Experiment : 01

Aim :

Define a class ‘product’ with data members pcode, pname and price. Create 3 objects of the class and find the product having the lowest price.

CO 1:

Understand object-oriented concepts and design classes and objects to solve problems

Procedure

import java.util.Scanner;

public class Product {

int pcode;

String pname;

int price;

public void ask() {

Scanner cin = new Scanner(System.in);

System.out.println("Enter product code: ");

pcode = cin.nextInt();

System.out.println("Enter product name: ");

pname = cin.next();

System.out.println("Enter product price: ");

price = cin.nextInt();

}

public void printProductId() {

System.out.println("Product code is: " + pcode);

System.out.println("Product name is: " + pname);

System.out.println("Product price is: " + price);

}

public static void main(String[] args) {

Product p1 = new Product();

Product p2 = new Product();

Product p3 = new Product();

p1.ask();

p2.ask();

p3.ask();

p1.printProductId();

p2.printProductId();

p3.printProductId();

if(p1.price<p2.price && p1.price <p3.price)

{

System.out.println(p1.price+"is lowest price");

}

else if(p2.price<p1.price && p2.price<p3.price)

{

System.out.println(p2.price+"is lowest price");

}

else{

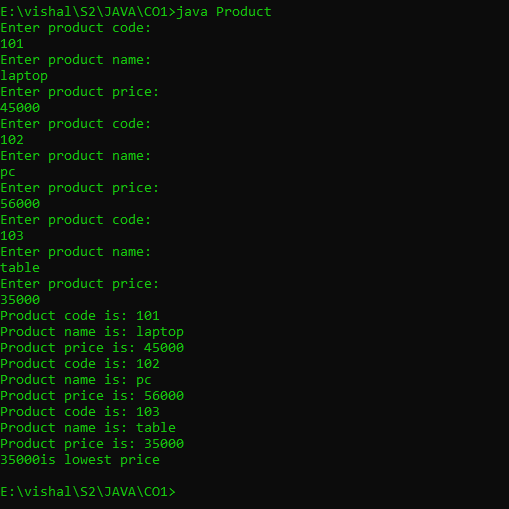
System.out.println(p3.price+"is lowest price");

}

}

}

Output



Experiment : 02

Aim :

Read 2 matrices from the console and perform matrix addition.

CO 1:

Understand object-oriented concepts and design classes and objects to solve problems

Procedure

import java.util.\*;

public class addmatrix{

public static void main(String[] args)

{

int val,n,i,j,r,c;

int arr[][]=new int[50][50];

int arr1[][]=new int[50][50];

Scanner obj=new Scanner(System.in);

System.out.println("enter the size of row :");

r=obj.nextInt();

System.out.println(" enter the size of the column :");

c=obj.nextInt();

System.out.println("enter the elments of first array:");

for(i=0;i<r;i++)

{

for(j=0;j<c;j++)

{

arr[i][j]=obj.nextInt();

}

}

System.out.println("the first array elments are :");

for(i=0;i<r;i++)

{

for(j=0;j<c;j++)

{

System.out.print(arr[i][j]+" ");

}

System.out.println();

}

System.out.println("enter the elments of second array :");

for(i=0;i<r;i++)

{

for(j=0;j<c;j++)

{

arr1[i][j]=obj.nextInt();

}

}

System.out.println("the second array elments are :");

for(i=0;i<r;i++)

{

for(j=0;j<c;j++)

{

System.out.print(arr1[i][j]+" ");

}

System.out.println();

}

System.out.println("the sum of array elments is :");

for(i=0;i<r;i++)

{

for(j=0;j<c;j++)

{

System.out.print(arr[i][j]+arr1[i][j]+" ");

}

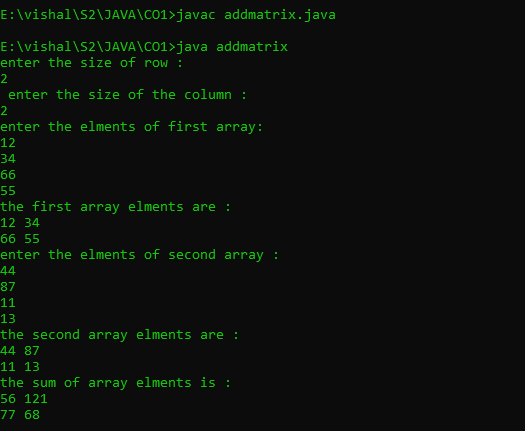
System.out.println();

}

}

}

Output



Experiment : 03

Aim :

Add complex numbers

CO 1:

Understand object-oriented concepts and design classes and objects to solve problems

Procedure

import java.util.\*;

public class complex{

public static void main(String[] args)

{

Scanner obj=new Scanner(System.in);

System.out.println("enter a real number :");

int real=obj.nextInt();

System.out.println("enter imaginary number :");

int img=obj.nextInt();

int comp=real+img;

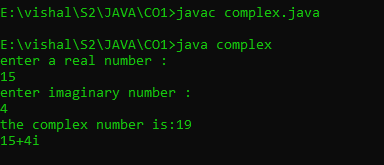
System.out.println("the complex number is:"+comp);

System.out.println(real+"+"+img+ "i " );

}

}

Output



Experiment : 04

Aim :

Read a matrix from the console and check whether it is symmetric or not.

CO 1:

Understand object-oriented concepts and design classes and objects to solve problems

Procedure

import java.util.Scanner;

public class SymtrcMatrx

{

public static void main(String[] args)

{

Scanner obj=new Scanner(System.in);

System.out.println("Enter the row and col of first matrix");

Integer r1=obj.nextInt();

Integer c1=obj.nextInt();

int a[][]=new int[r1][c1];

int c[][]=new int[r1][c1];

int i,j;

System.out.println("enter elements to first array");

for(i=0;i<r1;i++)

{

for(j=0;j<c1;j++)

{

a[i][j]=obj.nextInt();

}

}

System.out.println("Printing 2d first Matrix");

for(i=0;i<r1;i++)

{

for(j=0;j<c1;j++)

{

System.out.print(a[i][j]+" ");

}

System.out.println();

}

int flag=1;

for(i=0;i<r1;i++)

{

for(j=0;j<c1;j++)

{

c[i][j]=a[j][i];

if(c[i][j]!=a[i][j])

{

flag=0;

}

}

}

if(flag==1)

{

System.out.println("The matrix is a Symmetric Matrix");

}

else

{

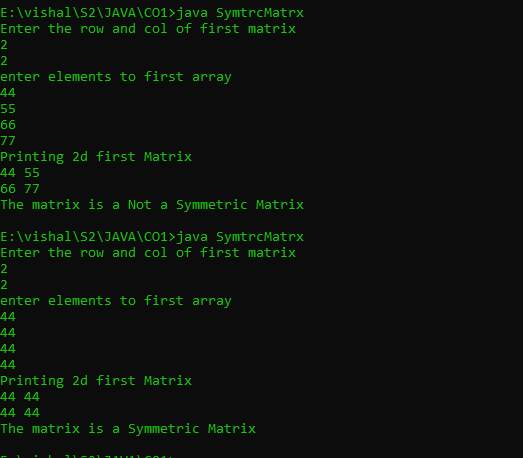
System.out.println("The matrix is a Not a Symmetric Matrix");

