**ASSINGMENT -3**

**SHILPA PANDEY-JH**

**Snippet 1**:

public class InfiniteForLoop {

public static void main(String[] args) {

for (int i = 0; i < 10; i--) {

System.out.println(i);

}

}

}

**Error**: Could not find or load main class InfiniteForLoopp

Caused by: java.lang.ClassNotFoundException: InfiniteForLoopp

**Explanation:**

The reason this for loop runs infinitely is because of the way the loop control variable i is being manipulated.

**Corrected**:

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?

**Answer**:

1. In each iteration, the loop decrements i, making it more negative.

2. As i continues to decrease (go negative), it will **never** reach a value greater than or equal to 10, so the condition i < 10 will always be true.

3. This results in the loop continuing to run forever.

To prevent the loop from running infinitely, the loop control variable should be **incremented** rather than decremented. This ensures that i increases and eventually breaks the loop when i reaches 10.

**Snippet 2:**

public class IncorrectWhileCondition {

public static void main(String[] args) {

int count = 5; while (count = 0) {

System.out.println(count); count--;

}

}

}

**Error:**

error: incompatible types: int cannot be converted to boolean

while (count = 0) {

^

1 error

**Explanation:**

The issue in the code is with the **condition** in the while loop:

while (count = 0)

The = operator in Java is an **assignment** operator, not a comparison operator. When count = 0 is executed, it **assigns** the value 0 to count and then evaluates the condition.

**Corrected code:**

public class IncorrectWhileCondition {

public static void main(String[] args) {

int count = 5;

while (count != 0) {

System.out.println(count);

count--; } } }

**Snippet 3:**

public class DoWhileIncorrectCondition {

public static void main(String[] args) {

int num = 0;

do {

System.out.println(num);

num++;

}

while (num > 0);

} }

**Errors or Unexpected Behavior**:

1.Loop Continues.

2.The issue with the do-while loop in the code is related to the **loop condition**:

while (num > 0);

3.The loop body runs **once** because the condition num > 0 is checked **after** the first iteration.

* In the first iteration, num starts at 0, so it prints 0 and then increments num to 1.
* The condition num > 0 is then evaluated after the first iteration and it **evaluates to true** (since num is now 1), so the loop continues.

**Corrected code**:

public class DoWhileIncorrectCondition {

public static void main(String[] args) {

int num = 0; do {

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

} while (num < 5); }}

**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?

**Explanation:**

The issue in the loop lies in the **loop condition**:

for (int i = 1; i <= 10; i++) {

System.out.println(i);

}

**Issue:**

The expected output is to print **only numbers from 1 to 9**, but the loop currently prints **1 to 10** because the condition is set to i <= 10, which allows the loop to run until i reaches 10.

**Corrected code:**

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); }

}

}

* By changing the condition to i < 10, the loop will now run **while i is less than 10**, thus printing only numbers from 1 to 9.
* The loop will stop when i reaches 10 because the condition i < 10 will evaluate to **false** when i equals 10.

Now the program will print the numbers 1, 2, 3, 4, 5, 6, 7, 8, 9 as expected.

**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?

**Explanation:**

The issue with the for loop in the code lies in how the initialization, loop condition, and update statements are set up:

for (int i = 10; i >= 0; i++) {

System.out.println(i);

}

**The problem:**

1.The **update statement** i++ increments i, which causes the loop to count **upwards**, rather than counting **downwards** from 10 to 0.

2.Since the loop condition is i >= 0, as i gets larger (due to i++), the loop will eventually exit because i will no longer satisfy i >= 0.

3.Therefore, the loop will never print the numbers in the expected order (from 10 down to 0).

>>To make the loop print numbers in descending order,you need to decrement I after each iteration,which can be done by changing the update statement to i-- instead of i++

**Corrected code:**

public class WrongInitializationForLoop {

public static void main(String[] args) {

for (int i = 10; i >= 0; i--) {

// Change i++ to i-- to count downwards

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");

}

}

**Output:**

C:\Users\HP\Desktop\java>java MisplacedForLoopBody

0

1

2

3

4

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?

