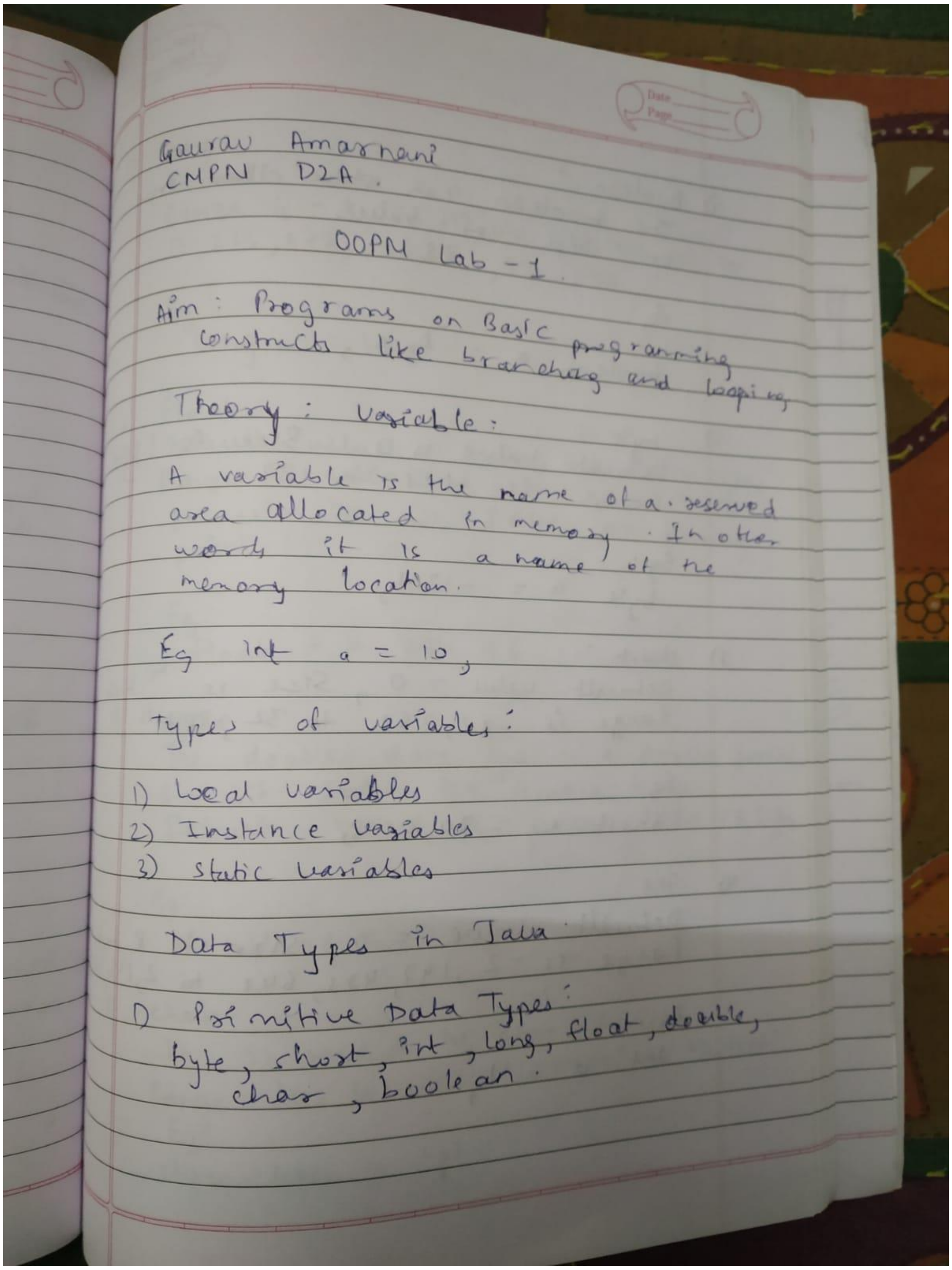
Name: Gaurav Amarnani.
Class: CMPN DSE.
OOPM Lab 01.



