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);
                }
        }
}
```

Correct code:

```
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?
Ans: The given code snippet contains an infinite loop due to the loop control variable i being decremented (i--) in each iteration instead of incremented. This causes the loop's exit condition (i < 10) to never be met after the first iteration.
How should the loop control variable be adjusted?
Ans: The loop control variable should be adjusted to increment rather than decrement. Specifically should change i-- to i++.

Snippet 2:

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

Correct code:

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

Error to investigate:
Why does the loop not execute as expected?
The loop in the provided code does not execute as expected due to an incorrect condition in the while loop
What is the issue with the condition in the while loop?
The condition in the while loop is written as count = 0.
This is an assignment operation, not a comparison. In Java, the = operator assigns the value 0 to count, and then the condition evaluates to 0, which is treated as false in a boolean context.
Since the condition is false from the start, the loop body never executes.

Snippet 3:

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

	}
}
```

Correct code:

```
public class CorrectWhileCondition {
    public static void main(String[] args) {
        int count = 5;  // Initialize the count variable to 5
        while (count > 0) {  // Continue looping while count is greater than 0
            System.out.println(count);  // Print the current value of count
            count--;  // Decrement count by 1 after each iteration
        }
    }
}
```

Error to investigate:
Why does the loop only execute once?
In the given snippet, the do-while loop executes only once because of the condition used in the while statement.
What is wrong with the loop condition in the `do- while` loop?
The issue with the loop condition in the do-while loop is that it checks whether count> 0. This condition will always be true after the first iteration because num is incremented within the loop and will always remain greater than 0. As a result, the loop will continue indefinitely, leading to an infinite loop.

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
        }
}
```

Correct 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
        }
}
```

Error to investigate:
What is the issue with the loop boundaries?
The condition i <= 10 allows the loop to run when i is 10, which leads to the 10th iteration, printing the number 10. However, the expected output was to print only up to 9, so the loop runs one iteration too many.
How should the loop be adjusted to meet the expected output?
Adjustment: Change the loop condition to i < 10 to meet the expected output of printing numbers from 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);
                }
        }
}
```

Correct code:

```
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?
Expected : The loop is intended to print numbers in descending order from 10 to 0.
Actual : The loop doesn't behave as expected because it prints numbers indefinitely or may not execute properly depending on the environment.

What is the problem with the initialization and update statements in the `for` loop?
Problem: The update statement i++ causes the loop to count up instead of down.
Solution: Change i++ to i-- to correctly decrement i and achieve the expected descending order.

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");

        }
}
```

Correct 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");
                }
        }
}
```

Error to investigate:
Why does "Done" print only once, outside the loop?
In the given code, the statement System.out.println("Done"); is not enclosed within the loop's body. As a result, it is treated as a separate statement that executes after the loop has finished running, rather than being part of the loop.
How should the loop body be enclosed to include all statements within the loop?
Enclose the loop's body within braces {} to include all intended statements inside the loop, ensuring they are executed during each iteration.

Snippet 7:

```
public class UninitializedWhileLoop {
        public static void main(String[] args) {
                int count;

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

Correct code:

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

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

Error to investigate:
Why does this code produce a compilation error?
Compilation Error: The code produces a compilation error because the variable count is declared but not initialized before it is used in the while loop condition and within the loop body. In Java, local variables (like count in this case) must be explicitly initialized before they are accessed.
What needs to be done to initialize the loop variable properly?
Initialization: The variable count is initialized to 0 with int count = 0; before entering the while loop.
Loop Execution: The loop will now execute as expected, starting with count = 0 and printing.

Snippet 8:

```
public class OffByOneDoWhileLoop {
        public static void main(String[] args) {
                int num = 1; do {
                        System.out.println(num);
                        num--;
                } while (num > 0);
        }
}
```

Correct code:
//What adjustments are needed to print the numbers from 1 to 5?

```
public class OffByOneDoWhileLoop {
        public static void main(String[] args) {
                int num = 1;
                do {
                        System.out.println(num);
                        num++;
                } while (num <= 5);
        }
}
```

