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:

1. Why does the error or unexpected behavior occur?
2. 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);
      }
   }
}
```

ANS: Loop run infinitely because we use i—and it will
decrease each time instead of add 1 .i is negative & it is always less than 10 so this condition is always true i<10

```
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?

Snippet 2:

```
public class
  IncorrectWhileCondition { public
  static void main(String[] args) {
    int count = 5;
    while (count =
    0) {
      System.out.println(co
      unt); count--;
    }
  }
}
// Error to investigate: Why does the loop not execute as expected? What is the issue with the
condition in the
`while` loop?
```

The while loop is count=0 it is not Boolean expression so while loop need Boolean condition but count =0 is assign 0 to count which is integer and it not valid for Boolean so it compile an error the right condition is while(count>0) now loop is execute because count is greater than zero

```
public class IncorrectWhileCondition {
   public static void main(String[] args) {
      int count = 5;
      while (count > 0) {
         System.out.println(count);
         count--;
      }
   }}
```

Output:

5

4

3

2

1

Snippet 3:

```
public class
   DoWhileIncorrectCondition {
   public static void main(String[]
   args) {
      int num =0;
      do {
         System.out.println(nu
         m); 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?
```

ANS: the value of num is 0. Since a do-while loop always runs at least once the program first prints 0 and then increases num to 1. After this the condition while (num > 0); is checked. Since num is now 1 the condition is true, so the loop continues. With each iteration num keeps increasing (2 3, 4, and so on). Because num is always greater than 0 the condition never becomes false, which means the loop never stops and runs infinitely.

```
public class DoWhileIncorrectCondition {
    public static void main(String[] args) {
        int num = 0;
        do {
            System.out.println(num);
            num++;
        } while (num < 0);
    }
}
```

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?
```

Upper code loop run 10 times we have expected only till so we changing it to i<10 so it will stop at 9

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

Output:

```
1
2
3
4
5
6
7
8
9
```

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?
```

In condition i=10 and i>=0 so it print 10 and update statement i++ I increase to 100 instead of decreasing so we have to decrease I am using i—instead of i++

```
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?
```

In for loop the body will not close properly so the first statement are in loop and done statement are run after loop finish
For including all statement within loop, we have to enclose loop in {}

Snippet 7:

```
public class
   UninitializedWhileLoop {public
   static void main(String[] args) {
      int count;
      while (count < 10) {
         System.out.println(co
         unt); count++;
      }
   }
}
// Error to investigate: Why does this code produce a compilation error? What needs to be done to initialize the loop variable properly?

public class UninitializedWhileLoop {
    public static void main(String[] args) {
        int count = 0;
        while (count < 10) {
            System.out.println(count);
            count++;
        }
    }
}
```

Code gives error because count the variable count is created but not given a starting value before being used in the while loop. We must always assign a value to a variable before using it.

In java local variable didn't get value means count does not start from zero we try to use count in while condition before assign value

Snippet 8:

```
public class
   OffByOneDoWhileLoop { public
   static void main(String[] args) {
      int num =
      1; do {
         System.out.println(n
         um); 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?
```

In variable num is decremented instead of incremented
To adjustment we have to num start from 1 and increase the n++ and run the loop while num<=5

```
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 given for loop does not run infinitely, but it prints unexpected results because of how the update expression i += 2 works. The loop starts with i = 0 and increases i by 2 in each step. This means it prints 0, then 2, then 4, and stops when i reaches 6, which is outside the loop condition (i < 5). The reason for the unexpected output is that i is skipping numbers instead of counting every single one. To fix this and print all numbers from 0 to 4, we should change the update expression from i += 2 to i++. This way, i increases by 1 in each step, ensuring that the loop prints 0, 1, 2, 3, 4 as expected.

```
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(n
         um); num--;
      }
   }
}
// Error to investigate: Why does the loop execute indefinitely? What is wrong with the loop condition?
```

The loop does not actually run indefinitely instead it gives a compilation error because of a mistake in the condition. In the while condition, num = 10 is used but this is an assignment instead of a comparison. In Java = is used to assign values, while == is used to check if two values are equal. Since the condition is incorrect, the program does not compile. To fix this we should change while (num = 10) to while (num == 10) so the loop checks if num is equal to 10 instead of trying to assign 10 to num.

