**Assignment 3**

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

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 controlvariable be adjusted?

**Why Does This Loop Run Infinitely?**

Loop Condition Issue:

The loop condition is i < 10, starting from i = 0.

The update statement is i--, which decreases i by 1 in each iteration.

As a result, i keeps getting smaller (0, -1, -2, ...) and will always satisfy the condition i < 10.

This causes the loop to never terminate, leading to an infinite loop.

**Infinite Loop Behavior:**

The loop continuously prints decreasing values (0, -1, -2, -3, ...) until manually stopped.

It may eventually cause a stack overflow or memory issues if not terminated.

**How to Fix the Loop Control Variable?**

The loop control variable should be incremented (i++) to eventually break the loop when i reaches 10.

-------------------------------------------------------------------------------------------------------------------------- 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?

**Why Does the Loop Not Execute as Expected?**

Assignment vs. Comparison:

-In the while condition, count = 0 is an assignment, not a comparison.

-This line sets count to 0 and evaluates the condition as false since 0 is considered false in Java.

-As a result, the loop does not execute even once.

-Using = (assignment) instead of == (comparison) in conditional statements is a frequent logical error.

**How to Fix the Condition?**

Use == to compare the value of count instead of assigning it.

--------------------------------------------------------------------------------------------------------------------------

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?

**Error: Do-While Incorrect Condition**

Why Does the Loop Only Execute Once?

-A do-while loop executes the block at least once, before checking the condition.

-After the first execution, the condition num > 0 is evaluated.

-Initial num Value: 0.

-First Iteration Output: Prints 0 and increments num to 1.

-Condition Check: Now, num > 0 becomes 1 > 0 (true), so the loop continues infinitely, not just once.

-If you expected the loop to run once, the condition likely should have been num < 1 or similar.

-The provided condition num > 0 actually causes the loop to run infinitely because num keeps increasing.

--------------------------------------------------------------------------------------------------------------------------

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 expected output was numbers from 1 to 9, but the loop currently prints 1 to 10.

-This is a classic "Off-By-One" error, a common issue in loops where boundaries are incorrectly set.

**Adjust the Loop Condition**

for (int i = 1; i < 10; i++) { // Changed <= to <

System.out.println(i);

}

**Condition:** Now the loop runs while i < 10, stopping before printing 10.

--------------------------------------------------------------------------------------------------------------------------

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?

**Error Investigation**: Wrong Initialization in For Loop

Expected Output:

Print numbers from 10 down to 0 in descending order:

10 9 8 7 6 5 4 3 2 1 0

Actual Output:

The loop runs infinitely or doesn't decrement the value of i, causing unexpected results.

Fixing the Loop to Count Down Correctly:

Solution: Use i-- Instead of i++

public class CorrectedForLoop {

public static void main(String[] args) {

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

System.out.println(i);

}

}

}

Update: Now i-- decrements i by 1 on each iteration.

------------------------------------------------------------------------------------------------------------------------

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?

**Answer :**

Error Investigation – Misplaced For Loop Body

Why Does "Done" Print Only Once Outside the Loop?

Java for-loops execute only the immediate next statement if curly braces {} are omitted.

In this case:

System.out.println(i); is considered the only loop body.

System.out.println("Done"); is outside the loop and executes once after the loop finishes.

How to Fix It?

Enclose multiple statements within curly braces {} to ensure they are treated as part of the loop.

----------------------------------------------------------------------------------------------------------------------

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?

**Answer :**

Error Investigation – Uninitialized Variable in While Loop

Why Does This Code Produce a Compilation Error?

Error: variable count might not have been initialized

Why?

In Java, local variables (like count here) must be initialized before use.

The line while (count < 10) tries to read the value of count before it has been assigned any value.

This results in a compilation error since Java doesn’t auto-initialize local variables.

How to Fix It?

Initialize count before using it in the while loop.

--------------------------------------------------------------------------------------------------------------------------

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?

**Answer :**

The loop is counting down from 1 instead of counting up to 5.

The decrement num-- prevents reaching numbers 2 to 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?

