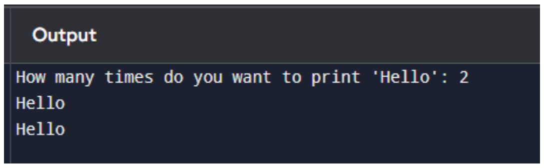
Pseudo code :

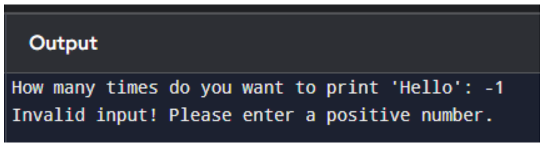
```
START
Create Scanner for input
Prompt user for number of times to print "Hello"
Read the number and store in variable 'times'
Set counter to 0
WHILE counter is less than times
   PRINT "Hello"
   INCREMENT counter
CLOSE Scanner
END
```

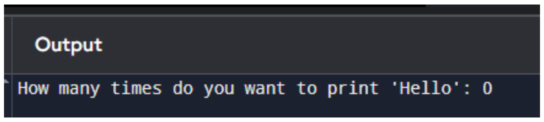
Code :

```
import java.util.Scanner;

public class UserInputLoop {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("How many times do you want to print 'Hello': ");
        int times = scanner.nextInt();

        if (times < 0) {
            System.out.println("Invalid input!.");
        } else {
            int counter = 0;

            while (counter < times) {
                System.out.println("Hello");
                counter++;
            }
        }

        scanner.close();
    }
}
```

Test cases	Input	Expected Out put

TC1	How many times do you want to print 'Hello': 2	 Output
		`How many times do you want to print 'Hello': 2`
		`Hello`
		`Hello`

TC2	How many times do you want to print 'Hello': -1	 Output
		`How many times do you want to print 'Hello': -1`
		`Invalid input! Please enter a positive number.`

TC3	How many times do you want to print 'Hello': 0	 Output `How many times do you want to print 'Hello': 0`

Observation :

This program prints "Hello" a user-defined number of times.It uses a while loop, which checks the condition before each execution.The use of `counter++` is important to avoid an infinite loop.Demonstrates basic programming concepts:User input (`Scanner`),Loops (`while`),Condition checkingThe program is dynamic: the user controls how many times the output is printed.If the user enters `0`, the loop won't run at all.

Section 2:
Problem Statement 5 :
Print even numbers from 2 to 20 using a for loop.

Algorithm :

Start the program.

Create a `Scanner` object to take input from the user.

Ask the user to enter a maximum number (should be ≥ 2).

Store the input in a variable called `max`.

Check if `max` is less than 2:

- If true, print "Invalid input!"
- Else, do the following:
    - Print a message: "Even numbers from 2 to max:"
    - Use a for loop starting from 2 up to `max` with step size 2.
    - Print each value of `i` during the loop.

Close the `Scanner`.

End the program.

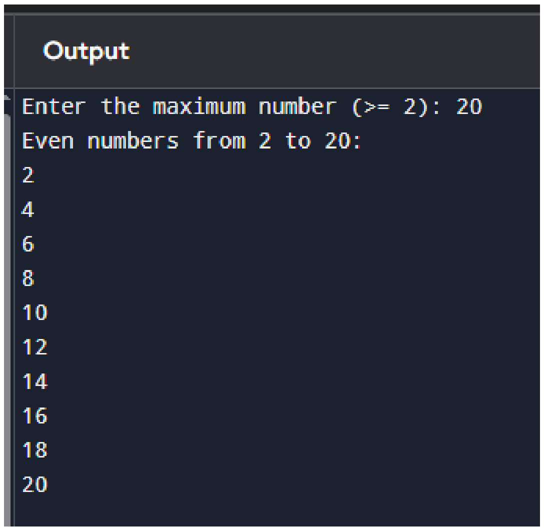
Pseudo code :
```
START
Create a scanner to read user input
Prompt user to enter a maximum number (>= 2)
Read the number into variable 'max'
IF max < 2 THEN
 PRINT "Invalid input!"
ELSE
PRINT "Even numbers from 2 to max"
FOR i from 2 to max, incrementing by 2
PRINT i
END IF
```

