Name – Prerna Sunil Jadhav **SAP ID - 60004220127**

Experiment No - 02

AIM: TO IMPLEMENT JAVA CONTROL STATEMENTS AND LOOPS

THEORY:

- Java compiler executes the code from top to bottom. The statements in the code are executed according to the order in which they appear.
- Java provides statements that can be used to control the flow of Java code. Such statements are called control flow statements.
- It is one of the fundamental features of Java, which provides a smooth flow of program.
- Java provides three types of control flow statements.
  - Decision Making statements
    - ✓ if statements
    - ✓ switch statement
  - Loop statements
    - ✓ do while loop
    - ✓ while loop
    - ✓ for loop
    - ✓ for-each loop
  - Jump statements
    - ✓ break statement
    - ✓ continue statement

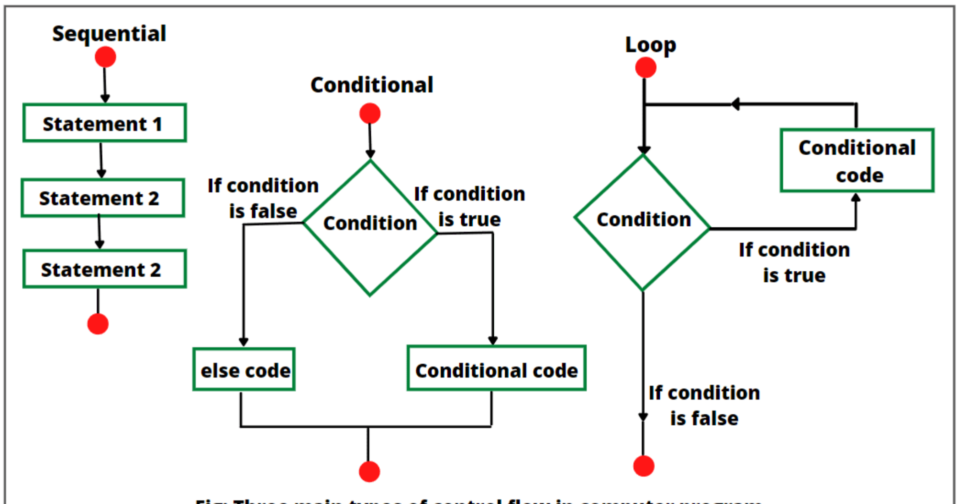


Fig: Three main types of control flow in computer program

<u>PROGRAM 1</u>: Write A Program to find roots of a Quadratic equation. Take care of imaginary values.

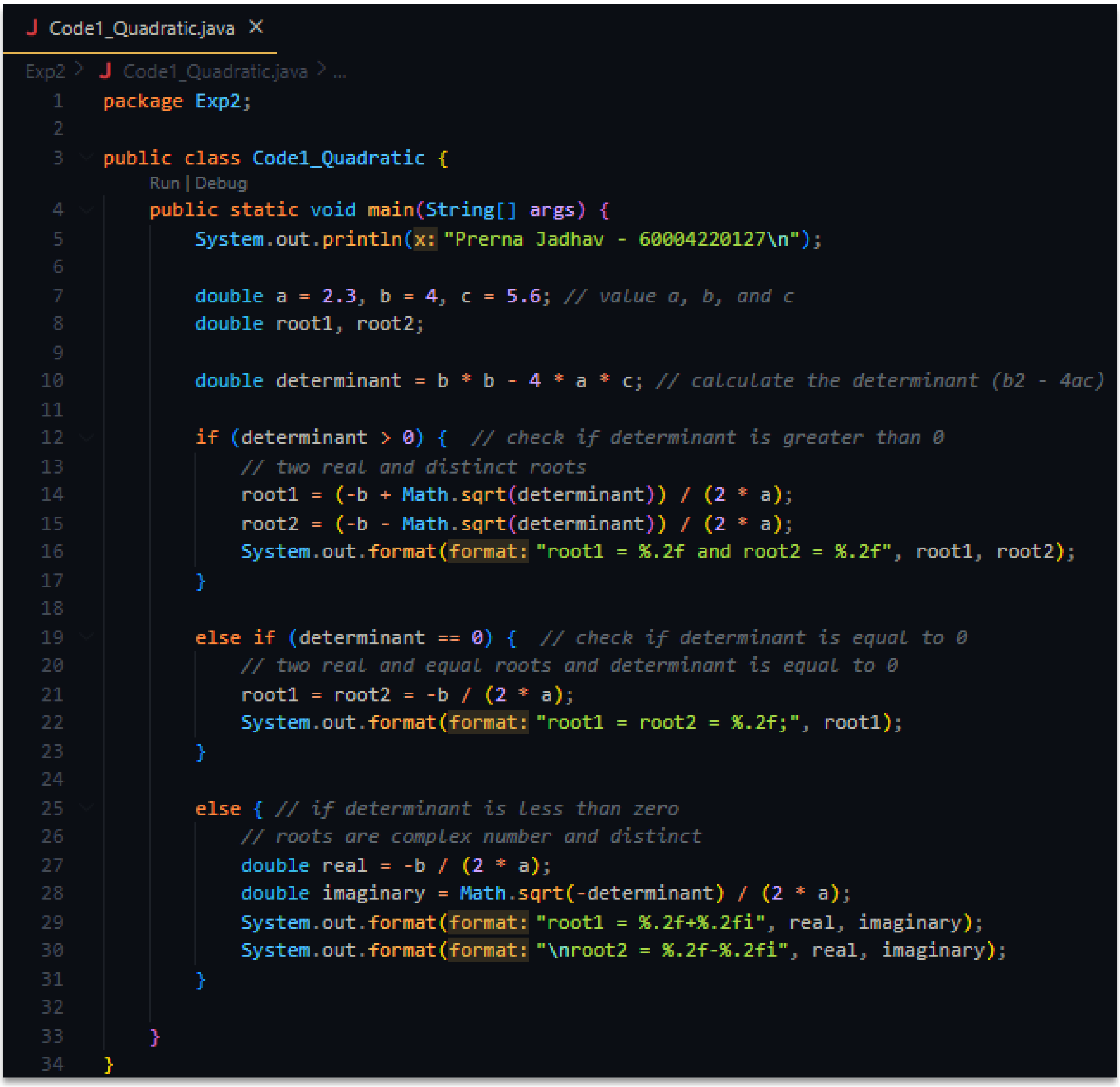
THEORY:
The standard form of a quadratic equation is: <u>ax2 + bx + c = 0</u>
Here, a, b, and c are real numbers and a can't be equal to 0.
We can calculate the root of a quadratic by using the formula: <u>x = (-b ± √(b2-4ac)) / (2a)</u>
The ± sign indicates that there will be two roots:

- ✓ root1 = (-b + √(b2-4ac)) / (2a)
- ✓ root1 = (-b - √(b2-4ac)) / (2a)

The term b2-4ac is known as the determinant of a quadratic equation. It specifies the nature of roots. That is,

- ✓ if determinant > 0, roots are real and different
- ✓ if determinant == 0, roots are real and equal
- ✓ if determinant < 0, roots are complex and different

CODE:

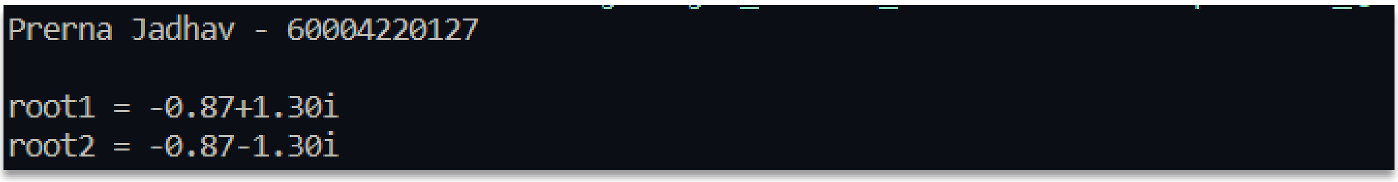


Code1_Quadratic.java

Exp2 > Code1_Quadratic.java > ...

```java
package Exp2;

public class Code1_Quadratic {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Jadhav - 60004220127\n");

        double a = 2.3, b = 4, c = 5.6; // value a, b, and c
        double root1, root2;

        double determinant = b * b - 4 * a * c; // calculate the determinant (b2 - 4ac)

        if (determinant > 0) {  // check if determinant is greater than 0
            // two real and distinct roots
            root1 = (-b + Math.sqrt(determinant)) / (2 * a);
            root2 = (-b - Math.sqrt(determinant)) / (2 * a);
            System.out.format(format: "root1 = %.2f and root2 = %.2f", root1, root2);
        }

        else if (determinant == 0) {  // check if determinant is equal to 0
            // two real and equal roots and determinant is equal to 0
            root1 = root2 = -b / (2 * a);
            System.out.format(format: "root1 = root2 = %.2f;", root1);
        }

        else { // if determinant is less than zero
            // roots are complex number and distinct
            double real = -b / (2 * a);
            double imaginary = Math.sqrt(-determinant) / (2 * a);
            System.out.format(format: "root1 = %.2f+%.2fi", real, imaginary);
            System.out.format(format: "\nroot2 = %.2f-%.2fi", real, imaginary);
        }

    }
}
```

OUTPUT:



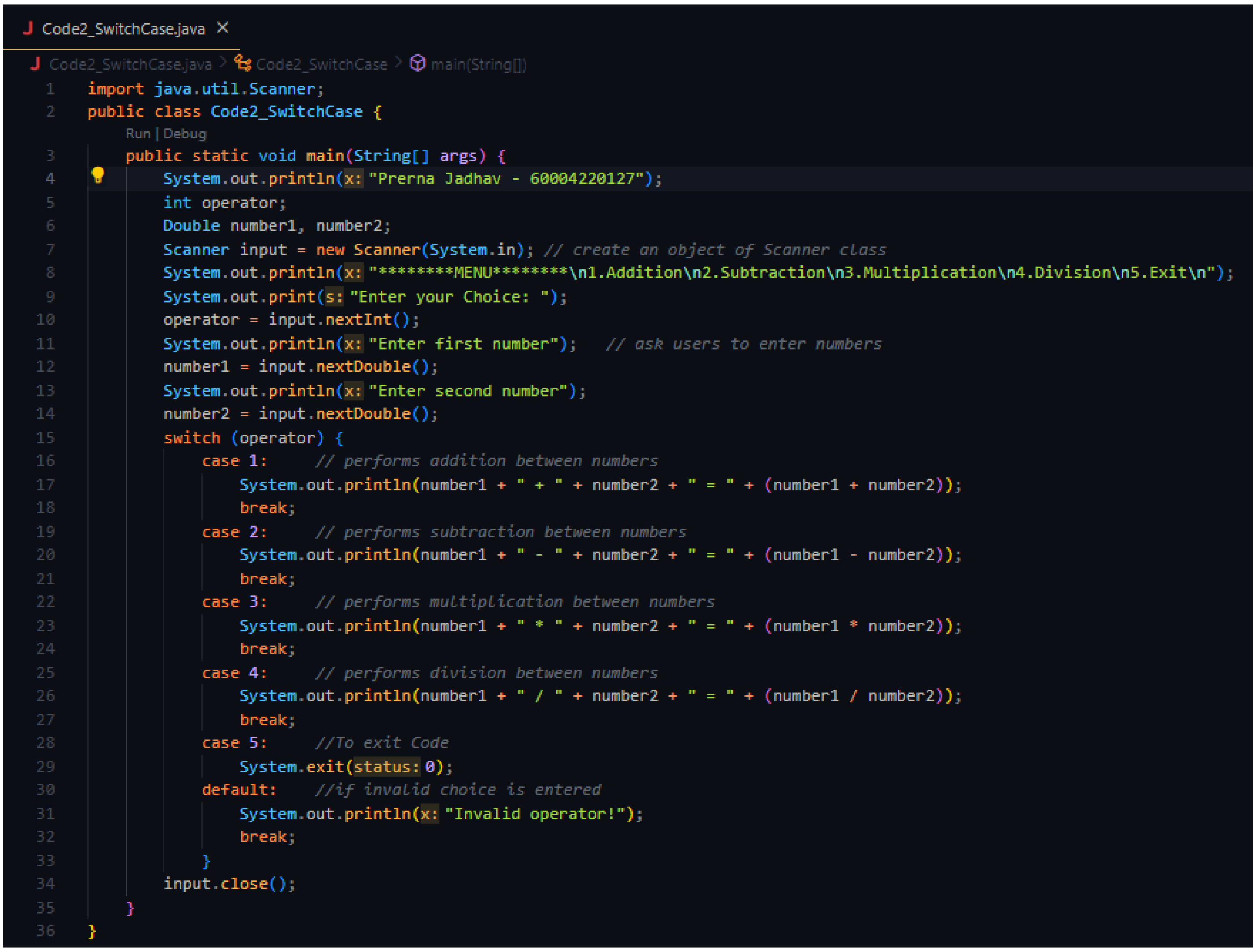
```
Prerna Jadhav - 60004220127

root1 = -0.87+1.30i
root2 = -0.87-1.30i
```