```
public class IncorrectWhileLoopControl {
   public static void main(String[] args) {
      int num = 10;
      while (num == 10) {
         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 given loop prints numbers but it skips some values because of the way i is updated. The loop starts with i = 0 and increases i by 2 in each step (i += 2). This means the loop prints 0 then 2 then 4 and then stops when i becomes 6 which is not less than 5. As a result, the numbers 1 and 3 are missing from the output.
To print all numbers from 0 to 4 correctly we should update i by 1 instead of 2. Changing i += 2 to i++ will ensure that i increases by 1 in each step, printing 0, 1, 2, 3, 4 as expected.

```
public class IncorrectWhileLoopControl {
    public static void main(String[] args) {
        int num = 10;
        while (num == 10) {
            System.out.println(num);
            num--;
        }
    }
}
```

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 causes a compilation error because it is declared inside the for loop. In Java a variable declared inside a block exists only within that block. Since x is created inside the loop it disappears after the loop ends. But later the program tries to use x outside the loop where it does not exist. This causes the error. To fix this we should declare x before the loop starts so it can be used both inside and outside the loop.

```
public class LoopVariableScope {
    public static void main(String[] args) {
        int x = 0;
        for (int i = 0; i < 5; i++) {
            x = i * 2;
        }
        System.out.println(x);
    }
}
```

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.
```

Code has 2 loops The outer loop runs from i = 1 to i = 3, and for each value of i, the inner loop runs from j = 1 to j = 2. The System.out.print(i + " " + j + " "); statement prints i and j together followed by a space. After the inner loop finishes, System.out.println(); moves to the next line.
First when i = 1 the inner loop runs and prints 1 1 and 1 2. Then, a new line is printed.
Next when i = 2, the inner loop runs again and prints 2 1 and 2 2 followed by a new line.
Finally when i = 3 it prints 3 1 and 3 2 then moves to a 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.
```

The program starts with total = 0 and a loop that decreases i from 5 to 1. Inside the loop total += i adds i to total. If i == 3 it skips the subtraction step using continue. For other values it subtracts 1 from total. Step by step the values are
When i = 5 total becomes 5 then subtracts 1 so total = 4
When i = 4 total becomes 8 then subtracts 1 so total = 7
When i = 3 total becomes 10 but skips the subtraction
When i = 2 total becomes 12 then subtracts 1 so total = 11
When i = 1 total becomes 12 then subtracts 1 so total = 11
The final output is 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.
```

The program initializes count = 0 and enters a while loop that runs while count < 5. It prints the value of count and increases it by 1. If count reaches 3 the loop breaks.
First count = 0 so prints 0 then count = 1 so prints 1 then count = 2 so prints 2 then count = 3 and the loop breaks. Finally count is printed so the output is
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.
```

The program initializes i = 1 and enters a do-while loop. The loop runs at least once and prints i then increases i. The loop stops when i = 5.
Step by step it prints 1 then increases i = 2 then prints 2 then increases i = 3 then prints 3 then increases i = 4 then prints 4 then increases i = 5 and stops. Finally i = 5 is printed on a new line.
The output is 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 {
        num -= i;
      }
    }
    System.out.println(num);
  }
}
// Guess the output of this loop.
```

The program starts with num = 1 and runs a loop from i = 1 to i = 4. If i is even num increases by i. If i is odd num decreases by i.
When i = 1 num becomes 1 - 1 = 0
When i = 2 num becomes 0 + 2 = 2
When i = 3 num becomes 2 - 3 = -1
When i = 4 num becomes -1 + 4 = 3
The final output is 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.
The program initializes x = 5 and calculates y = ++x - x-- + --x + x++.
First ++x increases x to 6 and returns 6
Then x-- returns 6 and decreases x to 5
Then --x decreases x to 4 and returns 4
Then x++ returns 4 and increases x to 5
So y = 6 - 6 + 4 + 4 = 8
The final output is 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.
The program initializes a = 10 and b = 5. It calculates result = ++a * b a + b++.
First ++a increases a to 11
Then b++ returns 5 but increases b to 6
The expression is 11 * 5 a + 5 but a is not valid here so the code will not compile due to a syntax error

---

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.
```

