OOP Practicals

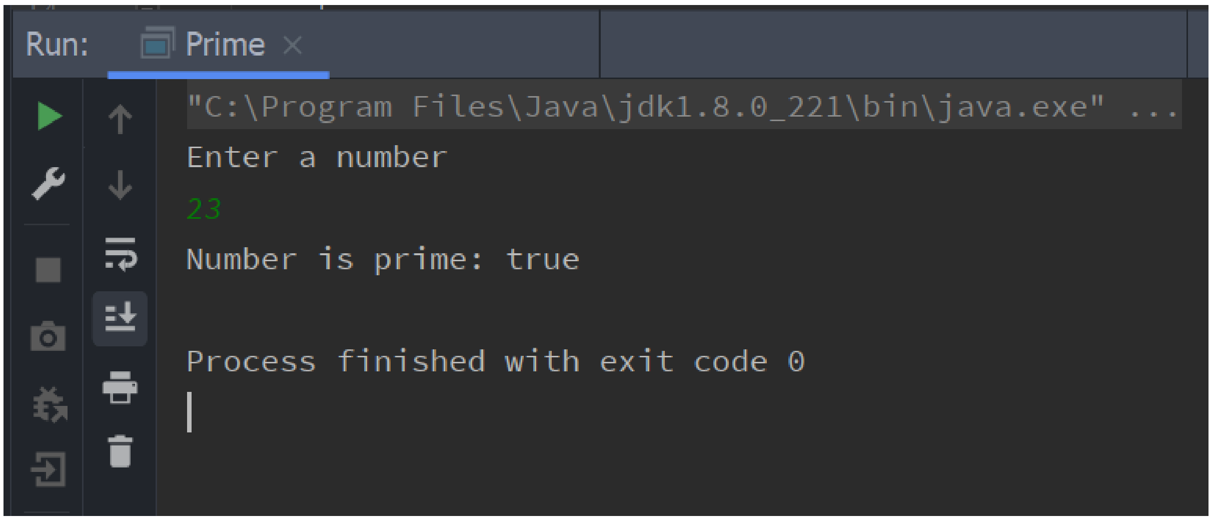
Name: Samruddhi Anil Ghodake **Enroll:** BE21S05F005

Practical no. 1: Program to check whether a given number is prime or not.

Program

```
//1 : Program to check whether a given number is prime or not.
import java.util.Scanner;

public class Prime {
    public static boolean prime(int num1){
        if(num1<2){
            return false;
        }
        for (int i = 2; i < Math.sqrt(num1); i++) {
            if(num1%i==0){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        int num;
        System.out.println("Enter a number");
        Scanner sc = new Scanner(System.in);
        num = sc.nextInt();
        boolean ans = prime(num);
        System.out.println("Number is prime: "+ ans);
    }
}
```

Output



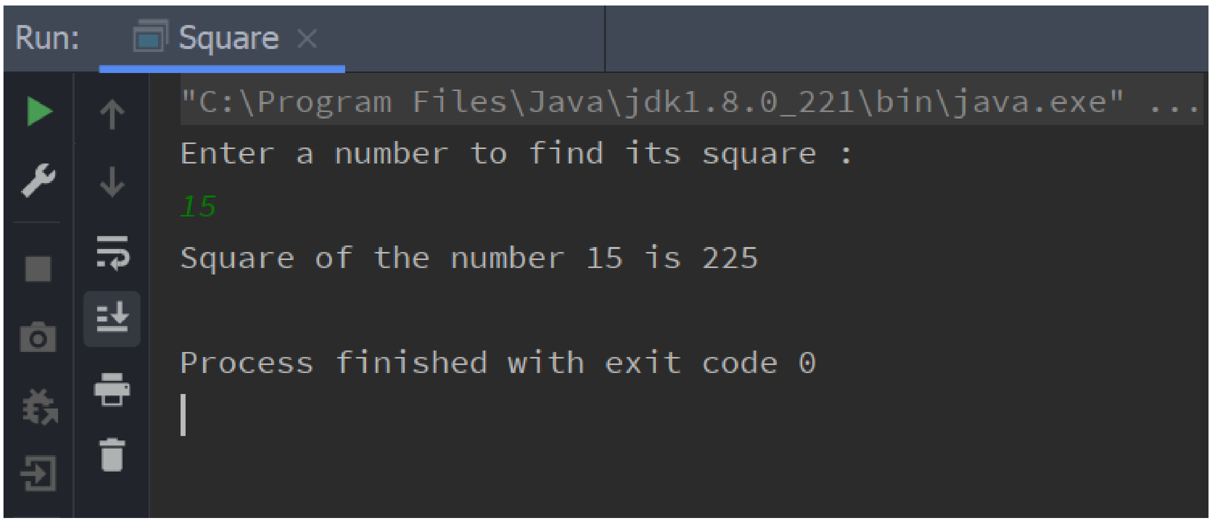
```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...
Enter a number
23
Number is prime: true

Process finished with exit code 0
```

Practical no. 2: Program to find square of a given number.