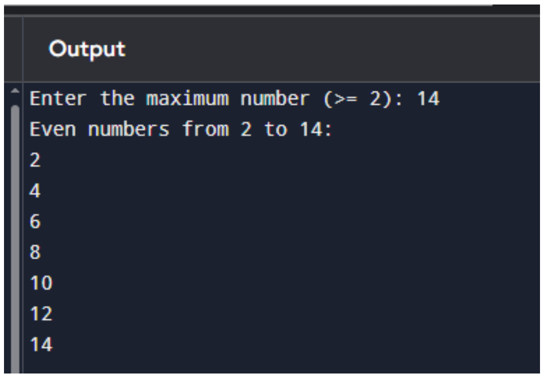
```
Close scanner
END
```

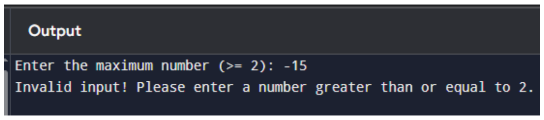
Code :

```
import java.util.Scanner;

public class EvenNumbers {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the maximum number (>= 2): ");
        int max = scanner.nextInt();

        if (max < 2) {
            System.out.println("Invalid input! Please enter a number greater than or equal to 2.");
        } else {
            System.out.println("Even numbers from 2 to " + max + ":");
            for (int i = 2; i <= max; i += 2) {
                System.out.println(i);
            }
        }

        scanner.close();
    }
}
```

Test cases	Input	Expected Out put

TC1	Enter the maximum number (>= 2): 20	 Output
		`Enter the maximum number (>= 2): 20`
		`Even numbers from 2 to 20:`
		`2`
		`4`
		`6`
		`8`
		`10`
		`12`
		`14`
		`16`
		`18`
		`20`

TC2	Enter the maximum number (>= 2): 14	 Output
		`Enter the maximum number (>= 2): 14`
		`Even numbers from 2 to 14:`
		`2`
		`4`
		`6`
		`8`
		`10`
		`12`
		`14`

TC3	Enter the maximum number (>= 2): -15	 Output
		`Enter the maximum number (>= 2): -15`
		`Invalid input! Please enter a number greater than or equal to 2.`

Observation :
This program demonstrates input validation for numeric limits.It ensures the user cannot proceed with a number less than 2.The use of a for loop with `i += 2` makes it efficient to directly print even numbers without additional condition checks.The program is simple and interactive, suitable for beginners learning loops and conditions.If the user enters a valid number (e.g. 10), it prints: `2 4 6 8 10`If the user enters a number like `1` or `-5`, it immediately responds with an error message.

Problem Statement 6 :
Calculate `n!` for a user-input number n. Handle the edge case when `n == 0` (since 0! = 1).

Algorithm :

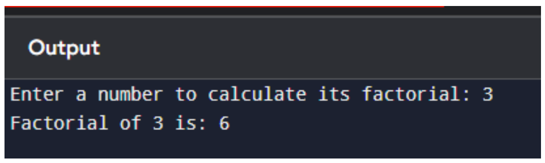
1. Start the program.
2. Create a `Scanner` object to read user input.
3. Ask the user to enter a number.
4. Store the number in variable n.
5. Check if n is less than 0:
   a. If yes, print: "Factorial is not defined for negative numbers."
   b. If no:
      i. Initialize a variable `factorial` to 1.
      ii. Loop from `i = 1` to `i <= n`, multiply `factorial` by `i` on each iteration.
      iii. After the loop ends, print the value of `factorial`.
6. Close the scanner.
7. End the program.

