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?

The issue with the code is in the loop's increment/decrement condition. Specifically, the loop control variable i is being decremented (i--) in each iteration, but the condition to run the loop is i < 10.

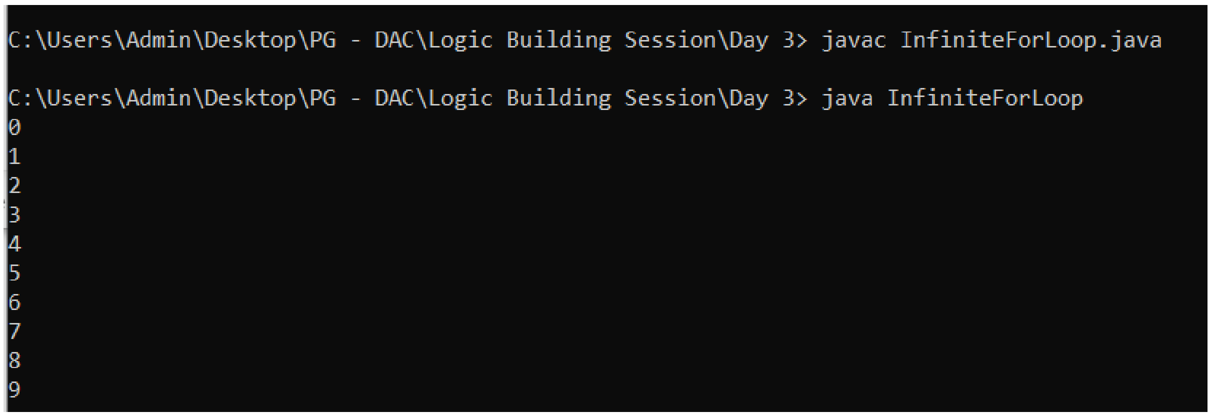
- The loop starts with i = 0, and the condition is to run as long as i < 10.
- In each iteration, i-- decrements i by 1 (making i more negative).
- As i becomes negative (starting from -1 after the first iteration), it will always be less than 10, causing the loop to run indefinitely.
- The loop should increment `i` (using `i++`) so that it eventually reaches a value equal to or greater than 10, causing the loop to terminate.

Correct code:

```
public class InfiniteForLoop {
    public static void main(String[] args) {
        for (int i = 0; i < 10; i++) {  // Change i-- to i++
System.out.println(i);
```

```
            }
        }
    }
```

Output:



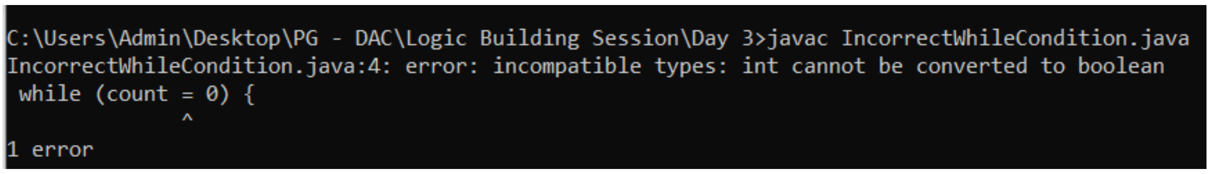
```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3> javac InfiniteForLoop.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3> java InfiniteForLoop
0
1
2
3
4
5
6
7
8
9
```

****

Snippet 2:

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

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac IncorrectWhileCondition.java
IncorrectWhileCondition.java:4: error: incompatible types: int cannot be converted to boolean
 while (count = 0) {
              ^
1 error
```

// Error to investigate: Why does the loop not execute as expected? What is the issue with the condition in the `while` loop?

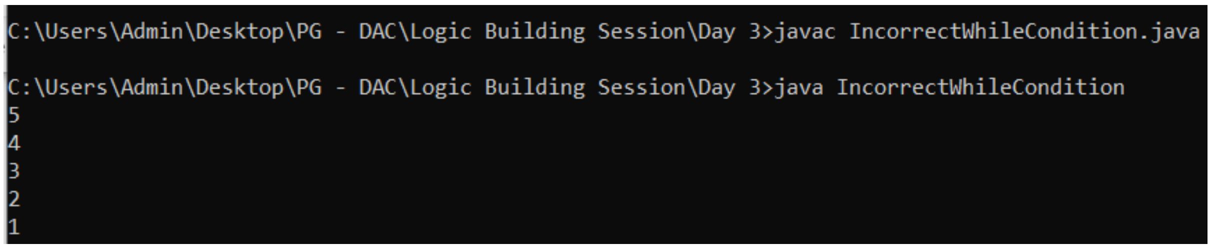
Here, the = operator is used, which is an assignment operator, not a comparison operator. This means that count = 0 assigns the value 0 to the variable count and then evaluates the condition as 0, which is treated as false in Java. As a result, the loop condition is always false, and the loop does not execute at all.

Correct code:

You should use the **equality comparison operator** == to check if count is equal to 0, not the assignment operator =.

```
public class IncorrectWhileCondition {

    public static void main(String[] args) {

        int count = 5;

        while (count > 0) {  // Use count > 0 to run the loop until count reaches 0

            System.out.println(count);

            count--;

        }

    }

}
```

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac IncorrectWhileCondition.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java IncorrectWhileCondition
5
4
3
2
1
```

count > 0 ensures that the loop continues as long as count is greater than 0. When count reaches 0, the loop terminates.

count-- decreases the value of count by 1 after each iteration.

Now, the loop will print the values of count starting from 5 and decrementing until it reaches 0, executing as expected.

****

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?

Loop get executed continually
The issue with your code is in the loop condition: while (num > 0);

why this causes the loop to execute only once:

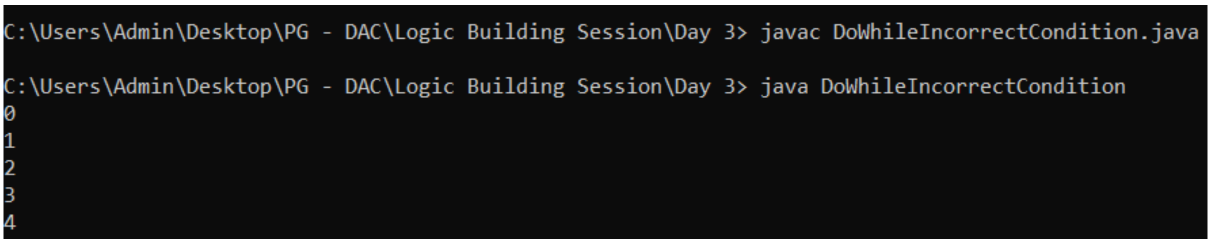
1. **do-while loop structure**: The do-while loop guarantees that the block of code inside the do section will run **at least once**, regardless of the loop condition. So, in this case, the first time the loop runs, num starts at 0, and System.out.println(num) prints 0.
2. **Condition check**: After the first iteration, num is incremented to 1. The loop then checks if the condition num > 0 is true. Since num is now 1, the condition num > 0 is true, but when the loop checks the condition at the end of the iteration, it checks the updated value of num (which is 1).

However, this doesn't explain why the loop stops after one iteration. **The condition is num > 0** — the problem here is that, initially, the loop starts with num = 0, so the condition is **false** during the second iteration after the increment, and the loop stops running.

Correct code:

```
 public class DoWhileIncorrectCondition {
    public static void main(String[] args) {
        int num = 0;
        do {
            System.out.println(num);  // Prints num
            num++;  // Increment num after each iteration
        } while (num < 5);  // Continue the loop as long as num is less than 5
    }
}
```

O/P:



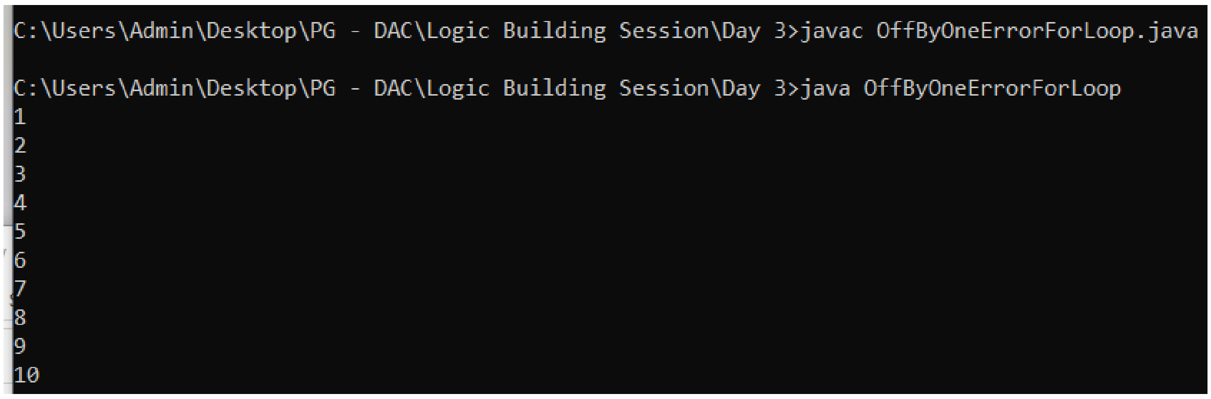
```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3> javac DoWhileIncorrectCondition.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3> java DoWhileIncorrectCondition
0
1
2
3
4
```

****

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
 }
}
```

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac OffByOneErrorForLoop.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java OffByOneErrorForLoop
1
2
3
4
5
6
7
8
9
10
```

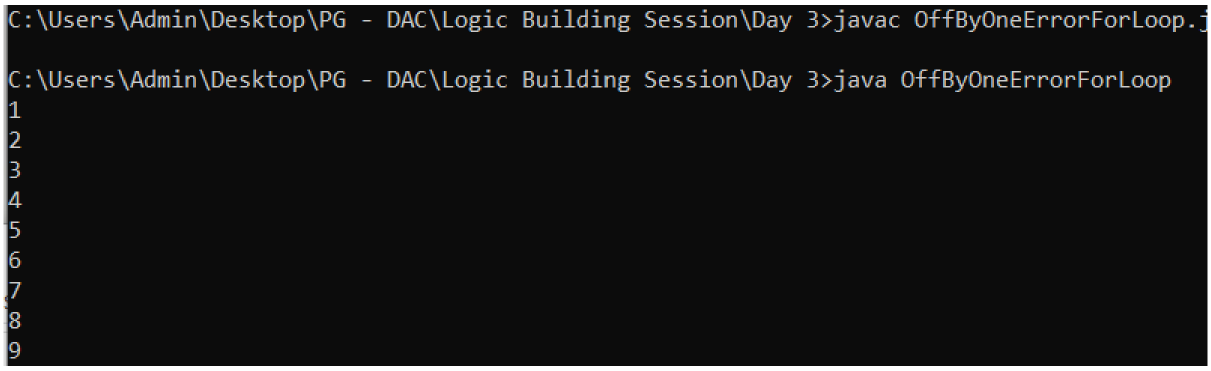
// Error to investigate: What is the issue with the loop boundaries? How should the loop be adjusted to meet the expected output?