Program

```
//Practical – 2 : Program to find square of a given number.
import java.util.Scanner;

public class Square {
    public static void main(String[] args) {
        int num, square;
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter a number to find its square : ");
        num = sc.nextInt();

        square = num * num;

        System.out.println("Square of the number " + num + " is " + square);

    }
}
```

Output



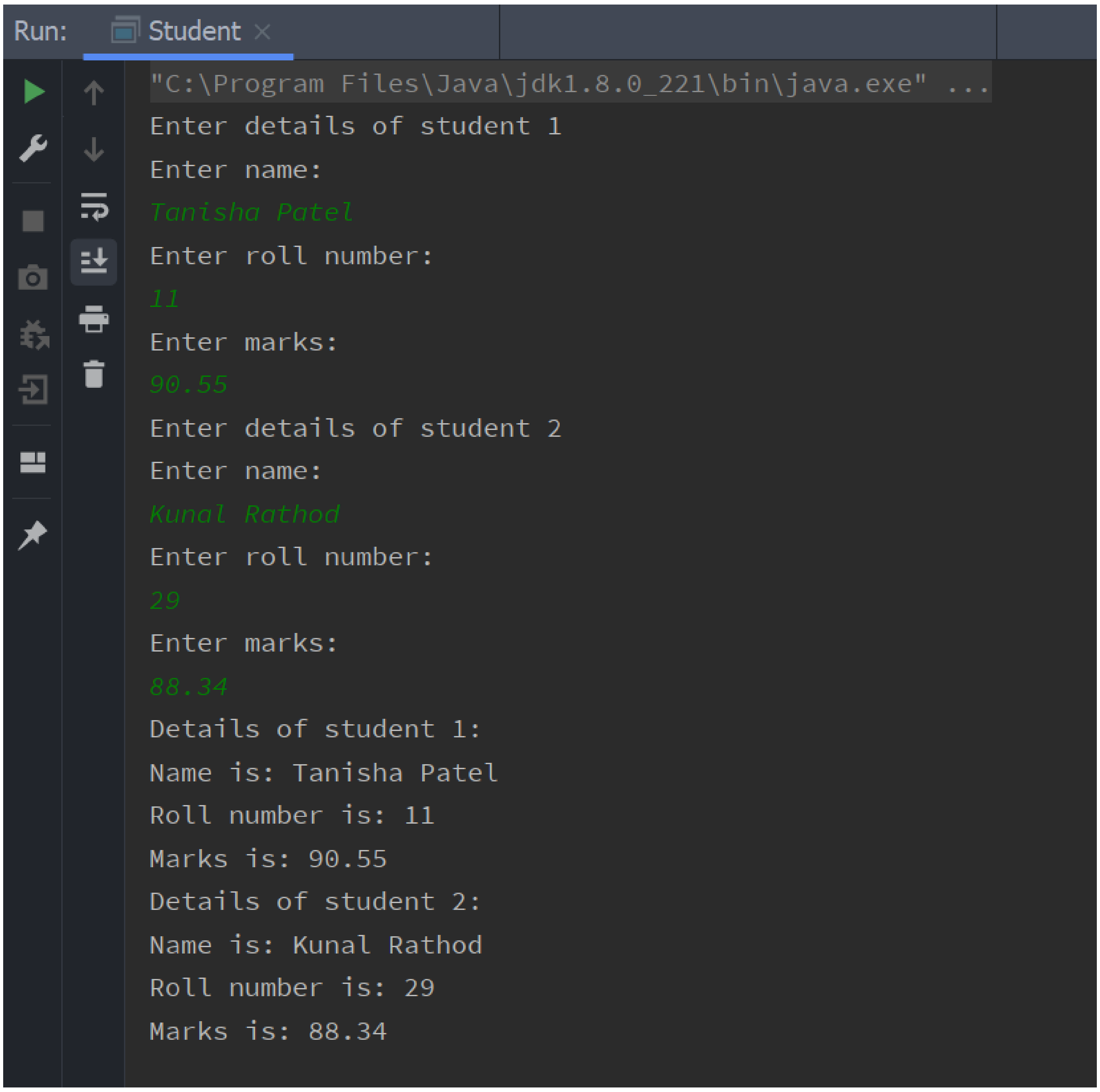
```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...
Enter a number to find its square :
15
Square of the number 15 is 225

Process finished with exit code 0
```

Practical no. 3: Program for classes and objects.

