Day 3 Shrayanth S 26/06/2025

Activity 3.1: Repetitive Tasks List three tasks you perform regularly that involve repetition. For each:

1. What is being repeated?

2. What determines when it stops?

A) **Brushing Teeth**

- **What is being repeated?**
- Brushing back and forth on each side of the teeth.
- **What determines when it stops?**
- After brushing for around 2 minutes or once all areas of the mouth have been cleaned.

B) **Commuting to School/Work**

- **What is being repeated?**
- Traveling the same route daily.
- **What determines when it stops?**
- Reaches the destination (school/workplace).

C) **Checking Phone for Notifications**

- **What is being repeated?**
- Unlocking the phone and checking messages or apps.
- **What determines when it stops?**
- When there are no new notifications or tasks to respond to.

Activity 3.2: Code Duplication Write how you would print “Hello!” 10 times without loops. Reflect on how loops make this easier for 1000 times.

Printing “Hello!” 10 times without loops:

```
System.out.println("Hello!");

System.out.println("Hello!");

System.out.println("Hello!");

System.out.println("Hello!");

System.out.println("Hello!");

System.out.println("Hello!");

System.out.println("Hello!");

System.out.println("Hello!");

System.out.println("Hello!");

System.out.println("Hello!");
```

Observation :

Without loops, we must manually repeat the print statement, which is time-consuming and prone to error.

Using a loop like for makes it easier and more efficient, especially when printing 1000 times:

```
for(int i = 0; i < 1000; i++) {

   System.out.println("Hello!");

}
```

Problem 1.1: Countdown

Print numbers from 10 to 1, then 'Blastoff!'

Algorithm:

Start from 10 and count down to 1

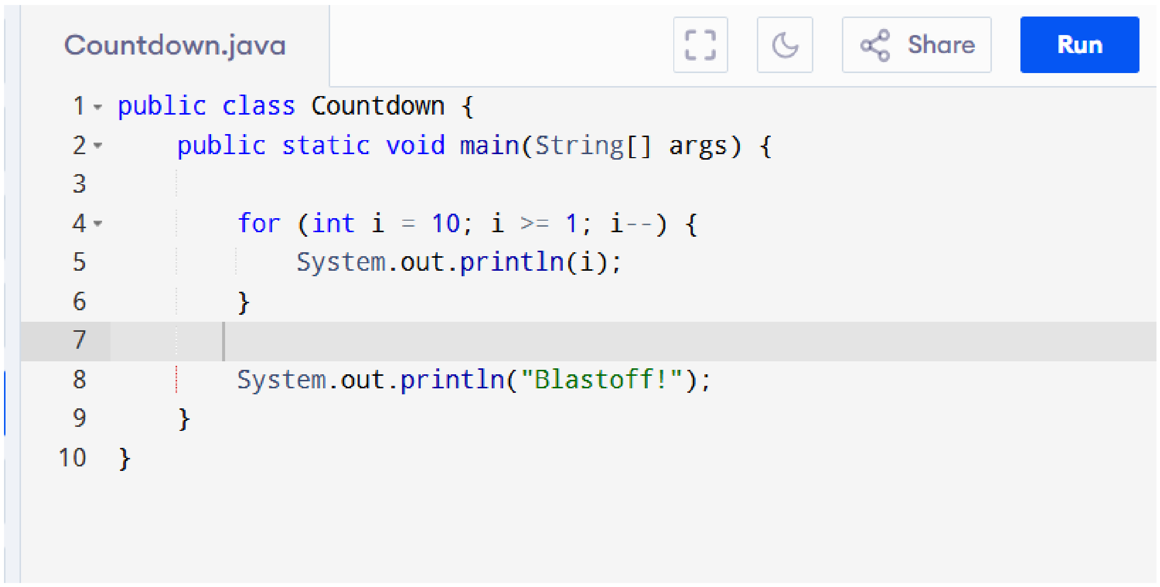
Print each number

After the loop, print 'Blastoff!'

Pseudocode:

```
for i from 10 down to 1:
    print i
print 'Blastoff!'
```

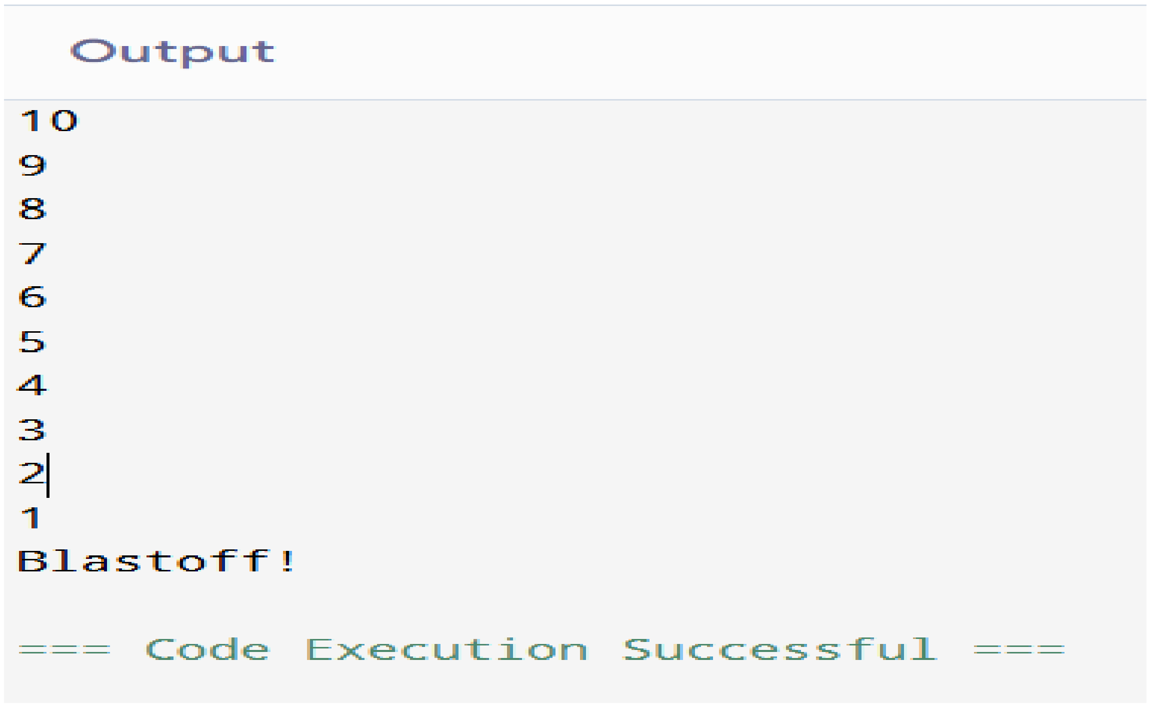
Java Code:



```java
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 Case: N/A (no input required)

Output:



```
10
9
8
7
6
5
4
3
2
1
Blastoff!

=== Code Execution Successful ===
```

Problem 1.2: Sum Until Zero

Ask user for numbers repeatedly until they enter 0. Sum and print the total.

Algorithm:

Initialize sum to 0

Loop: ask user for a number

If number is 0, break the loop

Add number to sum

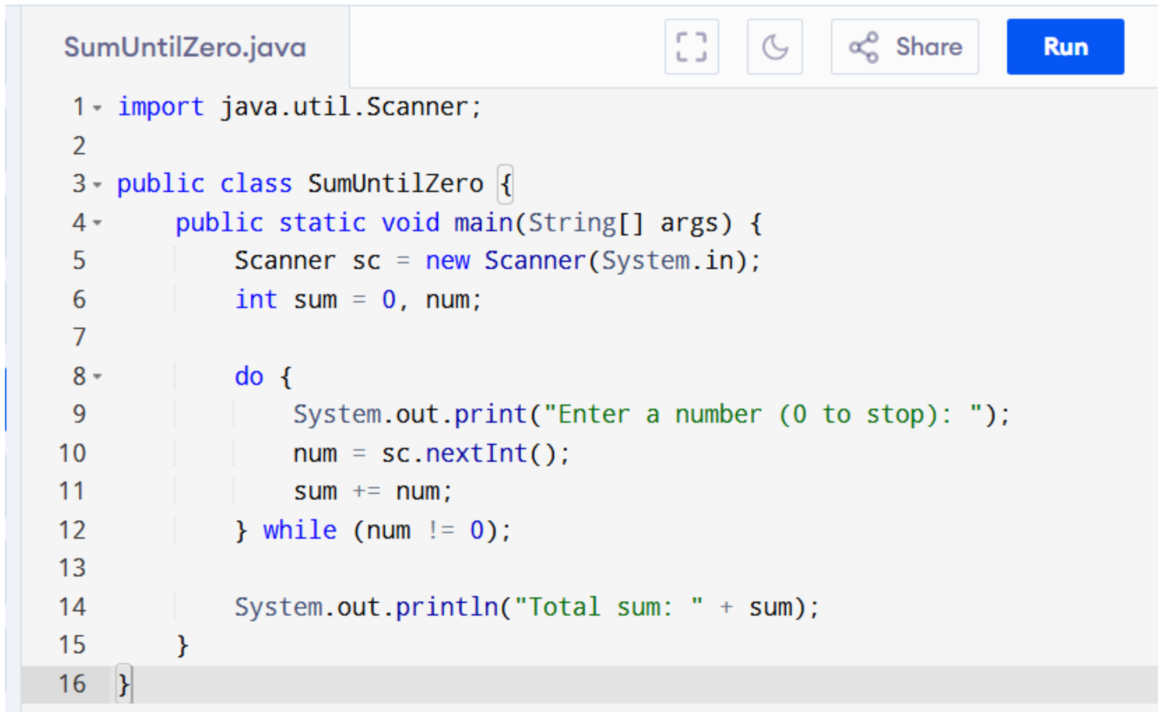
After loop, print sum

Pseudocode:

```
sum = 0
repeat:
  input number
```

```
    if number == 0: break
    sum = sum + number
print sum
```

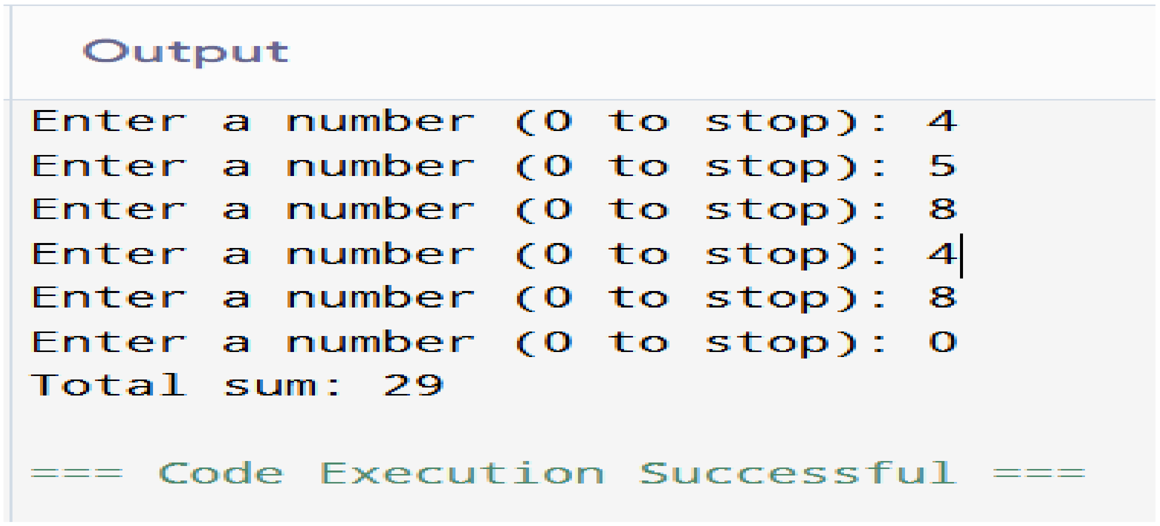
Java Code:



```java
import java.util.Scanner;

public class SumUntilZero {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int sum = 0, num;

        do {
            System.out.print("Enter a number (0 to stop): ");
            num = sc.nextInt();
            sum += num;
        } while (num != 0);

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

Test Case: Input: 4 5 8 4 8 0

Output:



```
Enter a number (0 to stop): 4
Enter a number (0 to stop): 5
Enter a number (0 to stop): 8
Enter a number (0 to stop): 4
Enter a number (0 to stop): 8
Enter a number (0 to stop): 0
Total sum: 29

=== Code Execution Successful ===
```

Problem 1.3: Guess the Number

Generate a random number between 1 and 10. Ask user to guess. Provide feedback and loop until correct.

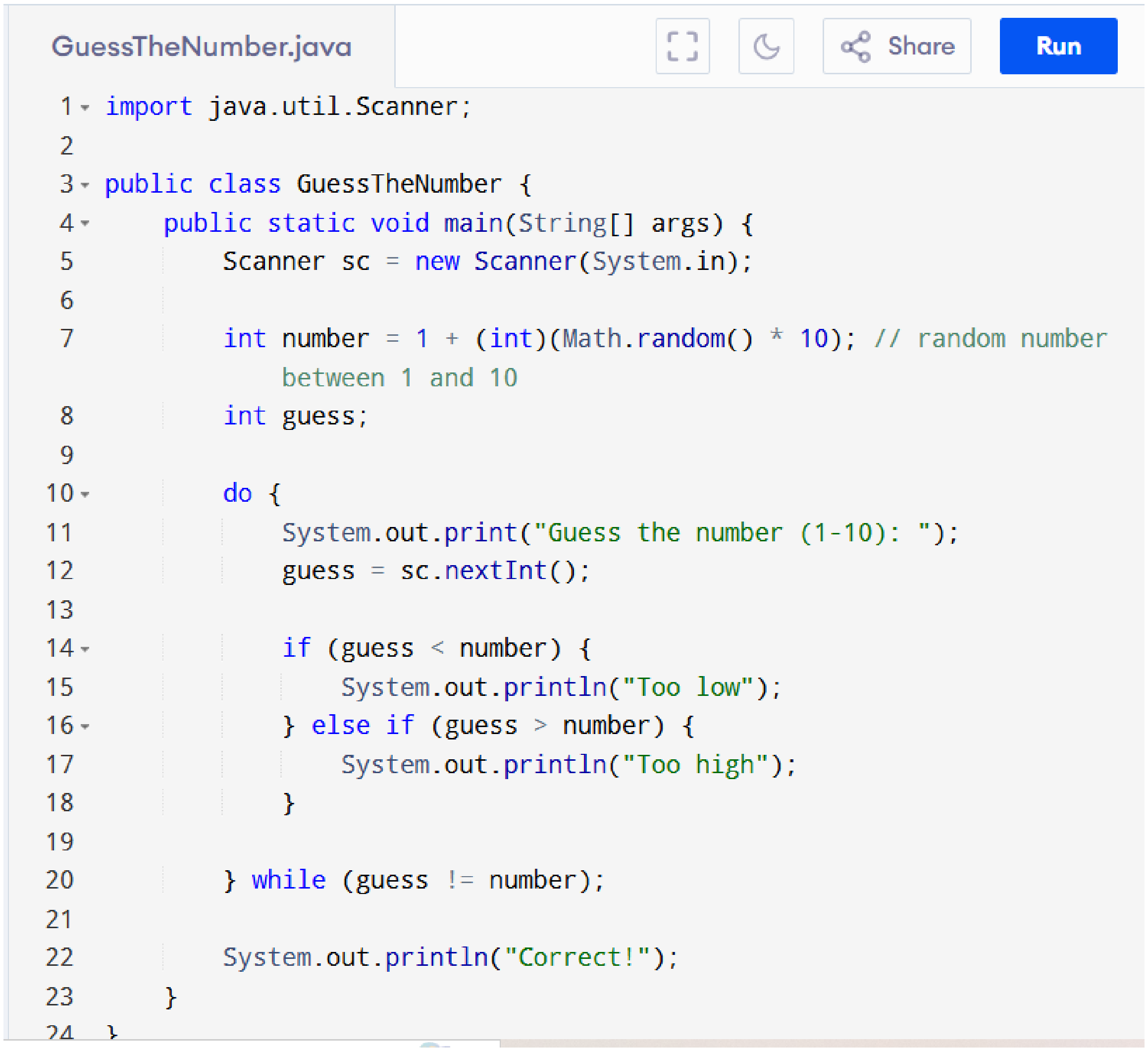
Algorithm:

```
Generate random number between 1 and 10

Loop:

  Ask user for a guess

  If guess is less than number, print 'Too low'

  If guess is greater than number, print 'Too high'

  If guess equals number, print 'Correct' and break
```

Pseudocode:

```
target = random number 1-10

repeat:

    input guess

    if guess < target: print 'Too low'

    elif guess > target: print 'Too high'

    else: print 'Correct'; break
```

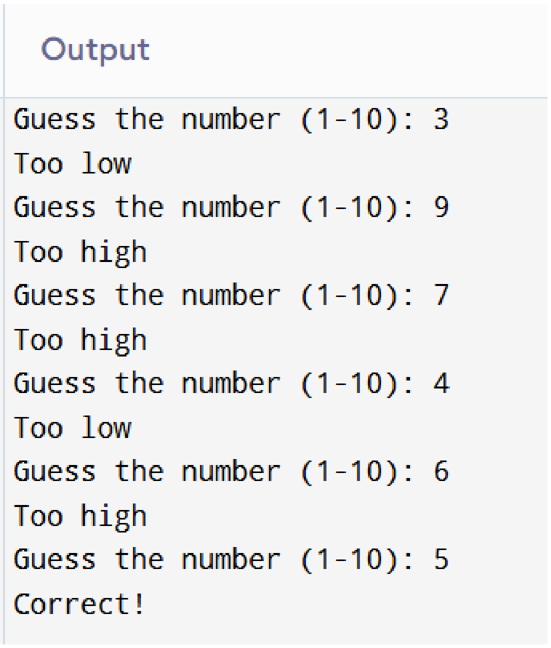
Java Code:



```java
import java.util.Scanner;

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

        int number = 1 + (int)(Math.random() * 10); // random number
            between 1 and 10
        int guess;

        do {
            System.out.print("Guess the number (1-10): ");
            guess = sc.nextInt();

            if (guess < number) {
                System.out.println("Too low");
            } else if (guess > number) {
                System.out.println("Too high");
            }

        } while (guess != number);

        System.out.println("Correct!");
    }
}
```

Test Case: Input: 5, 8, 7 (assuming number is 7)

Output:



```
Guess the number (1-10): 3
Too low
Guess the number (1-10): 9
Too high
Guess the number (1-10): 7
Too high
Guess the number (1-10): 4
Too low
Guess the number (1-10): 6
Too high
Guess the number (1-10): 5
Correct!
```

Problem 1.4: Infinite Loop Debugging

Fix the infinite loop in the given code snippet.