How to adjust the loop:

Change the loop condition to i < 10, which ensures the loop stops when i reaches 10 and only prints numbers from 1 to 9.

```
public class OffByOneErrorForLoop {
    public static void main(String[] args) {
        for (int i = 1; i < 10; i++) {  // Change condition to i < 10
            System.out.println(i);
        }
        // Now the loop will print numbers from 1 to 9, as expected
    }
}
```

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac OffByOneErrorForLoop.j

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java OffByOneErrorForLoop
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?

Infinite loop print

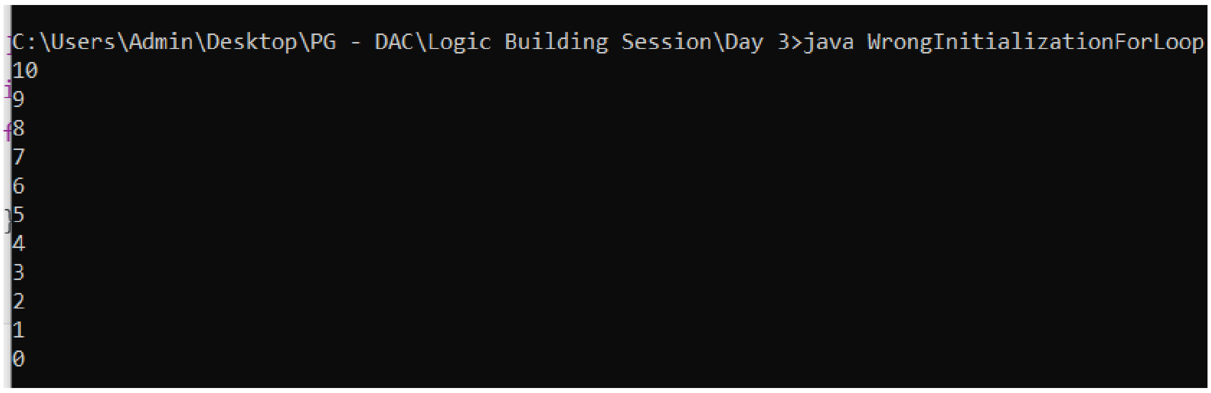
Initialization………… **int i = 10; — This sets the initial value of i to 10, which is fine.**

Condition …………i >= 0; — This condition checks whether i is greater than or equal to 0. This is also fine, as the loop should continue running as long as i is non-negative.

Update…………i++ — This is the key issue. The i++ increments i by 1 after each iteration, so i is increasing with each loop. However, this is not what we want because we want the loop to print numbers in descending order (from 10 down to 0), so i should decrease, not increase.

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

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java WrongInitializationForLoop
10
9
8
7
6
5
4
3
2
1
0
```

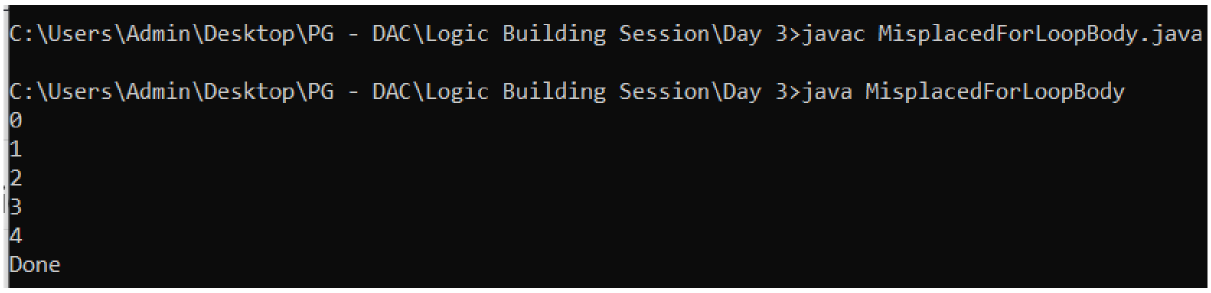
i-- ensures that i is decremented after each iteration, so the loop prints numbers from 10 down to 0 (inclusive).

****

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");
 }  }
```

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac MisplacedForLoopBody.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java MisplacedForLoopBody
0
1
2
3
4
Done
```

// Error to investigate: Why does "Done" print only once, outside the loop?

In Java, when you have a single statement immediately following a control structure (like if, for, while, etc.), curly braces { } are optional. So, in this case, the loop body only contains the single statement System.out.println(i);, which will execute five times.

The second statement, System.out.println("Done");, is not part of the loop because it is outside the loop body. This means it will execute once after the loop finishes, printing "Done" after the loop completes.

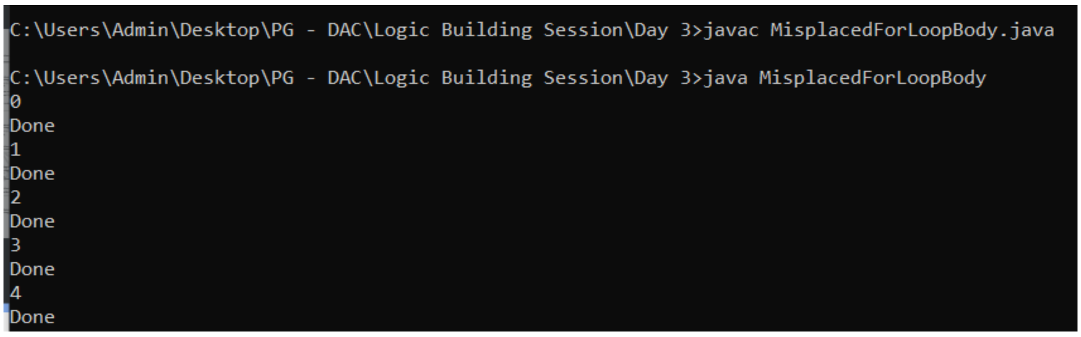
How should the loop body be enclosed to include all statements within the loop?

You need to enclose the body of the loop (including both System.out.println(i); and System.out.println("Done");) inside curly braces { } if you want both statements to be part of the loop.

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");  //  "Done" will be printed within the loop
        } } }
```

O/P:

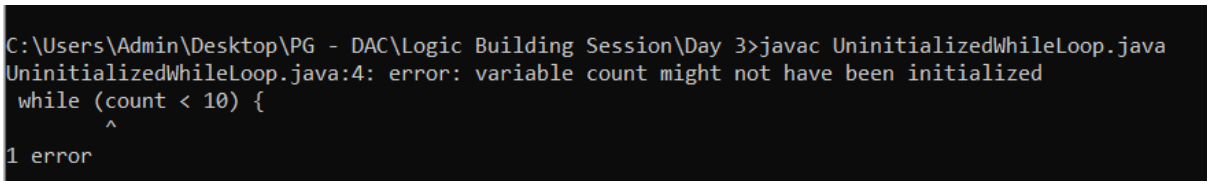


```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac MisplacedForLoopBody.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java MisplacedForLoopBody
0
Done
1
Done
2
Done
3
Done
4
Done
```

Snippet 7:

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

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac UninitializedWhileLoop.java
UninitializedWhileLoop.java:4: error: variable count might not have been initialized
 while (count < 10) {
        ^
1 error
```

// Error to investigate: Why does this code produce a compilation error?

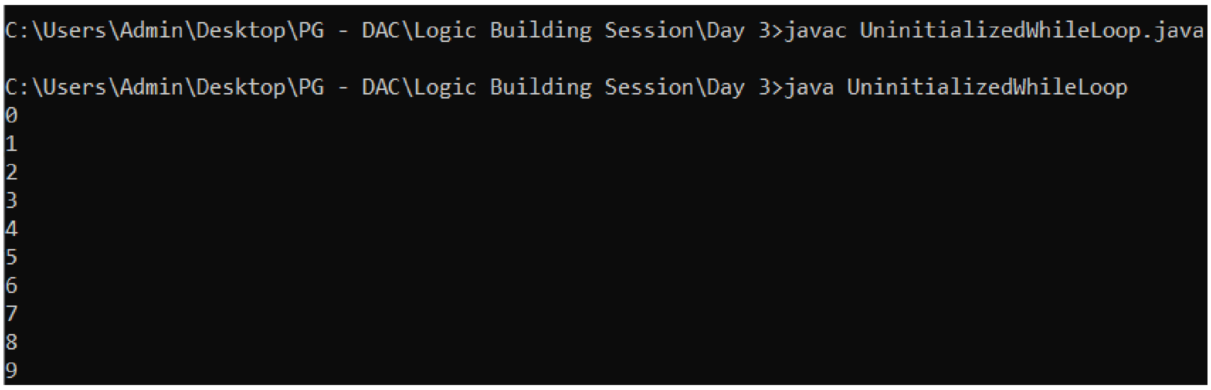
What needs to be done to initialize the loop variable properly?

The issue with this code is that the variable count is declared but not initialized before the while loop is executed:

Correct code:

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

O/P:



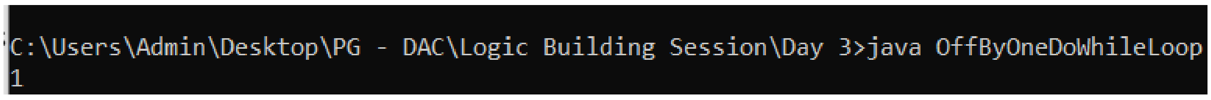
```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac UninitializedWhileLoop.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java UninitializedWhileLoop
0
1
2
3
4
5
6
7
8
9
```

****

Snippet 8:

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

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java OffByOneDoWhileLoop
1
```

// Error to investigate: Why does this loop print unexpected numbers?

The issue with this code is in the **decrementing** logic inside the do-while

What adjustments are needed to print the numbers from 1 to 5?
Initial value of num: The loop starts with num = 1, which is fine.

Loop behavior: In each iteration of the do-while loop, num is decremented by 1 (num--).

The loop runs as long as num > 0. Since the loop runs at least once (this is how do-while works), on the first iteration:

- num = 1, so it prints 1, then decrements num to 0.

