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: The reason that this loop will run infinitely is because the value of i is 0 and the condition get satisfied. Since there is decrement give i—value of I will be decrementing every time the loop will execute.
Since i is decreasing, it will always be less than 10, thus the condition i < 10 will always be true. The loop will never terminate because the condition for stopping the loop (i < 10) is always satisfied due to i continuously decreasing and never reaching a point where it is greater than or equal to 10.

The loop control variable can be adjusted in following way:

```
public class FiniteForLoop {  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 due to: The problem lies in the while loop condition:

In Java, = is the assignment operator. It is used to assign a value to a variable. The condition count = 0 assigns the value 0 to count. The result of this assignment operation is the value assigned, which in this case is 0.

In a while loop condition, the expression needs to evaluate to a boolean value (true or false). In Java, any non-zero value is considered true, and 0 is considered false. The assignment count = 0 results in 0, which is treated as false.

Since the condition count = 0 evaluates to false, the loop body will not execute at all. The loop effectively becomes a no-op (no operation) because the condition is never true.

Correct code:

```
public class CorrectWhileCondition {
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 `dowhile` loop?

Ans: do-while loop executes the block of code once before checking the condition. The condition is checked after the loop body has executed. This means the loop body will always run at least once, regardless of the condition.

The loop execute only once because the condition in the loop is incorrect. If the condition would have satisfied the declared variable int num = 0; then the loop would have executed more than once.

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: The discrepancy between expected and actual behavior is due to the condition i <= 10. This condition allows the loop to continue executing as long as i is less than or equal to 10. Consequently, the loop executes 10 times and prints numbers from 1 to 10.

Correct Code:

```
public class CorrectedLoopBoundaries {
    public static void main(String[] args) {
for (int i = 1; i < 10; i++) {
            System.out.println(i);
        }
        // Now prints numbers 1 to 9, which matches the expected behavior
    }
}
```

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: This loop do not print numbers in the expected order because:
i++ increases i by 1 in each iteration. Since i is initially set to 10, the loop will start with 10, and on eachiteration, i becomes 11, 12, and so on. This causes i to increase indefinitely, which means i will neverbecome less than 0, and the loop will never terminate.

In this code for loop, the initialization statement int i = 10 sets the loop variable i to 10, which is correct. However, the issue is with the update statement i++.

Correct 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?

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: This code produce a compilation error because we have not initialized count before using it in while loop. First the count needs to be initialized and then it should be used in while loop condition.

Correct Code:

```
public class UninitializedWhileLoop {
   public static void main(String[] args) {
int count = 0; // Initialize count         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: he do-while loop in this code prints unexpected numbers because the loop condition is checked after the loop body executes, leading to an off-by-one error.

Correct code:

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

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 is not infinite and does not produce unexpected results based on the given code. It correctly prints 0, 2, and 4.

The increment expression i += 2 is appropriate if you want to print numbers starting from 0 and increasing by 2 each time until the condition i < 5 is false.

Correct 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 will execute indefinitely due to a mistake in the while loop's condition. The condition used is num = 10, which is an assignment operation rather than a comparison operation.

The thing that is wrong with the loop condition is that assignment operator is used in place of a comparision operator.

Correct Code:

```
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?

Ans: The output of this loop will be the following: 0

2

4

Correct Code:

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

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 causes a compilation error because of the scope in which it is declared.
The variable x is declared inside the for loop's block: for (int i = 0; i < 5; i++) { int x = i * 2; }.
This means that x is local to the block of the for loop.
Variables declared inside a block are only accessible within that block. Once the block ends, the variable is no longer in scope and cannot be accessed.

