**JAVA ALL PROGRAMS FOR LAB EXAM**

1. **TERMWORK 1(a)(STUDENT MARKS)**

**2) It is required to store and analyze data for 6 students scores in six different subjects.**

**Demonstrate how you would store the data in a two dimensional matrix and do the following**

**1. Write a function to Find for a given student, the subject in which, the score is maximum.**

**2. Write a function to Find the average marks scored by each student.**

**3. Write a function to Find the total marks scored by each student.**

**4. Write a function to find the total scores by all students in each subject.**

**Assume – row index 0 – Student 1, 1 – Student, 2 – Student 3- Student etc**

**Col index 0 – ‘ Maths’, 1 – ‘DS’, 2-‘OOPJ’, 3- ‘Python’, 4 – ‘SED’, 5 – ‘ESC’, 6 – ‘C++’**

**Demonstrate the working of the program with appropriate values for each student and marks scored.**

import java.util.Scanner;

class student\_marks{

    int[][]marks=new int[6][6];

    void read(){

        Scanner s=new Scanner(System.in);

        for(int i=0;i<6;i++){

            System.out.println("enter the marks of student:"+(i+1));

            for(int j=0;j<6;j++){

                marks[i][j]=s.nextInt();

            }

        }

    }

    void display(){

        for(int i=0;i<6;i++){

            System.out.println("marks of student"+(i+1)+"is:");

            for(int j=0;j<6;j++){

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

            }

            System.out.println();

        }

    }

    void total\_marks(){

        for(int i=0;i<6;i++){

            int total=0;

            for(int j=0;j<6;j++){

                total+=marks[i][j];

            }

            System.out.println("total marks of student"+(i+1)+"is:"+(total));

        }

    }

    void avg\_marks(){

        for(int i=0;i<6;i++){

            int total=0;

            for(int j=0;j<6;j++){

                total+=marks[i][j];

            }

            System.out.println("Average marks of student:"+(i+1)+"is:"+(total/6));

        }

    }

    void sub\_marks(){

        for(int j=0;j<6;j++){

            int total=0;

            for(int i=0;i<6;i++){

                total+=marks[i][j];

            }

            System.out.println("total marks scored in subject"+(j+1)+"is:"+total);

        }

    }

    void max\_marks(){

        Scanner s=new Scanner(System.in);

        int max=0,subno=0,studno=0,i,j;

        System.out.println("ener the student number(0-5):");

        studno=s.nextInt();

        for(i=studno,j=0;j<6;j++){

            if(marks[i][j]>max){

                max=marks[i][j];

                subno=j;

            }

        }

        System.out.println("maximum marks scored by student"+(i+1)+"is:"+(max)+"in subject"+(subno+1));

    }

        }

public class student\_marks\_Demo{

            public static void main(String args[]){

                student\_marks sm=new student\_marks();

sm.read();

                sm.display();

                sm.avg\_marks();

                sm.sub\_marks();

                sm.max\_marks();

            }

        }

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

1. **TERM WORK 2( 3 TESTS STUDENT MARKS)**

**2) Define a class to represent the student details such as name, roll number, marks obtained in three internal assessment tests.**

**Identify type and declare the instance variables**

**Identify and develop the constructors to initialize the instance variables**

**Write a method computeAverage() to compute the average as the average of best two marks**

**Write a method to display the student details**

**Write the corresponding Driver class to instantiate an array of student objects and demonstrate the working of the application by invoking appropriate methods.**

import java.util.Scanner;

class Student{

int roll\_no;

String name;

double ia1,ia2,ia3,average;

void computeAverage(){

if(ia1<ia2 && ia1<ia3){

average=(ia2+ia3)/2;

System.out.println("the average of best two marks is="+average);

}

else if(ia2<ia1 &&ia2<ia3){

average=(ia1+ia3)/2.0;

System.out.println("the average of best two marks is="+average);

}

else{

average=(ia1+ia2)/2.0;

System.out.println("the average of best two marks is="+average);

}

}

void displayData(){

System.out.println(" Name :" +name+"\n Roll number :" +roll\_no+ "\n marks in ia1: " +ia1+"\n marks in ia2 :" +ia2+"\n marks in ia3: "+ia3);

}

void set(){

Scanner s=new Scanner(System.in);

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

name=s.nextLine();

System.out.println("enter the roll no:");

roll\_no=s.nextInt();