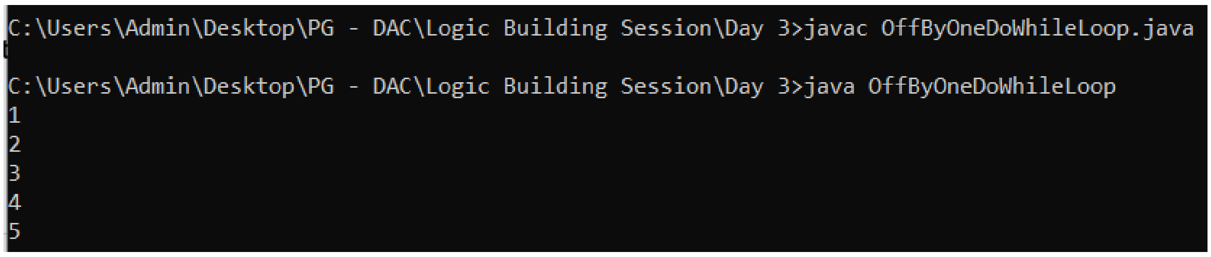
- On the second iteration, num = 0, so the condition num > 0 becomes false, and the loop terminates.

So the loop will only print 1 and stop because of the decrementing logic.

Correct Code:

```
public class OffByOneDoWhileLoop {
    public static void main(String[] args) {
        int num = 1;
        do {
            System.out.println(num);  // Print current num
            num++;  // Increment num
        } while (num <= 5);  // Continue the loop as long as num is less than or equal to 5
    }
}
```

O/P:



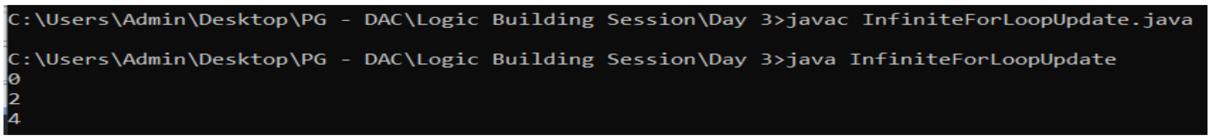
```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac OffByOneDoWhileLoop.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java OffByOneDoWhileLoop
1
2
3
4
5
```

****

Snippet 9:

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

// Why does the loop print unexpected results or run infinitely?



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac InfiniteForLoopUpdate.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java InfiniteForLoopUpdate
0
2
4
```

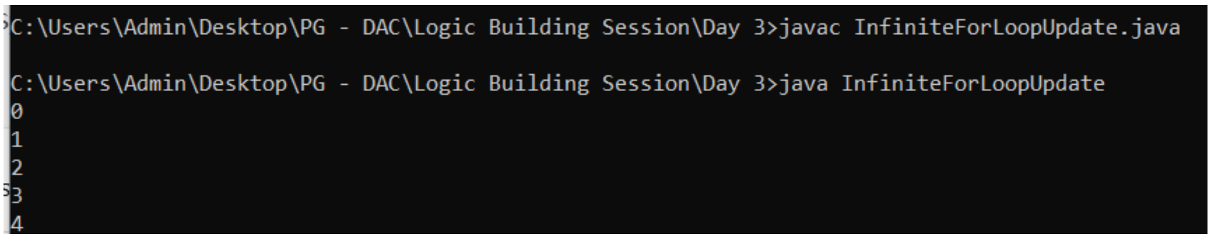
How should the loop update expression be corrected?

if you were expecting it to print all values from 0 to 4 (inclusive), you'd need to change the increment to i++ so that it increments by 1 rather than 2. This would then print 0, 1, 2, 3, and 4.

Correct Code:

```
public class InfiniteForLoopUpdate {

   public static void main(String[] args) {

      for (int i = 0; i < 5; i++) {  // Change i += 2 to i++ (increment by 1)

         System.out.println(i);

      }

   }

}
```

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac InfiniteForLoopUpdate.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java InfiniteForLoopUpdate
0
1
2
3
4
```

****

Snippet 10:

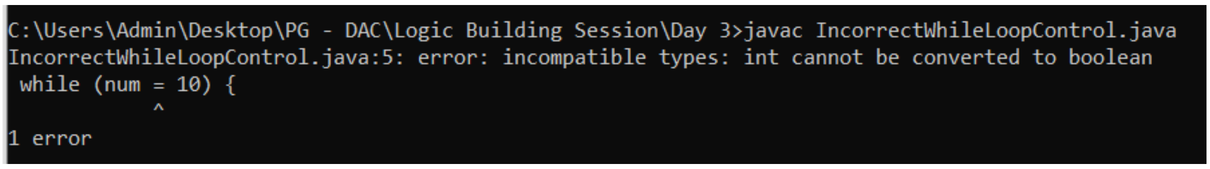
```
public class IncorrectWhileLoopControl {

 public static void main(String[] args) {

 int num = 10;

 while (num = 10) {

 System.out.println(num);

 num--;
```

}

}

}

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac IncorrectWhileLoopControl.java
IncorrectWhileLoopControl.java:5: error: incompatible types: int cannot be converted to boolean
 while (num = 10) {
            ^
1 error
```

// Why does the loop execute indefinitely? What is wrong with the loop condition?

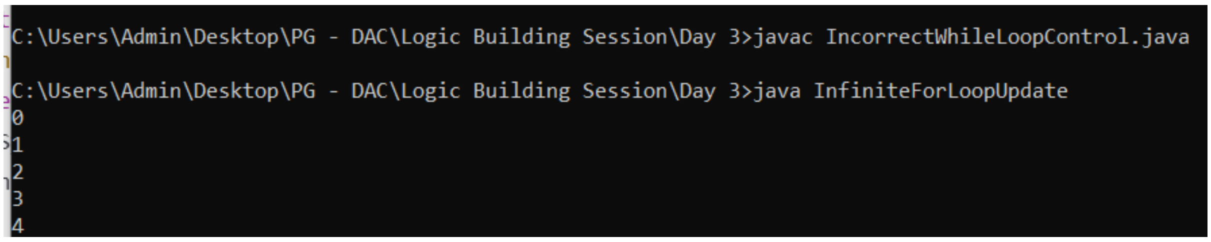
The expression **num = 10** is an assignment, not a comparison. This means that instead of checking if num is equal to 10, it assigns the value 10 to the variable num. The result of this assignment is 10, which is considered true in Java because any non-zero value is treated as true in a boolean context.

As a result, the loop condition always evaluates to true, and the loop will run indefinitely.

Correct Code:

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

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac IncorrectWhileLoopControl.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java InfiniteForLoopUpdate
0
1
2
3
4
```

****

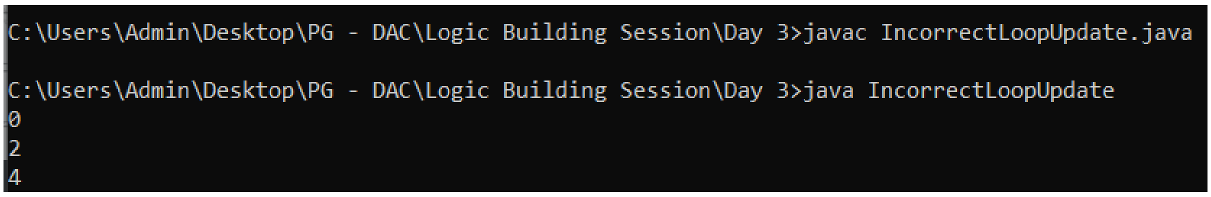
Snippet 11:

```
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
 }
 }
}
```

O/P:

// What will be the output of this loop?

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac IncorrectLoopUpdate.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java IncorrectLoopUpdate
0
2
4
```

The error in the provided code lies in the way the loop variable i is updated within the loop. Specifically, the code is incrementing i by 2 in each iteration, which means the loop will skip some values of i and may not print the desired output.

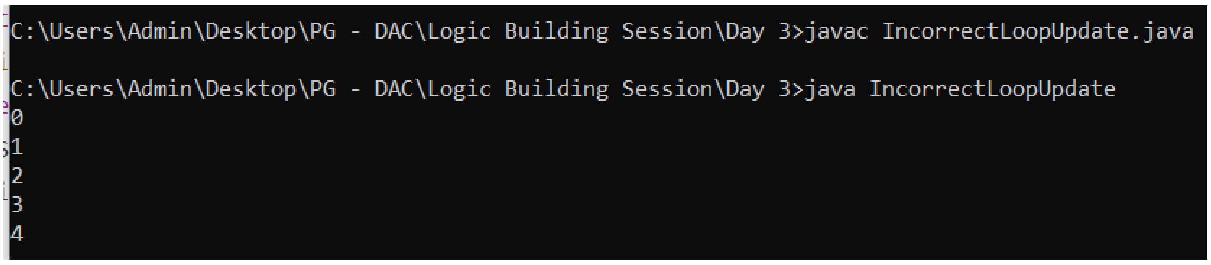
How should the loop variable be updated to achieve the desired result?

To ensure that the loop prints all the integers from 0 to 4, you should update i by 1 instead of 2. This will allow the loop to go through every integer in that range.

```
public class IncorrectLoopUpdate {
   public static void main(String[] args) {
      int i = 0;
      while (i < 5) {
         System.out.println(i);
         i++;  // Fixed: increment by 1
```

```
        }
    }
}
```

O/P:



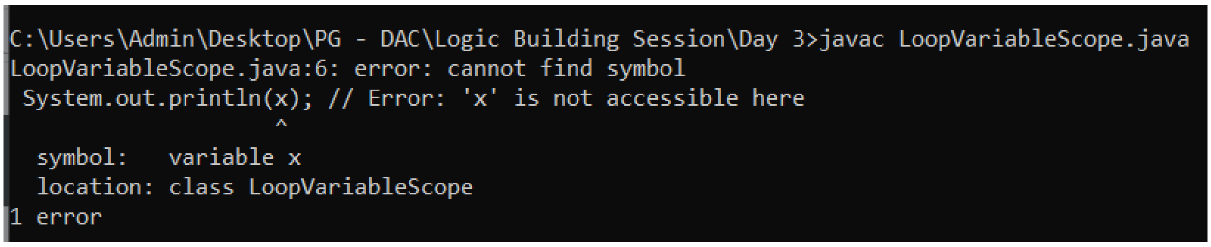
```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac IncorrectLoopUpdate.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java IncorrectLoopUpdate
0
1
2
3
4
```

****

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
 }
}
```

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac LoopVariableScope.java
LoopVariableScope.java:6: error: cannot find symbol
 System.out.println(x); // Error: 'x' is not accessible here
                    ^
  symbol:   variable x
  location: class LoopVariableScope
1 error
```

// Error to investigate: Why does the variable 'x' cause a compilation error? How does scope

The variable x is declared inside the for loop with the statement int x = i * 2;. This means x is local to the loop block.

In Java, a variable declared within a block (such as a for loop) is only accessible within that block. Once the block ends (i.e., when the loop finishes), the variable x is no longer available.

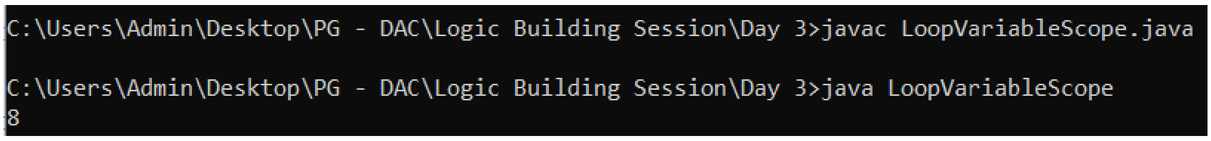
The statement System.out.println(x); is trying to access x outside of the for loop, where x is out of scope. This results in a compilation error because x does not exist in that part of the code.

Correct code:

Need to declare x outside the loop if you want to access it after the loop finishes**.**

```
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;  // Update x inside the loop
    }
    System.out.println(x);  // Now x is accessible here
  }
}
```

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac LoopVariableScope.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java LoopVariableScope
8
```

++

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.