}

}

}

Output



Experiment : 05

Aim :

Create CPU with attribute price. Create inner class Processor (no. of cores, manufacturer) and static nested class RAM (memory, manufacturer). Create an object of CPU and print information of Processor and RAM

CO 1:

Understand object-oriented concepts and design classes and objects to solve problems

Procedure

import java.util.\*;

public class CpuPrgrm {

int price;

public class Processor {

int numCores;

String manufacturer;

Processor(int numCores, String manufacturer) {

this.numCores = numCores;

this.manufacturer = manufacturer;

}

void display() {

System.out.println(" ");

System.out.println("::PROCESSOR INFORMATION::");

System.out.println("Number of cores: " + numCores);

System.out.println("Manufacturer: " + manufacturer);

System.out.println(" ");

}

}

static class RAM {

int memory;

String manufacturer;

RAM(int memory, String manufacturer) {

this.memory = memory;

this.manufacturer = manufacturer;

}

void display() {

System.out.println(" ");

System.out.println("::RAM INFORMATION::");

System.out.println("Memory: " + memory);

System.out.println("Manufacturer: " + manufacturer);

System.out.println(" ");

}

}

public static void main(String[] args) {

Scanner Snr = new Scanner(System.in);

CpuPrgrm cpu = new CpuPrgrm();

System.out.print("Enter CPU price: ");

cpu.price = Snr.nextInt();

System.out.print("Enter number of processor cores: ");

int numCores = Snr.nextInt();

System.out.print("Enter processor manufacturer: ");

String processorManufacturer = Snr.next();

CpuPrgrm.Processor processor = cpu.new Processor(numCores, processorManufacturer);

processor.display();

System.out.print("Enter RAM memory: ");

int ramMemory = Snr.nextInt();

System.out.print("Enter RAM manufacturer: ");

String ramManufacturer = Snr.next();

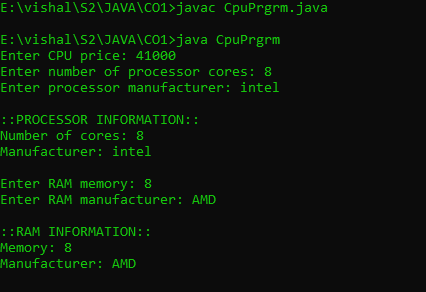
CpuPrgrm.RAM ram = new CpuPrgrm.RAM(ramMemory, ramManufacturer);

ram.display();

}

}

Output



Experiment : 01

Aim :

Program to Sort strings

CO 2:

Implement arrays and strings

Procedure

import java.util.Scanner;

public class CO\_2sortstrings {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter the size of the Array :");

int l = sc.nextInt();

String str[] = new String[l];

int i;

System.out.println("Enter the String Elements");

for (i = 0; i < str.length; i++) {

str[i] = sc.next();

}

String temp;

int j;

for (i = 0; i < str.length; i++) {

for (j = i + 1; j < str.length; j++) {

if (str[i].compareTo(str[j]) > 0) {

temp = str[i];

str[i] = str[j];

str[j] = temp;

}

}

}

System.out.println("The Sorted String order : ");

for (i = 0; i < str.length; i++) {

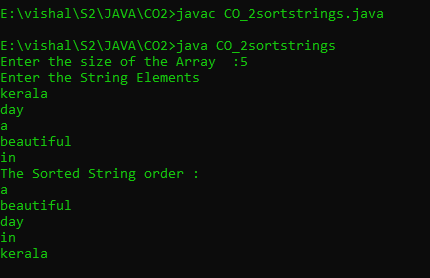
System.out.println(str[i]);

}

}

}

Output



Experiment : 02

Aim :

Search an element in an array.

CO 2:

Implement arrays and strings

Procedure

import java.util.Scanner;

public class Search\_Array

{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter the limit");

Integer size=sc.nextInt();

int arr[]=new int[size];

int i,flag=0;

System.out.println("Enter the elements");

for(i=0;i<arr.length;i++)

{

arr[i]=sc.nextInt();

}

System.out.println("Enter item to be searched");

Integer item=sc.nextInt();

for(i=0;i<arr.length;i++)

{

if(item==arr[i])

{

System.out.println("element found at loc "+i);

flag=1;

}

}

if(flag==0)

{

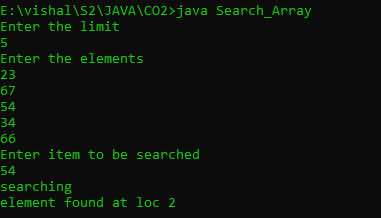
System.out.println("element not found");

}

}

}

Output



Experiment : 03

Aim :

Perform string manipulations

CO 2:

Implement arrays and strings

Procedure

import java.util.\*;

public class string\_manip{

public static void main(String[] args){

Scanner obj=new Scanner(System.in);

System.out.println("enter first string :");

String first=obj.nextLine();

System.out.println("enter second string :");

String second=obj.nextLine();

System.out.println(first.concat(second));

if(first == second)

{

System.out.println("two strings are same");

}

else

{

System.out.println("strings are not same");

}

System.out.println("the length of the first string is :"+first.length());

System.out.println("the length of the first string is :"+second.length());

System.out.println(first.equals(second));

int len1=first.length();

int len2=second.length();

if(len1==len2)

{

System.out.println("the length is same");

}

else

{

System.out.println("lengths are not same");

}

System.out.println(first.toUpperCase());

System.out.println(second.toUpperCase());

System.out.println(first.toLowerCase());

System.out.println(second.toLowerCase());

System.out.println("enter string to locate :");

String loc=obj.nextLine();

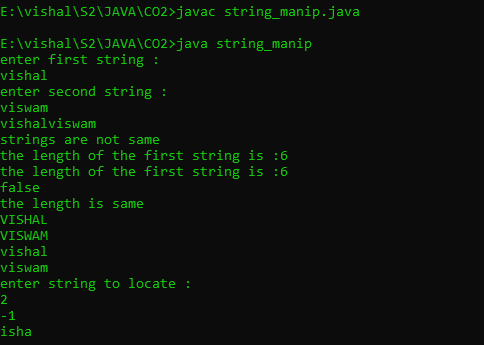
System.out.println(first.indexOf(loc));

System.out.println(first.substring(1,5));

}

}

Output



Experiment : 04

Aim :

Program to create a class for Employee having attributes eNo, eName eSalary. Read n employ information and Search for an employee given eNo, using the concept of Array of Objects.