Gaurav Amarnani
CMPN D2A.

OOPM Lab - 1.

Aim : Programs on Basic programming constructs like branching and looping,

Theory : variable :

A variable is the name of a reserved area allocated in memory. In other words it is a name of the memory location.

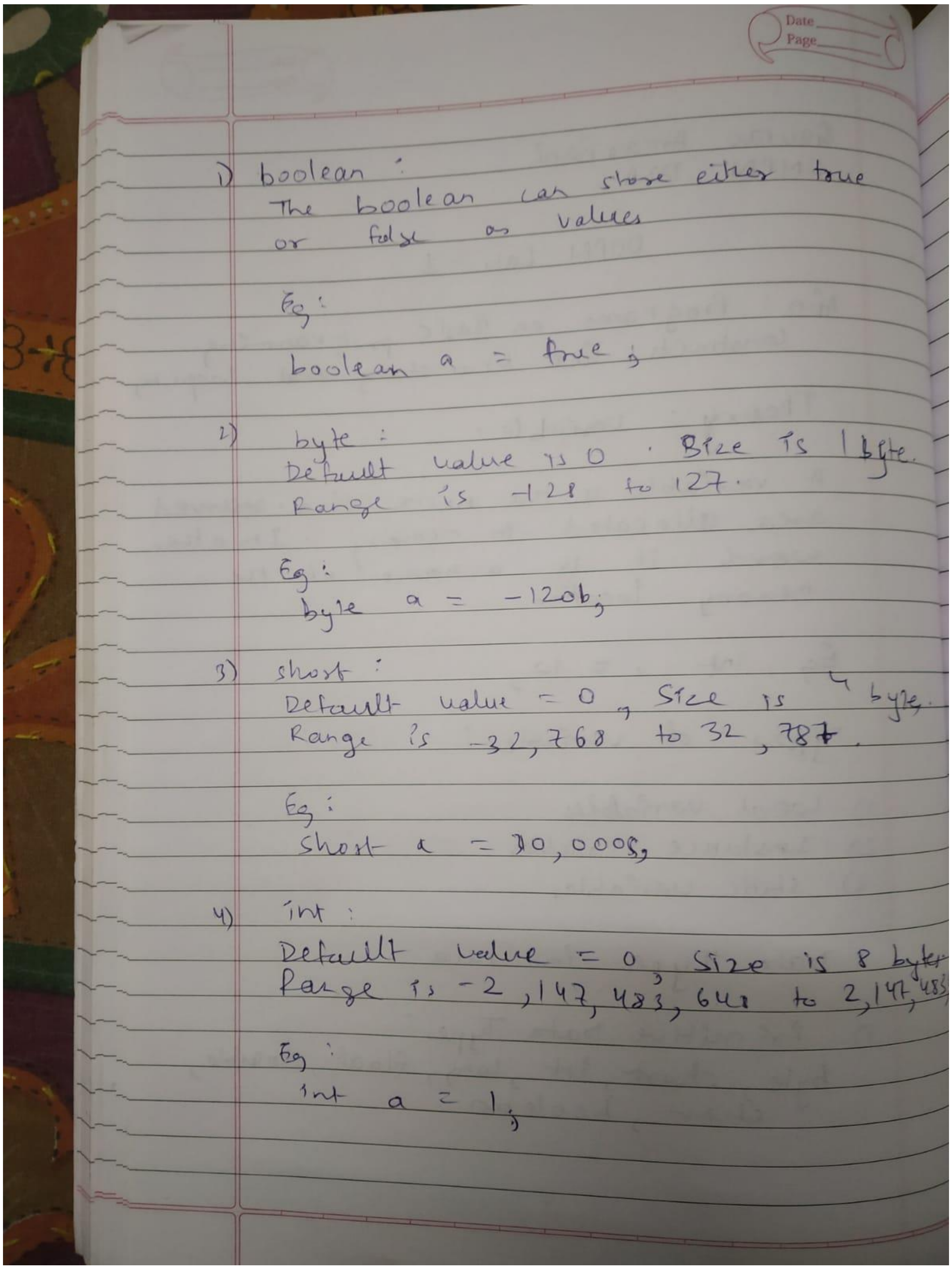
Eg int a = 10;