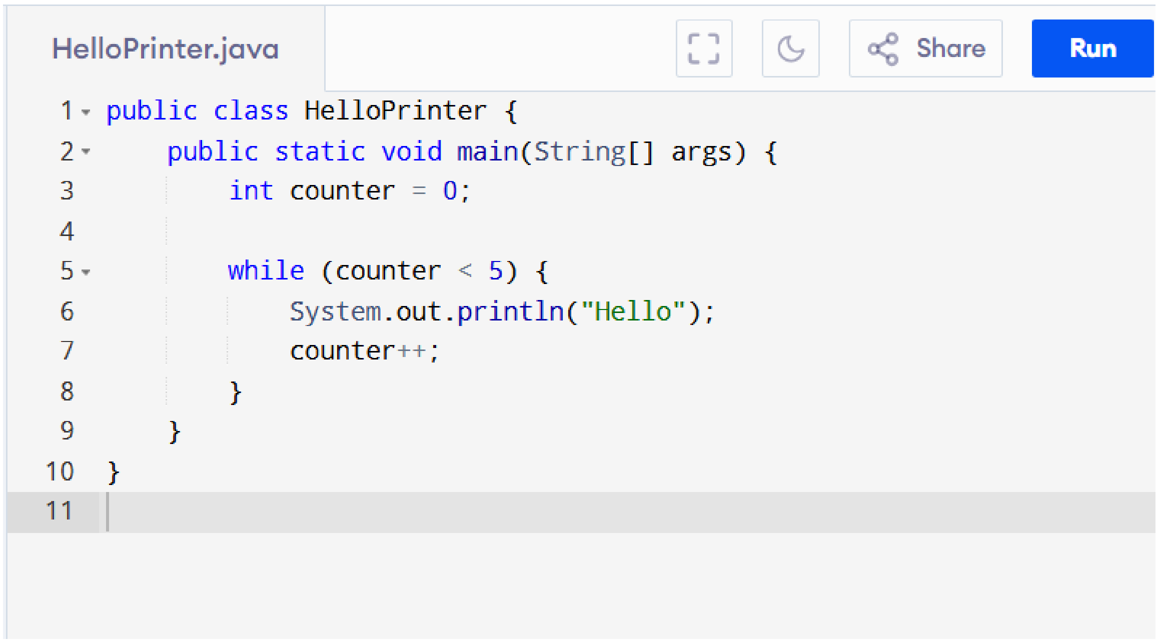
Algorithm:

```
Initialize counter to 0
While counter < 5:
  Print 'Hello'
  Increment counter
```

Pseudocode:

```
counter = 0
while counter < 5:
    print 'Hello'
    counter += 1
```

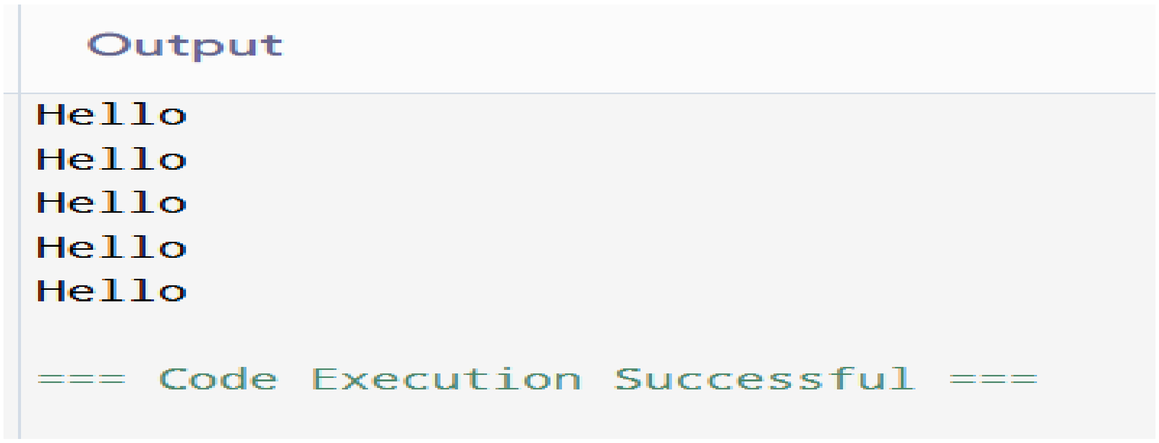
Java Code:



```java
public class HelloPrinter {
    public static void main(String[] args) {
        int counter = 0;

        while (counter < 5) {
            System.out.println("Hello");
            counter++;
        }
    }
}
```

Test Case: N/A

Output:
Hello printed 5 times



```
Output
Hello
Hello
Hello
Hello
Hello

=== Code Execution Successful ===
```

Problem 2.1: Even Numbers

Print even numbers from 2 to 20 using a for loop.

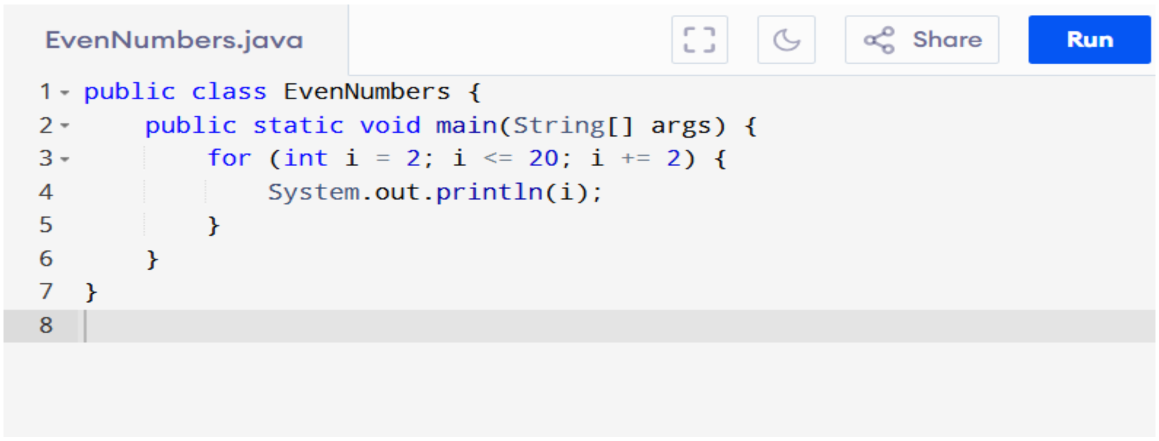
Algorithm:

- Start from 2 and go up to 20
- In steps of 2, print each number

Pseudocode:

```
for i from 2 to 20 step 2:
    print i
```

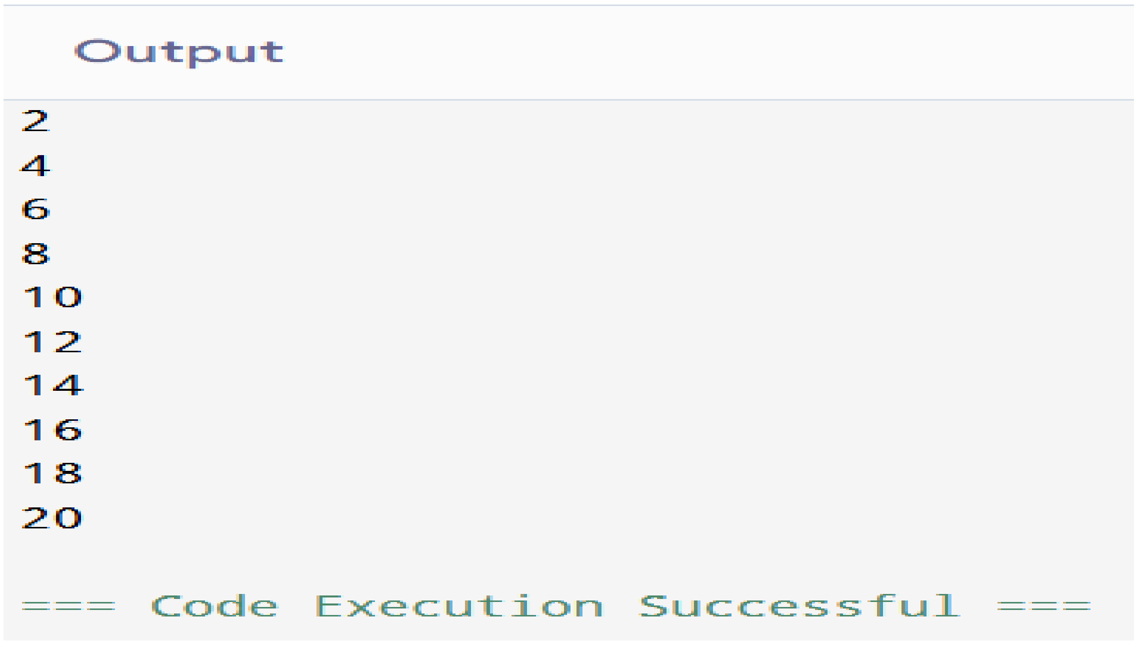
Java Code:



```java
public class EvenNumbers {
    public static void main(String[] args) {
        for (int i = 2; i <= 20; i += 2) {
            System.out.println(i);
        }
    }
}
```

Test Case: N/A

Output:
2 4 6 8 10 12 14 16 18 20



```
2
4
6
8
10
12
14
16
18
20

=== Code Execution Successful ===
```

Problem 2.2: Factorial Calculator

Calculate n! for user input n. Handle edge case when n == 0.