The program initializes count = 0 and runs a loop with i from 0 to less than 4. Inside the loop count += i++ - ++i.
First i = 0 so count += 0 - 2 makes count -2
Then i = 3 so count += 3 - 5 makes count -4
Loop ends because i reaches 4
The final output is -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 SumNaturalNumbers {
   public static void main(String[] args) {
      int n = 50;
      int sum = (n * (n + 1)) / 2;
      System.out.println("Sum of first 50 natural numbers: " + sum);
   }
    }
```

OUTPUT: Sum of first 50 natural numbers: 1275

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

```
public class Factorial {
   public static void main(String[] args) {
      int fact = 1;
      for (int i = 1; i <= 10; i++) {
         fact *= i;
      }
      System.out.println("Factorial of 10: " + fact);
   }
    }
```

OUTPUT: Factorial of 10: 3628800

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

```
public class MultiplesOfSeven {
   public static void main(String[] args) {
      for (int i = 1; i <= 100; i++) {
         if (i % 7 == 0) {
            System.out.print(i + " ");
         }
      }
   }
    }
```

OUTPUT: 7 14 21 28 35 42 49 56 63 70 77 84 91 98

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

```
class Main {
  public static void main(String[] args) {
    int num = 1234, reversed = 0;
    for (; num != 0; num /= 10) {
      int digit = num % 10;
      reversed = reversed * 10 + digit;
    }
    System.out.println("Reversed Number: " + reversed);
  }
}
```

OUTPUT: Reversed Number: 4321

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

```
class FibonacciSequence {
  public static void main(String[] args) {
    int a = 0, b = 1, next;

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

    for (next = a + b; next <= 21; next = a + b) {
      System.out.print(next + " ");
      a = b;
      b = next;
    }}}
```

OUTPUT: 0 1 1 2 3 5 8 13 21

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

```
class FirstFivePrimes {
 public static void main(String[] args) {
  for (int num = 2, count = 0; count < 5; num++)
   if (isPrime(num)) System.out.print(num + " "), count++;
 }

 static boolean isPrime(int n) {
  for (int i = 2; i * i <= n; i++)
   if (n % i == 0) return false;
  return true;
 }
    }
```

OUTPUT: 2 3 5 7 11

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

```
class SumOfDigits {
public static void main(String[] args) {
int num = 9876, sum = 0;

while (num > 0) {
sum += num % 10;
num /= 10;
}
    System.out.println("Sum of digits: " + sum);
}}
```

OUTPUT: Sum of digits: 30

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

```
class CountdownForLoop {
  public static void main(String[] args) {
    for (int i = 10; i >= 0; i--)
      System.out.println(i);
  }}
```

```
OUT{UT: 10
9
8
7
6
5
4
3
2
1
0
```

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

```
class LargestDigit {
  public static void main(String[] args) {
    int num = 4825;
    int largest = 0;
    while (num > 0) {
      int digit = num % 10;
      if (digit > largest) {
        largest = digit;
      }
      num /= 10;
    }
    System.out.println("Largest digit: " + largest);
  }}
```

OUTPUT: Largest digit: 8

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

```
public class EvenNumbers {
    public static void main(String[] args) {
        for (int i = 2; i <= 50; i += 2) {
            System.out.print(i + " ");
        }
    }
     }
```

Output; 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

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

```
class PreIncrementPostDecrement {
  public static void main(String[] args) {
    int x = 5, y = 10;

    int result = ++x + y--;

    System.out.println("Result: " + result);
    System.out.println("Updated x: " + x);
    System.out.println("Updated y: " + y);
  }
     }
```

OUTPUT: Result: 16
Updated x: 6
Updated y: 9

1. Write a program to draw the following pattern:

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

