|  |
| --- |
|  |
| JAVA PROGRAMMING |
| Theory Assignment |

*Name:D Dharma teja*

*Roll no.: 22011F0027.*

**Dept. of IT, JNTUHUCEST**

**MCA-II Sem. 2022-24 Batch**

**JAVA PROGRAMMING ASSIGNMENT-1**

**1. Write a Java program that uses functions to perform the following:**

**i)sorts a list of names in ascending order using bubble sort.**

**ii)then searches for a key value(name) non recursively in the above sorted list**

**using binary search.**

**Code:**

import java.util.Scanner;

public class NameListSort {

static void bSortNames(String[] s){

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

for(int j=0;j<s.length-i-1;j++){

if(s[j].compareTo(s[j+1])>0){

String temp=s[j];

s[j]=s[j+1];

s[j+1]=temp;

}

}

}

}

static void bSearch(String []s,String key){

boolean found = false;

int min=0,max=s.length-1;

while(min<=max&&found==false){

int mid=(min+max)/2;

if(key.equalsIgnoreCase(s[mid])){

found=true;

System.out.println("\'"+key+"\' found at index: "+mid);

break;

}

if(key.compareToIgnoreCase(s[mid])<0){

max=mid-1;

}else if(key.compareToIgnoreCase(s[mid])>0){

min=mid+1;

}

}

if(found==false)

System.out.println("\'"+key+"\' not found in the list of names.");

}

static void printArray(String[] s){

for(int i=0;i<s.length;i++)

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

System.out.println();

}

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

System.out.println("enter the number of names you want in the list: ");

int n =sc.nextInt();

String[] names = new String[n];

System.out.println("enter the names to be added in the list");

for (int i=0;i<names.length;i++)

names[i]=sc.next();

System.out.println("names entered are: ");

printArray(names);

bSortNames(names);

System.out.println("after sorting");

printArray(names);

System.out.println("enter the name to be searched in the list ");

String key = sc.next();

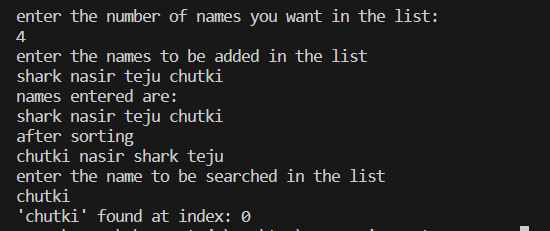
bSearch(names, key);

sc.close();

}

}

**Output:**

****

**2. Write a Java program to implement the matrix ADT using a class. The operations supported by this ADT are:**

**a)Reading a matrix. b)Addition of two matrices. c)Printing a matrix.**

**d)Multiplication of two matrices. Program should handle the exceptions that may arise duringthe matrix operations.**

**Code:**

import java.util.\*;

public class Matrix {

    //member variables

    int row,column;

    int[][] matrix;

    //constructors

    Matrix(){

        row=0;column=0;

        matrix = new int[row][column];

    }

    Matrix(int a,int b){

        row=a;column=b;

        matrix=new int[row][column];

    }

    //member functions

    void readMatrix(){

        Scanner sc = new Scanner(System.in);

        System.out.println("enter the elements of the matrix:");

        for(int i=0;i<this.row;i++){

            for(int j=0;j<this.column;j++){

                this.matrix[i][j]=sc.nextInt();

            }

        }

    }

    void printMatrix(){

        for(int i=0;i<this.row;i++){

            for(int j=0;j<this.column;j++){

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

            }

            System.out.println();

        }

    }

    Matrix addMatrix(Matrix a)throws MatrixAdditionException{

        Matrix result=null;

        if(this.row==a.row&&this.column==a.column){

            result= new Matrix(this.row,this.column);

            for(int i=0;i<this.row;i++){

                for(int j=0;j<this.column;j++){

                    result.matrix[i][j]=this.matrix[i][j]+a.matrix[i][j];

                }

            }

        }else{

            // System.out.println("addition is not possible");

            throw new MatrixAdditionException();

