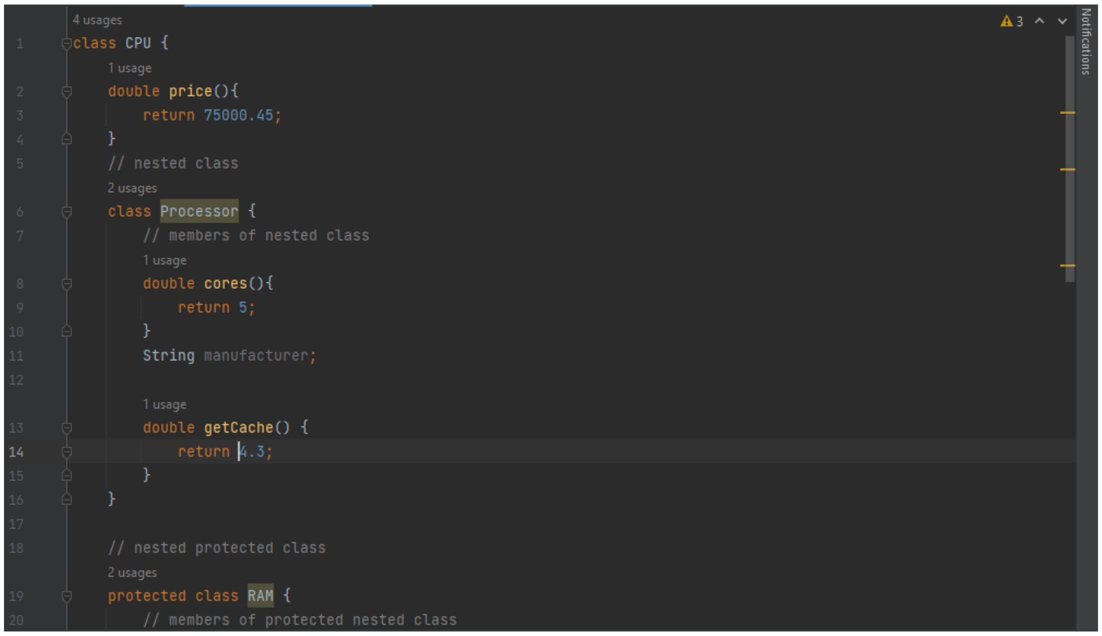
CORE JAVA

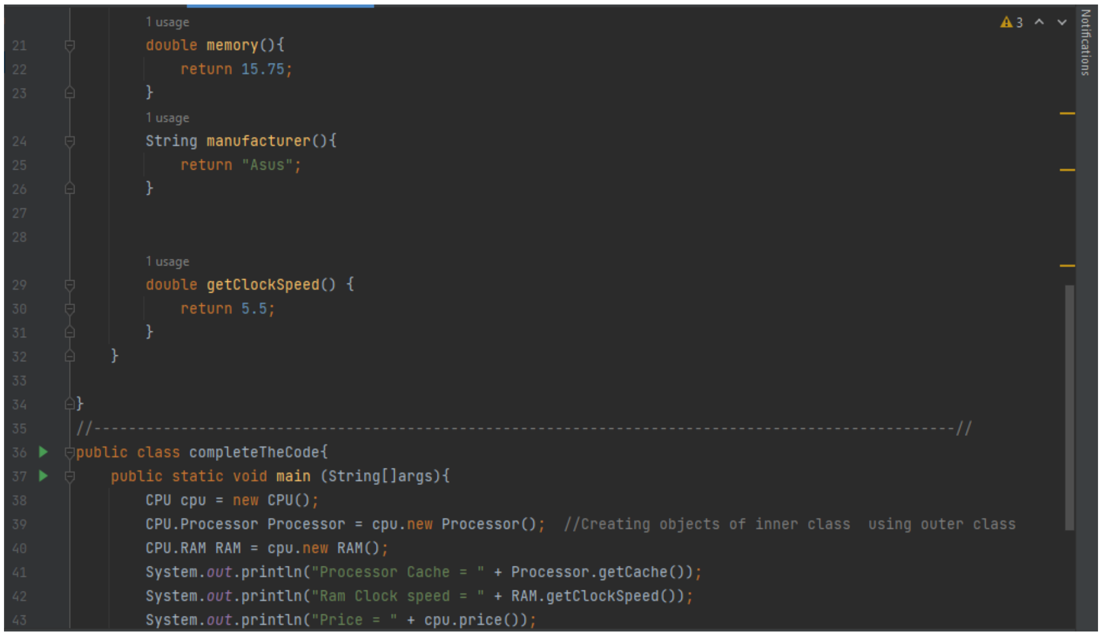
DAY - 3

By Devayush BAjaj

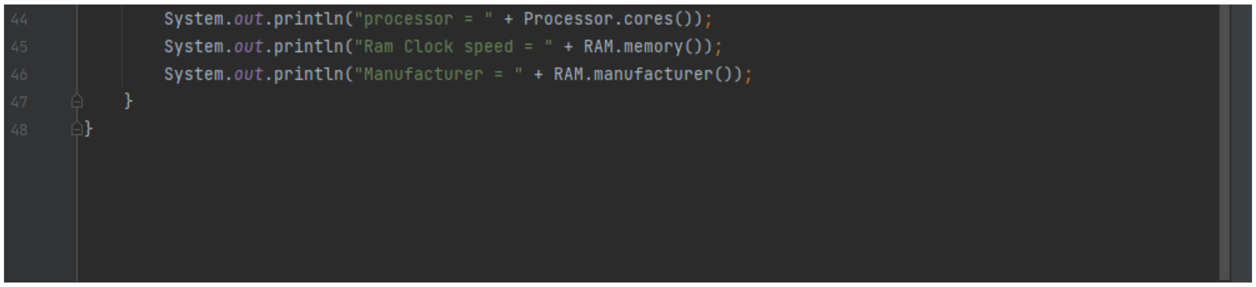
Q) Complete the code



```java
class CPU {
    double price(){
        return 75000.45;
    }
    // nested class
    class Processor {
        // members of nested class
        double cores(){
            return 5;
        }
        String manufacturer;

        double getCache() {
            return 4.3;
        }
    }

    // nested protected class
    protected class RAM {
        // members of protected nested class
```

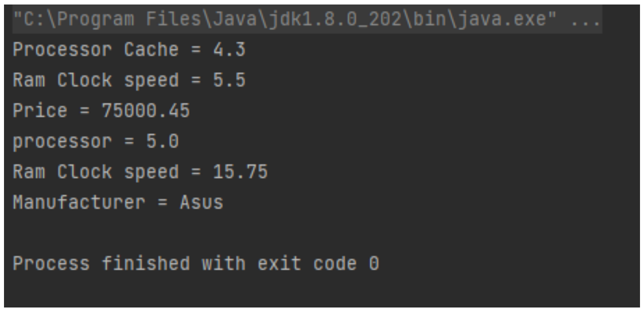


```java
        double memory(){
            return 15.75;
        }
        String manufacturer(){
            return "Asus";
        }


        double getClockSpeed() {
            return 5.5;
