

# **OBJECT ORIENTED PROGRAMMING LAB**

**SUBMITTED BY,**

**BLESSY P ROY**

**S2 MCA**

**ROLL NO:MCA214**

## **LABCYCLE 3**

## **PROGRAM:1**

**AIM:** Area of different shapes using overloaded functions

### **ALGORITHM**

STEP 1: Create a class called area

STEP 2: Create 3 member functions to calculate the area of square, rectangle and circle

STEP 3: Perform the area finding operations inside the functions

STEP 4: create object of the class area

STEP 5: call the functions which is created using objects

STEP 6: print the values of area of each shape

STEP 7: Stop

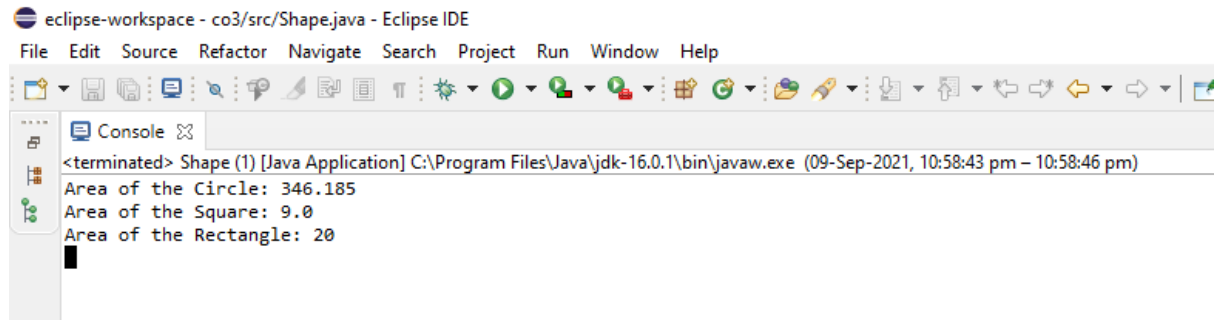
### **SOURCE CODE**

Shape.java	<pre>class overload {     void Area(float a)     {         float A = a * a;         System.out.println("Area of the Square: " + A);     }      void Area(double a)     {         double A = 3.14 * a * a;         System.out.println("Area of the Circle: " + A);     }      void Area(int a, int b)     {         int A = a * b;         System.out.println("Area of the Rectangle: " + A);     } } class Shape {     public static void main(String[] args)     {         overload obj = new overload();         obj.Area(10.5);         obj.Area(3);         obj.Area(5, 4);     } }</pre>
------------	---

## RESULT

The above program is executed and obtained the output.

## OUTPUT

A screenshot of the Eclipse IDE's console window. The title bar reads 'eclipse-workspace - co3/src/Shape.java - Eclipse IDE'. The menu bar includes 'File', 'Edit', 'Source', 'Refactor', 'Navigate', 'Search', 'Project', 'Run', 'Window', and 'Help'. The toolbar contains various icons for file operations, search, and execution. The console tab is active, showing the following output:

```
<terminated> Shape (1) [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (09-Sep-2021, 10:58:43 pm - 10:58:46 pm)
Area of the Circle: 346.185
Area of the Square: 9.0
Area of the Rectangle: 20
█
```

## **PROGRAM: 2**

**AIM:** Create a class 'Employee' with data members Empid, Name, Salary, Address and constructors to initialize the data members. Create another class 'Teacher' that inherits the properties of class employee and contains its own data members department, Subjects taught and constructors to initialize these data members and also include a display function to display all the data members. Use an array of objects to display details of N teachers.

### **ALGORITHM**

STEP 1: Create a class employee

STEP 2: Create data members empid, name salary, address

STEP 3: Create a constructor to initialize these data members

STEP 4: Create another class teacher which is inherited from the class employee

STEP 5: Create teacher class's data members and initialize it with constructor

STEP 6: Create function to display all data members

STEP 7: Create array of objects

STEP 8: call the display function to print all the data members

STEP 9: Stop

### **SOURCE CODE**

Employee.java	<pre>import java.util.Scanner; public class Employee {     int empId;     double salary;     String name,address;     public Employee(int empId,double salary,String name,String address)     {         this.empId=empId;         this.salary=salary;         this.name=name;         this.address=address;     } }  class Teacher extends Employee {     String department,subject;     public Teacher(int empId,double salary,String name,String address,String department,String subject)     {</pre>
---------------	--

```

        super(empId,salary,name,address);
        this.department=department;
        this.subject=subject;
    }
    void display()
    {

        System.out.println("\nEMPID\t\tSALARY\t\tNAME\t\tADDRESS\t\t
        DEPARTMENT\t\tSUBJECT\n");

        System.out.println(empId+"\t\t"+salary+"\t\t"+name+"\t\t"+address+"\t\t
        "+department+"\t\t"+subject+"\n");
    }


    public static void main(String[] args){
        Scanner s=new Scanner(System.in);
        System.out.println("Enter the total number of records to be stored
: ");
        int count=s.nextInt();
        Teacher t[]=new Teacher[count];
        int i;
        for(i=0;i<count;i++)
        {
            System.out.println("Enter
empid,salary,name,address,department and subject");
            int p=s.nextInt();
            double a=s.nextDouble();
            String b=s.nextLine();
            String c=s.nextLine();
            String d=s.nextLine();
            String e=s.nextLine();
            t[i]=new Teacher(p,a,b,c,d,e);

        }
        for(i=0;i<count;i++)
        {
            t[i].display();
        }

    }

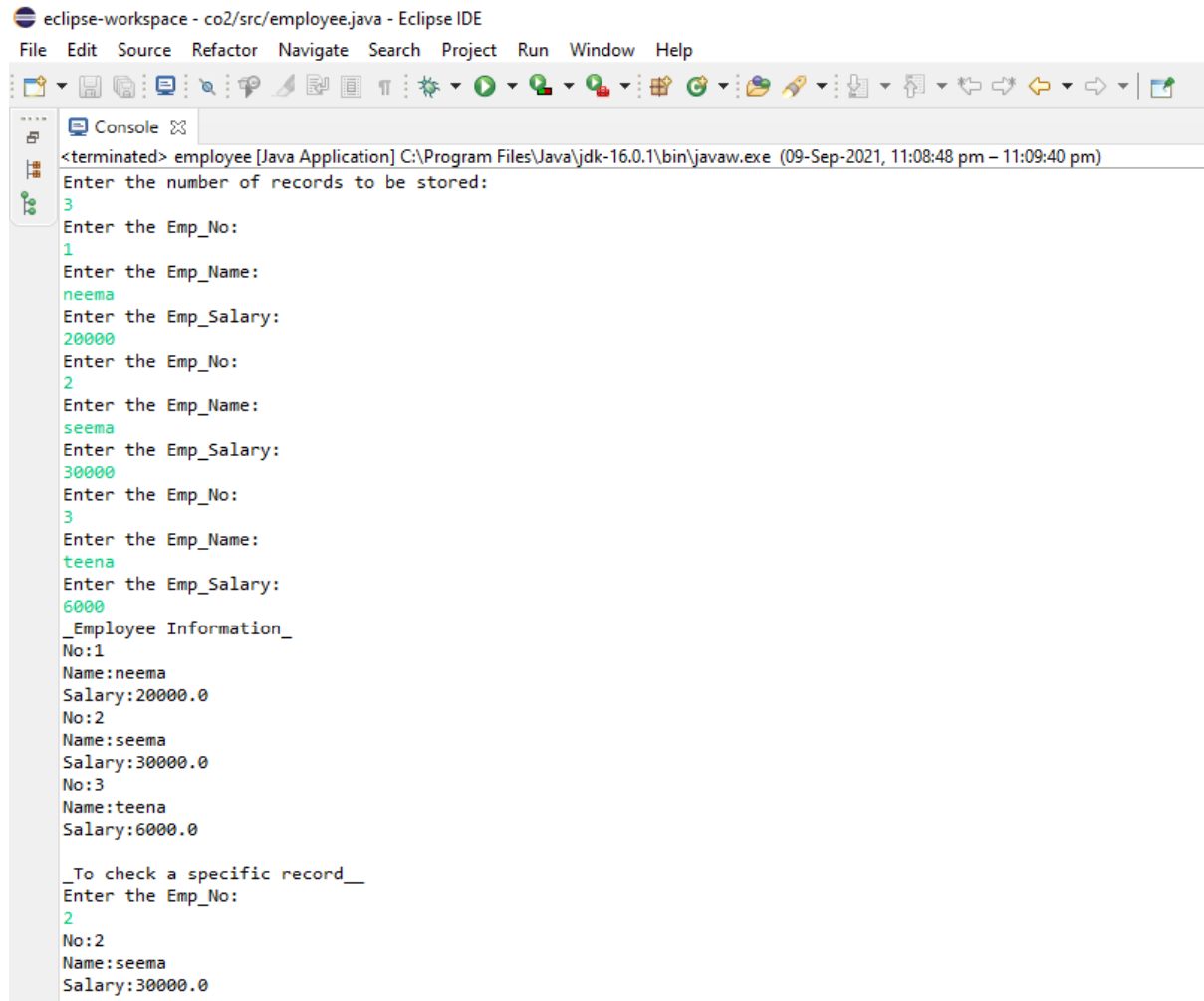
}

```

## RESULT

The above program is executed and obtained the output.

## OUTPUT



The screenshot shows the Eclipse IDE interface with the console window open. The title bar reads 'eclipse-workspace - co2/src/employee.java - Eclipse IDE'. The menu bar includes 'File', 'Edit', 'Source', 'Refactor', 'Navigate', 'Search', 'Project', 'Run', 'Window', and 'Help'. The toolbar contains various icons for file operations, search, and development. The console output is as follows:

```
<terminated> employee [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (09-Sep-2021, 11:08:48 pm - 11:09:40 pm)
Enter the number of records to be stored:
3
Enter the Emp_No:
1
Enter the Emp_Name:
neema
Enter the Emp_Salary:
20000
Enter the Emp_No:
2
Enter the Emp_Name:
seema
Enter the Emp_Salary:
30000
Enter the Emp_No:
3
Enter the Emp_Name:
teena
Enter the Emp_Salary:
6000
_Employee Information_
No:1
Name:neema
Salary:20000.0
No:2
Name:seema
Salary:30000.0
No:3
Name:teena
Salary:6000.0

_To check a specific record__
Enter the Emp_No:
2
No:2
Name:seema
Salary:30000.0
```

### **PROGRAM: 3**

**AIM:** Create a class 'Person' with data members Name, Gender, Address, Age and a constructor to initialize the data members and another class 'Employee' that inherits the properties of class Person and also contains its own data members like Empid, CompanyName, Qualification, Salary and its own constructor. Create another class 'Teacher' that inherits the properties of class Employee and contains its own data members like Subject, Department, Teacherid and contain constructors and methods to display the data members. Use array of objects to display details of N teachers

### **ALGORITHM**

STEP 1: Create a class person with data members name, gender, address, age

STEP 2: Create constructor to initialize these object

STEP 3: Create another class Employee that inherited the properties of class person and also creates its data members like empid, companyname , qualification, salary and its constructor

STEP 4: Create another class Teacher that inherits the properties of class employee and creates its data members

