COURSE OUTCOME 3

PROGRAM NO: 1

AIM:

To find area of different shapes using overloaded functions.

ALGORITHM:

Step 1: Start

Step 2: Define the main class

Step 3: Define methods with the same methodname that performs the area operation for each shape

Step 4: Display the areas of each shapes.

PROGRAM CODE:

CO3Q1.java

```
import java.util.Scanner;

public class CO3Q1
{
   double area(float r)
   {
      double pi = 3.14;
      double ar;
      ar = pi*r*r;
      return ar;
   }
   double area(float h,float b)
   {
      double ar;
      ar = (h*b)/2;
      return ar;
   }
   double area(double s)
   {
      double ar;
      ar = s*s;
      return ar;
   }
```

```
    double area(double l,double br)
    {
       double ar;
       ar = l*br;
       return ar;
    }
    public static void main(String[] args)
    {
       CO3Q1 obj = new CO3Q1();
       int ch;
       float r,h,b;
       double s,l,br;
       System.out.println("Enter the option:");
       System.out.println("1. Area of the circle");
       System.out.println("2. Area of the triangle");
       System.out.println("3. Area of the square");
       System.out.println("4. Area of the rectangle");
       Scanner sc = new Scanner(System.in);
       System.out.println("Enter the choice");
       ch = sc.nextInt();
       switch(ch)
       {
          case 1: System.out.println("Enter the radius:");
                  r = sc.nextFloat();
                  System.out.println(obj.area(r));
                  break;
          case 2: System.out.println("Enter the height and breadth:");
                  h = sc.nextFloat();
                  b = sc.nextFloat();
                  System.out.println(obj.area(h,b));
                  break;
          case 3: System.out.println("Enter the length of the side:");
                  s = sc.nextDouble();
                  System.out.println(obj.area(s));
                  break;
          case 4: System.out.println("Enter the length and breadth:");
                  l = sc.nextDouble();
                  br = sc.nextDouble();
                  System.out.println(obj.area(l,br));
                  break;
          default: System.out.println("Invalid choice.");
       }
    }
}
```

OUTPUT:

```
run:
Enter the option:
1. Area of the circle
2. Area of the triangle
3. Area of the square
4. Area of the rectangle
Enter the choice
1
Enter the radius:
5
78.5
BUILD SUCCESSFUL (total time: 13 seconds)
```

```
run:
Enter the option:
1. Area of the circle
2. Area of the triangle
3. Area of the square
4. Area of the rectangle
Enter the choice
2
Enter the height and breadth:
10
5
25.0
BUILD SUCCESSFUL (total time: 36 seconds)
```

```
run:
Enter the option:
1. Area of the circle
2. Area of the triangle
3. Area of the square
4. Area of the rectangle
Enter the choice
4
Enter the length and breadth:
4
6
24.0
BUILD SUCCESSFUL (total time: 10 seconds)
```

```
run:
Enter the option:
1. Area of the circle
2. Area of the triangle
3. Area of the square
4. Area of the rectangle
Enter the choice
3
Enter the length of the side:
5
25.0
BUILD SUCCESSFUL (total time: 10 seconds)
```

RESULT:

The program is successfully executed and the output is verified.

PROGRAM NO: 2

AIM:

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

ALGORITHM:

Step.1: Start the program.

Step.2: Define a class '*Employee*' with data members Empid, Name, Salary, Address and a constructor to initialize these members.

Step.3: Define a class '*Teacher*' that inherit the properties of class 'Employee' and contain its own data members Department, Subjects taught and constructors to initialize these data members and also include a method Display() to display all the data members.

Step.4: Define a main() to create array of objects for the class to display the details of 'N' teachers.

Step.5: Stop the program.

PROGRAM CODE:

employee.java	class Employee2
	{
	int Empid;
	String Name;
	float Salary;
	String Address;
	
	Employee2()
	{
	
	}
	public Employee2(int id, String name, float sal, String addr)
	{
	Empid = id;
	Name = name;
	Salary = sal;

```
        Address = addr;
    }

}
class Teacher extends Employee2
{
    String department;
    String Subjects;

    public Teacher(int id, String name, float sal, String addr, String dept, String sub)
    {
        super(id, name, sal, addr);
        department = dept;
        Subjects = sub;

    }

    Teacher()
    {

    }

    public void display()
    {
        System.out.println("Employee ID - "+Empid);
        System.out.println("Employee Name - "+Name);
        System.out.println("Salary - "+Salary);
        System.out.println("Address - "+Address);
        System.out.println("Department - "+department);
        System.out.println("Subject - "+Subjects);
    }

}
public class CO3Q2
{
    public static void main(String[] args)
    {
        int n;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of Teachers to be added:");
        n = sc.nextInt();
        Teacher obj[] = new Teacher[n];
        for (int i=0; i<n; i++)
        {
            obj[i] = new Teacher();
        }
        for(int i=0;i<n;i++)
        {
            System.out.println("\t*****");
            System.out.println("Enter Employee ID");
            obj[i].Empid = sc.nextInt();
```

```
            sc.nextLine();
            System.out.println("Enter Employee Name");
            obj[i].Name = sc.nextLine();
            System.out.println("Enter Employee Salary");
            obj[i].Salary = sc.nextFloat();
            sc.nextLine();
            System.out.println("Enter Employee Address");
            obj[i].Address = sc.nextLine();
            System.out.println("Enter Employee Department");
            obj[i].department = sc.nextLine();
            System.out.println("Enter Employee Subject");
            obj[i].Subjects = sc.nextLine();
        }
        System.out.println("\t***************");
        System.out.println("Employee Details:-");
        for(int i=0;i<n;i++)
            obj[i].display();
    }
}
```

OUTPUT:

```
run:
Enter the number of Teachers to be added:
2
        *****
Enter Employee ID
101
Enter Employee Name
Dustin
Enter Employee Salary
60000
Enter Employee Address
London
Enter Employee Department
Literature
Enter Employee Subject
English
        *****
Enter Employee ID
102
Enter Employee Name
Millie
Enter Employee Salary
85000
Enter Employee Address
New York
Enter Employee Department
Science
Enter Employee Subject
Physics
        ***************
Employee Details:-
Employee ID - 101
Employee Name - Dustin
Salary - 60000.0
Address - London
Department - Literature
Subject - English
Employee ID - 102
Employee Name - Millie
Salary - 85000.0
Address - New York
Department - Science
Subject - Physics
BUILD SUCCESSFUL (total time: 1 minute 10 seconds)
```

RESULT:

The program is successfully executed and the output is verified.

PROGRAM NO: 3

AIM:

To 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, Company_name, 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 also contain constructors and methods to display the data members. Use array of objects to display details of N teachers.

ALGORITHM:

Step 1: Start

Step 2: Create a class named '*Person'* with data members name, gender, address and age & a constructor to initialize them.

Step 3: Create a class named '*Employee'* which is derived from Person, with data members empid, cmpnyname, qualification and sal & a constructor Employee() to initialize them.

Step 4: Create class named 'Teach" which is derived from Employee, with data members subject, dept and tid ; a constructor to initilize members ; and a function named display() to display details.

Step 5: Create an array of objects to display details.

Step 6: Stop

PROGRAM CODE:

CO3Q3.java	**import java.util.Scanner;** **class Person** **{** **String Name;** **String Gender;** **String Address;** **int Age;**

```
    Person()
    {

    }
    Person(String name, String gender, String addr, int age)
    {
        Name = name;
        Gender = gender;
        Address = addr;
        Age = age;
    }
}

class Employee extends Person
{
    int Empid;
    String Company_name;
    String Qualification;
    float Salary;
    Employee()
    {

    }
    public Employee(String name, String gender, String addr, int age)
    {
        super(name, gender, addr, age);
    }
    public Employee(int id,String name, String qual, float sal)
    {
        Empid = id;
        Company_name = name;
        Qualification = qual;
        Salary = sal;
    }

}
class Teacher extends Employee
{
    String Subject;
    String Department;
    String Teachersid;
    Teacher()
    {

    }
    Teacher(String sub, String dept, String id)
```

```
    {
        Subject = sub;
        Department = dept;
        Teachersid = id;
    }
    public void display()
    {
        System.out.println("Name:" + Name);
            System.out.println("Age:" + Age);
            System.out.println("Gender:" + Gender);
            System.out.println("Address:" + Address);
            System.out.println("Emp id:" + Empid);
            System.out.println("Salary:" + Salary);
            System.out.println("Qualification:" + Qualification);
            System.out.println("Company Name:" + Company_name);
            System.out.println("Teacher id:" + Teachersid);
            System.out.println("Subject:" + Subject);
            System.out.println("Department:" + Department);
        System.out.println("\n\n");
    }
}

public class CO3Q3
{
    public static void main(String[] args)
    {
        int n;
        System.out.println("Enter the no. of Teachers:");
        Scanner sc = new Scanner(System.in);
        n = sc.nextInt();
        Teacher obj[] = new Teacher[n];
        for(int i=0;i<n;i++)
            obj[i] = new Teacher();
        sc.nextLine();
        for(int i=0;i<n;i++)
        {
            System.out.println("\t*****");
            System.out.println("Enter the name:");
            obj[i].Name = sc.nextLine();

            System.out.println("Enter the Age:");
            obj[i].Age = sc.nextInt();
            sc.nextLine();
            System.out.println("Enter the Gender:");
            obj[i].Gender = sc.nextLine();
            System.out.println("Enter the Address:");
```

```
            obj[i].Address = sc.nextLine();
            System.out.println("Enter the Emp id:");
            obj[i].Empid = sc.nextInt();
            System.out.println("Enter the Salary:");
            obj[i].Salary = sc.nextFloat();
            sc.nextLine();
            System.out.println("Enter the Qualification:");
            obj[i].Qualification = sc.nextLine();
            System.out.println("Enter the Company Name:");
            obj[i].Company_name = sc.nextLine();
            System.out.println("Enter the Teacher id:");
            obj[i].Teachersid = sc.nextLine();
            System.out.println("Enter the Subject:");
            obj[i].Subject = sc.nextLine();
            System.out.println("Enter the Department:");
            obj[i].Department = sc.nextLine();
        }
        System.out.println("\t***************");
        System.out.println("Teachers Details:-\n");
        for(int i=0;i<n;i++)
            obj[i].display();
    }

}
```