Types of variables :

1) Local variables
2) Instance variables
3) static variables

Data Types in Java :

1) Primitive Data Types :
byte, short, int, long, float, double, char, boolean.



Date
Page

1) boolean :
The boolean can store either true or false as values

Eg :

boolean a = true ;

2) byte :
Default value is 0 . Size is 1 byte.
Range is -128 to 127.

Eg :
byte a = -120b;

3) short :
Default value = 0 , Size is 4 byte.
Range is -32,768 to 32,787.

Eg :
short a = 30,000s;

4) int :
Default value = 0 , Size is 8 bytes
Range is -2,147,483,648 to 2,147,483

Eg :
int a = 1;

5) long :
Default value = 0 Size is 16 bytes.
Range is -9,223,372,036,854,775,808 to
9,223,372,036,854,775,807.

Eg :
long a = 10L;

6) float :
Default value = 0.00. Size is 4 bytes.
Range is unlimited. It is a single
precision 32 bit IEEE 754 floating
point.

Eg.
float f = 234.4f;

7) double :
The double data type is a double precision
64 bit IEEE 754 floating point.
Default value is 0.00. Size is 8 bytes.

Eg.
double d = 0.01;

8) char :
It is used to store characters.
Range lies between '/u0000' to '/uffff'.
Eg:
char letter = 'p';

Date
Page

Types of operators :

1) Unary operator :
++ , -- , ^ , ! .

2) Arithmetic operators :
+ , - , * , / , %

3) Shift operator :
<< , >> , >>> .

4) Relational operator.
< , > , <= , >= , == , != .

5) Bitwise operator.
AND (&) , XOR (^) , OR (|) .

6) Logical operators :
AND (&&) , OR (||)

7) Ternary operator :
? :

8) Assignment operator :
= , += , -= , *= , /= , %=
^= , |= , <<= , >>= , >>>=

Taking input from keyboard:

1) Console (java.io):

Methods:

String readLine()
String readLine(String format, Object...args)
char [] readPassword()
char [] readPassword(String f, object...a)

2) InputStreamReader & BufferedReader

```
ISR isR = new ISR(System.in);
BR bR = new BR(isR);
String input - bR.readLine();
```

3) Scanner (java.util):

```
Scanner scanner = new Scanner(System.in);
```

Methods:

String	next()
String	nextLine()
int	nextInt()
byte	nextByte()
short	nextShort()
float	nextFloat()
double	nextDouble()