CO 2:

Implement arrays and strings

Procedure

import java.util.\*;

public class employee\_search{

int eno;

String ename;

int salary;

public void get()

{

Scanner obj=new Scanner(System.in);

System.out.println("enter employee number :");

eno =obj.nextInt();

System.out.println("enter employee name :");

ename =obj.next();

System.out.println("enter employee salary :");

salary =obj.nextInt();

}

public void display()

{

System.out.println("employee number :"+eno);

System.out.println("employee name :"+ename);

System.out.println("employee salary :"+salary);

}

public static void main(String[] args)

{

int flag=0;

Scanner obj=new Scanner(System.in);

System.out.println("enter the size :");

int n=obj.nextInt();

employee\_search e1[]= new employee\_search[n];

for(int i=0;i<n;i++)

{

e1[i]=new employee\_search();

e1[i].get();

}

for(int i=0;i<2;i++)

{

e1[i].display();

}

System.out.println("enter employee number to search for the employee details :");

int item=obj.nextInt();

for(int i=0;i<n;i++)

{

if(e1[i].eno ==item)

{

e1[i].display();

flag++;

break;

}

}

if(flag==0)

{

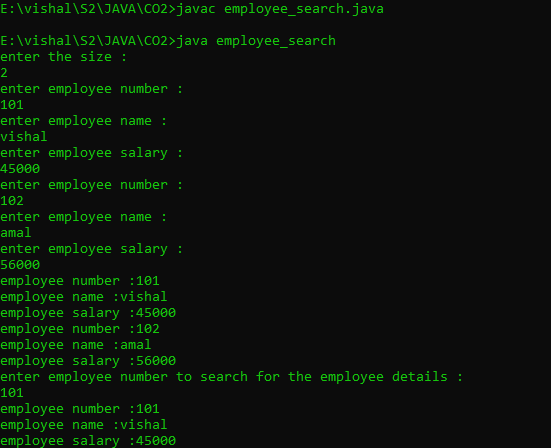
System.out.println("not found");

}

}

}

Output



Experiment : 01

Aim :

Area of different shapes using overloaded functions

CO 3:

Implement object-oriented concepts like inheritance, overloading and interfaces

Procedure

import java.util.\*;

class CO3\_Shape

{

int r,h,a,p1,p2,t2,t3;

double b,l,w,t1;

public void area(double r)

{

double c=3.14\*(r\*r);

System.out.println("Area of circle: "+c);

}

public void area(double b,int h)

{

double t=(b\*h)/2;

System.out.println("Area of triangle: "+t);

}

public void area(int a)

{

int s=a\*a;

System.out.println("Area of square: "+s);

}

public void area(double l,double w)

{

double r=l\*w;

System.out.println("Area of rectangle: "+r);

}

public void area(int p1,int p2)

{

int p=p1\*p2;

System.out.println("Area of parallelogram: "+p);

}

public void area(float e1,float e2)

{

double c=3.14\*(e1\*e2);

System.out.println("Area of ellipse: "+c);

}

public static void main(String[] args)

{ int ch;

Scanner s=new Scanner(System.in);

CO3\_Shape sh =new CO3\_Shape();

System.out.println("Area of different shapes"+"\n"+" 1.Circle"+"\n"+" 2.Triangle"+"\n"+" 3.Square "+"\n"+" 4.Rectangle "+"\n"+" 5.Parallelogram "+"\n"+" 6.Ellipse");

do

{

System.out.println("Enter choice: ");

ch=s.nextInt();

switch(ch)

{

case 1:

{

System.out.println("Enter the radius: ");

double r=s.nextDouble();

sh.area(r);

}

break;

case 2:

{

System.out.println("Enter the breadth: ");

double b=s.nextDouble();

System.out.println("Enter the height: ");

int h=s.nextInt();

sh.area(b,h);

}

break;

case 3:

{

System.out.println("Enter the length: ");

int a=s.nextInt();

sh.area(a);

}

break;

case 4:

{

System.out.println("Enter the length: ");

double l=s.nextDouble();

System.out.println("Enter the breadth: ");

double w=s.nextDouble();

sh.area(l,w);

}

break;

case 5:

{

System.out.println("Enter the base: ");

int p1=s.nextInt();

System.out.println("Enter the vertical height: ");

int p2=s.nextInt();

sh.area(p1,p2);

}

break;

case 6:

{

System.out.println("Enter the minor axis: ");

float e1=s.nextFloat();

System.out.println("Enter the major axis: ");

float e2=s.nextFloat();

sh.area(e1,e2);

}

break;

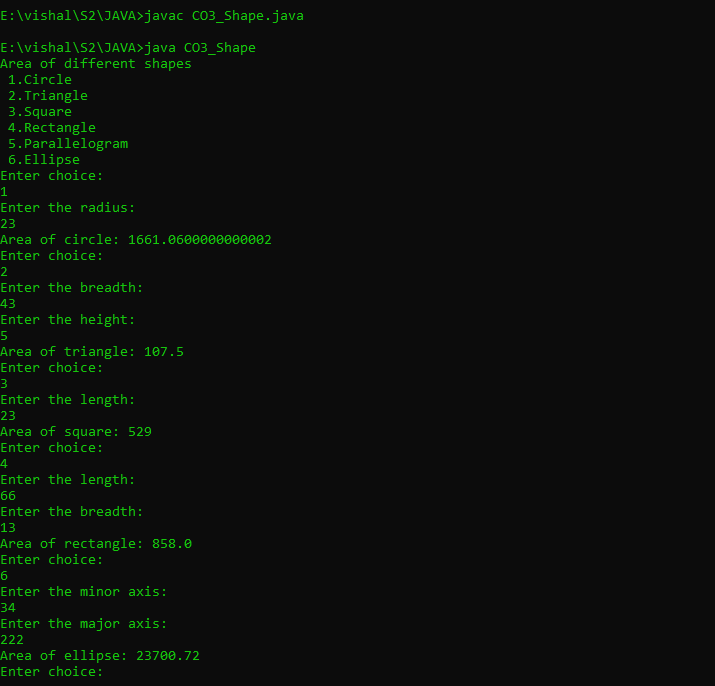
}

}

while(ch!=7);

}}

Output



Experiment : 02

Aim :

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.

CO 3:

Implement object-oriented concepts like inheritance, overloading and interfaces

Procedure

import java.util.\*;

class Employee

{

int Empid,Salary;

String Name,Address;

Scanner sc=new Scanner(System.in);

Employee()

{

System.out.println("Enter the Employee ID: ");

Empid=sc.nextInt();

System.out.println("Enter the Employee Name: ");

Name=sc.next();

System.out.println("Enter the Employee salary: ");

Salary=sc.nextInt();

System.out.println("Enter the Employee Address: ");

Address=sc.next();

}

}

class Teacher extends Employee

{

String Dep,Sub;

Teacher()

{

System.out.println("Enter the Department: ");

Dep=sc.next();

System.out.println("Enter the Subject: ");

Sub=sc.next();

}

void Display()

{

System.out.println("Employee ID :"+Empid);

System.out.println("Employee Name: "+Name);

System.out.println("Employee salary: "+Salary);

System.out.println("Employee Address: "+Address);

System.out.println("Department: "+Dep);

System.out.println("Subject: "+Sub);

}

}

class EmpTeacher{

public static void main(String[] a)

{

int n,i;

Scanner obj=new Scanner(System.in);

System.out.println("enter the number of employees :");

n=obj.nextInt();

Teacher array[]=new Teacher[n];

for(i=0;i<n;i++)

{

array[i]=new Teacher();

}

System.out.println("Employee Details ::");

for(i=0;i<n;i++)

{

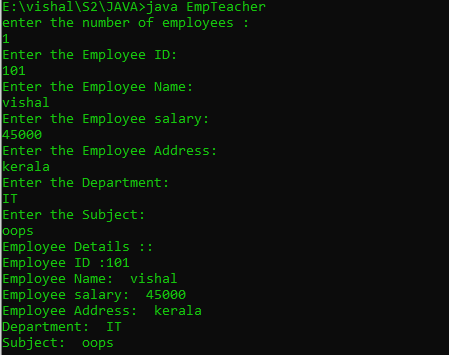
array[i].Display();

}

}

}

Output



Experiment : 03

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, 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.