Algorithm:

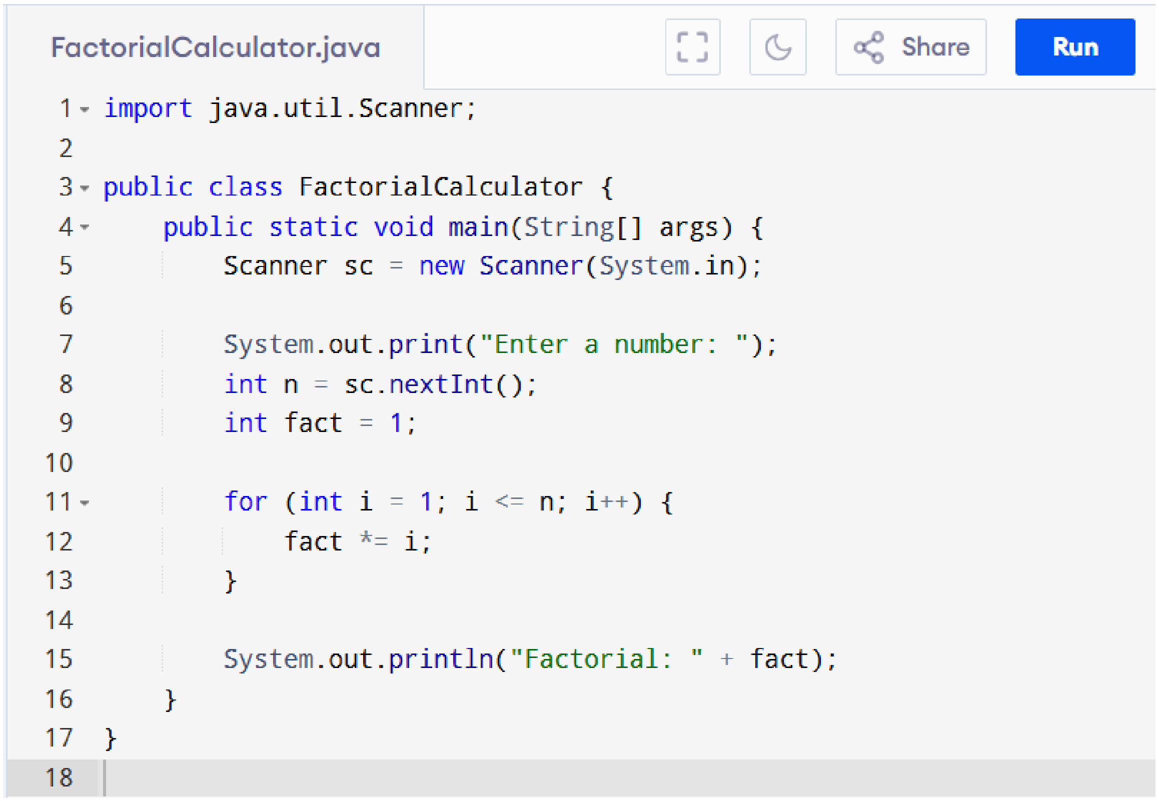
- Input n
- If n == 0, return 1
- Else, initialize factorial = 1
- Loop i from 1 to n, multiply factorial *= i

Pseudocode:

- input n
- if n == 0: print 1
- else:

- factorial = 1
- for i from 1 to n:
-   factorial *= i
- print factorial

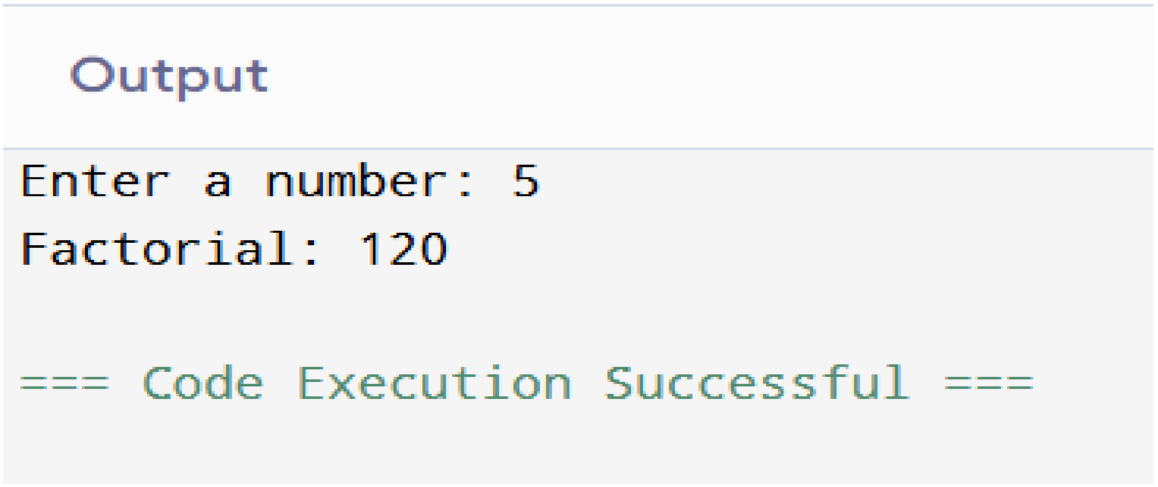
Java Code:



```java
import java.util.Scanner;

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

        System.out.print("Enter a number: ");
        int n = sc.nextInt();
        int fact = 1;

        for (int i = 1; i <= n; i++) {
            fact *= i;
        }

        System.out.println("Factorial: " + fact);
    }
}

```

Test Case: Input: 5

Output:
Factorial: 120



```
Enter a number: 5
Factorial: 120


=== Code Execution Successful ===
```

Problem 2.3: Count 'a' in String

Ask for a string input. Count how many times ‘a’ or ‘A’ appears.

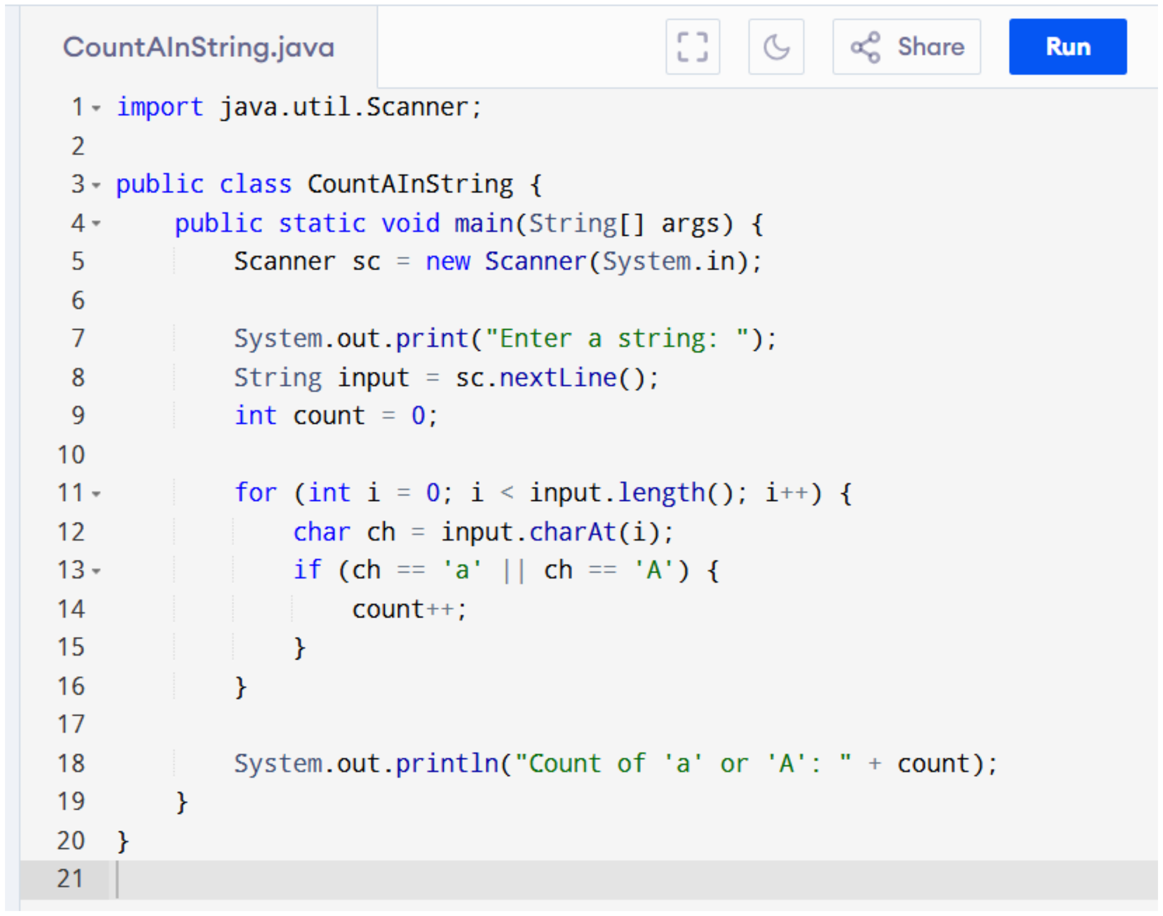
Algorithm:

- Input string
- Initialize count = 0
- Loop through each character of string
- If char == 'a' or 'A', increment count
- Print count

Pseudocode:

```
input str
count = 0
for each character in str:
    if char == 'a' or char == 'A':
        count += 1
print count
```

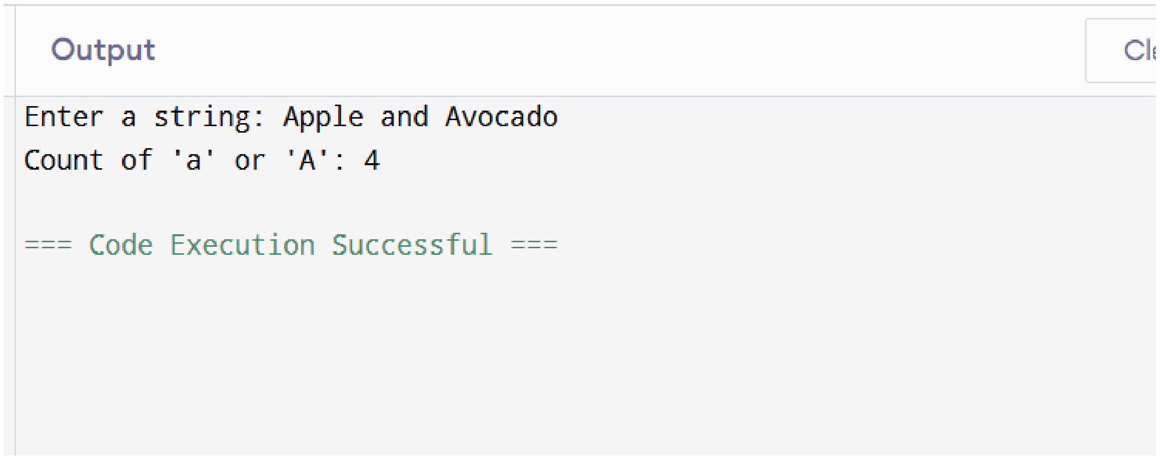
Java Code:



```java
import java.util.Scanner;

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

        System.out.print("Enter a string: ");
        String input = sc.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("Count of 'a' or 'A': " + count);
    }
}
```

Test Case: Input: 'Apple and Avocado'

Output:
Count of 'a' or 'A': 4



```
Enter a string: Apple and Avocado
Count of 'a' or 'A': 4

=== Code Execution Successful ===
```

Problem 2.4: Simple Star Pattern

Print: ***** using one for loop.

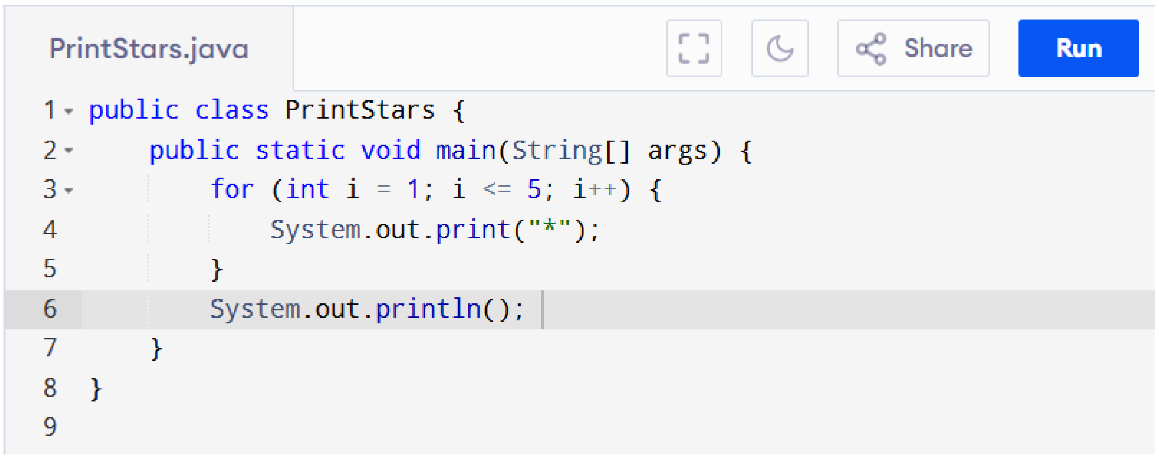
Algorithm:

- Loop from 1 to 5
- In each iteration, print '*' without newline

Pseudocode:

```
for i from 1 to 5:
    print '*' (no newline)
```

Java Code:

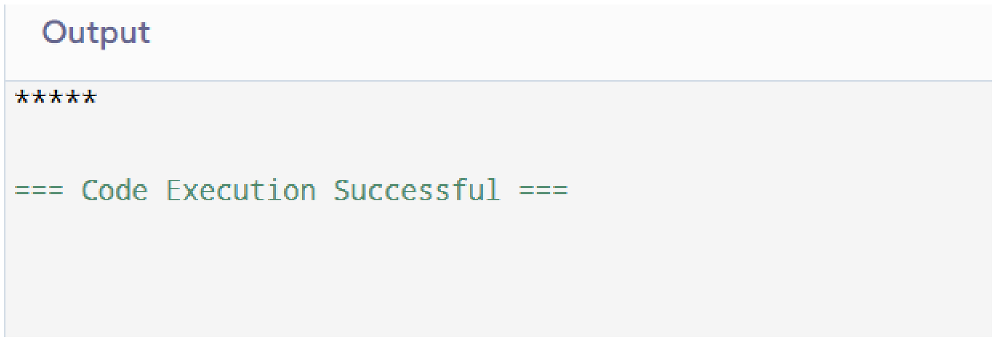


```java
public class PrintStars {
    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++) {
            System.out.print("*");
        }
        System.out.println();
    }
}
```

Test Case: N/A

Output:

*****


Output

```
*****

=== Code Execution Successful ===
```

Problem 3.1: Prime Checker

Problem Statement:

Check if a number is prime using a loop and break.

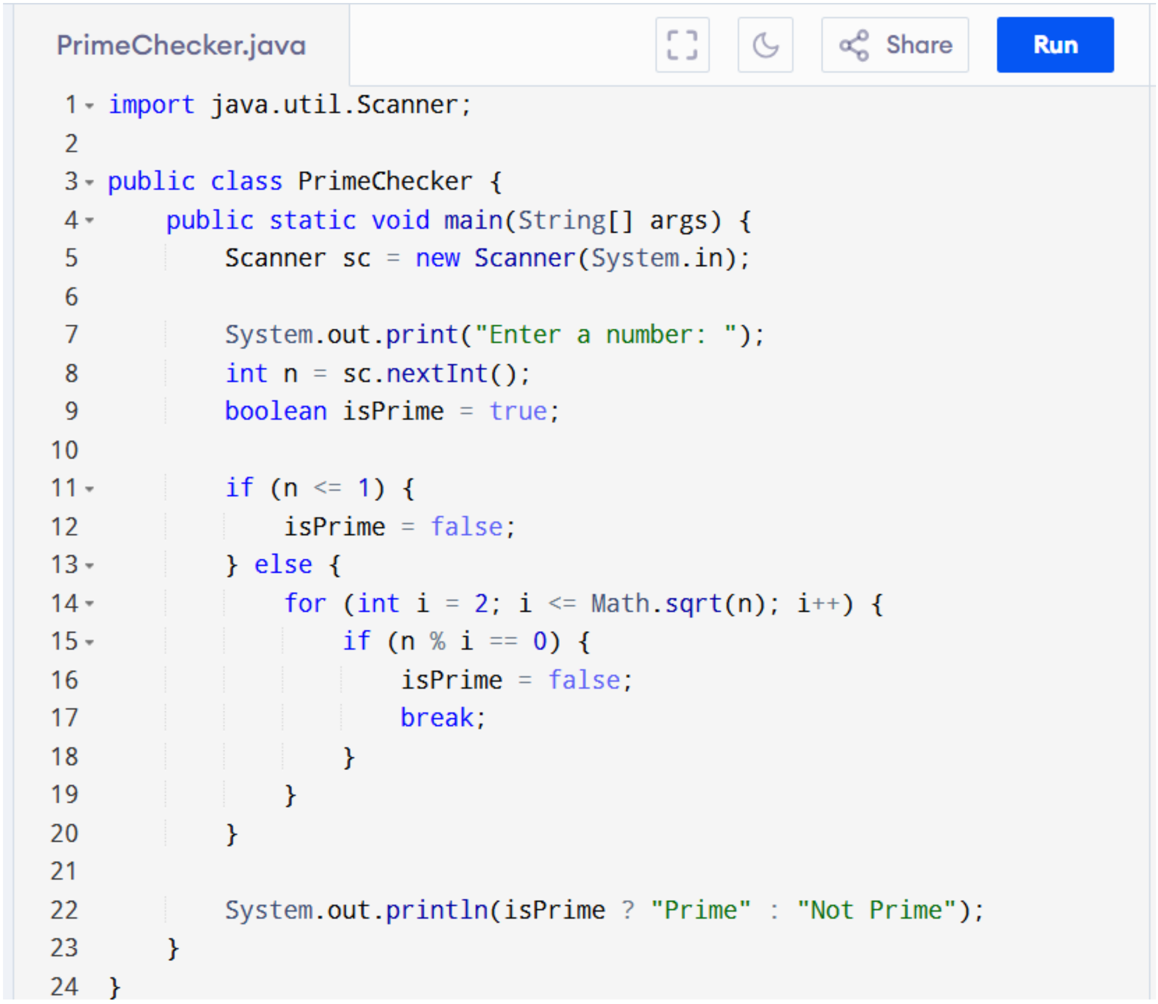
Algorithm:

- Input number n
- If n <= 1, not prime
- Loop from 2 to sqrt(n):
-   If n % i == 0, it's not prime (break)
- If loop completes, it's prime

Pseudocode:

```
input n
if n <= 1: not prime
for i from 2 to sqrt(n):
    if n % i == 0: not prime, break
else: prime
```

Java Code:

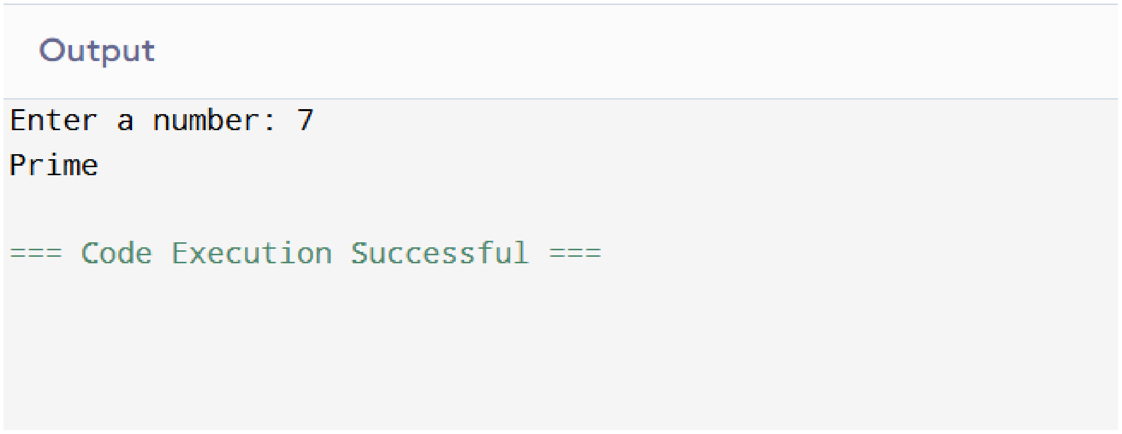


```java
import java.util.Scanner;

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

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

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

        System.out.println(isPrime ? "Prime" : "Not Prime");
    }
}
```

Test Case: Input: 7

Output:

Prime



```
Enter a number: 7
Prime

=== Code Execution Successful ===
```

Problem 3.2: Skip Negatives

Input 5 numbers. Use continue to skip negative ones and sum the rest.

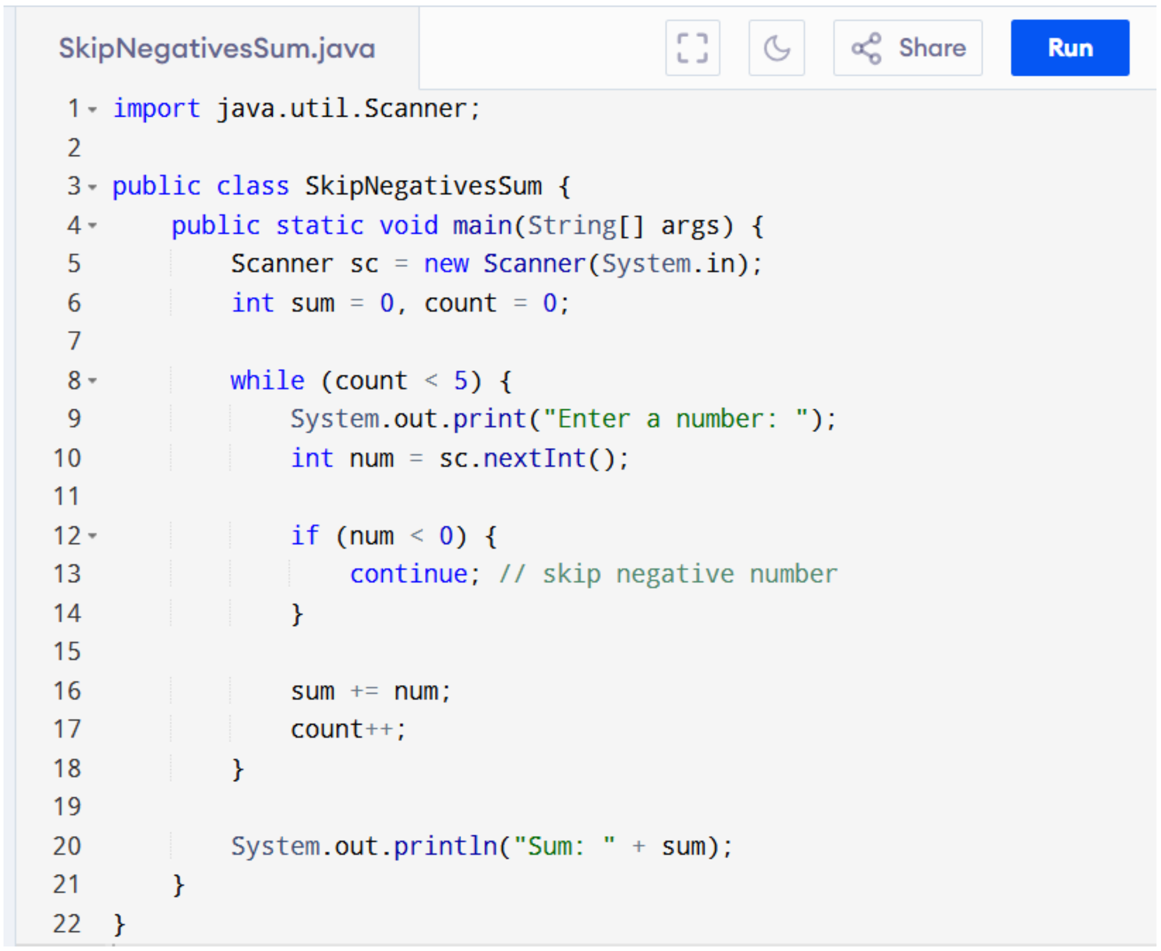
Algorithm:

- Initialize sum = 0, count = 0
- While count < 5:
    - Input number
    - If number < 0, continue
    - Add to sum, increment count

Pseudocode:

```
sum = 0, count = 0
while count < 5:
  input num
  if num < 0: continue
  sum += num
  count += 1
```

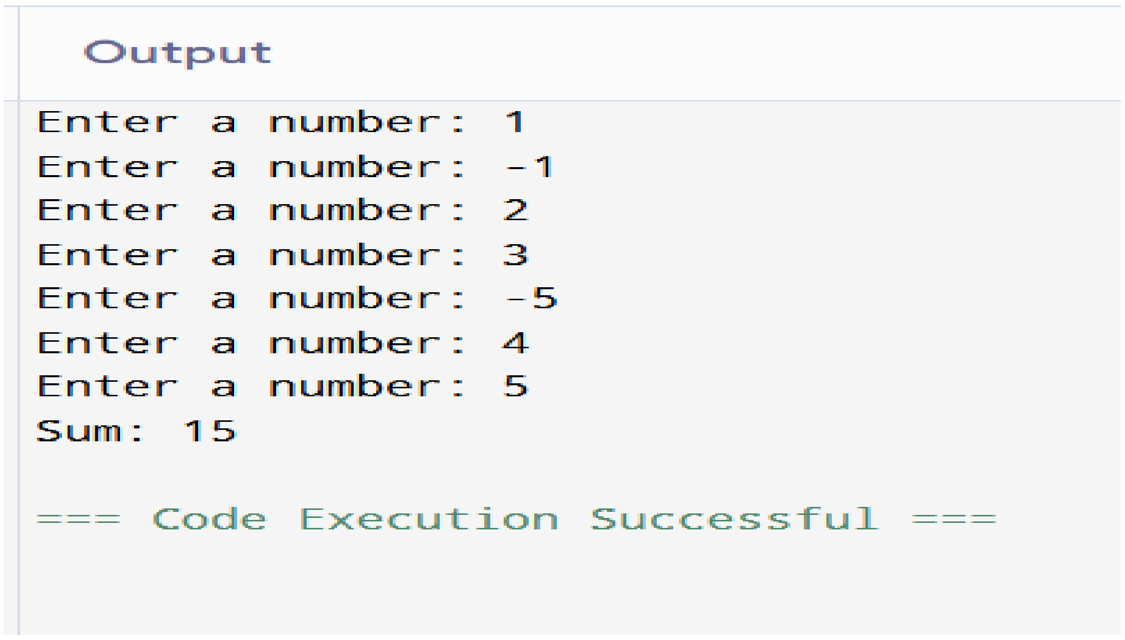
Java Code:



```java
import java.util.Scanner;

public class SkipNegativesSum {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int sum = 0, count = 0;

        while (count < 5) {
            System.out.print("Enter a number: ");
            int num = sc.nextInt();

            if (num < 0) {
                continue; // skip negative number
            }

            sum += num;
            count++;
        }

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

Test Case: Input: 1, -1, 2, 3, -5, 4, 5

Output:
Sum: 15