System.out.println("enter the ia1 marks:");

ia1=s.nextDouble();

System.out.println("enter the ia2 marks :");

ia2=s.nextDouble();

System.out.println("enter the ia3 marks:");

ia3=s.nextDouble();

}

}

public class Student\_demo {

public static void main(String args[]){

Student s1=new Student();

s1.set();

s1.displayData();

s1.computeAverage();

Student s2=new Student();

s2.set();

s2.displayData();

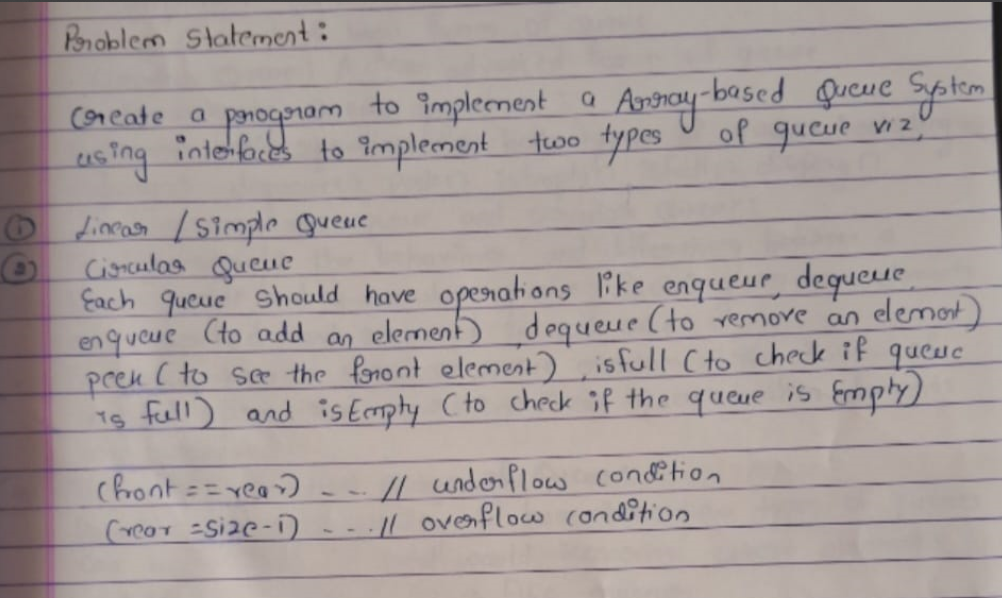
s2.computeAverage();

}

}

================================================================================================================

1. **TERM WORK 4☹ENQUEUE AND DEQUEUE)**

****

interface QueueD {

void enqueue(int element);

int dequeue();

int peek();

boolean isEmpty();

boolean isFull();

void display();

}

class Linearq implements QueueD{

private int[] queue;

private int front, rear, size;

public Linearq(int capacity) {

queue = new int[capacity];

front = rear = -1;

size = capacity;

}

public void enqueue(int element) {

if (isFull()) {

System.out.println("Queue is full");

return;

}

if (front == -1) {

front = 0;

}

queue[++rear] = element;

}

public int dequeue() {

if (isEmpty()) {

System.out.println("Queue is empty");

return -1;

}

int element = queue[front];

if (front == rear) {

front = rear = -1;

} else {

front++;

}

return element;

}

public int peek() {

if (isEmpty()) {

System.out.println("Queue is empty");

return -1;

}

return queue[front];

}

public boolean isEmpty() {

return front == -1;

}

public boolean isFull() {

return rear == size - 1;

}

public void display() {

if (isEmpty()) {

System.out.println("Queue is empty");

return;

}

System.out.print("Queue elements: ");

for (int i = front; i <= rear; i++) {

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

}

System.out.println();

}

}

class Circularq implements QueueD {

private int[] queue;

private int front, rear, size, capacity;

public Circularq(int capacity) {

queue = new int[capacity];

front = rear = -1;

this.capacity = capacity;

size = 0;

}

public void enqueue(int element) {

if (isFull()) {

System.out.println("Queue is full");

return;

}

if (front == -1) {

front = 0;

}

rear = (rear + 1) % capacity;

queue[rear] = element;

size++;

}

public int dequeue() {

if (isEmpty()) {

System.out.println("Queue is empty");

return -1;

}

int element = queue[front];

front = (front + 1) % capacity;

size--;

if (size == 0) {

front = rear = -1;