        }
    }

}
//-------------------------------------------------------------------------------------------------//
public class completeTheCode{
    public static void main (String[]args){
        CPU cpu = new CPU();
        CPU.Processor Processor = cpu.new Processor();  //Creating objects of inner class  using outer class
        CPU.RAM RAM = cpu.new RAM();
        System.out.println("Processor Cache = " + Processor.getCache());
        System.out.println("Ram Clock speed = " + RAM.getClockSpeed());
        System.out.println("Price = " + cpu.price());
```



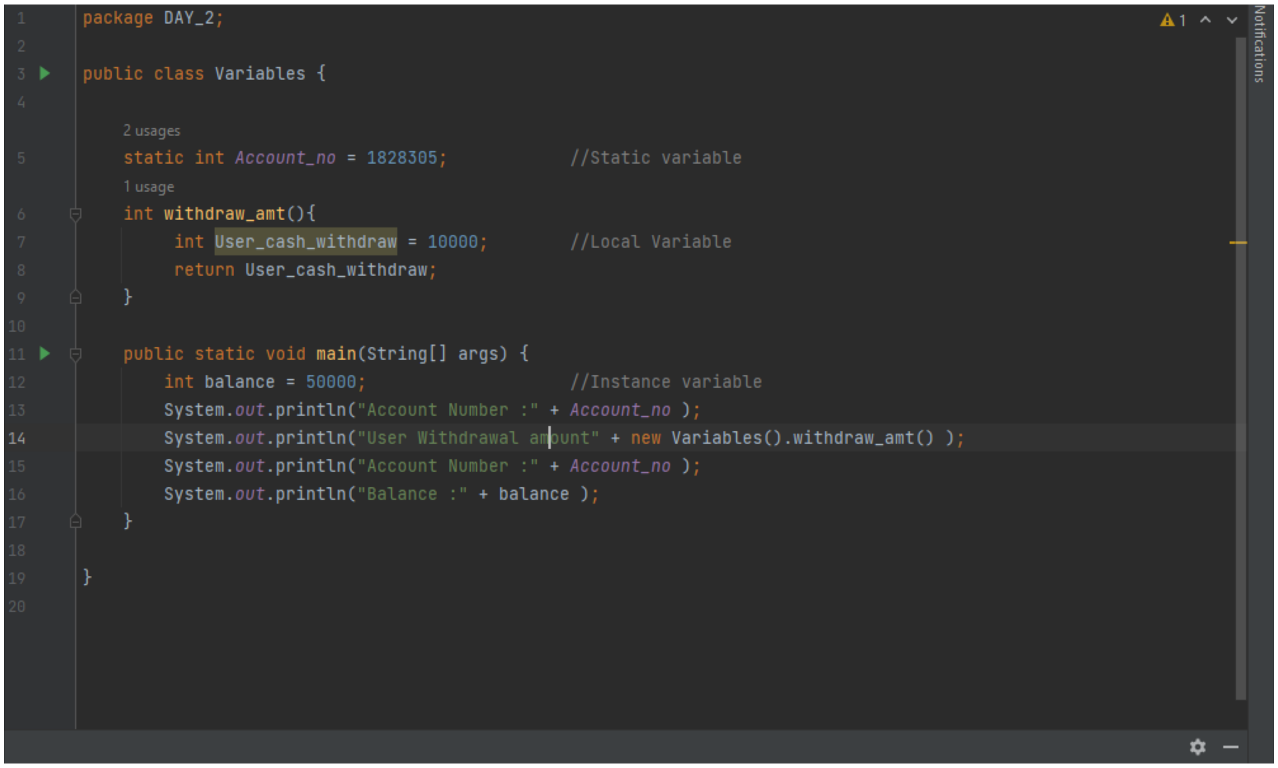
```java
        System.out.println("processor = " + Processor.cores());
        System.out.println("Ram Clock speed = " + RAM.memory());
        System.out.println("Manufacturer = " + RAM.manufacturer());
    }
}
```