STEP 5: Creates array of object

STEP 6: Using these arrays of objects print the values

STEP 7: Stop

### **SOURCE CODE**

person.java	<pre>import java.util.Scanner; public class person {     String Name;     String Address;     String Gender;     int Age;      public person(String nam,String gen, String adr, int age) {         Name=nam;         Gender=gen;         Address=adr;         Age=age;     }     static class Employee extends person     {         int Empid;         String company_name;         String Qualification;         double Salary;          public Employee(String nam,String gen,         String adr, int age,int id,String cname,String qu, double sal)</pre>
-------------	---



```

        {
            super(nam,gen,adr,age);
            Empid=id;
            company_name=cname;
            Qualification=qu;
            Salary=sal;
        }
        static class Teacher extends Employee
        {
            int teacher_id;
            String department;
            String subject;
public Teacher(String nam,String gen, String adr, int age,int
id,String cname,String qu,
            double sal,int tid,String dept,String sub) {
                super(nam,gen,adr,age,id,cname,qu,sal);
                teacher_id=tid;
                department=dept;
                subject=sub;
            }
            void display()
            {
                System.out.println("....Personal details...");
                System.out.println("Person name:"+Name);

System.out.println("Gender:"+Gender);

System.out.println("Address:"+Address);
                System.out.println("Age:"+Age);
                System.out.println("...Employee details....");
                System.out.println("Employee id:"+Empid);
                System.out.println("Company Name:"+company_name);
                System.out.println("Salary of employee"+Salary);

System.out.println("Qualification"+Qualification);
                System.out.println("...Teacher's details...");
                System.out.println("Teacher id:"+teacher_id);
                System.out.println("Department:"+department);
                System.out.println("Subject:"+subject);
            }
        }
        public static void main(String[] args) {

            int i,count;
            Scanner s=new Scanner(System.in);
            System.out.println("Enter number of person:");
            count=s.nextInt();

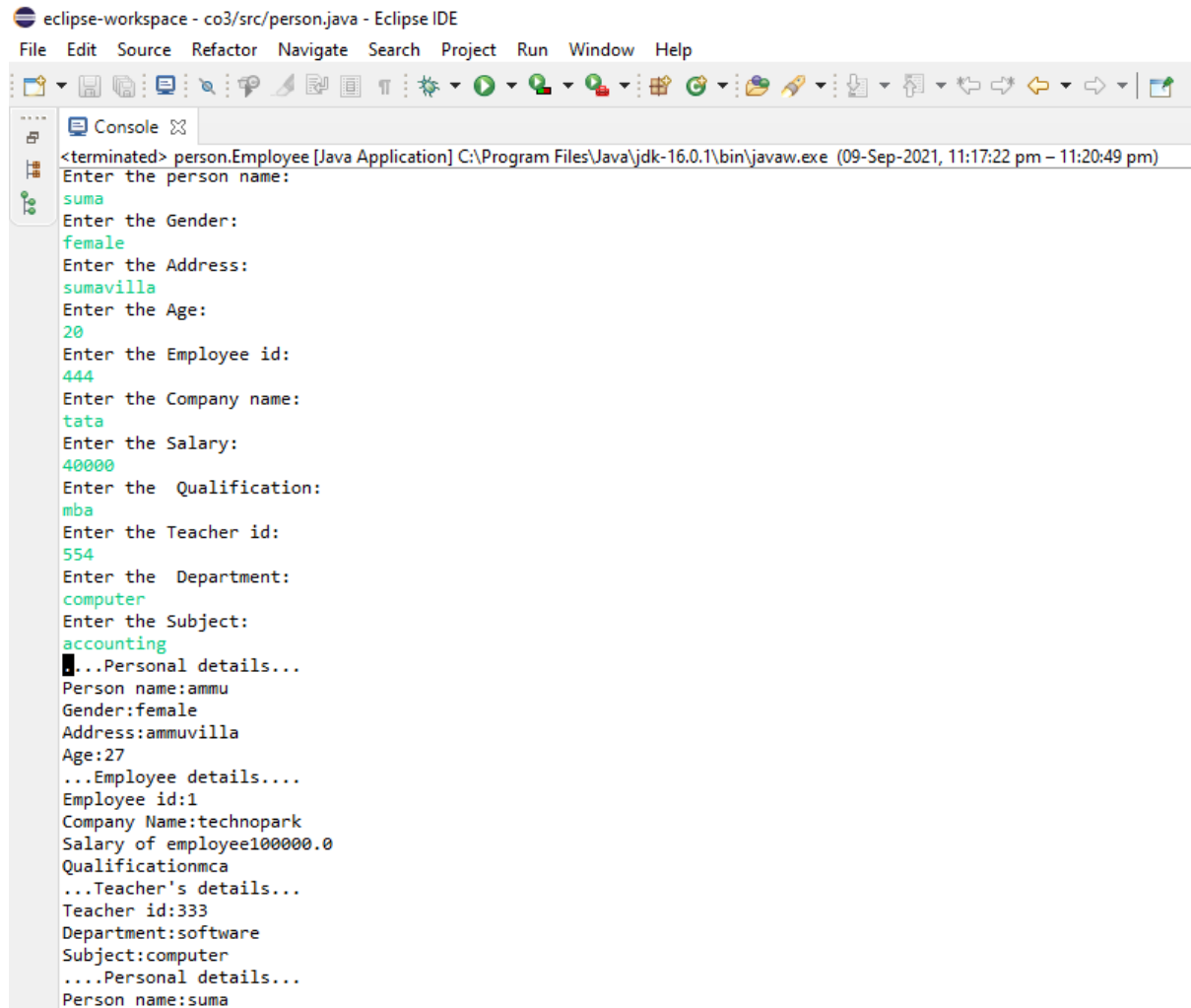
```

	<pre> Teacher t[]=new Teacher[count];  for(i=0;i&lt;count;i++) {     System.out.println("Enter the person name:");      String nam1=s.next();     System.out.println("Enter the Gender: ");     String gen1=s.next();     System.out.println("Enter the Address: ");     String adr1=s.next();     System.out.println("Enter the Age:");     int age1=s.nextInt();     System.out.println("Enter the Employee id: ");      int id1=s.nextInt();     System.out.println("Enter the Company name: ");      String cname1=s.next();     System.out.println("Enter the Salary:");     double sal1=s.nextDouble();     System.out.println("Enter the Qualification:");      String qu1=s.next();      System.out.println("Enter the Teacher id: ");     int tid1=s.nextInt();     System.out.println("Enter the Department:");      String dept1=s.next();     System.out.println("Enter the Subject:");     String sub1=s.next();      t[i]=new Teacher(nam1,gen1,adr1,age1,id1,cname1,qu1,sal1,tid1,dept1,sub1 );}      for(i=0;i&lt;count;i++)     {         t[i].display();      } } } </pre>
--	--

## RESULT

The above program is executed and obtained the output.

## OUTPUT



The screenshot shows the Eclipse IDE interface with the console window open. The title bar reads "eclipse-workspace - co3/src/person.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations, search, and execution. The console window displays the following output:

```
<terminated> person.Employee [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (09-Sep-2021, 11:17:22 pm – 11:20:49 pm)
Enter the person name:
suma
Enter the Gender:
female
Enter the Address:
sumavilla
Enter the Age:
20
Enter the Employee id:
444
Enter the Company name:
tata
Enter the Salary:
40000
Enter the Qualification:
mba
Enter the Teacher id:
554
Enter the Department:
computer
Enter the Subject:
accounting
...Personal details...
Person name:ammu
Gender:female
Address:ammuvilla
Age:27
...Employee details....
Employee id:1
Company Name:technopark
Salary of employee100000.0
Qualificationmca
...Teacher's details...
Teacher id:333
Department:software
Subject:computer
....Personal details...
Person name:suma
```

### **PROGRAM: 4**

**AIM:** Write a program has class Publisher, Book, Literature and Fiction. Read the information and print the details of books from the category, using inheritance.

### **ALGORITHM**

STEP 1: Create a class publisher, Book, Literature and Fiction using inheritance

STEP 2: Create each class's data members and member functions

STEP 3: Read this information from the user

STEP 4: Print the details of book from the category

STEP 5: Stop

### **SOURCE CODE**

Publisher.java	<pre>import java.util.Scanner; public class Publisher {     String Pubname;      Publisher()     {         Scanner s=new Scanner(System.in);         System.out.println("Enter publisher name");         Pubname=s.next();     } } class Book extends Publisher {     String title, author;     int price;      Book()     {         Scanner s=new Scanner(System.in);         System.out.println("Enter Title of the book");         title=s.next();         System.out.println("Enter Author's name");         author=s.next();         System.out.println("Enter price");         price=s.nextInt();     } } class Literature extends Book {     Literature()     {         System.out.println("Literature Books");     } }</pre>
----------------	--

	<pre> void display() {     System.out.println("Publisher name: "+Pubname);     System.out.println("Title of the book: "+title);     System.out.println("Author's name: "+author);     System.out.println("Price: "+price); } } class Fiction extends Literature {     Fiction()     {         System.out.println("Friction Books");     }     void display()     {         super.display();     }      public static void main(String args[])     {         int n;         Scanner s=new Scanner(System.in);          System.out.println("Enter the No of literature book: ");         int a=s.nextInt();         Literature L[]=new Literature[a];         for(int i=0;i&lt;a;i++)         {             L[i]=new Literature();         }          System.out.println("Enter the No of Fiction book: ");         int b=s.nextInt();         Fiction F[]=new Fiction[b];         for(int i=0;i&lt;b;i++)         {             F[i]=new Fiction();         }         int no;         System.out.println("Enter your choice of book");         no=s.nextInt();         int type =no;         switch (no)         {             case 1: </pre>
--	--

	<pre> books");         System.out.println(".....Details of literature         books");         for(int i=0;i&lt;a;i++)         L[i].display();         break;         case 2:         System.out.println(".....Details of fiction         books");         for(int i=0;i&lt;b;i++)         F[i].display();         break;         default:         System.out.println("Wrong input");     } } } </pre>
--	--

## RESULT

The above program is executed and obtained the output.

## OUTPUT

```

eclipse-workspace - myproject/src/myproject/Publisher.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer Console Coverage Debug
myproject
  JRE System Library [J
  src
    myproject
      complex.java
      employee.java
      Employee1.java
      main.java
      matrix.java
      Matrix1.java
      Person.java
      product.java
      Publisher.java
      search.java
      Shapes.java
      string.java
      stringss.java
      module-info.java
  <terminated> Publisher.Book.Literature.Fiction [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (May 25, 2021, 2:28:11 PM - 2:30:45 PM)
  Enter the number of Literature Books :
  2
  Enter the name of the book :
  Ulysses
  Enter the author's name :
  Joyce
  Enter the publisher's name :
  McMac
  Enter the price :
  450
  Enter the name of the book :
  Hamlet
  Enter the author's name :
  Shakespeare
  Enter the publisher's name :
  abs
  Enter the price :
  350
  Enter the number of Fiction books :
  2
  Enter the name of the book :
  Alchemist
  Enter the author's name :
  Paulo Coelho
  Enter the publisher's name :
  asdf
  Enter the price of the book :
  250
  Enter the name of the book :
  Secret
  Enter the author's name :
  Rhonda Byrne
  Enter the publisher's name :
  <chr
  <chr

```

## **PROGRAM: 5**

**AIM:** Create classes Student and Sports. Create another class Result inherited from Student and Sports. Display the academic and sports score of a student.

### **ALGORITHM**

STEP 1: Create classes student and sports

STEP 2: Create another class Result inherited from Student and Sports

STEP 3: Create a function called display to print

STEP 4: display academic and sports score of student

STEP 5: Stop

### **SOURCE CODE**

result.java	<pre>import java.util.Scanner; class sports{     String sport;     int Rating;     sports(String spo, int ra){         sport = spo;         Rating = ra;     } } class student extends sports{     String Grade;     double Overall_per;     student(String spo, int ra,String gd, double per ){         super(spo, ra);         Grade = gd;         Overall_per = per;     } } public class result extends student {     result(String spo, int ra,String gd, double per ){         super(spo, ra, gd, per);     }     void display(){         System.out.println("\nSports Details of Student");         System.out.println("Sport :"+sport);         System.out.println("Rating :"+Rating);         System.out.println("\nAcademic Details of Student");         System.out.println("Academic Grade :"+Grade);         System.out.println("Overall percentage :"+Overall_per);     } }  public static void main(String[] args) {</pre>
-------------	---

	<pre> Scanner sc =new Scanner(System.in); System.out.println("\nEnter the Sports Details of Student"); System.out.println("\n Sport: "); String a =sc.next(); System.out.println("\n Sport Rating  out of 10: "); int b =sc.nextInt(); System.out.println("\nEnter the Sports Details of Student"); System.out.println("\n Academic Grade: "); String c =sc.next(); System.out.println("\n Overall percentage: "); double d =sc.nextDouble(); sc.close(); result obj= new result(a,b,c,d); obj.display(); } } </pre>
--	--

## **RESULT**

The above program is executed and obtained the output.

## **OUTPUT**

```

eclipse-workspace - co3/src/result.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Enter the Sports Details of Student
Sport:
football
Sport Rating  out of 10:
8
Enter the Sports Details of Student
Academic Grade:
40
Overall percentage:
50
Sports Details of Student
Sport :football
Rating :8
Academic Details of Student
Academic Grade :40
Overall percentage :50.0

```



## **PROGRAM: 6**

**AIM:** Create an interface having prototypes of functions area() and perimeter(). Create two classes Circle and Rectangle which implement the above interface. Create a menu driven program to find the area and perimeter of objects.