}

return element;

}

public int peek() {

if (isEmpty()) {

System.out.println("Queue is empty");

return -1;

}

return queue[front];

}

public boolean isEmpty() {

return size == 0;

}

public boolean isFull() {

return size == capacity;

}

public void display() {

if (isEmpty()) {

System.out.println("Queue is empty");

return;

}

System.out.print("Queue elements: ");

int i = front;

do {

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

i = (i + 1) % capacity;

} while (i != (rear + 1) % capacity);

System.out.println();

}

}

public class QueueTest {

public static void main(String[] args) {

QueueD linearq = new Linearq(5);

QueueD Circularq = new Circularq(5);

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

linearq.enqueue(10);

linearq.enqueue(20);

linearq.enqueue(30);

linearq.display();

System.out.println("Front element: " + linearq.peek());

System.out.println("Is full: " + linearq.isFull());

System.out.println("Dequeue: " + linearq.dequeue());

linearq.display();

System.out.println("Front element: " + linearq.peek());

System.out.println("Is empty: " + linearq.isEmpty());

System.out.println("\nCircular Queue:");

Circularq.enqueue(10);

Circularq.enqueue(20);

Circularq.enqueue(30);

Circularq.enqueue(40);

Circularq.enqueue(50);

Circularq.enqueue(60);

Circularq.display();

System.out.println("Is full: " + Circularq.isFull());

System.out.println("Front element: " + Circularq.peek());

System.out.println("Dequeue: " + Circularq.dequeue());

}

}

================================================================================================================

**4.TERM WORK ☹COMPLEX NUMBERS ADDITION)**

**Write a program to demonstrate the implementation of Parameterized Methods and**

**Constructors**

**Write a Java program to represent a Complex number. Include appropriate data members and**

**the following member functions:**

**a default- constructor to initialize the complex number to zero.**

**A parameterized const. to initialize the complex number to user specified values**

**Add two complex nos. and return the result**

**Add an integer value to complex no. and return the result**

**Display a complex no.**

**Create complex nos. to demonstrate the working of complex class**

import java.util.Scanner;

class Complex {

double real, img;

// Constructor for default complex number

Complex() {

real = 0;

img = 0;

}

// Constructor for initializing complex number with real and imaginary parts

Complex(double r, double i) {

real = r;

img = i;

}

// Method to add two complex numbers

Complex add(Complex c1, Complex c2) {

Complex result = new Complex();

result.real = c1.real + c2.real;

result.img = c1.img + c2.img;

return result;

}

// Method to add complex number with an integer

Complex add(Complex c1, int x) {

Complex result = new Complex();

result.real = c1.real + x;

result.img = c1.img;

return result;

}

// Method to display the complex number

void display() {

if (img >= 0) {

System.out.println("Complex No: " + real + " + " + img + "i");

} else {

System.out.println("Complex No: " + real + " - " + (-img) + "i");

}

}

}

public class ComplexDemo {

public static void main(String[] args) {

Complex c1 = new Complex(); // Default complex number

Complex c2 = new Complex(4, 6); // Complex number with values (4, 6)

Complex c3 = new Complex(6, 8); // Complex number with values (6, 8)

System.out.println("Displaying complex numbers one by one:");

System.out.println("Initial Complex c1:");

c1.display(); // Display default complex number

System.out.println("Complex c2:");

c2.display(); // Display c2

System.out.println("Complex c3:");

c3.display(); // Display c3

c1 = c1.add(c2, c3); // Add c2 and c3 and store in c1

System.out.println("After adding c2 and c3, new c1:");

c1.display(); // Display the result of addition

c1 = c1.add(c2, 4); // Add c2 and integer 4 and store in c1

System.out.println("After adding c2 and 4, new c1:");