OUTPUT:



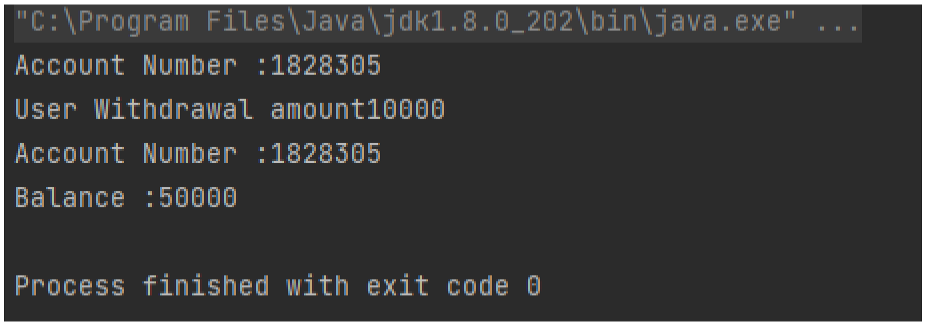
```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
Processor Cache = 4.3
Ram Clock speed = 5.5
Price = 75000.45
processor = 5.0
Ram Clock speed = 15.75
Manufacturer = Asus

Process finished with exit code 0
```

Q2) Program of local, instance and static variable



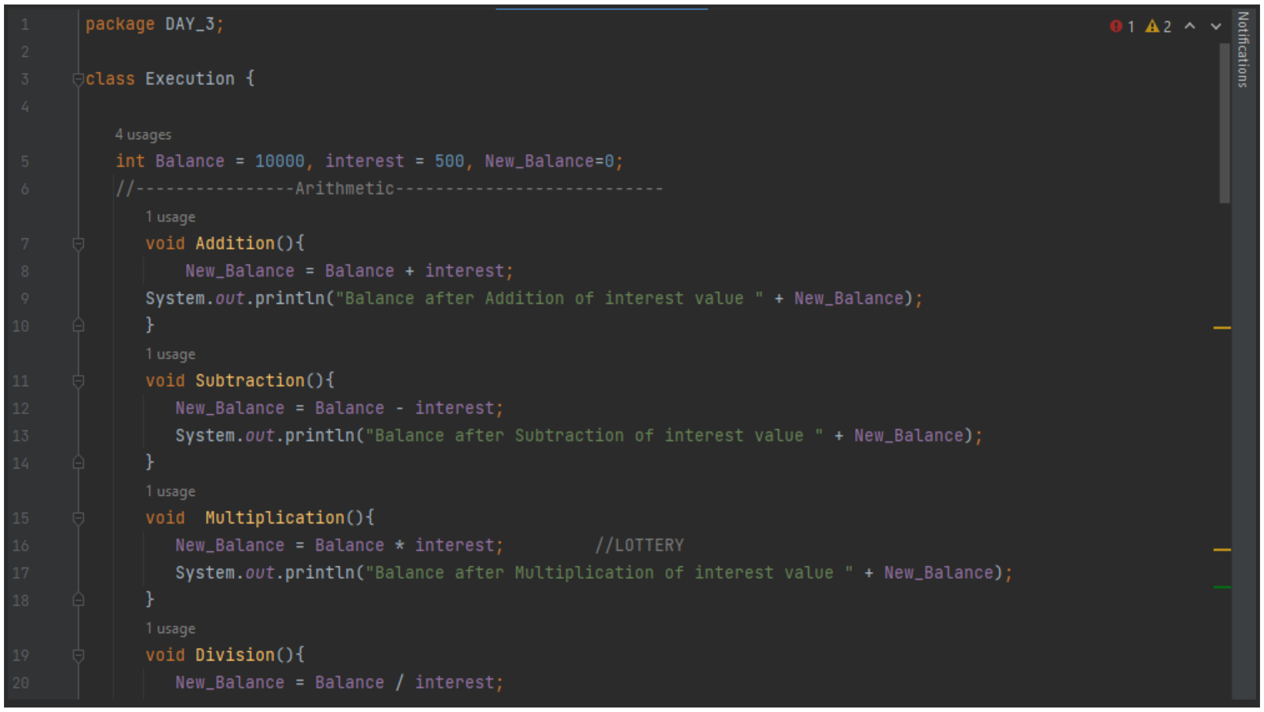
```java
package DAY_2;

public class Variables {

    2 usages
    static int Account_no = 1828305;            //Static variable
    1 usage
    int withdraw_amt(){
        int User_cash_withdraw = 10000;         //Local Variable
        return User_cash_withdraw;
    }

    public static void main(String[] args) {
        int balance = 50000;                    //Instance variable
        System.out.println("Account Number :" + Account_no );
        System.out.println("User Withdrawal amount" + new Variables().withdraw_amt() );
        System.out.println("Account Number :" + Account_no );
        System.out.println("Balance :" + balance );
    }

}
```