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2022-2023

PROGRAM 2: Write a menu driven program using switch case to perform mathematical operations.

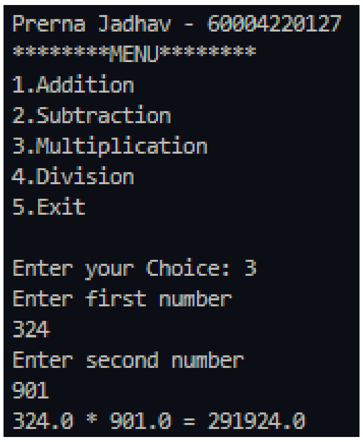
CODE:



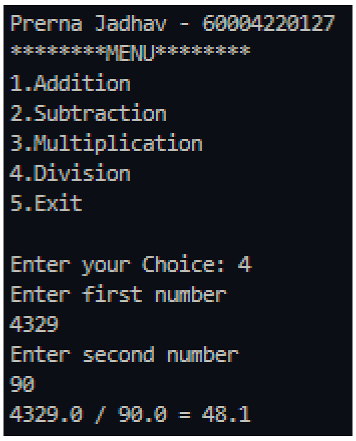
Code2_SwitchCase.java
Code2_SwitchCase.java > Code2_SwitchCase > main(String[])

```java
import java.util.Scanner;
public class Code2_SwitchCase {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Jadhav - 60004220127");
        int operator;
        Double number1, number2;
        Scanner input = new Scanner(System.in); // create an object of Scanner class
        System.out.println(x: "********MENU********\n1.Addition\n2.Subtraction\n3.Multiplication\n4.Division\n5.Exit\n");
        System.out.print(s: "Enter your Choice: ");
        operator = input.nextInt();
        System.out.println(x: "Enter first number");   // ask users to enter numbers
        number1 = input.nextDouble();
        System.out.println(x: "Enter second number");
        number2 = input.nextDouble();
        switch (operator) {
            case 1:     // performs addition between numbers
                System.out.println(number1 + " + " + number2 + " = " + (number1 + number2));
                break;
            case 2:     // performs subtraction between numbers
                System.out.println(number1 + " - " + number2 + " = " + (number1 - number2));
                break;
            case 3:     // performs multiplication between numbers
                System.out.println(number1 + " * " + number2 + " = " + (number1 * number2));
                break;
            case 4:     // performs division between numbers
                System.out.println(number1 + " / " + number2 + " = " + (number1 / number2));
                break;
            case 5:     //To exit Code
                System.exit(status: 0);
            default:    //if invalid choice is entered
                System.out.println(x: "Invalid operator!");
                break;
        }
        input.close();
    }
}
```

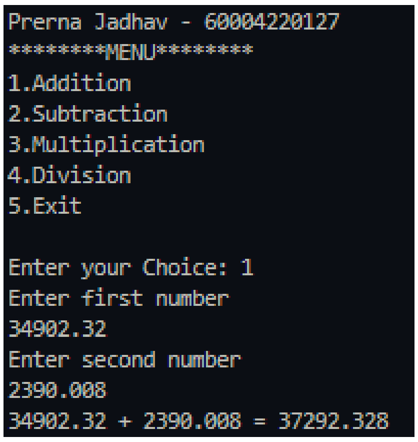
OUTPUT:



```
Prerna Jadhav - 60004220127
********MENU********
1.Addition
2.Subtraction
3.Multiplication
4.Division
5.Exit

Enter your Choice: 3
Enter first number
324
Enter second number
901
324.0 * 901.0 = 291924.0
```



```
Prerna Jadhav - 60004220127
********MENU********
1.Addition
2.Subtraction
3.Multiplication
4.Division
5.Exit

Enter your Choice: 4
Enter first number
4329
Enter second number
90
4329.0 / 90.0 = 48.1
```



```
Prerna Jadhav - 60004220127
********MENU********
1.Addition
2.Subtraction
3.Multiplication
4.Division
5.Exit

Enter your Choice: 1
Enter first number
34902.32
Enter second number
2390.008
34902.32 + 2390.008 = 37292.328
```