**Explanation :**

* The line System.out.println(i); is inside the for loop because it is not enclosed in braces {}. This means that only **the first statement** after the for loop declaration (i.e., System.out.println(i);) is part of the loop.
* The second statement System.out.println("Done"); is **not part of the for loop** because it is outside the braces. As a result, it only executes once **after** the loop finishes, and that’s why "Done" is printed only once after the loop has completed.

>>To include both System.out.println(i); and System.out.println("Done"); within the loop, we need to **enclose the loop body with braces** {}.

**Corrected code:**

public class MisplacedForLoopBody {

public static void main(String[] args) {

for (int i = 0; i < 5; i++) {

System.out.println(i);

System.out.println("Done");

}

}

}

**Output:**

C:\Users\HP\Desktop\java>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++;

}

}

}

**Error to investigate**: Why does this code produce a compilation error? What needs to be done to initialize the loop variable properly?

**Explanation**:

The error in the code occurs because the variable count is **declared but not initialized** before it is used in the while loop condition.

We need to **initialize** count with a value before it is used in the loop. Typically, if we are starting a counter loop, will initialize it to a value that makes sense for the loop (such as 0).

**Corrected code:**

public class UninitializedWhileLoop {

public static void main(String[] args) {

int count = 0;

// Initialize the count variable 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?

**Explanation:**

The issue with the do-while loop in the code is that the loop is **decrementing** the variable num (num--), which causes the loop to print **unexpected numbers**

To print the numbers from 1 to 5, we need to increment num instead of decrementing it, and adjust the loop condition accordingly to ensure it prints exactly the numbers 1 to 5.

**Corrected code:**

public class OffByOneDoWhileLoop {

public static void main(String[] args) {

int num = 1;

do {

System.out.println(num);

num++; //Increment num to count up

while (num <= 5); }} // Loop condition: stop when num exceeds 5

**Snippet 9:**

public class InfiniteForLoopUpdate {

public static void main(String[] args) {

for (int i = 0; i < 5; i += 2) {

System.out.println(i);

}

}

}

**Output:**

**0**

**2**

**4**

**Error to investigate:** Why does the loop print unexpected results or run infinitely? How should the loop update expression be corrected?

**Explanation:**

Issue with the expected behavior:

* The loop does not run infinitely, but it does print unexpected results because it's skipping over the expected numbers (1, 3).
* This happens because the update expression increments i by 2, causing the loop to only print even numbers and skip odd numbers.

Loop update expression can be corrected by printing all numbers from 0 to 4

,we need to change the update expression to increment I by 1 instead of 2.

**Corrected code**:

public class InfiniteForLoopUpdate {

public static void main(String[] args) {

for (int i = 0; i < 5; i++) {

// Increment i by 1 to print every number

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?

**Explanation:**

The issue in the code is with the loop condition.

while (num = 10) {

System.out.println(num);

num--;

}

>>The condition in the while loop is num = 10.

>> In Java, the **=** operator is the **assignment** operator, not the **comparison** operator. So instead of checking if num is equal to 10, this code **assigns** the value 10 to num on each iteration.

>>The result of the assignment expression num = 10 is 10, which is a **non-zero value** (and in Java, any non-zero value is treated as true in a boolean context).

>> Therefore, the condition num = 10 always evaluates to **true**, causing the loop to execute indefinitely.

Here, we need to **compare** num with 10, which should be done using the **comparison operator ==** instead of the assignment operator =.

**Corrected code:**

public class IncorrectWhileLoopControl {

public static void main(String[] args) {

int num = 10;

while (num >= 0) {

// Use '==' for comparison and adjust the condition

System.out.println(num);

num--; } } }

**Expected output:**

10

9

8

7

6

5

4

3

2

1

0

**Snippet 11:**

public class IncorrectLoopUpdate {

public static void main(String[] args) {

int i = 0;

while (i < 5) {

System.out.println(i);

i += 2;

**}**

**}**

**}**

**Error to investigate:** What will be the output of this loop? How should the loop variable be updated to achieve the desired result?