OUTPUT:

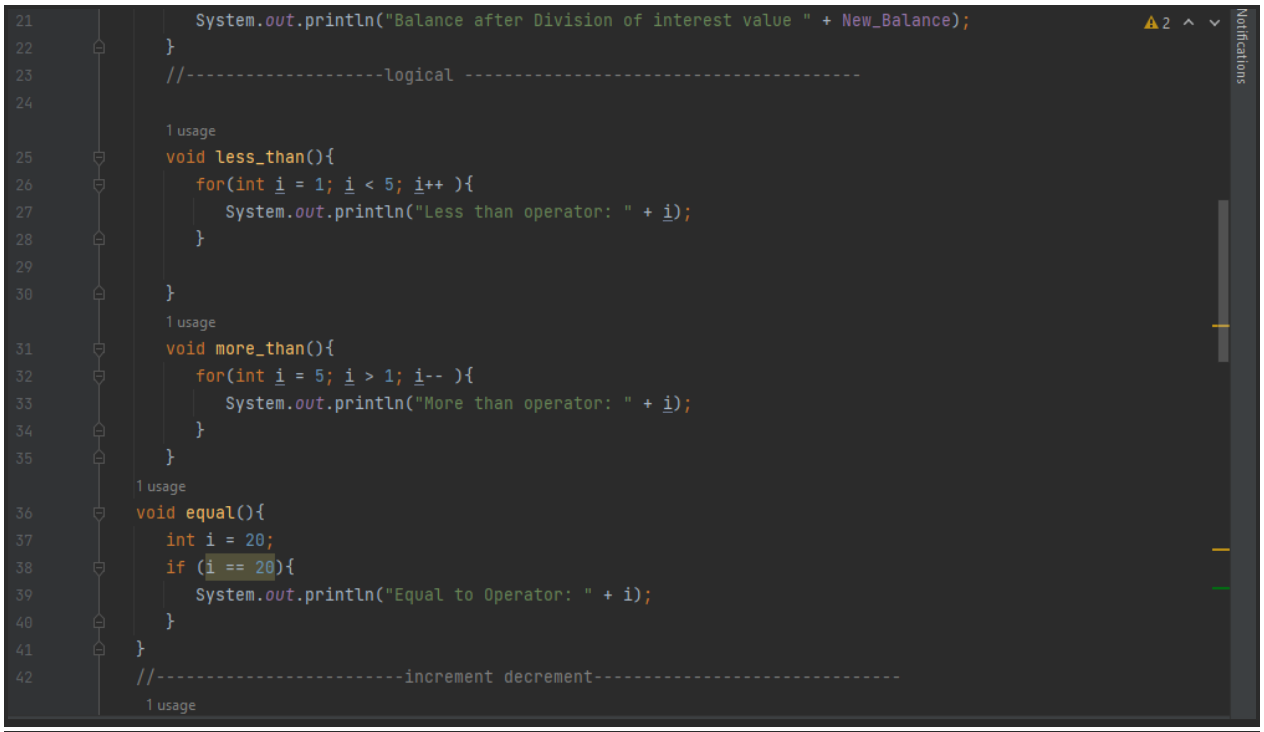


```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
Account Number :1828305
User Withdrawal amount10000
Account Number :1828305
Balance :50000

Process finished with exit code 0
```