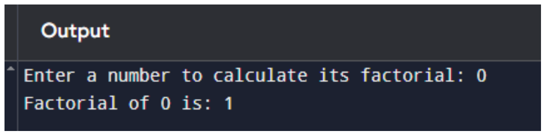
Pseudo code :
```
START
Create a scanner to read user input
Prompt user to enter a maximum number (>= 2)
Read the number into variable 'max'
IF max < 2 THEN
 PRINT "Invalid input!"
ELSE
PRINT "Even numbers from 2 to max"
FOR i from 2 to max, incrementing by 2
PRINT i
END IF
Close scanner
END
```

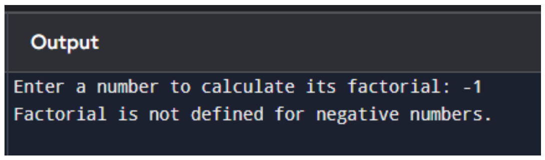
Code :

```
import java.util.Scanner;

public class FactorialCalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a number to calculate its factorial: ");
        int n = scanner.nextInt();
        if (n < 0) {
            System.out.println("Factorial is not defined for negative numbers.");
        } else {
            long factorial = 1;
            for (int i = 1; i <= n; i++) {
                factorial *= i;
            }

            System.out.println("Factorial of " + n + " is: " + factorial);
        }

        scanner.close();
    }
}
```

Test cases	Input	Expected Out put

TC1	Enter a number to calculate its factorial: 3	 Output
		`Enter a number to calculate its factorial: 3`
		`Factorial of 3 is: 6`

TC2	Enter a number to calculate its factorial: 0	 Output
		`Enter a number to calculate its factorial: 0`
		`Factorial of 0 is: 1`

TC3	Enter a number to calculate its factorial: -1	 Output
		`Enter a number to calculate its factorial: -1`
		`Factorial is not defined for negative numbers.`

Observation :

This program calculates the factorial (n!) of a non-negative integer using a `for` loop.It correctly handles edge cases:For `n = 0`, the result is 1 (as 0! = 1).For `n < 0`, it prints an error message.The result is stored in a `long` variable to support larger outputs.It's a good example of:Input handling,Looping,Conditional logic,Basic error checking

If the user enters 5, the output will be:

```
Factorial of 5 is: 120
```

Problem Statement 7 :
Ask for a string input. Count how many times `'a'` or `'A'` appears.

Algorithm :

8. Start the program.
9. Create a `Scanner` object to take string input from the user.
10. Prompt the user to enter a string and store it in a variable called `input`.
11. Initialize a variable `count` to 0.
12. Use a `for` loop to iterate through each character in the string:
13. Extract the character at the current position.
14. Check if the character is `'a'` or `'A'`.
15. If yes, increment `count` by 1.
16. After the loop ends, print the total count of `'a'` and `'A'`
17. Close the scanner.
18. End the program.

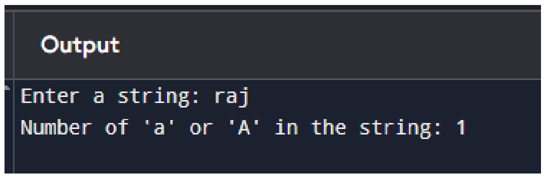
Pseudo code :

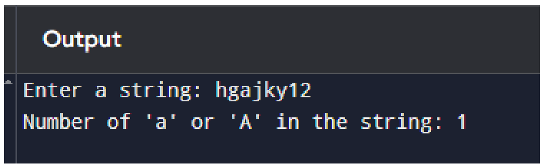
```
START
Create Scanner to read user input
PRINT "Enter a string:"
READ string into variable 'input'
SET count = 0
FOR i from 0 to length of input - 1
SET ch = character at position i
 IF ch == 'a' OR ch == 'A'
 INCREMENT count by 1
END FOR
PRINT "Number of 'a' or 'A' in the string: " + count
CLOSE scanner
END
```

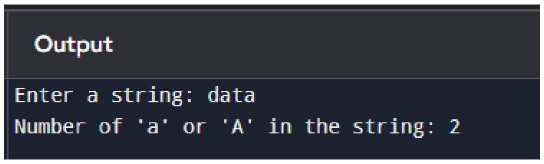
Code :

```java
import java.util.Scanner;

public class CountA {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a string: ");
        String input = scanner.nextLine();

        int count = 0;

        for (int i = 0; i < input.length(); i++) {
            char ch = input.charAt(i);

            if (ch == 'a' || ch == 'A') {
                count++;
            }
        }

        System.out.println("Number of 'a' or 'A' in the string: " + count);

        scanner.close();
    }
}
```