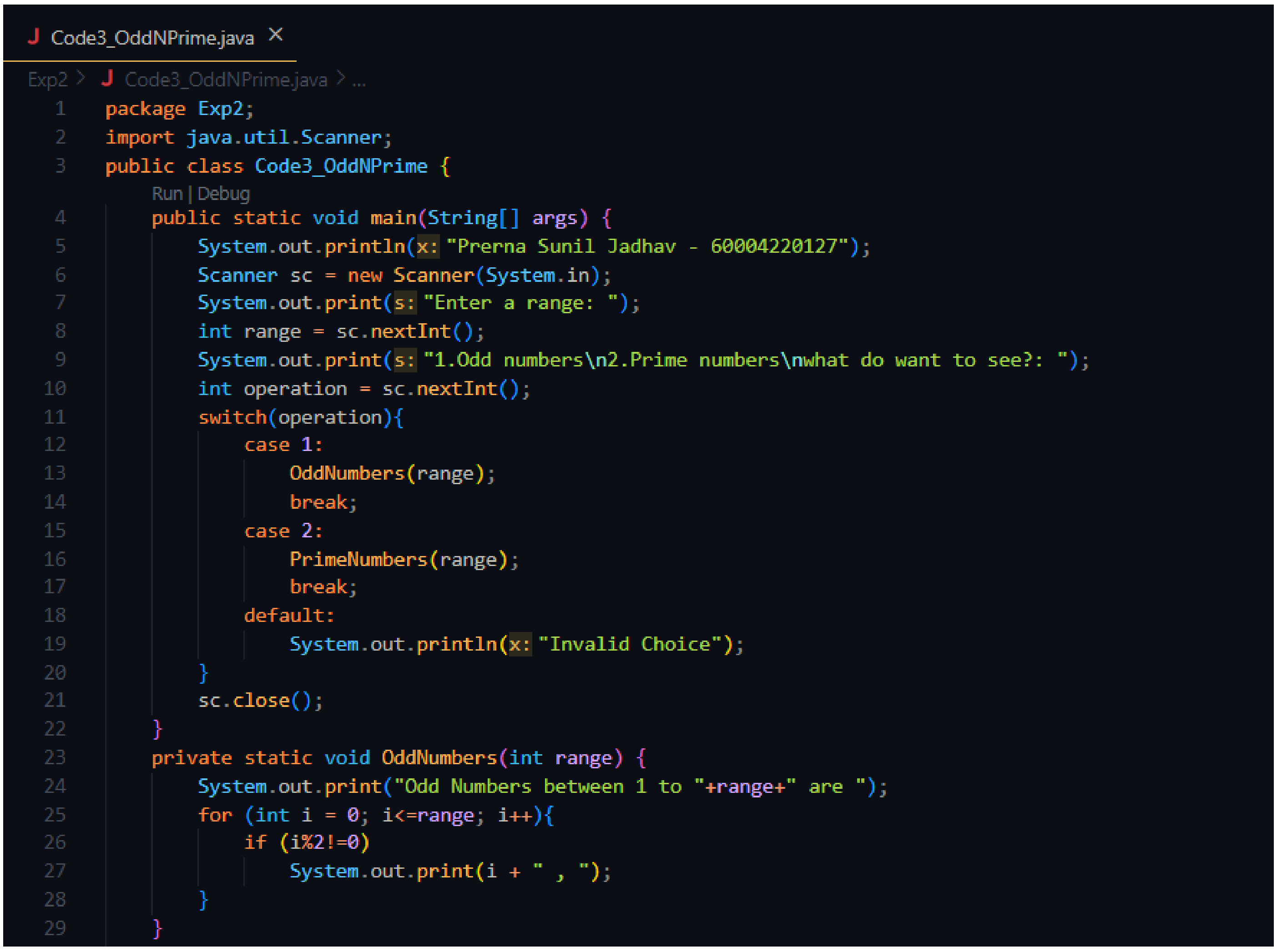
THEORY:
Here, we have used the Scanner class to take 3 inputs from the user.
operator - specifies the operation to be performed
number1/number2 - operands to perform operation on
Since the operator matches the case 3, so the corresponding codes are executed.

```
System.out.println(number + " * " + number2 + " = " +( number1 * number2));
break;
```

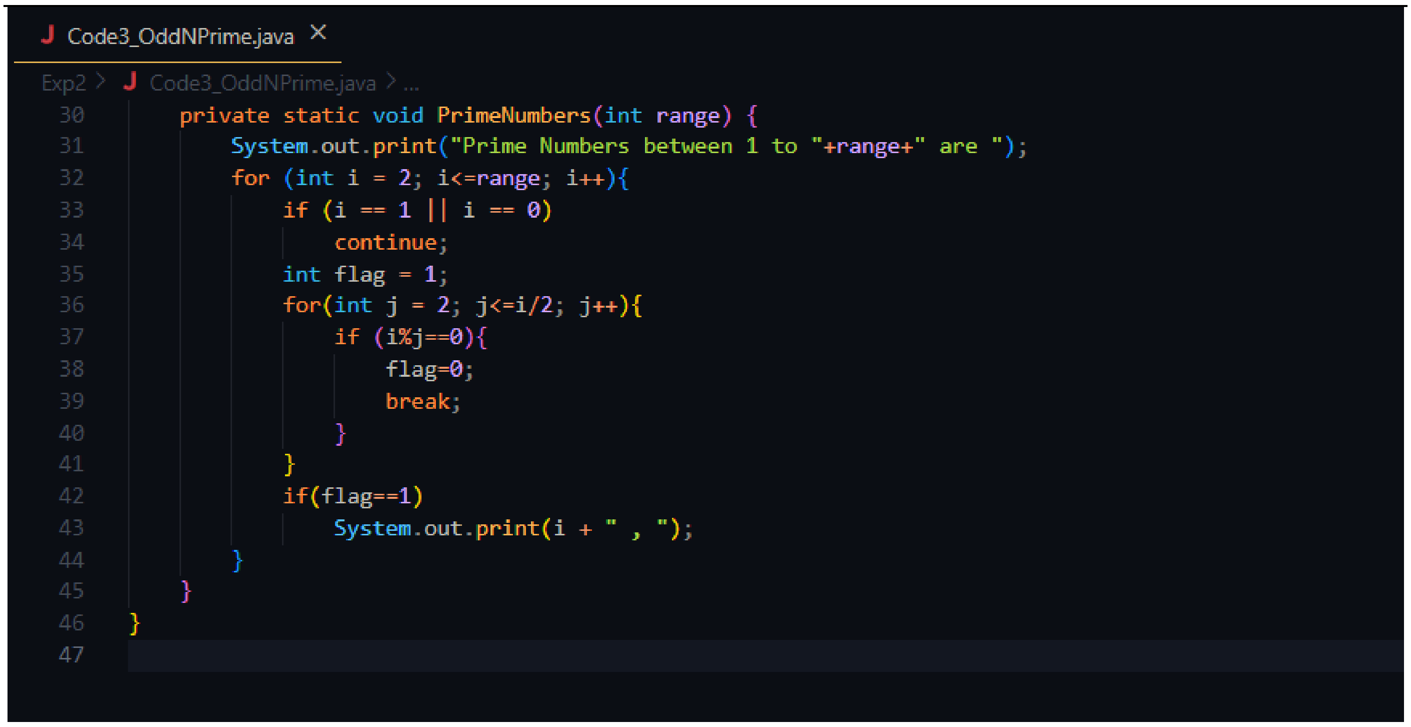
These statements compute the product of two numbers and print the output. Finally, the break statement ends the switch statement.
Similarly, for different operators, different cases are executed.