**Answer:**

Error Investigation – Infinite For Loop Update

This loop does not run infinitely and works as intended based on the given code. It increments by 2, stopping once i reaches 5.

--------------------------------------------------------------------------------------------------------------------------

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?

**Answer:**

Error Investigation – Incorrect While Loop Control

What’s Wrong With the Loop Condition?

Issue: The condition while (num = 10) uses assignment (=) instead of a comparison (==).

In Java, the = operator assigns the value 10 to num, which results in 10 being treated as true (since non-zero values are truthy in this context).

Since num = 10 reassigns num to 10 on every iteration, the loop never exits — it keeps resetting num to 10 before decrementing.

--------------------------------------------------------------------------------------------------------------------------

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?

**Answer:**

Error Investigation – Incorrect Loop Update

What Will Be the Output?

The while loop starts with i = 0 and increments i by 2 on each iteration.

Loop condition: i < 5

Step-by-Step Execution:

First Iteration:

i = 0 → prints 0

i += 2 → i = 2

Second Iteration:

i = 2 → prints 2

i += 2 → i = 4

Third Iteration:

i = 4 → prints 4

i += 2 → i = 6

Condition Check:

i = 6 → condition i < 5 is false → loop exits.

Output:

0

2

4

Is This the Expected Output?

If the goal was to print even numbers less than 5, this output is correct.

If the goal was to print all numbers from 0 to 4, this update logic is incorrect.

How Should the Loop Variable Be Updated?

To Print All Numbers (0 to 4):

Use i++ to increment by 1:

public class CorrectLoopUpdate {

public static void main(String[] args) {

int i = 0;

while (i < 5) {

System.out.println(i);

i++; // Increment by 1

}

}

}

Output:

0

1

2

3

4

--------------------------------------------------------------------------------------------------------------------------

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

**Answer :**

Error Investigation – Variable Scope Issue

Scope of Variables:

In Java, a variable declared inside a block (enclosed by {}) is only accessible within that block.

In this code:

The variable x is declared inside the for loop block ({}).

Once the loop ends, x goes out of scope, meaning it no longer exists in the context where System.out.println(x); is trying to access it.

Result: The compiler throws:

error: cannot find symbol

System.out.println(x);

^

symbol: variable x location: class LoopVariableScope

--------------------------------------------------------------------------------------------------------------------------

**SECTION 2: Guess the Output**

Instructions:

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

the output.

2. Write Down Your Observations: Document each step of your dry run, including the values of

variables at each stage of execution.

3. Guess the Output: Based on your dry run, provide the expected output of the code.

4. 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.

**Answer:**

Outer loop (i) runs from 1 to 3.

Inner loop (j) runs from 1 to 2 for each iteration of i.

Inside the inner loop, it prints the pair (i j) followed by a space.

After each iteration of the outer loop, System.out.println(); moves the cursor to a new line.

Step-by-Step Execution:

i = 1:

j = 1 → Prints: 1 1

j = 2 → Prints: 1 2

Newline.

i = 2:

j = 1 → Prints: 2 1

j = 2 → Prints: 2 2

Newline.

i = 3:

j = 1 → Prints: 3 1

j = 2 → Prints: 3 2

Newline.

Final Output:

1 1 1 2

2 1 2 2

3 1 3 2

Each line corresponds to one iteration of the outer loop, printing two pairs per line based on the inner loop.

-------------------------------------------------------------------------------------------------------------------------

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.

**Answer:**

Initialization: total = 0

Loop starts with i = 5 and decrements i each iteration until i > 0.

Inside the Loop:

total += i; → Adds i to total.

if (i == 3) continue; → Skips the total -= 1; line when i is 3.

total -= 1; → Subtracts 1 from total unless continue is triggered.

Step-by-Step Execution:

Iteration i total += i i == 3? total -= 1 total After Iteration

1 5 0 + 5 = 5 false 5 - 1 = 4 4

2 4 4 + 4 = 8 false 8 - 1 = 7 7

3 3 7 + 3 = 10 true(skip) (skipped) 10

4 2 10 + 2 = 12 false 12 - 1 = 11 11

5 1 11 + 1 = 12 false 12 - 1 = 11 11

Final Output:

11

The continue at i == 3 prevented the subtraction of 1, which slightly increased the total.

