**Java**

**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:** The loop in the “InfiniteForLoop” class runs infinitely because of the way the loop control variable ‘i’ is being decremented in the for loop.

**Solution:** To make the loop finite, the loop control variable i should be incremented rather than decremented so that it eventually reaches a value where the condition i < 10 becomes false, allowing the loop to terminate.

**Snippet 2:**

public class IncorrectWhileCondition {

public static void main(String[] args) {

int count = 5;

while (count = 0) {

System.out.println(count);

count--;

}

}

}

**Error:** The loop in the “IncorrectWhileCondition” class does not execute as expected because of a mistake in the while loop condition.

while (count = 0)

Here, the = operator is an assignment operator, not a comparison operator. This means that count = 0 sets the value of count to 0 and then evaluates the expression to 0. In Java, 0 is considered false in a conditional statement. As a result, the loop condition immediately evaluates to false, and the loop never executes.

**Solution:** To check if count is equal to 0, we should use the == comparison operator instead of the = assignment operator.

**Snippet 3:**

public class DoWhileIncorrectCondition {

public static void main(String[] args) {

int num = 0;

do {

System.out.println(num);

num++;

} while (num > 0);

}

}

**Error :** The loop in the DoWhileIncorrectCondition class only executes once because of the way the condition in the do-while loop is structured. The do-while loop starts by executing the code inside the do block, printing 0 and then incrementing num to 1. After the first iteration, the loop checks the condition num > 0. Since num is now 1, which is greater than 0, the condition is true, and the loop continues. However, as num keeps increasing (1, 2, 3, ...), the condition num > 0 remains true, which suggests the loop should continue indefinitely.

**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**: The issue with the loop boundaries in the OffByOneErrorForLoop class is related to the condition used in the for loop, which causes the loop to run one iteration more than expected.

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

System.out.println(i);

}

This loop starts with i = 1 and continues as long as i <= 10. This means that the loop will iterate from 1 to 10, inclusive, resulting in 10 iterations and printing the numbers 1 through 10.

**Solution:** To adjust the loop to print numbers from 1 to 9, the condition in the loop should be changed to i < 10 instead of i <= 10.

**Snippet 5:**

public class WrongInitializationForLoop {

public static void main(String[] args) {

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

System.out.println(i);

}

}

}

**Error:** The loop in the WrongInitializationForLoop class does not print numbers in the expected order because the loop is not correctly updating the loop control variable i.

**Problem:** The loop's goal seems to be to count down from 10 to 0, but the update statement i++ actually increases the value of i instead of decreasing it. This causes the loop to continue indefinitely, printing numbers in increasing order starting from 10 (10, 11, 12, ...), which is not the expected behaviour.

**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:** The reason "Done" prints only once, outside the loop, is due to the way the for loop and its associated statements are structured.

**Solution:** To include both System.out.println(i); and System.out.println("Done"); within the loop, you should enclose them in curly braces {} to define the loop body properly.

**Snippet 7:**

public class UninitializedWhileLoop {

public static void main(String[] args) {

int count;

while (count < 10) {

System.out.println(count);

count++;

}

}

}

**Error:** The code produces a compilation error because the variable count is declared but not initialized before being used in the while loop condition. In Java, local variables must be initialized before they are accessed.

**Solution**: We need to initialize the variable count with a value before using it in the while loop. For example, if we want the loop to start from 0, you can initialize count to 0.

**Snippet 8:**

public class OffByOneDoWhileLoop {

public static void main(String[] args) {

int num = 1;

do {

System.out.println(num);

num--;

} while (num > 0);

}

}

**Error:** The loop in the OffByOneDoWhileLoop class prints unexpected numbers because the loop is decrementing the value of num instead of incrementing it. As a result, the loop terminates prematurely.

**Solution:** To print the numbers from 1 to 5, you need to increment num inside the loop instead of decrementing it. Also, adjust the loop condition so that it runs until num reaches 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**: The loop in the InfiniteForLoopUpdate class does not run infinitely, but it might produce unexpected results based on what you expect the loop to print. The issue here is not about infinite looping but about the loop's update expression (i += 2), which increments i by 2 in each iteration.

**Solution:** If the intention is to print every number from 0 to 4, you should use a standard increment of i++ instead of i += 2.

**Snippet 10:**

public class IncorrectWhileLoopControl {

public static void main(String[] args) {

int num = 10;

while (num = 10) {

System.out.println(num);

num--;

}

}

}

**Error:** The loop in the IncorrectWhileLoopControl class executes indefinitely because of a mistake in the loop condition.

In above code, the condition while (num = 10) uses the assignment operator = instead of the comparison operator ==. This means that in each iteration, num is being assigned the value 10, and since 10 is a non-zero value, it is considered true in the loop condition. As a result, the loop never exits, causing an infinite loop.