Error to investigate:
Why does this loop print unexpected numbers?
The loop starts with num = 1, prints 1, decrements num to 0, and then exits because the condition num > 0 is no longer true. As a result, the loop only prints 1 once.
What adjustments are needed to print the numbers from 1 to 5?
Replace num-- with num++ to increment num.
Change the loop condition to num <= 5 to ensure the loop stops after printing 5

Snippet 9:

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

Correcte code:

```
public class InfiniteForLoopUpdate {
   public static void main(String[] args) {
      for (int i = 0; i <= 10; i += 2) {  // Adjust condition to i <= 10
         System.out.println(i);
      }
```

```
    }
}
```

Error to investigate:
Why does the loop print unexpected results or run infinitely?
The loop does not run infinitely and prints numbers 0, 2, and 4 before terminating correctly.
How should the loop update expression be corrected?
If a different pattern or range is required, adjust the loop update expression and condition to match the desired output.

Snippet 10:

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

Correct code:

```
public class IncorrectWhileLoopControl {
    public static void main(String[] args) {
        int num = 10;
        while (num > 0) {  // Correct the condition to loop while num is greater than 0
            System.out.println(num);
            num--;
        }
    }
}
```

Error to investigate:
Why does the loop execute indefinitely?
The while loop in the code executes indefinitely because of the condition while (num = 10).
What is wrong with the loop condition?
The assignment num = 10 in the loop condition caused the loop to run indefinitely because it was not a valid comparison.

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
                }
        }
}
```

Correct code:

```
public class IncorrectLoopUpdate {
    public static void main(String[] args) {
        int i = 0;
        while (i < 5) {
            System.out.println(i);
            i++;  // Increment by 1 to include all integers from 0 to 4
        }
    }
}
```

Error to investigate:
What will be the output of this loop?
output

```
0
2
4
```

How should the loop variable be updated to achieve the desired result?
To achieve the desired result of printing all integers from 0 to 4, update the loop variable with i++ to increment by 1 in each iteration.

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
        }
}
```

Correct code:

```
public class LoopVariableScope {
        public static void main(String[] args) {
                int x = 0;// Declare x outside the loop
                for (int i = 0; i < 5; i++) {
                                x = i * 2;

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

Error to investigate:
Why does the variable 'x' cause a compilation error?
Error Message: 'x' is not accessible here indicates that the variable x is not declared in the scope where it's being accessed. Since x is declared inside the loop, it can't be used outside of it.
How does scope
Scope refers to the region of a program where a variable or method is accessible. In Java, scope is determined by where a variable or method is declared. There are several types of scopes:
Block scope, local scope, method scope, class scope

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: nested loop

```
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();
    }
  }
}
```

Dry Run:

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

Output:

```
1 1 1 2
2 1 2 2
3 1 3 2
```

Snippet 2: Decrementing Loop

```
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);
  }
}
```

Dry Run:

i=5: total += 5 (total = 5), total -= 1 (total = 4)
i=4: total += 4 (total = 8), total -= 1 (total = 7)
i=3: total += 3 (total = 10), continue skips the decrement
i=2: total += 2 (total = 12), total -= 1 (total = 11)
i=1: total += 1 (total = 12), total -= 1 (total = 11)

Output:

```
11
```

Snippet 3: While Loop with Break

```
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);
   }
}
```

Dry Run:

count=0: prints 0
count=1: prints 1
count=2: prints 2
count=3: breaks the loop

Output:

```
0 1 2 3
```

Snippet 4: Do-While Loop

```
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);
   }
}
```

Dry Run:

i=1: prints 1
i=2: prints 2
i=3: prints 3
i=4: prints 4
i=5: loop exits and prints 5

Output:

```
1 2 3 4 5
```

Snippet 5: Conditional Loop

```
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);
    }
}
```

Dry Run:

i=1: num -= 1 (num = 0)
i=2: num += 2 (num = 2)
i=3: num -= 3 (num = -1)
i=4: num += 4 (num = 3)

Output:

```
3
```

Snippet 6: Increment Decrement

```
public class IncrementDecrement {
    public static void main(String[] args) {
        int x = 5;
        int y = ++x - x-- + --x + x++;
        System.out.println(y);
    }
}
```

Dry Run:

Initial x=5
++x (x=6, y=6)
x-- (x=6, y=6-6=0, x=5 after decrement)
--x (x=4, y=0+4=4)
x++ (x=4, y=4+4=8, x=5 after increment)

Output:

```
8
```

Snippet 7: Nested Increment

```
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);
    }
}
```

Dry Run:

Initial a=10, b=5
++a (a=11)
result = 11 * 5 (result=55)
a is 11
b++ (b=6 after increment)
result = 55 - 11 + 5 (result=49)

Output:

```
49
```

Snippet 8: Loop Increment
java
Copy code

```
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);
    }
}
```

Dry Run:

i=0, count=0: i++ (i=1), ++i (i=2), count += 0 - 2 (count=-2)
i=2, count=-2: i++ (i=3), ++i (i=4), count += 2 - 4 (count=-4)
i=4: exits the loop

Output:

```
-4
```

SECTION 3:

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

---

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

   Program code:

```
public class SumOfFirst50{
    public static void main(String[] args) {
        int sum = 0;
        for (int i = 1; i <= 50; i++) {
            sum += i;
        }
        System.out.println("Sum of the first 50 natural numbers: " + sum);
    }
}
```

Output:

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

---

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

Program code:

```
public class factorialof10{
    public static void main(String arge[]){
            int number = 10;
            long factorial =1;
            for (int i = 1; i<= number;i++){
                    factorial *= i;
            }
            System.out.println("factorial of 10;" + factorial);
    }
}
```

Output:

```
    factorial of 10;3628800
```

---

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

Program code:

```
public class multipleof7{
        public static void main(String args[]){
            System.out.println("Multiples of 7 between 1 and 100:");
            int i = 7;
            for( i=7; i<=100; i+=7){
                    System.out.println(i);
            }
        }
}
```

Output:

```
        Multiples of 7 between 1 and 100:
        7
        14
        21
        28
        35
        42
        49
        56
        63
```

```
70
77
84
91
98
```

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

Program code:

```
public class reversenumbers{
        public static void main(String args[]){
            int number = 1234;
            int reversed = 0;
            while(number != 0){
                    int digit = number % 10;
                    reversed = reversed* 10+ digit;
                    number /= 10;
            }
            System.out.println("reversed number :" +reversed);
        }
}
```

Output:

```
reversed number :4321
```

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

Program code:

```
public class Fibonacci{
  public static void main(String[] args) {
    int a = 0, b = 1;
    System.out.println("Fibonacci sequence up to 21:");
    while (a <= 21) {
      System.out.println(a);
      int next = a + b;
      a = b;
      b = next;
    }
  }
}
```

Output:

```
Fibonacci sequence up to 21:
0
1
1
2
3
5
8
13
21
```

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

Program code:

```
public class First5Primes {
   public static void main(String[] args) {
      int count = 0;
      int num = 2;
      System.out.println("First 5 prime numbers:");
      while (count < 5) {
         if (isPrime(num)) {
            System.out.println(num);
            count++;
         }
         num++;
      }
   }

   public static boolean isPrime(int number) {
      if (number <= 1) return false;
      for (int i = 2; i <= Math.sqrt(number); i++) {
         if (number % i == 0) return false;
      }
      return true;
   }
}
```

Output:

```
First 5 prime numbers:
2
3
5
7
11
```

---

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

Program code:

```
public class SumOfDigits {
   public static void main(String[] args) {
         int number = 9876;
         int sum = 0;
         while (number != 0) {
                 sum += number % 10;
                 number /= 10;
         }
         System.out.println("Sum of the digits: " + sum);
   }
}
```

Output:

```
Sum of the digits: 30
```

---

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

Program code:

```
public class CountDown {
    public static void main(String[] args) {
        System.out.println("Counting down from 10 to 0:");
        for (int i = 10; i >= 0; i--) {
            System.out.println(i);
        }
    }
}
```

Output:

```
Counting down from 10 to 0:
10
9
8
7
6
5
4
3
2
1
0
```

---

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

Program code:

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

Output:

```
Largest digit: 8
```

---

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

Program code:

```
public class EvenNumbers {
    public static void main(String[] args) {
        System.out.println("Even numbers between 1 and 50:");
        for (int i = 2; i <= 50; i += 2) {
            System.out.println(i);
        }
    }
}
```

Output:

```
Even numbers between 1 and 50:
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
```

---

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

Program code:

```
public class IncrementDecrementDemo {
    public static void main(String[] args) {
        int a = 5;
        int b = 10;
        int result = ++a + b-- - --b;
        System.out.println("Result of the expression ++a + b-- - --b: " + result);
        System.out.println("Value of a: " + a);
        System.out.println("Value of b: " + b);
    }
}
```

Output:

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

```
Value of a: 6
Value of b: 8
```

---

12. Write a program to draw the following pattern:

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

Program code:

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

---

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
```

Program code:

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

```
                System.out.print("*");
            }
        }
        System.out.println();
    }
  }
}
```

---

14. Write a program to print the following pattern:

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

Program code:

```
import java.util.Scanner;
public class Demo1{
    public static void main(String args[]){

            int n;
             System.out.println("Enter a number");
             Scanner scanner = new Scanner(System.in);
             n = scanner.nextInt();

             for(int i=1;i<=n;i++){
                 if(i%2==0&&i>2){
                     continue;
                 }
                 for(int j=1;j<=i;j++){
                      System.out.print("*");
                  }
                  System.out.println();
           }

    }
}
```

---

15. Write a program to print the following pattern:

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

progam code:

```
public class starPyramid {
    public static void main(String[] args) {
        int rows = 5; // Number of rows in the pyramid

        for (int i = 1; i <= rows; i++) {
            // Print spaces
            for (int j = rows; j > i; j--) {
```

```
            System.out.print(" ");
        }

        // Print numbers
        for (int k = 1; k <= i; k++) {
            System.out.print("* ");
        }

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

---

16. Write a program to print the following pattern:

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

program code:

```
public class StarPyramid {
    public static void main(String[] args) {
        int rows = 5; // Number of rows in the pyramid

        for (int i = 1; i <= rows; i++) {
            // Print leading spaces
            for (int j = rows; j > i; j--) {
                System.out.print(" ");
            }

            // Print stars
            for (int k = 1; k <= (2 * i - 1); k++) {
                System.out.print("*");
            }

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

---

17. Write a program to print the following pattern:

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

program code:

```
public class InvertedPattern {
    public static void main(String[] args) {
        int rows = 5; // Number of rows in the pattern

        for (int i = 0; i < rows; i++) {
            // Print leading spaces
            for (int j = 0; j < i; j++) {
                System.out.print(" ");
            }

            // Print stars
            for (int k = rows - i; k > 0; k--) {
                System.out.print("* ");
            }

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

---

18. Write a program to print the following pattern:

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

Program code:

```
public class Pattern{
    public static void main(String[] args) {
        for (int i = 1; i <= 7; i += 2) {
            for (int j = 1; j <= i; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
        for (int i = 5; i >= 1; i -= 2) {
            for (int j = 1; j <= i; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

---

19. Write a program to print the following pattern:

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

1*2*3*4*5

program code:

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

---

20. Write a program to print the following pattern:

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

program code:

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

---

21. Write a program to print the following pattern:

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

program code:

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

---

22. Write a program to print the following pattern:

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

program code:

```
public class StarPattern {
    public static void main(String[] args) {
        int n = 5; // Maximum width of the pattern (at the center)

        // First part: Upper pyramid create
        for (int i = n; i >= 1; i--) {
            // Print leading spaces
            for (int j = 1; j <= n - i; j++) {
                System.out.print(" ");
            }
            // Print stars
            for (int k = 1; k <= (2 * i - 1); k++) {
                System.out.print("*");
            }
            System.out.println();
        }

        // Second part: Lower inverted pyramid create
        for (int i = 2; i <= n; i++) {
            // Print leading spaces
            for (int j = 1; j <= n - i; j++) {
                System.out.print(" ");
            }
            // Print stars
            for (int k = 1; k <= (2 * i - 1); k++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

23. Write a program to print the following pattern:

11111
22222
33333
44444
55555

Program code:

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

24. Write a program to print the following pattern:

1
22
333
4444
55555

Program code:

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

25. Write a program to print the following pattern:

1
12
123
1234
12345

Program code:

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

---

26. Write a program to print the following pattern:

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

Program code:

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