Test cases	Input	Expected Out put

TC1	Enter a string: raj	 Output
		`Enter a string: raj`
		`Number of 'a' or 'A' in the string: 1`

TC2	Enter a string: hgajky12	 Output
		`Enter a string: hgajky12`
		`Number of 'a' or 'A' in the string: 1`

TC3	Enter a string: data	 Output
		`Enter a string: data`
		`Number of 'a' or 'A' in the string: 2`

Observation :
This program counts how many times the letter `'a'` or `'A'` appears in a user-entered string.It works by:Looping through each character in the string.Using `charAt(i)` to access each character.Comparing each character to `'a'` and `'A'`.It uses case-insensitive checking for just one letter.The logic is efficient for small and large strings.

Problem Statement 8 :
Print a single line with five stars (*), using one `for` loop.

Algorithm :

- Start the program.
- Create a `Scanner` object to read user input.
- Prompt the user to enter the number of stars (n).
- Store the input in variable n.
- Check if `n <= 0`
- If true, print "`Invalid input! Please enter a positive number.`
- Else:
- Run a for loop from `i = 1` to `i <= n`:
  In each iteration, print a `*` on the same line.
  After the loop, print a newline to finish the output nicely.
- Close the scanner.
  End the program.

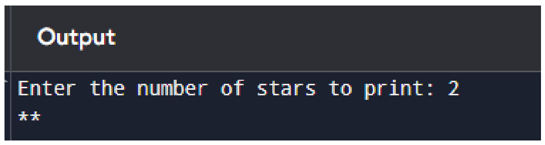
Pseudo code :

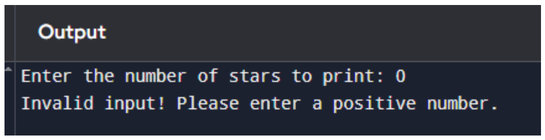
```
START
Create Scanner to take input from user
PRINT "Enter the number of stars to print: "
READ integer n
IF n <= 0 THEN
 PRINT "Invalid input! Please enter a positive number."
ELSE
 FOR i from 1 to n
 PRINT "*" (on same line)
 END FOR
PRINT newline
END IF
```

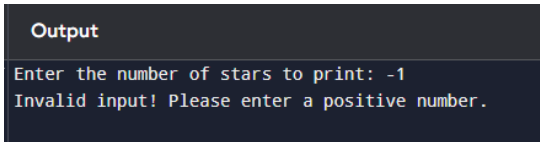
Code :

```
import java.util.Scanner;

public class StarPattern {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of stars to print: ");
        int n = scanner.nextInt();

        if (n <= 0) {
            System.out.println("Invalid input! Please enter a positive number.");
        } else {
            for (int i = 1; i <= n; i++) {
                System.out.print("*");
            }
            System.out.println(); // Move to next line after printing stars
        }

        scanner.close();
    }
}
```

Test cases	Input	Expected Out put

TC1	Enter the number of stars to print: 2	 Output
		`Enter the number of stars to print: 2`
		`**`

TC2	Enter the number of stars to print: 0	 Output
		`Enter the number of stars to print: 0`
		`Invalid input! Please enter a positive number.`

TC3	Enter the number of stars to print: -1	 Output
		`Enter the number of stars to print: -1`
		`Invalid input! Please enter a positive number.`

Observation :

The program prints n number of asterisks (*) on the same line.It performs input validation:Only accepts positive integers if a negative number or 0 is entered, it prints an error message.Useful concepts demonstrated:`Scanner` for input,`if-else` for input validation,`for` loop for repetition,`System.out.print()` vs `System.out.println(`

Section 3 :
Problem Statement 9 :
Check if a number is prime using a loop and break

Algorithm :

Start the program.
Create a `Scanner` object to read user input.

Prompt the user to enter the number of stars (n).
Store the input in variable n.
Check if `n <= 0`:
If true, print `"Invalid input! Please enter a positive number."`
Else:
Run a for loop from `i = 1` to `i <= n`:
In each iteration, print a `*` on the same line

After the loop, print a newline to finish the output nicely.

Close the scanner.

