CHINMAY GAIKWAD\_KH

CDAC MUMBAI

# Lab Assignment

**SECTION 1: Error-Driven Learning Assignment: Loop Errors**

### Instructions:

Analyze each code snippet for errors or unexpected behavior. For each snippet, determine:

#### Why does the error or unexpected behavior occur?

1. How can the code be corrected to achieve the intended behavior?

**Snippet 1:**

public class InfiniteForLoop {

public static void main(String[] args) { for (int i = 0; i < 10; i--) {

System.out.println(i);

}

}

}

// Error to investigate: Why does this loop run infinitely? How should the loop control variable be adjusted?

* **The provided code results in an infinite loop because the loop control variable i is being decremented (i--)**
* **To avoid an infinite loop, the loop control variable should be incremented (i++) instead of decremented. This will eventually cause i to reach a value where i < 10 is false, and the loop will terminate.**

**Solution:**

public class InfiniteForLoop {

public static void main(String[] args) {

for (int i = 0; i < 10; i++) { // Use i++ to increment i

System.out.println(i);

}

}

}

## Snippet 2:

public class IncorrectWhileCondition { public static void main(String[] args) {

int count = 5; while (count = 0) {

System.out.println(count); count--;

}

}

}

// Error to investigate: Why does the loop not execute as expected? What is the issue with the condition in the

`while` loop?

**- Assignment vs. Comparison: The expression count = 0 is an assignment operation, which assigns the value 0 to count. In Java, the result of an assignment operation is the value assigned, which in this case is 0.**

**- Condition in While Loop: The while loop expects a boolean condition, but because count = 0 is an assignment, it evaluates to 0, which is treated as false in Java. Consequently, the loop does not execute even once**

**.**

**- Resulting Behaviour: Since the condition evaluates to false, the loop body is skipped entirely, and no output is produced.**

**Solution:**

public class IncorrectWhileCondition {

public static void main(String[] args) {

int count = 5;

while (count != 0) { // Corrected condition with comparison operator

System.out.println(count);

count--;

}

}

}

## Snippet 3:

public class DoWhileIncorrectCondition { public static void main(String[] args) {

int num = 0; do {

System.out.println(num); num++;

} while (num > 0);

}

}

// Error to investigate: Why does the loop only execute once? What is wrong with the loop condition in the `do- while` loop?

- **The original loop only appears to execute once due to the specific nature of the do-while loop and its condition, but actually, it would continue indefinitely. The condition num > 0 allows the loop to continue indefinitely once num becomes positive. To stop the loop after one iteration, you can change the condition to num <= 0.**

**Solution:**

public class DoWhileIncorrectCondition {

public static void main(String[] args) {

int num = 0;

do {

System.out.println(num);

num++;

} while (num <= 0); // Change condition to ensure it only executes once

}

}

## Snippet 4:

public class OffByOneErrorForLoop { public static void main(String[] args) {

for (int i = 1; i <= 10; i++) { System.out.println(i);

}

// Expected: 10 iterations with numbers 1 to 10

// Actual: Prints numbers 1 to 10, but the task expected only 1 to 9

}

}

// Error to investigate: What is the issue with the loop boundaries? How should the loop be adjusted to meet the expected output?

* **The original loop included the number 10 because the condition was i <= 10. To correct this off-by-one error and achieve the expected output of printing numbers 1 through 9, you should adjust the loop condition to i < 10. This change ensures that the loop excludes 10 and stops after 9 iterations.**

**Solution:**

* A close up of a logo

  Description automatically generatedpublic class OffByOneErrorForLoop { public static void main(String[] args) {
* for (int i = 1; i <= 10; i++) { System.out.println(i);
* }

## Snippet 5:

public class WrongInitializationForLoop { public static void main(String[] args) {

for (int i = 10; i >= 0; i++) { System.out.println(i);

}

}

}

// Error to investigate: Why does this loop not print numbers in the expected order? What is the problem with the initialization and update statements in the `for` loop?

* **The original loop does not print numbers in descending order because the update statement incorrectly increments i instead of decrementing it. This leads to an infinite loop that constantly prints the starting value of 10. The solution is to use the decrement operator i-- in the update statement, which correctly reduces i on each iteration and results in the expected descending sequence.**

**Solution:**

public class WrongInitializationForLoop { public static void main(String[] args) {

for (int i = 10; i >= 0; i--) { System.out.println(i);

}

}

}

## Snippet 6:

public class MisplacedForLoopBody { public static void main(String[] args) {

for (int i = 0; i < 5; i++) System.out.println(i); System.out.println("Done");

}

}

// Error to investigate: Why does "Done" print only once, outside the loop? How should the loop body be enclosed to include all statements within the loop?