Program

```
//3 : Program for  classes and objects.
import java.util.Scanner;

public class Student {
    int roll;
    String name;
    float marks;

    public void getDetails(){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter name: ");
        name= sc.nextLine();
        System.out.println("Enter roll number: ");
        roll = sc.nextInt();
        System.out.println("Enter marks: ");
        marks=sc.nextFloat();
    }
    public void showDetails(){
        System.out.println("Name is: "+name);
        System.out.println("Roll number is: "+roll);
        System.out.println("Marks is: "+marks);
    }
    public static void main(String[] args) {
        Student s1 = new Student();
        Student s2  = new Student();
        System.out.println("Enter details of student 1");
        s1.getDetails();
        System.out.println("Enter details of student 2");
        s2.getDetails();
        System.out.println("Details of student 1: ");
        s1.showDetails();
        System.out.println("Details of student 2: ");
        s2.showDetails();
    }
}
```

Output



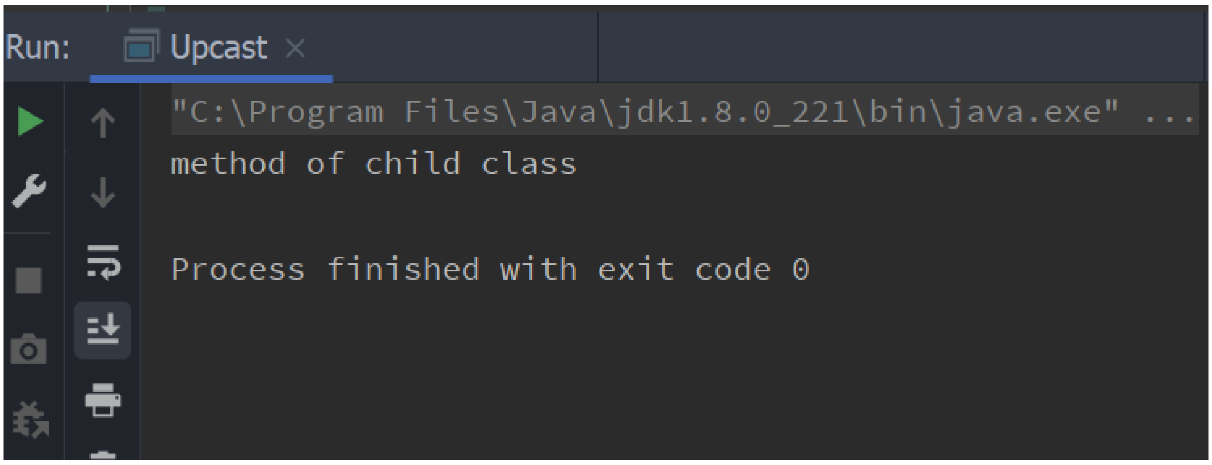
```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...
Enter details of student 1
Enter name:
Tanisha Patel
Enter roll number:
11
Enter marks:
90.55
Enter details of student 2
Enter name:
Kunal Rathod
Enter roll number:
29
Enter marks:
88.34
Details of student 1:
Name is: Tanisha Patel
Roll number is: 11
Marks is: 90.55
Details of student 2:
Name is: Kunal Rathod
Roll number is: 29
Marks is: 88.34
```

Practical no. 4: Program of inheritance with up casting.

Porgram

```
//4.Program of inheritance with up casting.
class  Parent{
    void PrintData() {
        System.out.println("method of parent class");
    }
}

class Child extends Parent {
    void PrintData() {
        System.out.println("method of child class");
    }
}
public class Upcast{
    public static void main(String args[]) {

        Parent obj1 =  new Child(); // referencing child class object to parent
class
        obj1.PrintData(); //here method of child class gets executed
    }
}
```

Output



```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...
method of child class

Process finished with exit code 0
```