### **ALGORITHM**

STEP 1: Create an interface having prototypes of function area() and perimeter()

STEP 2: Create classes that implements the interface

STEP 3: Create two classes circle and rectangle

STEP 4: find the perimeter and area of rectangle and circle

STEP 5: Print the results

STEP 6: Stop

### **SOURCE CODE**

calculate.java	<pre>import java.util.Scanner; interface math{     void area();     void perimeter();     void input(); } class circle implements math{     float r;     double pi=3.14,area,perimeter;     Scanner cr=new Scanner(System.in);      public void input() {         System.out.println("enter the radius:");         r=cr.nextFloat();     }      @Override     public void area() {         area=pi*r*r;         System.out.println("Area of the circle:"+area);     }      @Override     public void perimeter() {         perimeter=2*pi*r;         System.out.println("Perimeter of the circle:"+perimeter);     } }</pre>
----------------	--

```

class rectangle implements math{
    int l,w,area,perimeter;
    Scanner rl=new Scanner(System.in);

    @Override
    public void input() {
        System.out.println("enter the length:");
        l=rl.nextInt();
        System.out.println("enter the width:");
        w=rl.nextInt();

    }

    @Override
    public void area() {
        area=l*w;
        System.out.println("Area of the rectangle:"+area);

    }

    @Override
    public void perimeter() {
        perimeter=2*(l+w);
        System.out.println("Perimeter of the
rectangle:"+perimeter);

    }

}

public class calculate{

    public static void main(String[] args) {
        circle obj1=new circle();
        rectangle obj2=new rectangle();
        int y=0;
        while(y<3) {
            Scanner sc=new Scanner(System.in);
            System.out.println("\nChoose any one:");
            System.out.println("1.Circle:\n2.Rectangle:");
            y=sc.nextInt();
            switch(y) {
                case 1:
                    obj1.input();
                    obj1.area();
                    obj1.perimeter();
                    break;
                case 2:

```

	<pre> obj2.input(); obj2.area(); obj2.perimeter(); break; default:     System.out.println("wrong choice!!!!");     }     }     }     }     } </pre>
--	---

## RESULT

The above program is executed and obtained the output.

## OUTPUT

```

eclipse-workspace - co3/src/Publisher.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
calculate [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (10-Sep-2021, 12:14:30 pm)

Choose any one:
1.Circle:
2.Rectangle:
1
enter the radius:
4
Area of the circle:50.24
Perimeter of the circle:25.12

Choose any one:
1.Circle:
2.Rectangle:
2
enter the length:
7
enter the width:
9
Area of the rectangle:63
Perimeter of the rectangle:32

Choose any one:
1.Circle:
2.Rectangle:

```

## **PROGRAM: 7**

**AIM:** Prepare bill with the given format using calculate method from interface.

Order No.

Date:

Product Id	Name	Quantity	unit price	Total
------------	------	----------	------------	-------

101	A	2	25	50
-----	---	---	----	----

102	B	1	100	100
-----	---	---	-----	-----

---

Net. Amount 150

### **ALGORITHM**

STEP 1: Create a class Co37.java

STEP 2: Create another class called cal that inherits from the interface cal\_bill

STEP 3: Create a display function

STEP 4: Print the values as shown as the question

STEP 5: Stop

### **SOURCE CODE**

co37.java

```
import java.util.*;
import java.time.*;

interface cal_bill
{
    public void calc();
}

class cal implements cal_bill
{
    int p_id,qty,up,tot,o_n,date;
    String name;
    public void details()
    {
        Scanner ss = new Scanner(System.in);

        System.out.println("Enter Product ID : ");
        p_id = ss.nextInt();
        System.out.println("Enter Name : ");
        name = ss.next();
    }
}
```

```

        System.out.println("Enter Quantity : ");
        qty = ss.nextInt();
        System.out.println("Unit Price : ");
        up = ss.nextInt();

    }

    public void order()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter Order No : ");
        o_n = sc.nextInt();
    }

    public void calc()
    {
        tot = qty * up;
        System.out.println("  "+p_id+"\t\t "+name+"\t\t "+qty+"\t\t "+up+"\t\t "+tot);

    }

    void display()
    {
        System.out.print("\n");
        System.out.println("BILL");
        System.out.println("=====");
        System.out.print("\n");
        System.out.println("Order No : " + o_n);
        LocalDate obj = LocalDate.now();
        System.out.print("\n");
        System.out.println("Date : " + obj);
        System.out.print("\n");
        System.out.println("Product Id\tName\tQuantity\tunit
price\tTotal");

        System.out.print("_____");
        System.out.print("\n");

    }

}

public class co37
{
    public static void main(String[] args)

```

```

{
    int no,net = 0;

    Scanner s = new Scanner(System.in);
    System.out.println("Enter no of products : ");
    no = s.nextInt();
    cal or = new cal();

    cal obj[] = new cal[no];

    or.order();

    for(int i=0;i<no;i++)
    {
        System.out.print("\n");
        System.out.println("Enter Details of product "+ (i+1));

System.out.println("=====");

        obj[i] = new cal();
        obj[i].details();

    }

    or.display();

    for(int i=0;i<no;i++)
    {
        obj[i].calc();
        net = net + obj[i].tot;

    }

System.out.println("_____");
    System.out.println("    "+"\\t\\t "+"\\t    "+"\\t\\tNet amount :"+"\\t
    "+"net);

}
}

```

## RESULT

The above program is executed and obtained the output.

## OUTPUT

```
eclipse-workspace - co3/src/co37.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

<terminated> co37 [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (10-Sep-2021, 12:27:24 pm – 12:28:46 pm)
55
Enter Details of product 1
=====
Enter Product ID :
565
Enter Name :
saree
Enter Quantity :
7000
Unit Price :
800

Enter Details of product 2
=====
Enter Product ID :
99
Enter Name :
kurti
Enter Quantity :
90
Unit Price :
8000

BILL
=====

Order No : 55

Date : 2021-09-10

Product Id      Name      Quantity      unit price      Total
-----
565            saree      7000          800             5600000
99             kurti      90            8000            720000
-----
Net amount :    6320000
```

## **LABCYCLE 4**



## **PROGRAM 1**

**AIM:** Create a Graphics package that has classes and interfaces for figures Rectangle, Triangle, Square and Circle. Test the package by finding the area of these figures.

### **ALGORITHM**

STEP 1 : Start

STEP 2: Define a package contain functions to assign calculate and display .

STEP 3: Read inputs from user and assign values to objects.

STEP 4: Perform desired operations.

STEP 5: Print the Outputs.

STEP 6: Stop

### **SOURCE CODE**

area.java	<pre>package Graphics; import java.util.Scanner; interface figure{     void rectangle();     void triangle();     void square();     void circle(); }  public class area implements figure {     int x,y,b,h,r;     double a,ar,are,circle_ar;     Scanner sc=new Scanner(System.in);     public void rectangle()     {         System.out.println("Enter the length of Rectangle: ");         x=sc.nextInt();         System.out.println("Enter the breadth of Rectangle: ");         y=sc.nextInt();         a=x*y;         System.out.println("Area of Rectangle: "+a);     }     public void triangle()     {         System.out.println("Enter the base length of Triangle: ");</pre>
-----------	--

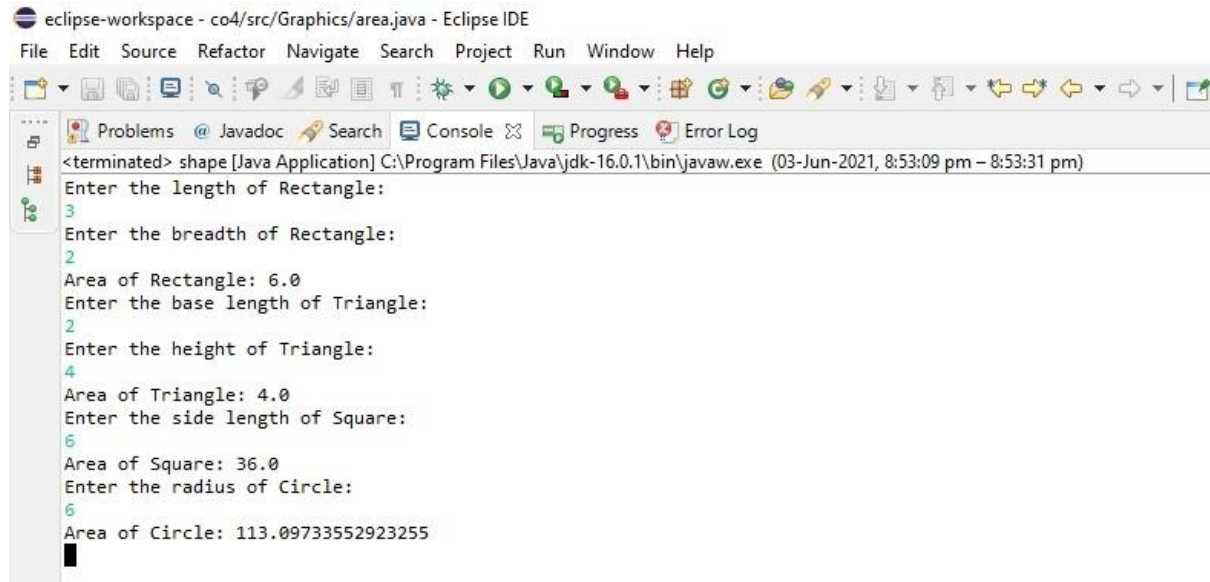
	<pre>         b=sc.nextInt();         System.out.println("Enter the height of Triangle: ");         h=sc.nextInt();         ar=0.5*b*h;         System.out.println("Area of Triangle: "+ar);     }     public void square()     {         System.out.println("Enter the side length of Square: ");         a=sc.nextInt();         are=a*a;         System.out.println("Area of Square: "+are);     }     public void circle()     {         System.out.println("Enter the radius of Circle: ");         r=sc.nextInt();         circle_ar=Math.PI*r*r;         System.out.println("Area of Circle: "+circle_ar);     } } </pre>
--	--

shape.java	<pre> package Graphics; import Graphics.area; public class shape {      public static void main(String[] args)     {         area ar=new area();         ar.rectangle();         ar.triangle();         ar.square();         ar.circle();     }  } </pre>
------------	---

## RESULT

The above program is executed and obtained the output.

## OUTPUT



The screenshot shows the Eclipse IDE interface. The title bar reads 'eclipse-workspace - co4/src/Graphics/area.java - Eclipse IDE'. The menu bar includes 'File', 'Edit', 'Source', 'Refactor', 'Navigate', 'Search', 'Project', 'Run', 'Window', and 'Help'. The toolbar contains various icons for file operations, search, and development. The 'Console' tab is active, displaying the following output:

```
<terminated> shape [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (03-Jun-2021, 8:53:09 pm – 8:53:31 pm)
Enter the length of Rectangle:
3
Enter the breadth of Rectangle:
2
Area of Rectangle: 6.0
Enter the base length of Triangle:
2
Enter the height of Triangle:
4
Area of Triangle: 4.0
Enter the side length of Square:
6
Area of Square: 36.0
Enter the radius of Circle:
6
Area of Circle: 113.09733552923255
█
```

## **PROGRAM 2**

**AIM:** Create an Arithmetic package that has classes and interfaces for the 4 basic arithmetic operations. Test the package by implementing all operations on two given numbers

### **ALGORITHM**

STEP 1 : Start

STEP 2: Define a package contain functions to assign calculate and display .

STEP 3: Read inputs from user and assign values to objects.

STEP 4: Perform desired operations.

STEP 5: Print the Outputs.

