**Name: Sruthy chandran**

**Roll No:43**

**Batch:MCA-B**

**Date:31-5-2022**

**OBJECT ORIENTED PROGRAMMING LAB**

**Co4**

**Aim**

1. 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.

**Procedure**

**Circle.java**

package Graphics;

interface AreaInterface {

void Area();

}

public class Circle implements AreaInterface{

double radius;

public Circle(double radius){

this.radius= radius;

}

public void Area() {

double area= 3.14 \* this.radius \* this.radius;

System.out.println("The area of the given circle is : "+area);

}

}

Rectangle.java

package Graphics;

interface AreaInterface {

void Area();

}

public class Circle implements AreaInterface{

double radius;

public Circle(double radius){

this.radius= radius;

}

public void Area() {

double area= 3.14 \* this.radius \* this.radius;

System.out.println("The area of the given circle is : "+area);

}

}

Square.java

package Graphics;

interface AreaInterface {

void Area();

}

public class Square implements AreaInterface{

double side;

public Square(double side){

this.side= side;

}

public void Area() {

double area= this.side \* this.side;

System.out.println("The area of the given square is : "+area);

}

}

Triangle.java

package Graphics;

interface AreaInterface {

void Area();

}

public class Triangle implements AreaInterface{

double length, breadth;

public Triangle(double length, double breadth){

this.length= length;

this.breadth= breadth;

}

public void Area() {

double area= 0.5 \* this.length \* this.breadth;

System.out.println("The area of the given triangle is : "+area);

}

}

Areacirculation.java

import java.util.\*;

import Graphics.\*;

public class AreaCalculation {

public static void main(String[] args) {

Scanner sc= new Scanner(System.in);

int choice,isexit=0;

while(isexit==0){

double length, breadth, side, radius;

System.out.println("\n1. Area of Triangle.\n2. Area of Circle.\n3. Area of Square.\n4. Area of Rectangle.\n5. Exit");

System.out.print("Please enter the operation choice to perform - ");

choice= sc.nextInt();

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

switch(choice){

case 1:{

System.out.print("Enter the length of the triangle : ");

length= sc.nextDouble();

System.out.print("Enter the height of the triangle : ");

breadth= sc.nextDouble();

Triangle triangle = new Triangle(length, breadth);

triangle.Area();

break;

}

case 2:{

System.out.print("Enter the radius of the circle : ");

radius= sc.nextDouble();

Circle cir= new Circle(radius);

cir.Area();

break;

}

case 3:{

System.out.print("Enter the side length of the square : ");

side= sc.nextDouble();

Square square= new Square(side);

square.Area();

break;

}

case 4:{

System.out.print("Enter the length of the rectangle : ");

length= sc.nextDouble();

System.out.print("Enter the breadth of the rectangle : ");

breadth= sc.nextDouble();

Rectangle rec= new Rectangle(length, breadth);

rec.Area();

break;

}

case 5:{

isexit=1;

break;

}

default:{

break;