OUTPUT:

```
run:
Enter the no. of Teachers:
2
        *****
Enter the name:
Mike
Enter the Age:
19
Enter the Gender:
Male
Enter the Address:
Germany
Enter the Emp id:
101
Enter the Salary:
54000
Enter the Qualification:
MCA
Enter the Company Name:
Google
Enter the Teacher id:
101
Enter the Subject:
Science
Enter the Department:
Physics
        *****
Enter the name:
Nancy
Enter the Age:
20
Enter the Gender:
Female
Enter the Address:
Australia
Enter the Emp id:
102
Enter the Salary:
70000
Enter the Qualification:
Msc
Enter the Company Name:
Microsoft
Enter the Teacher id:
102
Enter the Subject:
Statistics
Enter the Department:
Maths
        ***************
```

```
Teachers Details:-

Name:Mike
Age:19
Gender:Male
Address:Germany
Emp id:101
Salary:54000.0
Qualification:MCA
Company Name:Google
Teacher id:101
Subject:Science
Department:Physics


Name:Nancy
Age:20
Gender:Female
Address:Australia
Emp id:102
Salary:70000.0
Qualification:Msc
Company Name:Microsoft
Teacher id:102
Subject:Statistics
Department:Maths
```

RESULT:

The program is successfully executed and the output is verified.

PROGRAM NO: 4

AIM:

To write a program has class Publisher, Book, Literature and Fiction. Read the information and print the details of books from either the category, using inheritance.

ALGORITHM:

Step 1: Start

Step 2:Create a class named ‘Publisher’ with data members pname, pid; a constructor named

Publisher().

Step 3: Create a class named ‘Book’ which is derived ‘Publisher’ with data members nop, price; a constructor named Book().

Step 4: Create a class named ‘literature’ which is derived from Book with data members title, author; a constructor; a function show() to display details.

Step 5: Create a class named ‘fiction’ which is derived from Book with data members bname, auth; a constructor; a function display() to print details.

Step 6: Print a menu defining the type of genres; if literature create an object of literature

type and object of type fiction if fiction is chosen.

Step 7 : Stop

PROGRAM CODE:

CO3Q4.java	**import java.util.Scanner;** **public class CO3Q4** **{** **public class publisher** **{** **String pub_name;** **publisher(){}** **publisher(String name)** **{** **pub_name = name;** **}** **}** **static public class Book extends publisher** **{**

```
    String book_name;
    Book(){}
    Book(String bname, String pname)
    {
       super(pname);
       book_name = bname;
    }
    public void display()
    {
       System.out.println("\n\n**********\n\nBook Details");
       System.out.println("Publisher:"+ pub_name);
       System.out.println("Book:"+book_name);
    }
}
static public class literature extends Book
{
    String book_genre;

    public literature() {}
    literature(String name,String book,String genre)
    {
       super(name,book);
       book_genre = genre;
       super.display();
       System.out.println("Genre:"+book_genre);
    }

}
static public class fiction extends Book
{
    String book_genre;
    fiction(){}
    fiction(String name,String book,String genre)
    {
       super(name,book);
       book_genre = genre;
       super.display();
       System.out.println("Genre:"+book_genre);
    }
}


public static void main(String[] args)
{
    String name,book,genre;
```

```
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the book details...");
        System.out.println("Enter the Book name:");
        book = sc.nextLine();
        System.out.println("Enter the Publisher name:");
        name = sc.nextLine();
        System.out.println("Enter the Genre name:");
        genre = sc.nextLine();
        fiction obj;
        literature ob;
        if(genre.toLowerCase().equals("fiction"))
        {
           obj = new fiction(name,book,genre);
        }
        else if(genre.toLowerCase().equals("literature"))
           ob = new literature(name,book,genre);
        else
           System.out.println("Enter Fiction or Literature");
    }
}
```

OUTPUT:

```
run:
Enter the book details...
Enter the Book name:
Harry Potter
Enter the Publisher name:
Pottermore Publishing
Enter the Genre name:
Fiction


**********

Book Details
Publisher:Harry Potter
Book:Pottermore Publishing
Genre:Fiction
BUILD SUCCESSFUL (total time: 22 seconds)
```

RESULT:

The program is successfully executed and the output is verified.

PROGRAM NO: 5

AIM:

To 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: Start the program.

Step.2: Define a class '*Student*' which will read a student's academic information from the user.

Step.3: Define another class '*Sports*' that extends '*Student*' and reads the sports data of the student.

Step.4: Define another interface 'R*esults*' that extends '*Sports*'and has a *Display()* to display the profile, academic score and sports score of the student.

Step.5: Define a main () method to create objects for the above classes and to call the associated member methods.

Step.6: Stop the program.

PROGRAM CODE:

CO3Q5.java	import java.util.Scanner;
	
	interface results
	{
	  void getdata();
	  int display();
	}
	
	class Student implements results
	{
	  int std_id,std_tmark;
	  String std_name;
	  @Override
	  public void getdata()
	  {
	  Scanner sc = new Scanner(System.in);

```
        System.out.println("Enter the name of the student:");
        std_name = sc.nextLine();
        System.out.println("Enter the student id:");
        std_id = sc.nextInt();
        System.out.println("Enter total academic mark:");
        std_tmark = sc.nextInt();
    }
    @Override
    public int display()
    {
        System.out.println("\t--------Student details--------");
        System.out.println("Student name: " +std_name);
        System.out.println("Student id: " +std_id);
        System.out.println("Total mark:" +std_tmark);
        return std_tmark;
    }
}

class Sports implements results
{
    int tmarks;
    @Override
    public void getdata()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the marks obtained in sports:");
        tmarks = sc.nextInt();
    }
    @Override
    public int display()
    {
        System.out.println("Marks obtained in Sports:"+ tmarks);
        return tmarks;
    }

}
public class CO3Q5
{
    public static void main(String[] args)
    {
```

```
        int mark;
        Student ob = new Student();
        Sports obj = new Sports();
        ob.getdata();
        obj.getdata();
        mark = ob.display();
        mark = mark + obj.display();
        System.out.println("Marks(Academic+Sports)="+ mark);
    }
}
```

OUTPUT:

```
run:
Enter the name of the student:
Tokyo
Enter the student id:
501
Enter total academic mark:
200
Enter the marks obtained in sports:
25
        --------Student details--------
Student name: Tokyo
Student id: 501
Total mark:200
Marks obtained in Sports:25
Marks(Academic+Sports)=225
BUILD SUCCESSFUL (total time: 26 seconds)
```

RESULT:

The program is successfully executed and the output is verified.

PROGRAM NO: 6

AIM:

To create an interface having prototypes of functions area() and perimeter(). Create two classes Circle and Rectangle which implements the above interface. Create a menu driven program to find area and perimeter of objects.

ALGORITHM:

Step.1: Start the program.

Step.2: Define an interface '*prop*' with methods to read inputs and calculate area and perimeter.

Step.3: Define a class '*Rectangle*' that extends *Circle* to initialize its data members l, b and to calculate and display the area and perimeter of a rectangle.

Step.4: Define a main () to create objects for the above classes to invoke its member methods to print the results.

Step.5: Stop the program.