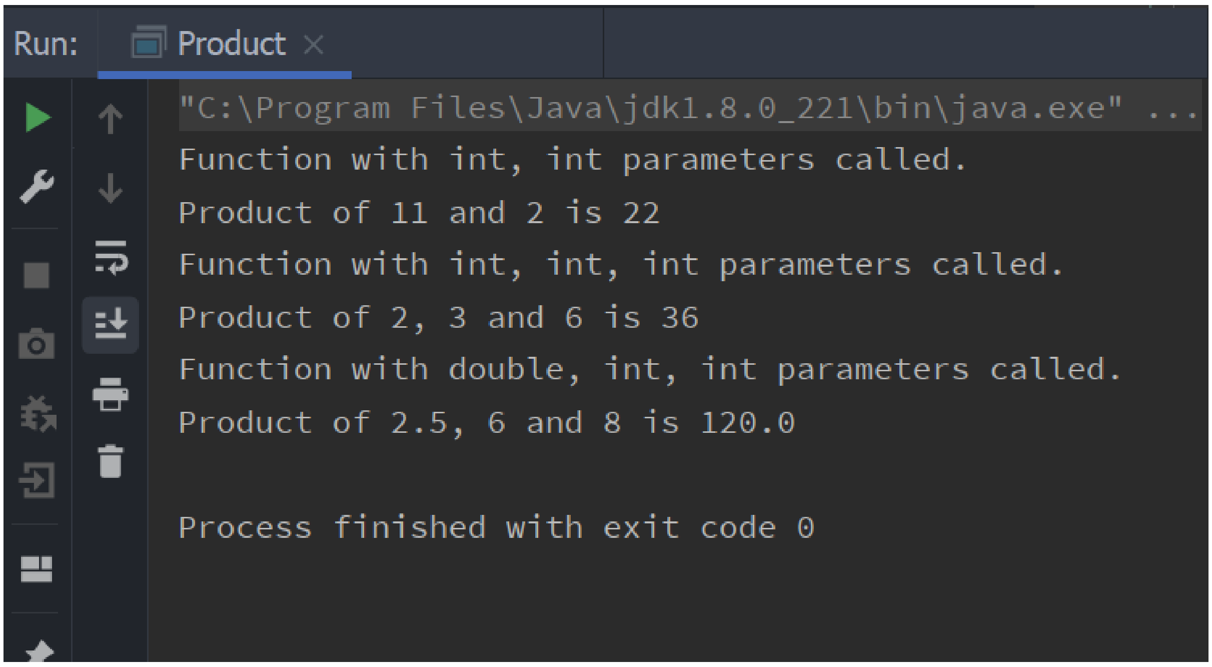
Program no. 5: Program for overloading

Program

```
//5 : Program for function overloading
public class Product {
   int product(int a, int b)
   {
      System.out.println("Function with int, int parameters called.");
      int res;
      res = a * b;
      return res;
   }
   int product(int a, int b, int c)
   {
      System.out.println("Function with int, int, int parameters called.");
      int res;
      res = a * b * c;
      return res;
   }
   double product(double a, int b, int c)
   {
      System.out.println("Function with double, int, int parameters called.");
      double res;
      res = a * b * c;
      return res;
   }

   public static void main(String[] args) {

      Product o1 = new Product();
      int prod1, prod2;
      double prod3;

      prod1 = o1.product(10,20);
      System.out.println("Product of 10 and 20 is " + prod1);

      prod2 = o1.product(2,3,6);
      System.out.println("Product of 2, 3 and 6 is " + prod2);
```

```
        prod3 = o1.product(2.5,6,8);
        System.out.println("Product of 2.5, 6 and 8 is " + prod3);
    }
}
```

Output



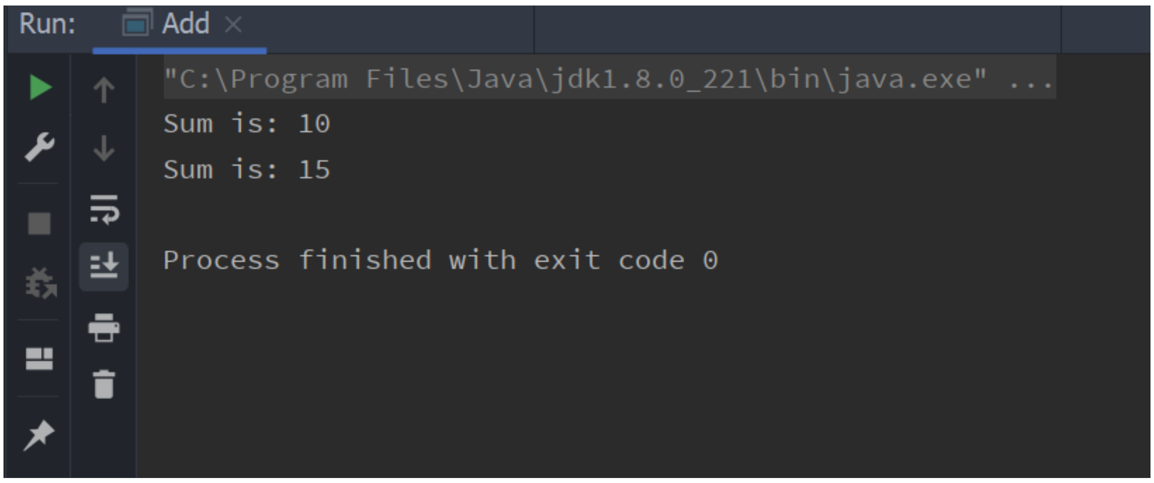
```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...
Function with int, int parameters called.
Product of 11 and 2 is 22
Function with int, int, int parameters called.
Product of 2, 3 and 6 is 36
Function with double, int, int parameters called.
Product of 2.5, 6 and 8 is 120.0

Process finished with exit code 0
```