        }

        return result;

    }

    Matrix multiplyMatrix(Matrix a)throws MatrixMultiplicationException {

        Matrix result = null;

        if(this.column==a.row){

            result = new Matrix(this.row, a.column);

            for(int i=0;i<this.row;i++){

                for(int j=0;j<a.column;j++){

                    for(int k=0;k<a.row;k++){

                        result.matrix[i][j]+=this.matrix[i][k]+a.matrix[k][j];

                    }

                }

            }

        }

        else{

            // System.out.println("matrix multiplication is NOT possible!!");

            throw new MatrixMultiplicationException();

        }

        return result;

    }

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        System.out.println("enter the order of matrix 1");

        int r1=sc.nextInt(),c1=sc.nextInt();

        Matrix m1=new Matrix(r1,c1);

        System.out.println("enter the order of matrix 2");

        int r2=sc.nextInt(),c2=sc.nextInt();

        Matrix m2= new Matrix(r2,c2);

        System.out.println("for matrix 1");

        m1.readMatrix();

        System.out.println("for matrix 2");

        m2.readMatrix();

        do{

            System.out.print("1.Add \n2. multiply \n3.exit \nenter your choice: ");

            int choice=sc.nextInt();

            switch(choice){

                case 1:

                        try{

                            System.out.println("after adding ");

                            m1.addMatrix(m2).printMatrix();

                        }catch(MatrixAdditionException ma){

                            System.out.println("matrix addition exception caught");

                            ma.printStackTrace();

                        }catch(Exception e){

                            System.out.println("unknown exception caught!!");

                            e.printStackTrace();

                        }

                        break;

                case 2:

                        try{

                            System.out.println("after multiplication");

                            m1.multiplyMatrix(m2).printMatrix();

                        }catch(MatrixMultiplicationExceptionmme){

                            System.out.println("matrix multiplication exception caught");

                            mme.printStackTrace();

                        }catch(Exception e){

                            System.out.println("unknown exception caught");

                            e.printStackTrace();

                        }

                        break;

                case 3:

                        System.out.println(" Program Terminated Successfully.");

                        sc.close();

                        System.exit(0);

            }

        }while(true);

    }

}

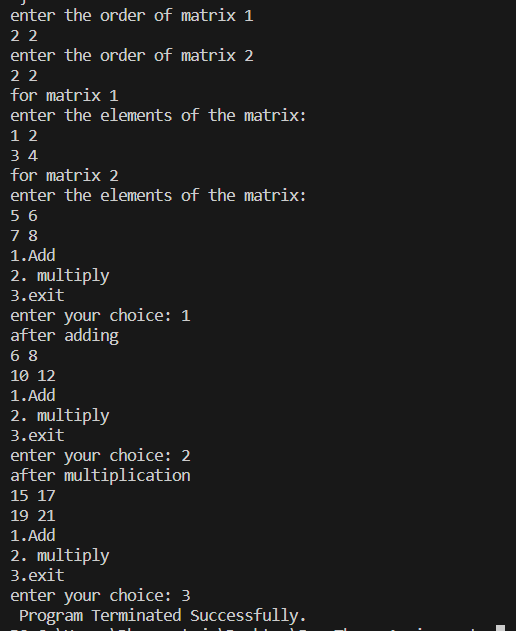
class MatrixAdditionException extends Exception {

}

class MatrixMultiplicationException extends Exception{

}

**Output:**

****

**3. Implement the complex number ADT in Java using a class. The complex ADT is used to represent complex numbers of the form c = a + ib, where a and b are real numbers. The operations supported by this ADT are:**

**a)Reading a complex number. b)Writing a complex number. c)Addition of two complex numbers.d)Multiplication of two complex numbers.**

**Code:**

import java.util.\*;

public class Complex {

int real,img;

void readComplex() {

Scanner input =new Scanner(System.in);

System.out.println("enter the real part ");

this.real=input.nextInt();

System.out.println("enter the imaginary part ");

this.img=input.nextInt();