* **The "Done" message prints only once outside the loop because, without curly braces, only the first statement after the loop header is considered part of the loop. To include multiple statements within the loop body, enclose them in curly braces {}. This ensures all enclosed statements are executed with each iteration of the loop.**

**Solution:**

public class MisplacedForLoopBody { public static void main(String[] args) {

for (int i = 0; i < 5; i++) { System.out.println(i); System.out.println("Done");

}

}

}

## Snippet 7:

public class UninitializedWhileLoop { public static void main(String[] args) {

int count;

while (count < 10) { System.out.println(count); count++;

}

}

}

// Error to investigate: Why does this code produce a compilation error? What needs to be done to initialize the loop variable properly?

- **The original code produces a compilation error because the variable count was declared but not initialized before being used in the while loop. In Java, local variables must be initialized before they are used. The solution is to initialize count (e.g., to 0) before the loop starts, which ensures the loop functions as intended.**

**Solution:**

public class UninitializedWhileLoop { public static void main(String[] args) {

int count;

int i = 0;

while (count < 10)

{ System.out.println(count);

count++;

}

}

}

## Snippet 8:

public class OffByOneDoWhileLoop { public static void main(String[] args) {

int num = 1; do {

System.out.println(num); num--;

} while (num > 0);

}

}

// Error to investigate: Why does this loop print unexpected numbers? What adjustments are needed to print the numbers from 1 to 5?

* **The original loop only prints the number 1 and then exits because the num-- operation makes num 0, causing the loop condition num > 0 to fail. To print numbers from 1 to 5, you can either increment num and use the condition num <= 5 or decrement num if starting from a higher number**.

Solution:

public class OffByOneDoWhileLoop { public static void main(String[] args) {

int num = 1; do {

System.out.println(num); num++;

} while (num <= 5);

}

}

## Snippet 9:

public class InfiniteForLoopUpdate { public static void main(String[] args) {

for (int i = 0; i < 5; i += 2) { System.out.println(i);

}

}

}

// Error to investigate: Why does the loop print unexpected results or run infinitely? How should the loop update expression be corrected?

* **The original loop does not run infinitely, but it prints only even numbers (0, 2, 4) due to the i += 2 update expression. To print numbers from 0 to 4 sequentially, change the update expression to i++, which increments i by 1 on each iteration.**

**Solution:**

public class InfiniteForLoopUpdate { public static void main(String[] args) {

for (int i = 0; i < 5; i ++) { System.out.println(i);

}

}

}

## Snippet 10:

public class IncorrectWhileLoopControl { public static void main(String[] args) {

int num = 10; while (num = 10) {

System.out.println(num); num--;

}

}

}

// Error to investigate: Why does the loop execute indefinitely? What is wrong with the loop condition?

* **The original loop executes indefinitely because the loop condition num = 10 is an assignment, not a comparison. This assignment continuously sets num to 10, making the loop run forever. To fix this, you should use the equality operator == to compare num to 10. If the intention is to decrement num in each iteration, and stop when it is no longer positive, use a loop condition like while (num > 0).**

**Solution :**

public class IncorrectWhileLoopControl { public static void main(String[] args) {

int num = 10; while (num > 0) {

System.out.println(num); num--;

}

}

}

## Snippet 11:

public class IncorrectLoopUpdate { public static void main(String[] args) {

int i = 0; while (i < 5) {

System.out.println(i);

i += 2; // Error: This may cause unexpected results in output

}

}

}

// Error to investigate: What will be the output of this loop? How should the loop variable be updated to achieve the desired result?