**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:** The loop starts with i = 0 and increments i by 2 in each iteration. The loop condition is i < 5, so the loop will continue to execute as long as i is less than 5.

Output should be,

0

2

4

**Solution**: If we want the loop to print all numbers from 0 to 4 inclusively, we should update i by 1 each time instead of 2.

public class IncorrectLoopUpdate {

public static void main(String[] args) {

int i = 0;

while (i < 5) {

System.out.println(i);

i++;

}

}

}

Above is correct code.

**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:** The variable x in the LoopVariableScope class causes a compilation error because of its scope.The error occurs because x is declared inside the for loop’s block. Variables declared inside a block are only accessible within that block. This is known as **block scope**. Once the ends, the variable x is no longer accessible.

**SECTION 3: Lamborghini Exercise:**

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

**package demo;**

public class SumOfNaturalNumbers {

public static void main(String[] args) {

int n = 50;

int sum = 0;

// Loop to calculate the sum of the first 50 natural numbers

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

sum += i;

}

// Print the result

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

}

}

**Output:**

The sum of the first 50 natural numbers is: 1275

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

package demo;

public class FactorialCalculation {

public static void main(String[] args) {

int number = 10;

long factorial = 1;

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

factorial \*= i;

}

// Print the result

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

}

}

**Output –**

The factorial of 10 is: 3628800

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

package demo;

public class MultiplesOfSeven {

public static void main(String[] args) {

int start = 1;

int end = 100;

int multiple = 7;

System.out.println("Multiples of " + multiple + " between " + start + " and " + end + ":");

// Loop to find and print multiples of 7 between 1 and 100

for (int i = multiple; i <= end; i += multiple) {

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.**

package demo;

public class ReverseDigits {

public static void main(String[] args) {

int number = 1234;

int reversedNumber = 0;

// Loop to reverse the digits

while (number != 0) {

int digit = number % 10;

reversedNumber = reversedNumber \* 10 + digit;

number = number / 10;

}

System.out.println("Reversed number: " + reversedNumber);

}

}

**Output –**

Reversed number: 4321

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

package demo;

public class FibonacciSequence {

public static void main(String[] args) {

int limit = 21;

int a = 0;

int b = 1;

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

while (a <= limit) {

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

int next = a + b;

a = b; // Update a to the next number

b = next; // Update b to the next number

}

}

}

**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.**

package demo;

public class FirstFivePrimes {

public static void main(String[] args) {

int count = 0;

int number = 2;

System.out.println("The first 5 prime numbers are:");

while (count < 5) {

if (isPrime(number)) {

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

count++;

}

number++;

}

}

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;

}

}

**Output –**

The first 5 prime numbers are:

2 3 5 7 11

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

package demo;

public class SumOfDigits {

public static void main(String[] args) {

int number = 9876;

int sum = 0;

while (number > 0) {

int digit = number % 10;

sum += digit;

number = number / 10;

}

// Print the result

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

}

}

**Output –**

The sum of the digits is: 30

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

package demo;

public class Countdown {

public static void main(String[] args) {

int start = 10; // Starting number for countdown

// Loop to count down from 10 to 0

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

System.out.println(i);

}

}

}

**Output –**

10

9

8

7

6

5

4

3

2

1

0

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

package demo;

public class LargestDigitFinder {

public static void main(String[] args) {

int number = 4825;

int largestDigit = 0;

while (number > 0) {

int digit = number % 10;

if (digit > largestDigit) {

largestDigit = digit;

}

number = number / 10;

}

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

}

}

**Output –**

The largest digit is: 8

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

package demo;

public class EvenNumbers {

public static void main(String[] args) {

int start = 1; // Starting number

int end = 50; // Ending number

System.out.println("Even numbers between " + start + " and " + end + ":");

// Loop to print even numbers between 1 and 50

for (int i = start; i <= end; i++) {

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

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

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

package demo;

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 after the expression: " + a);

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

}

}

**Output –**

Result of the expression (++a + b-- + --b): 24

Value of a after the expression: 6

Value of b after the expression: 8

**12 ) Write a program to draw the following pattern:**

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

package demo;

public class StarPattern {

public static void DisplayPattern(int n) {

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

// Loop for each column

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

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

}

System.out.println();

}

}

public static void main(String[] args) {

int n = 5;

DisplayPattern(n);

}

}

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

package demo;

public class NumberPattern {

public static void DisplayPattern(int n) {

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();

}

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();

}

}

public static void main(String[] args) {

int n = 5;

DisplayPattern(n);

}

}

14. Write a program to print the following pattern:

\*

\*\*

\*\*\*

\*\*\*\*\*

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

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

package demo;

public class StarPattern {

public static void DisplayPattern(int n) {

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

// Loop to print stars in each row

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

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

}

System.out.println();

}

}

public static void main(String[] args) {

int n = 6;

DisplayPattern(n);

}

}

15. Write a program to print the following pattern:

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

package demo;

public class TriangleStar {

public static void main(int n) {

int i;

int j;

for (i = 0; i < n; i++) {

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

System.out.print(" ");

}

// inner loop to print stars.