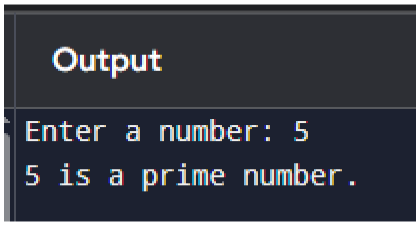
End the program

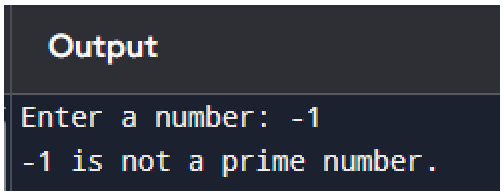
Pseudo code :
```
START
Create Scanner to take input from user
PRINT "Enter the number of stars to print: "
READ integer n
IF n <= 0 THEN
PRINT "Invalid input! Please enter a positive number."
ELSE
FOR i from 1 to n
PRINT "*" (on same line)
END FOR
PRINT newline
END IF
CLOSE scanner
END
```

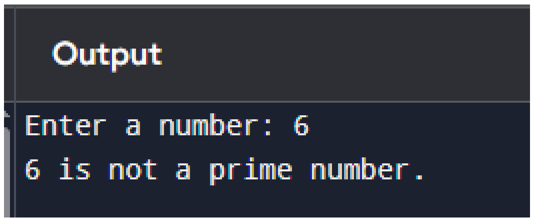
Code :

```
import java.util.Scanner;

public class SimplePrimeCheck {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a number: ");
        int num = scanner.nextInt();
        boolean isPrime = true;

        if (num <= 1) {
            isPrime = false;
        } else {
            for (int i = 2; i < num; i++) {
                if (num % i == 0) {
                    isPrime = false;
                    break;
                }
            }
        }

        if (isPrime) {
            System.out.println(num + " is a prime number.");
        } else {
            System.out.println(num + " is not a prime number.");
        }

        scanner.close();
    }
}
```

Test cases	Input	Expected Out put

TC1	Enter a number: 5	 Output
		`Enter a number: 5`
		`5 is a prime number.`

TC2	Enter a number: -1	 Output
		`Enter a number: -1`
		`-1 is not a prime number.`

TC3	Enter a number: 6	 Output
		`Enter a number: 6`
		`6 is not a prime number.`

Observation :

The program prints n number of asterisks (`*`) on the same line.
It performs input validation:
Only accepts positive integers.
If a negative number or 0 is entered, it prints an error message.
Useful concepts demonstrated:
`Scanner` for input
`if-else` for input validation
`for` loop for repetition

Problem Statement 10:
Input 5 numbers, use continue to skip negative ones and sum the rest

Algorithm :

- Start
- Initialize `sum = 0`
- Repeat steps 4–7 for `i = 1` to `5`
- Prompt: "Enter number i"
- Read number → `num`
- If `num < 0`, skip to next iteration (`continue`)
- Else, add `num` to `sum`
- After loop ends, print `sum`
- End

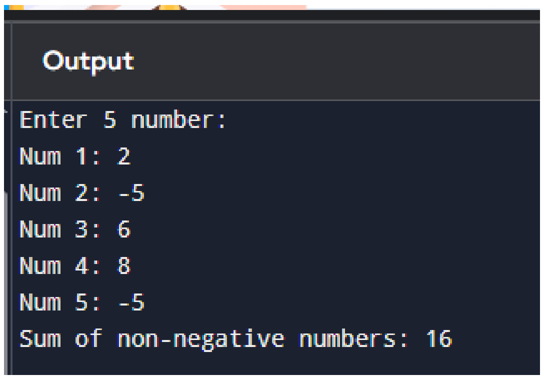
Pseudo code :