PROGRAM CODE:

CO3Q6.java	import java.util.Scanner;
	interface prop{
	void getdata();
	void area();
	void perimeter();
	}
	class Circle implements prop{
	double pi = 3.14;
	double r;
	Scanner sc = new Scanner(System.in);
	@Override
	public void getdata(){
	System.out.println("Enter the radius of the circle:");
	r = sc.nextDouble();
	}
	@Override
	public void perimeter(){
	System.out.println("Perimeter of the circle: "+(2*pi*r));

```
    }
    @Override
    public void area(){
        System.out.println("Perimeter of the circle: "+(pi*r*r));
    }
}
class Rectangle implements prop{
    double l,b;
    Scanner sc = new Scanner(System.in);
    @Override
    public void getdata(){
        System.out.println("Enter the length of the rectangle:");
        l = sc.nextDouble();
        System.out.println("Enter the breadth of the rectangle:");
        b = sc.nextDouble();
    }
    @Override
    public void area(){
        System.out.println("Perimeter of a rectangle: "+(l*b));
    }
    @Override
    public void perimeter(){
        System.out.println("Perimeter of a rectangle: "+(2*(l+b)));
    }
}
public class CO3Q6 {
    public static void main(String[] args) {
        int ch;
        Scanner sc = new Scanner(System.in);
        Circle ob = new Circle();
        Rectangle obj = new Rectangle();
        do{
            System.out.println("\n1.Circle\n2.Rectangle\n3.exit");
            System.out.println("Enter your choice:");
            ch = sc.nextInt();
            switch(ch){
                case 1 :ob.getdata();
                        ob.area();
                        ob.perimeter();
                        break;
                case 2 :obj.getdata();
                        obj.area();
                        obj.perimeter();
                        break;
                case 3 :System.out.println("Exited...");
                        System.exit(0);
```

```
            }
        }while(true);
    }
}
```

OUTPUT:

```
run:

1.Circle
2.Rectangle
3.exit
Enter your choice:
1
Enter the radius of the circle:
5
Perimeter of the circle: 78.5
Perimeter of the circle: 31.400000000000002

1.Circle
2.Rectangle
3.exit
Enter your choice:
2
Enter the length of the rectangle:
10
Enter the breadth of the rectangle:
6
Perimeter of a rectangle: 60.0
Perimeter of a rectangle: 32.0

1.Circle
2.Rectangle
3.exit
Enter your choice:
3
Exited...
BUILD SUCCESSFUL (total time: 20 seconds)
```

RESULT:

The program is successfully executed and the output is verified.

PROGRAM NO: 7

AIM:

To 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: Start the program.

Step.2: Define an interface *calc* with a method calculate().

Step.3: Define a class *bill* that implements *calc* to calculate the total amount for each product and has methods to generate a bill as given in the question.

Step.4: Define a main () to create objects for the class.

Step.5: Invoke the above methods by passing the data collected from user to generate the required bill.

Step.6: Stop the program.

PROGRAM CODE:

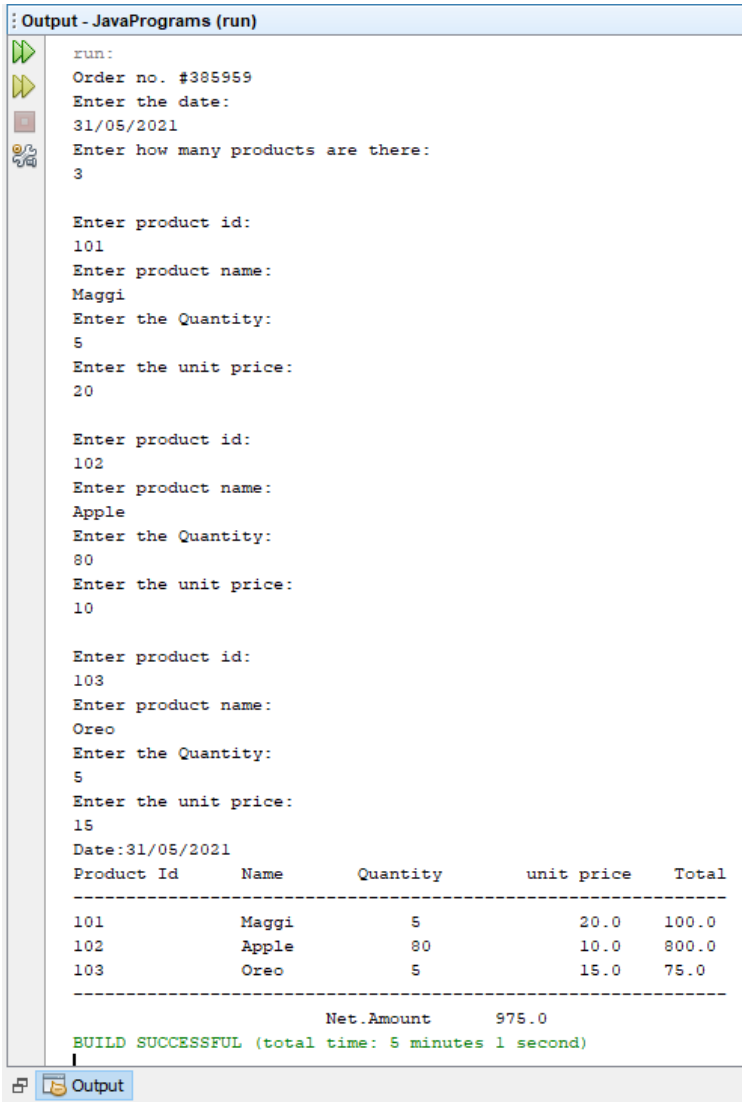
CO3Q7.java

```
import java.util.Scanner;
interface calc{
   void calculate();
}
class bill implements calc{
   String date,name,p_id;
   int quantity;
```

```
    double unit_price,total,namount=0;
    Scanner sc = new Scanner(System.in);
    public void getdata(){
        System.out.println("\nEnter product id:");
        p_id = sc.nextLine();
        System.out.println("Enter product name:");
        name = sc.nextLine();
        System.out.println("Enter the Quantity:");
        quantity = sc.nextInt();
        System.out.println("Enter the unit price:");
        unit_price = sc.nextDouble();
    }
    @Override
    public void calculate(){
        total = quantity * unit_price;
    }
    public void display(){
        System.out.println(p_id+"\t\t"+name+"\t\t"+quantity+"\t\t"+unit_price+"\t"+total);
    }
}
public class CO3Q7 {
    public static void main(String[] args) {
        int n,i;
        double namount=0,t;
        int ran;
        String date;
        t = Math.random() *1000000;
        ran = (int) t;
        Scanner sc = new Scanner(System.in);
        System.out.println("Order no. #"+ran);
        System.out.println("Enter the date:");
        date = sc.nextLine();
        System.out.println("Enter how many products are there:");
        n = sc.nextInt();
        bill ob[] = new bill[n];
        for(i=0;i<n;i++)
            ob[i] = new bill();
        for(i=0;i<n;i++){
            ob[i].getdata();
            ob[i].calculate();
        }
        System.out.println("Date:"+date);
        System.out.println("Product Id \tName\t   Quantity\t   unit price\t Total ");
        System.out.println("-----------------------------------------------------------");
        for(i=0;i<n;i++){
            ob[i].display();
```

```
            namount += ob[i].total;
        }
        System.out.println("-----------------------------------------------------------------");
        System.out.println("\t\t\tNet.Amount\t"+ namount);

    }}
```

OUTPUT:



```
Output - JavaPrograms (run)
run:
Order no. #385959
Enter the date:
31/05/2021
Enter how many products are there:
3

Enter product id:
101
Enter product name:
Maggi
Enter the Quantity:
5
Enter the unit price:
20

Enter product id:
102
Enter product name:
Apple
Enter the Quantity:
80
Enter the unit price:
10

Enter product id:
103
Enter product name:
Oreo
Enter the Quantity:
5
Enter the unit price:
15
Date:31/05/2021
Product Id      Name        Quantity        unit price     Total
-----------------------------------------------------------------
101             Maggi           5               20.0    100.0
102             Apple           80              10.0    800.0
103             Oreo            5               15.0    75.0
-----------------------------------------------------------------
                        Net.Amount      975.0
BUILD SUCCESSFUL (total time: 5 minutes 1 second)
Output
```

RESULT:

The program is successfully executed and the output is verified.

COURSE OUTCOME 4

PROGRAM NO-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 class having name Product and members as pcode,pname and price.

Step 3: Declare three objects in the class and add the values of each data members into objects.

Step 4: Using if condition check which object has the lowest price and print it.

Step 5: Stop.