}

}

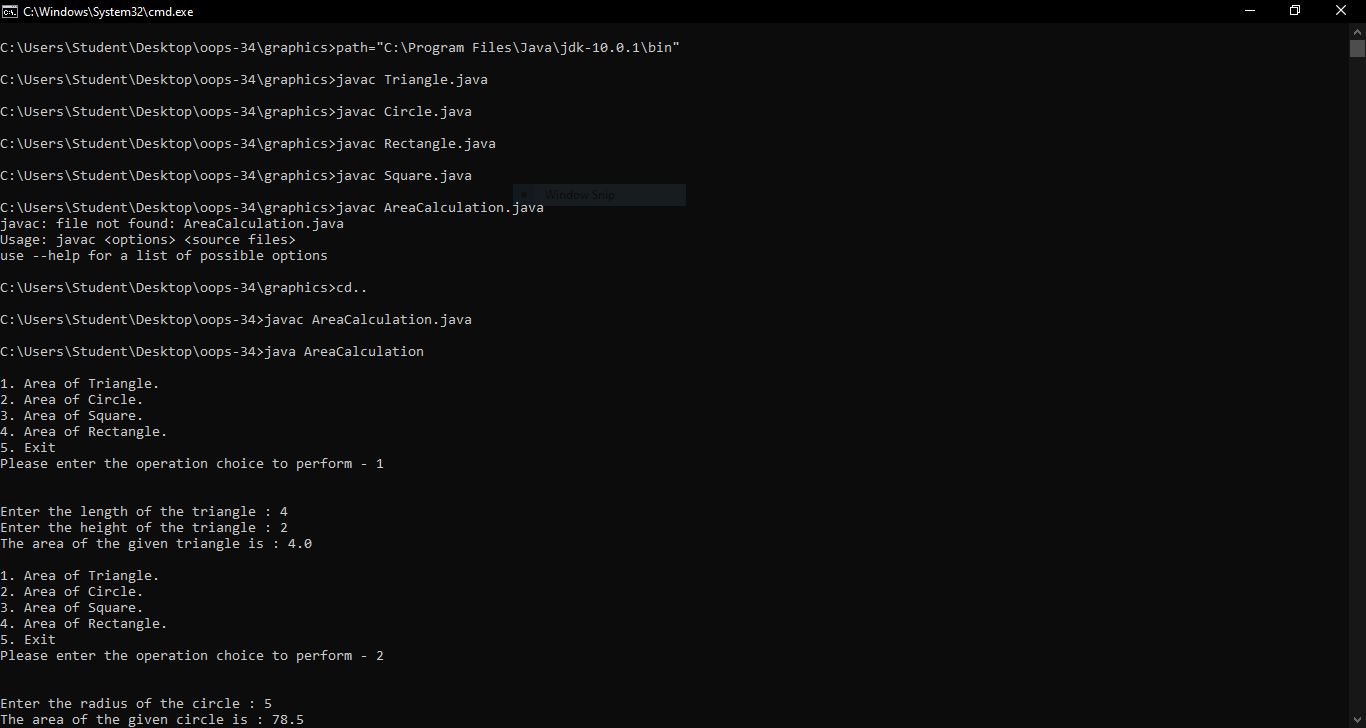
}

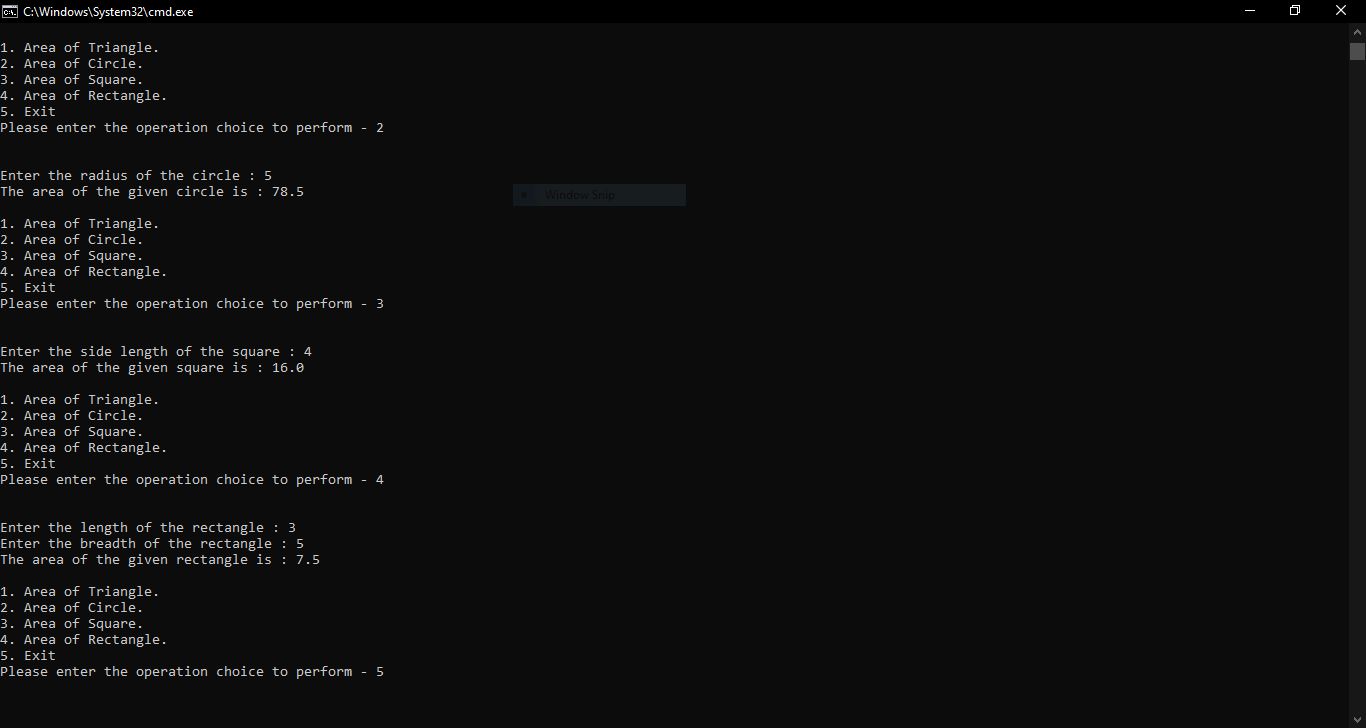
sc.close();