}

void writeComplex(){

if(this.img>0)

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

else

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

}

void addComplex(Complex c){

Complex result=new Complex();

result.real=this.real+c.real;

result.img=this.img+c.img;

System.out.println("after addition:");

result.writeComplex();

}

void multiplyComplex(Complex c){

Complex result =new Complex();

result.real=this.real\*c.real-(this.img\*c.img);

result.img=this.real\*c.img+this.img+c.real;

System.out.println("after multiplication :");

result.writeComplex();

}

public static void main(String[] args) {

System.out.println("first number :");

Complex c1 = new Complex();

c1.readComplex();

System.out.println("second number");

Complex c2=new Complex();

c2.readComplex();

Scanner Sc = new Scanner(System.in);

int choice;

do {

System.out.println("enter\n1.for Addition\n2.for multiplication\n3. To exit");

choice=Sc.nextInt();

switch(choice){

case 1:c1.addComplex(c2);

break;

case 2:c1.multiplyComplex(c2);

break;

default:

Sc.close();

System.exit(0);

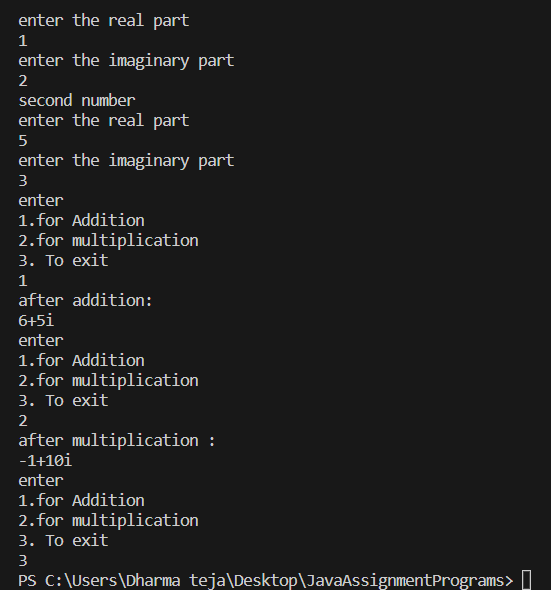
}

}while(true);

}

}

**Output:**

****

**4. Write a Java program that has an abstract class Polygon with two abstract methods**

**area() and perimeter(). Derive three classes Rectangle, Square, and Triangle from**

**Polygon class.Write methods to get the details of their dimensions and hence**

**calculate their areas and perimeters The sub classes should override the two abstract**

**methods (area() and perimeter()) by providing appropriate implementations.**

**Demonstrate run time polymorphism.**

**Code:**

import java.util.Scanner;

abstract class Polygon{

    abstract double area();

    abstract double perimeter();

}

class Square extends Polygon{

    double side;

    Square(){

        side=0.0;

    }

    Square(double a){

        side=a;

    }

    double area(){

        return side\*side;

    }

    double perimeter(){

        return 4\*side;

    }

}

class Rectangle extends Polygon{

    double length,width;

    Rectangle(){

        length=0.0;width=0.0;

    }

    Rectangle(double a,double b){

        length=a;width=b;

    }

    double area(){

        return length\*width;

    }

    double perimeter(){

        return 2\*(length+width);

    }

}

class Triangle extends Polygon{

    double a,b,c;

    Triangle(){

        a=b=c=0.0;

    }

    Triangle(double x,doubley,double z){

        a=x;b=y;c=z;

    }

    double area(){

        double s=(a+b+c)/2;

        return Math.sqrt(s\*(s-a)\*(s-b)\*(s-c));

    }

    double perimeter(){

        return a+b+c;

    }

}

public class Polymorphism {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        do{

            System.out.println("enter choice:\n1.Square\n2.Rectangle\n3.Triangle\n4.Exit");

            int choice=sc.nextInt();

            switch(choice){

                case 1:

                        System.out.println("enter the side of the square");

                        double s=sc.nextDouble();

                        Square sq = new Square(s);

                        System.out.println("perimeter of the Square: "+sq.perimeter());