Correct Code:

```
public class LoopVariableScope {
   public static void main(String[] args) {
      int x = 0; // Declare x outside the loop and initialize it

      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. **Ans:**
The output of this snippet is:

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.
Ans: The output of this snippet 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.
Ans: The output of this while loop 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. **Ans:**
The output of this do-while loop 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.
Ans: The output of this loop will be:

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. **Ans:**
The output of the code 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.
Ans: The output of the code is

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.
Ans: The output of this code will be:

-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. **Ans:**

Program to calculate the sum of the first 50 natural numbers:

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

Output:

The sum of the first 50 natural numbers is: 1275

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

Ans: Program to compute the factorial of the number 10: public class Factorial {

```
public static void main(String[] args) {        int
number = 10; // Number to compute the factorial of
long factorial = 1; // Initialize factorial to 1

      // Loop to calculate the factorial        for
(int i = 1; i <= number; i++) {
factorial *= i; // Multiply factorial by i
      }
       System.out.println("The factorial of " + number + " is: " + factorial);
   }
}
```

Output:

The factorial of 10 is: 3628800

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

Ans: Program to print all multiples of 7 between 1 and 100 is:

```
public class MultiplesOfSeven {
public static void main(String[] args) {                //
Iterate from 1 to 100                for (int i = 1; i <=
100; i++) {
        // Check if the current number is a multiple of 7
if (i % 7 == 0) {
          // Print the multiple of 7
```

```
            System.out.println(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. **Ans:**

Program to reverse the digits of the number 1234 is:

```
public class ReverseDigits {
   public static void main(String[] args) {
int number = 1234; // Number to reverse
      int reversed = 0; // Variable to store the reversed number

      while (number != 0) {
         int digit = number % 10; // Extract the last digit          reversed =
reversed * 10 + digit; // Append digit to reversed number          number =
number / 10; // Remove the last digit from the original number
      }
     System.out.println("The reversed number is: " + reversed);
   }
}
```

Output:

The reversed number is: 4321

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

Ans: Program to print the Fibonacci sequence up to the number 21 is:

```
public class FibonacciSequence {
```

```
   public static void main(String[] args) {         int target = 21; // The number up to
which we want to print the Fibonacci sequence         int a = 0; // First number in the
Fibonacci sequence
        int b = 1; // Second number in the Fibonacci sequence

        // Print Fibonacci sequence up to target
        System.out.println("Fibonacci sequence up to " + target + ":");

        // Print the first two numbers if they are within the target
if (a <= target) {
            System.out.print(a + " ");
        }
        if (b <= target) {
            System.out.print(b + " ");
        }

        // Compute the rest of the Fibonacci sequence         while (true) {
int next = a + b; // Compute the next Fibonacci number             if
(next > target) { // Stop if the next number exceeds the target
break;
            }
            System.out.print(next + " "); // Print the next number
a = b; // Update a to the previous number             b = next; //
Update b to the new number
        }
    }
}
```

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. **Ans:**

Program to find and print the first 5 prime numbers

```
public class FirstFivePrimes {     public
static void main(String[] args) {
        int count = 0; // To keep track of the number of primes found
int number = 2; // The number to check for primality

        // Continue until we find the first 5 prime
numbers         while (count < 5) {             if
(isPrime(number)) {
```

```
System.out.println(number); // Print the prime number
count++; // Increment the count of primes found
            }
            number++; // Move to the next number
        }
    }

    // Method to check if a number is prime     public static boolean
isPrime(int num) {         if (num <= 1) {             return false; //
Numbers less than or equal to 1 are not prime
        }
        for (int i = 2; i <= Math.sqrt(num); i++) {             if (num %
i == 0) {                 return false; // Number is divisible by i, so it's
not prime
            }
        }
        return true; // Number is prime
    }
}
```

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).

Ans: Program to calculate the sum of the digits of the number 9876 are:

```
public class SumOfDigits {     public static void main(String[] args) {
        int number = 9876; // Number whose digits we want to sum
int sum = 0; // Initialize sum to 0

        // Loop to calculate the sum of digits
while (number > 0) {
            int digit = number % 10; // Extract the last digit
sum += digit; // Add digit to sum
            number /= 10; // Remove the last digit from the number
        }

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

Output:

The sum of the digits is: 30

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

Ans: Write a program to count down from 10 to 0 is:

```
public class Countdown {     public static void
main(String[] args) {
      int i = 10; // Start counting from 10

      // Count down from 10 to 0 using a while loop
while (i >= 0) {
         System.out.println(i); // Print each number
i--; // Decrement the value of i
      }
   }
}
```

Output:

```
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.