CO 3:

Implement object-oriented concepts like inheritance, overloading and interfaces

Procedure

import java.util.Scanner;

class person{

String pname;

String pgender;

String paddress;

int page;

person(String name,String gender,String address,int age){

pname=name;

pgender=gender;

paddress=address;

page=age;

}

}

class employee extends person{

int empid;

String cmpny\_name;

String qualificatiion;

int salary;

employee(String name,String gender,String address,int age,int eid,String cmpny,String qualif,int sal){

super(name,gender,address,age);

empid=eid;

cmpny\_name=cmpny;

qualificatiion=qualif;

salary=sal;

}

}

class teacher extends employee{

int teacherid;

String subject;

String department;

teacher(String name,String gender,String address,int age,int eid,String cmpny,String qualif,int sal,int tid,String sub,String dep){

super(name,gender,address,age,eid,cmpny,qualif,sal);

teacherid=tid;

subject=sub;

department=dep;

}

void display(){

System.out.println("\n");

System.out.println("Person name:"+pname);

System.out.println("Person gender:"+pgender);

System.out.println("Person address:\n"+paddress);

System.out.println("Person age:"+page);

System.out.println("Employee id:"+empid);

System.out.println("Company name: "+cmpny\_name);

System.out.println("Employee qualification: "+qualificatiion);

System.out.println("Employee salary: "+salary);

System.out.println("Teacher id: "+teacherid);

System.out.println("Department: "+department);

System.out.println("Subject taught: "+subject);

}

}

public class SuperClass2{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

System.out.println("Enter the Limit:");

int limit=sc.nextInt();

teacher tcher[]=new teacher[limit];

for(int i=0;i<limit;i++)

{

System.out.println("Enter the Person name:");

String name=sc.next();

System.out.println("Enter the Person gender:");

String gender=sc.next();

System.out.println("Enter the Person address:");

String address=sc.next();

System.out.println("Enter the Person age:");

int age=sc.nextInt();

System.out.println("Enter the Employee id:");

int eid=sc.nextInt();

System.out.println("Enter the Company name:");

String cmpny=sc.next();

System.out.println("Enter the qualification:");

String qualif=sc.next();

System.out.println("Enter the Salary:");

int sal=sc.nextInt();

System.out.println("Enter the Teacher id:");

int tid=sc.nextInt();

System.out.println("Enter the Department:");

String dep=sc.next();

System.out.println("Enter the Subject:");

String sub=sc.next();

tcher[i]=new teacher(name,gender,address,age,eid,cmpny,qualif,sal,tid,dep,sub);

}

for(int i=0;i<limit;i++)

{

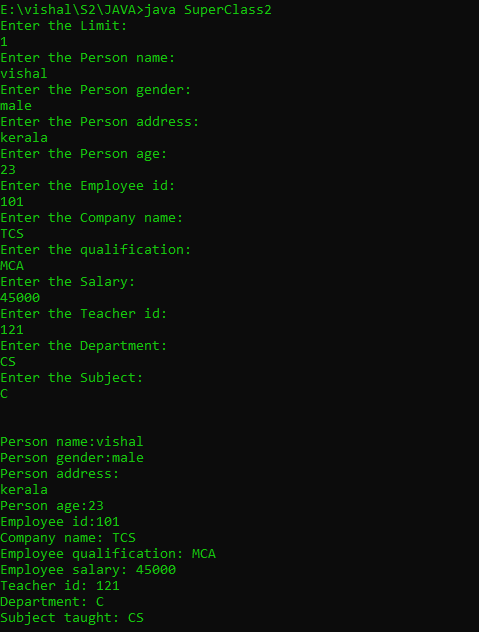
tcher[i].display();

}

}

}

Output



Experiment : 04

Aim :

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.

CO 3:

Implement object-oriented concepts like inheritance, overloading and interfaces

Procedure

import java.util.\*;

class publisher{

String pub\_name;

publisher()

{

Scanner obj=new Scanner(System.in);

System.out.println("Publisher name :");

pub\_name=obj.next();

}

}

class book extends publisher

{

String book\_name;

book()

{

Scanner obj=new Scanner(System.in);

System.out.println("book name :");

book\_name=obj.next();

}

}

class literature extends book

{

void display()

{

System.out.println("Publisher name is :"+pub\_name);

System.out.println("book name is :"+book\_name);

}

}

class fiction extends book

{

void display()

{

System.out.println("Publisher name is :"+pub\_name);

System.out.println("book name is :"+book\_name);

}

}

class library

{

public static void main(String[] args)

{

int i=0;

Scanner obj=new Scanner(System.in);

System.out.println("enter the total number :");

int size=obj.nextInt();

literature arr1[]= new literature[size];

fiction arr2[]=new fiction[size];

System.out.println("enter the details of literature books :");

for(i=0;i<size;i++)

{

arr1[i]=new literature();

}

System.out.println("enter the details of fiction books :");

for(i=0;i<size;i++)

{

arr2[i]=new fiction();

}

System.out.println("\n\n\n\n");

System.out.println("details of literature books :");

for(i=0;i<size;i++)

{

arr1[i].display();

}

System.out.println("\n\n\n\n");

System.out.println("details of fiction books :");

for(i=0;i<size;i++)

{

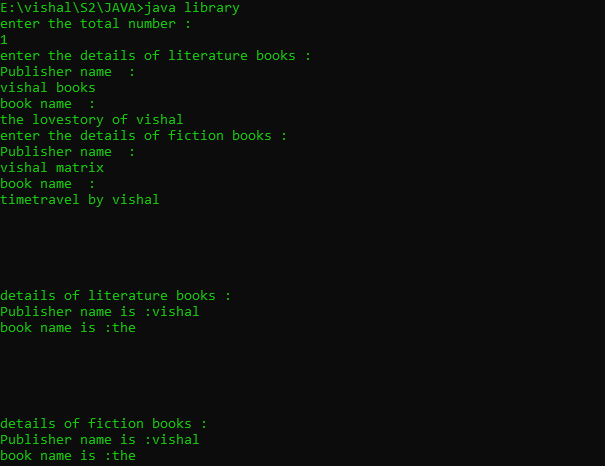
arr1[i].display();

}

}

}

Output



Experiment : 05

Aim :

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

CO 3:

Implement object-oriented concepts like inheritance, overloading and interfaces

Procedure

import java.util.\*;

interface student{

public void getS();

public void dispS();

}

interface sports

{

public void getSp();

public void dispSp();

}

class result implements student,sports

{

Scanner obj=new Scanner(System.in);