                        System.out.println("area of the square :"+sq.area());

                        break;

                case 2:

                        System.out.println("enter the length and width of the Rectangle");

                        double l=sc.nextDouble(),w=sc.nextDouble();

                        Rectangle r = new Rectangle(l, w);

                        System.out.println("perimeter of the Rectangle: "+r.perimeter());

                        System.out.println("area of the Rectangle :"+r.area());

                        break;

                case 3:

                        System.out.println("enter the sides of the triangle: ");

                        double a=sc.nextDouble(),b=sc.nextDouble(),c=sc.nextDouble();

                        Triangle t = new Triangle(a,b,c);

                        System.out.println("area of the Triangle :"+t.area());

                        System.out.println("perimeter of the Triangle: "+t.perimeter());

                        break;

                case 4:

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

                        sc.close();

                        System.exit(0);

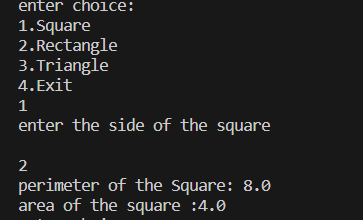
            }

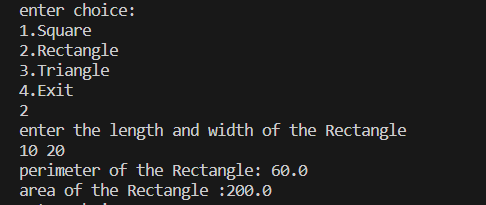
        }while(true);

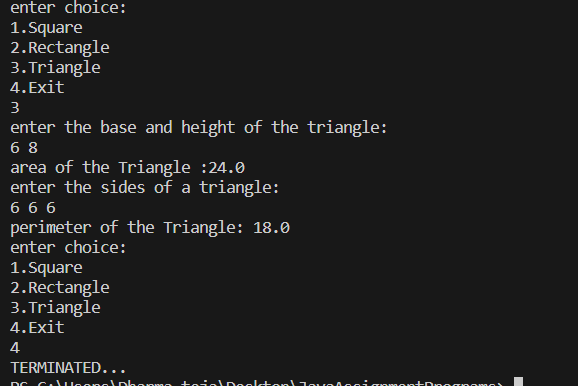
    }

}

**Output:**







**5. Write a class for stack in Java. The class should have the methods, push(),pop(),isEmpty(),isFull() and size().Use an array representation for the stack. Write a Java program that readsa list of integers, uses the above stack methods and displays list of integers in reverse order.**

**Code:**

import java.util.Scanner;

public class Stack {

int size;

int top=-1;

int[] arr ;

Stack(int s){

size=s;

arr=new int[size];

}

booleanisEmpty(){

if(this.top==-1)

return true;

else

return false;

}

booleanisFull(){

if(this.top>=this.size){

return true;

}

else

return false;

}

void push(int a){

++top;

if(isFull()){

System.out.println("Stack overflow!!-Cannot push the element.");

top--;

return;

}

else{

arr[top]=a;

System.out.println("pushed '" +a+"' into Stack");

}

}

void pop(){

if(this.isEmpty()){

System.out.println("stack is empty!!");

}else{

System.out.println("poped '" +arr[top]+"' from Stack");

--top;

}

}

void displayStack(){

if(isEmpty()){

System.out.println("No elements in the stack!");

}else{

System.out.println("elements in the stack: ");

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

System.out.println("\t"+arr[i]);

}

}

}

public static void main(String[] args) {

Scanner sc= new Scanner(System.in);

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

int S=sc.nextInt(),choice;

Stack s=new Stack(S);

do{

System.out.println(":::enterOperarionToBePerformed:::\n\t1.push\n\t2.pop\n\t3.Display\n\t4.exit");

choice=sc.nextInt();

switch(choice){

case 1:

System.out.println("enter the integer to be pushed: ");

s.push(sc.nextInt());

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

break;

case 2:

s.pop();

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

break;

case 3:

s.displayStack();

break;

case 4:

System.exit(0);

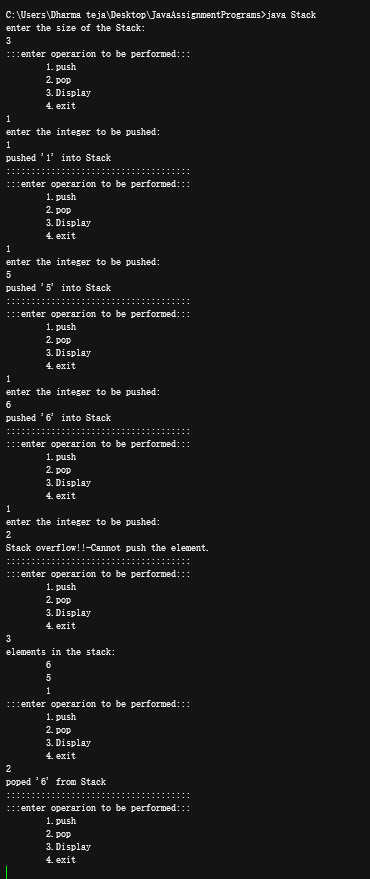
}

}while(true);

}

}

**Output:**



**6. Write Java program that uses functions to perform the following operations:**

**a)create a doubly linked list of integers.**

**b)traverse the above list and display its contents.**

**c)delete an integer from the above list and display the contents of the list after deletion.**

**Code:**

public class DoublyLL {

private Node head;

private Node tail;

private class Node {

int data;

Node previous;

Node next;

Node(int data) {

this.data = data;

this.previous = null;

this.next = null;

}

}

public void insert(int data) {

Node newNode = new Node(data);

if (head == null) {

head = tail = newNode;

} else {

tail.next = newNode;

newNode.previous = tail;

tail = newNode;

}

}

public void display() {

Node current = head;

if (head == null) {

System.out.println("No elements in the list");

return;

}

System.out.println("elements in the list:");

while (current != null) {

System.out.print(current.data + " ");

current = current.next;

}

System.out.println();

}

public void delete(int data) {

Node current = head;

while (current != null) {

if (current.data == data) {

if (current == head) {

head = head.next;

if (head != null) {

head.previous = null;

} else {

tail = null;

}

} else if (current == tail) {

tail = tail.previous;

tail.next = null;

} else {

current.previous.next = current.next;

current.next.previous = current.previous;

}

System.out.println("Deleted " + data + " from the list.");

return;

}

current = current.next;

}

System.out.println(data + " not found in the list.");

}

public static void main(String[] args) {

DoublyLL list = new DoublyLL();

list.insert(10);

list.insert(20);

list.insert(30);

list.insert(40);

list.insert(50);

list.display();

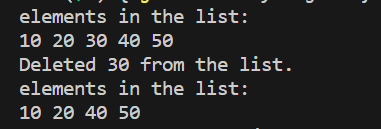
list.delete(30);

list.display();

}

}

**Output:**



**7. Write your own example Java program that demonstrates how run time polymorphism isachieved using Interface reference.**

**Code:**

import java.util.Scanner;

interface Polygon{

    double area();

    double perimeter();

}

class Square implements Polygon{

    double side;

    Square(){

        side=0.0;

    }

    Square(double a){

        side=a;

    }

    public double area(){

        return side\*side;

    }

    public double perimeter(){

        return 4\*side;

    }

}

class Rectangle implements Polygon{

    double length,width;

    Rectangle(){

        length=0.0;width=0.0;

    }

    Rectangle(double a,double b){

        length=a;width=b;

    }

    public double area(){

        return length\*width;

    }

    public double perimeter(){

        return 2\*(length+width);

    }

}

class Triangle implements Polygon{

    double a,b,c;

    Triangle(){

        a=b=c=0.0;

    }

    Triangle(double x,doubley,double z){

        a=x;b=y;c=z;

    }

    public double area(){

        double s=(a+b+c)/2;

        return Math.sqrt(s\*(s-a)\*(s-b)\*(s-c));