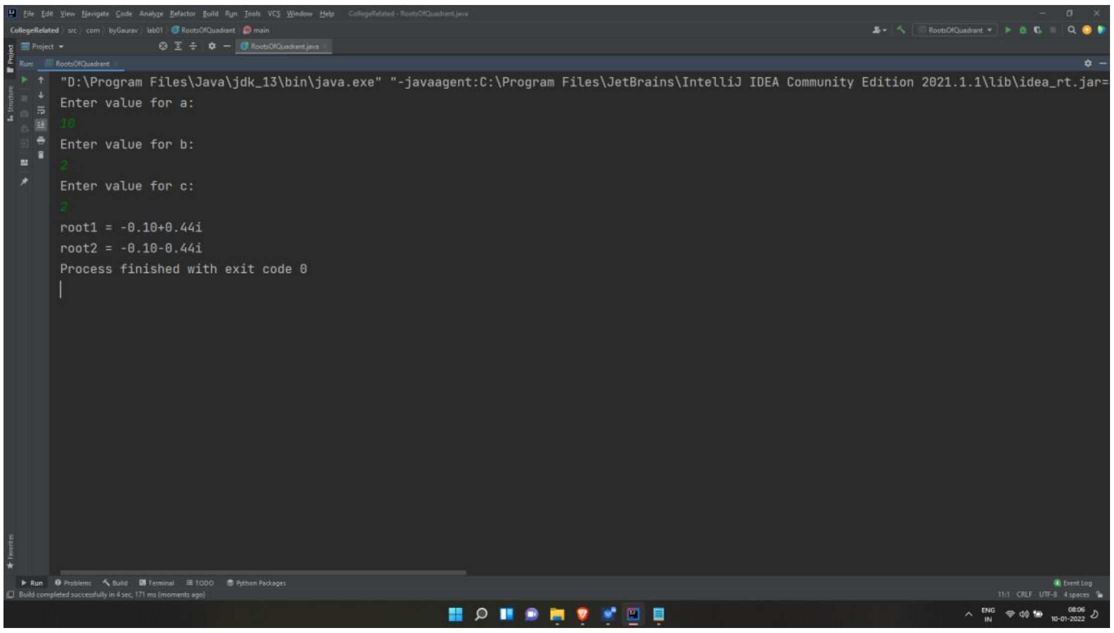
Programs:

Program 1: To implement a program to print the roots of quadratic equations.

```
package com.byGaurav.lab01;

import java.util.Scanner;

import static java.lang.System.*;

/**
 * @author  Gaurav Amarnani.
 */

public class RootsOfQuadrant {

    public static void main(String...args) {

        int t = 20;
        Scanner scanner = new Scanner(in);
        out.println("Enter value for a: ");
        double a = scanner.nextDouble();
        out.println("Enter value for b: ");
        double b = scanner.nextDouble();
        out.println("Enter value for c: ");
        double c = scanner.nextDouble();
        double root1, root2;

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

        if(a == 0)
            out.println("Not a Quadratic Equation.");

        else {
            // Check if Determinant is greater than 0
            if (determinant > 0) {

                // Two Real and Distinct Roots
                root1 = (-b + Math.sqrt(determinant)) / (2 * a);
                root2 = (-b - Math.sqrt(determinant)) / (2 * a);

                out.format("root1 = %.2f and root2 = %.2f", root1, root2);
            }

            // Check if Determinant is equal to 0
            else if (determinant == 0) {
```

```
            // Two Real and Equal Roots
            // Determinant is equal to 0
            root1 = -b / (2 * a);
            out.format("root1 = root2 = %.2f;", root1);
        }

        // If Determinant is less than 0
        else {

            // Roots are Complex Number and Distinct
            double real = -b / (2 * a);
            double imaginary = Math.sqrt(-determinant) / (2 * a);
            out.format("root1 = %.2f+%.2fi", real, imaginary);
            out.format("\nroot2 = %.2f-%.2fi", real, imaginary);
        }
    }
    scanner.close();
  }
}
```

Output:

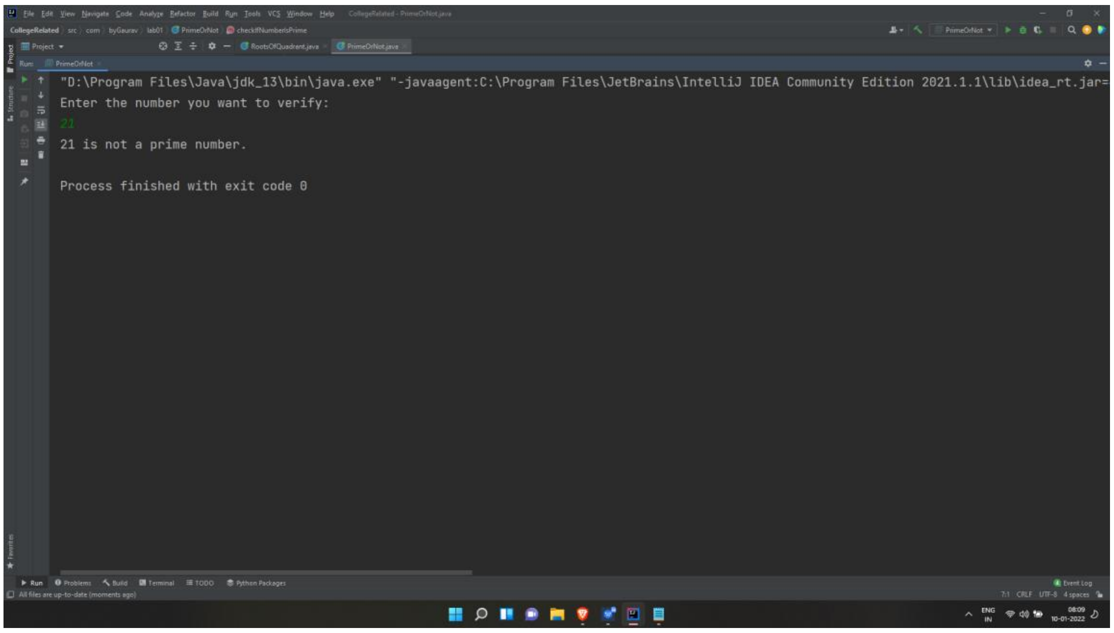


```
"D:\Program Files\Java\jdk_13\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.1.1\lib\idea_rt.jar=
Enter value for a:
10
Enter value for b:
2
Enter value for c:
2
root1 = -0.10+0.44i
root2 = -0.10-0.44i
Process finished with exit code 0
```

Program 2: To implement a Program to check if the entered no. is a prime no. or not.

```
package com.byGaurav.lab01;

import java.util.Scanner;

import static java.lang.System.*;

/**
 * @author  Gaurav Amarnani.
 */

public class PrimeOrNot {

    public static void main(String...args){
        Scanner scanner = new Scanner(in);
        out.println("Enter the number you want to verify: ");
        Integer input = scanner.nextInt();
        if(checkIfNumberIsPrime(input))
            out.println(input + " is a prime number.");
        else
            out.println(input + " is not a prime number.");
        scanner.close();
    }

    public static Boolean checkIfNumberIsPrime(Integer number) {
        for(int i = 2; i < number; i++) {
            if(number % i == 0)
                return false;
        }
        return true;
    }
}
```

Output:

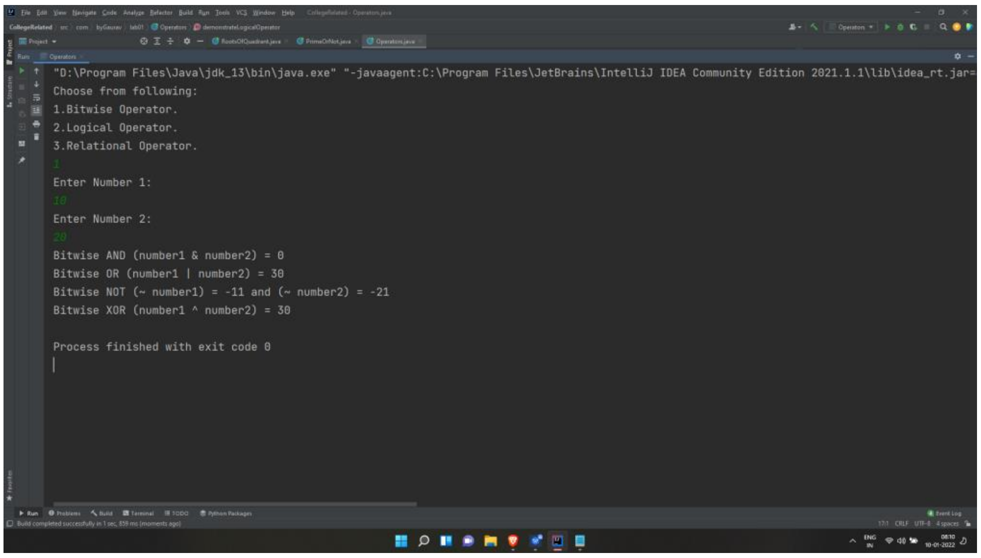


```
"D:\Program Files\Java\jdk_13\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.1.1\lib\idea_rt.jar=
Enter the number you want to verify:
21
21 is not a prime number.

Process finished with exit code 0
```