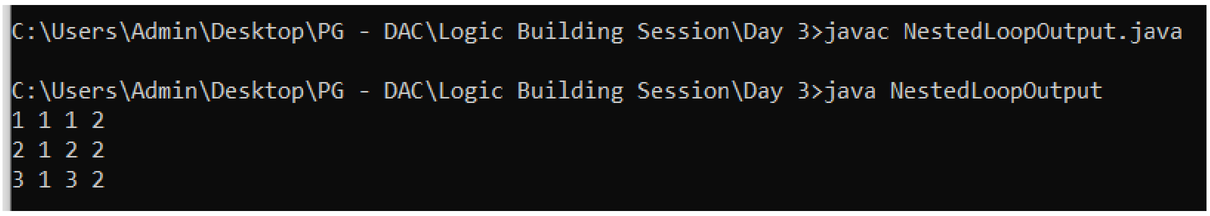
Dry Run:

1. Outer loop (i) starts at 1, runs while i <= 3.
   - Inner loop (j) starts at 1, runs while j <= 2.
   - First iteration:
     - i = 1
     - Inner loop runs for j = 1 and j = 2:
       - Prints 1 1 and 1 2 (i.e., i and j values).
     - After the inner loop finishes, System.out.println() adds a new line.
   - Second iteration:

        - i = 2
        - Inner loop runs for j = 1 and j = 2:
            - Prints 2 1 and 2 2.
        - After the inner loop finishes, System.out.println() adds a new line.
    - Third iteration:
        - i = 3
        - Inner loop runs for j = 1 and j = 2:
            - Prints 3 1 and 3 2.
        - After the inner loop finishes, System.out.println() adds a new line.

Output :

- First iteration: 1 1 1 2
- Second iteration: 2 1 2 2
- Third iteration: 3 1 3 2



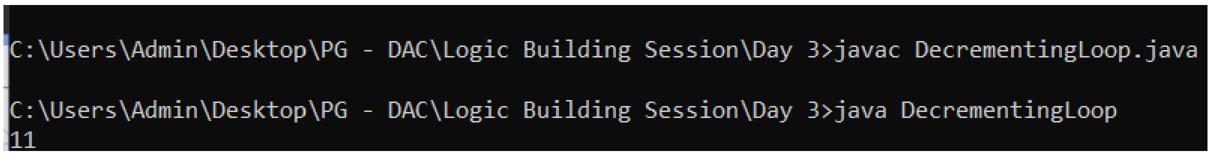
```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac NestedLoopOutput.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java NestedLoopOutput
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.



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac DecrementingLoop.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java DecrementingLoop
11
```

Dry Run:

1. **Initialization:**
   - total = 0
   - i = 5 (loop starts)
2. **First iteration (i = 5):**
   - total += i → total = 0 + 5 = 5
   - i != 3, so we don't skip the loop.
   - total -= 1 → total = 5 - 1 = 4
   - **At the end of iteration, total = 4.**
3. **Second iteration (i = 4):**
   - total += i → total = 4 + 4 = 8
   - i != 3, so we don't skip the loop.
   - total -= 1 → total = 8 - 1 = 7
   - **At the end of iteration, total = 7.**
4. **Third iteration (i = 3):**
   - total += i → total = 7 + 3 = 10
   - i == 3, so we hit the continue statement, which skips the rest of the loop for this iteration. Thus, total -= 1 is not executed.
   - **At the end of iteration, total = 10.**
5. **Fourth iteration (i = 2):**
   - total += i → total = 10 + 2 = 12
   - i != 3, so we don't skip the loop.
   - total -= 1 → total = 12 - 1 = 11
   - **At the end of iteration, total = 11.**
6. **Fifth iteration (i = 1):**
   - total += i → total = 11 + 1 = 12

   - i != 3, so we don't skip the loop.
   - total -= 1 → total = 12 - 1 = 11
   - **At the end of iteration, total = 11.**

7. **Loop ends**, as i becomes 0 and the condition i > 0 is no longer satisfied.

Output:

The value of total is 11 after all the iterations.

****

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:

1. **Initialization:**
   - **count = 0**
   - **Condition count < 5 is true, so the loop starts.**
2. **First iteration (count = 0):**

   Print 0 (current value of count).

   Increment count → count = 1.

   Condition count == 3 is false, so continue.
3. **Second iteration (count = 1):**

Print 1.

Increment count → count = 2.

Condition count == 3 is false, so continue.

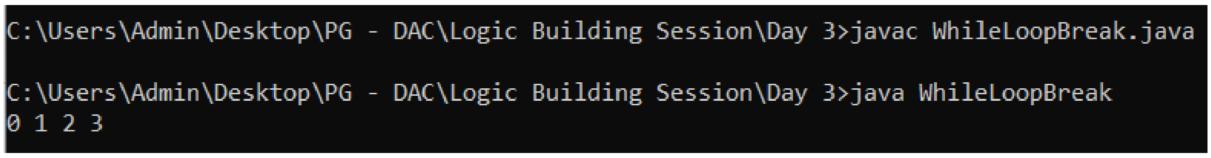
4. **Third iteration (count = 2):**

Print 2.

Increment count → count = 3.

Condition count == 3 is true, so break the loop.

5. **After the loop ends, print count (which is now 3).**



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac WhileLoopBreak.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java WhileLoopBreak
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:

1. **Initialization:**

i = 1

2. **First iteration (i = 1):**

Print 1

Increment i → i = 2

3. **Second iteration (i = 2):**

   Print 2.

   Increment i → i = 3

4. **Third iteration (i = 3):**

   Print 3.

   Increment i → i = 4.

5. **Fourth iteration (i = 4):**

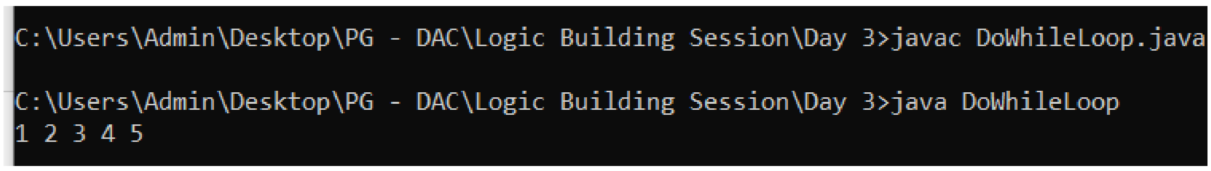
   Print 4.

   Increment i → i = 5.

6. **Condition i < 5 is false now, so the loop ends.**

   Print i (which is 5).

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac DoWhileLoop.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java DoWhileLoop
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:

1. **Initialization:**

   num = 1

2. **First iteration (i = 1):**

   i % 2 != 0, so num -= i → num = 1 - 1 = 0.

3. **Second iteration (i = 2):**

   i % 2 == 0, so num += i → num = 0 + 2 = 2.

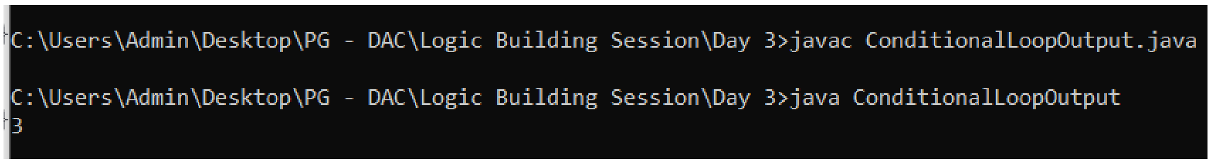
4. **Third iteration (i = 3):**

   i % 2 != 0, so num -= i → num = 2 - 3 = -1.

5. **Fourth iteration (i = 4):**

   i % 2 == 0, so num += i → num = -1 + 4 = 3.

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac ConditionalLoopOutput.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java ConditionalLoopOutput
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:

1. **Initialization:**

   x = 5

2. **First operation: ++x:**

   x is incremented first → x = 6.

   Use 6 in the expression.

3. **Second operation: x--:**

   Use x = 6 in the expression.

   Then x is decremented → x = 5.

4. **Third operation: --x:**

   x is decremented first → x = 4.

   Use 4 in the expression.

5. **Fourth operation: x++:**

   Use x = 4 in the expression.

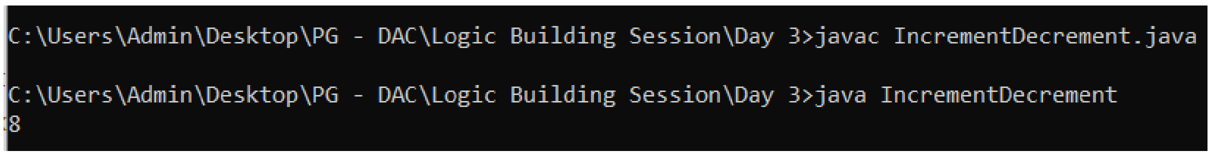
   Then x is incremented → x = 5.

6. **Expression evaluation:**

   y = 6 - 6 + 4 + 4

   y = 8

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac IncrementDecrement.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java IncrementDecrement
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.

Dry Run:

1. **Initialization:**

   a = 10, b = 5

2. **First operation: ++a:**

   a is incremented first → a = 11.

   Use 11 in the expression.

3. **Second operation: b--:**

   Use b = 5 in the expression.

   Then b is decremented → b = 4.

4. **Third operation: --a:**

   a is decremented first → a = 10.