String name,spitem;

int m1,m2,roll,rank;

double total,percentage;

public void getS()

{

System.out.println("enter the name :");

name=obj.next();

System.out.println("enter the roll number :");

roll=obj.nextInt();

System.out.println("enter the mark 1 :");

m1=obj.nextInt();

System.out.println("enter the mark2 :");

m2=obj.nextInt();

total=m1+m2;

percentage=(total\*200)/100;

}

public void getSp()

{

System.out.println("enter the sports item which the student participated in :");

spitem=obj.next();

System.out.println("enter the rank of the student :");

rank=obj.nextInt();

}

public void dispS()

{

System.out.println("Name of the Student :"+name);

System.out.println("Roll Number of the Student :"+roll);

System.out.println("Mark 1 the Student :"+m1);

System.out.println("Mark 2 of the Student :"+m2);

System.out.println("Total Mark of the Student :"+total);

System.out.println("Percentage of the Student :"+percentage);

}

public void dispSp()

{

System.out.println("Sports item :"+spitem);

System.out.println("The Rank :"+rank);

}

}

public class multiinher{

public static void main(String[] args)

{

result obj=new result();

obj.getS();

obj.getSp();

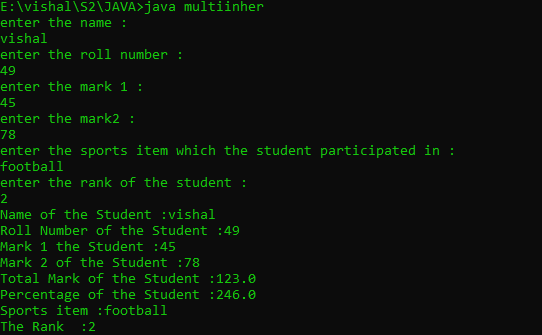
obj.dispS();

obj.dispSp();

}

}

Output



Experiment : 06

Aim :

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.

CO 3:

Implement object-oriented concepts like inheritance, overloading and interfaces

Procedure

import java.util.\*;

interface prototype{

public void getdata();

public void area();

public void perimeter();

}

class circle implements prototype

{

Scanner obj=new Scanner(System.in);

int radius;

double z=3.14;

public void getdata()

{

System.out.println("enter the radius :");

radius=obj.nextInt();

}

public void area()

{

System.out.println("area of circle :"+z\*(radius\*radius));

}

public void perimeter()

{

System.out.println("perimeter of circle :"+(2\*z)\*radius);

}

}

class rectangle implements prototype

{

Scanner obj=new Scanner(System.in);

int l,b;

public void getdata()

{

System.out.println("enter the lenght :");

l=obj.nextInt();

System.out.println("enter the breadth :");

b=obj.nextInt();

}

public void area()

{

System.out.println("area of rectangle :"+l\*b);

}

public void perimeter()

{

System.out.println("perimeter of rectangle :"+l+b);

}

}

class shape{

public static void main(String[] args)

{

int ch,u=0;

circle cc=new circle();

rectangle jj=new rectangle();

while(u==0)

{

Scanner obj=new Scanner(System.in);

System.out.println("1.Circle \n2.Rectangle \n3.Exit");

System.out.println("choose one :");

ch=obj.nextInt();

switch(ch)

{

case 1:cc.getdata();

cc.area();

cc.perimeter();

break;

case 2:jj.getdata();

jj.area();

jj.perimeter();

break;

case 3:System.exit(0);

default:

System.out.println("choose valid one :");

break;

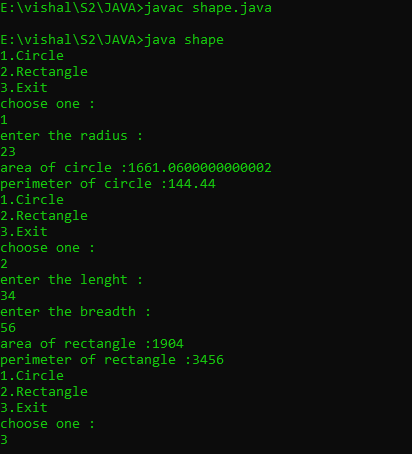
}

}

}

}

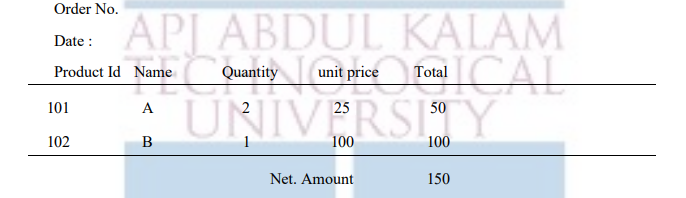
Output



Experiment : 07

Aim :

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



CO 3:

Implement object-oriented concepts like inheritance, overloading and interfaces

Procedure

import java.util.\*;

interface product

{

public void getdata();

public void display();

public void calc();

}

class calculate implements product

{

int pid;

String name;

double quantity;

double unit\_price,total;

public void getdata()

{

Scanner obj=new Scanner(System.in);

System.out.println("enter the product id :");

pid=obj.nextInt();

System.out.println("enter the name :");

name=obj.next();

System.out.println("enter the quantity :");

quantity=obj.nextDouble();

System.out.println("enter the unit\_price :");

unit\_price=obj.nextDouble();

}

public void calc()

{

total =quantity \* unit\_price;

}

public void display()

{

System.out.println(pid+"\t\t"+name+"\t\t"+quantity+"\t\t"+unit\_price+"\t\t"+total);

}

}

class bill{

public static void main(String[] args)

{

int i,odno,n;

String date;

double net\_amnt =0;

Scanner obj=new Scanner(System.in);

System.out.println("enter order number :");

odno=obj.nextInt();

System.out.println("enter the date :");

date=obj.next();

System.out.println("enter the total number products :");

n=obj.nextInt();

calculate arr[]= new calculate[n];

for(i=0;i<n;i++)

{

arr[i] = new calculate();

arr[i].getdata();

arr[i].calc();

}

System.out.println("Order no : #"+odno);

System.out.println("Date :"+date);

System.out.println("Product Id Name Quantity Unit Price Total");

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

for(i=0;i<n;i++)

{

arr[i].display();

net\_amnt +=arr[i].total;

}

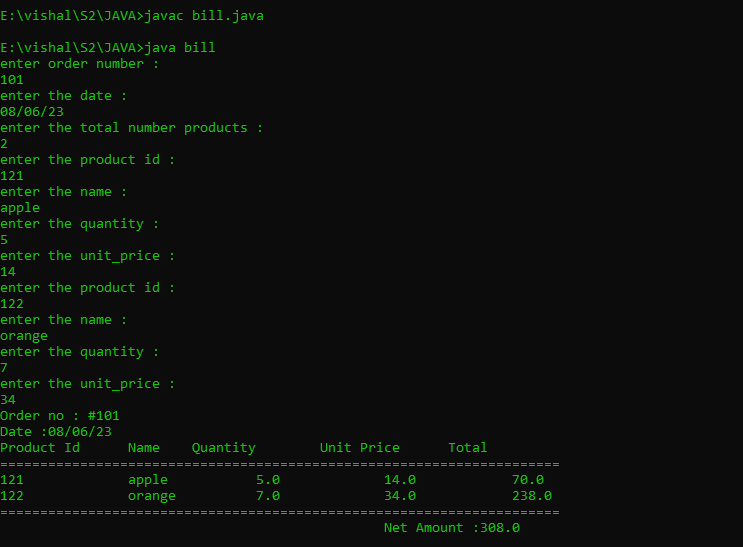
System.out.println("======================================================================");

System.out.println("\t\t\t\t\t\tNet Amount :"+net\_amnt);

}

}

Output



Experiment : 01

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.