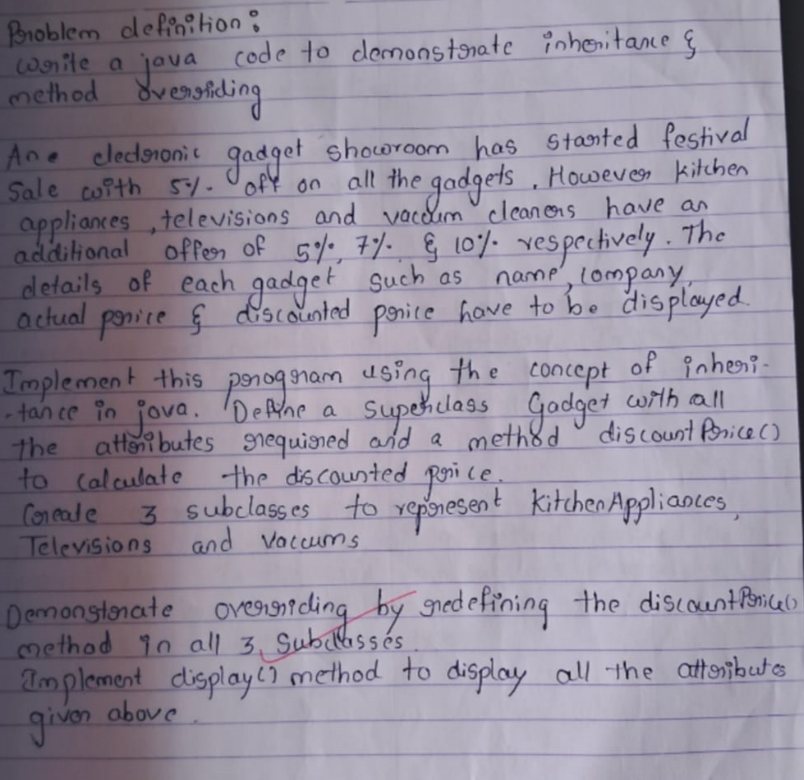
c1.display(); // Display the result

}

}

================================================================================================================

1. **TERM WORK:(GADGETS SAME AS BOOK PROG)**

****

class Gadget {

String name;

String company;

double price;

Gadget(String name, String company, double price) {

this.name = name;

this.company = company;

this.price = price;

}

double discountPrice() {

// 5% basic discount

return price - (price \* 0.05);

}

void display() {

System.out.println("Gadget: " + name);

System.out.println("Company: " + company);

System.out.println("Original Price: " + price);

System.out.println("Discounted Price: " + discountPrice());

}

}

class KitchenAppliances extends Gadget {

KitchenAppliances(String name, String company, double price) {

super(name, company, price);

}

double discountPrice() {

return price - (price \* 0.10); // 5% + 5% additional discount

}

}

class Televisions extends Gadget {

Televisions(String name, String company, double price) {

super(name, company, price);

}

double discountPrice() {

return price - (price \* 0.12); // 5% + 7% additional discount

}

}

class Vacuums extends Gadget {

Vacuums(String name, String company, double price) {

super(name, company, price);

}

double discountPrice() {

return price - (price \* 0.15); // 5% + 10% additional discount

}

}

public class GadgetStore {

public static void main(String[] args) {

Gadget kitchenAppliance = new KitchenAppliances("Mixer", "Philips", 5000);

Gadget television = new Televisions("LED TV", "Sony", 40000);

Gadget vacuum = new Vacuums("Vacuum Cleaner", "Dyson", 15000);

kitchenAppliance.display();

System.out.println();

television.display();

System.out.println();

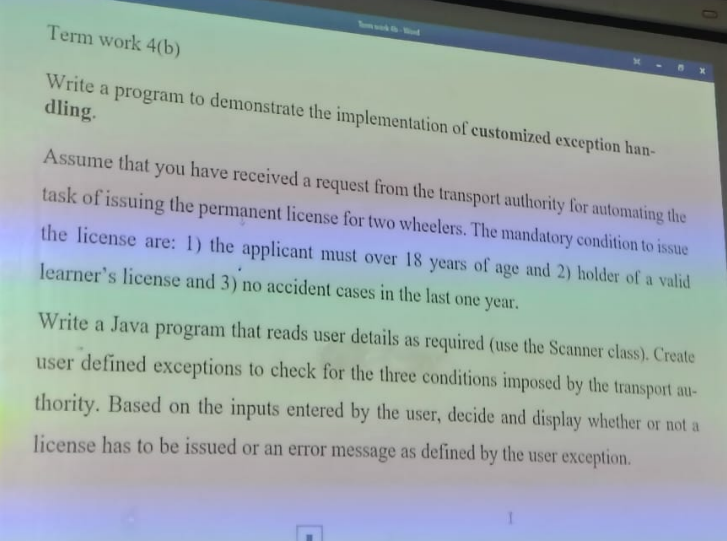
vacuum.display();

}

}

================================================================================================================