For all other iterations, 1 was subtracted after adding i.

--------------------------------------------------------------------------------------------------------------------------

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.

Answer :

Initialization:

count = 0

While Loop Logic:

Loop continues while count < 5.

Each iteration:

Prints the current count.

Increments count.

Checks if (count == 3) and breaks the loop if true.

Step-by-Step Execution:

Iteration count Before Print Output count After Increment Break Condition (count == 3)

1 0 0 1 false

2 1 1 2 false

3 2 2 3 true (breaks)

The loop breaks when count becomes 3.

System.out.println(count); → prints 3 on a new line.

Final Output:

0 1 2 3

0 1 2 are printed from the loop.

3 is printed after the loop ends.

--------------------------------------------------------------------------------------------------------------------------

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.

**Answer:**

Initialization: i = 1

do-while Loop:

do block runs first, then the condition i < 5 is checked.

So, it will print i from 1 to 4 before the condition fails.

Step-by-Step Execution:

Iteration i Value (before print) Output i After Increment Condition Check (i < 5)

1 1 1 2 true

2 2 2 3 true

3 3 3 4 true

4 4 4 5 false

The loop exits when i = 5.

System.out.println(i); → prints 5 on a new line.

Final Output:

1 2 3 4 5

The numbers 1 to 4 are printed in the loop, and the 5 is printed after exiting the loop.

--------------------------------------------------------------------------------------------------------------------------

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 {

num -= i;

}

}

System.out.println(num);

}

}

// Guess the output of this loop.

**Answer:**

Initial Value:

num = 1

We loop from i = 1 to i = 4 and apply the condition:

If i is even → num += i

If i is odd → num -= i

Step-by-Step Execution:

Iteration i Value i % 2 Operation Updated num

1 1 Odd num -= 1 → 1-1 0

2 2 Even num += 2 → 0+2 2

3 3 Odd num -= 3 → 2-3 -1

4 4 Even num += 4 → -1+4 3

Final 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.

**Answer :**

We need to evaluate:

y = ++x - x-- + --x + x++;

Order of precedence:

Pre-increment/decrement (++x, --x)

Post-increment/decrement (x++, x--)

Left-to-right evaluation for operators of the same precedence.

Initial Value:

x = 5

++x → Pre-Increment x

++x increments x before using it.

x becomes 6, and 6 is used.

Expression so far: 6 - x-- + --x + x++

x-- → Post-Decrement x

x-- uses 6 first, then decrements to 5.

Expression so far: 6 - 6 + --x + x++

--x → Pre-Decrement x

--x decrements x to 4 and uses 4.

Expression so far: 6 - 6 + 4 + x++

x++ → Post-Increment x

x++ uses 4, then increments x to 5.

Expression so far: 6 - 6 + 4 + 4

Final Expression:

6 - 6 + 4 + 4

6 - 6 = 0

0 + 4 = 4

4 + 4 = 8

Final Output:

8

-------------------------------------------------------------------------------------------------------------------------

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.

**Answer :**

Pre and Post Increments/Decrements:

Initial values: a = 10, b = 5

evaluate from left to right:

++a → Pre-Increment a

++a increments a before using it.

a = 11

So, this part becomes 11.

b-- → Post-Decrement b

b-- uses 5 first, then decrements.

So, b = 4 after this operation.

The value used in multiplication is 5.

Evaluate the Multiplication:

++a \* b-- → 11 \* 5 = 55

--a → Pre-Decrement a

--a decrements a to 10 (from 11) before using.

So, this part becomes 10.

b++ → Post-Increment b

Current b = 4

b++ uses 4, then increments b to 5.

Final Expression:

Now, the expression simplifies to:

result = 55 - 10 + 4

Following operator precedence (left to right):

55 - 10 = 45

45 + 4 = 49

Final 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.

**Answer :**

Initialization: count = 0, i = 0

The loop runs while i < 4.

