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!");

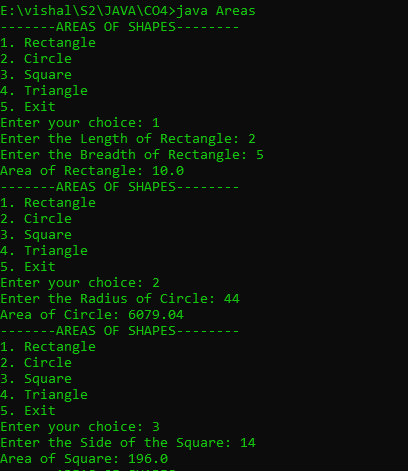
}

}

}

}

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!");

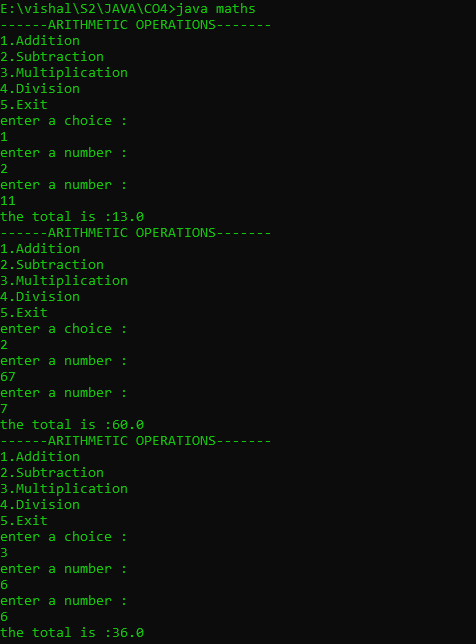
}

}

}

}

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 {

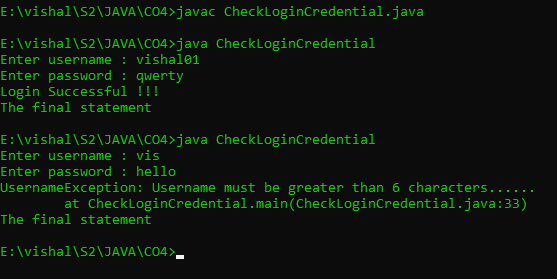
System.out.println("The final statement");

}

}

}

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){

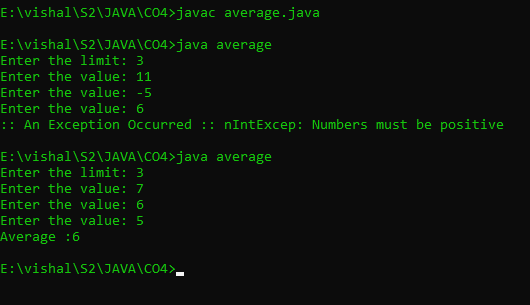
System.out.println(":: An Exception Occurred :: "+ e);

}

}

}

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();

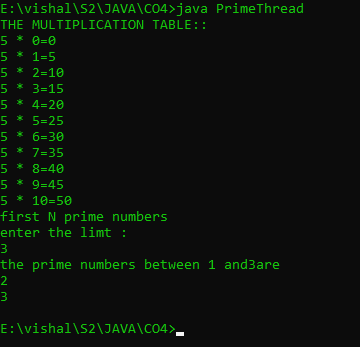
b.start();

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);

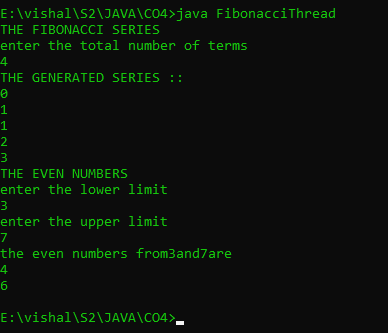
b.start();

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;

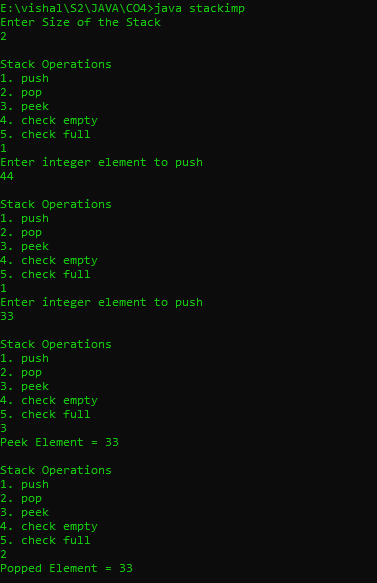
}

} 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: ");

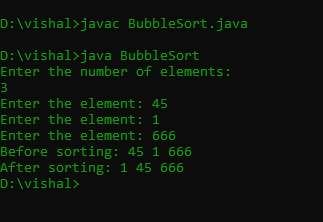
for(int i=0; i<size; i++){

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);

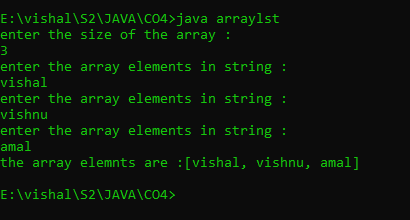
}

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);

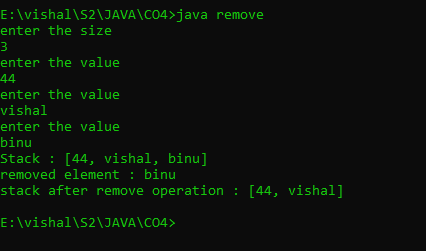
System.out.println("removed element : "+rem\_ele);

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());

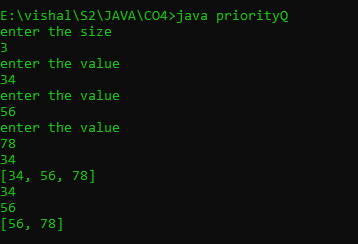
System.out.println(priorQ.peek());

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();

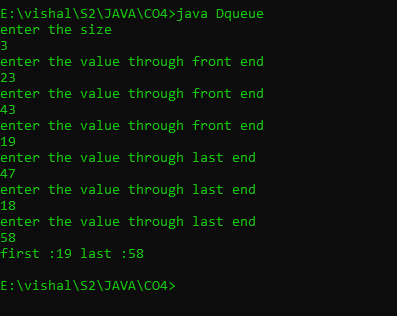
int last = dq.removeLast();

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