**Output:**

C:\Users\HP\Desktop\java>java IncorrectLoopUpdate

0

2

4

The error in the code is due to the **incorrect update expression** in the while loop.

**What is wrong with the loop?**

* The loop prints **only** the numbers 0, 2, and 4, skipping over the numbers 1, 3, and not including 5.
* This is because the update expression i += 2 causes i to increment by 2 in each iteration, meaning it skips over odd numbers and doesn't print all numbers from 0 to 4.

If we want to print **every integer** from 0 to 4, the loop variable i should simply be incremented by 1 after each iteration.

**Corrected 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 print every number from 0 to 4

}

}

}

By updating **i++:** Changing i += 2 to i++ ensures that i is incremented by 1 in each iteration, so all numbers from 0 to 4 are printed.

The loop condition i < 5 ensures the loop stops when i reaches 5.

**Expected output**:

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

}

}

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

**Explanation of the issue:**

* Inside the for loop, the variable x is declared and initialized in each iteration.

int x = i \* 2;

* The variable x is **local** to the loop body. This means that x is created and destroyed during each iteration of the loop, and it is **not accessible outside** the loop.
* After the loop finishes executing, x goes out of scope. Therefore, the line:
* System.out.println(x); // Error: 'x' is not accessible here
* causes a compilation error because x no longer exists outside the loop

>> **Scope of variables** in Java defines where a variable can be accessed. A variable declared inside a block (such as a loop or a method) is only accessible within that block, and it ceases to exist once the block is finished executing.

>> In this case, x is declared inside the for loop, so it can only be accessed within the loop. It is out of scope once the loop finishes.

**Corrected code:**

public class LoopVariableScope

public static void main(String[] args) {

int x = 0;

for (int i = 0; i < 5; i++) {

x = i \* 2;

}

System.out.println(x);

}

}

**SECTION 2: Guess the Output**

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

The outer loop runs for i = 1, 2, 3 (from 1 to 3).

Inner loop:

For each value of i, the inner loop runs for j = 1, 2 (from 1 to 2).

Now at the time of iteration:

When i = 1:

Inner loop runs:

* + For j = 1: Prints 1 1
  + For j = 2: Prints 1 2

After the inner loop completes, it moves to the next line (System.out.println()).

When i = 2:

* Inner loop runs:
  + For j = 1: Prints 2 1
  + For j = 2: Prints 2 2
* After the inner loop completes, it moves to the next line.

When i = 3:

* Inner loop runs:
  + For j = 1: Prints 3 1
  + For j = 2: Prints 3 2
* After the inner loop completes, it moves to the next line.

Output:

1 1 1 2

2 1 2 2

3 1 3 2

**Snippet 2:**

public class DecrementingLoop {

public static void main(String[] args) {

int total = 0;

for (int i = 5; i > 0; i--) {

total += i;

if (i == 3) continue;

total -= 1;

} System.out.println(total); } } // Guess the output of this loop.

**Dry Run:**

Initial values:

* total = 0
* i starts from 5 and will decrement until 1.

**Iteration 1 (i = 5):**

1. total += i:
   * total = 0 + 5 = 5
2. The if (i == 3) condition is **false** because i = 5. So, we don't skip to the next iteration.
3. total -= 1:
   * total = 5 - 1 = 4
4. After the first iteration, i becomes 4 (i--).

**Total after Iteration 1:** 4

**Iteration 2 (i = 4):**

1. total += i:
   * total = 4 + 4 = 8
2. The if (i == 3) condition is **false** because i = 4. So, we don't skip to the next iteration.
3. total -= 1:
   * total = 8 - 1 = 7
4. After the second iteration, i becomes 3 (i--).

**Total after Iteration 2:** 7

**Iteration 3 (i = 3):**

1. total += i:
   * total = 7 + 3 = 10
2. The if (i == 3) condition is **true** because i = 3, so we **skip** the rest of the loop and proceed to the next iteration without performing total -= 1.
3. After skipping, i becomes 2 (i--).