STEP 6: Stop

### **SOURCE CODE**

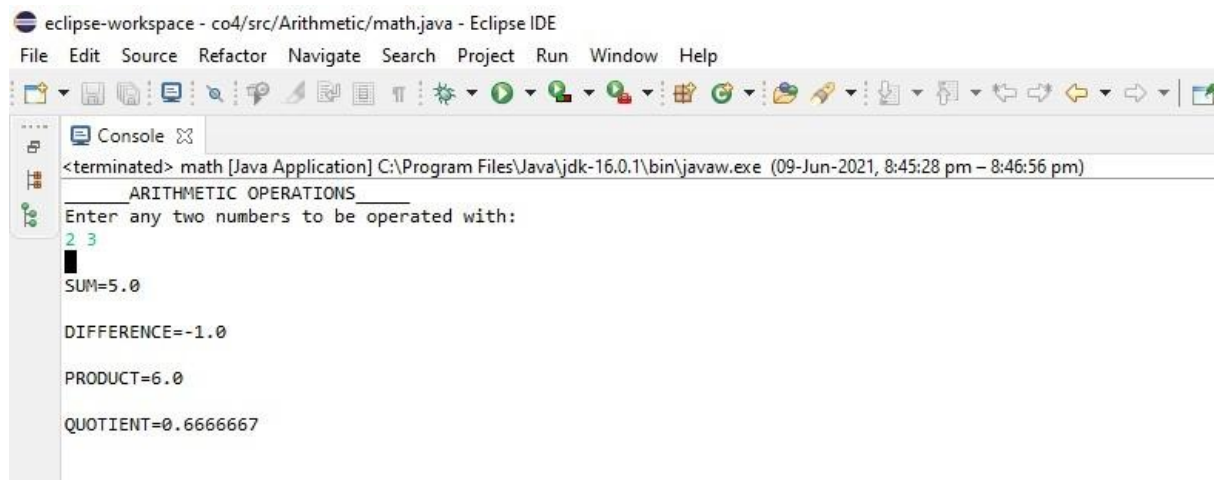
math.java	<pre>package Arithmetic; import Arithmetic.operands; import java.util.Scanner;  public class math {      public static void main(String[] args) {         Scanner s = new Scanner(System.in);          float a,b;          System.out.println("_____ARITHMETIC OPERATIONS_____");          System.out.println("Enter any two numbers to be operated with:");         a = s.nextFloat();         b = s.nextFloat();          operands op = new operands(a,b);         op.add();         op.sub();         op.mul();         op.div();          s.close();     } }</pre>
-----------	--

operands.java	<pre>package Arithmetic;  interface operations{     void add();     void sub();     void div();     void mul(); }  public class operands implements operations {     float x,y;      public operands(float a, float b) {          x=a;         y=b;      }      public void add() {         System.out.println("\nSUM="+(x+y));     }      public void sub() {         System.out.println("\nDIFFERENCE="+(x-y));     }      public void div() {         System.out.println("\nQUOTIENT="+(x/y));     }      public void mul() {         System.out.println("\nPRODUCT="+(x*y));     }  }</pre>
---------------	---

## RESULT

The above program is executed and obtained the output.

## OUTPUT



The screenshot shows the Eclipse IDE interface. The title bar reads "eclipse-workspace - co4/src/Arithmetic/math.java - Eclipse IDE". The menu bar includes "File", "Edit", "Source", "Refactor", "Navigate", "Search", "Project", "Run", "Window", and "Help". The toolbar contains various icons for file operations, editing, and running. The "Console" view is active, displaying the following output:

```
<terminated> math [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (09-Jun-2021, 8:45:28 pm – 8:46:56 pm)
_____ARITHMETIC OPERATIONS_____
Enter any two numbers to be operated with:
2 3
SUM=5.0
DIFFERENCE=-1.0
PRODUCT=6.0
QUOTIENT=0.6666667
```

### **PROGRAM 3**

**AIM:** Write a user defined exception class to authenticate the user name and password.

#### **ALGORITHM**

STEP 1 : Start

STEP 2: Read inputs as username and password.

STEP 3: Verify the username and password.

STEP 4: If its true; Print Succesful message:.

STEP 5: Else print error meessage.

STEP 6: Stop

#### **SOURCE CODE**

login.java	<pre>import java.util.Scanner; class Username_Exception extends Exception {     public Username_Exception(String string) {         super(string);     } }  class Password_Exception extends Exception {     public Password_Exception(String string) {         super(string);     } }  public class login {     public static void main(String[] args) {         Scanner sc = new Scanner(System.in);         String username, password;          System.out.print("Enter username:");         username = sc.nextLine();          System.out.print("Enter password:");         password = sc.nextLine();          int length = username.length();          try {             if(length&gt;8)                 throw new Username_Exception("username must be atmost 8 characters!");             if(!password.equals("1212"))                 throw new Password_Exception("Incorrect password\nEnter the correct password??");</pre>
------------	--

	<pre> else     System.out.println("***Successfully login***"); } catch (Username_Exception U) {     U.printStackTrace(); } catch (Password_Exception P) {     P.printStackTrace(); } finally {     System.out.println("The finally statement is executed "); } } } </pre>
--	---

## RESULT

The above program is executed and obtained the output.

## OUTPUT

eclipse-workspace - co4/src/login.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Console

```

<terminated> login [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (09-Jun-2021, 8:55:19 pm – 8:55:42 pm)
Enter username:blessy
Enter password:1212
***Successfully login***
The finally statement is executed

```

eclipse-workspace - co4/src/login.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Console

```

<terminated> login [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (09-Jun-2021, 8:56:36 pm – 8:56:51 pm)
Enter username:blessyproy
Enter password:1212
Username Exception: username must be atmost 8 characters!
The finally statement is executed
    at login.main(login.java:30)

```

eclipse-workspace - co4/src/login.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Console

```

<terminated> login [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (09-Jun-2021, 8:52:09 pm – 8:52:27 pm)
Enter username:blessy
Enter password:blessy11
Password Exception: Incorrect password
Enter the correct password??
The finally statement is executed
    at login.main(login.java:32)

```



### **PROGRAM: 4**

**AIM:** Find the average of N positive integers, raising a user defined exception for each negative input.

### **ALGORITHM**

STEP 1 : Start

STEP 2: Enter a limit n.

STEP 3:Read n elements.

STEP 4:iterate the loop .

STEP 5:if i>0;. then

STEP 6: sum=sum+i

STEP 7: Else Print error message

STEP 8: Av=sum/n

STEP 9: Print av

STEP 10: Stop

### **SOURCE CODE**

positive.java	<pre>import java.util.*;  class Neg_Exception extends Exception {     public Neg_Exception(String s)     {         super(s);     } }  public class positive {     public static void main(String[] args)     {         int i;         double sum=0;         Scanner sc=new Scanner(System.in);         System.out.println("Count of numbers: ");         int N=sc.nextInt();         int[] intArray=new int[N];         System.out.println("Enter the numbers: ");         for(i=0;i&lt;N;i++)         {             intArray[i]=sc.nextInt();         }     } }</pre>
---------------	--

	<pre>         }         for(i=0;i&lt;N;i++)         {             try             {                 if (intArray[i]&lt;0)                 { throw new Neg_Exception(+intArray[i]+"tis a negative number");                 }                 else                 {                     sum=sum+intArray[i];                 }             }             catch( Neg_Exception e)             {                 e.printStackTrace();                 System.out.println("Enter a positive number: ");                 intArray[i]=sc.nextInt();                 sum=sum+intArray[i];             }         }         System.out.println("Average of N positive numbers:"+sum/N);     } } </pre>
--	--

## RESULT

The above program is executed and obtained the output.

## OUTPUT

```

eclipse-workspace - co4/src/positive.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
<terminated> positive [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (10-Jun-2021, 9:58:41 pm - 9:59:18 pm)
Count of numbers:
2
Enter the numbers:
-3
4
Neg_Exception: -3 is a negative number
Enter a positive number:
at positive.main(positive.java:34)
5
Average of N positive numbers:4.5

```

## **PROGRAM 5**

**AIM:** Define 2 classes; one for generating a multiplication table of 5 and another for displaying first N prime numbers. Implement using threads. (Thread class)

### **ALGORITHM**

STEP 1 : Start

STEP 2: Define a class for multiplication and another for prime numbers.

STEP 3: Read a input as num.

STEP 4: Print the multiplication table of num.

STEP 5: Print num prime numbers.

STEP 6: Stop

### **SOURCE CODE**

threadmul1.java	<pre>import java.util.Scanner;  class MulTable extends Thread{     public void run() {         int num = 5;         System.out.printf("_____Multiplication Table of 5 _____\n");         for(int i = 1; i &lt;= 10; ++i)         {             System.out.printf("%d * %d = %d \n", num, i, num * i);         }     } }  class PrimeNo extends Thread{     public void run() {         int i, j, flag;         Scanner s = new Scanner(System.in);         System.out.println("\n_____To generate first N prime numbers_____");         System.out.println("Enter the limit (N):");         int N = s.nextInt();          System.out.println("Prime numbers between 1 and " + N + " are:");          for (i = 1; i &lt;= N; i++)         {</pre>
-----------------	--

```

        if (i == 1 || i == 0)
            continue;

        flag = 1;

        for (j = 2; j <= i / 2; ++j)
        {
            if (i % j == 0)
            {
                flag = 0;
                break;
            }
        }

        if (flag == 1)
            System.out.print(i + " ");
    }
}

```

```

public class threadmul {

    public static void main(String[] args) throws
InterruptedException {
        MulTable m = new MulTable();
        m.start();
        m.sleep(100);

        PrimeNo p = new PrimeNo();
        p.start();
        p.sleep(100);

    }

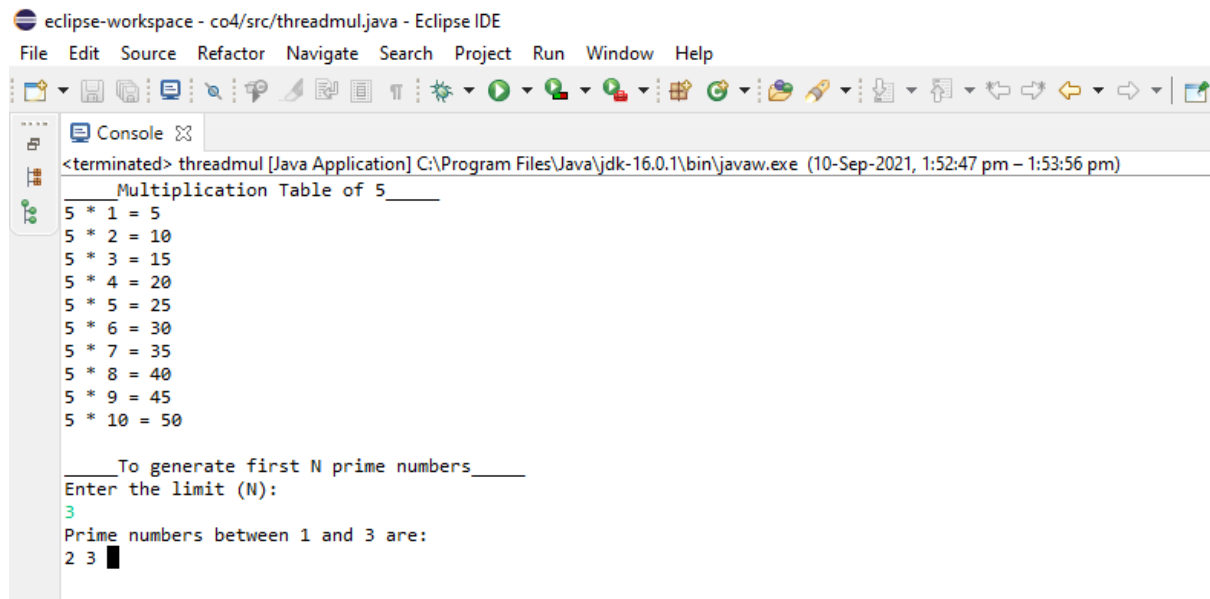
}

```

## RESULT

The above program is executed and obtained the output.

## OUTPUT



```
eclipse-workspace - co4/src/threadmul.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

<terminated> threadmul [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (10-Sep-2021, 1:52:47 pm - 1:53:56 pm)

Multiplication Table of 5_____
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50

_____To generate first N prime numbers_____
Enter the limit (N):
3
Prime numbers between 1 and 3 are:
2 3
```

## PROGRAM 6

**AIM:** Define 2 classes; one for generating Fibonacci numbers and other for displaying even numbers in a given range. Implement using threads. (Runnable Interface)

### ALGORITHM

STEP 1 : Start

STEP 2: Define a class for Fibonacci and another for even numbers.

STEP 3: Read a input as n.

STEP 4: Print the Fibonacci series of n number.

STEP 5: Print n even numbers.