CODE:

Graphics.Shapes.java	package Graphics;
	interface figures{
	void Rectangle(double a,double b);
	void Triangle(double a,double b);
	void Square(double a);
	void Circle(double a);
	}
	public class Shapes implements figures{
	@Override
	public void Rectangle(double a,double b){
	System.out.println("Area:"+(a*b));
	}
	@Override
	public void Triangle(double a,double b){
	System.out.println("Area:"+((a*b)/2));
	}
	@Override
	public void Square(double a) {
	System.out.println("Area:"+(a*a));
	}
	@Override
	public void Circle(double a) {
	double pi = 3.14;

	System.out.println("Area:"+(pi*a*a));
	}
	public static void main(String[] args) {
	}
	}

CO4Q1.java	import Graphics.Shapes;
	import java.util.Scanner;
	
	public class CO4Q1 {
	public static void main(String[] args) {
	int ch;
	double l,b;
	Shapes ob = new Shapes();
	Scanner sc = new Scanner(System.in);
	do
	{
	System.out.println("\n1.Rectangle\n2.Triangle\n3.Square\n4.Circle\n5.Exit\nEnter your choice:");
	ch = sc.nextInt();
	switch(ch){
	case 1:System.out.println("Enter the length and breadth:");
	l = sc.nextDouble();
	b = sc.nextDouble();
	ob.Rectangle(l, b);
	break;
	case 2:System.out.println("Enter the breadth and height:");
	l = sc.nextDouble();
	b = sc.nextDouble();
	ob.Triangle(l, b);
	break;
	case 3:System.out.println("Enter the side:");
	l = sc.nextDouble();
	ob.Square(l);
	break;
	case 4:System.out.println("Enter the radius:");
	l = sc.nextDouble();
	ob.Circle(l);
	break;
	case 5:System.exit(0);
	break;
	default:System.out.println("Invalid choice");
	}
	}while(true);
	}
	}

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
1.Rectangle
2.Triangle
3.Square
4.Circle
5.Exit
Enter your choice:
3
Enter the side:
5
Area:25.0

1.Rectangle
2.Triangle
3.Square
4.Circle
5.Exit
Enter your choice:
5
BUILD SUCCESSFUL (total time: 14 seconds)
```

PROGRAM NO-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: To create a package named arithmetic, create a folder of the same name in the directory. Here inside that we have another module named operation
Step 3: Inside arithmetic package, create modules to perform addition, subtraction, multiplication and division of 2 numbers.
Step 4: Outside the folder, write another program that access the above module and print the output.
Step 5:Stop

CODE:

Arithmetic.operations.java

```
package arithmetic;

interface basic
{
   void addition(double a,double b);
   void subtraction(double a,double b);
   void multiplication(double a,double b);
   void division(double a,double b);
}


public class operations implements basic
{
   @Override
   public void addition(double a, double b) {
      System.out.println(a+" + "+b+" = "+(a+b));
   }

   @Override
   public void subtraction(double a, double b) {
      System.out.println(a+" - "+b+" = "+(a-b));
   }
```

	@Override
	public void multiplication(double a, double b) {
	System.out.println(a+" x "+b+" = "+(a*b));
	}
	
	@Override
	public void division(double a, double b) {
	System.out.println(a+" / "+b+" = "+(a/b));
	}
	
	}

CO4Q2.java	package c04q2;
	//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
	import arithmetic.operations;
	import java.util.Scanner;
	public class C04Q2
	{
	public static void main(String[] args)
	{
	double n1,n2;
	int ch;
	operations ob = new operations();
	Scanner sc = new Scanner(System.in);
	System.out.println("Enter the 2 numbers:");
	n1 = sc.nextDouble();
	n2 = sc.nextDouble();
	
	System.out.println("\n1.Addition\n2.Subtraction\n3.Multiplication\n4.Division\nEnter the choice:");
	ch = sc.nextInt();
	switch(ch)
	{
	case 1:ob.addition(n1,n2);
	break;
	case 2:ob.subtraction(n1,n2);
	break;
	case 3:ob.multiplication(n1,n2);
	break;
	case 4:ob.division(n1,n2);
	break;
	default:System.out.println("Invalid choice");

	} } }

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
run:
Enter the 2 numbers:
5
5

1.Addition
2.Subtraction
3.Multiplication
4.Division
Enter the choice:
3
5.0 x 5.0 = 25.0
BUILD SUCCESSFUL (total time: 5 seconds)
```

PROGRAM NO-3

AIM:

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

ALGORITHM:

Step 1: Start the program.

Step 2: Create a User-defined exception named authException.

Step 3: Check for the validity of the username and the password against the required parameters. If any parameter is not met, then throw the authentication failed exception.

Step 6: To authenticate the credentials provided, check if the passwords and usernames given and entered are matching. If they match, display login successful, otherwise throw an exception.

Step 7: Stop the program.

CODE:

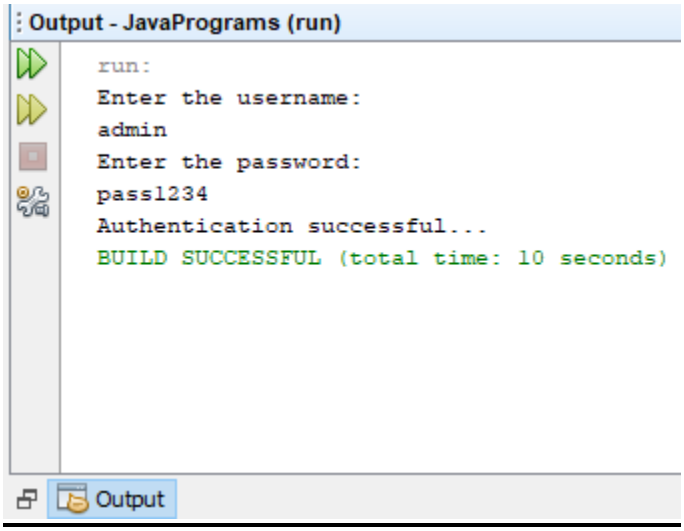
CO4Q3.java	package javaprograms;
	//Write a user defined exception class to authenticate the user name and password
	
	import java.util.Scanner;
	
	class authException extends Exception
	{
	
	public authException(String s) {
	super(s);
	}
	
	}
	public class CO4Q3
	{
	public static void main(String[] args) {

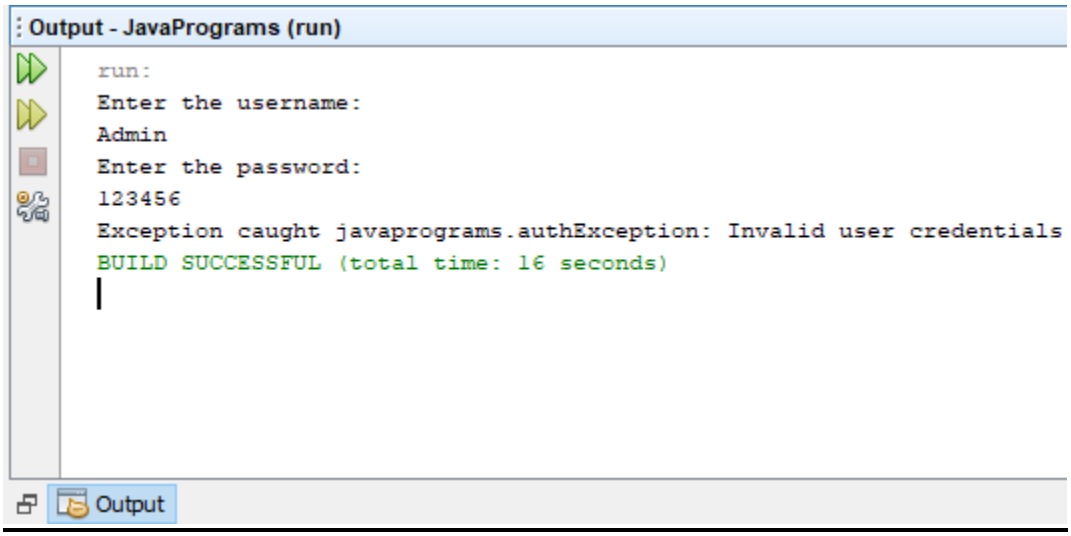
```
        String username = "admin";
        String passcode = "pass1234";
        String user_name,password;
        Scanner sc = new Scanner(System.in);
        try
        {
           System.out.println("Enter the username:");
           user_name = sc.nextLine();
//           sc.nextLine();
           System.out.println("Enter the password:");
           password = sc.nextLine();
           if(username.equals(user_name) && passcode.equals(password))
           {
              System.out.println("Authentication successful...");
           }
           else
              throw new authException("Invalid user credentials");

        }
        catch(authException e)
        {
           System.out.println("Exception caught "+e);
        }
     }

}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:


Output - JavaPrograms (run)
run:
Enter the username:
admin
Enter the password:
pass1234
Authentication successful...
BUILD SUCCESSFUL (total time: 10 seconds)
Output


Output - JavaPrograms (run)
run:
Enter the username:
Admin
Enter the password:
123456
Exception caught javaprograms.authException: Invalid user credentials
BUILD SUCCESSFUL (total time: 16 seconds)
Output

PROGRAM NO-4

AIM:

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

ALGORITHM:

Step 1: Start the program.
Step 2: Define a class named '*NegativeIntegerException*' which throws a negative number exception.
Step 3: Get user inputs for integers at run time and throw an exception if the entered number is negative.
Step 4: Else, proceed to calculate the average and display the result.
Step 5: Stop the program.

CODE:

CO4Q4.java

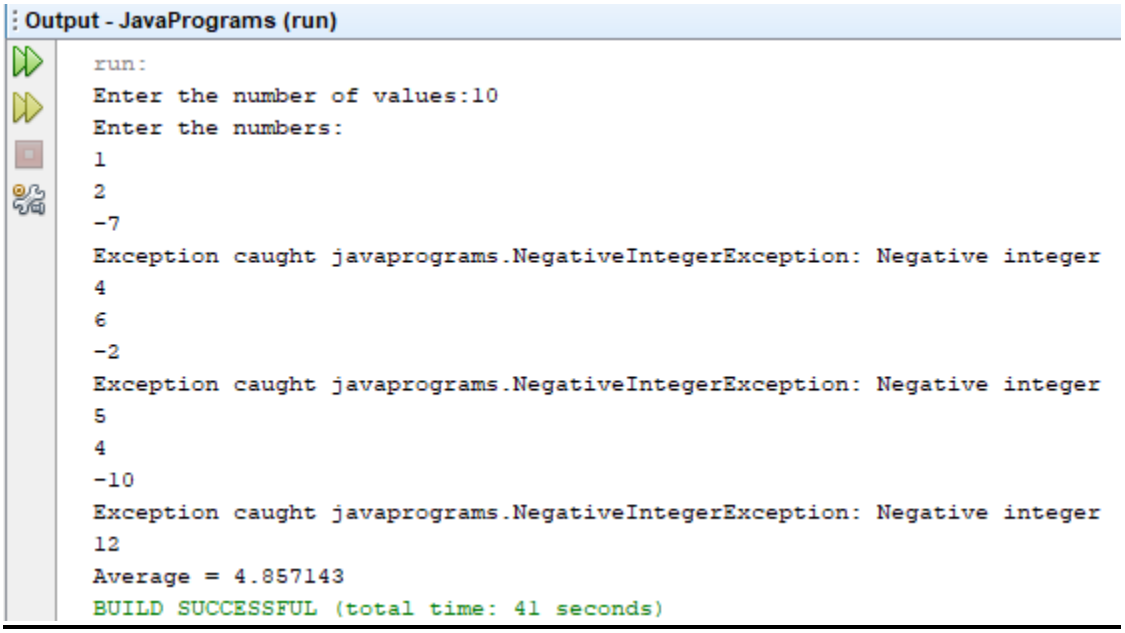
```
package javaprograms;
//Find the average of N positive integers, raising a user defined exception for each negative
//input