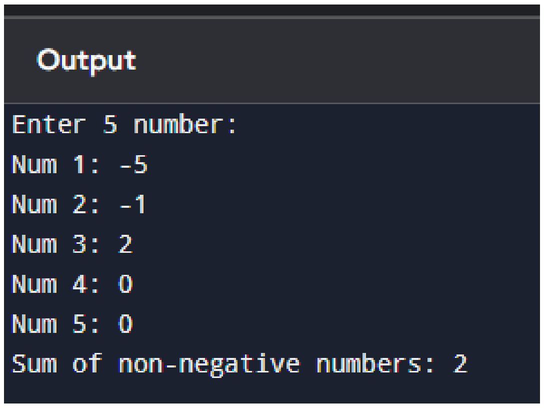
```
Start
Initialize sum as 0
Print "Enter 5 numbers"
Repeat for i = 1 to 5:
    Prompt user to enter number i
    Read number
    If number is negative:
        Skip this number (continue loop)
    Add number to sum
Print sum of non-negative numbers
End
```

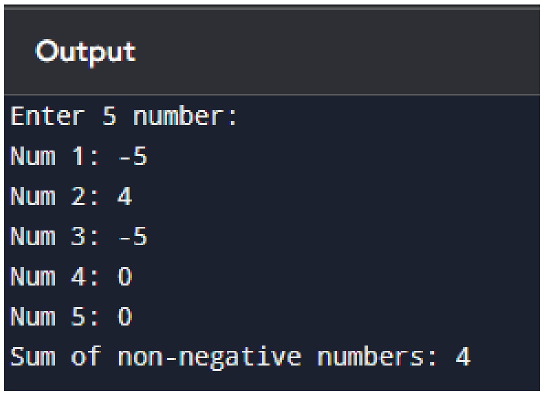
Code :

```
import java.util.Scanner;

public class SkipNegativeNumbers {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int sum = 0;
        System.out.println("Enter 5 number:");

        for (int i = 1; i <= 5; i++) {
            System.out.print("Num " + i + ": ");
            int num = scanner.nextInt();
            if (num < 0) {

                continue;
            }

            sum += num;
        }

        System.out.println("Sum of non-negative numbers: " + sum);

        scanner.close();
    }
}
```

Test cases	Input	Expected Out put

TC1	Enter a number: 5	Output
		`Enter 5 number:`
		`Num 1: 2`
		`Num 2: -5`
		`Num 3: 6`
		`Num 4: 8`
		`Num 5: -5`
		`Sum of non-negative numbers: 16`

TC2	Enter a number: -1	Output
		`Enter 5 number:`
		`Num 1: -5`
		`Num 2: -1`
		`Num 3: 2`
		`Num 4: 0`
		`Num 5: 0`
		`Sum of non-negative numbers: 2`

TC3	Enter a number: 6	Output
		`Enter 5 number:`
		`Num 1: -5`
		`Num 2: 4`
		`Num 3: -5`
		`Num 4: 0`
		`Num 5: 0`
		`Sum of non-negative numbers: 4`

Observation :
The program demonstrates the use of loop control (continue) to skip unwanted input.Negative numbers are not processed or added to the sum.Input is taken 5 times regardless of whether the number is valid or not.It's an efficient way to filter out unwanted values during input.`continue` helps in maintaining code clarity and avoiding nested `if-else` structures.

Problem Statement 11:
Rectangle Pattern . input rows and cols, print a rectangle of *

Algorithm :

- Start
- Input the number of rows and store in variable `rows`
- Input the number of columns and store in variable `cols`
- Repeat the following steps for `i = 1` to `rows`
- For each `i`, repeat the following for `j = 1` to `cols`:
    - Print `"*"` without moving to the next line
- After inner loop ends, print a newline to move to the next row
- End

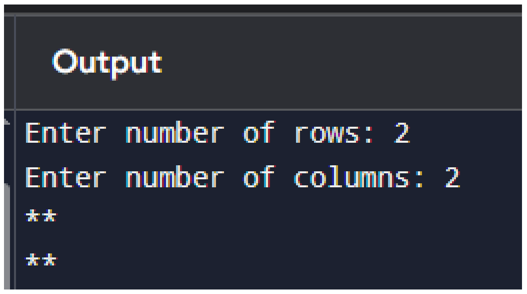
Pseudo code :