STEP 6: Stop

### SOURCE CODE

fib.java	<pre>import java.util.Scanner;  class Fibonacci implements Runnable{     public void run(){         int a=0,b=1,c=0,l=20;         System.out.println("FIBONACCI SERIES UPTO "+l+": \n");         while (l&gt;0)         {             System.out.print(c+" ");             a=b;             b=c;             c=a+b;             l=l-1;             if(l%10==0)             {                 System.out.println("\n");             }         }     } }  class EvenNumber implements Runnable{     public void run(){         int n;         Scanner sc=new Scanner(System.in);         System.out.println("Enter the limit : ");         n=sc.nextInt();         System.out.println("Even Numbers from 1 to "+n+"\n");         for(int i=1;i&lt;=n;i++) {</pre>
----------	---

	<pre>                                 if(i%2==0) {                                     System.out.println(i);                                 }                             }                         }                     }                 }             }         }     }      public class fib {          public static void main(String[] args) {             Fibonacci obj1=new Fibonacci();             Thread t1=new Thread(obj1);             t1.start();              EvenNumber obj2=new EvenNumber();             Thread t2=new Thread(obj2);             t2.start();         }     } </pre>
--	---

## RESULT

The above program is executed and obtained the output.

## OUTPUT

```

eclipse-workspace - co4/src/fib.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
<terminated> fib [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (16-Jun-2021, 9:55:39 pm - 9:55:44 pm)
FIBONACCI SERIES UPTO 20:
0 1 1 2 3 5 8 13 21 34
55 89 144 233 377 610 987 1597 2584 4181
Enter the limit :
20
Even Numbers from 1 to 20
2
4
6
8
10
12
14
16
18
20

```

## **PROGRAM 7**

**AIM:** Producer/Consumer using ITC

### **ALGORITHM**

STEP 1 : Start

STEP 2: Define a class name as producer with members pname, pcode and price.

STEP 3: Define objects to Class and add 3 products and values to each data using the object.

STEP 4: Check whether the product has the lowest price using an if-else statement.

STEP 5: Print the details of the product.

STEP 6: Stop

### **SOURCE CODE**

producercustomer.java	<pre>import java.util.LinkedList; public class producerconsumer { public static void main(String[] args) { // shared list LinkedList&lt;Integer&gt; list = new LinkedList&lt;Integer&gt;(); Thread t1 = new Thread(new Producer(list), "Producer"); Thread t2 = new Thread(new Consumer(list), "Consumer"); t1.start(); t2.start(); } }  //Producer task  class Producer implements Runnable{ LinkedList&lt;Integer&gt; list; Producer(LinkedList&lt;Integer&gt; list){ this.list = list; } @Override public void run() { for(int i = 1; i &lt;=10; i++){ synchronized(list) { // If there is already an element in the list wait while(list.size() &gt;= 1){ System.out.println("Waiting as queue is full.."); try { list.wait(); } catch (InterruptedException e) { e.printStackTrace(); }</pre>
-----------------------	--



```

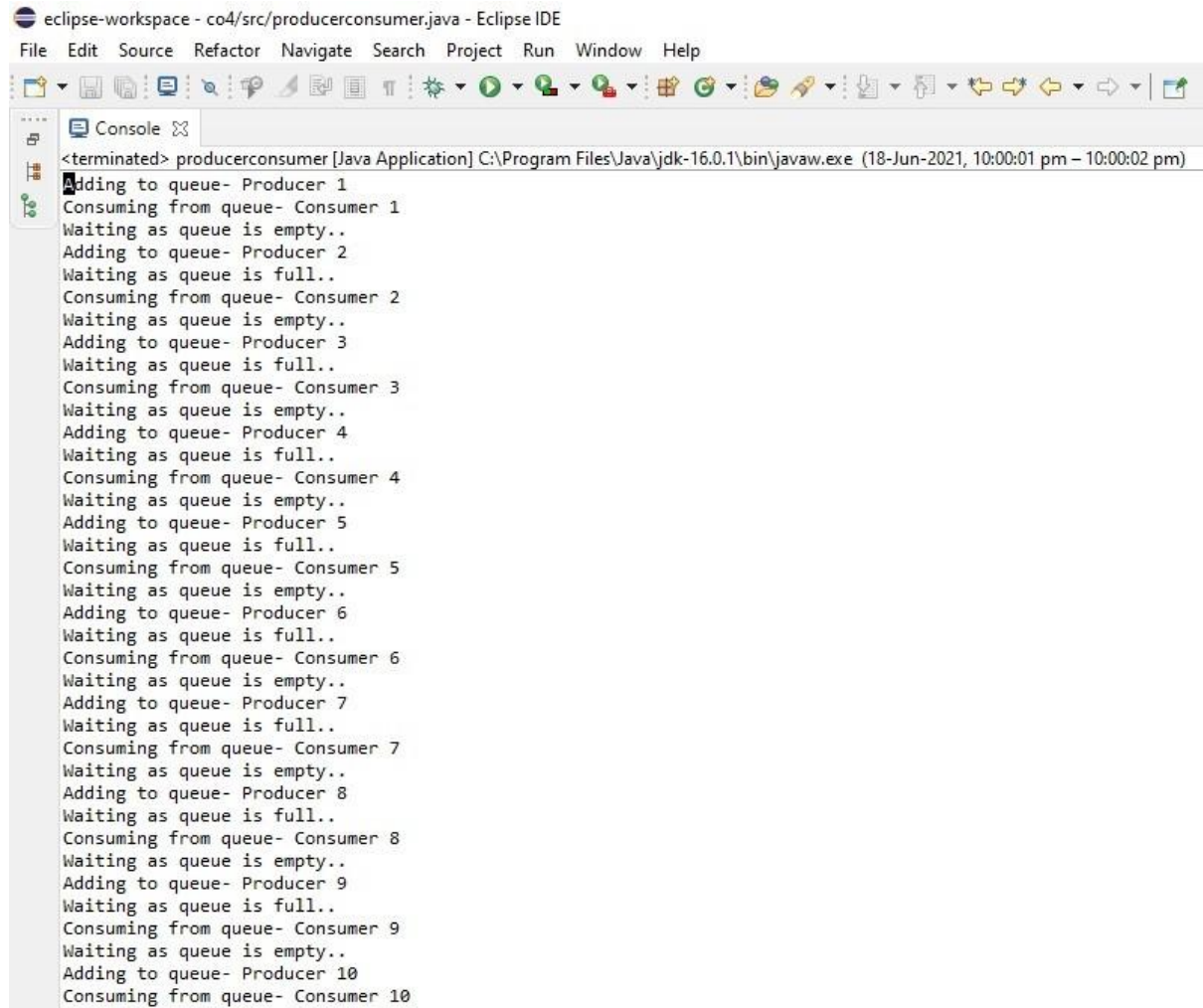
    }
}
System.out.println("Adding to queue- " +
Thread.currentThread().getName() + " " + i);
list.add(i);
list.notify();
}
}
}
}
//Consumer task
class Consumer implements Runnable{
LinkedList<Integer> list;
Consumer(LinkedList<Integer> list){
this.list = list;
}
@Override
public void run() {
for(int i = 1; i <= 10; i++){
synchronized(list) {
// if there is no element in the list wait
while(list.size() < 1){
System.out.println("Waiting as queue is empty..");
try {
list.wait();
} catch (InterruptedException e) {
e.printStackTrace();
}
}
// if there is element in the list then retrieve it
System.out.println("Consuming from queue- " +
Thread.currentThread().getName() + " " + list.remove());
list.notify();
}
}
}
}
}
}

```

## RESULT

The above program is executed and obtained the output.

## OUTPUT



The screenshot shows the Eclipse IDE interface with the console window open. The title bar indicates the workspace is 'eclipse-workspace - co4/src/producerconsumer.java - Eclipse IDE'. The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations, search, and development. The console window displays the output of a Java application named 'producerconsumer'. The output shows a sequence of operations where 10 producers (numbered 1 to 10) add items to a queue and 10 consumers (numbered 1 to 10) consume items from the queue. The queue has a capacity of 3. The output alternates between 'Adding to queue- Producer X' and 'Consuming from queue- Consumer Y', with 'Waiting as queue is empty..' or 'Waiting as queue is full..' messages indicating when a producer or consumer must wait for the queue to become available. The process terminates successfully.

```
<terminated> producerconsumer [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (18-Jun-2021, 10:00:01 pm - 10:00:02 pm)
Adding to queue- Producer 1
Consuming from queue- Consumer 1
Waiting as queue is empty..
Adding to queue- Producer 2
Waiting as queue is full..
Consuming from queue- Consumer 2
Waiting as queue is empty..
Adding to queue- Producer 3
Waiting as queue is full..
Consuming from queue- Consumer 3
Waiting as queue is empty..
Adding to queue- Producer 4
Waiting as queue is full..
Consuming from queue- Consumer 4
Waiting as queue is empty..
Adding to queue- Producer 5
Waiting as queue is full..
Consuming from queue- Consumer 5
Waiting as queue is empty..
Adding to queue- Producer 6
Waiting as queue is full..
Consuming from queue- Consumer 6
Waiting as queue is empty..
Adding to queue- Producer 7
Waiting as queue is full..
Consuming from queue- Consumer 7
Waiting as queue is empty..
Adding to queue- Producer 8
Waiting as queue is full..
Consuming from queue- Consumer 8
Waiting as queue is empty..
Adding to queue- Producer 9
Waiting as queue is full..
Consuming from queue- Consumer 9
Waiting as queue is empty..
Adding to queue- Producer 10
Consuming from queue- Consumer 10
```

## **PROGRAM 8**

**AIM:** Program to create a generic stack and do the Push and Pop operations.

### **ALGORITHM**

STEP 1 : Start

STEP 2: Define a stack .

STEP 3: Push(); enter an element n

If top<n

Top++

Stack[top]=n.

STEP 4:Pop()

If top!=1

.top--

STEP 5: Print the stack.

STEP 6: Stop

### **SOURCE CODE**

generic.java	<pre>import java.util.Scanner; class StackArr { int a[] = new int[20]; int top=-1,ch,item,i; Scanner sc = new Scanner(System.in); public void stackoperation() {     System.out.println("Enter the size of the array : ");     int n=sc.nextInt();  do { System.out.println("\n\t**CHOICES** "); System.out.println("\n 1.PUSH \n 2.POP \n 3.DISPLAY \n 4.EXIT \n"); System.out.println("\n Enter your choice : "); ch=sc.nextInt(); switch(ch) { case 1: if(top &gt;=n-1) { System.out.println("stack overflow");</pre>
--------------	--

```

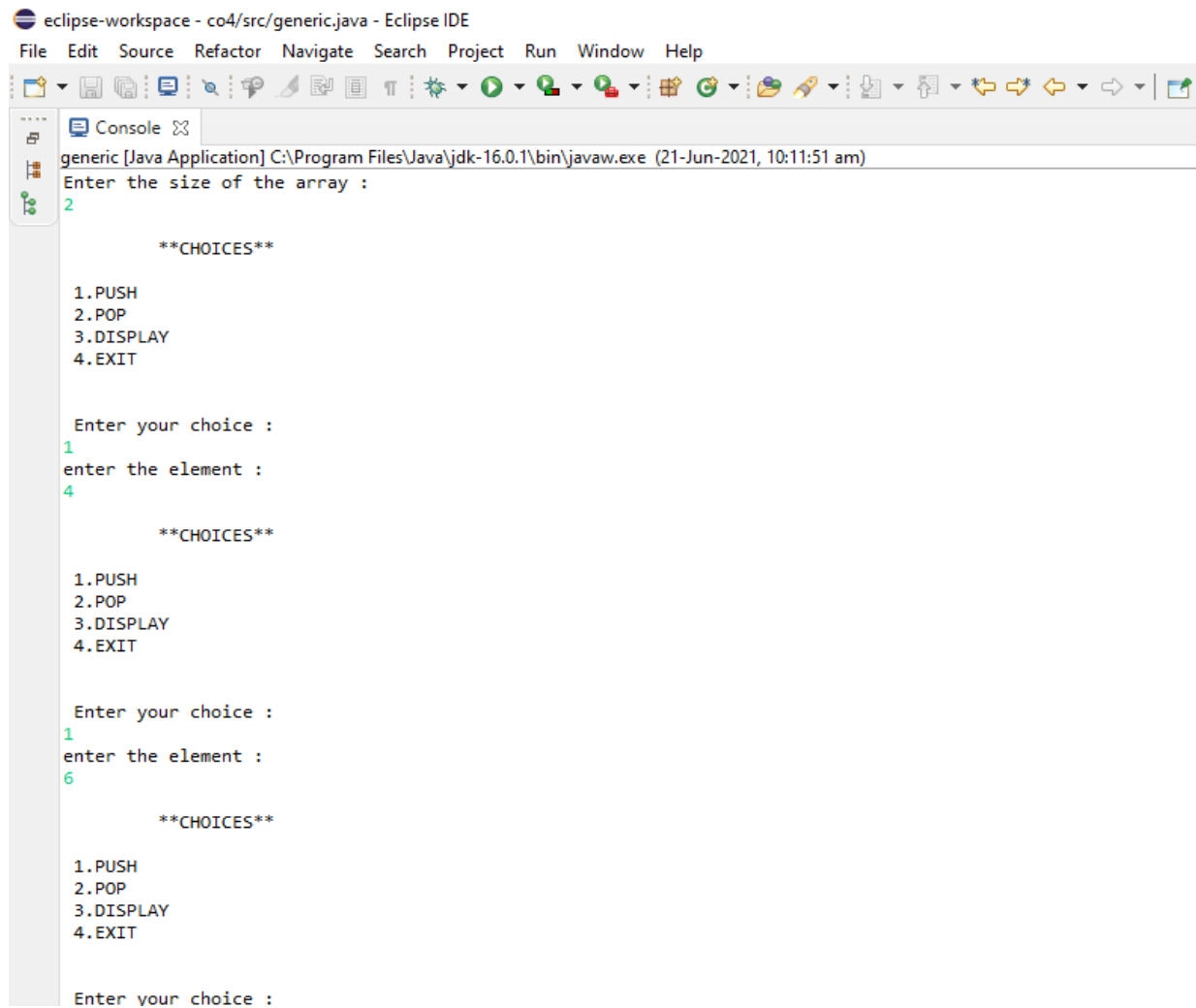
    }
    else
    {
        System.out.println("enter the element :");
        item =sc.nextInt();
        top=top+1;
        a[top]=item;
    }
    break;
case 2 : if(top<0)
    {
        System.out.println("stack underflow");
    }
    else
    {
        a[top]='\0';
        top=top-1;
    }
    break;
case 3 :  if(top < 0)
    {
        System.out.println("\n stack is empty");
    }
    else
    {
        System.out.println("\n stack is \n");
        for(i=top;i>=0;i--)
        {
            System.out.println(a[i]);
        }
    }
    break;
case 4 :
    System.out.println("\nTerminate the stack");
    break;
default : System.out.println("\n Invalid choice");
}
}
while(ch!=4);
}
}
class generic
{
public static void main(String[] args)
{
    StackArr sa =new StackArr();
    sa.stackoperation();
}}