* **The original loop prints only the even numbers 0, 2, and 4 because the loop variable i is incremented by 2 on each iteration. To print all numbers from 0 to 4, update i by 1 instead (i++), so the loop iterates over every integer in the desired range.**

**Solution:**

public class IncorrectLoopUpdate { public static void main(String[] args) {

int i = 0; while (i < 5) {

A close up of a logo

Description automatically generatedSystem.out.println(i);

i ++; // Error: This may cause unexpected results in output

}

}

}

## Snippet 12:

public class LoopVariableScope {

public static void main(String[] args) { for (int i = 0; i < 5; i++) {

int x = i \* 2;

}

System.out.println(x); // Error: 'x' is not accessible here

}

}

// Error to investigate: Why does the variable 'x' cause a compilation error? How does scope

* **The variable x caused a compilation error because it was declared inside the for loop, making it inaccessible outside the loop due to its limited scope. To correct this, declare x outside the loop, giving it a broader scope so that it can be accessed both inside and outside the loop.**

**Solution :**

public class LoopVariableScope {

public static void main(String[] args) {

int x = 0; // Declare 'x' outside the loop

for (int i = 0; i < 5; i++) {

x = i \* 2;

}

System.out.println(x); // Now 'x' is accessible here

}

}

# SECTION 2: Guess the Output

### Instructions:

#### **Perform a Dry Run:** Carefully trace the execution of each code snippet manually to determine the output.

1. **Write Down Your Observations:** Document each step of your dry run, including the values of variables at each stage of execution.
2. **Guess the Output:** Based on your dry run, provide the expected output of the code.
3. **Submit Your Assignment:** Provide your dry run steps along with the guessed output for each code snippet.

**Snippet 1:**

public class NestedLoopOutput {

public static void main(String[] args) { for (int i = 1; i <= 3; i++) {

for (int j = 1; j <= 2; j++) { System.out.print(i + " " + j + " ");

}

System.out.println();

}

}

}

// Guess the output of this nested loop.

**Dry Run:**

* **Outer Loop (i=1):**

Inner Loop (j=1): prints 1 1

Inner Loop (j=2): prints 1 2

New Line

* **Outer Loop (i=2):**

Inner Loop (j=1): prints 2 1

Inner Loop (j=2): prints 2 2

New Line

* **Outer Loop (i=3):**
  + Inner Loop (j=1): prints 3 1
  + Inner Loop (j=2): prints 3 2
  + New Line

**Output:**

**1 1 1 2**

**2 1 2 2**

**3 1 3 2**

|  |
| --- |
|  |
|  |  |

## Snippet 2:



public class DecrementingLoop {

public static void main(String[] args) { int total = 0;

for (int i = 5; i > 0; i--) { total += i;

if (i == 3) continue; total -= 1;

}

System.out.println(total);

}

}

// Guess the output of this loop.

**Dry Run:**

i=5: total += 5 (total = 5), total -= 1 (total = 4)

i=4: total += 4 (total = 8), total -= 1 (total = 7)

i=3: total += 3 (total = 10), continue skips the decrement

**i=2:** total += 2 (total = 12), total -= 1 (total = 11)

**i=1:** total += 1 (total = 12), total -= 1 (total = 11)

**Output:**

11

## Snippet 3:

public class WhileLoopBreak {

public static void main(String[] args) { int count = 0;

while (count < 5) { System.out.print(count + " "); count++;

if (count == 3) break;

}

System.out.println(count);

}

}

// Guess the output of this while loop.

**Dry Run:**

**count=0:** prints 0

**count=1:** prints 1

**count=2:** prints 2

**count=3:** breaks the loop

**Output:**

0 1 2 3

## Snippet 4:

public class DoWhileLoop {

public static void main(String[] args) { int i = 1;

do {

System.out.print(i + " "); i++;

} while (i < 5); System.out.println(i);

}

}

// Guess the output of this do-while loop

**Dry Run:**

**i=1:** prints 1

**i=2:** prints 2

**i=3:** prints 3

**i=4:** prints 4

**i=5:** loop exits and prints 5

**Output:**

1 2 3 4 5

## Snippet 5:

public class ConditionalLoopOutput { public static void main(String[] args) {

int num = 1;

for (int i = 1; i <= 4; i++) { if (i % 2 == 0) {

num += i;

} else {

A close up of a logo

Description automatically generatednum -= i;

}

}

System.out.println(num);

}

}

// Guess the output of this loop.

**Dry Run:**

**i=1:** num -= 1 (num = 0)

**i=2:** num += 2 (num = 2)

**i=3:** num -= 3 (num = -1)

**i=4:** num += 4 (num = 3)

**Output:**

3

## Snippet 6:

public class IncrementDecrement { public static void main(String[] args) {

int x = 5;

int y = ++x - x-- + --x + x++; System.out.println(y);

}

}

// Guess the output of this code snippet.