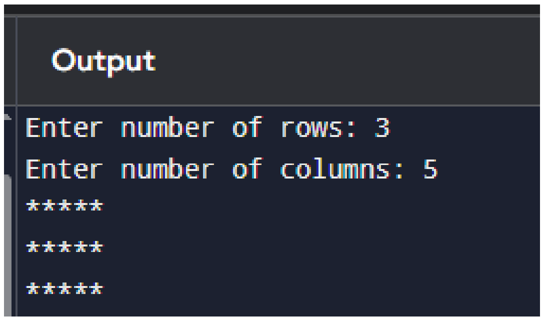
```
Start
Prompt user to enter number of rows
Read rows
Prompt user to enter number of columns
Read columns
For i = 1 to rows:
   For j = 1 to columns:
      Print "*", staying on the same line
   Move to next line after printing one row

End
```

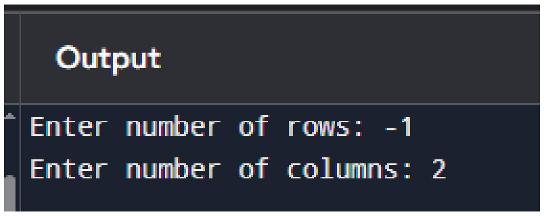
Code :

```
import java.util.Scanner;

public class RectanglePattern {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter number of rows: ");
        int rows = scanner.nextInt();

        System.out.print("Enter number of columns: ");
        int cols = scanner.nextInt();

        for (int i = 1; i <= rows; i++) {
            for (int j = 1; j <= cols; j++) {
                System.out.print("*");
            }
            System.out.println();
        }

        scanner.close();
    }
}
```

Test cases	Input	Expected Out put

TC1	Enter a number: 2,3	 Output
		`Enter number of rows: 2`
		`Enter number of columns: 2`
		`**`
		`**`

TC2	Enter a number: 3,5	 Output
		`Enter number of rows: 3`
		`Enter number of columns: 5`
		`*****`
		`*****`
		`*****`

TC3	Enter a number: -1,5	 Output
		`Enter number of rows: -1`
		`Enter number of columns: 2`

Observation :
The program uses nested loops:The outer loop controls rows, and the inner loop controls columns.It prints a solid rectangle of asterisks (*).Each row contains the same number of asterisks equal to the number of columns.The shape and size of the rectangle depend entirely on the user input.

Problem Statement 12:
Triangle Pattern input height , print right angled triangle with*

Algorithm :

- Start
- Prompt the user: "Enter the height of the triangle"
- Read the input and store it in variable `height`
- Repeat for `i = 1` to `height` (inclusive)
- Repeat for `j = 1` to `i` (inclusive)
    - Print `*` (without newline)
- Print newline (`System.out.println()`) to move to next row
- End

Pseudo code :

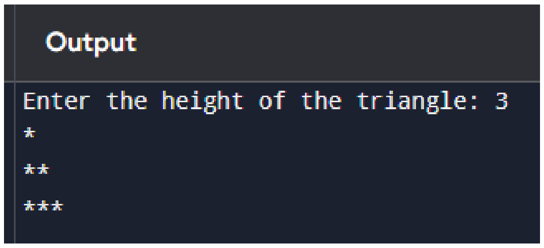
```
Start
Prompt user to enter height of the triangle
Read height
For i = 1 to height:
   For j = 1 to i:
      Print "*", staying on same line
   Move to the next line
End
```

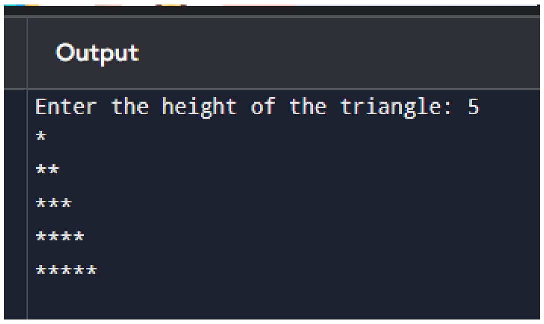
Code :

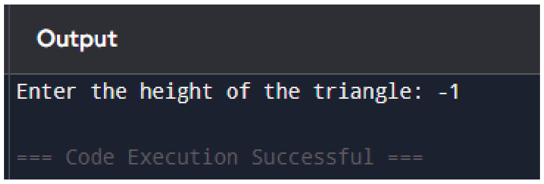
```
import java.util.Scanner;

public class TrianglePattern {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter the height of the triangle: ");
      int height = scanner.nextInt();
      for (int i = 1; i <= height; i++) {
         for (int j = 1; j <= i; j++) {
            System.out.print("*");
         }
         System.out.println();
      }

     scanner.close();
```

```
    }
  }
```