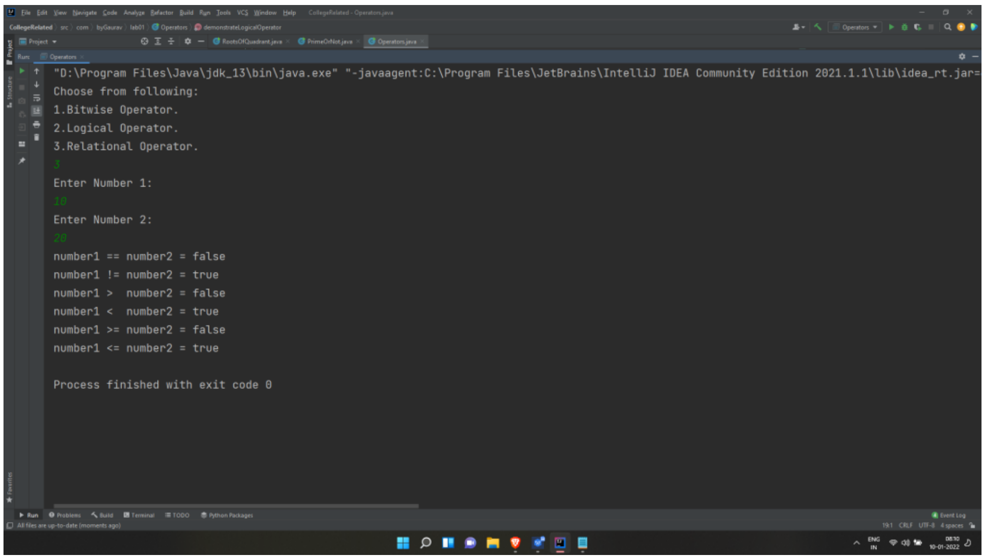
Program 3: To implement a Program to demonstrate the working of types of operators(Bitwise, Logical and relational) using switch case.

```
package com.byGaurav.lab01;

import java.util.Scanner;

import static java.lang.System.*;

/**
 * @author  Gaurav Amarnani.
 */

public class Operators {

   private static Integer number1, number2;
   private static Scanner scanner;

   public static void main(String...args) {
      scanner = new Scanner(in);
      out.println("Choose from following: \n1.Bitwise Operator.\n2.Logical
Operator.\n3.Relational Operator.");
      switch (scanner.nextInt()) {
         case 1:
            takeInput();
            demonstrateBitwiseOperator(number1, number2);
            break;
         case 2:
            demonstrateLogicalOperator(true, false);
            break;
         case 3:
            takeInput();
            demonstrateRelationalOperator(number1, number2);
            break;
         default:
            out.println("Please enter proper choice.");
            main();
      }
      scanner.close();
   }

   public static void takeInput() {
      scanner = new Scanner(in);
      out.println("Enter Number 1: ");
      number1 = scanner.nextInt();
      out.println("Enter Number 2: ");
      number2 = scanner.nextInt();
   }
   public static void demonstrateBitwiseOperator(Integer number1, Integer
number2) {
        out.println("Bitwise AND (number1 & number2) = " + (number1 &
number2));
        out.println("Bitwise OR (number1 | number2) = " + (number1 | number2));
        out.println("Bitwise NOT (~ number1) = " + (~ number1) + " and (~
number2) = " + (~ number2));
        out.println("Bitwise XOR (number1 ^ number2) = " + (number1 ^
number2));
   }

   public static void demonstrateLogicalOperator(Boolean value1, Boolean
value2) {
        out.println("Logical AND (true && false) = " + (value1 && value2));
        out.println("Logical OR (true | false) = " + (value1 | value2));
        out.println("Logical NOT (! false) = " + (! value2));
   }

   public static void demonstrateRelationalOperator(Integer number1, Integer
number2) {
        out.println("number1 == number2 = " + (number1 == number2) );
        out.println("number1 != number2 = " + (number1 != number2) );
        out.println("number1 >  number2 = " + (number1 >  number2) );
        out.println("number1 <  number2 = " + (number1 <  number2) );
        out.println("number1 >= number2 = " + (number1 >= number2) );
        out.println("number1 <= number2 = " + (number1 <= number2) );
   }
}
```