```
Enter a number: 1
Enter a number: -1
Enter a number: 2
Enter a number: 3
Enter a number: -5
Enter a number: 4
Enter a number: 5
Sum: 15

=== Code Execution Successful ===
```

Problem 3.3: Rectangle Pattern

Input rows and cols, print a rectangle of *.

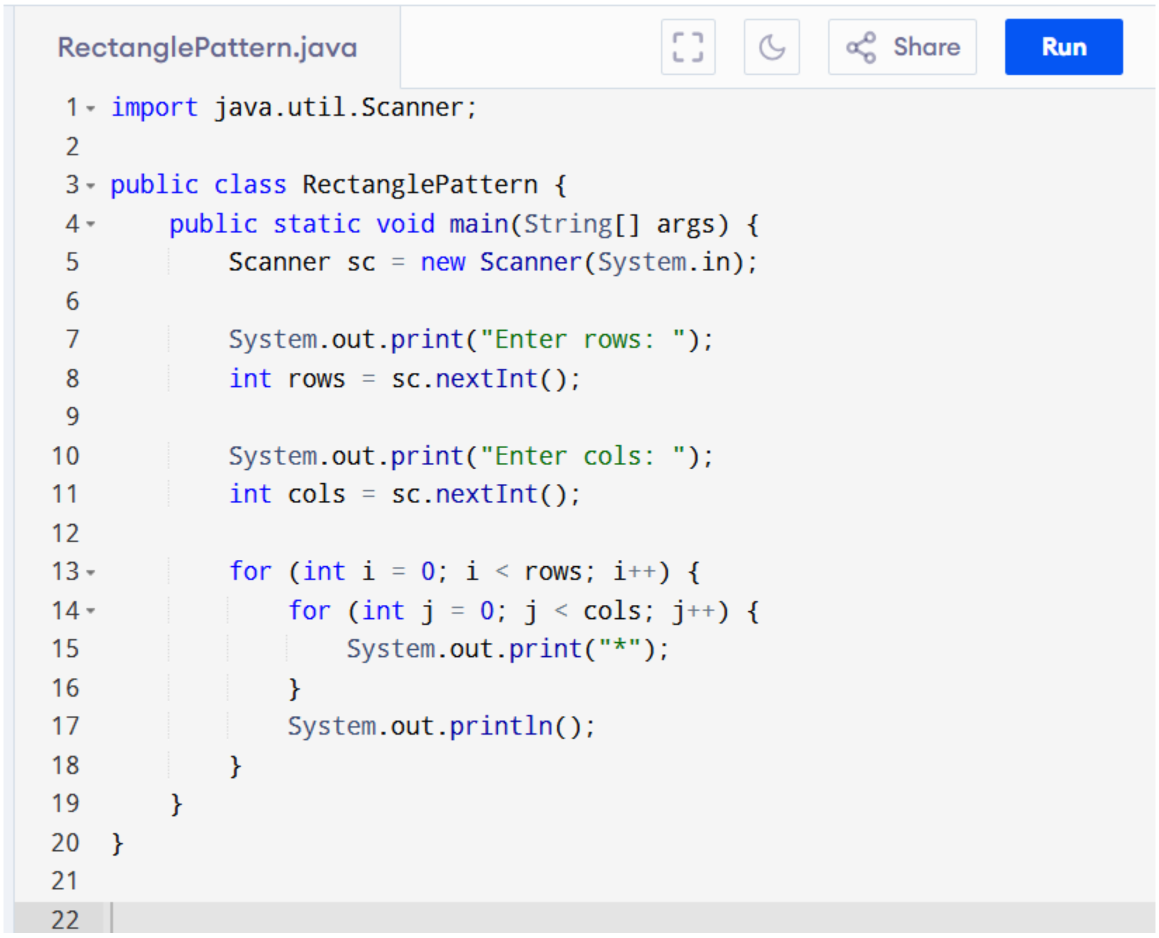
Algorithm:

- Input rows and columns
- Loop through rows
- Loop through columns
-   Print '*' without newline
- Print newline after each row

Pseudocode:

```
input rows, cols
for i in 1 to rows:
    for j in 1 to cols:
        print '*' (no newline)
    print newline
```

Java Code:

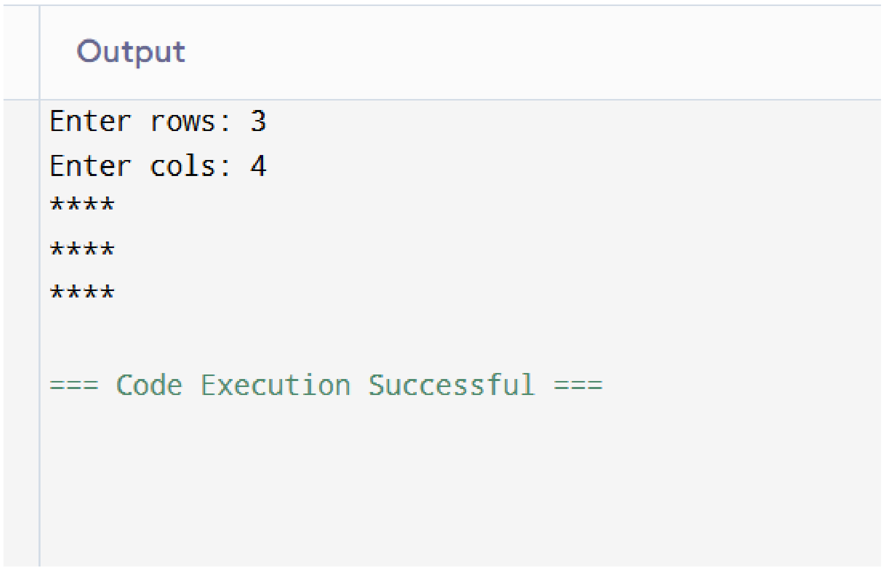


```java
import java.util.Scanner;

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

        System.out.print("Enter rows: ");
        int rows = sc.nextInt();

        System.out.print("Enter cols: ");
        int cols = sc.nextInt();

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}


```

Test Case: Input: 3 4

Output:

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



Output