```
class StarPattern {
  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();
    }
  }
}
```

2. 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
```

```
class SimpleNumberPattern {
  public static void main(String[] args) {
    String[] 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"
    };

    for (String row : pattern) {
      System.out.println(row);
    }  }}
```

3. Write a program to print the following pattern:

```
*
 **
 ***
 *****
 *******
 *********
 class StarPattern {
  public static void main(String[] args) {
    for (int i = 1; i <= 5; i++) {
      for (int j = 1; j <= (2 * i - 1); j++) {
        System.out.print("*");
      }
      System.out.println();
    }
  }
 }
```

4. Write a program to print the following pattern:

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

 class CustomStarPattern {
  public static void main(String[] args) {
    String[] pattern = {"*", "**", "***", "***", "*", "****", "*"};

    for (String line : pattern) {
      System.out.println(line);
    }
  }
}
```

5. Write a program to print the following pattern:

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

```
class PyramidPattern {
  public static void main(String[] args) {
    for (int i = 1; i <= 5; i++) {
      System.out.println("*".repeat(2 * i - 1));
    }
  }
}
```

6. Write a program to print the following pattern:

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

```
class Pattern {
  public static void main(String[] args) {
    int[] lengths = {4, 1, 4, 3, 2, 1};

    for (int length : lengths) {
      for (int j = 0; j < length; j++) {
        System.out.print("*");
      }
      System.out.println();
    }
  }
}
```

7. Write a program to print the following pattern:

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

```
class StarPattern {
  public static void main(String[] args) {
    int[] lengths = {1, 3, 5, 7, 5, 3, 1};

    for (int length : lengths) {
      for (int j = 0; j < length; j++) {
        System.out.print("*");
      }
      System.out.println();
    }
  }
}
```

8. Write a program to print the following pattern:

```
1
1*2
1*2*3
1*2*3*4
1*2*3*4*5
```

```
class NumberPattern {
  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();
    }
  }
}
```

9. Write a program to print the following pattern:

```
5
5*4
5*4*3
5*4*3*2
5*4*3*2*1

class NumberPattern {
  public static void main(String[] args) {
    printPattern(5, 5, "");
  }

  static void printPattern(int n, int num, String str) {
    if (num == 0) return;

    str = str.isEmpty() ? num + "" : str + "*" + num;
    System.out.println(str);

    printPattern(n, num - 1, str);
  }
}
```

10. Write a program to print the following pattern:

```
1
1*3
1*3*5
1*3*5*7
1*3*5*7*9
class OddNumberPattern {
  public static void main(String[] args) {
    printPattern(1, 1, "");
  }

  static void printPattern(int num, int count, String str) {
    if (count > 5) return;

    str = str.isEmpty() ? num + "" : str + "*" + num;
    System.out.println(str);

    printPattern(num + 2, count + 1, str);
  }
}
```

11. Write a program to print the following pattern:

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

```
public class DiamondPattern {
    public static void main(String[] args) {
        int[] rows = {9, 7, 5, 3, 1, 3, 5, 7, 9};

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

12. Write a program to print the following pattern:

```
11111
22222
33333
44444
55555
```

```
public class NumberPattern {
    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();
        }
    }
}
```

13. Write a program to print the following pattern:

```
1
22
333
4444
55555
```

```
public class NumberPattern {
   public static void main(String[] args) {
      int i = 1;
      while (i <= 5) {
         int j = 1;
         while (j <= i) {
            System.out.print(i);
            j++;
         }
         System.out.println();
         i++;
      }
   }
}
```

14. Write a program to print the following pattern:

```
1
12
123
1234
12345
```

```
public class NumberPattern {
   public static void main(String[] args) {
      int i = 1;
      while (i <= 5) {
         int j = 1;
         while (j <= i) {
            System.out.print(j);
            j++;
         }
         System.out.println();
         i++;
      }
   }
}
```

15. 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 NumberTriangle {
    public static void main(String[] args) {
        int i = 1, num = 1;
        while (i <= 5) {
            int j = 1;
            while (j <= i) {
                System.out.print(num + " ");
                num++;
                j++;
            }
            System.out.println();
            i++;
        }
    }}
```