}

}

**Output Screenshot**

****

****

**Aim**

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

**Procedure**

public class CustomExceptionExample {

public static class InvalidUserException extends Exception {

public InvalidUserException() {

super("Invalid username / password provided!");

}

}

public static void main(String[] args) {

String username = "salu";

String password = "pass";

try {

if (username.equals("user") && password.equals("pass")) {

System.out.println("Authenticated successfully!");

} else {

throw new InvalidUserException();

}

} catch (InvalidUserException e) {

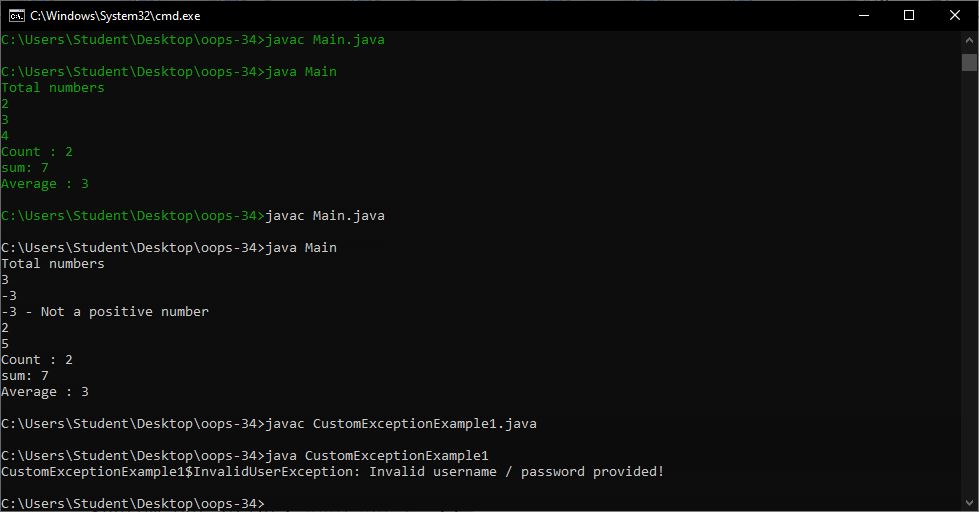
System.out.println(e);

}

}

}

**Output Screenshot**

****

**Aim**

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

**Procedure**

import java.util.\*;

class MyException extends Exception {

public MyException(String value) {

super(value);

}

}

class Main {

public static void main(String args[]) {

int totalNums;

int i;

int temp, count = 0;

int sum = 0;

Scanner sc = new Scanner(System.in);

System.out.println("Total numbers");

totalNums = Integer.parseInt(sc.nextLine());

for (i = 0; i < totalNums; i++) {

try {

temp = Integer.parseInt(sc.nextLine());

if (temp > 0) {

sum += temp;

count += 1;

} else {

throw new MyException(Integer.toString(temp));

}

} catch (MyException ex) {

System.out.print(ex.getMessage());

System.out.println(" - Not a positive number");

}

}

System.out.print("Count : ");

System.out.println(count);

System.out.print("sum: ");

System.out.println(sum);

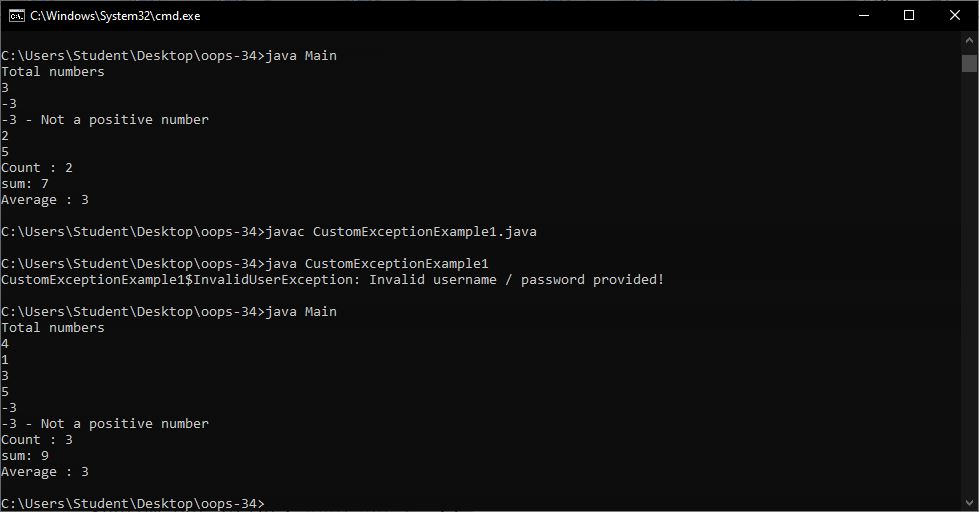
System.out.print("Average : ");