Practical no. 6: Program for parameterized constructor.

Program

```
//6.program for parametrized constructor
public class Add {
    //parameterized constructor with 2 parameters
    public Add(int a, int b) {
        int s = a+b;
        System.out.println("Sum is: "+s);
    }
    //parameterized constructor with 3 parameters
    public Add(int a, int b, int c) {
        int s = a+b+c;
        System.out.println("Sum is: "+s);
    }
    public static void main(String[] args) {
    Add obj1 = new Add(5, 5); //invokes the constructor with 2 parameters
    Add obj2 = new Add(5, 5, 5); //invokes the constructor with 3 parameters
    }
}
```

Output



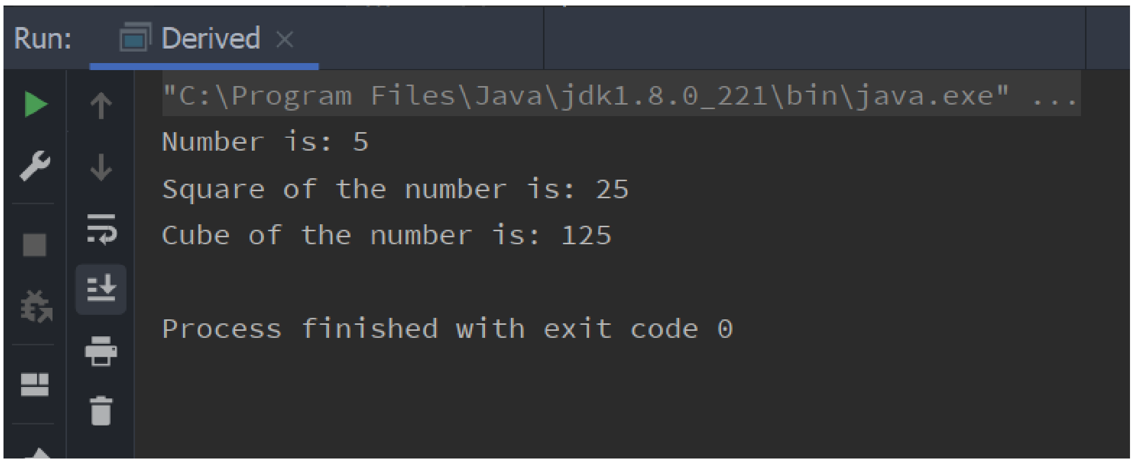
```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...
Sum is: 10
Sum is: 15

Process finished with exit code 0
```

Practical no. 7: Program for single inheritance

Program

```
//7. program for single inheritance
class Base{
    int num1=5;
    public void square(){
        int s = num1*num1;
        System.out.println("Square of the number is: "+ s);
    }
}
//derived class: here we are extending base class members into derived class
public class Derived extends Base{
    void cube(){
        int c = num1*num1*num1; //accessing the data member of base class
        System.out.println("Cube of the number is: "+ c);
    }
    public static void main(String[] args) {
        Derived d= new Derived(); //making a object of derived class
        System.out.println("Number is: "+d.num1); //accessing data member of
base class through derived class object
        d.square(); //calling method of base class through the object of derived
class
        d.cube();
    }
}
```