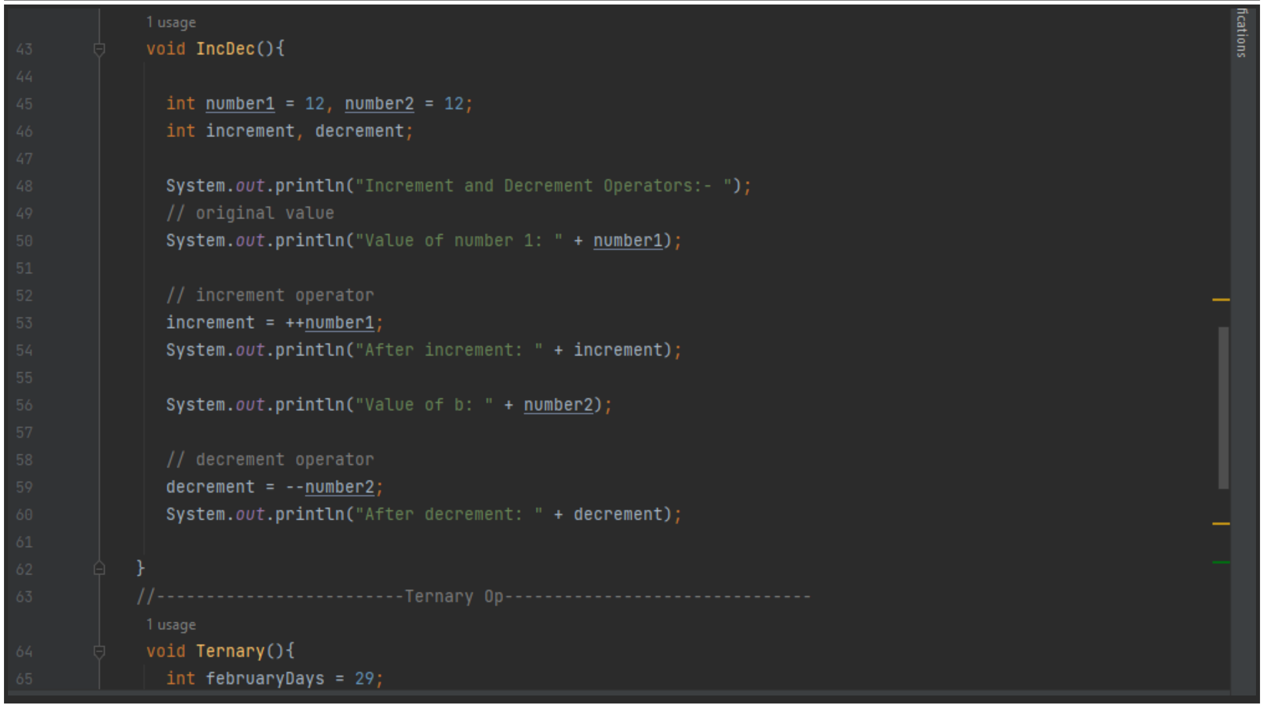
Q3) Program for types of operators



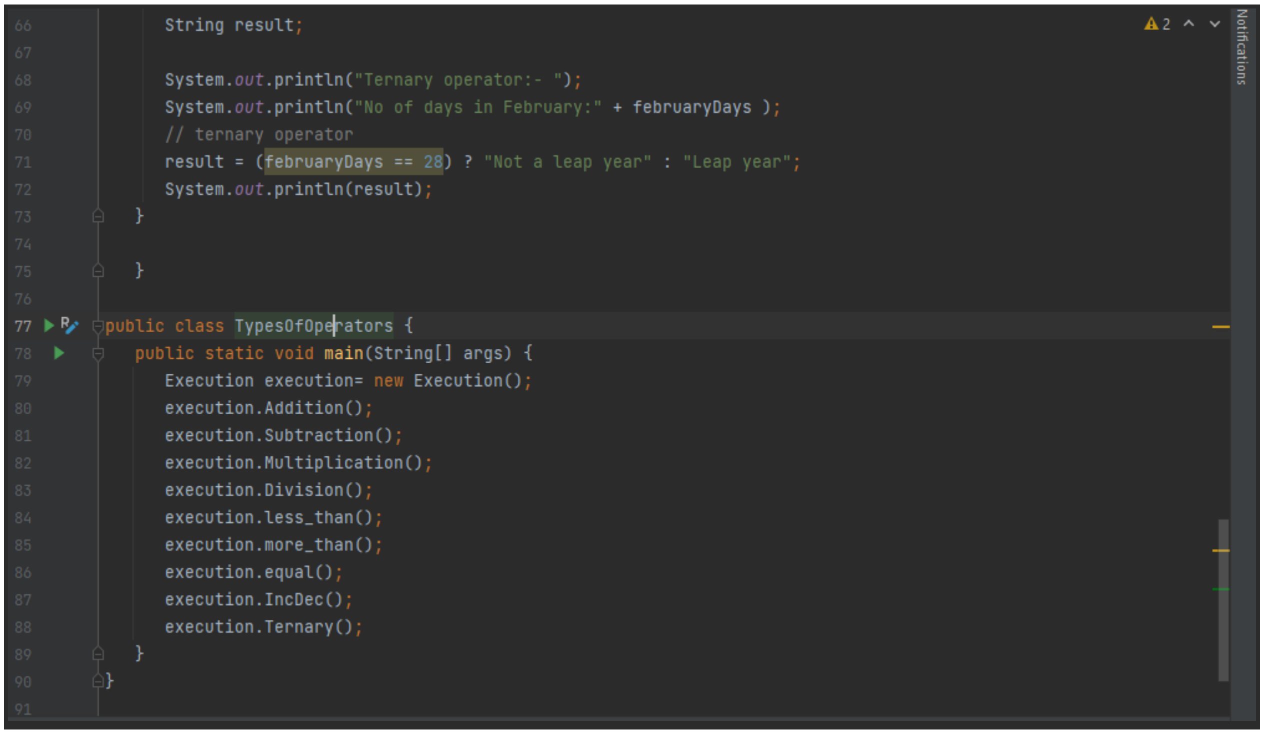
```java
package DAY_3;

class Execution {

    4 usages
    int Balance = 10000, interest = 500, New_Balance=0;
    //---------------Arithmetic--------------------------
        1 usage
        void Addition(){
            New_Balance = Balance + interest;
        System.out.println("Balance after Addition of interest value " + New_Balance);
        }
        1 usage
        void Subtraction(){
            New_Balance = Balance - interest;
            System.out.println("Balance after Subtraction of interest value " + New_Balance);
        }
        1 usage
        void  Multiplication(){
            New_Balance = Balance * interest;         //LOTTERY
            System.out.println("Balance after Multiplication of interest value " + New_Balance);
        }
        1 usage
        void Division(){
            New_Balance = Balance / interest;
```



```
        System.out.println("Balance after Division of interest value " + New_Balance);
    }
    //-------------------logical ---------------------------------------

    1 usage
    void less_than(){
        for(int i = 1; i < 5; i++ ){
            System.out.println("Less than operator: " + i);
        }

    }
    1 usage
    void more_than(){
        for(int i = 5; i > 1; i-- ){
            System.out.println("More than operator: " + i);
        }
    }
1 usage
void equal(){
    int i = 20;
    if (i == 20){
        System.out.println("Equal to Operator: " + i);
    }
}
//------------------------increment decrement------------------------------
 1 usage
```