1. **TERM WORK ☹EXCEPTION HANDLING)**



import java.util.Scanner;

// Custom exception for age

class AgeException extends Exception {

AgeException(String msg) {

super(msg);

}

}

// Custom exception for learner's license

class LearnersLicenseException extends Exception {

LearnersLicenseException(String msg) {

super(msg);

}

}

// Custom exception for accidents

class AccidentsException extends Exception {

AccidentsException(String msg) {

super(msg);

}

}

public class TW4b {

// Method to process the application

static void processApplication(int age, boolean validLL, boolean noAccidents)

throws AgeException, LearnersLicenseException, AccidentsException {

if (age < 18) {

throw new AgeException("Invalid age: Age must be above 18 years.");

}

if (!validLL) {

throw new LearnersLicenseException("Invalid learner's license.");

}

if (!noAccidents) {

throw new AccidentsException("Invalid: An accident occurred in the last year.");

}

System.out.println("Application approved: You are eligible for a license.");

}

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

try {

// Input from the user

System.out.println("Enter user's name:");

String name = s.nextLine();

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

int age = s.nextInt();

System.out.println("Enter 'true' if you have a valid learner's license, otherwise 'false':");

boolean validLL = s.nextBoolean();

System.out.println("Enter 'true' if no accidents occurred in the last year, otherwise 'false':");

boolean noAccidents = s.nextBoolean();

// Process the application

processApplication(age, validLL, noAccidents);

}

catch (AgeException | LearnersLicenseException | AccidentsException ae) {

System.out.println("User cannot be issued a license.");

System.out.println(ae.getMessage());

} catch (Exception e) {

System.out.println("An unexpected error occurred.");

} finally {

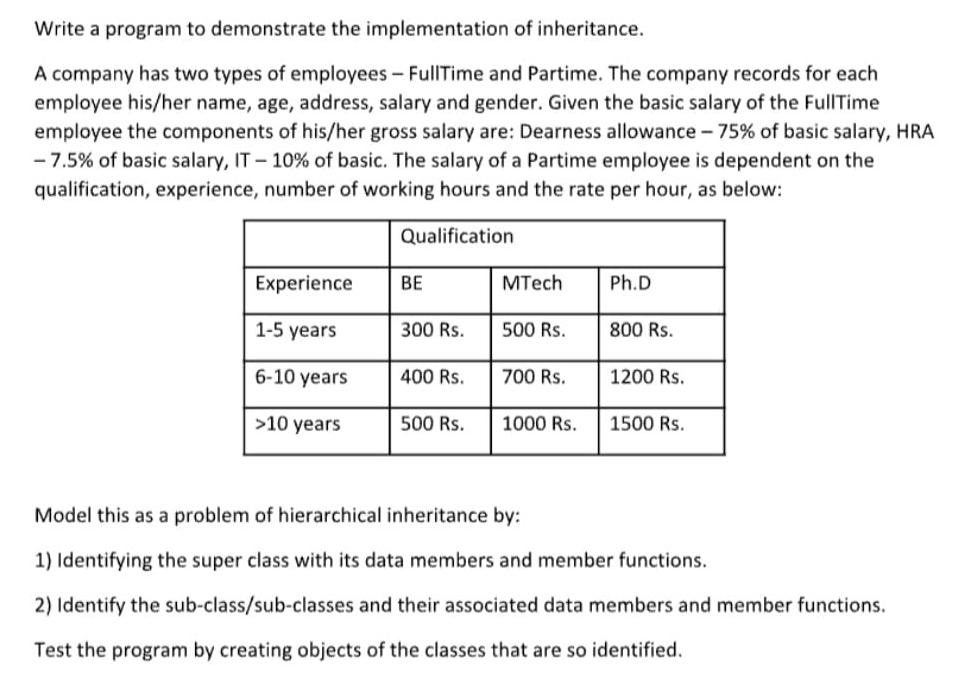
s.close(); // Close the scanner

}

}

}

1. **TERM WORK☹EMPLOYEE GROSS SALARY**

****

package TWA;

// Base class Employee

public class Employee {

protected String ename, address, gender;

protected int age;

protected double salary;

// Default constructor

Employee() {

ename = "";

address = "";

gender = "";

age = 0;

salary = 0.0;

}

// Parameterized constructor

Employee(String ename, String address, String gender, int age, double salary) {

this.ename = ename;

this.address = address;

this.gender = gender;

this.age = age;

this.salary = salary;