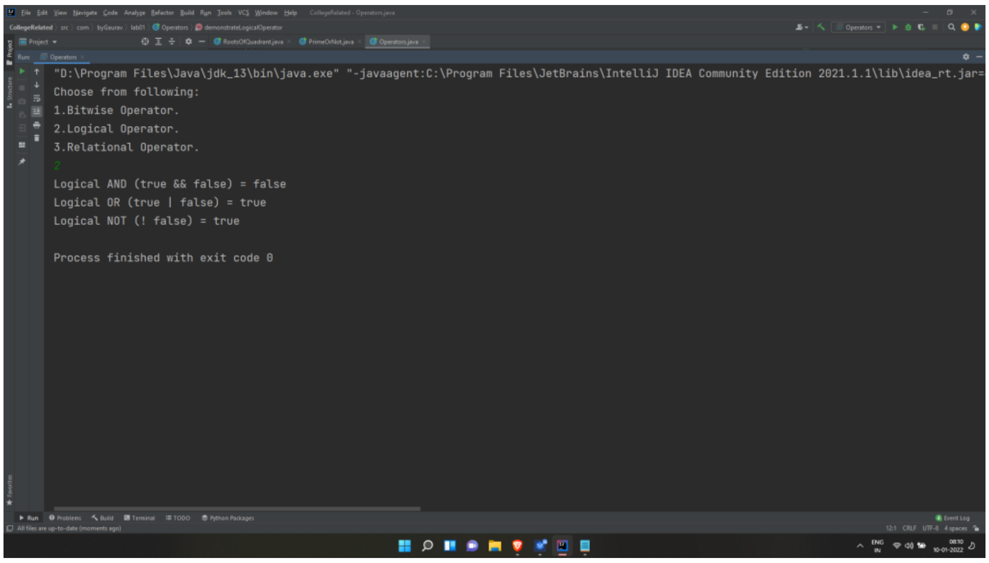
Output:



```
"D:\Program Files\Java\jdk_13\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.1.1\lib\idea_rt.jar=
Choose from following:
1.Bitwise Operator.
2.Logical Operator.
3.Relational Operator.
1
Enter Number 1:
10
Enter Number 2:
20
Bitwise AND (number1 & number2) = 0
Bitwise OR (number1 | number2) = 30
Bitwise NOT (~ number1) = -11 and (~ number2) = -21
Bitwise XOR (number1 ^ number2) = 30

Process finished with exit code 0
```



```
"D:\Program Files\Java\jdk_13\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.1.1\lib\idea_rt.jar=
Choose from following:
1.Bitwise Operator.
2.Logical Operator.
3.Relational Operator.
3
Enter Number 1:
10
Enter Number 2:
20
number1 == number2 = false
number1 != number2 = true
number1 >  number2 = false
number1 <  number2 = true
number1 >= number2 = false
number1 <= number2 = true

Process finished with exit code 0
```