System.out.println(sum / count);

}

}

**Output Screenshot**

**Aim**

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

**Procedure**

import java.util.\*;

class fibonacci implements Runnable {

int l;

fibonacci(int n) {

l = n;

}

public void run() {

int c;

int a = 0, b = 1;

System.out.print(a + " " + b);

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

c = a + b;

System.out.print(" " + c);

a = b;

b = c;

}

}

}

class even implements Runnable {

int l;

even(int n) {

l = n;

}

public void run() {

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

if (i % 2 == 0)

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

}

System.out.println("");

}

}

class My{

public static void main(String args[]) {

Scanner sc = new Scanner(System.in);

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

int l = sc.nextInt();

fibonacci f = new fibonacci(l);

Thread T1 = new Thread(f);

T1.start();

even e = new even(l);

Thread T2 = new Thread(e);

T2.start();

}

}

**Output Screenshot**

**Aim**

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

**Procedure**

class Stack {

private int arr[];

private int top;

private int capacity;

Stack(int size) {

arr = new int[size];

capacity = size;

top = -1;

}

public void push(int x) {

if (isFull()) {

System.out.println("Stack OverFlow");

System.exit(1);

}

System.out.println("Inserting " + x);

arr[++top] = x;

}

public int pop() {

if (isEmpty()) {

System.out.println("STACK EMPTY");

System.exit(1);

}

return arr[top--];

}

public int getSize() {

return top + 1;

}

public Boolean isEmpty() {

return top == -1;

}

public Boolean isFull() {

return top == capacity - 1;

}

public void printStack() {

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

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

}

}

public static void main(String[] args) {

Stack stack = new Stack(5);

stack.push(1);

stack.push(2);

stack.push(3);

System.out.print("Stack: ");

stack.printStack();

stack.pop();

System.out.println("\nAfter popping out");

stack.printStack();

}

}

**Output Screenshot**

**Aim**

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

**Procedure**

import java.util.ArrayList;

import java.util.Collections;

class Mycls {

public static void main(String[] args) {

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

data.add("A");

data.add("B");

data.add("C");

data.add("D");

data.set(1, "BB");

System.out.println(data);

System.out.println(data.get(0));

System.out.println(data.get(1));

data.remove(0);

System.out.println(data);

System.out.println(data.size());

for (String d : data) {

System.out.println(d);

}

Collections.sort(data);

System.out.println(data);

data.clear();

System.out.println(data);

}

}

**Output Screenshot**

**Aim**

1. Program to demonstrate the creation of queue object using the Priority Queue class.

**Procedure**

import java.util.\*;

class Collection\_Framework\_Queue {

public static void main(String args[]) {

Queue<Integer> q = new PriorityQueue<Integer>(new Comp());

int ch;

Scanner sc = new Scanner(System.in);

do {

System.out.println("\n1.ADD\n2.PEEK\n3.POLL or REMOVE\n4.DISPLAY\n5.EXIT");

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

ch = sc.nextInt();

switch (ch) {

case 1:

System.out.println("\n\tEnter Integer : ");

int n1 = sc.nextInt();

q.add(n1);

System.out.println("\n\tADDED SUCCESSFULLY ! ! ! ");

break;

case 2:

if (q.isEmpty()) {

System.out.print("\n\tQueue Empty ! ! !");

} else {

System.out.print("\n\tPeeked element is " + q.peek());

}

break;

case 3:

if (!q.isEmpty()) {

System.out.print("\n\tRemoved element is " + q.poll());

} else {

System.out.print("\n\tQueue Empty ! ! !");

}

break;

case 4:

if (!q.isEmpty()) {

System.out.print("\nSize of queue : " + q.size());

System.out.print("\nQueue elements : " + q);

System.out.println("\nQueue elements are");

for (int i : q) {

System.out.println(i);

}

} else {

System.out.print("\n\tQueue Empty ! ! !");

}

break;

case 5:

break;

default:

System.out.println("\n\tPlease enter valid choice ! ! ! ");

}

} while (ch != 5);

}

}

class Comp implements Comparator<Integer> {

public int compare(Integer a, Integer b) {

return a % 10 > b % 10 ? 1 : -1;

}

}

**Output Screenshot**





**Aim**

1. Program to demonstrate the addition and deletion of elements in dequeue.

**Procedure**

**Output Screenshot**

**Aim**

1. Write a Java program to compare two hash set.

**Procedure**

**Output Screenshot**

**Aim**

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

**Procedure**

**Output Screenshot**