}

// Display method

void display() {

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

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

System.out.println("Gender: " + gender);

System.out.println("Age: " + age);

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

}

}

// Derived class FullTime

class FullTime extends Employee {

private double DA, HRA, IT, grossSalary;

// Parameterized constructor

FullTime(String ename, String address, String gender, int age, double salary) {

super(ename, address, gender, age, salary);

}

// Compute salary method

void computeSalary() {

DA = 0.75 \* salary;

HRA = 0.075 \* salary;

IT = 0.1 \* salary;

grossSalary = salary + DA + HRA - IT;

}

// Override display method

@Override

void display() {

super.display();

System.out.println("Gross Salary: " + grossSalary);

}

}

// Derived class PartTime

class PartTime extends Employee {

private double workingHours, ratePerHour, totalSalary;

// Parameterized constructor

PartTime(String ename, String address, String gender, int age, double salary, double workingHours, double ratePerHour) {

super(ename, address, gender, age, salary);

this.workingHours = workingHours;

this.ratePerHour = ratePerHour;

}

// Compute salary method

void computeSalary() {

totalSalary = workingHours \* ratePerHour;

}

// Override display method

@Override

void display() {

super.display();

System.out.println("Working Hours: " + workingHours);

System.out.println("Rate Per Hour: " + ratePerHour);

System.out.println("Total Salary: " + totalSalary);

}

}

// Main class

public class RAPDemo {

public static void main(String[] args) {

// Create objects of FullTime and PartTime

FullTime ftEmployee = new FullTime("Samantha", "NY", "Female", 30, 50000);

PartTime ptEmployee = new PartTime("John", "LA", "Male", 25, 0, 20, 50);

// Compute salaries

ftEmployee.computeSalary();

ptEmployee.computeSalary();

// Display details

System.out.println("Full-Time Employee Details:");

ftEmployee.display();

System.out.println("\nPart-Time Employee Details:");

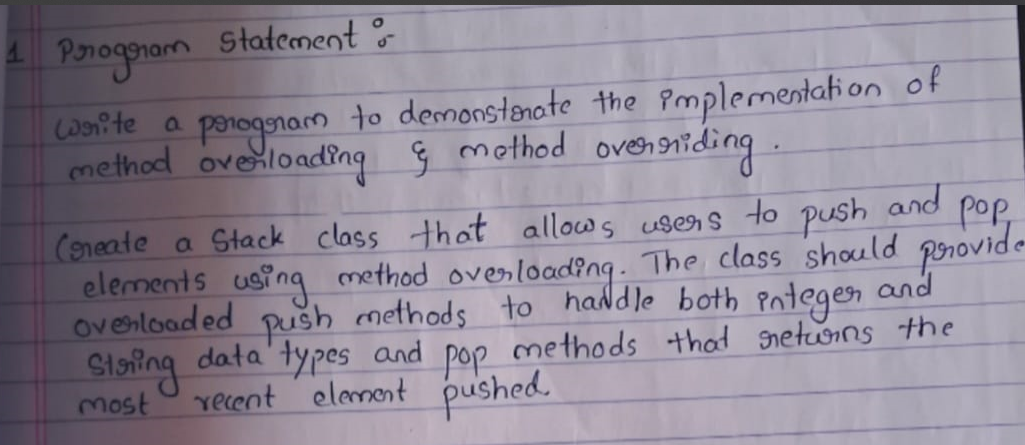
ptEmployee.display();

}

}

================================================================================================================

1. **TERM WORK☹PUSH AND POP)**

****

// Stack Class

class Stack {

int[] istk; // Integer stack

String[] sstk; // String stack

int topInt, topStr; // Top pointers for integer and string stacks

int maxSize; // Maximum size of the stack

// Constructor to initialize stacks

Stack(int size) {

maxSize = size;

istk = new int[maxSize];

sstk = new String[maxSize];

topInt = -1;

topStr = -1;

}

// Push integer value onto the integer stack

void push(int value) {

if (topInt == maxSize - 1) {

System.out.println("Integer stack is full!");

} else {

istk[++topInt] = value;

System.out.println("Pushed integer: " + value);

}

}

// Push string value onto the string stack

void push(String value) {

if (topStr == maxSize - 1) {

System.out.println("String stack is full!");

} else {

sstk[++topStr] = value;