   Use 10 in the expression.

5. **Fourth operation: b++:**

   Use b = 4 in the expression.

   Then b is incremented → b = 5.

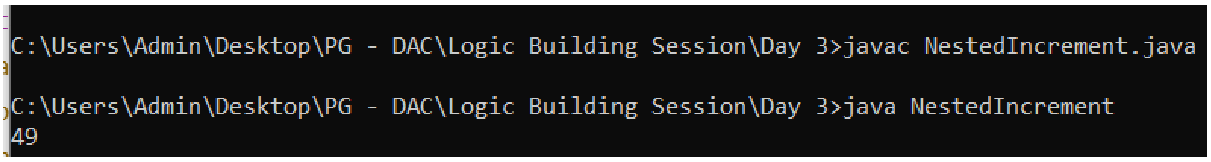
6. **Expression evaluation:**

   result = 11 * 5 - 10 + 4

   result = 55 - 10 + 4

   result = 49

O/P:



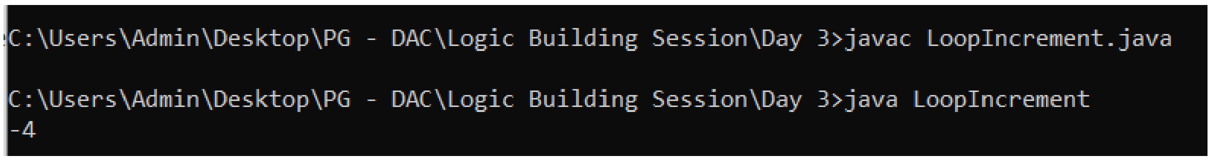
```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac NestedIncrement.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java NestedIncrement
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.

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>javac LoopIncrement.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3>java LoopIncrement
-4
```

Dry Run:

1. **Initialization:**
   - count = 0
2. **First iteration (i = 0):**
   - i++ → i = 0, then i = 1.
   - ++i → i = 2.
   - count += 0 - 2 → count = 0 - 2 = -2.
3. **Second iteration (i = 2):**
   - i++ → i = 2, then i = 3.
   - ++i → i = 4.
   - count += 2 - 4 → count = -2 + (-2) = -4.
4. **Third iteration (i = 4):**
   - i++ → i = 4, then i = 5.

- ++i → i = 6.
- count += 4 - 6 → count = -4 + (-2) = -6.

5. **Fourth iteration (i = 6):**
    - i++ → i = 6, then i = 7.
    - ++i → i = 8.
    - count += 6 - 8 → count = -6 + (-2) = -8.

SECTION 3: Lamborghini Exercise:

Instructions:

1. Complete Each Program: Write a Java program for each of the tasks listed below.

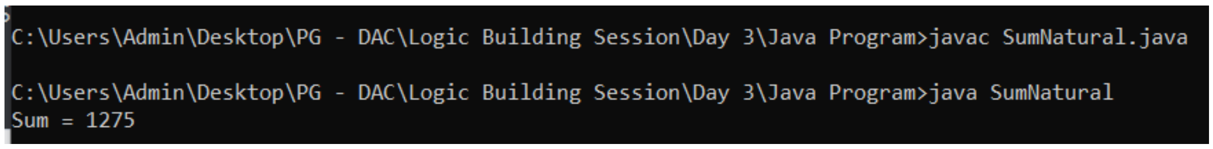
2. Test Your Code: Make sure your code runs correctly and produces the expected output.

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

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

```
 public class SumNatural {


        public static void main(String[] args) {

            int num = 50, sum = 0;

            for(int i = 1; i <= num; ++i)
            {
                // sum = sum + i;
                sum += i;
            }

            System.out.println("Sum = " + sum);
        }
    }
```

O/P:



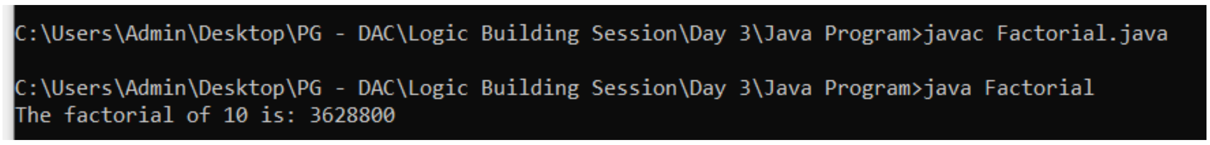
```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac SumNatural.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java SumNatural
Sum = 1275
```

****

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

```
public class Factorial {

   public static void main(String[] args) {

      int number = 10;

      int factorial = 1;


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

O/P:



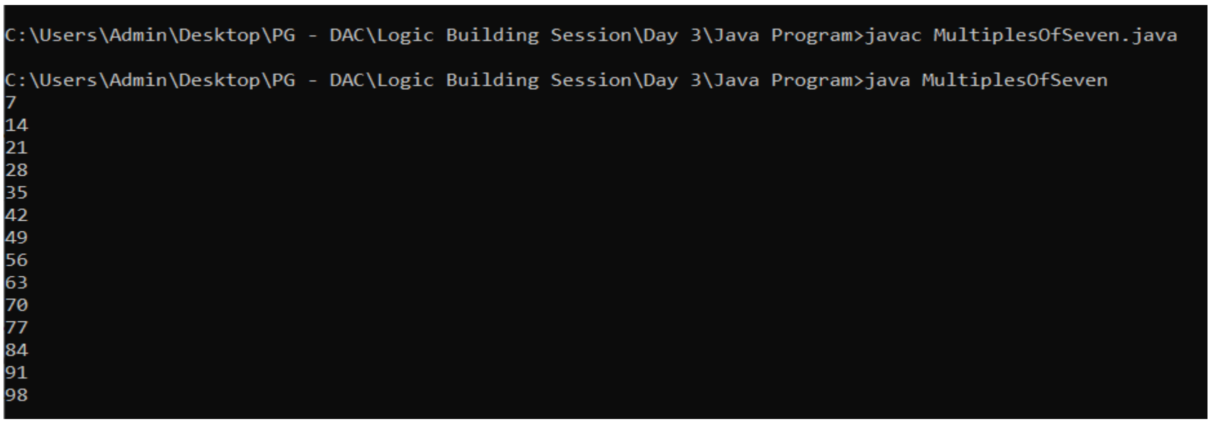
```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac Factorial.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java Factorial
The factorial of 10 is: 3628800
```

****

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

```
public class MultiplesOfSeven {
    public static void main(String[] args) {
        // Loop 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) {
                System.out.println(i);  // Print the number
            }
        }  } }
```

O/P:

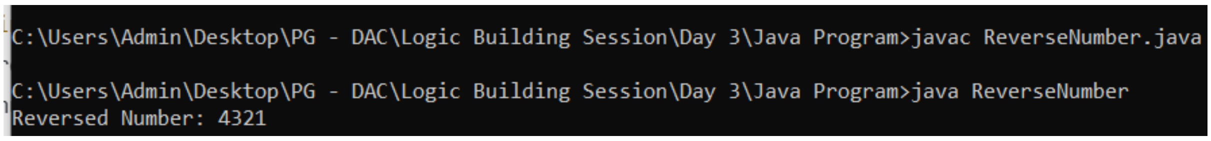


```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac MultiplesOfSeven.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java MultiplesOfSeven
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.**

```
public class ReverseNumber {

public static void main(String[] args) {

    int number = 1234;  // Original number

    int reversed = 0;   // Variable to store the reversed number


   // Loop until the number becomes 0

    while (number != 0) {

        int digit = number % 10;  // Get the last digit of the number

        reversed = reversed * 10 + digit;  // Build the reversed number

        number = number / 10;  // Remove the last digit from the number

    }


    // Print the reversed number

    System.out.println("Reversed Number: " + reversed);

}

}
```

O/P:



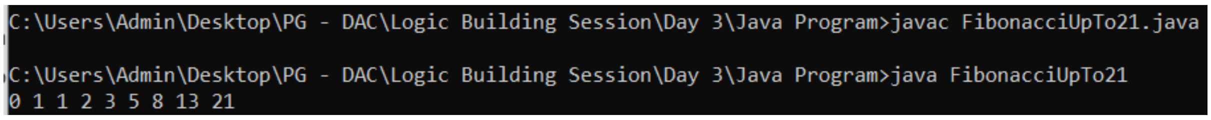
```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac ReverseNumber.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java ReverseNumber
Reversed Number: 4321
```

****

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

```
public class FibonacciUpTo21 {
    public static void main(String[] args) {
        int a = 0, b = 1;
        while (a <= 21) {
            System.out.print(a + " ");
            int next = a + b;
            a = b;
            b = next;
        }
    }
}
```

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac FibonacciUpTo21.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java FibonacciUpTo21
0 1 1 2 3 5 8 13 21
```

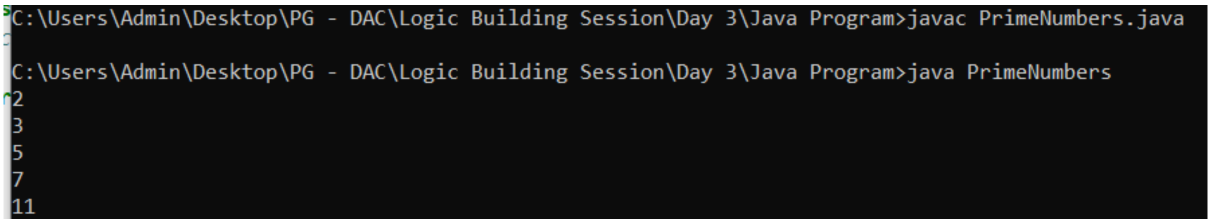
****

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

```
public class PrimeNumbers {

    // Function to check if a number is prime or not

    public static boolean isPrime(int N) {

        for (int i = 2; i < N; i++) {

            if (N % i == 0) {

                return false;

            }

        }

        return true;

    }


    // Main function

    public static void main(String[] args) {

        // Variable to store the number of primes printed so far

        int cnt = 0;

        // Variable to store the number to be checked for prime

        int num = 2;


        // Iterate until we have printed the first 10 primes

        while (cnt < 5) {

            // Prime Check

            if (isPrime(num)) {
```

```
            System.out.println(num);

            cnt++;

          }

          num++;