import java.util.Scanner;

class NegativeIntegerException extends Exception
{
   public NegativeIntegerException(String s)
   {
      super(s);
   }
}
public class CO4Q4 {
   public static void sample()
   {
      try {
         int n,count=0;
         float num[];
         float total=0;
         Scanner sc = new Scanner(System.in);
         System.out.print("Enter the number of values:");
```

```
            n = sc.nextInt();
            num = new float[n];
            System.out.println("Enter the numbers:");
            for(int i=0;i<n;i++)
            {
               num[i] = sc.nextInt();
               try{
               if(num[i]<0)
               {
                  throw new NegativeIntegerException("Negative integer");
               }
               else
               {
                  total += num[i];
                  count++;
               }
               }catch(NegativeIntegerException e)
               {
                  System.out.println("Exception caught "+e);
               }
            }
            System.out.println("Average = "+(total/count));
         } catch (Exception e) {
            System.out.println("Exception caught "+e);
         }
      }
      public static void main(String[] args) {
         try {
            sample();
         } catch (Exception e) {
         }
      }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:


Output - JavaPrograms (run)
run:
Enter the number of values:10
Enter the numbers:
1
2
-7
Exception caught javaprograms.NegativeIntegerException: Negative integer
4
6
-2
Exception caught javaprograms.NegativeIntegerException: Negative integer
5
4
-10
Exception caught javaprograms.NegativeIntegerException: Negative integer
12
Average = 4.857143
BUILD SUCCESSFUL (total time: 41 seconds)

PROGRAM NO-5

AIM:

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

ALGORITHM:

Step 1: Start the program.

Step 2: Define classes named '*MultiplicationTable*' and '*PrimeNumbers*' that extends *Thread class* and contain methods to compute the multiplication table of 5 and to generate first N prime prime numbers respectively.

Step 3: Define a main method to create objects for the classes and invoke the associated methods.

Step 4: Stop the program.

CODE:

CO4Q5.java	//Define 2 classes; one for generating multiplication table of 5 and other for displaying first
	//N prime numbers. Implement using threads. (Thread class)
	package javaprograms;
	
	import java.util.Scanner;
	import java.util.logging.Level;
	import java.util.logging.Logger;
	
	class MultiplicationTable extends Thread
	{
	public void run()
	{
	System.out.println("Thread is running");
	for(int i=1;i<=10;i++)
	{

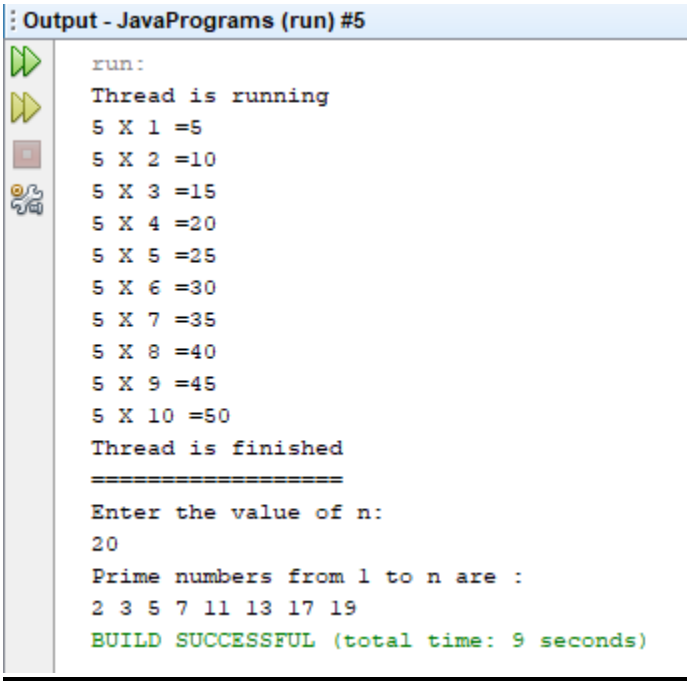
```
            try
            {
               System.out.println("5 X "+i+" ="+(5*i));
               Thread.sleep(200);
            } catch (InterruptedException ex) {

            }
        }
        System.out.println("Thread is finished");
        System.out.println("==================");
    }
}
class PrimeNumbers extends Thread
{
    public void run()
    {
      Scanner sc = new Scanner(System.in);
      int i =0;
      int num =0;
      String  primeNumbers = "";
      System.out.println("Enter the value of n:");
      int n = sc.nextInt();
      for (i = 1; i <= n; i++)
      {
        int counter=0;
        for(num =i; num>=1; num--)
        {
             if(i%num==0)
             {
                  counter = counter + 1;
             }
           }
           if (counter ==2)
           {
             primeNumbers = primeNumbers + i + " ";
           }
      }
      System.out.println("Prime numbers from 1 to n are :");
      System.out.println(primeNumbers);
    }
}
public class CO4Q5 {
    public static void main(String[] args) throws InterruptedException {
        MultiplicationTable MTOb = new MultiplicationTable();
        MTOb.start();
        Thread.sleep(5000);
```

```
        PrimeNumbers PNob = new PrimeNumbers();
        PNob.start();
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:



PROGRAM NO-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 the program.

Step.2: Define classes named '*Fibonacci*' and '*Even*' that implements *Runnable interface* and contain methods to generate Fibonacci series and to display even numbers in a given range respectively.

Step.3: Define a main method to create objects for the classes and invoke the associated methods.

Step.4: Stop the program.

CODE:

CO4Q6.java

```
//Define 2 classes; one for generating Fibonacci numbers and other for displaying even
//numbers in a given range. Implement using threads. (Runnable Interface)
package javaprograms;

import java.util.Scanner;

class Fibonacci implements Runnable
{
   int n,first,second,t;
   String str;

   public Fibonacci(int num)
   {
      n = num;
      first = 0;
      second = 1;
   }

   @Override
   public void run()
```

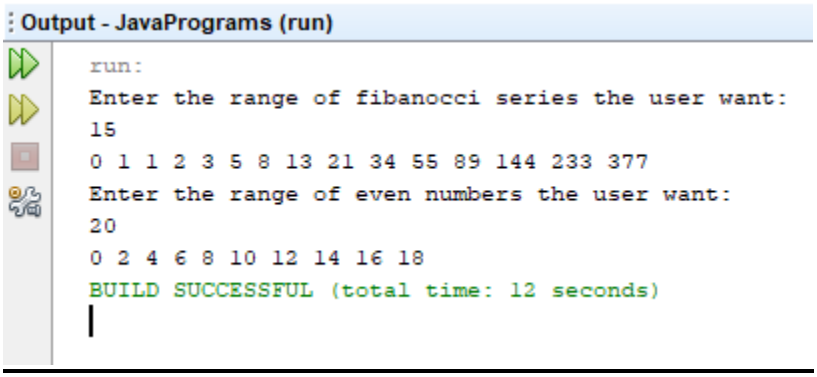
```
    {
        str = first+" "+second;
        for(int i=0;i<=n-3;i++)
        {
            t = first + second;
            first = second;
            second = t;
            str += " "+t;
        }
        System.out.println(str);
    }