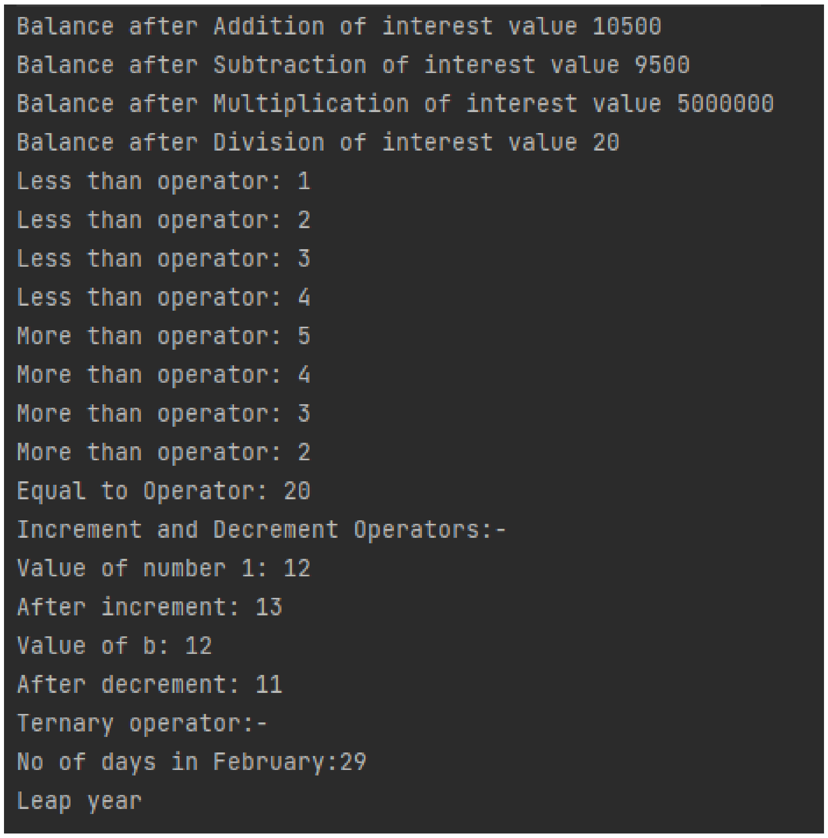


```
 1 usage
 void IncDec(){

    int number1 = 12, number2 = 12;
    int increment, decrement;

    System.out.println("Increment and Decrement Operators:- ");
    // original value
    System.out.println("Value of number 1: " + number1);

    // increment operator
    increment = ++number1;
    System.out.println("After increment: " + increment);

    System.out.println("Value of b: " + number2);

    // decrement operator
    decrement = --number2;
    System.out.println("After decrement: " + decrement);

}
//------------------------Ternary Op------------------------------
 1 usage
 void Ternary(){
    int februaryDays = 29;
```



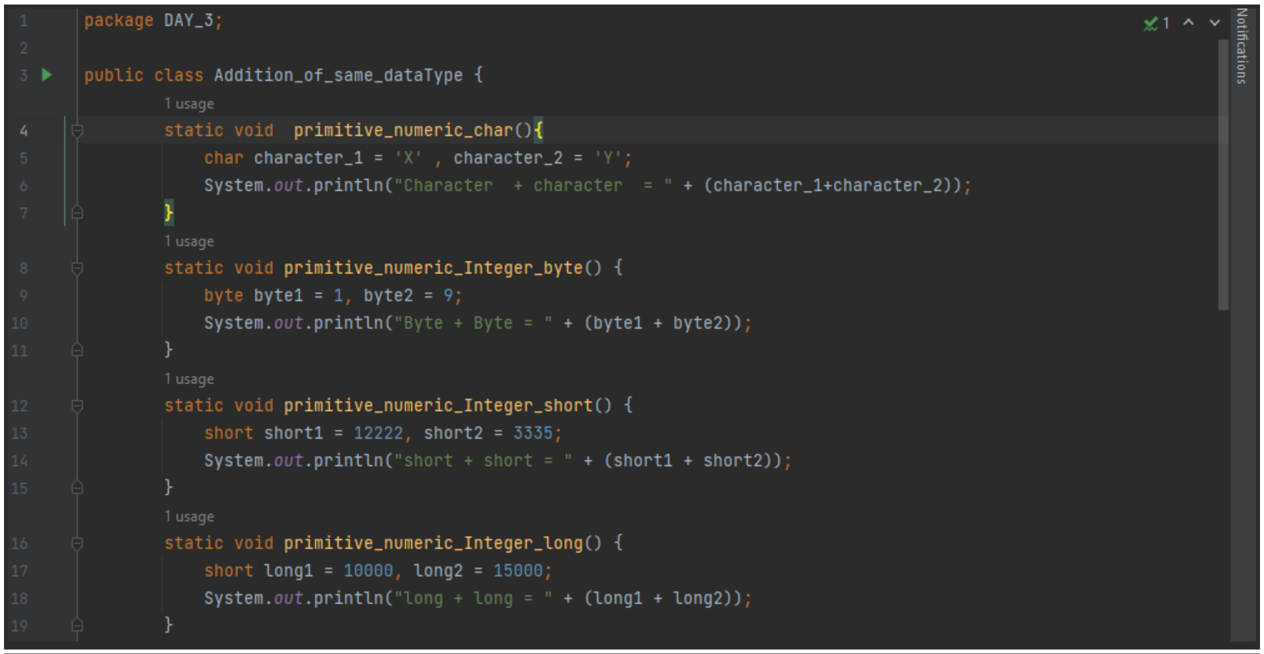
```java
            String result;

            System.out.println("Ternary operator:- ");
            System.out.println("No of days in February:" + februaryDays );
            // ternary operator
            result = (februaryDays == 28) ? "Not a leap year" : "Leap year";
            System.out.println(result);
        }

    }

public class TypesOfOperators {
    public static void main(String[] args) {
        Execution execution= new Execution();
        execution.Addition();
        execution.Subtraction();
        execution.Multiplication();
        execution.Division();
        execution.less_than();
        execution.more_than();
        execution.equal();
        execution.IncDec();
        execution.Ternary();
    }
}
```