**Dry Run:**

**Initial x=5**

++x (x=6, y=6)

x-- (x=6, y=6-6=0, x=5 after decrement)

--x (x=4, y=0+4=4)

x++ (x=4, y=4+4=8, x=5 after increment)

* **The output of the code will be: 10**

## Snippet 7:

public class NestedIncrement {

public static void main(String[] args) { int a = 10;

int b = 5;

int result = ++a \* b a + b++;

System.out.println(result);

}

}

// Guess the output of this code snippet.

**Dry Run:**

**Initial a=10, b=5**

++a (a=11)

result = 11 \* 5 (result=55)

a is 11

b++ (b=6 after increment)

result = 55 - 11 + 5 (result=49)

**Output:**

49

## Snippet 8:

public class LoopIncrement {

public static void main(String[] args) { int count = 0;

for (int i = 0; i < 4; i++) { count += i++ - ++i;

}

System.out.println(count);

}

}

// Guess the output of this code snippet.

**Dry Run:**

**i=0, count=0:** i++ (i=1), ++i (i=2), count += 0 - 2 (count=-2)

**i=2, count=-2:** i++ (i=3), ++i (i=4), count += 2 - 4 (count=-4)

**i=4:** exits the loop

**Output:**

-4

# SECTION 3: Lamborghini Exercise:

### Instructions:



1. **Complete Each Program:** Write a Java program for each of the tasks listed below.

#### **Test Your Code:** Make sure your code runs correctly and produces the expected output.

1. **Submit Your Solutions:** Provide the complete code for each task along with sample output.

***Tasks:***

#### Write a program to calculate the sum of the first 50 natural numbers.

#### public class SumOfNaturalNumbers {

#### public static void main(String[] args) {

#### int sum = 0; // Variable to store the sum

#### int n = 50; // Number of natural numbers to sum

#### // Loop through numbers from 1 to 50

#### for (int i = 1; i <= n; i++) {

#### sum += i; // Add each number to the sum

#### }

#### // Print the result

#### System.out.println("The sum of the first " + n + " natural numbers is: " + sum);

#### }

#### }

**Output:**

**Sum of the first 50 natural numbers: 1275**

#### 

#### 

* 1. Write a program to compute the factorial of the number 10.

**Program code:**

public class factorialof10{

public static void main(String arge[]){

int number = 10;

long factorial =1;

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

factorial \*= i;

}

System.out.println("factorial of 10;" + factorial);

}

}

**Output:**

**factorial of 10;3628800**

#### 

#### 

#### Write a program to print all multiples of 7 between 1 and 100.

**Program code:**

public class multipleof7{

public static void main(String args[]){

System.out.println("Multiples of 7 between 1 and 100:");

int i = 7;

for( i=7; i<=100; i+=7){

System.out.println(i);

}

}

}

**Output:**

**Multiples of 7 between 1 and 100:**

**7**

**14**

**21**

**28**

**35**

**42**

**49**

**56**

**63**

**70**

**77**

**84**

**91**

**98**

#### 

#### 

* 1. Write a program to reverse the digits of the number 1234. The output should be 4321.

**Program code:**

public class reversenumbers{

public static void main(String args[]){

int number = 1234;

int reversed = 0;

while(number != 0){

int digit = number % 10;

reversed = reversed\* 10+ digit;

number /= 10;

}

System.out.println("reversed number :" +reversed);

}

}

**Output:**

**reversed number :4321**

#### 

#### 

#### Write a program to print the Fibonacci sequence up to the number 21.

**Program code:**

public class Fibonacci{

public static void main(String[] args) {

int a = 0, b = 1;

System.out.println("Fibonacci sequence up to 21:");

while (a <= 21) {

System.out.println(a);

int next = a + b;

a = b;

b = next;

}

}

}

**Output:**

**Fibonacci sequence up to 21:**

**0**

**1**

**1**

**2**

**3**

**5**

**8**

**13**

**21**

* 1. Write a program to find and print the first 5 prime numbers.