CO 4:

Implement packages, exception handling, multithreading and generic programming.Use java.util package and Collection framework

Procedure

package graphics;

import java.util.\*;

interface shapes{

public double RecArea();

public double CircArea();

public double SquareArea();

public double TriangArea();

}

public class Graphics implements shapes {

Scanner obj = new Scanner(System.in);

int r,l,b,s;

double pi = 3.14, area;

public double RecArea(){

System.out.print("Enter the Length of Rectangle: ");

l=obj.nextInt();

System.out.print("Enter the Breadth of Rectangle: ");

b=obj.nextInt();

area=l\*b;

return area;

}

public double CircArea(){

System.out.print("Enter the Radius of Circle: ");

r =obj.nextInt();

area = pi \* r \* r;

return area;

}

public double SquareArea(){

System.out.print("Enter the Side of the Square: ");

s = obj.nextInt();

area = s \* s;

return area;

}

public double TriangArea(){

System.out.print("Enter the Width of the Triangle: ");

double base = obj.nextDouble();

System.out.print("Enter the Height of the Triangle: ");

double height = obj.nextDouble();

double area = (base\* height)/2;

return area;

}

}

import graphics.Graphics;

import java.util.\*;

public class Areas{

public static void main(String []args){

Scanner sc = new Scanner(System.in);

Graphics Obj = new Graphics();

int choice = 0;

while(choice != 5){

System.out.println("-------AREAS OF SHAPES--------\n1. Rectangle\n2. Circle\n3. Square\n4. Triangle\n5. Exit");

System.out.print("Enter your choice: ");

choice = sc.nextInt();

switch(choice){

case 1:

System.out.println("Area of Rectangle: " + Obj.RecArea());

break;

case 2:

System.out.println("Area of Circle: " + Obj.CircArea());

break;

case 3:

System.out.println("Area of Square: " + Obj.SquareArea());

break;

case 4:

System.out.println("Area of Triangle: " + Obj.TriangArea());

break;

case 5:

System.exit(0);

break;

default:

System.out.println("Select a valid option!");

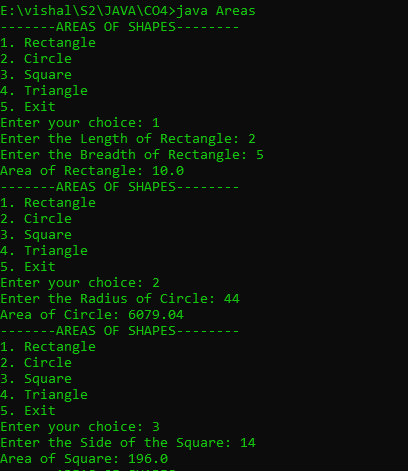
}

}

}

}

Output



Experiment : 02

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

CO 4:

Implement packages, exception handling, multithreading and generic programming.Use java.util package and Collection framework

Procedure

package arithmetic;

import java.util.\*;

interface maths

{

public double add();

public double subtract();

public double multiply();

public double division();

}

public class Arithmetic implements maths

{

Scanner obj=new Scanner(System.in);

int a1,a2,s1,s2,m1,m2,d1,d2;

double total;

public double add()

{

System.out.println("enter a number :");

a1=obj.nextInt();

System.out.println("enter a number :");

a2=obj.nextInt();

total=a1+a2;

return total;

}

public double subtract()

{

System.out.println("enter a number :");

s1=obj.nextInt();

System.out.println("enter a number :");

s2=obj.nextInt();

total=s1-s2;

return total;

}

public double multiply()

{

System.out.println("enter a number :");

m1=obj.nextInt();

System.out.println("enter a number :");

m2=obj.nextInt();

total=m1\*m2;

return total;

}

public double division()

{

System.out.println("enter a number :");

d1=obj.nextInt();

System.out.println("enter a number :");

d2=obj.nextInt();

total=d1/d2;

return total;

}

}

import arithmetic.Arithmetic;

import java.util.\*;

public class maths

{

public static void main(String args[])

{

Scanner obj=new Scanner(System.in);

Arithmetic sc=new Arithmetic();

int choice=0;

while(choice!=5)

{

System.out.println("------ARITHMETIC OPERATIONS-------\n1.Addition\n2.Subtraction\n3.Multiplication\n4.Division\n5.Exit");

System.out.println("enter a choice :");

choice=obj.nextInt();

switch(choice)

{

case 1:

System.out.println("the total is :"+sc.add());

break;

case 2:

System.out.println("the total is :"+sc.subtract());

break;

case 3:

System.out.println("the total is :"+sc.multiply());

break;

case 4:

System.out.println("the total is :"+sc.division());

break;

case 5:

System.exit(0);

break;

default:

System.out.println("Select a valid option!");

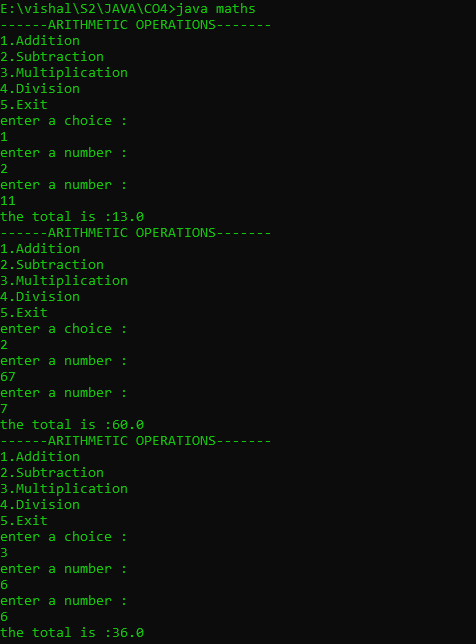
}

}

}

}

Output



Experiment : 03

Aim :

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

CO 4:

Implement packages, exception handling, multithreading and generic programming.Use java.util package and Collection framework

Procedure

import java.util.\*;

class UsernameException extends Exception {

public UsernameException(String msg) {

super(msg);

}

}

class PasswordException extends Exception {

public PasswordException(String msg) {

super(msg);

}

}

public class CheckLoginCredential {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

String username, password;

System.out.print("Enter username : ");

username = s.nextLine();

System.out.print("Enter password : ");

password = s.nextLine();

int length = username.length();

try {

if(length < 6)

throw new UsernameException("Username must be greater than 6 characters......");

else if(!password.equals("qwerty"))

throw new PasswordException("Incorrect password\nType correct password.......");

else

System.out.println("Login Successful !!!");

}

catch (UsernameException u) {

u.printStackTrace();

}

catch (PasswordException p) {

p.printStackTrace();