- First Iteration (i = 0):

Expression: i++ - ++i

i++ → uses 0, then increments i to 1.

++i → increments i again to 2 before using.

Result: 0 - 2 = -2

count = 0 + (-2) = -2

Loop increments i to 2 (due to the two increments inside the expression).

-Second Iteration (i = 2):

Expression: i++ - ++i

i++ → uses 2, increments i to 3.

++i → increments i to 4 before using.

Result: 2 - 4 = -2

count = -2 + (-2) = -4

i is now 4, so the loop exits.

Final Values:

count = -4

The loop ends when i = 4.

**Final Output:**

-4

--------------------------------------------------------------------------------------------------------------------------

**SECTION 3: Lamborghini Exercise:**

Instructions:

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

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

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

Tasks:

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

public class SumOfNaturalNumbers {

public static void main(String[] args) {

int n = 50; // First 50 natural numbers

int sum = 0;

// Using a for loop to calculate the sum

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

sum += i; // Adding each number to sum

}

System.out.println("The sum of the first 50 natural numbers is: " + sum);

}

}

--------------------------------------------------------------------------------------------------------------------------

2. Write a program to compute the factorial of the number 10.

public class FactorialCalculator {

public static void main(String[] args) {

int number = 10; // Number to calculate factorial

long factorial = 1; // Using long to handle large results

// Using a for loop to calculate factorial

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

factorial \*= i; // Multiply factorial by i

}

System.out.println("The factorial of " + number + " is: " + factorial);

}

}

-------------------------------------------------------------------------------------------------------------------------

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

public class MultiplesOfSeven {

public static void main(String[] args) {

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

// Loop from 1 to 100

for (int i = 1; i <= 100; i++) {

// Check if the number is a multiple of 7

if (i % 7 == 0) {

System.out.println(i);

}

}

}

}

----------------------------------------------------------------------------------------------------------------------

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

public class ReverseNumber {

public static void main(String[] args) {

int number = 1234; // Original number

int reversed = 0; // Variable to store the reversed number

// Logic to reverse the digits

while (number != 0) {

int digit = number % 10; // Extract last digit

reversed = reversed \* 10 + digit; // Append digit to reversed number

number /= 10; // Remove the last digit

}

System.out.println("Reversed Number: " + reversed);

}

}

--------------------------------------------------------------------------------------------------------------------

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

public class FibonacciSequence {

public static void main(String[] args) {

int limit = 21; // The maximum value in the sequence

int a = 0, b = 1; // Starting numbers of the sequence

System.out.println("Fibonacci Sequence up to " + limit + ":");

// Print the sequence

while (a <= limit) {

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

// Calculate the next number

int next = a + b;

a = b;

b = next;

}

}

}

--------------------------------------------------------------------------------------------------------------------

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

public class FirstFivePrimes {

public static void main(String[] args) {

int count = 0; // To count prime numbers found

int number = 2; // Starting from the first prime number

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

while (count < 5) {

if (isPrime(number)) {

System.out.println(number);

count++;

}

number++;

}

}

// Method to check if a number is prime

public static boolean isPrime(int num) {

if (num <= 1) {

return false;

}

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

if (num % i == 0) {

return false;

}

}

return true;

}

}

-------------------------------------------------------------------------------------------------------------------

7. Write a program to calculate the sum of the digits of the number 9876. The output should be

30 (9 + 8 + 7 + 6).

public class SumOfDigits {

public static void main(String[] args) {

int number = 9876;

int sum = 0;

// Loop to extract and sum digits

while (number != 0) {

int digit = number % 10; // Get the last digit

sum += digit; // Add it to the sum

number /= 10; // Remove the last digit

}

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

}

}

--------------------------------------------------------------------------------------------------------------------

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

public class Countdown {

public static void main(String[] args) {

// Loop from 10 down to 0

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

System.out.println(i);

}

}

}

-------------------------------------------------------------------------------------------------------------------