**Total after Iteration 3 (skipping total -= 1):** 10

**teration 4 (i = 2):**

1. total += i:
   * total = 10 + 2 = 12
2. The if (i == 3) condition is **false** because i = 2. So, we proceed with the next statement.
3. total -= 1:
   * total = 12 - 1 = 11
4. After the fourth iteration, i becomes 1 (i--).

**Total after Iteration 4:** 11

**Iteration 5 (i = 1):**

1. total += i:
   * total = 11 + 1 = 12
2. The if (i == 3) condition is **false** because i = 1. So, we proceed with the next statement.
3. total -= 1:
   * total = 12 - 1 = 11
4. After the fifth iteration, i becomes 0, and the loop ends (i > 0 condition fails).

* **Total after Iteration 5:** 11

**Output:**

After the loop completes, the final value of total 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.

**Dry Run:**

* First Iteration (count = 0): Prints 0, increments count to 1.
* Second Iteration (count = 1): Prints 1, increments count to 2.
* Third Iteration (count = 2): Prints 2, increments count to 3, then breaks the loop.
* After the loop breaks, the final value of count (which is 3) is printed.

Thus, the final output is:

0 1 2

3

**Snippet 4:**

public class DoWhileLoop {

public static void main(String[] args) {

int i = 1;

do {

System.out.print(i + " ");

i++;

}

while (i < 5);

System.out.println(i); } }

// Guess the output of this do-while loop

**Dry Run:**

* First Iteration (i = 1): Prints 1, increments i to 2.
* Second Iteration (i = 2): Prints 2, increments i to 3.
* Third Iteration (i = 3): Prints 3, increments i to 4.
* Fourth Iteration (i = 4): Prints 4, increments i to 5. The loop exits as i = 5 no longer satisfies the condition.
* After the loop, it prints the final value of i, which is 5.

Thus, the final output is:

1 2 3 4 5

**Snippet 5:**

public class ConditionalLoopOutput {

public static void main(String[] args) {

int num = 1;

for (int i = 1; i <= 4; i++) {

if (i % 2 == 0) {

num += i;

} else

{ num -= i;

}

} System.out.println(num);

} }

// Guess the output of this loop.

**Dry Run:**

* **First Iteration (i = 1)**: num = 1 - 1 = 0 (because i is odd).
* **Second Iteration (i = 2)**: num = 0 + 2 = 2 (because i is even).
* **Third Iteration (i = 3)**: num = 2 - 3 = -1 (because i is odd).
* **Fourth Iteration (i = 4)**: num = -1 + 4 = 3 (because i is even).
* Finally, the program prints the value of num, which is 3.

Thus, the final output is:

3

**Snippet 6:**

public class IncrementDecrement {

public static void main(String[] args) {

int x = 5;

int y = ++x - x-- + --x + x++;

System.out.println(y);

} }

// Guess the output of this code snippet.

**Dry Run**:

* **++x**: x becomes 6, and the expression uses 6.
* **x--**: The expression uses 6, then x becomes 5.
* **--x**: x becomes 4, and the expression uses 4.
* **x++**: The expression uses 4, then x becomes 5.

Final arithmetic:

ini

y = 6 - 6 + 4 + 4 = 8

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.

**Initial Values:**

a = 10

b = 5

The expression is: ++a \* b-- - --a + b++

Let's break this down

1. ++a: a becomes 11, and the expression uses 11.

2. b--: The expression uses 5, then b becomes 4.

3 --a: a becomes 10, and the expression uses 10.

4++: The expression uses 4, then b becomes 5.

**result** = 11 \* 5 - 10 + 4 = 49

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.

**Initial Values:**

count = 0 (initialized before the loop starts).

i = 0 (initialized before the loop starts).

 **First Iteration (i = 0)**:

i++ → 0, ++i → 2.

Expression: 0 - 2 = -2.

count = -2.

**Second Iteration (i = 3)**:

i++ → 3, ++i → 5.

Expression: 3 - 5 = -2.

count = -4.