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

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

}

System.out.println();

}

}

public static void main(String[] args) {

int n=5;

main(n);

}

}

16. Write a program to print the following pattern:

\*

\*\*\*

\*\*\*\*\*

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

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

package demo;

public class PyramidPattern {

public static void DisplayPattern(int n) {

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

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

System.out.print(" ");

}

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

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

}

System.out.println();

}

}

public static void main(String[] args) {

int n = 5;

DisplayPattern(n);

}

}

17. Write a program to print the following pattern:

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

package demo;

public class InvertedPyramidPattern {

public static void DisplayPattern(int n) {

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

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

System.out.print(" ");

}

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

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

}

System.out.println();

}

}

public static void main(String[] args) {

int n = 5;

DisplayPattern(n);

}

}

18. Write a program to print the following pattern:

\*

\*\*\*

\*\*\*\*\*

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

\*\*\*\*\*

\*\*\*

\*

package demo;

public class DiamondPattern {

public static void DisplayPattern(int n) {

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

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

System.out.print(" ");

}

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

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

}

System.out.println();

}

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

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

System.out.print(" ");

}

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

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

}

System.out.println();

}

}

public static void main(String[] args) {

int n = 4;

DisplayPattern(n);

}

}

19. Write a program to print the following pattern:

1

1\*2

1\*2\*3

1\*2\*3\*4

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

package demo;

public class NumberStarPattern {

public static void DisplayPattern(int n) {

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

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

System.out.print(j);

if (j < i) {

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

}

}

System.out.println();

}

}

public static void main(String[] args) {

int n = 5;

DisplayPattern(n);

}

}

20. Write a program to print the following pattern:

5

5\*4

5\*4\*3

5\*4\*3\*2

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

package demo;

public class ReverseNumberStarPattern {

public static void DisplayPattern(int n) {

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

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

System.out.print(j);

if (j > n - i + 1) {

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

}

}

System.out.println();

}

}

public static void main(String[] args) {

int n = 5; // Number of rows

DisplayPattern(n);

}

}

21. Write a program to print the following pattern:

1

1\*3

1\*3\*5

1\*3\*5\*7

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

package demo;

public class OddNumberStarPattern {

public static void DisplayPattern(int n) {

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

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

System.out.print((2 \* j - 1)); if (j < i) {

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

}

}

System.out.println();

}

}

public static void main(String[] args) {

int n = 5;

DisplayPattern(n);

}

}

22. Write a program to print the following pattern:

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

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

\*\*\*\*\*

\*\*\*

\*

\*\*\*

\*\*\*\*\*

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

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

package demo;

public class StarSymmetryPattern {

public static void DisplayPattern(int n) {

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

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

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

}

System.out.println();

}

for (int i = 3; i <= n; i += 2) {

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

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

}

System.out.println();

}

}

public static void main(String[] args) {

int n = 9;

DisplayPattern(n);

}

}

23. Write a program to print the following pattern:

11111

22222

33333

44444

55555

package demo;

public class RepeatedDigitPattern {

public static void DisplayPattern(int n) {

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

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

System.out.print(i);

}

System.out.println();

}

}

public static void main(String[] args) {

int n = 5;

DisplayPattern(n);

}

}

24. Write a program to print the following pattern:

1

22

333

4444

55555

package demo;

public class NumberRepeatedPattern {

public static void DisplayPattern(int n) {

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

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

System.out.print(i);

}

System.out.println();

}

}

public static void main(String[] args) {

int n = 5;

DisplayPattern(n);

}

}

25. Write a program to print the following pattern:

1

12

123

1234

12345

package demo;

public class IncreasingNumberPattern {

public static void DisplayPattern(int n) {

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

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

System.out.print(j);

}

System.out.println();

}

}

public static void main(String[] args) {

int n = 5;

DisplayPattern(n);

}

}

26. Write a program to print the following pattern:

1

2 3

4 5 6

7 8 9 10

11 12 13 14 15

package demo;

public class ConsecutiveNumberPattern {

public static void DisplayPattern(int n) {

int number = 1;

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

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

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

number++;

}

System.out.println();

}

}

public static void main(String[] args) {

int n = 5;

DisplayPattern(n);

}

}