**Program code:**

public class First5Primes {

public static void main(String[] args) {

int count = 0;

int num = 2;

System.out.println("First 5 prime numbers:");

while (count < 5) {

if (isPrime(num)) {

System.out.println(num);

count++;

}

num++;

}

}

public static boolean isPrime(int number) {

if (number <= 1) return false;

for (int i = 2; i <= Math.sqrt(number); i++) {

if (number % i == 0) return false;

}

return true;

}

}

**Output:**

First 5 prime numbers:

2

3

5

7

11





#### Write a program to calculate the sum of the digits of the number 9876. The output should be 30 (9 + 8 + 7 + 6).

**Program code**:

public class SumOfDigits {

public static void main(String[] args) {

int number = 9876;

int sum = 0;

while (number != 0) {

sum += number % 10;

number /= 10;

}

System.out.println("Sum of the digits: " + sum);

}

}

**Output:**

**Sum of the digits: 30**

#### 

* 1. Write a program to count down from 10 to 0, printing each number.

**Program code:**

public class CountDown {

public static void main(String[] args) {

System.out.println("Counting down from 10 to 0:");

for (int i = 10; i >= 0; i--) {

System.out.println(i);

}

}

}

**Output:**

**Counting down from 10 to 0:**

**10**

**9**

**8**

**7**

**6**

**5**

**4**

**3**

**2**

**1**

**0**





#### Write a program to find and print the largest digit in the number 4825.

**Program code:**

public class LargestDigit {

public static void main(String[] args) {

int number = 4825;

int largest = 0;

while (number != 0) {

int digit = number % 10;

if (digit > largest) {

largest = digit;

}

number /= 10;

}

System.out.println("Largest digit: " + largest);

}

}

**Output:**

**Largest digit: 8**



* 1. Write a program to print all even numbers between 1 and 50.

**Program code:**

public class EvenNumbers {

public static void main(String[] args) {

System.out.println("Even numbers between 1 and 50:");

for (int i = 2; i <= 50; i += 2) {

System.out.println(i);

}

}

}

**Output:**

Even numbers between 1 and 50:

2

4

6

8

10

12

14

16

18

20

22

24

26

28

30

32

34

36

38

40

42

44

46

48

50





#### Write a Java program to demonstrate the use of both pre-increment and post-decrement operators in a single expression

**Program code:**

public class IncrementDecrementDemo {

public static void main(String[] args) {

int a = 5;

int b = 10;

int result = ++a + b-- - --b;

System.out.println("Result of the expression ++a + b-- - --b: " + result);

System.out.println("Value of a: " + a);

System.out.println("Value of b: " + b);

}

}

**Output:**

**Result of the expression ++a + b-- - --b: 8**

**Value of a: 6**

**Value of b: 8**



#### 

* 1. Write a program to draw the following pattern:

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

**Program code:**

public class Pattern {

public static void main(String[] args) {

for (int i = 1; i <= 5; i++) {

for (int j = 1; j <= 5; j++) {

System.out.print("\*");

}

System.out.println();

}

}

}





#### Write a program to print the following pattern:

1

2\*2

3\*3\*3

4\*4\*4\*4

5\*5\*5\*5\*5

5\*5\*5\*5\*5

4\*4\*4\*4

3\*3\*3

2\*2

**Program code:**

public class Pattern {

public static void main(String[] args) {

for (int i = 1; i <= 5; i++) {

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

System.out.print(i);

if (j < i) {

System.out.print("\*");

}

}

System.out.println();

}

for (int i = 5; i >= 1; i--) {

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

System.out.print(i);

if (j < i) {

System.out.print("\*");

}

}

System.out.println();

}

}

}





#### Write a program to print the following pattern:

\*

\*\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

**Program code:**

import java.util.Scanner;  
public class Demo1{  
       public static void main(String args[]){  
  
                 int n;  
                  System.out.println("Enter a number");                  
                  Scanner scanner = new Scanner(System.in);    
                  n = scanner.nextInt();   
                    
                  for(int i=1;i<=n;i++){  
                           if(i%2==0&&i>2){  
                                    continue;  
                           }  
                           for(int j=1;j<=i;j++){  
                                 System.out.print("\*");  
                            }  
                            System.out.println();  
              }  
  
     }  
}

#### Write a program to print the following pattern:

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

**progam code:**

public class starPyramid {

public static void main(String[] args) {

int rows = 5; // Number of rows in the pyramid

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

// Print spaces

for (int j = rows; j > i; j--) {

System.out.print(" ");

}

// Print numbers

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

System.out.print("\* ");

}

// Move to the next line

System.out.println();