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

public class LargestDigitFinder {

public static void main(String[] args) {

int number = 4825;

int largestDigit = 0;

while (number > 0) {

int digit = number % 10; // Extract the last digit

if (digit > largestDigit) {

largestDigit = digit; // Update largest if current digit is greater

}

number /= 10; // Remove the last digit

}

System.out.println("The largest digit is: " + largestDigit);

}

}

------------------------------------------------------------------------------------------------------------------

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

public class EvenNumbers {

public static void main(String[] args) {

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

for (int i = 1; i <= 50; i++) {

if (i % 2 == 0) { // Check if the number is even

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

}

}

}

}

----------------------------------------------------------------------------------------------------------------

11. Write a Java program to demonstrate the use of both pre-increment and post-decrement

operators in a single expression

public class IncrementDecrementDemo {

public static void main(String[] args) {

int a = 5;

int b = 10;

// Using pre-increment and post-decrement in a single expression

int result = ++a + b--;

// Displaying the results

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

System.out.println("Value of a after pre-increment: " + a);

System.out.println("Value of b after post-decrement: " + b);

}

}

--------------------------------------------------------------------------------------------------------------------------

12. Write a program to draw the following pattern:

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

public class StarPattern {

public static void main(String[] args) {

int rows = 5; // Total rows

int columns = 5; // Total columns

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

for (int j = 0; j < columns; j++) { // Inner loop for columns

System.out.print("\*"); // Print star without newline

}

System.out.println(); // Move to the next line after each row

}

}

}

-------------------------------------------------------------------------------------------------------------------------

13. 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

1

public class NumberPattern {

public static void main(String[] args) {

int n = 5; // Number of rows for the upper half

// Upper half of the pattern

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

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

System.out.print(i);

if (j != i) {

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

}

}

System.out.println();

}

// Lower half of the pattern

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

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

System.out.print(i);

if (j != i) {

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

}

}

System.out.println();

}

}

}

--------------------------------------------------------------------------------------------------------------------------

14. Write a program to print the following pattern:

\*

\*\*

\*\*\*

\*\*\*\*\*

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

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

public class TrianglePattern {

public static void main(String[] args) {

int n = 5; // Number of rows

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

// Print spaces for alignment

for (int j = i; j < n; 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();

}

}

}

-------------------------------------------------------------------------------------------------------------------------

15. Write a program to print the following pattern:

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

public class RightAlignedTriangle {

public static void main(String[] args) {

int n = 5; // Number of rows

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

// Print spaces for alignment

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

System.out.print(" ");

}

// Print stars

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

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

}

// Move to the next line

System.out.println();

}

}

}

--------------------------------------------------------------------------------------------------------------------------

16. Write a program to print the following pattern:

\*

\*\*\*

\*\*\*\*\*

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

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

public class PyramidPattern {

public static void main(String[] args) {

int n = 5; // Number of rows

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

// Print spaces for alignment

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

System.out.print(" ");

}

// Print stars for the pyramid

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

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

}

// Move to the next line

System.out.println();

}

}

}

--------------------------------------------------------------------------------------------------------------------------

17. Write a program to print the following pattern:

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

public class InvertedRightAlignedTriangle {

public static void main(String[] args) {

int n = 5; // Number of rows

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

// Print leading spaces

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

System.out.print(" ");

}

// Print stars

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

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

}

// Move to next line

System.out.println();

}

}

}

-------------------------------------------------------------------------------------------------------------------------

18. Write a program to print the following pattern:

\*

\*\*\*

\*\*\*\*\*

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

\*\*\*\*\*

\*\*\*

\*

public class DiamondPattern {

public static void main(String[] args) {

int n = 4; // Height of the upper half

// Upper Half

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

// Print spaces

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

System.out.print(" ");

}

// Print stars

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

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

}

// Move to next line

System.out.println();

}

// Lower Half

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

// Print spaces

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

System.out.print(" ");

}

// Print stars

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

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

}

// Move to next line

System.out.println();

}

}

}

-------------------------------------------------------------------------------------------------------------------------