}
class Even implements Runnable
{
    int n;
    String str;
    public Even(int n)
    {
        this.n = n;
        str = "";
    }
    @Override
    public void run()
    {
        for(int i=0;i<n;i=i+2)
            if(i%2==0)
            {
                str+=i+" ";
            }
        System.out.println(str);
    }

}
public class CO4Q6 {
    public static void main(String[] args) throws InterruptedException {
        int n1,n2;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the range of fibanocci series the user
want:");
        n1 = sc.nextInt();
        Fibonacci fibob = new Fibonacci(n1);
        Thread th = new Thread(fibob);
        th.start();
        Thread.sleep(400);
        System.out.println("Enter the range of even numbers the user
```

```
want:");
        n2 = sc.nextInt();
        Even evob = new Even(n2);
        Thread th2 = new Thread(evob);
        th2.start();
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:



PROGRAM NO-7

AIM:

Producer/Consumer using ITC

ALGORITHM:

Step 1: Start the program.
Step 2: Define a class that contains two threads that will simulate producer and consumer.
Step 3: Define a class ‘Producer’ and ‘Consumer’ which will contain a LinkedList of integers.
Step 4: Define a method produce() that produces items till its capacity is reached and notify the consumer thread to start consuming.
Step 5: Define a method consume() that consumes items till the list is empty and notify the producer thread to start producing.
Step 6: Stop the program.

CODE:

CO4Q7.java	package javaprograms;
	//Producer/Consumer using ITC
	
	import java.util.ArrayList;
	import java.util.List;
	
	class Producer implements Runnable
	{
	List<Integer> flist;
	int max_size = 5;
	int i=0;
	Producer(List<Integer> flist)
	{
	this.flist = flist;
	}
	@Override

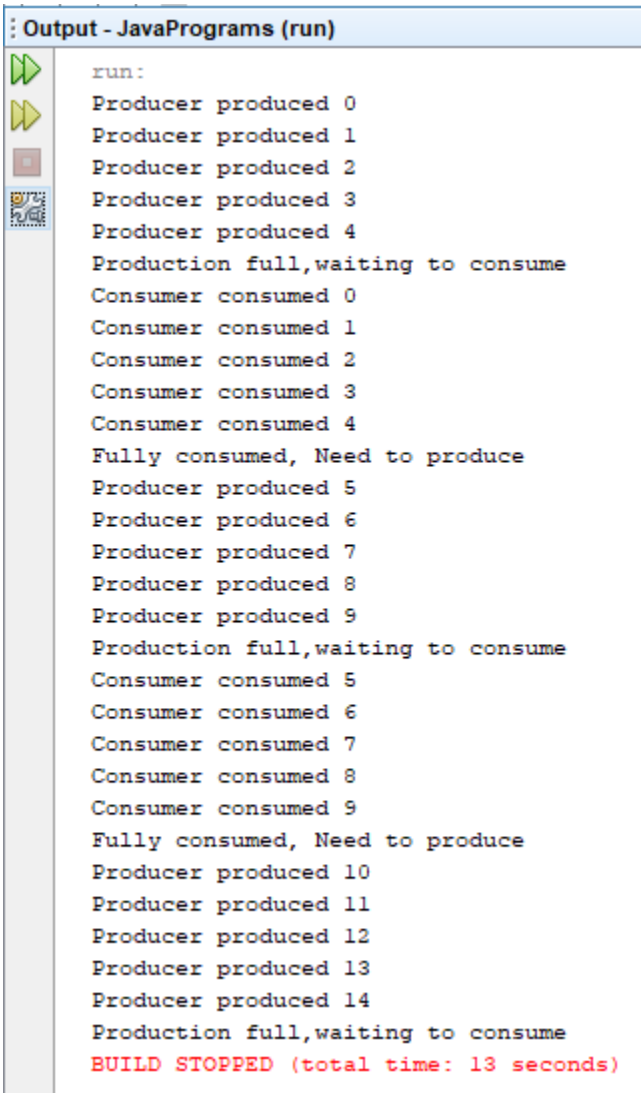
```
    public void run()
    {
        while(true)
        {
            try
            {
                produce(i++);
            } catch (Exception e)
            {
                System.out.println("Intteruption "+e);
            }
        }
    }
    public void produce(int i) throws InterruptedException
    {
        synchronized (flist)
        {
            while(flist.size()==max_size)
            {
                System.out.println("Production full,waiting to consume");
                flist.wait();
            }
        }
        synchronized(flist)
        {
            System.out.println("Producer produced "+i);
            flist.add(i);
            flist.notify();
        }
    }

}
class Consumer implements Runnable
{
    List<Integer> flist;
    Consumer(List<Integer> flist)
    {
        this.flist = flist;
    }

    @Override
    public void run()
    {
        while(true)
        {
            try
```

```
            {
                consume();
            } catch (Exception e)
            {
                System.out.println("Exception "+e);
            }
        }
    }
    public void consume() throws InterruptedException
    {
        synchronized (flist)
        {
            while(flist.isEmpty())
            {
                System.out.println("Fully consumed, Need to produce");
                flist.notify();
                Thread.sleep(500);
                flist.wait();
            }
        }
        synchronized(flist)
        {
            Thread.sleep(1000);
            System.out.println("Consumer consumed "+flist.remove(0));
        }
    }

}
public class CO4Q7 {
    public static void main(String[] args)
    {
        List<Integer> flist = new ArrayList<Integer>();
        Thread th1 = new Thread(new Producer(flist));
        Thread th2 = new Thread(new Consumer(flist));
        th1.start();
        th2.start();
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:


Output - JavaPrograms (run)
run:
Producer produced 0
Producer produced 1
Producer produced 2
Producer produced 3
Producer produced 4
Production full,waiting to consume
Consumer consumed 0
Consumer consumed 1
Consumer consumed 2
Consumer consumed 3
Consumer consumed 4
Fully consumed, Need to produce
Producer produced 5
Producer produced 6
Producer produced 7
Producer produced 8
Producer produced 9
Production full,waiting to consume
Consumer consumed 5
Consumer consumed 6
Consumer consumed 7
Consumer consumed 8
Consumer consumed 9
Fully consumed, Need to produce
Producer produced 10
Producer produced 11
Producer produced 12
Producer produced 13
Producer produced 14
Production full,waiting to consume
BUILD STOPPED (total time: 13 seconds)

PROGRAM NO-8

AIM:

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

ALGORITHM:

Step 1: Start the program.

Step 2: Define a class '*StackElement*' that contains methods to push, pop and display the stack elements.

Step 4: Display the results.

Step 5: Stop the program.

CODE:

CO4Q8.java

```
class StackElement<T>
{
   T value;
   StackElement<T> next;
   public StackElement(T value,StackElement<T> next)
   {
      this.value = value;
      this.next = next;
   }
   public StackElement<T> getNext()
   {
      return next;
   }
   public T getValue()
   {
      return value;
   }

}
public class CO4Q8 <T>
{
   int size;
```

```
    StackElement<T> top;
    public CO4Q8()
    {
        size = 0;
        top = null;
    }
    public void push(T newValue)
    {
        StackElement<T> newElement = new
StackElement<T>(newValue,top);
        top = newElement;
        size++;
    }
    public T pop()
    {
        StackElement<T> oldTop = top;
        if(size==0)
        {
            return null;
        }
        top = top.getNext();
        size--;
        return oldTop.getValue();
    }

    public static void main(String[] args)
    {
        CO4Q8 <String> strStack = new CO4Q8<String>();
        strStack.push("Apple");
        strStack.push("Mango");
        strStack.push("Watermelon");
        strStack.push("Cherry");
        System.out.println(strStack.pop());
        System.out.println(strStack.pop());
        System.out.println(strStack.pop());
        System.out.println(strStack.pop());
        System.out.println(strStack.pop());
    }
}
```

PROGRAM NO-9

AIM:

Using generic method, perform Bubble sort.

ALGORITHM:

Step 1: Start the program.

Step 2: Define a class using generics

Step 3: Define a function '*bubblesort*'.

Step 6: Read the numbers and perform bubblesort.

Step 7: Stop the program.

CODE:

CO4Q9.java

```
import java.util.*;
public class CO4Q9<T extends Comparable<? super T>>
{
   T[] array;
   CO4Q9(T[] array)
   {
      this.array = array;
   }
   public T[] bubbleSort()
   {
      for(int i = array.length; i>1; i--)
      {
         for(int j=0; j<i-1; j++)
         {
            if(array[j].compareTo(array[j+1])>0)
            {
               swap(j,array);
            }
         }
      }
      return array;
```