}

finally {

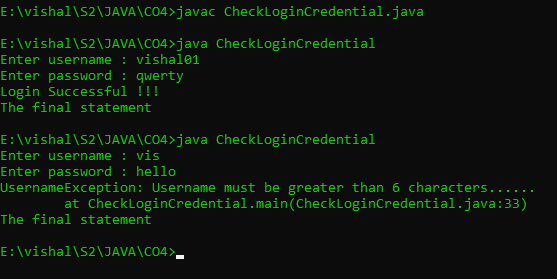
System.out.println("The final statement");

}

}

}

Output



Experiment : 04

Aim :

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

CO 4:

Implement packages, exception handling, multithreading and generic programming.Use java.util package and Collection framework

Procedure

import java.util.Scanner;

class nIntExcep extends Exception{

public nIntExcep(String str){

super(str);

}

}

public class average{

public static void main(String[] args){

Scanner Snr=new Scanner(System.in);

int arr[];

int sz, total=0, avg, count=0;

System.out.print("Enter the limit: ");

sz = Snr.nextInt();

arr = new int[sz];

for(int i=0;i<sz;i++)

{

System.out.print("Enter the value: ");

int val = Snr.nextInt();

arr[i] = val;

}

try {

for(int i=0;i<sz;i++) {

if(arr[i]<0){

throw new nIntExcep("Numbers must be positive");

}

else{

total += arr[i];

count++;

} }

avg=total/count;

System.out.println("Average :"+avg);

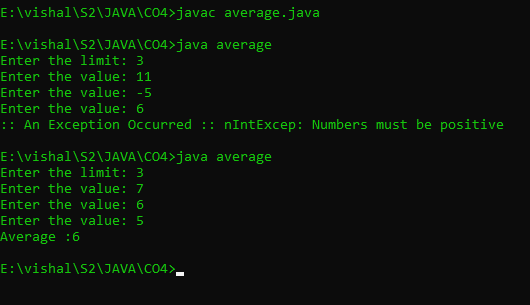
}

catch(nIntExcep e){

System.out.println(":: An Exception Occurred :: "+ e);

} }}

Output



Experiment : 05

Aim :

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

CO 4:

Implement packages, exception handling, multithreading and generic programming.Use java.util package and Collection framework

Procedure

import java.util.\*;

class MulTable extends Thread

{

public void run()

{

int num=5;

System.out.println("THE MULTIPLICATION TABLE::");

for(int i=0;i<=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 sc=new Scanner(System.in);

System.out.println("first N prime numbers");

System.out.println("enter the limt :");

int N=sc.nextInt();

System.out.println("the prime numbers between 1 and"+N+"are");

for(i=1;i<=N;i++)

{

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.println(i+" ");

}

}

}

}

public class PrimeThread

{

public static void main(String[] args) throws InterruptedException

{

MulTable a=new MulTable();

a.start();

a.sleep(2000);

PrimeNo b=new PrimeNo();

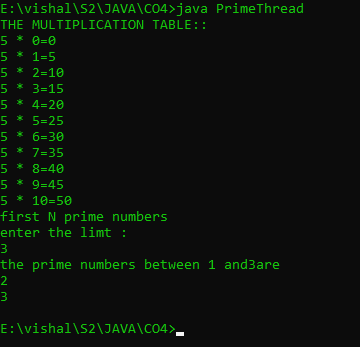
b.start();

b.sleep(200);

}

}

Output



Experiment : 06

Aim :

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

CO 4:

Implement packages, exception handling, multithreading and generic programming.Use java.util package and Collection framework

Procedure

import java.util.\*;

class fibonacci implements Runnable

{

public void run()

{

int first=0,second=1,next;

Scanner sc=new Scanner(System.in);

System.out.println("THE FIBONACCI SERIES");

System.out.println("enter the total number of terms");

int n=sc.nextInt();

System.out.println("THE GENERATED SERIES ::");

for(int i=0;i<=n;++i)

{

System.out.println(first+" ");

next=first+second;

first=second;

second=next;

}

}

}

class evenNo implements Runnable

{

public void run()

{

Scanner sc=new Scanner(System.in);

int lower,upper;

System.out.println("THE EVEN NUMBERS");

System.out.println("enter the lower limit");

lower=sc.nextInt();

System.out.println("enter the upper limit");

upper=sc.nextInt();

System.out.println("the even numbers from"+lower+"and"+upper+"are");

for(int i=lower;i<=upper;i++)

{

if(i%2!=0)

{

continue;

}

else

{

System.out.println(i+" ");

}

}

}

}

public class threadprgrm

{

public static void main(String[] args)throws InterruptedException

{

fibonacci obj1=new fibonacci();

Thread a=new Thread(obj1);

a.start();

a.sleep(2000);

evenNo obj2=new evenNo();

Thread b=new Thread(obj2);

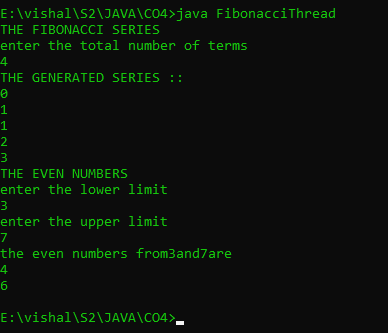
b.start();

b.sleep(1000);

}

}

Output



Experiment : 08

Aim :

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

CO 4:

Implement packages, exception handling, multithreading and generic programming.Use java.util package and Collection framework

Procedure

import java.util.\*;

class arrayStack

{

public int arr[];

public int top, size, len;

public arrayStack(int n)

{

size = n;

len = 0;

arr = new int[size];

top = -1;

}

public boolean isEmpty()

{

return top == -1;

}

public boolean isFull()

{

return top == size -1 ;

}

public int peek()

{

return arr[top];

}

public void push(int k)

{

if(top + 1 >= size)

System.out.println(" overflow ");

if(top + 1 < size )

arr[++top] = k;

}

public int pop()

{

if( isEmpty() )

System.out.println(" underflow ");

return arr[top--];

}

public void display()

{

System.out.print("\nStack = ");

for (int i = top; i >= 0; i--)

System.out.print(arr[i]+" ");

System.out.println();

}

}

public class stackimp

{

public static void main(String[] args)

{

Scanner obj = new Scanner(System.in);

System.out.println("Enter Size of the Stack ");

int n = obj.nextInt();

arrayStack stk = new arrayStack(n);

int ch = 7;

do{

System.out.println("\nStack Operations");

System.out.println("1. push");

System.out.println("2. pop");

System.out.println("3. peek");

System.out.println("4. check empty");

System.out.println("5. check full");

int choice = obj.nextInt();

switch (choice)

{

case 1 :

System.out.println("Enter integer element to push");

stk.push( obj.nextInt() );

break;

case 2 :

System.out.println("Popped Element = " + stk.pop());

break;

case 3 :

System.out.println("Peek Element = " + stk.peek());

break;

case 4 :

System.out.println("Empty status = " + stk.isEmpty());

break;

case 5 :

System.out.println("Full status = " + stk.isFull());

break;

default :

System.out.println("enter valid option \n ");

break;

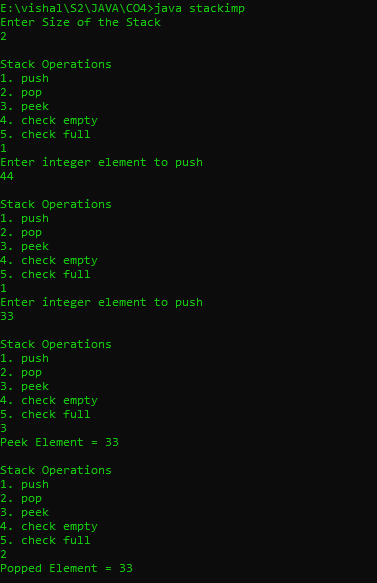
}

} while (ch != 0);

}

}

Output



Experiment : 09

Aim :

Using generic method perform Bubble sort.