OUTPUT:

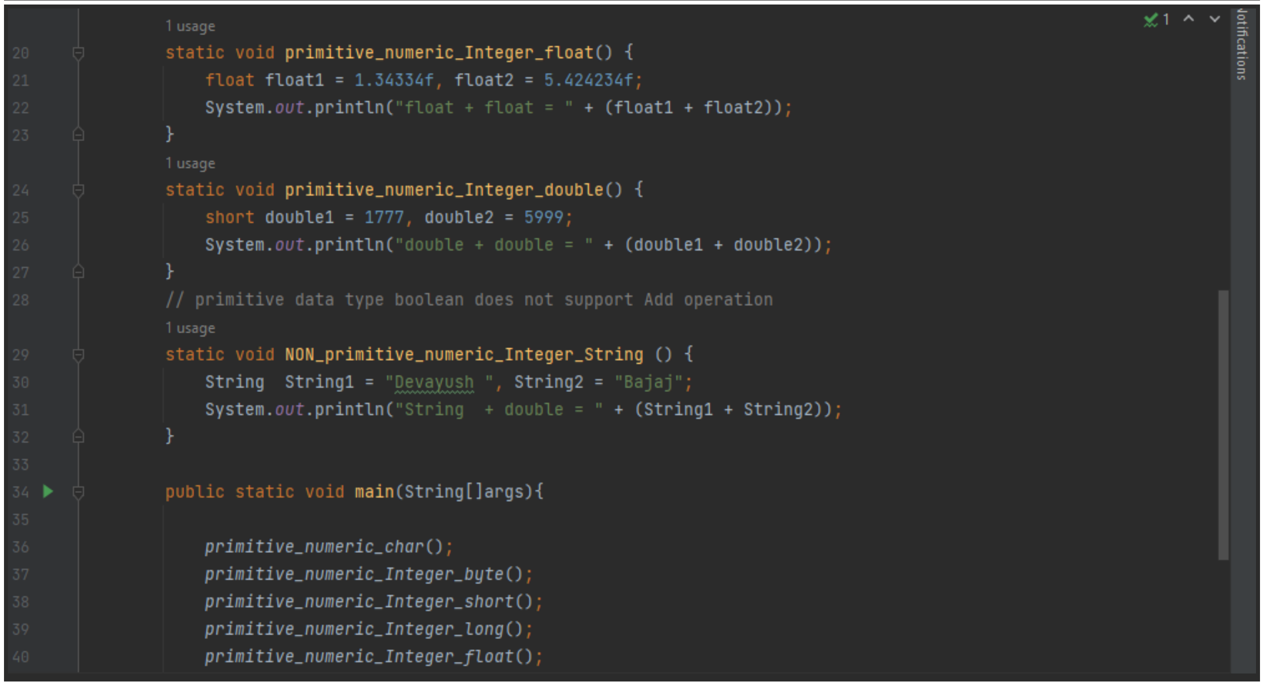


```
Balance after Addition of interest value 10500
Balance after Subtraction of interest value 9500
Balance after Multiplication of interest value 5000000
Balance after Division of interest value 20
Less than operator: 1
Less than operator: 2
Less than operator: 3
Less than operator: 4
More than operator: 5
More than operator: 4
More than operator: 3
More than operator: 2
Equal to Operator: 20
Increment and Decrement Operators:-
Value of number 1: 12
After increment: 13
Value of b: 12
After decrement: 11
Ternary operator:-
No of days in February:29
Leap year
```

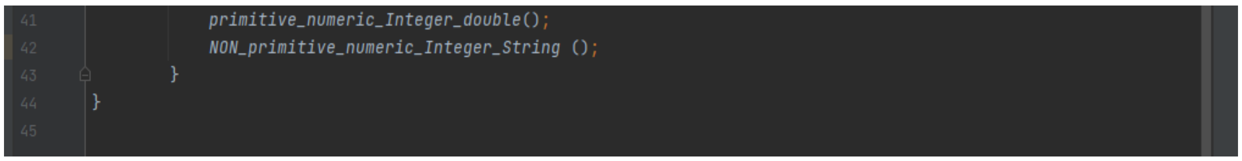
Q4) Program to add 2 same operators



```
package DAY_3;

public class Addition_of_same_dataType {
        1 usage
        static void  primitive_numeric_char(){
            char character_1 = 'X' , character_2 = 'Y';
            System.out.println("Character  + character  = " + (character_1+character_2));
        }
        1 usage
        static void primitive_numeric_Integer_byte() {
            byte byte1 = 1, byte2 = 9;
            System.out.println("Byte + Byte = " + (byte1 + byte2));
        }
        1 usage
        static void primitive_numeric_Integer_short() {
            short short1 = 12222, short2 = 3335;
            System.out.println("short + short = " + (short1 + short2));
        }
        1 usage
        static void primitive_numeric_Integer_long() {
            short long1 = 10000, long2 = 15000;
            System.out.println("long + long = " + (long1 + long2));
        }
```