Output



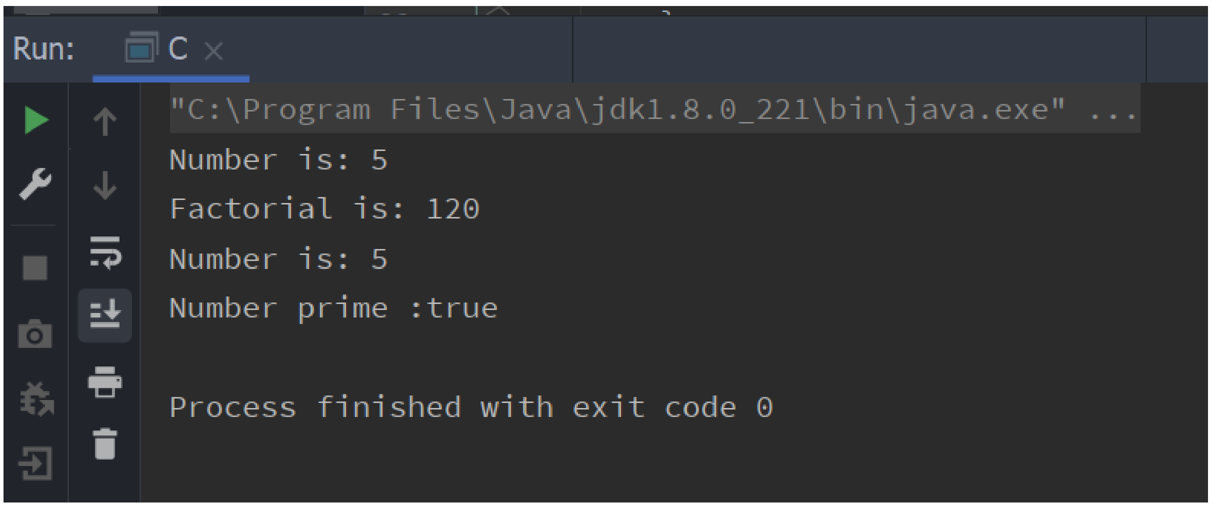
```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...
Number is: 5
Square of the number is: 25
Cube of the number is: 125

Process finished with exit code 0
```

Practical no. 8: Program for hierarchical inheritance

Program

```
//8.program for hierarchical interface

//base class
public class A {
    int num1=5;
    public void display(){
        System.out.println("Number is: "+num1);
    }
}

//derived class 1
class B extends A{
    public boolean prime(){
        if(num1<2){
           return false;
        }
        for (int i = 2; i < Math.sqrt(num1); i++) {
            if(num1%i==0){
                return false;
            }
        }
        return true;
    }
}

//derived class 2
class C extends A{
    int f=1;
    int factorial(){
        for (int i = 1; i <=num1 ; i++) {
            f=f*i;
        }
        return f;
    }

    public static void main(String[] args) {
        C obj = new C(); //object of derived class 2
        obj.display(); //accessing method of base class through derived class
object
        int f = obj.factorial();
        System.out.println("Factorial is: "+f);

        B obj2 = new B(); //object of derived class 1
        obj2.display(); //accessing method of base class through derived class
object
        boolean ans = obj2.prime();
        System.out.println("Number prime :"+ ans);
    }
}
```

Output



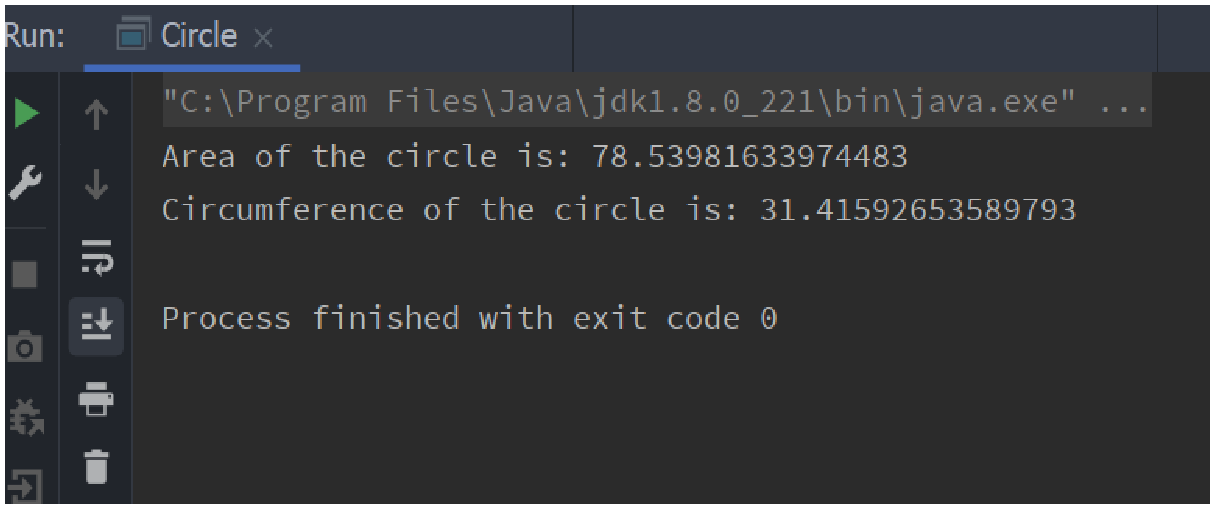
```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...
Number is: 5
Factorial is: 120
Number is: 5
Number prime :true

Process finished with exit code 0
```

Practical no. 9: Program for multiple inheritance using interface.