CO 4:

Implement packages, exception handling, multithreading and generic programming.Use java.util package and Collection framework

Procedure

import java.util.\*;

public class BubbleSort{

int sz;

int[] Arr;

public BubbleSort(int n){

sz = n;

Arr = new int[sz];

}

public void insert(int i, int f){

Arr[i] = f;

}

public void display(int i){

System.out.print(Arr[i]+ " ");

}

public void Sort(int n){

int temp;

for(int i=0; i<n; i++){

for(int j=i+1; j<n; j++){

if(Arr[i] > Arr[j]){

temp = Arr[i];

Arr[i] = Arr[j];

Arr[j] = temp;

} }}}

public static void main(String[] args){

Scanner Snr= new Scanner(System.in);

System.out.println("Enter the number of elements: ");

int size = Snr.nextInt();

BubbleSort arr = new BubbleSort(size);

for(int i=0; i<size; i++){

System.out.print("Enter the element: ");

int val = Snr.nextInt();

arr.insert(i, val);

}

System.out.print("Before sorting: ");

for(int i=0; i<size; i++){

arr.display(i);

}

System.out.print("\nAfter sorting: ");

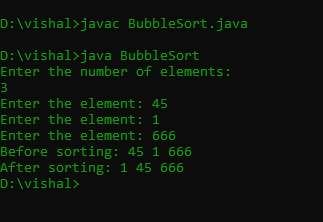
for(int i=0; i<size; i++){

arr.Sort(size);

arr.display(i);

}}}

Output



Experiment : 10

Aim :

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

CO 4:

Implement packages, exception handling, multithreading and generic programming.Use java.util package and Collection framework

Procedure

import java.util.\*;

public class arraylst

{

public static void main(String args[])

{

ArrayList<String> newlist=new ArrayList<String>();

Scanner obj=new Scanner(System.in);

System.out.println("enter the size of the array :");

int sz= obj.nextInt();

String item;

for(int i=0;i<sz;i++)

{

System.out.println("enter the array elements in string :");

item=obj.next();

newlist.add(item);

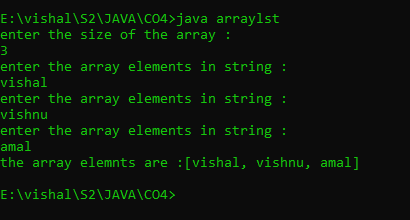
}

System.out.println("the array elemnts are :"+newlist);

}

}

Output



Experiment : 11

Aim :

Program to remove all the elements from a linked list

CO 4:

Implement packages, exception handling, multithreading and generic programming.Use java.util package and Collection framework

Procedure

Experiment : 12

Aim :

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

CO 4:

Implement packages, exception handling, multithreading and generic programming.Use java.util package and Collection framework

Procedure

import java.util.\*;

public class remove{

public static void main(String args[])

{

Stack<String> StackDemo=new Stack<String>();

Scanner obj=new Scanner(System.in);

String n;

System.out.println("enter the size");

int sz=obj.nextInt();

for(int i=0;i<sz;i++)

{

System.out.println("enter the value");

n=obj.next();

StackDemo.add(n);

}

System.out.println("Stack : "+ StackDemo);

String rem\_ele = StackDemo.remove(2);

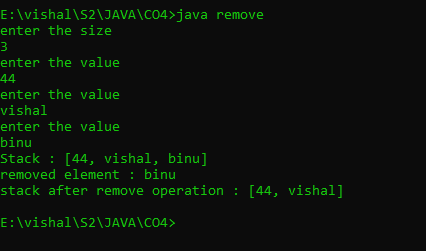
System.out.println("removed element : "+rem\_ele);

System.out.println("stack after remove operation : "+StackDemo);

}

}

Output



Experiment : 13

Aim :

Program to demonstrate the creation of queue object using the PriorityQueue class

CO 4:

Implement packages, exception handling, multithreading and generic programming.Use java.util package and Collection framework

Procedure

import java.util.\*;

class priorityQ{

public static void main(String args[])

{

PriorityQueue<Integer> priorQ = new PriorityQueue<Integer>();

Scanner obj=new Scanner(System.in);

int n;

System.out.println("enter the size");

int sz=obj.nextInt();

for(int i=0;i<sz;i++)

{

System.out.println("enter the value");

n=obj.nextInt();

priorQ.add(n);

}

System.out.println(priorQ.peek());

System.out.println(priorQ);

System.out.println(priorQ.poll());

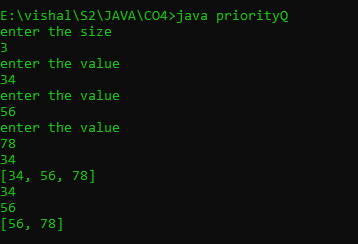
System.out.println(priorQ.peek());

System.out.println(priorQ);

}

}

Output



Experiment : 14

Aim :

Program to demonstrate the addition and deletion of elements in deque

CO 4:

Implement packages, exception handling, multithreading and generic programming.Use java.util package and Collection framework

Procedure

import java.util.\*;

public class Dqueue{

public static void main(String args[])

{

Deque<Integer> dq = new ArrayDeque<>();

Scanner obj=new Scanner(System.in);

int n;

System.out.println("enter the size");

int sz=obj.nextInt();

for(int i=0;i<sz;i++)

{

System.out.println("enter the value through front end");

n=obj.nextInt();

dq.addFirst(n);

}

for(int i=0;i<sz;i++)

{

System.out.println("enter the value through last end");

n=obj.nextInt();

dq.addLast(n);

}

int first = dq.removeFirst();

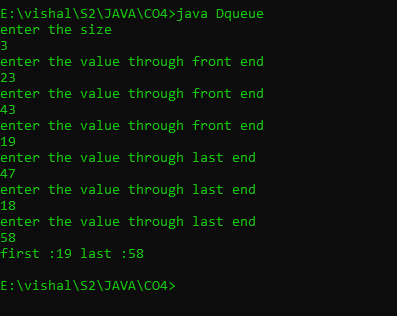
int last = dq.removeLast();

System.out.println("first :"+ first +" last :"+ last);

}

}

Output



Experiment : 15

Aim :

Program to demonstrate the addition and deletion of elements in deque

CO 4:

Implement packages, exception handling, multithreading and generic programming.Use java.util package and Collection framework

Procedure