```

## RESULT

The above program is executed and obtained the output.

## OUTPUT



The screenshot shows the Eclipse IDE interface with the console window open. The title bar reads "eclipse-workspace - co4/src/generic.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations, editing, and running. The console window shows the output of a Java application. The text in the console is as follows:

```
generic [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (21-Jun-2021, 10:11:51 am)
Enter the size of the array :
2

    **CHOICES**

1.PUSH
2.POP
3.DISPLAY
4.EXIT

Enter your choice :
1
enter the element :
4

    **CHOICES**

1.PUSH
2.POP
3.DISPLAY
4.EXIT

Enter your choice :
1
enter the element :
6

    **CHOICES**

1.PUSH
2.POP
3.DISPLAY
4.EXIT

Enter your choice :
```

## **PROGRAM 9**

**AIM:** Using generic method perform Bubble sort.

### **ALGORITHM**

STEP 1 : Start

STEP 2: Look at the first number in the list.

STEP 3: Compare the current number with the next number.

STEP 4: Is the next number smaller than the current number? If so, swap the two numbers around. If not, do not swap.

STEP 5: Move to the next number along in the list and make this the current number.

STEP 6: Repeat from step 2 until the last number in the list has been reached.

STEP 7: If any numbers were swapped, repeat again from step 1.

STEP 8: If the end of the list is reached without any swaps being made, then the list is ordered and the algorithm can stop.

STEP 9: Stop

### **SOURCE CODE**

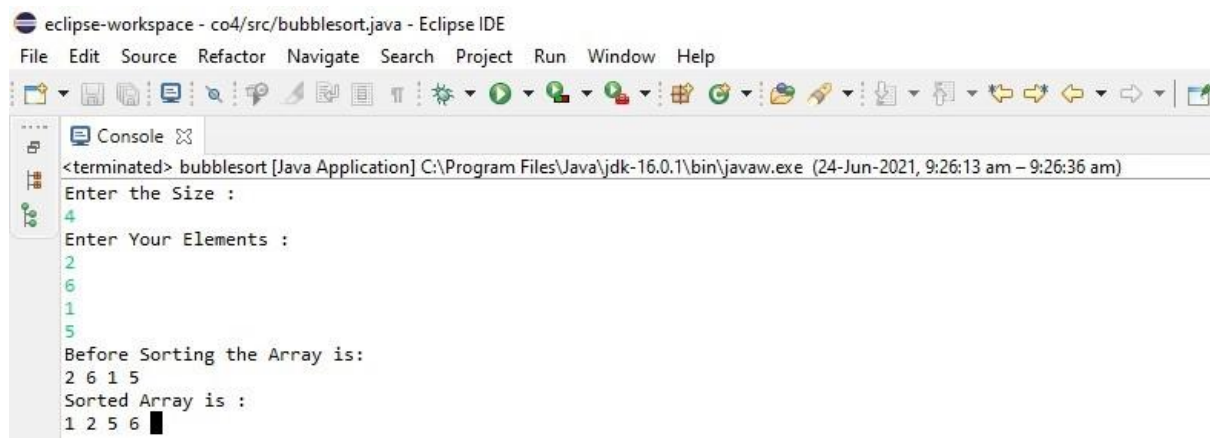
bubblesort.java	<pre>import java.util.Scanner; class bubble {      void sort(int ar[],int n) {         int temp;         int i,j;         for( i=0;i&lt;ar.length-1;i++) {             for(j=0;j&lt;ar.length-1-i;j++) {                  if(ar[j]&gt;ar[j+1]) {                     temp=ar[j];                     ar[j]=ar[j+1];                     ar[j+1]=temp;                  }             }         }          void display(int ar[],int n) {             for (int i=0;i&lt;n;i++) {                 System.out.print(ar[i]+" ");             }         }      public class bubblesort {         public static void main(String[] args) {</pre>
-----------------	--

	<pre> int n,i; Scanner sc=new Scanner(System.in); System.out.println("Enter the Size :"); n=sc.nextInt(); System.out.println("Enter Your Elements :");  int ar[] = new int[n]; for(i=0;i&lt;n;i++) {     ar[i] = sc.nextInt(); } bubble obj= new bubble (); System.out.println("Before Sorting the Array is:"); obj.display(ar,n); obj.sort(ar,n);  System.out.println("\nSorted Array is :"); obj.display(ar,n); } } </pre>
--	--

## **RESULT**

The above program is executed and obtained the output.

## **OUTPUT**



```

eclipse-workspace - co4/src/bubblesort.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
<terminated> bubblesort [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (24-Jun-2021, 9:26:13 am - 9:26:36 am)
Enter the Size :
4
Enter Your Elements :
2
6
1
5
Before Sorting the Array is:
2 6 1 5
Sorted Array is :
1 2 5 6

```

## **PROGRAM 10**

**AIM:** Maintain a list of Strings using ArrayList from collection framework, perform built-in operations.

### **ALGORITHM**

STEP 1 : Start

STEP 2: Define an array .

STEP 3: Define objects to array.

STEP 4: Perform operation on array.

STEP 5: Print the array after operations.

STEP 6: Stop

### **SOURCE CODE**

arraylist.java	<pre>import java.util.*;  public class arraylist{     public static void main(String args[]) {          ArrayList&lt;String&gt; obj = new ArrayList&lt;String&gt;();         obj.add("Ajeet");         obj.add("Harry");         obj.add("Chaitanya");         obj.add("Steve");         obj.add("Anuj");         // Displaying elements         System.out.println("\nOriginal ArrayList:");         for(String str:obj)             System.out.println(str);         /* Add element at the given index*/         obj.add(0, "Rahul");         obj.add(1, "Justin");         // Displaying elements         System.out.println("\nArrayList after add operation:");         for(String str:obj)             System.out.println(str);         //Remove elements from ArrayList like this         obj.remove("Chaitanya");         //Remove element from the specified index         obj.remove(1);         // Displaying elements         System.out.println("\nArrayList after remove operation:");         for(String str:obj)             System.out.println(str);         // Displaying elements</pre>
----------------	--

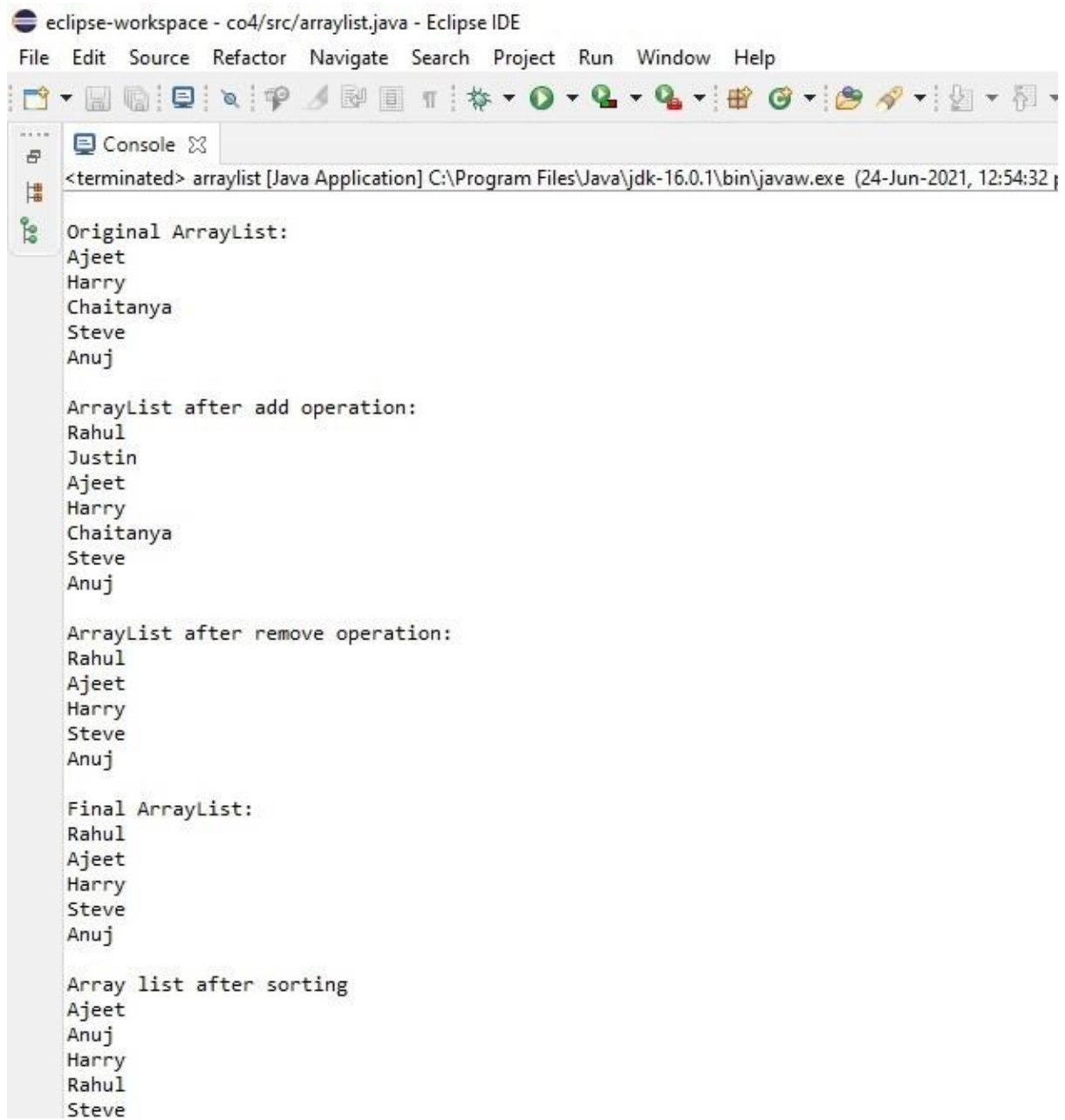


	<pre>System.out.println("\nFinal ArrayList:"); for(String str:obj)     System.out.println(str); //Sort the items// Collections.sort(obj); //Displaying elements// System.out.println(" \nArray list after sorting:"); for(String str : obj)     System.out.println(str); //Access the item// System.out.println("\nObject at index 2:"+obj.get(2)); //Contain or not// System.out.println("\nRahul is in the arraylist:"+obj.contains("Rahul")); System.out.println("\nJustin is in the arraylist:"+obj.contains("Justin")); //Size of list// System.out.println("\nSize of the arraylist :"+obj.size()); // Clear all items// obj.clear(); // Displaying elements System.out.println("\nArrayList after clear:"+obj);  } }</pre>
--	---

## **RESULT**

The above program is executed and obtained the output.