Program

```
//9.program for multiple inheritance using interface.

//interface 1
interface Circumference{
   public void circumference();
}

//interface 2
interface Area{
   public void area();
}

//class implementing two interfaces to demonstrate multiple inheritance
public class Circle implements Area, Circumference {
   int r=5;
   @Override
   public void circumference() {
      double c = 2*Math.PI*r;
      System.out.println("Circumference of the circle is: "+c);
   }

   @Override
   public void area() {
      double a = Math.PI*r*r;
      System.out.println("Area of the circle is: "+ a);
   }

   //main
   public static void main(String[] args) {
      Circle cir = new Circle();
      cir.area();
      cir.circumference();
   }
}
```

Output



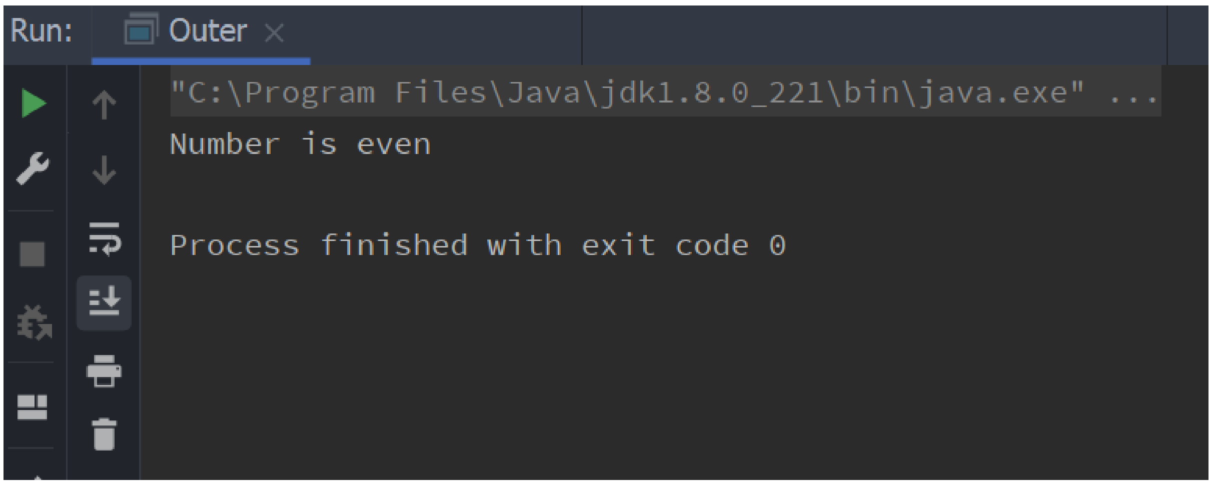
```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...
Area of the circle is: 78.53981633974483
Circumference of the circle is: 31.41592653589793

Process finished with exit code 0
```

Practical no. 10: Program of any nested class.

Program

```
//10. program of any nested class
public class Outer {
    int num=10;

    class Inner{
        public void fun(){
            if(num%2==0){
                System.out.println("Number is even");
            }
            else{
                System.out.println("Number is odd");
            }
        }
    }

    public static void main(String[] args) {
        Outer o = new Outer(); //making the object of the outer class
        Outer.Inner i = o.new Inner(); //making the object of inner class through
the object of outer class
        i.fun(); //calling the method

    }
}
```