 **Final Output:**

diff

-4

**SECTION 3: Lamborghini Exercise:**

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

**Code.**

public class SumOfNaturalNumbers {

public static void main(String[] args){

Int n = 50;

Int sum = (n\*(n+1)) / 2;

System.out.println("The sum of the first 50 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.

**Code:**

public class Factorial {

public static void main(String[] args) {

int number = 10;

long factorial = 1;

for (int i = 1; i <= number; i++) {

factorial \*= i;

}

System.out.println("The factorial of " + number + " is: " + factorial);

}

}

**Output:**

The factorial of 10 is: 3,628,800

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

**Code**:

public class MultiplesOfSeven{

public static void main(String[] args){

for(int i=7; i<=100; i+=7){

System.out.println(i);

}

}

}

**Output**:

C:\Users\HP\Desktop\java>java MultiplesOfSeven

7

14

21

28

35

42

49

56

63

70

77

84

91

98

1. Write a program to reverse the digits of the number 1234

**Code:**

public class ReverseNumber{

public static void main(String[] args){

int number = 12345;

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

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

**Code:**

public class FibonacciSequence {

public static void main(String[] args) {

int limit = 21;

int first = 0, second = 1;

System.out.println("Fibonacci sequence up to " + limit + ":");

System.out.print(first + " " + second + " ");

while (true) {

int next = first + second;

If (next > limit) {

break;

}

System.out.print(next + " ");

first = second;

second = next;

}

}

}

**Output:**

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

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

**Code:**

Public class FirstFivePrimes {

Public static void main(String[] args){

int count = 0;

int num = 2;

System.out.println(“First 5 prime number:”)

while(count < 5){

boolean isPrime = true;

for(int i = 2; i \* I <= num; i++){

if(num % i == 0){

isPrime = false;

break;

}

}

If (isPrime){

System.out.print(num + “ ”);

count++;

}

num++;

}

System.out.println();

}

}

**Output:**

First 5 prime numbers:

2 3 5 7 11

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

**Code:**

public class SumOfDigits {

public static void main(String[] args) {

int number = 9876;

int sum = 0;

while (number != 0) {

sum += number % 10;

number = number / 10;

}

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.

public class Countdown {

public static void main(String[] args) {

for (int i = 10; i >= 0; i--) {

System.out.println(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.

**Code:**

public class LargestDigit {

public static void main(String[] args) {

int number = 4825;

int largestDigit = 0;

while (number != 0) {

int digit = number % 10;

if (digit > largestDigit) {

largestDigit = digit;

}

number = number / 10;

}

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.

**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

**Code**:

public class IncrementDecrementDemo {

public static void main(String[] args) {

int a = 5;

int b = 10;

int result = ++a + b-- + --b;

System.out.println("The result of the expression is: " + result); System.out.println("The value of a after the operation: " + a); System.out.println("The value of b after the operation: " + b);

}

}

**Output**:

The result of the expression is: 24

The value of a after the operation: 6

The value of b after the operation: 8

**12.** Write a program to draw the following pattern:

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

**Code:**

public class AsteriskPattern {

public static void main(String[] args) {

for (int i = 0; i < 5; i++) {

for (int j = 0; 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

**Code**:

public class NumberPattern {

public static void main(String[] args) {

for (int i = 1; i <= 5; i++) {

for (int j = 1; j <= i; j++) {

if (j > 1) {

System.out.print("\*");

}

System.out.print(i);

}

System.out.println();

}

for (int i = 5; i >= 1; i--) {

for (int j = 1; j < i; j++) {

System.out.print(i + "\*"); }

System.out.println(i);

}

}

}

**14.** Write a program to print the following pattern:

\*

\*\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

**Code:**

public class StarPattern {

public static void main(String[] args) {

for (int i = 1; i <= 6; i++) {

for (int j = 1; j <= (2 \* i - 1); j++) {

System.out.print("\*");

}

System.out.println();

}

}

}

**15.** Write a program to print the following pattern:

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

**Code:**

public class StarPattern {

public static void main(String[] args) {

for (int i = 1; i <= 5; i++) {

row for (int j = 1; j <= i; j++) {

System.out.print("\*");