    }

    public double perimeter(){

        return a+b+c;

    }

}

public class InterfacePolymorphism {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        do{

            System.out.println("enter choice:\n1.Square\n2.Rectangle\n3.Triangle\n4.Exit");

            int choice=sc.nextInt();

            switch(choice){

                case 1:

                        System.out.println("enter the side of the square");

                        double s=sc.nextDouble();

                        Square sq = new Square(s);

                        System.out.println("perimeter of the Square: "+sq.perimeter());

                        System.out.println("area of the square :"+sq.area());

                        break;

                case 2:

                        System.out.println("enter the length and width of the Rectangle");

                        double l=sc.nextDouble(),w=sc.nextDouble();

                        Rectangle r = new Rectangle(l, w);

                        System.out.println("perimeter of the Rectangle: "+r.perimeter());

                        System.out.println("area of the Rectangle :"+r.area());

                        break;

                case 3:

                        System.out.println("enter the sides of the triangle: ");

                        double a=sc.nextDouble(),b=sc.nextDouble(),c=sc.nextDouble();

                        Triangle t = new Triangle(a,b,c);

                        System.out.println("area of the Triangle :"+t.area());

                        System.out.println("perimeter of the Triangle: "+t.perimeter());

                        break;

                case 4:

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

                        sc.close();

                        System.exit(0);

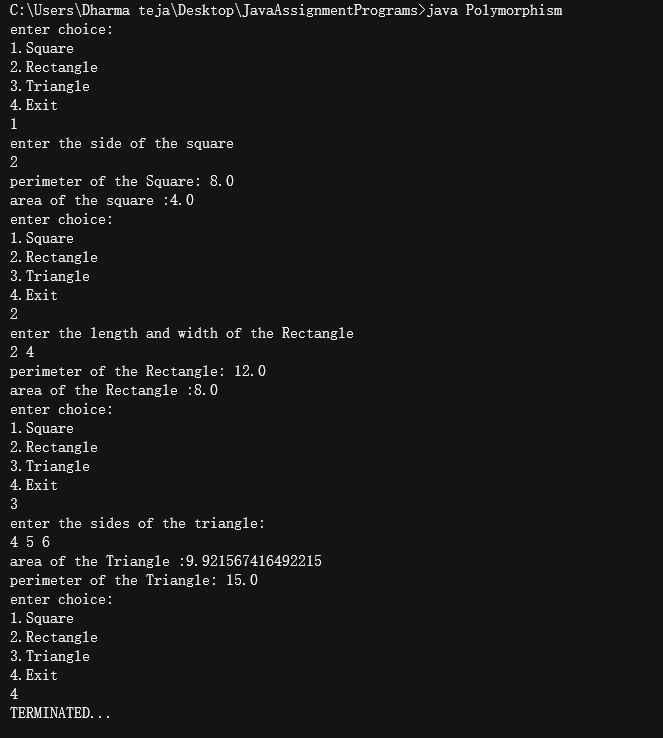
            }

        }while(true);

    }

}

**Output:**



**8. Write a Java program that copies the contents of one file to another. File names are passed**

**as command-line arguments. Program should handle the exceptions that may arise during**

**the file operations.**

**Code:**

import java.io.\*;

import java.util.Scanner;

public class FileE {

    public static void main(String[] args) {

       File f1=null,f2=null;

       Scanner sc=null;

       try{

        f1=new File(args[0]);

        f2=new File(args[1]);

        sc=new Scanner(f1);

        StringBuffer buffer = new StringBuffer();

        while(sc.hasNextLine()){

            buffer.append("\n"+sc.nextLine());

        }

        PrintWriter pw = new PrintWriter(f2);

        pw.println(buffer);

        pw.flush();

        pw.close();

        sc.close();

        System.out.println("copied contents of "+f1.getName()+" to "+f2.getName());

       }catch(FileNotFoundExceptionfnf){

        System.out.println("file Dosen't exist!!");

       }

    }

}

**Output:**