Output



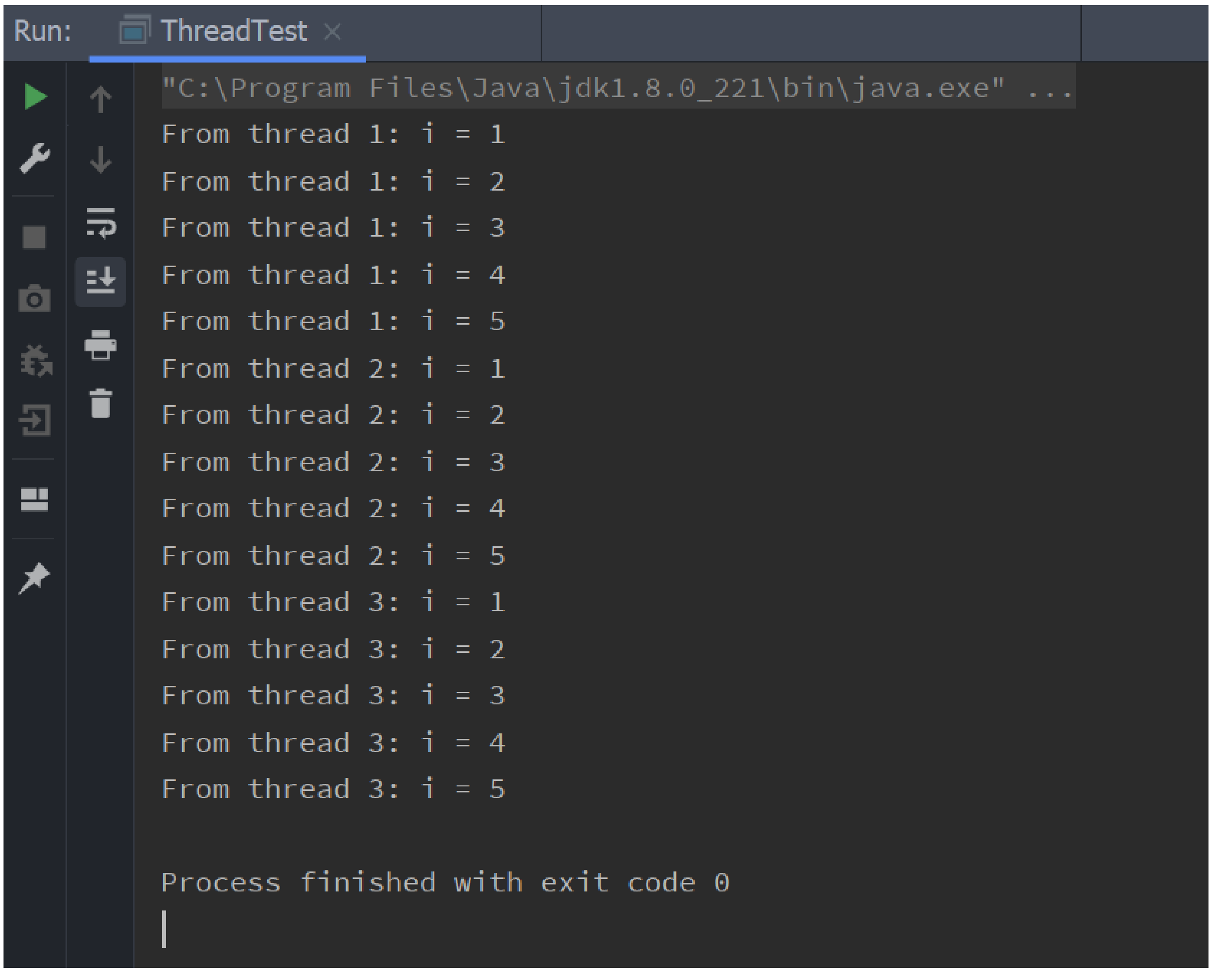
```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...
Number is even

Process finished with exit code 0
```

Practical no. 11: Program for multi-threading.

Program

```
//11.program for multi-threading

class ThreadDemo1 extends  Thread{
    public  void run(){
        for (int i = 1; i <=5 ; i++) {
            System.out.println("From thread 1: i = "+i);
        }
    }
}
class ThreadDemo2 extends  Thread{
    public  void run(){
        for (int j = 1; j <=5 ; j++) {
            System.out.println("From thread 2: i = "+j);
        }
    }
}class ThreadDemo3 extends  Thread{
    public  void run(){
        for (int k = 1; k <=5 ; k++) {
            System.out.println("From thread 3: i = "+k);
        }
    }
}

public class ThreadTest {
    public static void main(String[] args) {
        ThreadDemo1 t1 = new ThreadDemo1();
        t1.start(); //starting the thread 1
        ThreadDemo2 t2 = new ThreadDemo2();
        t2.start(); //starting the thread 2
        ThreadDemo3 t3 = new ThreadDemo3();
        t3.start(); //starting the thread 3
    }
}
```

Output



```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...
From thread 1: i = 1
From thread 1: i = 2
From thread 1: i = 3
From thread 1: i = 4
From thread 1: i = 5
From thread 2: i = 1
From thread 2: i = 2
From thread 2: i = 3
From thread 2: i = 4
From thread 2: i = 5
From thread 3: i = 1
From thread 3: i = 2
From thread 3: i = 3
From thread 3: i = 4
From thread 3: i = 5

Process finished with exit code 0
```