Test cases	Input	Expected Out put

TC1	Enter a number: 3	 Output
		`Enter the height of the triangle: 3`
		`*`
		`**`
		`***`

TC2	Enter a number: 5	 Output
		`Enter the height of the triangle: 5`
		`*`
		`**`
		`***`
		`****`
		`*****`

TC3	Enter a number: -1	 Output
		`Enter the height of the triangle: -1`
		
		`=== Code Execution Successful ===`

Observation :

This program prints a right-angled triangle using asterisks (*).The number of rows (height of the triangle) is taken from user input.The pattern grows by one * per row from top to bottom.It uses nested loops:The outer loop controls the number of rows.The inner loop controls how many * to print in each row

Problem Statement 13:
Pyramid Pattern Challenge input height print centered pyramid

Algorithm :

- Start
- Prompt the user: "Enter the height of the pyramid
- Read input → store in variable `height`
- Repeat steps 5–7 for `i = 1` to `height`
- Print `height - i` spaces
- Print 2 `* i - 1` stars (`*`)
- Move to next line (`System.out.println()`)
- After loop ends, close the scanner
- End

Pseudo code :

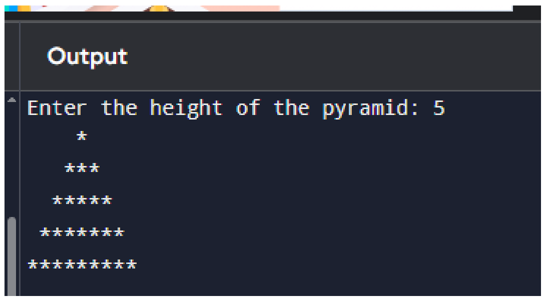
```
tart
Prompt user to enter height of the pyramid
Read height
For i = 1 to height:
  Print (height - i) spaces
  Print (2 * i - 1) stars
  Move to next line
End
```

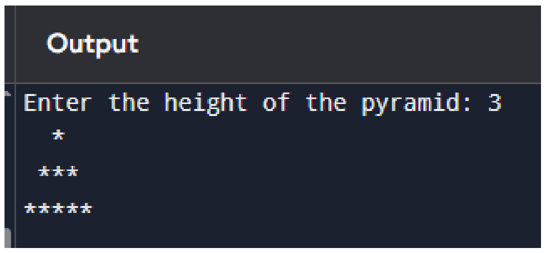
Code :

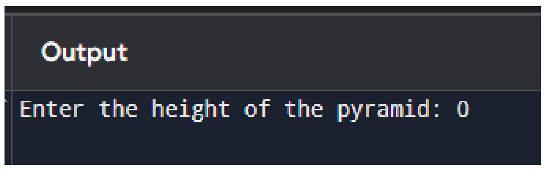
```
import java.util.Scanner;

public class PyramidPattern {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.print("Enter the height of the pyramid: ");
    int height = scanner.nextInt();

    for (int i = 1; i <= height; i++) {

      for (int space = 1; space <= height - i; space++) {
```

```
                System.out.print(" ");
            }

            for (int star = 1; star <= 2 * i - 1; star++) {
                System.out.print("*");
            }

            System.out.println();
        }

        scanner.close();
    }
}
```

Test cases	Input	Expected Out put

TC1	Enter a number: 5	 Output
		`Enter the height of the pyramid: 5`
		`    *`
		`   ***`
		`  *****`
		` *******`
		`*********`

TC2	Enter a number: 3	 Output
		`Enter the height of the pyramid: 3`
		`  *`
		` ***`
		`*****`

TC3	Enter a number: 0	 Output `Enter the height of the pyramid: 0`

Observation :

The program prints a center-aligned pyramid made of stars *.The height of the pyramid (number of rows) is taken as user input.For each row:

First, it prints some spaces to center-align the stars.Then, it prints odd number of stars, increasing in each row