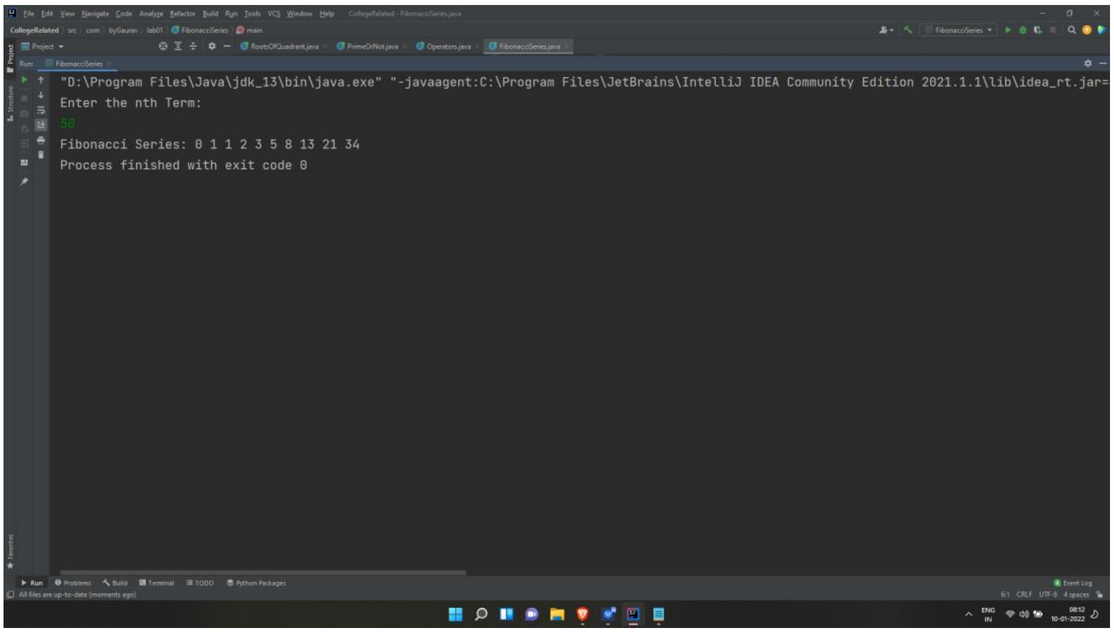


```
"D:\Program Files\Java\jdk_13\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.1.1\lib\idea_rt.jar=
Choose from following:
1.Bitwise Operator.
2.Logical Operator.
3.Relational Operator.
2
Logical AND (true && false) = false
Logical OR (true | false) = true
Logical NOT (! false) = true

Process finished with exit code 0
```

Program 4: WAJP Print the Fibonacci series upto the nth term taking the value of n from the user.

```
package com.byGaurav.lab01;

import java.util.Scanner;

import static java.lang.System.*;

/**
 * @author  Gaurav Amarnani.
 */

public class FibonacciSeries {
   public static void main(String...args) {
      Scanner scanner = new Scanner(in);
      out.println("Enter the nth Term: ");
      Integer nthTerm = scanner.nextInt();
      fibonacciSeriesUpToNthTerm(nthTerm);
      scanner.close();
   }

   public static void fibonacciSeriesUpToNthTerm(Integer limit) {
      if(limit > 0) {
         Integer first = 0, second = 1, temp;
         out.print("Fibonacci Series: " + first + " " + second + " ");
         while (second <= limit) {
            temp = first + second;
            first = second;
            second = temp;
            if (second < limit)
               out.print(second + " ");
         }
      }
      else {
         out.println("Wrong Input.");
      }
   }
}
```

Output:



```
"D:\Program Files\Java\jdk_13\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.1.1\lib\idea_rt.jar=
Enter the nth Term:
50
Fibonacci Series: 0 1 1 2 3 5 8 13 21 34
Process finished with exit code 0
```