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

Ans. This loop runs infinitely because of the loop control variable. It should be incremented after each iteration so that the loop will run finite times.

Corrected code:

```
public class InfiniteForLoop {
        public static void main(String[] args) {
                for (int i = 0; i < 10; 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?
```

Ans. The loop does not execute as expected because the condition given in while is incorrect. It should be as count>0, which means it is checking for the condition if count is greater than to 0 if not then terminate the loop.

Corrected code:

```
public class IncorrectWhileCondition {
    public static void main(String[] args) {
        int count = 5;
        while (count > 0) {
            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?
```

Ans. This loop runs infinite times because of the condition given in the while statement. If we keep the condition as num<0, then the loop will execute only once. In do while loop, the do block executes at least once, regardless of the condition in the while statement.

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?
```

Ans. If the task expects the numbers from 1 to 9 then we need to change the boundary of the loop as i<10. Then the last i=10 will check for the condition i< 10 it is false so the loop will terminate at i=9 by printing 1 to 9 numbers.

Corrected code:

```
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
    }
}
```

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?
```

Ans. The loop uses i++ instead of i--, due to this i is increasing rather than decrease. This leads to an infinite loop since the condition i >= 0 never becomes false. Changing i++ to i-- will correctly decrement i and print numbers in descending order.

Corrected code:

```
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?
```

Ans. The loop body only includes System.out.println(i); because it's not enclosed in braces {}. To include both statements in the loop, enclose them in braces:

Corrected code:

```
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?
Ans. The code produces a compilation error because the variable count is declared but not initialized before it's used in the while loop. To fix this, initialize count when it's declared:

Corrected code:

```
public class UninitializedWhileLoop {
   public static void main(String[] args) {
      int count=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?
Ans. The loop prints numbers from 1 down to 0 because num is decremented after printing, and the loop condition checks if num is greater than 0. To print numbers from 1 to 5, initialize num to 1 and adjust the loop condition and decrement to print in the correct range:

Corrected code:

```
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?
Ans. The loop prints numbers 0 and 2 and then terminates, not running infinitely. If the intent is to print numbers from 0 to 4, the update expression should be corrected to i++:

Corrected code:

```
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?
Ans. The loop executes indefinitely because num = 10 is an assignment, not a comparison. This assignment always evaluates to 10, which is truthy, so the loop never terminates. To fix this, use a comparison operator (==) instead:

Corrected code:

```
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?
Ans. The loop will print 0, 2, and 4, then stop because i is incremented by 2 each time and eventually becomes 6, which does not satisfy the loop condition i < 5.
To achieve printing numbers from 0 to 4, update the loop variable to increment by 1:

Corrected code:

```
public class IncorrectLoopUpdate {
   public static void main(String[] args) {
      int i = 0;
      while (i < 5) {
         System.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
```

Ans. The variable x is declared inside the for loop, so its scope is limited to the loop block. It is not accessible outside the loop, causing a compilation error. To fix this, declare x outside the loop if you need to access it after the loop:

Corrected code:

```
public class LoopVariableScope {
    public static void main(String[] args) {
    intx;
    for (int i = 0; i < 5; i++) {
            int x = i * 2;
        }
        System.out.println(x); // Error: 'x' is not accessible here
    }
}
```

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.
```

Dry run:

iteration	Value of i	Value of j	Output line
1st	1	1,2	1 1 1 2
2nd	2	1,2	2 1 2 2
3rd	3	1,2	3 1 3 2

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:

iteration	Value of i	Value of total	Output line
1st	5	4	
2nd	4	7	
3rd - skip	3	10	
4th	2	11	
5th	1	11	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:

iteration	Output line	Value of count
1st	0	1
2nd	1	2
3rd	2	3
4th	3	break

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:

iteration	Value of i	Output
1st	1	1
2nd	2	1 2
3rd	3	1 2 3
4th	4	1 2 3 4
After loop gets executes	5	1 2 3 4 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 {
            num -= i;
         }
      }
      System.out.println(num);
   }
}
// Guess the output of this loop.
```

Dry run:

Iteration	Value of i	Value of num	Output
1st	1	0	
2nd	2	2	
3rd	3	-1	
4th	4	3	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:
X=5
++x -> first increment then use x
X++ -> first use then increment x
Y= 6- 6+4+4
Y=8

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.
```

Dry run:
a=10
b=5
++a ->11, a=11
b-- ->5, b=4
--a ->10, a=10
b++ ->4, b=5

result = 11*5-10+4= 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:

Iteration	Value of i	Value of count
1st	0	Count=0+ 0-2=-2
2nd	3	Count=-2+3-5=-4

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.
2. Write a program to compute the factorial of the number 10.
3. Write a program to print all multiples of 7 between 1 and 100.
4. Write a program to reverse the digits of the number 1234. The output should be 4321.
5. Write a program to print the Fibonacci sequence up to the number 21.
6. Write a program to find and print the first 5 prime numbers.
7. Write a program to calculate the sum of the digits of the number 9876. The output should be 30 (9 + 8 + 7 + 6).
8. Write a program to count down from 10 to 0, printing each number.
9. Write a program to find and print the largest digit in the number 4825.
10. Write a program to print all even numbers between 1 and 50.
11. Write a Java program to demonstrate the use of both pre-increment and post-decrement operators in a single expression
12. Write a program to draw the following pattern:

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

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
```

14. Write a program to print the following pattern:

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

15. Write a program to print the following pattern:

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

16. Write a program to print the following pattern:

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

17. Write a program to print the following pattern:

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

18. Write a program to print the following pattern:

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

19. Write a program to print the following pattern:

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

20. Write a program to print the following pattern:

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

21. Write a program to print the following pattern:

```
1
1*3
1*3*5
1*3*5*7
1*3*5*7*9
```

22. Write a program to print the following pattern:

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

23. Write a program to print the following pattern:

```
11111
22222
33333
44444
55555
```

24. Write a program to print the following pattern:

```
1
22
333
4444
55555
```

25. Write a program to print the following pattern:

```
1
12
123
1234
12345
```

26. Write a program to print the following pattern:

1
2 3
4 5 6
7 8 9 10
11 12 13 14 15