        }

    }

}
```

O/P:



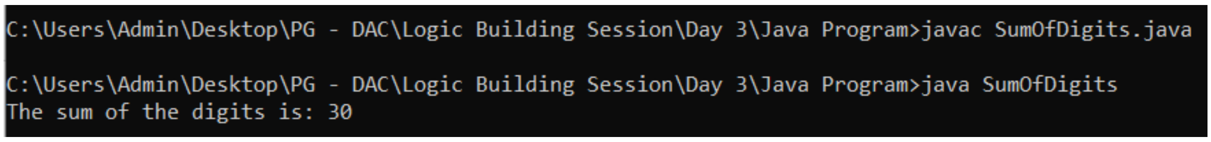
```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac PrimeNumbers.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java PrimeNumbers
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).**

```
public class SumOfDigits {

    public static void main(String[] args) {

        int number = 9876;

        int sum = 0;


        // Loop to extract each digit and add it to sum

        while (number != 0) {

            sum += number % 10;  // Get the last digit

            number /= 10;        // Remove the last digit

        }


        // Output the sum of digits

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

    }

}
```

O/P:



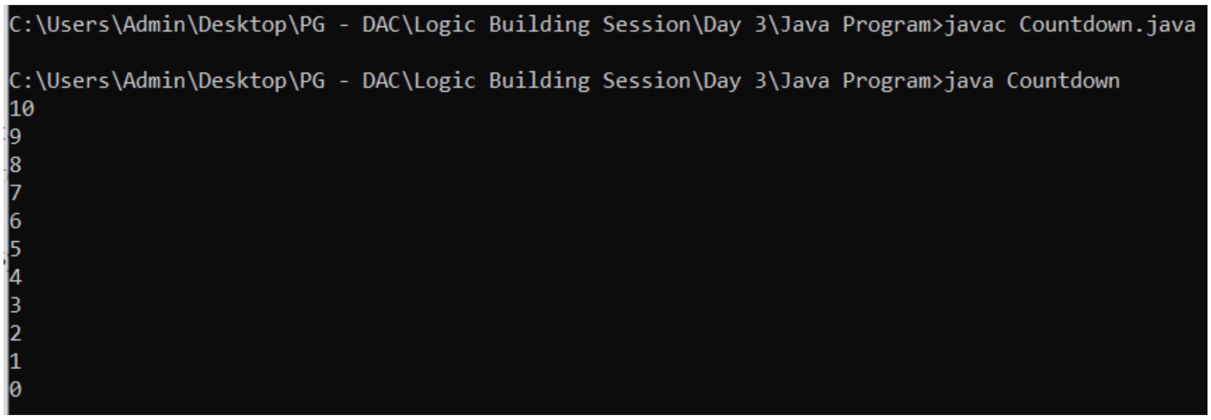
```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac SumOfDigits.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java SumOfDigits
The sum of the digits is: 30
```

****

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

```
public class Countdown {
    public static void main(String[] args) {
        // Start counting down from 10
        for (int i = 10; i >= 0; i--) {
            System.out.println(i);
        }
    }
}
```

O/P:



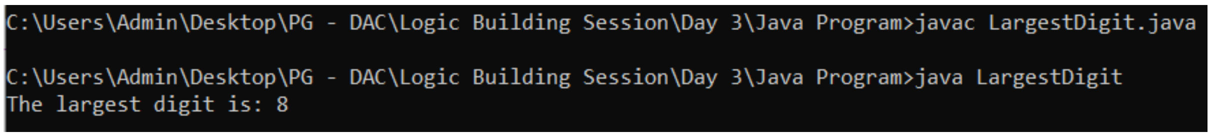
```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac Countdown.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java Countdown
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.

```
public class LargestDigit {
    public static void main(String[] args) {
        int number = 4825;
        int largestDigit = 0;

        // Loop to extract each digit and find the largest
        while (number != 0) {
            int digit = number % 10;  // Get the last digit
            if (digit > largestDigit) {
                largestDigit = digit;  // Update largest digit if current digit is greater
            }
            number /= 10;  // Remove the last digit
        }

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

O/P:



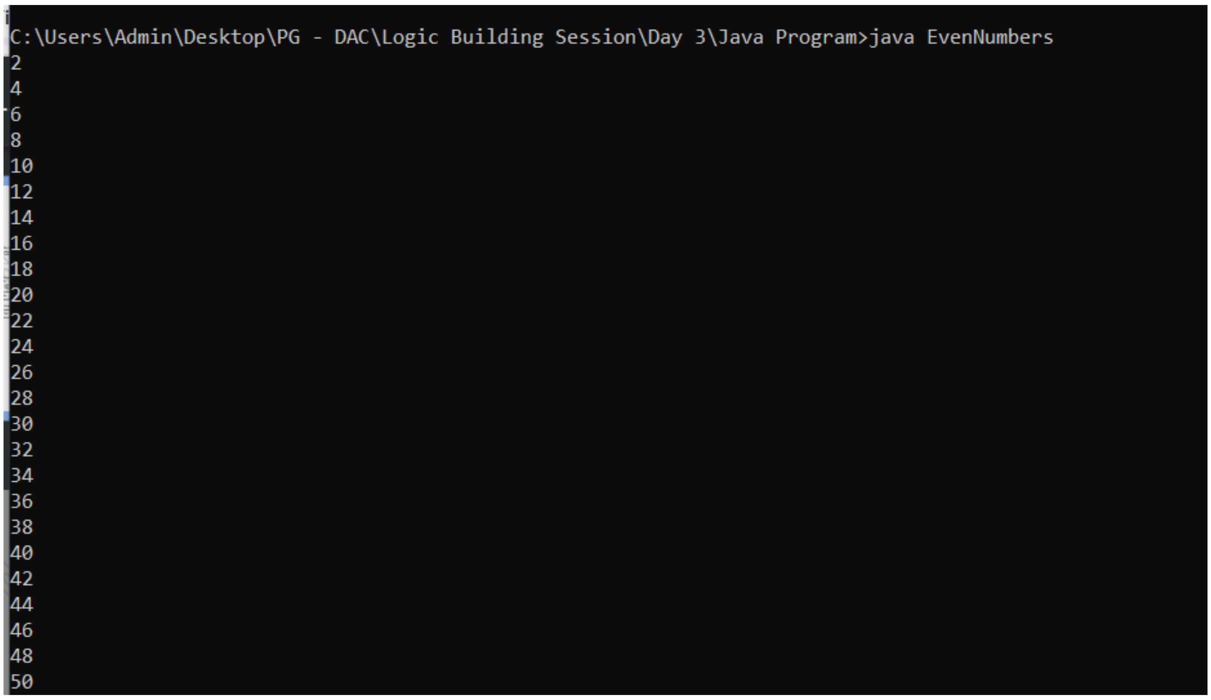
```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac LargestDigit.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java LargestDigit
The largest digit is: 8
```

****

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

```
public class EvenNumbers {
    public static void main(String[] args) {
        // Loop 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);
            }
        }
    }
}
```

O/P:



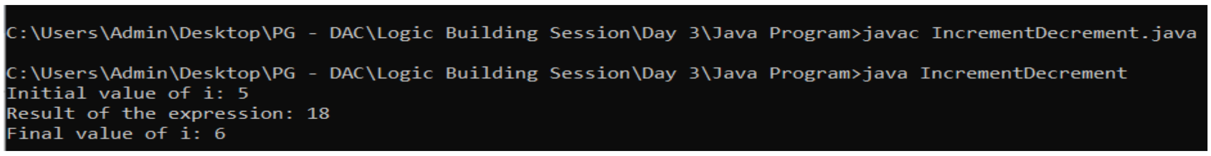
```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java EvenNumbers
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
```

****

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

```
public class IncrementDecrement {
   public static void main(String[] args) {
      int i = 5;

      // Using both pre-increment and post-decrement operators in a single expression
      int result = ++i + i-- + ++i;

      // Output the results
      System.out.println("Initial value of i: 5");
      System.out.println("Result of the expression: " + result);
      System.out.println("Final value of i: " + i);
   }
}
```

O/P:



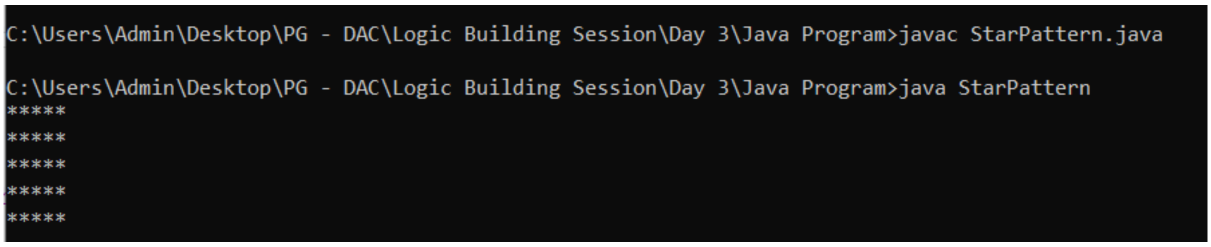
```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac IncrementDecrement.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java IncrementDecrement
Initial value of i: 5
Result of the expression: 18
Final value of i: 6
```

12. Write a program to draw the following pattern:

```
*****

*****

*****

*****

*****
```

```
public class StarPattern {
    public static void main(String[] args) {
        // Loop through 5 rows
        for (int i = 0; i < 5; i++) {
            // Loop through 5 columns
            for (int j = 0; j < 5; j++) {
                System.out.print("*"); // Print a star
            }
            System.out.println(); // Move to the next line after printing stars in one row
        }
    }
}
```

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac StarPattern.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java StarPattern
*****
*****
*****
*****
*****
```

****

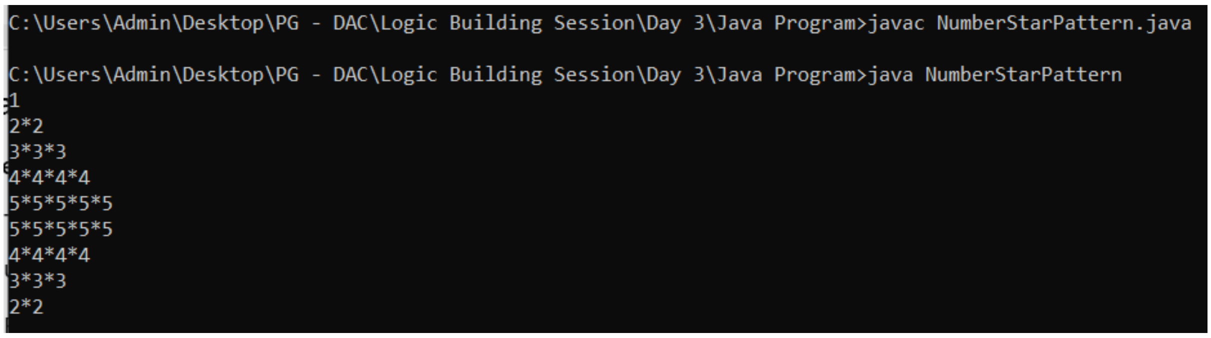
13. Write a program to print the following pattern:

```
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
```

Code:

```
public class NumberPattern {
    public static void main(String[] args) {
        int n = 5; // This is the maximum number, can be adjusted to any value
        // Print the top part of the pattern (including the middle line)
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print(i + "*");
            }
            System.out.println();
        }

        // Print the bottom part of the pattern
        for (int i = n; i >= 1; i--) {
            for (int j = 1; j < i; j++) {
                System.out.print(i + "*");
            }
            System.out.println();
        }
    }
}
```

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac NumberStarPattern.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java NumberStarPattern
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
```

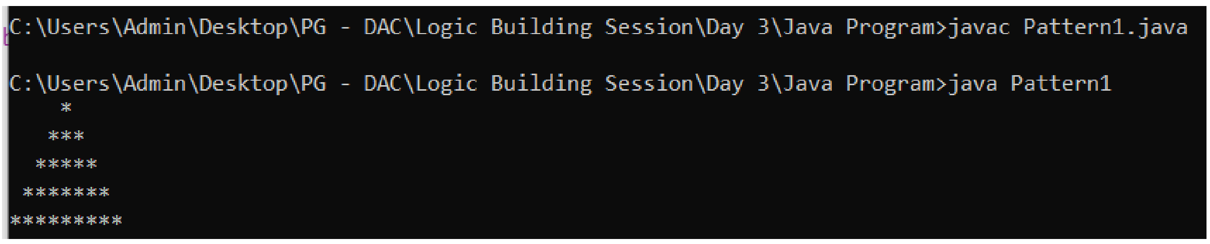
14. Write a program to print the following pattern:

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

Code:

```
public class Pattern1 {
   public static void main(String[] args) {
      int rows = 5;  // You can adjust the number of rows here

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

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac Pattern1.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java Pattern1
    *
   ***
  *****
 *******
*********
```

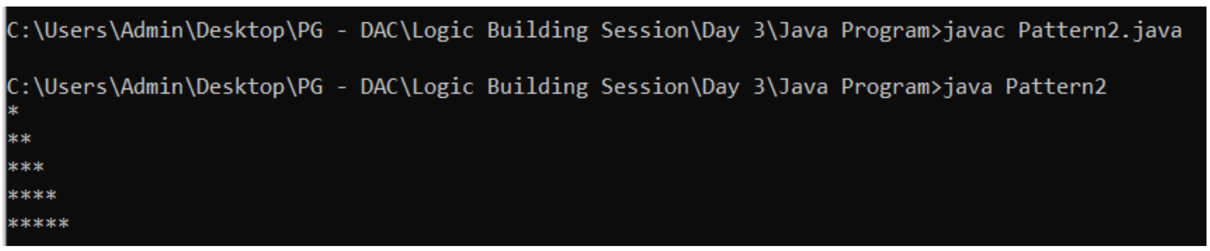
15. Write a program to print the following pattern:

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

Code:

```
public class Pattern2 {
    public static void main(String[] args) {
        int rows = 5;  // You can adjust the number of rows here

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

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac Pattern2.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java Pattern2
*
**
***
****
*****
```

****

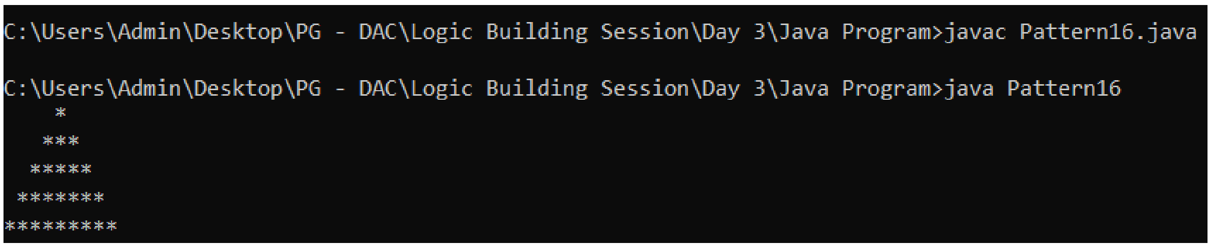
16. Write a program to print the following pattern:

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

Code:

```
public class Pattern16 {
    public static void main(String[] args) {
        int n = 5; // The number of rows

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

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

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

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac Pattern16.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java Pattern16
    *
   ***
  *****
 *******
*********
```

****

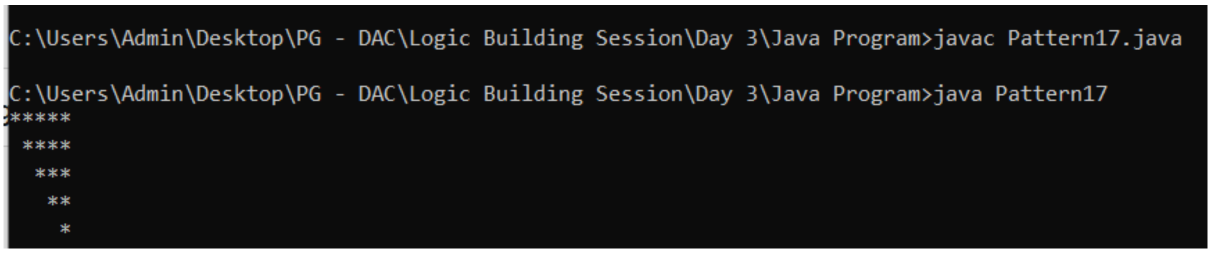
17. Write a program to print the following pattern:

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

Code:

```
public class Pattern17 {
    public static void main(String[] args) {
        int n = 5; // The number of rows

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

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

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

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac Pattern17.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java Pattern17
*****
 ****
  ***
   **
    *
```

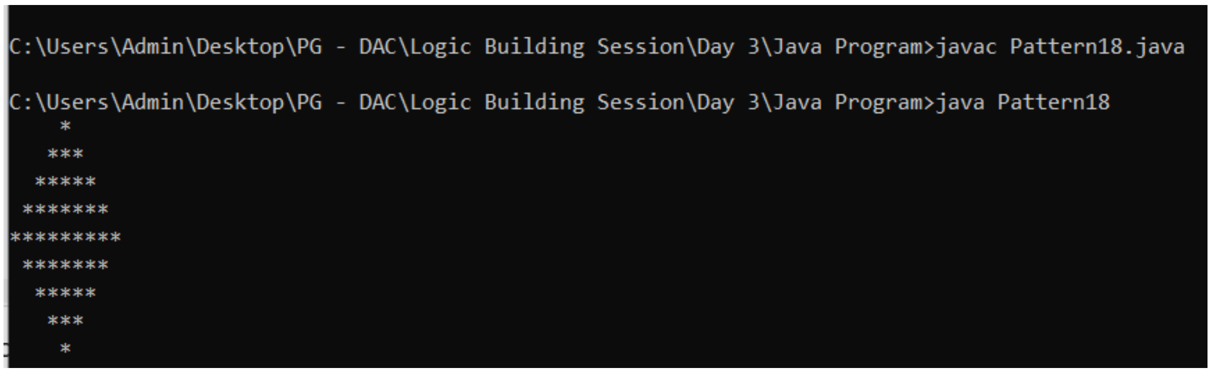
18. Write a program to print the following pattern:

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

Code:

```
public class Pattern18 {
  public static void main(String[] args) {
    int n = 5; // The number of rows for the upper half of the diamond

    // Upper half of the diamond
    for (int i = 1; i <= n; i++) {
      // Print spaces
      for (int j = i; j < n; j++) {
        System.out.print(" ");
      }

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

      // Move to the next line after each row
      System.out.println();
    }

    // Lower half of the diamond
    for (int i = n - 1; i >= 1; i--) {
            // Print spaces
            for (int j = n; 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 after each row
            System.out.println();
        }
    }
}
```

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac Pattern18.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java Pattern18
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

****

19. Write a program to print the following pattern:

1

1*2

1*2*3

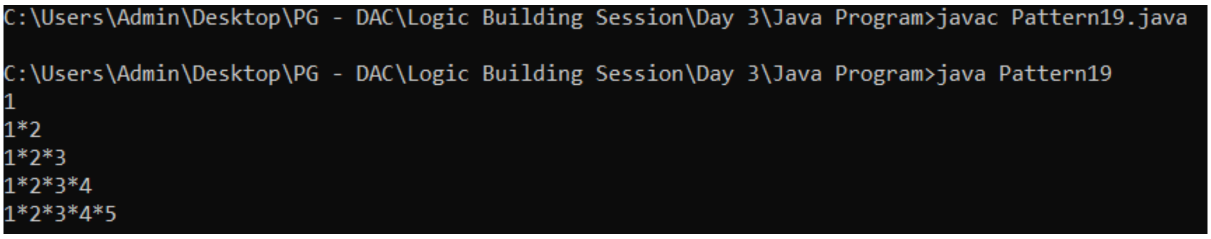
1*2*3*4

1*2*3*4*5

Code:

```
public class Pattern1 {
    public static void main(String[] args) {
        int n = 5; // number of rows
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print(j);
                if (j < i) {
                    System.out.print("*");
                }
            }
            System.out.println();
        }
    }
}
```

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac Pattern19.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java Pattern19
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

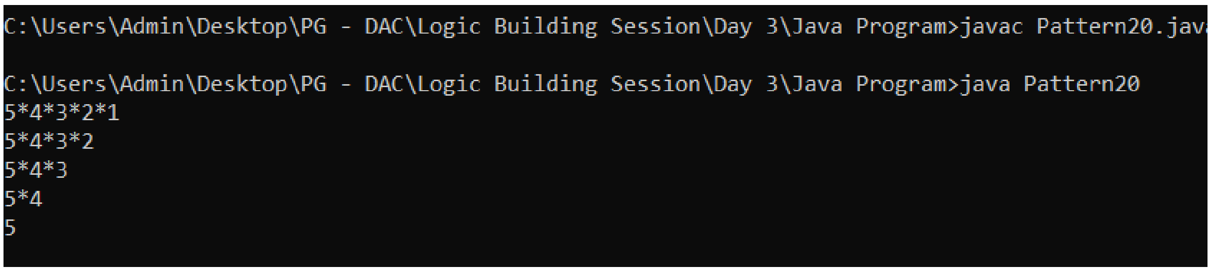
5*4*3*2

5*4*3*2*1

Code:

```
public class Pattern20 {
    public static void main(String[] args) {
        int n = 5; // number of rows
        for (int i = 0; i < n; i++) {
            for (int j = n; j > i; j--) {
                System.out.print(j);
                if (j > i + 1) {
                    System.out.print("*");
                }
            }
            System.out.println();
        }
    }
}
```