PROGRAM 3: Write A Program to display odd numbers from given range/ prime numbers from given range.

CODE



```java
package Exp2;
import java.util.Scanner;
public class Code3_OddNPrime {
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127");
        Scanner sc = new Scanner(System.in);
        System.out.print(s: "Enter a range: ");
        int range = sc.nextInt();
        System.out.print(s: "1.Odd numbers\n2.Prime numbers\nwhat do want to see?: ");
        int operation = sc.nextInt();
        switch(operation){
            case 1:
                OddNumbers(range);
                break;
            case 2:
                PrimeNumbers(range);
                break;
            default:
                System.out.println(x: "Invalid Choice");
        }
        sc.close();
    }
    private static void OddNumbers(int range) {
        System.out.print("Odd Numbers between 1 to "+range+" are ");
        for (int i = 0; i<=range; i++){
            if (i%2!=0)
                System.out.print(i + " , ");
        }
    }
```

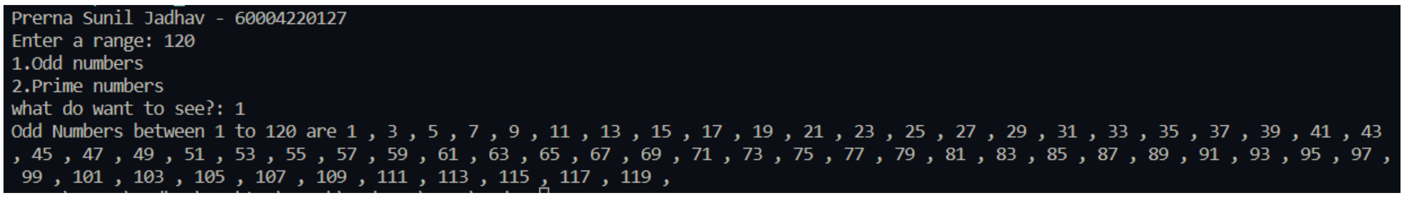


Code3_OddNPrime.java ×

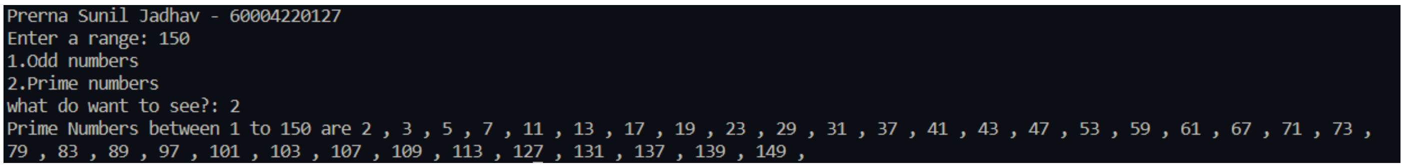
Exp2 > Code3_OddNPrime.java > ...

```java
    private static void PrimeNumbers(int range) {
        System.out.print("Prime Numbers between 1 to "+range+" are ");
        for (int i = 2; i<=range; i++){
            if (i == 1 || i == 0)
                continue;
            int flag = 1;
            for(int j = 2; j<=i/2; j++){
                if (i%j==0){
                    flag=0;
                    break;
                }
            }
            if(flag==1)
                System.out.print(i + " , ");
        }
    }
}
```

OUTPUT:



```
Prerna Sunil Jadhav - 60004220127
Enter a range: 120
1.Odd numbers
2.Prime numbers
what do want to see?: 1
Odd Numbers between 1 to 120 are 1 , 3 , 5 , 7 , 9 , 11 , 13 , 15 , 17 , 19 , 21 , 23 , 25 , 27 , 29 , 31 , 33 , 35 , 37 , 39 , 41 , 43
, 45 , 47 , 49 , 51 , 53 , 55 , 57 , 59 , 61 , 63 , 65 , 67 , 69 , 71 , 73 , 75 , 77 , 79 , 81 , 83 , 85 , 87 , 89 , 91 , 93 , 95 , 97 ,
 99 , 101 , 103 , 105 , 107 , 109 , 111 , 113 , 115 , 117 , 119 ,
```



```
Prerna Sunil Jadhav - 60004220127
Enter a range: 150
1.Odd numbers
2.Prime numbers
what do want to see?: 2
Prime Numbers between 1 to 150 are 2 , 3 , 5 , 7 , 11 , 13 , 17 , 19 , 23 , 29 , 31 , 37 , 41 , 43 , 47 , 53 , 59 , 61 , 67 , 71 , 73 ,
79 , 83 , 89 , 97 , 101 , 103 , 107 , 109 , 113 , 127 , 131 , 137 , 139 , 149 ,
```

THEORY:

For Odd:

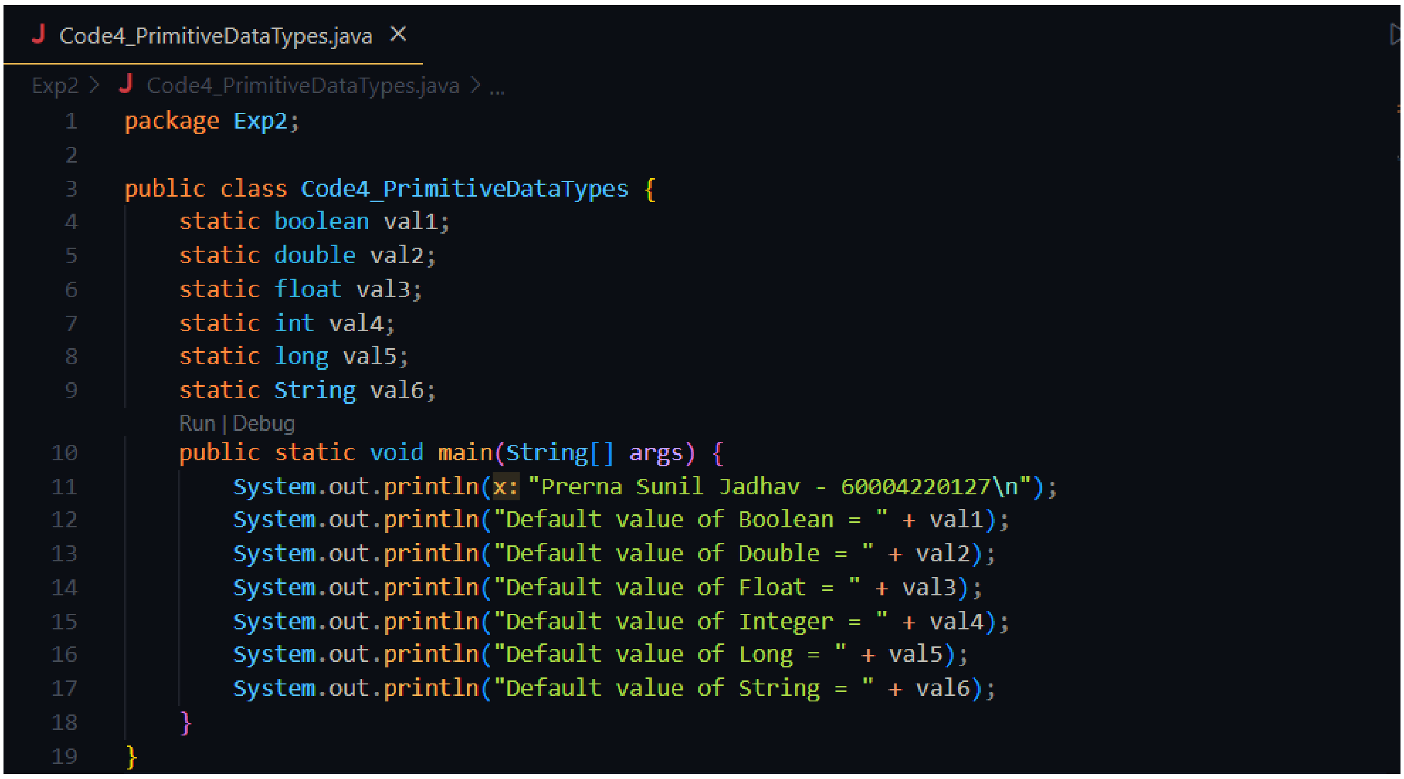
- ✓ Firstly, consider the given number N as input.
- ✓ Then apply a for loop in order to iterate the numbers from 1 to N.
- ✓ At last, check if each number is a odd number using the modulus and if it's a odd number then print it

For Prime:

- ✓ Firstly, consider the given number N as input.
- ✓ Then apply a for loop in order to iterate the numbers from 1 to N.
- ✓ At last, check if each number is a prime number by checking if that number is divisible by any other number other than 1 and itself and if it's a prime number then print it

PROGRAM4: Write A Program to display default value of primitive data types

CODE

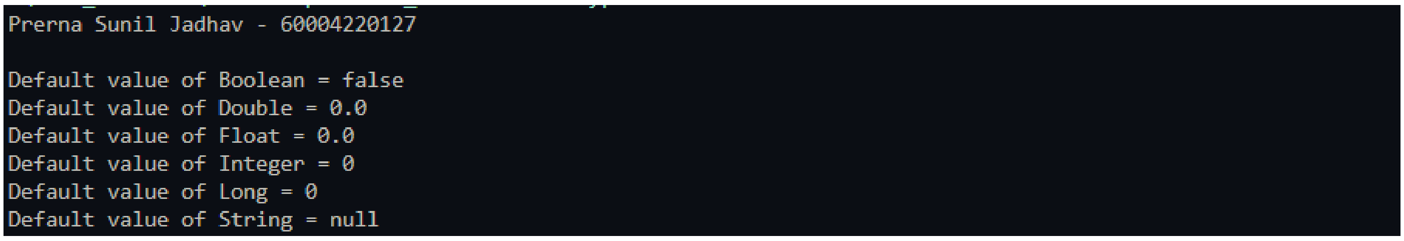


Code4_PrimitiveDataTypes.java

Exp2 > Code4_PrimitiveDataTypes.java > ...

```java
package Exp2;

public class Code4_PrimitiveDataTypes {
    static boolean val1;
    static double val2;
    static float val3;
    static int val4;
    static long val5;
    static String val6;
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127\n");
        System.out.println("Default value of Boolean = " + val1);
        System.out.println("Default value of Double = " + val2);
        System.out.println("Default value of Float = " + val3);
        System.out.println("Default value of Integer = " + val4);
        System.out.println("Default value of Long = " + val5);
        System.out.println("Default value of String = " + val6);
    }
}
```

OUTPUT:



```
Prerna Sunil Jadhav - 60004220127

Default value of Boolean = false
Default value of Double = 0.0
Default value of Float = 0.0
Default value of Integer = 0
Default value of Long = 0
Default value of String = null
```

THEORY:

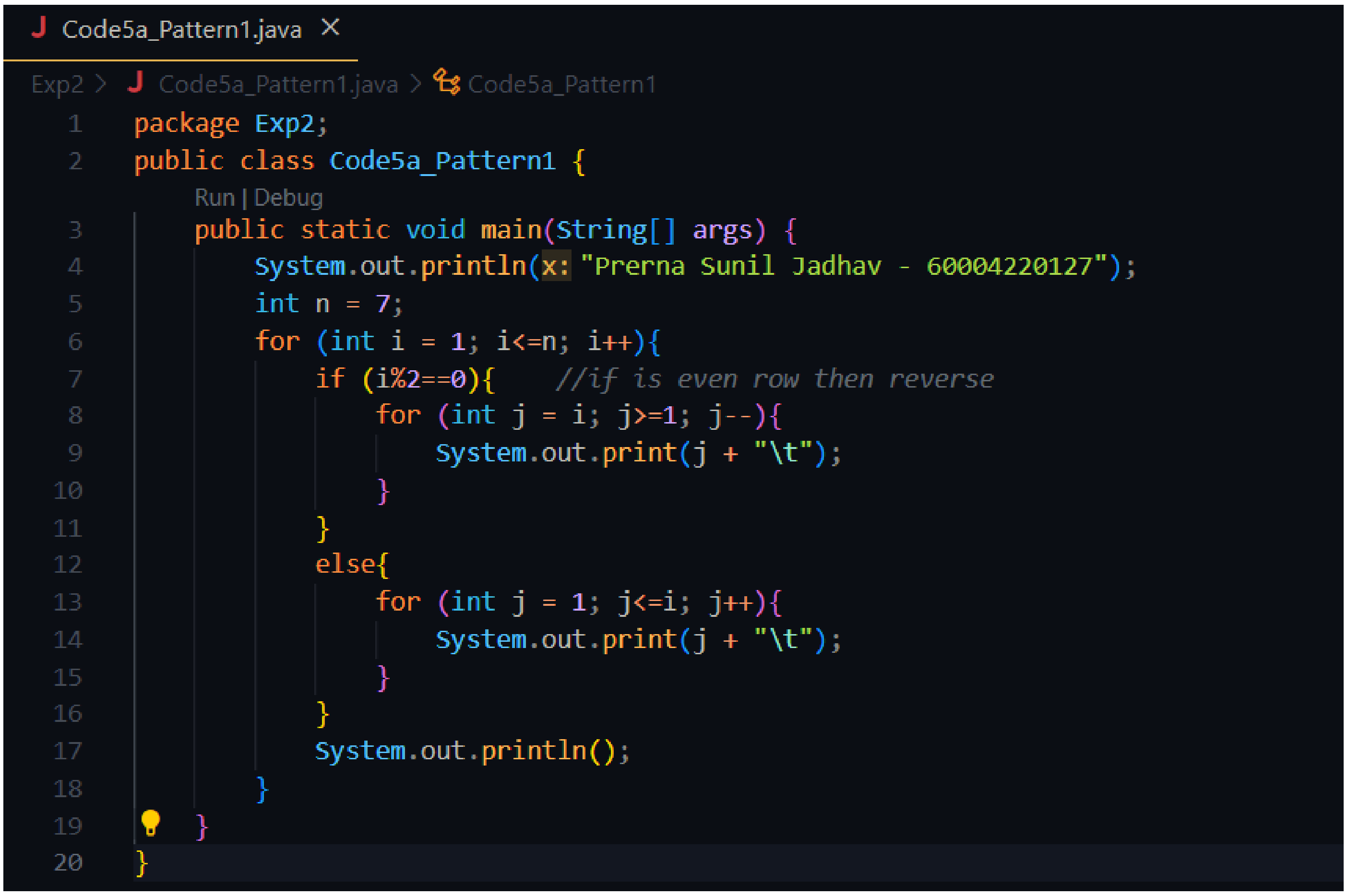
Primitive types are the Java data types used for data manipulation, for example, int, char, float, double, boolean, etc.

- ✓ byte: An 8-bit signed two's complement integer (128 – 127)
- ✓ short: A 16-bit signed two's complement integer (-32768 – 32767)
- ✓ int: A 32-bit signed two's complement integer (-2,147,483,648 - 2,147,483,647)
- ✓ long: A 64-bit two's complement integer (-9,223,372,036,854,775,808 - 9,223,372,036,854,775,807)
- ✓ char: A single 16-bit Unicode character. ('\u0000' (or 0) - '\uffff')
- ✓ float: A single-precision 32-bit IEEE 754 floating point (1.4E-45 - 3.4028235E38)
- ✓ double: A double-precision 64-bit IEEE 754 floating point (4.9E-324 - 1.7976931348623157E308)
- ✓ boolean: Possible values, TRUE and FALSE.

PROGRAM5A: Write A Program to display the following patterns:

```
1
2	1
1	2	3
4	3	2	1
1	2	3	4	5
6	5	4	3	2	1
1	2	3	4	5	6	7
```

CODE

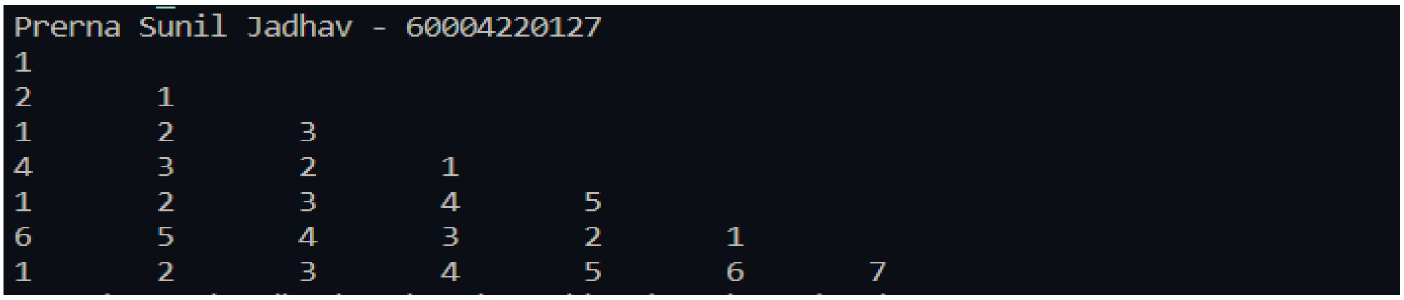


Code5a_Pattern1.java

Exp2 > Code5a_Pattern1.java > Code5a_Pattern1

```java
package Exp2;
public class Code5a_Pattern1 {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127");
        int n = 7;
        for (int i = 1; i<=n; i++){
            if (i%2==0){    //if is even row then reverse
                for (int j = i; j>=1; j--){
                    System.out.print(j + "\t");
                }
            }
            else{
                for (int j = 1; j<=i; j++){
                    System.out.print(j + "\t");
                }
            }
            System.out.println();
        }
    }
}
```

OUTPUT:



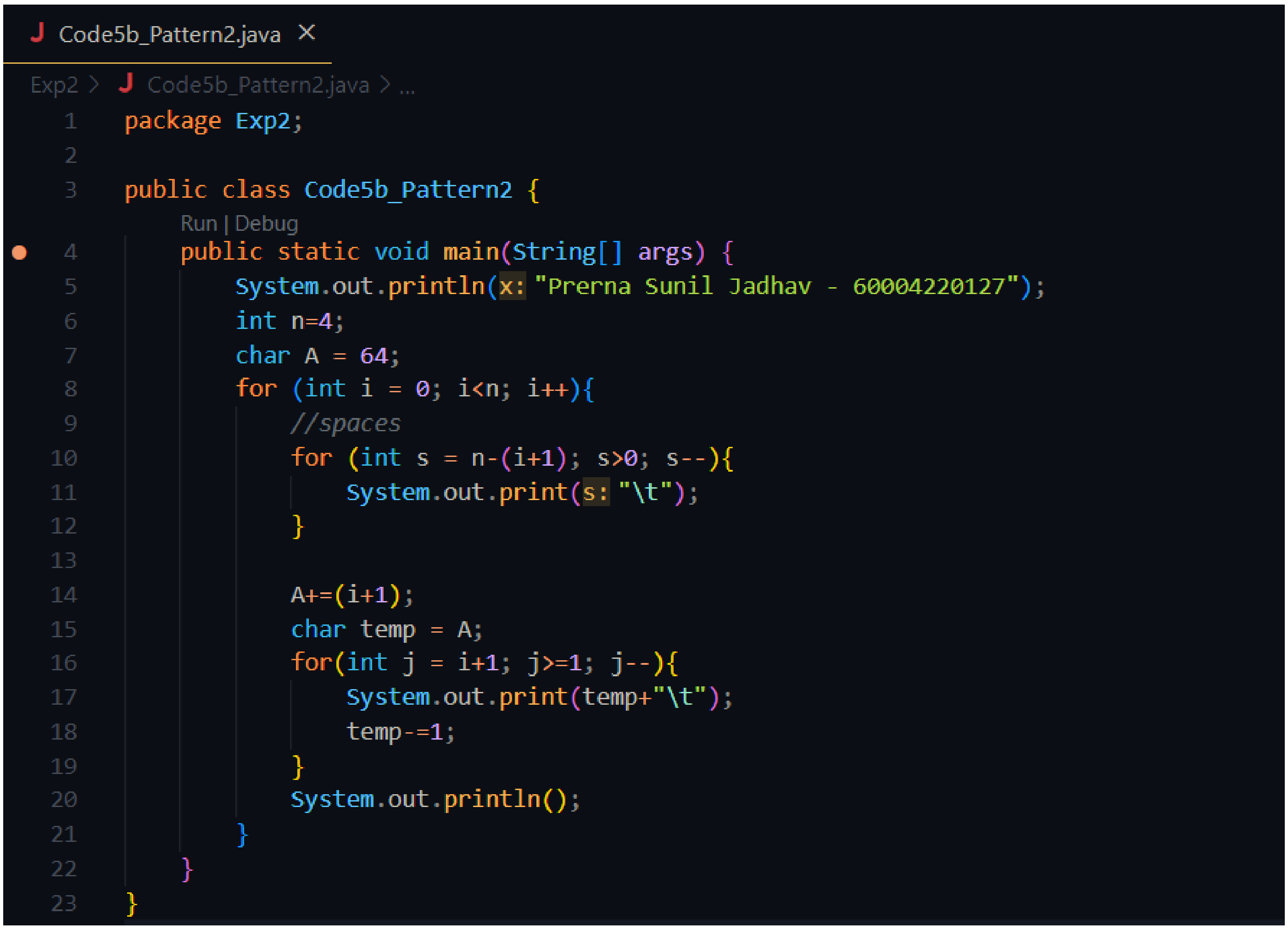
```
Prerna Sunil Jadhav - 60004220127
1
2	1
1	2	3
4	3	2	1
1	2	3	4	5
6	5	4	3	2	1
1	2	3	4	5	6	7
```

THEORY:
Here we started a loop till n, here its 7 And will check for every iteration of n if its even if its even reverse the order to print else print sequentially.

PROGRAM5B: Write A Program to display the following patterns:

```
         A
       C B
     F E D
   J I  K G
```

CODE:

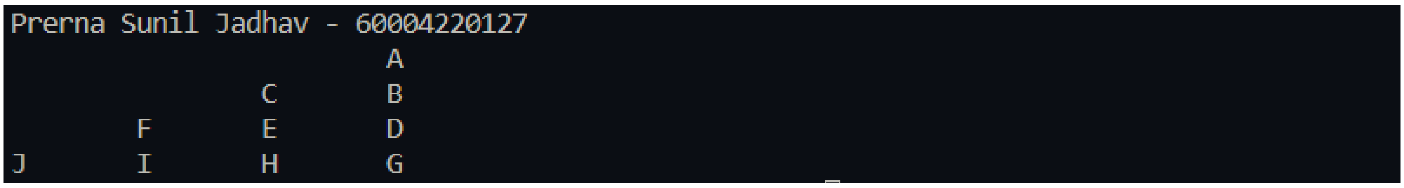


Code5b_Pattern2.java

Exp2 > Code5b_Pattern2.java > ...

```java
package Exp2;

public class Code5b_Pattern2 {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127");
        int n=4;
        char A = 64;
        for (int i = 0; i<n; i++){
            //spaces
            for (int s = n-(i+1); s>0; s--){
                System.out.print(s: "\t");
            }

            A+=(i+1);
            char temp = A;
            for(int j = i+1; j>=1; j--){
                System.out.print(temp+"\t");
                temp-=1;
            }
            System.out.println();
        }
    }
}
```

OUTPUT:



```
Prerna Sunil Jadhav - 60004220127
                              A
                    C         B
          F         E         D
J         I         H         G
```

THEORY:
Here we Initially assume the character to be printed 64 i.e., character before A and will add 1, 2, 3 ... in our Ascii value as our row increments and print the sequence in reverse by decrementing the duplicate value by 1 in inner for loop.

CONCLUSION:
Primitive data types are the building blocks of data manipulation.
For statement consumes the initialization, condition, and increment/decrement in one line thereby providing a shorter, easy to debug
structure of looping.
If a loop exists inside the body of another loop, it's called a nested loop.
That is why nested loops are also called as “loop inside loop“.
The Java switch statement only works with:

- ✓ Primitive data types: byte, short, char, and int
- ✓ Enumerated types
- ✓ String Class
- ✓ Wrapper Classes: Character, Byte, Short, and Integer.