Ans: Program to find and print the largest digit in the number 4825:

```
public class LargestDigit {     public static void main(String[] args) {
int number = 4825; // The number to analyze
      int largestDigit = 0; // Initialize largest digit to 0

      // Process each digit of the number
while (number > 0) {
         int digit = number % 10; // Extract the last digit
if (digit > largestDigit) {
            largestDigit = digit; // Update largest digit if current digit is larger
         }
         number /= 10; // Remove the last digit from the number
      }

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

Output:

The largest digit is: 8

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

Ans: Program to print all even numbers between 1 and 50 is:

```
public class EvenNumbers {     public static
void main(String[] args) {         // Iterate
through numbers from 1 to 50         for (int i
= 1; i <= 50; i++) {             // Check if the
number is even             if (i % 2 == 0) {
                System.out.println(i); // Print the even number
            }
        }
    }
}
```

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 **Ans:**

```
public class IncrementDecrementDemo {
public static void main(String[] args) {
int a = 5;         int b = 10;
        int result;
```

```
        // Using both pre-increment and post-decrement operators
        result = ++a + b-- - --b + a--;

        // Printing the results of the operations
        System.out.println("a after operations: " + a);
        System.out.println("b after operations: " + b);
        System.out.println("Result of the expression: " + result);
    }
}
```

Output:

```
a after operations: 5
b after operations: 8
Result of the expression: 14
```

12. Write a program to draw the following pattern:

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

Ans:

```
public class DrawPattern {
public static void main(String[] args) {
int rows = 5; // Number of rows  int
columns = 5; // Number of columns //
Loop through each row for (int i = 0; i <
rows; i++) {
 // Loop through each column in the current row for
(int j = 0; j < columns; j++) {
  System.out.print("*"); // Print an asterisk without newline
}
  System.out.println(); // Move to the next line after printing all columns
    }
 }
}
```

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
```

Ans:

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();
 }
 }
}
1
```

14. Write a program to print the following pattern:

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

Ans:

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();
 }
 }
}
```

15. Write a program to print the following pattern:

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

Ans:

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

16. Write a program to print the following pattern:

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

Ans:

Program code:

```
public class Pattern {  public static
void main(String[] args) {  for (int i =
1; i <= 9; i += 2) {  for (int j = 1; j <=
i; j++) {
 System.out.print("*");
}
 System.out.println();
}
}
}
```

17. Write a program to print the following pattern:

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

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

18. Write a program to print the following pattern:

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

Ans: 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
```

Ans: 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
```

Ans:
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
```

Ans:
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:

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

Ans:

Program code:

```
public class PatternPrinter {
    public static void main(String[] args) {
        int n = 5; // Number of rows for the hourglass and diamond parts
       // Hourglass Shape
        for (int i = n; i >= 1; i--) {
            // Print leading spaces
            for (int j = 0; j < n - i; j++) {
                System.out.print(" ");
            }
            // Print stars
            for (int j = 0; j < (2 * i - 1); j++) {
                System.out.print("*");
            }
            // Move to the next line
            System.out.println();
        }

        // Diamond Shape
        for (int i = 2; i <= n; i++) {
            // Print leading spaces
            for (int j = 0; j < n - i; j++) {
                System.out.print(" ");
```

```
        }
        // Print stars
        for (int j = 0; j < (2 * i - 1); j++) {
            System.out.print("*");
        }
        // Move to the next line
        System.out.println();
    }
  }
}
```

23. Write a program to print the following pattern:

```
11111
22222
33333
44444
55555
```

Ans:

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
```

Ans:

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
```

Ans:

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
```

Ans:

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();
 }
 }
}
```