```
Enter rows: 3
Enter cols: 4
****
****
****


=== Code Execution Successful ===
```

Problem 3.4: Triangle Pattern

Input height. Print right-angled triangle with *.

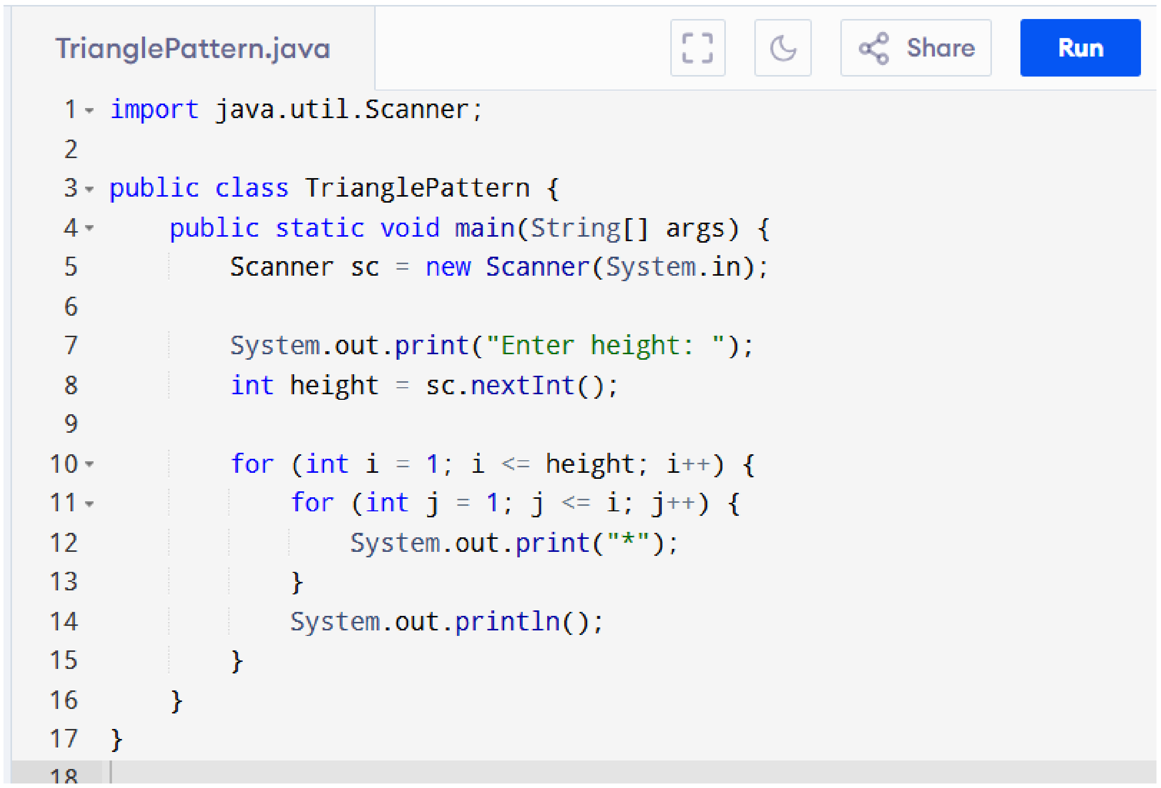
Algorithm:

- Input height
- Loop i from 1 to height:
- Print i stars

Pseudocode:

- input height
- for i from 1 to height:
-     print '*' i times

Java Code:



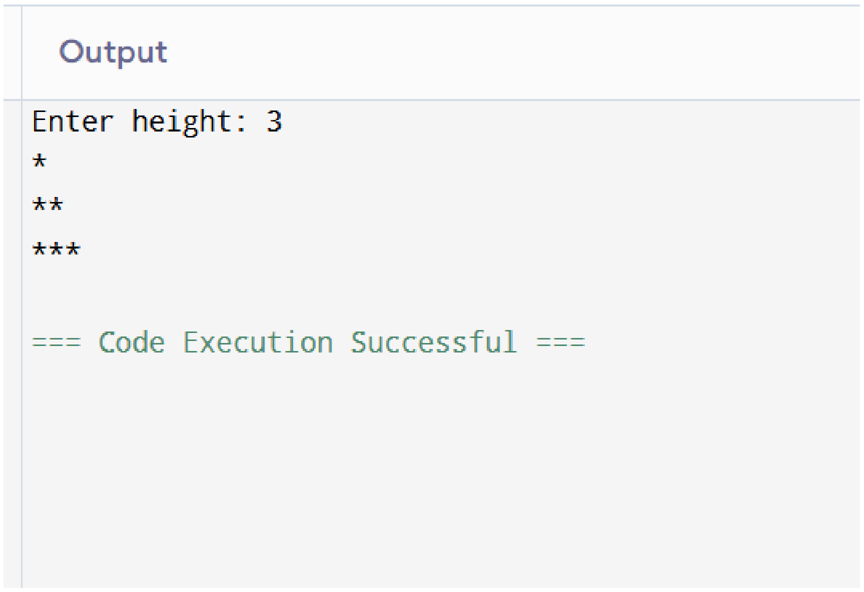
TrianglePattern.java

```java
import java.util.Scanner;

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

        System.out.print("Enter height: ");
        int height = sc.nextInt();

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

```

Test Case: Input: 3

Output:

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


Output

```
Enter height: 3
*
**
***


=== Code Execution Successful ===
```

Problem 3.5: Pyramid Pattern Challenge

Input height. Print centered pyramid.

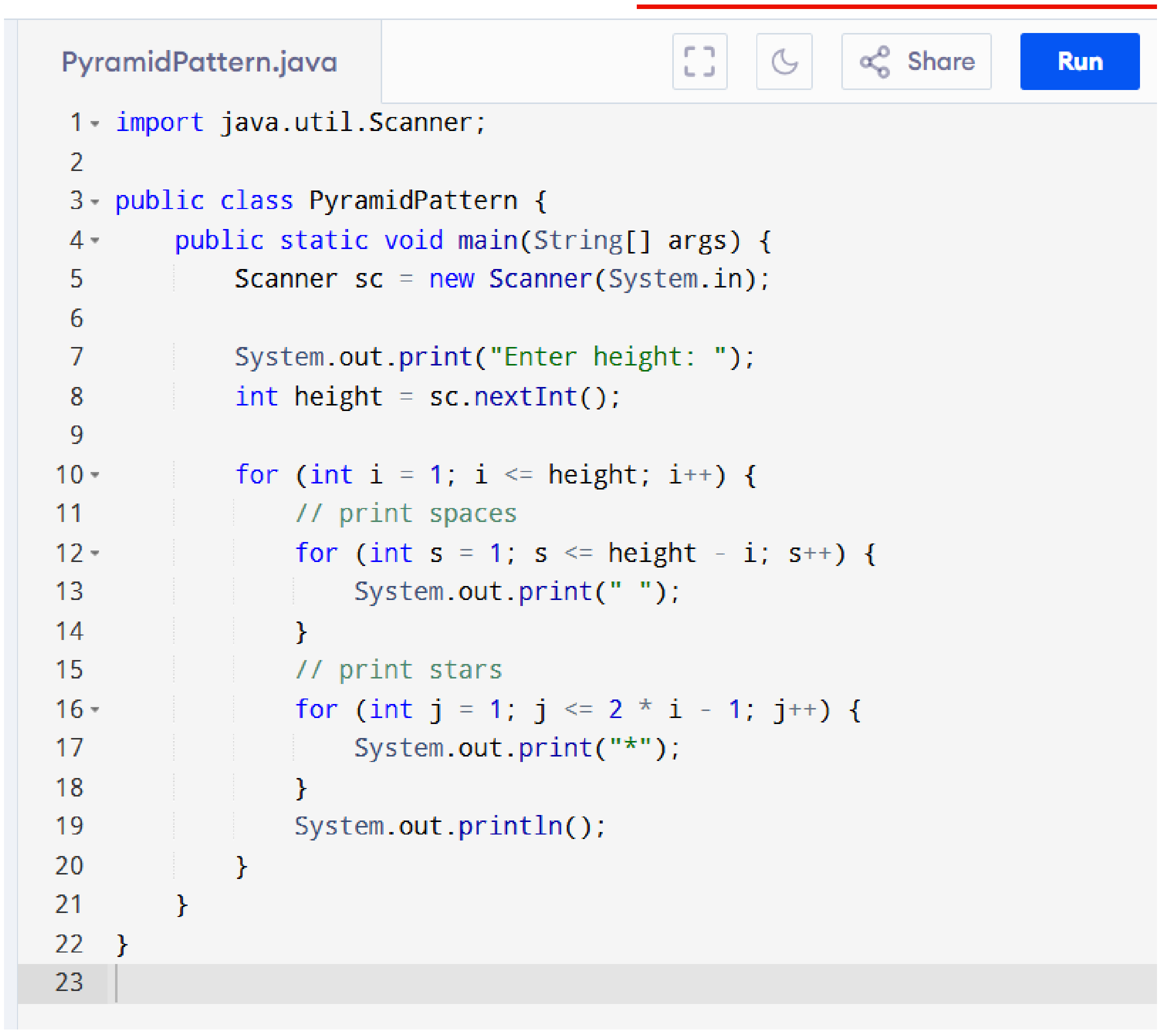
Algorithm:

- Input height
- For i from 1 to height:
-   Print (height - i) spaces
-   Print (2*i - 1) stars

Pseudocode:

```
input height
for i from 1 to height:
    print spaces (height - i)
    print stars (2*i - 1)
```

Java Code:

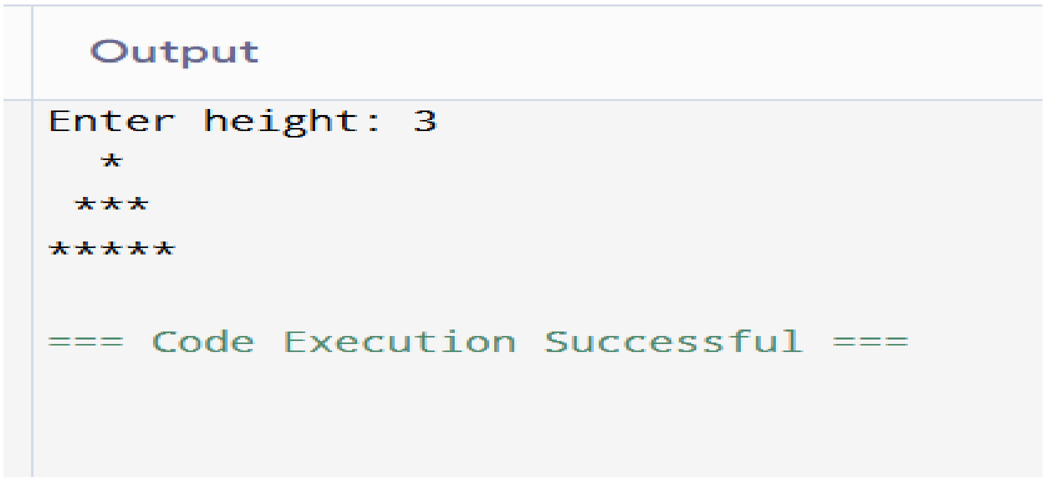


```java
import java.util.Scanner;

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

        System.out.print("Enter height: ");
        int height = sc.nextInt();

        for (int i = 1; i <= height; i++) {
            // print spaces
            for (int s = 1; s <= height - i; s++) {
                System.out.print(" ");
            }
            // print stars
            for (int j = 1; j <= 2 * i - 1; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

Test Case: Input: 3

Output:

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



Output

```
Enter height: 3
  *
 ***
*****

=== Code Execution Successful ===
```