19. Write a program to print the following pattern:

1

1\*2

1\*2\*3

1\*2\*3\*4

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

public class NumberStarPattern {

public static void main(String[] args) {

int n = 5; // Total number of rows

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

// Print numbers and stars

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

System.out.print(j);

if (j < i) {

System.out.print("\*"); // Print star between numbers

}

}

System.out.println(); // Move to next line

}

}

}

--------------------------------------------------------------------------------------------------------------------------20. Write a program to print the following pattern:

5

5\*4

5\*4\*3

5\*4\*3\*2

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

public class ReverseNumberStarPattern {

public static void main(String[] args) {

int n = 5; // Total number to start from

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

int num = n; // Start from 5 in each row

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

System.out.print(num); // Print the number

if (j < i) {

System.out.print("\*"); // Print star between numbers

}

num--; // Decrement the number

}

System.out.println(); // Move to next line

}

}

}

-------------------------------------------------------------------------------------------------------------------------

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 OddNumberStarPattern {

public static void main(String[] args) {

int n = 5; // Number of rows

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

int num = 1; // Start from 1 for odd numbers

int count = 0; // To keep track of printed numbers

while (count < i) {

System.out.print(num); // Print the odd number

count++;

if (count < i) {

System.out.print("\*"); // Print star between numbers

}

num += 2; // Move to next odd number

}

System.out.println(); // Move to next line

}

}

}

--------------------------------------------------------------------------------------------------------------------------

22. Write a program to print the following pattern:

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

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

\*\*\*\*\*

\*\*\*

\*

\*\*\*

\*\*\*\*\*

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

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

public class DiamondLikePattern {

public static void main(String[] args) {

int n = 5; // Maximum width (number of stars at the widest point)

// Upper half of the pattern (inverted pyramid)

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

// Print leading spaces

for (int space = 0; space < n - i; space++) {

System.out.print(" ");

}

// Print stars

for (int star = 0; star < (2 \* i - 1); star++) {

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

}

System.out.println();

}

// Lower half of the pattern (pyramid)

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

// Print leading spaces

for (int space = 0; space < n - i; space++) {

System.out.print(" ");

}

// Print stars

for (int star = 0; star < (2 \* i - 1); star++) {

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

}

System.out.println();

}

}

}

--------------------------------------------------------------------------------------------------------------------

23. Write a program to print the following pattern:

11111

22222

33333

44444

55555

public class NumberPattern {

public static void main(String[] args) {

int n = 5; // Number of rows and columns

for (int i = 1; i <= n; i++) { // Loop through rows

for (int j = 1; j <= n; j++) { // Loop through columns

System.out.print(i); // Print the current row number

}

System.out.println(); // Move to the next line after each row

}

}

}

--------------------------------------------------------------------------------------------------------------------------

24. Write a program to print the following pattern:

1

22

333

4444

55555

public class NumberTrianglePattern {

public static void main(String[] args) {

int n = 5; // Number of rows

for (int i = 1; i <= n; i++) { // Loop through rows

for (int j = 1; j <= i; j++) { // Loop to print numbers in each row

System.out.print(i); // Print the current row number

}

System.out.println(); // Move to the next line after each row

}

}

}

------------------------------------------------------------------------------------------------------------

25. Write a program to print the following pattern:

1

12

123

1234

12345

public class IncreasingNumberTriangle {

public static void main(String[] args) {

int n = 5; // Number of rows

for (int i = 1; i <= n; i++) { // Loop through rows

for (int j = 1; j <= i; j++) { // Loop to print numbers in each row

System.out.print(j); // Print increasing numbers in each row

}

System.out.println(); // Move to the next line after each row

}

}

}

--------------------------------------------------------------------------------------------------------------------------

26. Write a program to print the following pattern:

1

2 3

4 5 6

7 8 9 10

11 12 13 14 15

public class NumberTrianglePattern {

public static void main(String[] args) {

int n = 5; // Number of rows

int num = 1; // Starting number

for (int i = 1; i <= n; i++) { // Loop through rows

for (int j = 1; j <= i; j++) { // Loop to print numbers in each row

System.out.print(num + " "); // Print the current number

num++; // Increment the number

}

System.out.println(); // Move to the next line after each row

}

}

}