```
    }
    public void swap(int index, T[] arr)
    {
        T temp = arr[index];
        arr[index] = arr[index+1];
        arr[index] = temp;
    }
    public static void main(String[] args)
    {
        Integer[] intArr = {31,2,53,4,25};
        CO4Q9<Integer> BubbleSort = new CO4Q9<Integer>(intArr);
        Integer[] SortedArray = BubbleSort.bubbleSort();
        System.out.println("Sorted Array:- "+Arrays.toString(SortedArray));
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
run:
Sorted Array:- [31, 2, 53, 4, 25]
BUILD SUCCESSFUL (total time: 0 seconds)
```

PROGRAM NO-10

AIM:

To maintain a list of Strings using ArrayList from collection framework, perform built-in operations.

ALGORITHM:

Step 1: Start the program.

Step 2: Define a class with a main() to implement the concept.

Step 3: Declare an ArrayList of strings named '*fruits*' and start adding elements into it.

Step 4: Execute different ArrayList methods and display the results.

Step 5: Stop the program.

CODE:

CO4Q10.java

```
import java.util.*;

public class CO4Q10 {
   public static void main(String[] args) {
      ArrayList<String> fruits = new ArrayList<String>();
      fruits.add("Apple");
      fruits.add("Strawberry");
         fruits.add("Mango");
         fruits.add("Pineapple");
         fruits.add("Banana");
         fruits.add(4, "Grapes");
      System.out.println("Fruits-");
      for(String str: fruits)
         System.out.println(str+" ");
      fruits.remove("Banana");
      fruits.remove(1);
      System.out.println("\t**********");
      System.out.println("Fruits after items removed");
      for(String str: fruits)
         System.out.println(str+" ");
```

```
        Collections.sort(fruits);
        System.out.println("\t**********");
        System.out.println("Sorted-");
        for(String str: fruits)
           System.out.println(str+" ");
        System.out.println("\t**********");
        System.out.println("Total fruits-"+fruits.size());
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
run:
Fruits-
Apple
Strawberry
Mango
Pineapple
Grapes
Banana
        **********
Fruits after items removed
Apple
Mango
Pineapple
Grapes
        **********
Sorted-
Apple
Grapes
Mango
Pineapple
        **********
Total fruits-4
BUILD SUCCESSFUL (total time: 0 seconds)
```

PROGRAM NO-11

AIM:

To write a program to remove all the elements from a linked list.

ALGORITHM:

Step 1: Start the program.

Step 2: Define a class with a main() to implement the concept.

Step 3: Create a LinkedList.

Step 4: Using add() to add to LinkedList

Step 5: Using clear() to remove all from the LinkedList

Step 6: Stop the program.

CODE:

CO4Q11.java

```
import java.util.LinkedList;
import java.util.Scanner;


public class CO4Q11
{

    public static void main(String[] args) {
        int n;
        String data;
        LinkedList<String> ll = new LinkedList<String>();
        System.out.println("Enter the number of data");
        Scanner sc = new Scanner(System.in);
        n = sc.nextInt();
        System.out.println("Enter the data");
        sc.nextLine();
        for(int i=0;i<n;i++)
        {
            data = sc.nextLine();
            ll.add(data);
        }
```

```
        System.out.println("LinkedList: "+ll);
        System.out.println("Removing all the elements....");
        ll.clear();
        System.out.println(ll);
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
Enter the number of data
5
Enter the data
Apple
Mango
Orange
Grape
Kiwi
LinkedList: [Apple, Mango, Orange, Grape, Kiwi]
Removing all the elements....
[]
BUILD SUCCESSFUL (total time: 45 seconds)
```

PROGRAM NO-12

AIM:

To write a program to remove an object from the Stack when the position is passed as parameter.

ALGORITHM:

Step 1: Start the program.

Step 2: Define a class with a main () to implement the concept.

Step 3: Declare a Stack of Strings and start adding elements into it using add() method.

Step 5: Using remove(index) method, remove the element and display the updated stack.

Step 6: Stop the program.

CODE:

CO4Q12.java

```
import java.util.Scanner;
import java.util.Stack;

public class CO4Q12 {
   public static void main(String[] args) {
      int n;
      String str;
      Stack<String> s = new Stack<String>();
      System.out.println("Enter the number of elements:");
      Scanner sc = new Scanner(System.in);
      n = sc.nextInt();
      sc.nextLine();
      System.out.println("Enter the elements:");
      for(int i=0;i<n;i++)
      {
         str = sc.nextLine();
         s.add(str);
      }
      System.out.println("\nStack elements:"+s);
      System.out.println("\nTop element:"+s.peek());
      System.out.println("Popped element:"+s.pop());
```

```
        System.out.println("Stack elements after popped:"+s);
        System.out.println("\nRemove Element at position 1:"+s.remove(0));
        System.out.println("Stack elements after removed:"+s);
        System.out.println("\nRemove Luke Skywalker:");
        s.remove("Luke Skywalker");
        System.out.println("Stack elements after removing Luke
Skywalker:"+s);
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
run:
Enter the number of elements:
7
Enter the elements:
Han Solo
Leia
Luke Skywalker
Chewbacca
Yoda
R2-D2
C-3PO

Stack elements:[Han Solo, Leia, Luke Skywalker, Chewbacca, Yoda, R2-D2, C-3PO]

Top element:C-3PO
Popped element:C-3PO
Stack elements after popped:[Han Solo, Leia, Luke Skywalker, Chewbacca, Yoda, R2-D2]

Remove Element at position 1:Han Solo
Stack elements after removed:[Leia, Luke Skywalker, Chewbacca, Yoda, R2-D2]

Remove Luke Skywalker:
Stack elements after removing Luke Skywalker:[Leia, Chewbacca, Yoda, R2-D2]
BUILD SUCCESSFUL (total time: 43 seconds)
```

PROGRAM NO-13

AIM:

To write a program to demonstrate the creation of queue object using the PriorityQueue class.

ALGORITHM:

Step 1: Start the program.

Step 2: Define a class with a main () to implement the concept.

Step 3: Declare a *PriorityQueue* of Strings and start adding elements into it using *add()* method.

Step 4: Iterate and display the queue elements.

Step 5: Stop the program.

CODE:

CO4Q13.java

```
import java.util.Iterator;
import java.util.PriorityQueue;
import java.util.Scanner;


public class CO4Q13 {
    public static void main(String[] args) {
        int n;
        String str;
        PriorityQueue<String> pq = new PriorityQueue<String>();
        System.out.println("Enter the no. of data:");
        Scanner sc = new Scanner(System.in);
        n = sc.nextInt();
        sc.nextLine();
        System.out.println("Enter the data:");
        for(int i=0;i<n;i++)
        {
            str = sc.nextLine();
            pq.add(str);
        }
        Iterator itr = pq.iterator();
```

```
        System.out.println("\nPriority Queue\n");
        while(itr.hasNext())
            System.out.println(itr.next()+" ");
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
run:
Enter the no. of data:
10
Enter the data:
Inception
Interstellar
Finding Neverland
Forrest Gump
The Professor
Edge of Tomorrow
Sweeney Todd
The Lone Ranger
Edward Scissorhands
Ready Player One

Priority Queue

Edge of Tomorrow
Edward Scissorhands
Finding Neverland
Forrest Gump
Ready Player One
Inception
Sweeney Todd
The Lone Ranger
Interstellar
The Professor
BUILD SUCCESSFUL (total time: 1 minute 9 seconds)
```

PROGRAM NO-14

AIM:

To write a program to demonstrate the addition and deletion of elements in deque.

ALGORITHM:

Step 1: Start the program.

Step 2: Define a class with a main () to implement the concept.

Step 3: Declare a *Dequeue* of Strings and perform the following methods:

- addFirst() - inserts the element passed in the parameter to the front of the *Deque* if there is space.
- addLast() - inserts the element passed in the parameter to the end of the *Deque* if there is space.
- removeFirst() - removes the first element of Deque.
- removeLast()- removes the last element of Deque.

Step 4: Display the results.

Step 5: Stop the program.

CODE:

CO4Q14.java	import java.util.Deque;
	import java.util.LinkedList;
	import java.util.Scanner;
	
	public class CO4Q14 {
	public static void main(String[] args) {
	int ch;
	String data;
	Deque<String> dq = new LinkedList<String>();
	Scanner sc = new Scanner(System.in);
	do

```
        {
            System.out.println("\nChoose a number...");
            System.out.println("1.Insert the element at first");
            System.out.println("2.Insert the element at last");
            System.out.println("3.Delete the element at first");
            System.out.println("4.Delete the element at last");
            System.out.println("5.Display");
            System.out.println("6.Exit");
            System.out.println("\nEnter the choice:");
            ch = sc.nextInt();
            sc.nextLine();
            switch(ch)
            {
                case 1: System.out.println("Enter the element to be inserted at
first:");
                        data = sc.nextLine();
                        dq.addFirst(data);
                        break;
                case 2: System.out.println("Enter the element to be inserted at
last:");
                        data = sc.nextLine();
                        dq.addLast(data);
                        break;
                case 3: System.out.println("Element deleted from the first
position");
                        dq.removeFirst();
                        break;
                case 4: System.out.println("Element deleted from the last
position");
                        dq.removeLast();
                        break;
                case 5: System.out.println("Elements:");
                        System.out.println(dq);
                        break;
                case 6: System.exit(0);
                        break;
                default:System.out.println("Invalid choice...");
            }
        }while(true);
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
run:

Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
1
Enter the element to be inserted at first:
Iron Man

Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
2
Enter the element to be inserted at last:
Captain America

Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
1
Enter the element to be inserted at first:
Black Widow

Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
2
Enter the element to be inserted at last:
Thor
```

```
Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
1
Enter the element to be inserted at first:
Hawkeye

Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
2
Enter the element to be inserted at last:
Hulk

Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
5
Elements:
[Hawkeye, Black Widow, Iron Man, Captain America, Thor, Hulk]

Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
3
Element deleted from the first position
```

```
Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
5
Elements:
[Black Widow, Iron Man, Captain America, Thor, Hulk]

Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
4
Element deleted from the last position

Choose a number...
1.Insert the element at first
2.Insert the element at last
3.Delete the element at first
4.Delete the element at last
5.Display
6.Exit

Enter the choice:
5
Elements:
[Black Widow, Iron Man, Captain America, Thor]
```

PROGRAM NO-15

AIM:

To write a program to demonstrate the creation of Set object using the LinkedHashset class.

ALGORITHM:

Step 1: Start the program.

Step 2: Define a class with a main () to implement the concept.

Step 3: Declare an *LinkedHashSet* of strings.

Step 4: Start adding elements into it using add() method.

Step 5: Execute some LinkedHashSet methods and display the results.

Step 6: Stop the program.

CODE:

CO4Q15.java

```
import java.util.Iterator;
import java.util.LinkedHashSet;
import java.util.Scanner;

public class CO4Q15 {
   public static void main(String[] args) {
      int n;
      String str;
      LinkedHashSet<String> lsh = new LinkedHashSet<String>();
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the no. of values");
      n = sc.nextInt();
      sc.nextLine();
      System.out.println("Enter the values");
      for(int i=0;i<n;i++)
      {
         str = sc.nextLine();
         lsh.add(str);
      }
      System.out.println("\nOriginal LinkedHashSet:"+lsh);
      System.out.println("Removed 'Rachel' from
```

```
LinkedHashSet:"+lsh.remove("Rachel"));
        System.out.println("Size of LinkedHashSet:"+lsh.size());
        System.out.println("Checking if 'Joey' is
present:"+lsh.contains("Joey"));
        System.out.println("Final LinkedHashSet:"+lsh);
        System.out.println("\nIterating...");
        Iterator itr = lsh.iterator();
        while(itr.hasNext())
           System.out.println(itr.next());

    }

}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
run:
Enter the no. of values
6
Enter the values
Chandler
Joey
Ross
Rachel
Phoebe
Monica

Original LinkedHashSet:[Chandler, Joey, Ross, Rachel, Phoebe, Monica]
Removed 'Rachel' from LinkedHashSet:true
Size of LinkedHashSet:5
Checking if 'Joey' is present:true
Final LinkedHashSet:[Chandler, Joey, Ross, Phoebe, Monica]

Iterating...
Chandler
Joey
Ross
Phoebe
Monica
BUILD SUCCESSFUL (total time: 23 seconds)
```

PROGRAM NO-16

AIM:

To write a Java program to compare two hash sets.

ALGORITHM:

Step 1: Start the program.

Step 2: Define a class with a main () to implement the concept.

Step 3: Declare two HashSets (s1, s2) of strings and start adding elements into them using add() method.

Step 4: Display the elements of both the hashsets.

Step 5: Perform set operations.

- addAll() – gets all the elements
- retainAll() – gets the elements which are common
- removeAll() – gets the difference

Step 6: Display the results.

Step 7: Stop the program.

CODE:

CO4Q16.java	import java.util.*;
	
	public class CO4Q16 {
	public static void union(Set<String> s1,Set<String> s2)
	{
	Set<String> su = new HashSet<>(s1);
	su.addAll(s2);
	System.out.println("Union:"+su);
	}
	public static void intersection(Set<String> s1,Set<String> s2)
	{
	Set<String> su = new HashSet<>(s1);

```
        su.retainAll(s2);
        System.out.println("Intersection:"+su);
    }
    public static void difference(Set<String> s1,Set<String> s2)
    {
        Set<String> su1 = new HashSet<>(s1);
        su1.removeAll(s2);
        Set<String> su2 = new HashSet<>(s2);
        su2.removeAll(s1);
        System.out.println("Difference:First HashSet- "+su1+" and Second
HashSet- "+su2);
    }
    public static void main(String[] args) {
        Set<String> s1 = new HashSet<>();
        s1.add("Tokyo");
        s1.add("Berlin");
        s1.add("Rio");
        Set<String> s2 = new HashSet<>();
        s2.add("Tokyo");
        s2.add("Berlin");
        s2.add("Denver");
        System.out.println("Elements in first HashSet:"+s1);
        System.out.println("Elements in second HashSet:"+s2);
        union(s1,s2);
        intersection(s1,s2);
        difference(s1,s2);
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
run:
Elements in first HashSet:[Rio, Berlin, Tokyo]
Elements in second HashSet:[Berlin, Tokyo, Denver]
Union:[Rio, Tokyo, Berlin, Denver]
Intersection:[Tokyo, Berlin]
Difference:First HashSet- [Rio] and Second HashSet- [Denver]
BUILD SUCCESSFUL (total time: 0 seconds)
```

PROGRAM NO-17

AIM:

To write a program to demonstrate the working of Map interface by adding, changing and removing elements.

ALGORITHM:

Step.1: Start the program.

Step.2: Define a class with a main () to implement the concept.

Step.3: Declare a *HashMap* and insert elements into it using *put()* method.

Step.4: To update any value in hashmap using *hashmap.put()* or *hashmap.replace()*.

Step.5: Using *remove(key)* method, remove elements from hashmap.

Step.6: Display the results.

Step.7: Stop the program.

CODE:

CO4Q17.java	import java.util.HashMap;
	import java.util.Map;
	
	public class CO4Q17 {
	  public static void main(String[] args) {
	  Map<Integer, String> hm = new HashMap<>();
	  hm.put(1,"The Professor(2018)");
	  hm.put(2,"Jojo Rabbit(2019)");
	  hm.put(3,"Just Mercy(2019)");
	  hm.put(4,"Back to the Future(1985)");
	  System.out.println("Map:"+hm);
	  hm.put(4,"Back to the Future 2(1989)");
	  System.out.println("Updated Map:"+hm);
	  hm.remove(3);
	  System.out.println("Final Map:"+hm);
	  }
	}

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
run:
Map:{1=The Professor(2018), 2=Jojo Rabbit(2019), 3=Just Mercy(2019), 4=Back to the Future(1985)}
Updated Map:{1=The Professor(2018), 2=Jojo Rabbit(2019), 3=Just Mercy(2019), 4=Back to the Future 2(1989)}
Final Map:{1=The Professor(2018), 2=Jojo Rabbit(2019), 4=Back to the Future 2(1989)}
BUILD SUCCESSFUL (total time: 0 seconds)
```

PROGRAM NO-18

AIM:

To write a program to Convert HashMap to TreeMap.

ALGORITHM:

Step 1: Start the program.

Step 2: Define a class with a main () to implement the concept.

Step 3: Declare a *HashMap* and insert elements into it using *put()* method.

Step 4: Convert the above HashMap to TreeMap:

Map<i,s> treeMap = new TreeMap<>();

treeMap.putAll(hashMap);

Step 5: Display the resultant treeMap.

Step 6: Stop the program.

CODE:

CO4Q18.java

```
import java.util.*;

public class CO4Q18 {
   public static <i,s> Map<i,s> convert(Map<i,s> hashmap)
   {
      Map<i,s> treemap = new TreeMap<>();
      treemap.putAll(hashmap);
      return treemap;
   }
   public static void main(String[] args) {
      Map<String,Integer> hashmap = new HashMap<>();
      hashmap.put("Johnny Depp",1);
      hashmap.put("Tom Cruise",1);
      hashmap.put("Jackie Chan",1);
      hashmap.put("Keanu Reeves",1);
```

```
        System.out.println("HashMap:"+hashmap);
        Map<String,Integer>treemap = convert(hashmap);
        System.out.println("TreeMap:"+treemap);
    }
}
```

RESULT:

The above program is successfully executed and the output is obtained.

OUTPUT:

```
run:
HashMap:{Tom Cruise=1, Johnny Depp=1, Keanu Reeves=1, Jackie Chan=1}
TreeMap:{Jackie Chan=1, Johnny Depp=1, Keanu Reeves=1, Tom Cruise=1}
BUILD SUCCESSFUL (total time: 0 seconds)
```