## OUTPUT



The screenshot shows the Eclipse IDE interface with the console window open. The title bar reads "eclipse-workspace - co4/src/arraylist.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations, search, and development. The console window shows the following output:

```
<terminated> arraylist [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (24-Jun-2021, 12:54:32)  
  
Original ArrayList:  
Ajeet  
Harry  
Chaitanya  
Steve  
Anuj  
  
ArrayList after add operation:  
Rahul  
Justin  
Ajeet  
Harry  
Chaitanya  
Steve  
Anuj  
  
ArrayList after remove operation:  
Rahul  
Ajeet  
Harry  
Steve  
Anuj  
  
Final ArrayList:  
Rahul  
Ajeet  
Harry  
Steve  
Anuj  
  
Array list after sorting  
Ajeet  
Anuj  
Harry  
Rahul  
Steve
```

eclipse-workspace - co4/src/arraylist.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

```
<terminated> arraylist [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (24-Jun-2021, 12:54:32 p
Justin
Ajeet
Harry
Chaitanya
Steve
Anuj

ArrayList after remove operation:
Rahul
Ajeet
Harry
Steve
Anuj

Final ArrayList:
Rahul
Ajeet
Harry
Steve
Anuj

Array list after sorting
Ajeet
Anuj
Harry
Rahul
Steve

Object at index 2:Harry

Rahul is in the arraylist:true

Justin is in the arraylist:false

Size of the arraylist :5

ArrayList after clear:[]
```

## **PROGRAM 11**

**AIM:** Program to remove all the elements from a linked list

### **ALGORITHM**

STEP 1 : Start

STEP 2: Define a linked list.

STEP 3: Define objects to Linked list using add().

STEP 4: Delete the elements in list using clear()

STEP 5: Print the Outputs.

STEP 6: Stop

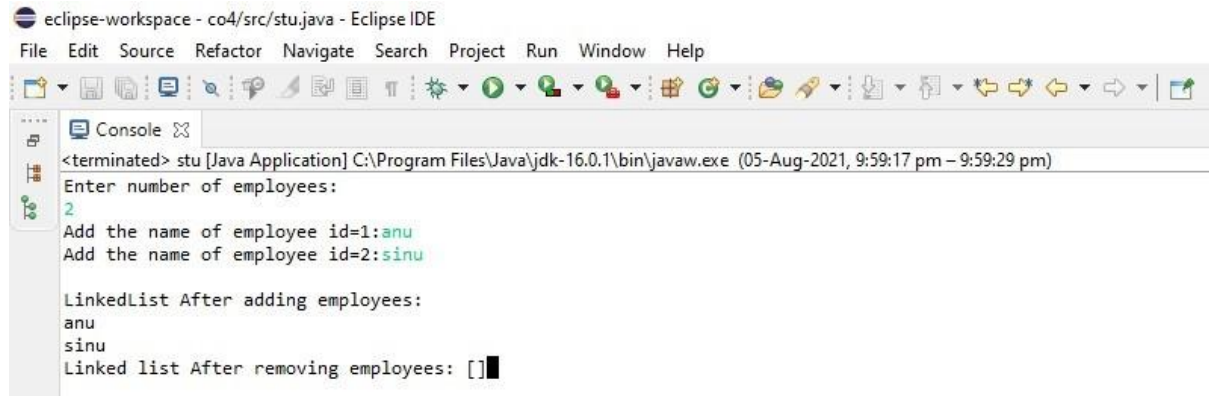
### **SOURCE CODE**

linked list.java	<pre>import java.util.*; public class linked list {     public static void main(String[] args) {         LinkedList&lt;String&gt; list=new LinkedList&lt;String&gt;();         Scanner sc=new Scanner(System.in);         System.out.println("Enter number of employees:");         int num=sc.nextInt();         for(int i=1;i&lt;=num;i++){             System.out.print("Add the name of employee id="+i+":");             String s=sc.next();             list.add(s);         }         System.out.println();         System.out.println("LinkedList After adding employees:");         Iterator&lt;String&gt; itr=list.iterator();         while(itr.hasNext()){             System.out.println(itr.next());         }         list.clear();         System.out.println("Linked list After removing employees: "+list);     } }</pre>
------------------	---

## RESULT

The above program is executed and obtained the output.

## OUTPUT

A screenshot of the Eclipse IDE's console window. The title bar reads 'eclipse-workspace - co4/src/stu.java - Eclipse IDE'. The menu bar includes 'File', 'Edit', 'Source', 'Refactor', 'Navigate', 'Search', 'Project', 'Run', 'Window', and 'Help'. The toolbar contains various icons for file operations, search, and development. The console output shows the execution of a Java application. It starts with a prompt 'Enter number of employees:' followed by the input '2'. Then, it prompts for employee names: 'Add the name of employee id=1:' with input 'anu', and 'Add the name of employee id=2:' with input 'sinu'. The output then displays 'LinkedList After adding employees:' followed by a list containing 'anu' and 'sinu'. Finally, it shows 'Linked list After removing employees: []' with a cursor at the end.

```
<terminated> stu [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (05-Aug-2021, 9:59:17 pm – 9:59:29 pm)
Enter number of employees:
2
Add the name of employee id=1:anu
Add the name of employee id=2:sinu

LinkedList After adding employees:
anu
sinu
Linked list After removing employees: []
```

## **PROGRAM 12**

**AIM:** Program to remove an object from the Stack when the position is passed as parameter

### **ALGORITHM**

STEP 1 : Start

STEP 2: Define a Stack.

STEP 3: Define objects to Stack using add().

STEP 4: Enter key to remove the item.

STEP 5: Remove the item using remove(key).

STEP 6: Print outputs

STEP 7: Stop

### **SOURCE CODE**

stack.java	<pre>import java.util.*; public class stack {     public static void main(String args[])     {         // Creating an empty Stack         Stack&lt;String&gt; stack = new Stack&lt;String&gt;();          // Use add() method to add elements in the Stack         stack.add("Blessy");         stack.add("Libu");         stack.add("Ponnu");         stack.add("Anu");         stack.add("Gopu");          // Output the Stack         System.out.println("Stack: " + stack);          // Remove the element using remove()         String rem = stack.remove(1);          // Print the removed element         System.out.println("Removed element: "+ rem);          // Print the final Stack         System.out.println("Final Stack: "+ stack);     } }</pre>
------------	---

## RESULT

The above program is executed and obtained the output.

## OUTPUT

A screenshot of the Eclipse IDE's console window. The title bar reads 'eclipse-workspace - co4/src/stack.java - Eclipse IDE'. The menu bar includes 'File', 'Edit', 'Source', 'Refactor', 'Navigate', 'Search', 'Project', 'Run', 'Window', and 'Help'. The toolbar contains various icons for file operations, debugging, and navigation. The console output shows the termination of a Java application named 'stack'. The output text is: '<terminated> stack [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (06-Aug-2021, 10:01:12 pm – 10:01:14 pm)', 'Stack: [Blessy, Libu, Ponnu, Anu, Gopu]', 'Removed element: Libu', and 'Final Stack: [Blessy, Ponnu, Anu, Gopu]' followed by a cursor.

```
<terminated> stack [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (06-Aug-2021, 10:01:12 pm – 10:01:14 pm)
Stack: [Blessy, Libu, Ponnu, Anu, Gopu]
Removed element: Libu
Final Stack: [Blessy, Ponnu, Anu, Gopu]
```

### **PROGRAM 13**

**AIM:** Program to demonstrate the creation of queue object using the PriorityQueue class

#### **ALGORITHM**

STEP 1 : Start

STEP 2: Define a Priority Queue.

STEP 3: Enter a limit n.

STEP 4: Add n elements to Queue.

STEP 5: Perform operations like remove(), add() etc.

STEP 6: Print outputs

STEP 7: Stop

#### **SOURCE CODE**

priQueue.java	<pre>import java.util.Iterator; import java.util.PriorityQueue; import java.util.Scanner; public class priQueue {     public static void main(String args[])     {         PriorityQueue&lt;String&gt; queue=new PriorityQueue&lt;String&gt;();         Scanner sc=new Scanner(System.in);         System.out.println("Enter Number Of elements ");         int n=sc.nextInt();         System.out.println("Enter the elements ");         for(int i =0;i&lt;n;i++)         {             String st=sc.next();             queue.add(st);          }         System.out.println("head:"+queue.element());         System.out.println("head:"+queue.peek());         System.out.println("Iterating the queue elements\n ");         Iterator itr=queue.iterator();         while(itr.hasNext()){             System.out.println(itr.next());         }         queue.remove();     } }</pre>
---------------	---



	<pre> queue.poll(); System.out.println("After removing two elements \n"); Iterator&lt;String&gt; itr2=queue.iterator(); while(itr2.hasNext()){     System.out.println(itr2.next()); } } } </pre>
--	--

## **RESULT**

The above program is executed and obtained the output.

## **OUTPUT**

```

eclipse-workspace - co4/src/priQueue.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
<terminated> priQueue [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (08-Aug-2021, 10:38:23 pm - 10:38:48 pm)
Enter Number Of elements
6
Enter the elements
1
2
3
4
5
6
head:1
head:1
Iterating the queue elements
1
2
3
4
5
6
After removing two elements
3
4
5
6

```

## **PROGRAM 14**

**AIM** : Program to demonstrate the addition and deletion of elements in deque

### **ALGORITHM**

STEP 1 : Start

STEP 2: Define a dequeue.

STEP 3: Insert the limit and numbers.

STEP 4: Perform pop() and remove() operations.

STEP 5: Print outputs

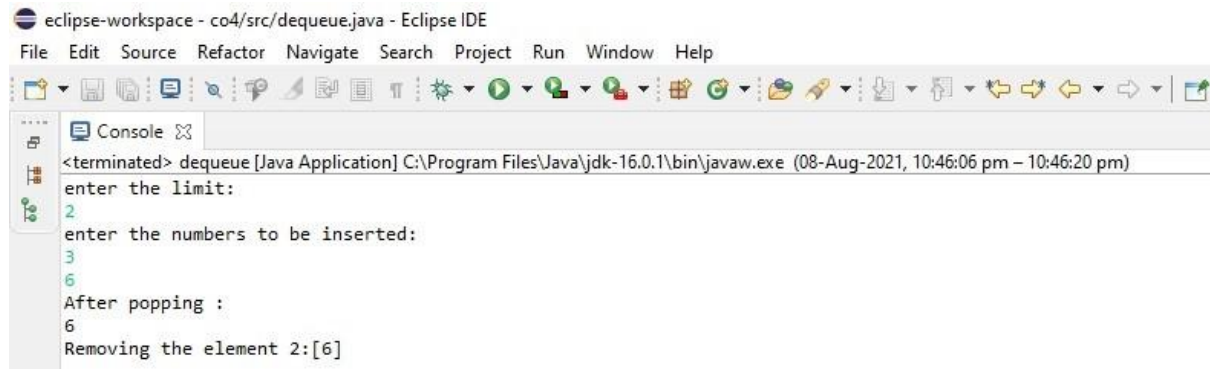
STEP 6: Stop

### **SOURCE CODE:**

dequeue.java	<pre>import java.util.*; public class dequeue {     public static void main(String[] args) {         Deque&lt;Integer&gt; deque = new ArrayDeque&lt;Integer&gt;();         // Inserts the elements         Scanner sc=new Scanner(System.in);         System.out.println("enter the limit:");         int n=sc.nextInt();         System.out.println("enter the numbers to be inserted: ");         for(int i =0;i&lt;n;i++)         {             Integer obj=sc.nextInt();             deque.add(obj);         }          // Popping the element         deque.pop();         System.out.println("After popping : ");         for (Integer integer : deque) {             System.out.println(integer);         }         deque.remove(2);         System.out.println("Removing the element 2:"+deque);     } }</pre>
--------------	--

**RESULT** : The above program is executed and obtained the output.

**OUTPUT:**



The screenshot shows the Eclipse IDE interface with the console window open. The title bar reads "eclipse-workspace - co4/src/dequeue.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations, debugging, and navigation. The console output is as follows:

```
<terminated> dequeue [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (08-Aug-2021, 10:46:06 pm - 10:46:20 pm)
enter the limit:
2
enter the numbers to be inserted:
3
6
After popping :
6
Removing the element 2:[6]
```

## **PROGRAM 15**

**AIM:** Program to demonstrate the creation of Set object using the LinkedHashSet class

### **ALGORITHM**

STEP 1 : Start

STEP 2: Define a hashset.