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac Pattern20.jav

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java Pattern20
5*4*3*2*1
5*4*3*2
5*4*3
5*4
5
```

****

21. Write a program to print the following pattern:

1

1*3

1*3*5

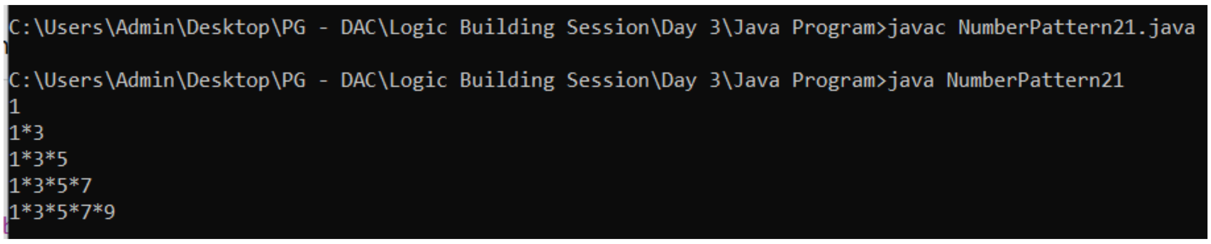
1*3*5*7

1*3*5*7*9

Code:

```
public class NumberPattern21 {
    public static void main(String[] args) {
        int rows = 5; // Number of rows
        int number = 1; // Starting number

        // Outer loop for rows
        for (int i = 1; i <= rows; i++) {
            // Inner loop to print numbers and stars
            for (int j = 1; j <= i; j++) {
                System.out.print(number); // Print the current number
                if (j < i) {
                    System.out.print("*"); // Print '*' if not the last number in the row
                }
                number += 2; // Increment number by 2
            }
            System.out.println(); // Move to the next line after each row
            number = 1; // Reset the starting number for each row
        }
    }
}
```

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac NumberPattern21.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java NumberPattern21
1
1*3
1*3*5
1*3*5*7
1*3*5*7*9
```

****

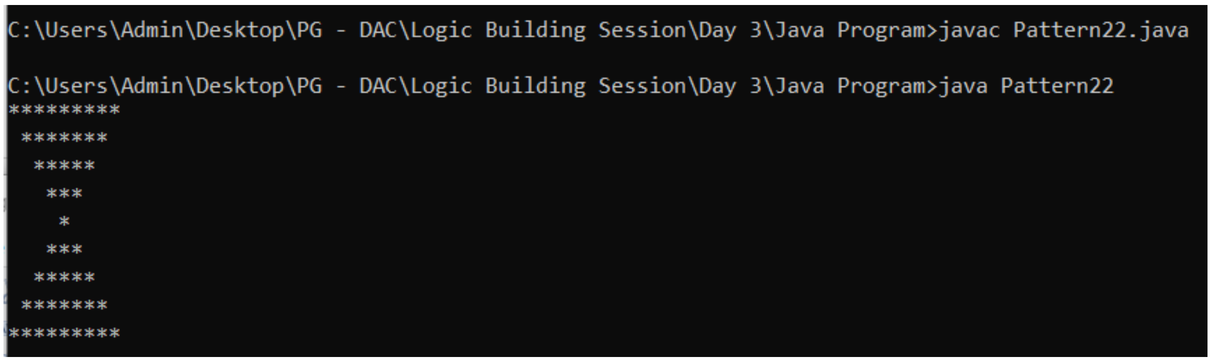
22. Write a program to print the following pattern:

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

Code:

```
public class Pattern22 {
    public static void main(String[] args) {
        int rows = 5; // Number of rows for the middle part

        // Upper half of the pattern
        for (int i = 1; i <= rows; i++) {
            // Printing spaces
            for (int j = 1; j <= i - 1; j++) {
                System.out.print(" ");
            }
            // Printing stars
            for (int k = 1; k <= 2 * (rows - i) + 1; k++) {
                System.out.print("*");
            }
            System.out.println();
        }

        // Lower half of the pattern
        for (int i = rows - 1; i >= 1; i--) {
            // Printing spaces
            for (int j = 1; j <= i - 1; j++) {
                System.out.print(" ");
            }
            // Printing stars
            for (int k = 1; k <= 2 * (rows - i) + 1; k++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac Pattern22.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java Pattern22
*********
 *******
  *****
   ***
    *
   ***
  *****
 *******
*********
```

****

23. Write a program to print the following pattern:

11111

22222

33333

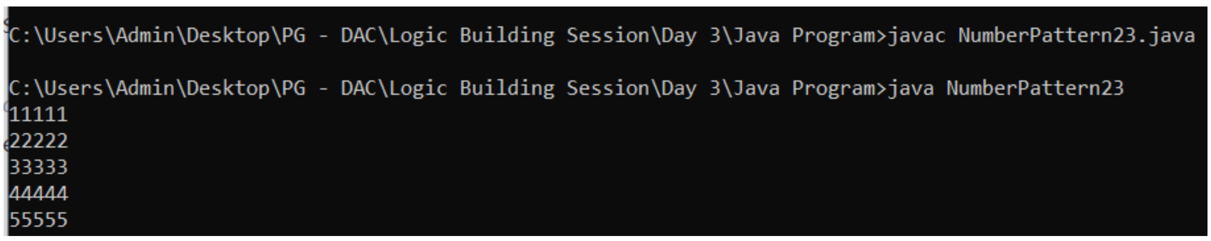
44444

55555

Code:

```
public class NumberPattern23 {
    public static void main(String[] args) {
        // Loop through rows
        for (int i = 1; i <= 5; i++) {
            // Loop through columns
            for (int j = 1; j <= 5; j++) {
                System.out.print(i);  // Print the current row number
            }
            // Move to the next line after each row
            System.out.println();
        }
    }
}
```

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac NumberPattern23.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java NumberPattern23
11111
22222
33333
44444
55555
```

****

24. Write a program to print the following pattern:

1

22

333

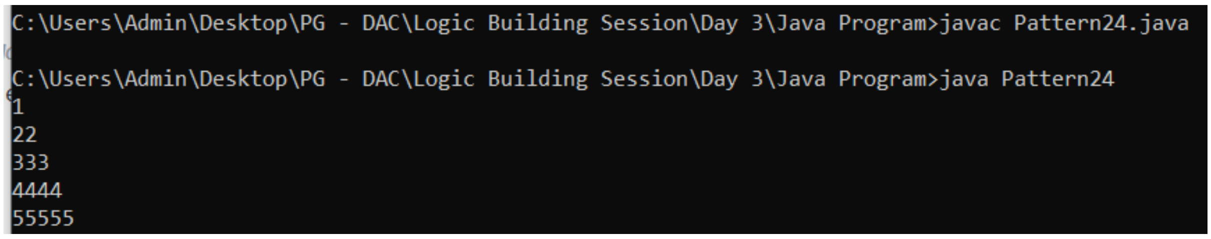
4444

55555

Code:

```
public class Pattern24 {
    public static void main(String[] args) {
        // Loop through rows
        for (int i = 1; i <= 5; i++) {
            // Loop through columns for each row
            for (int j = 1; j <= i; j++) {
                // Print the number i
                System.out.print(i);
            }
            // Move to the next line after each row
            System.out.println();
        }
    }
}
```

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac Pattern24.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java Pattern24
1
22
333
4444
55555
```

****

25. Write a program to print the following pattern:

1

12

123

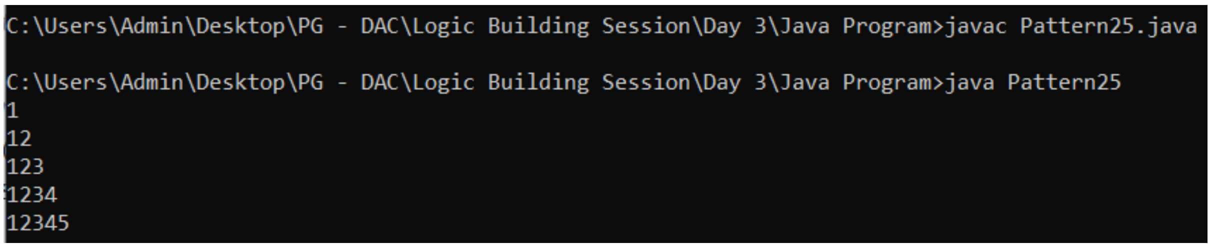
1234

12345

Code:

```
public class Pattern25 {
    public static void main(String[] args) {
        // Loop through rows
        for (int i = 1; i <= 5; i++) {
            // Print numbers from 1 to i in each row
            for (int j = 1; j <= i; j++) {
                System.out.print(j);
            }
            // Move to the next line after each row
            System.out.println();
        }
    }
}
```

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac Pattern25.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java Pattern25
1
12
123
1234
12345
```

****

26. Write a program to print the following pattern:

1

2 3

4 5 6

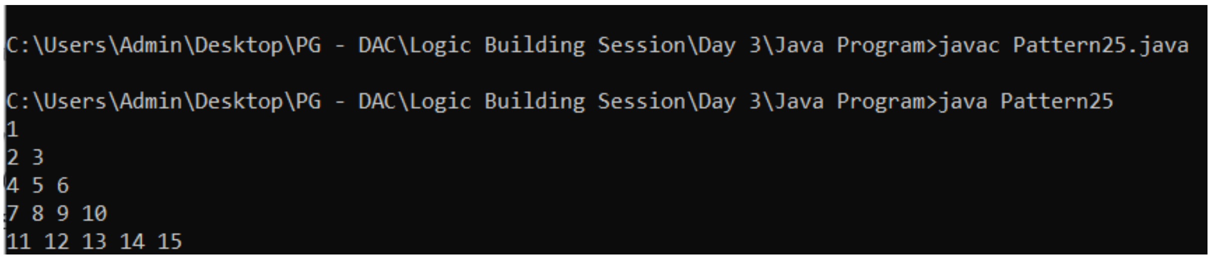
7 8 9 10

11 12 13 14 15

Code:

```
public class Pattern26 {
    public static void main(String[] args) {
        int num = 1;  // Start the number from 1

        // Loop through rows
        for (int i = 1; i <= 5; i++) {
            // Print numbers in each row
            for (int j = 1; j <= i; j++) {
                System.out.print(num + " ");
                num++;  // Increment the number after each print
            }
            // Move to the next line after each row
            System.out.println();
        }
    }
}
```

O/P:



```
C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>javac Pattern25.java

C:\Users\Admin\Desktop\PG - DAC\Logic Building Session\Day 3\Java Program>java Pattern25
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```