}

}

}

#### Write a program to print the following pattern:

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

**program code:**

public class StarPyramid {

public static void main(String[] args) {

int rows = 5; // Number of rows in the pyramid

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

// Print leading spaces

for (int j = rows; j > i; j--) {

System.out.print(" ");

}

// Print stars

for (int k = 1; k <= (2 \* i - 1); k++) {

System.out.print("\*");

}

// Move to the next line

System.out.println();

}

}

}

#### Write a program to print the following pattern:

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

**program code:**

public class InvertedPattern {

public static void main(String[] args) {

int rows = 5; // Number of rows in the pattern

for (int i = 0; i < rows; i++) {

// Print leading spaces

for (int j = 0; j < i; j++) {

System.out.print(" ");

}

// Print stars

for (int k = rows - i; k > 0; k--) {

System.out.print("\* ");

}

// Move to the next line

System.out.println();

}

}

}

#### Write a program to print the following pattern:

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*

\*\*\*

\*

**Program code:**

public class Pattern{

public static void main(String[] args) {

for (int i = 1; i <= 7; i += 2) {

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

System.out.print("\*");

}

System.out.println();

}

for (int i = 5; i >= 1; i -= 2) {

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

System.out.print("\*");

}

System.out.println();

}

}

}

#### Write a program to print the following pattern:

1

1\*2

1\*2\*3

1\*2\*3\*4

1\*2\*3\*4\*5

**program code:**

public class Pattern{

public static void main(String[] args) {

for (int i = 1; i <= 5; i++) {

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

System.out.print(j);

if (j < i) {

System.out.print("\*");

}

}

System.out.println();

}

}

}

#### Write a program to print the following pattern:

5

5\*4

5\*4\*3

5\*4\*3\*2

5\*4\*3\*2\*1

**program code:**

public class Pattern{

public static void main(String[] args) {

for (int i = 5; i >= 1; i--) {

for (int j = 5; j >= i; j--) {

System.out.print(j);

if (j > i) {

System.out.print("\*");

}

}

System.out.println();

}

}

}

#### 21.Write a program to print the following pattern:

1

1\*3

1\*3\*5

1\*3\*5\*7

1\*3\*5\*7\*9

public class Pattern{

public static void main(String[] args) {

for (int i = 1; i <= 5; i++) {

int num = 1;

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

System.out.print(num);

num += 2;

if (j < i) {

System.out.print("\*");

}

}

System.out.println();

}

}

}

#### Write a program to print the following pattern:

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*

\*\*\*

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

public class StarPattern {

public static void main(String[] args) {

int n = 5; // Maximum width of the pattern (at the center)

// First part: Upper pyramid create

for (int i = n; i >= 1; i--) {

// Print leading spaces

for (int j = 1; j <= n - i; j++) {

System.out.print(" ");

}

// Print stars

for (int k = 1; k <= (2 \* i - 1); k++) {

System.out.print("\*");

}

System.out.println();

}

// Second part: Lower inverted pyramid create

for (int i = 2; i <= n; i++) {

// Print leading spaces

for (int j = 1; j <= n - i; j++) {

System.out.print(" ");

}

// Print stars

for (int k = 1; k <= (2 \* i - 1); k++) {

System.out.print("\*");

}

System.out.println();

}

}

}

#### Write a program to print the following pattern:

11111

22222

33333

44444

55555

public class Pattern {

public static void main(String[] args) {

for (int i = 1; i <= 5; i++) {

for (int j = 1; j <= 5; j++) {

System.out.print(i);

}

System.out.println();

}

}

}

#### Write a program to print the following pattern:

1

22

333

4444

55555

**Program code:**

public class Pattern{

public static void main(String[] args) {

for (int i = 1; i <= 5; i++) {

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

System.out.print(i);

}

System.out.println();

}

}

}

#### Write a program to print the following pattern:

1

12

123

1234

12345

**Program code:**

public class Pattern {

public static void main(String[] args) {

for (int i = 1; i <= 5; i++) {

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

System.out.print(j);

}

System.out.println();

}

}

}

#### Write a program to print the following pattern:

1

2 3

4 5 6

7 8 9 10

A close up of a logo

Description automatically generated11 12 13 14 15

**Program code:**

public class Pattern{

public static void main(String[] args) {

int num = 1;

for (int i = 1; i <= 5; i++) {

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

System.out.print(num + " ");

num++;

}

System.out.println();

}

}

}