STEP 3: Define objects to Linked hash set using add().

STEP 4: Perform Desired operations like adding already existing elements.

STEP 5: Remove the item using remove(key).

STEP 6: Print outputs

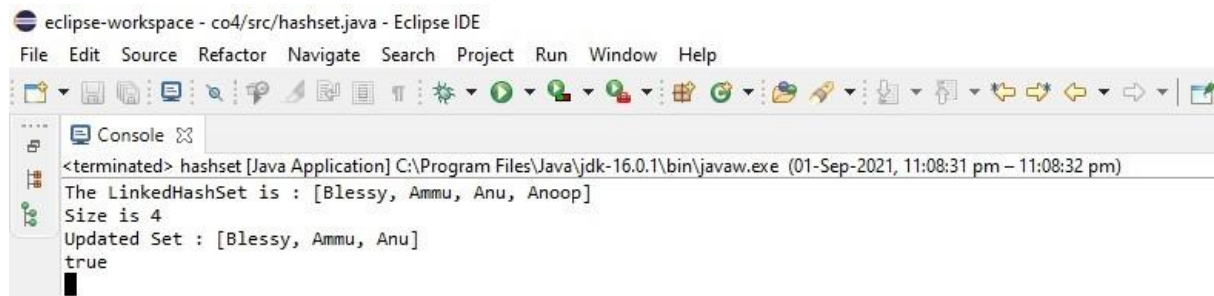
STEP 7: Stop

### **SOURCE CODE:**

hashset.java	<pre>import java.util.*; public class hashset {     public static void main(String[] args) {          LinkedHashSet&lt;String&gt; l = new LinkedHashSet&lt;String&gt;();          // Inserting data         l.add("Blessy");         l.add("Ammu");         l.add("Anu");         l.add("Anoop");         System.out.println("The LinkedHashSet is : "+l);          //Size         System.out.println("Size is "+l.size());          //removing         l.remove("Anoop");         System.out.println("Updated Set : "+l);          //Checking         System.out.println(l.contains("Blessy"));      } }</pre>
--------------	---

**RESULT** : The above program is successfully executed and obtained the output.

**OUTPUT:**

A screenshot of the Eclipse IDE's console window. The title bar reads 'eclipse-workspace - co4/src/hashset.java - Eclipse IDE'. The menu bar includes 'File', 'Edit', 'Source', 'Refactor', 'Navigate', 'Search', 'Project', 'Run', 'Window', and 'Help'. The toolbar contains various icons for file operations, search, and execution. The console output shows the following text: '<terminated> hashset [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (01-Sep-2021, 11:08:31 pm - 11:08:32 pm)', 'The LinkedHashSet is : [Blessy, Ammu, Anu, Anoop]', 'Size is 4', 'Updated Set : [Blessy, Ammu, Anu]', and 'true'. A small black cursor is visible at the end of the last line.

```
<terminated> hashset [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (01-Sep-2021, 11:08:31 pm - 11:08:32 pm)
The LinkedHashSet is : [Blessy, Ammu, Anu, Anoop]
Size is 4
Updated Set : [Blessy, Ammu, Anu]
true
```

## **PROGRAM 16**

**AIM** : Write a Java program to compare two hash set

### **ALGORITHM**

STEP 1 : Start

STEP 2: Define 2 hash sets h1 and h1.

STEP 3: Define objects to hash sets h\_set and h\_set2.

STEP 4: Compare the hash sets; if its same; .

STEP 5: Print the hashes are same

STEP 6: Else Print hashes are different

STEP 7: Stop

### **SOURCE CODE:**

color.java	<pre>import java.util.HashSet;  public class color {     public static void main(String[] args)     {          HashSet&lt;String&gt; h_set = new HashSet&lt;String&gt;();          h_set.add("Red");         h_set.add("Green");         h_set.add("Black");         h_set.add("White");          HashSet&lt;String&gt;h_set2 = new HashSet&lt;String&gt;();         h_set2.add("Red");         h_set2.add("Pink");         h_set2.add("Black");         h_set2.add("Orange");          System.out.println("Comparing");          HashSet&lt;String&gt;result_set = new HashSet&lt;String&gt;();         for (String element : h_set){             System.out.println(h_set2.contains(element) ? "Sets are same:Yes" : "Sets are same:No");         }     } }</pre>
------------	---

**RESULT** : The above program is successfully executed and obtained the output.

**OUTPUT:**

A screenshot of the Eclipse IDE's console window. The title bar reads 'eclipse-workspace - co4/src/color.java - Eclipse IDE'. The menu bar includes 'File', 'Edit', 'Source', 'Refactor', 'Navigate', 'Search', 'Project', 'Run', 'Window', and 'Help'. The toolbar contains various icons for file operations, debugging, and navigation. The console output shows the following text:

```
<terminated> color [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (01-Sep-2021, 11:10:34 pm - 11:10:35 pm)
Comparing
Sets are same:Yes
Sets are same:No
Sets are same:Yes
Sets are same:No
```

## **PROGRAM 17**

**AIM** : Program to demonstrate the working of Map interface by adding, changing and removing elements.

### **ALGORITHM**

STEP 1 : Start

STEP 2: Define a Map set maphash.

STEP 3: Define objects to Map set maphash using put().

STEP 4: Remove the elements using remove(); .

STEP 5: Print the Outputs.

STEP 6: Stop

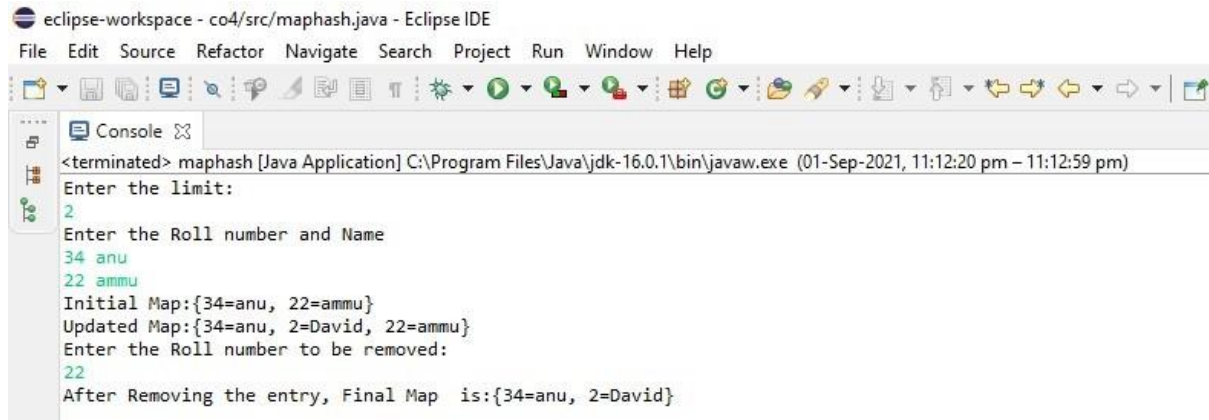
### **SOURCE CODE:**

maphash.java	<pre>import java.util.*; public class maphash {     public static void main(String args[])     {         Map&lt;Integer, String&gt; mp = new HashMap&lt;&gt;();         //Inserting elements..         System.out.println("Enter the limit:");         Scanner inp = new Scanner(System.in);         int n= inp.nextInt();         System.out.println("Enter the Roll number and Name");         while(n!=0) {             int e= inp.nextInt();             String s= inp.next();             mp.put(e, s);             n--;         }         System.out.println("Initial Map:"+mp);         mp.put( 2, "David");         //Updating..         System.out.println("Updated Map:"+mp);         //Removing..         System.out.println("Enter the Roll number to be removed:");         int r=inp.nextInt();         mp.remove(r);         // Final Map..         System.out.println("After Removing the entry, Final Map is:"+mp);      } }</pre>
--------------	---



**RESULT** : The above program is successfully executed and obtained the output.

## **OUTPUT:**



```
eclipse-workspace - co4/src/maphash.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

<terminated> maphash [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (01-Sep-2021, 11:12:20 pm - 11:12:59 pm)
Enter the limit:
2
Enter the Roll number and Name
34 anu
22 ammu
Initial Map:{34=anu, 22=ammu}
Updated Map:{34=anu, 2=David, 22=ammu}
Enter the Roll number to be removed:
22
After Removing the entry, Final Map is:{34=anu, 2=David}
```

## **PROGRAM 18**

**AIM** : Program to Convert HashMap to TreeMap.

### **ALGORITHM**

STEP 1 : Start

STEP 2: Define a map mp.

STEP 3: Define objects to mp using put().

STEP 4: Define a tree set treeMap

STEP 4: Convert map set into tree using treeMap.putAll(map).

STEP 5: Print the Outputs.

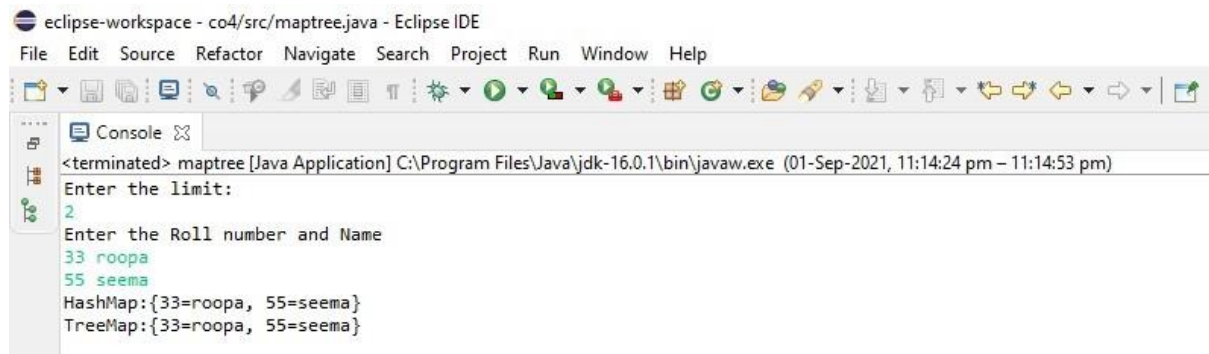
STEP 6: Stop

### **SOURCE CODE:**

maptree.java	<pre>import java.util.*;  public class maptree {     public static void main(String args[]) {          Map&lt;String, String&gt; map = new HashMap&lt;&gt;();         System.out.println("Enter the limit:");         Scanner inp = new Scanner(System.in);         int n= inp.nextInt();         System.out.println("Enter the Roll number and Name");         while(n!=0) {              String e= inp.next();             String s= inp.next();             map.put(e, s);             n--;         }          System.out.println("HashMap:"+map);         Map&lt;String, String&gt; treeMap = new TreeMap&lt;&gt;();         treeMap.putAll(map);         System.out.println("TreeMap:"+treeMap);     } }</pre>
--------------	---

**RESULT** : The above program is successfully executed and obtained the output.

## **OUTPUT:**



The screenshot shows the Eclipse IDE interface. The title bar reads 'eclipse-workspace - co4/src/maptree.java - Eclipse IDE'. The menu bar includes 'File', 'Edit', 'Source', 'Refactor', 'Navigate', 'Search', 'Project', 'Run', 'Window', and 'Help'. The toolbar contains various icons for file operations, editing, and running. The 'Console' window is active, displaying the following output:

```
<terminated> maptree [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (01-Sep-2021, 11:14:24 pm – 11:14:53 pm)
Enter the limit:
2
Enter the Roll number and Name
33 roopa
55 seema
HashMap:{33=roopa, 55=seema}
TreeMap:{33=roopa, 55=seema}
```