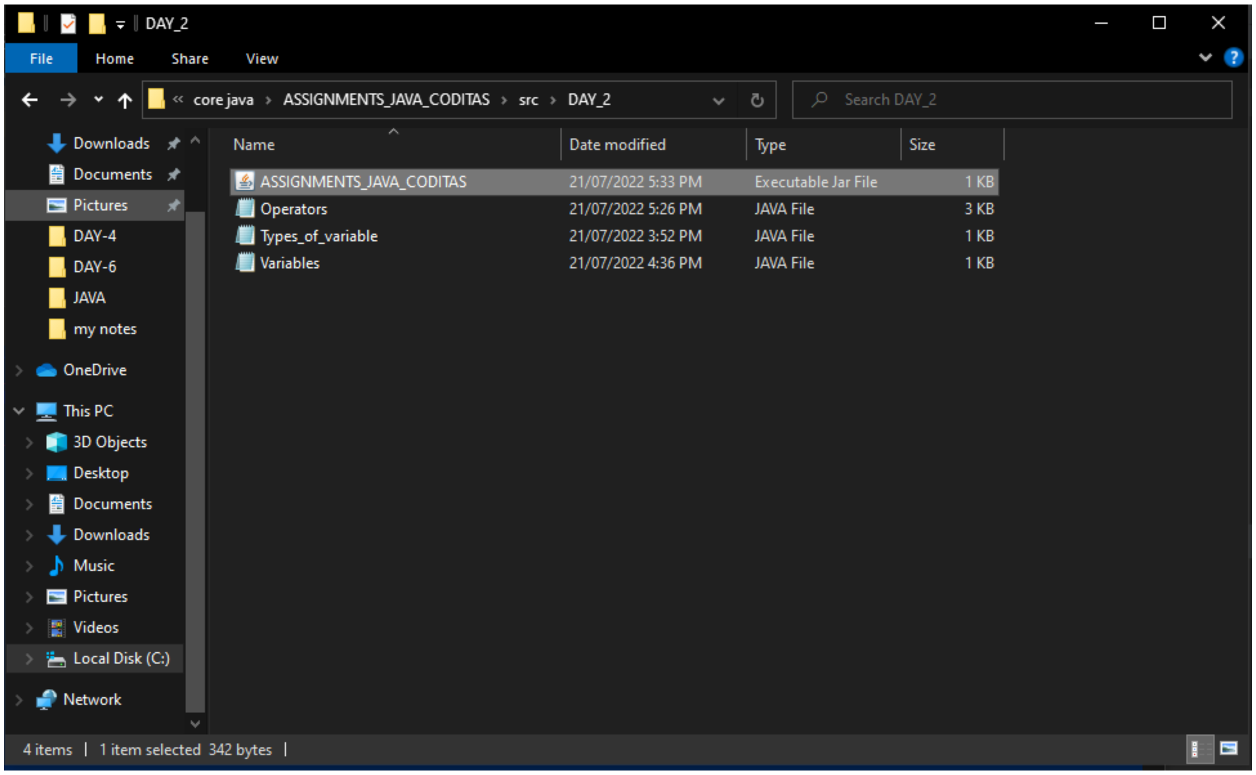


```
        1 usage
        static void primitive_numeric_Integer_float() {
            float float1 = 1.34334f, float2 = 5.424234f;
            System.out.println("float + float = " + (float1 + float2));
        }
        1 usage
        static void primitive_numeric_Integer_double() {
            short double1 = 1777, double2 = 5999;
            System.out.println("double + double = " + (double1 + double2));
        }
        // primitive data type boolean does not support Add operation
        1 usage
        static void NON_primitive_numeric_Integer_String () {
            String  String1 = "Devayush ", String2 = "Bajaj";
            System.out.println("String  + double = " + (String1 + String2));
        }

        public static void main(String[]args){

            primitive_numeric_char();
            primitive_numeric_Integer_byte();
            primitive_numeric_Integer_short();
            primitive_numeric_Integer_long();
            primitive_numeric_Integer_float();
```



```
            primitive_numeric_Integer_double();
            NON_primitive_numeric_Integer_String ();
        }
}
```

Q5) Create Jar file



Name	Date modified	Type	Size
ASSIGNMENTS_JAVA_CODITAS	21/07/2022 5:33 PM	Executable Jar File	1 KB
Operators	21/07/2022 5:26 PM	JAVA File	3 KB
Types_of_variable	21/07/2022 3:52 PM	JAVA File	1 KB
Variables	21/07/2022 4:36 PM	JAVA File	1 KB

4 items | 1 item selected 342 bytes |