}

System.out.println();

}

}

}

**16**. Write a program to print the following pattern:

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

**Code:**

public class StarPattern {

public static void main(String[] args) {

for (int i = 1; i <= 5; i++) {

for (int j = i; j < 5; j++) {

System.out.print(" ");

}

for (int k = 1; k <= (2 \* i - 1); k++) {

System.out.print("\*");

}

System.out.println();

}

}

}

17. Write a program to print the following pattern:

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

**Code:**

public class StarPattern {

public static void main(String[] args) {

for (int i = 5; i >= 1; i--) {

for (int j = 5; j > i; j--) {

System.out.print(" "); }

for (int k = 1; k <= i; k++) {

System.out.print("\*");

}

System.out.println();

}

}

}

18. Write a program to print the following pattern:

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*

\*\*\*

\*

**Code:**

public class StarPattern {

public static void main(String[] args) {

int n = 4;

for (int i = 1; i <= n; i++) {

for (int j = n; j > i; j--) {

System.out.print(" ");

}

for (int k = 1; k <= (2 \* i - 1); k++) {

System.out.print("\*");

}

System.out.println();

}

for (int i = n - 1; i >= 1; i--) {

for (int j = n; j > i; j--) {

System.out.print(" ");

}

}

for (int k = 1; k <= (2 \* i - 1); k++) {

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

**Code**

public class NumberPattern {

public static void main(String[] args) {

for (int i = 1; i <= 5; i++) {

for (int j = i; j < 5; j++) {

System.out.print(" ");

}

for (int k = 1; k <= i; k++) {

if (k > 1)

{

System.out.print("\*");

}

System.out.print(k);

}

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

**Code:**

public class NumberPattern {

public static void main(String[] args) {

for (int i = 5; i >= 1; i--) {

for (int j = 5; j >= i; j--) {

if (j < 5) {

System.out.print("\*");

}

System.out.print(j);

}

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

**Code:**

public class NumberPattern {

public static void main(String[] args) {

for (int i = 1; i <= 5; i++) {

for (int j = 1; j <= i; j++) {

System.out.print((2 \* j - 1));

(2j-1) if (j < i) {

System.out.print("\*");

}

}

System.out.println();

}

}

}

22. Write a program to print the following pattern:

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*

\*\*\*

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

**Code:**

public class StarPattern {

public static void main(String[] args) {

for (int i = 9; i >= 1; i -= 2) {

for (int j = 9; j > i; j -= 2) {

System.out.print(" ");

}

for (int k = 1; k <= i; k++) {

System.out.print("\*");

}

System.out.println();

}

for (int i = 3; i <= 9; i += 2) {

for (int j = 9; j > i; j -= 2) {

System.out.print(" ");

}

for (int k = 1; k <= i; k++) {

System.out.print("\*");

System.out.println();

}

}

}

**23.** Write a program to print the following pattern

11111

22222

33333

44444

55555

**Code**:

public class NumberPattern {

public static void main(String[] args) {

for (int i = 1; i <= 5; i++) {

for (int j = 1; j <= 5; j++) {

System.out.print(i);

}

System.out.println();

}

}

}

**24.**Write a program to print the following pattern:

1

22

333

4444

55555

**Code:**

public class NumberPattern {

public static void main(String[] args) {

for (int i = 1; i <= 5; i++) {

for (int j = 1; j <= i; j++) {

System.out.print(i);

}

System.out.println();

}

}

}

**25.** Write a program to print the following pattern

1

12

123

1234

**Code:**

public class NumberPattern {

public static void main(String[] args) {

for (int i = 1; i <= 4; 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

**Code:**

public class NumberPattern {

public static void main(String[] args) {

int num = 1;

for (int i = 1; i <= 5; i++) {

for (int j = 5; j > i; j--) {

System.out.print(" ");

}

for (int k = 1; k <= i; k++) {

System.out.print(num + " ");

num++;

}

System.out.println();

}

}

}