System.out.println("Pushed string: " + value);

}

}

// Pop integer value from the integer stack

int popInt() {

if (topInt == -1) {

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

return -1;

} else {

System.out.println("Popped integer: " + istk[topInt]);

return istk[topInt--];

}

}

// Pop string value from the string stack

String popString() {

if (topStr == -1) {

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

return null;

} else {

System.out.println("Popped string: " + sstk[topStr]);

return sstk[topStr--];

}

}

// Peek integer value from the integer stack

int peekInt() {

if (topInt == -1) {

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

return -1;

} else {

System.out.println("Peeked integer: " + istk[topInt]);

return istk[topInt];

}

}

// Peek string value from the string stack

String peekString() {

if (topStr == -1) {

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

return null;

} else {

System.out.println("Peeked string: " + sstk[topStr]);

return sstk[topStr];

}

}

}

// Main Class

public class StackDemo {

public static void main(String[] args) {

Stack intStack = new Stack(5); // Integer stack

Stack stringStack = new Stack(5); // String stack

// Push integers

intStack.push(10);

intStack.push(20);

intStack.push(30);

intStack.push(40);

intStack.push(50);

// Push strings

stringStack.push("Alice");

stringStack.push("Bob");

stringStack.push("Charlie");

stringStack.push("Daisy");

stringStack.push("Eve");

// Pop integers

intStack.popInt();

intStack.popInt();

// Pop strings

stringStack.popString();

stringStack.popString();

// Peek integers and strings

intStack.peekInt();

stringStack.peekString();

}

}

================================================================================================================

1. **TERMWORK ☹UPPER CASE AND LOWER CASE**

**Write a program to demonstrate String Handling**

**Develop a Java program that takes a sentence as input and performs the following**

**operations using the String class methods:**

**Convert the entire sentence to uppercase.**

**Find and display the position of the first occurrence of the word “java” in the**

**sentence.**

**Extract a substring from the 5th to the 10th character of the sentence.**

**Replace all occurrences of the letter ‘o’ with ‘a’**

public class StringOperations {

public static void main(String[] args) {

String sentence = "Java is often used for building cross-platform applications";

// Convert sentence to uppercase

String upperCaseSentence = sentence.toUpperCase();

System.out.println("Uppercase sentence: " + upperCaseSentence);

// Find the first occurrence of the word "Java"

int javaPosition = sentence.indexOf("Java");

if (javaPosition != -1) {

System.out.println("Position of 'Java': " + javaPosition);

} else {

System.out.println("'Java' not found in the sentence.");

}

// Extract a substring (4th to 10th characters)

String substring = sentence.substring(4, 10);

System.out.println("Substring from 5th to 10th character: " + substring);

// Replace 'o' with 'a'

String replacedSentence = sentence.replace('o', 'a');

System.out.println("Sentence after replacing 'o' with 'a': " + replacedSentence);

}

}

================================================================================================================

1. **TERM WORK( LINEAR SEARCH)**

import java.util.Scanner;

class LinearSearch{

boolean lsearch(int arr[], int key){

for (int element:arr){

if(element==key){

return true;

}

}

return false;

}

boolean lsearch(double arr[], double key){

for (double element:arr){

if(element==key){

return true;

}

}

return false;

}

boolean lsearch(char arr[], char key){

for (char element:arr){

if(element==key){

return true;

}

}

return false;

}

}

public class LinearSearchDemo {

public static void main(String[] args){

LinearSearch LS=new LinearSearch();

int iarr[]={1,2,3,4,5};

double darr[]={1.0,2.0,3.0,4.0};

char carr[]={'a','b','c','d'};

Scanner sc=new Scanner(System.in);

System.out.println("Enter the integer key:");

int ikey=sc.nextInt();

System.out.println("Enter the double key:");

double dkey=sc.nextDouble();

System.out.println("Enter the character key:");

char ckey=sc.next().charAt(0);

boolean ires=LS.lsearch(iarr, ikey);

if (ires==true){

System.out.println("Key element "+ikey+" is found in the integer array.");

}

else{

System.out.println("Key element "+ikey+" is not found in the integer array.");

}

boolean dres=LS.lsearch(darr, dkey);

if (dres==true){

System.out.println("Key element "+dkey+" is found in the double array.");

}

else{

System.out.println("Key element "+dkey+" is not found in the double array.");

}

boolean cres=LS.lsearch(carr, ckey);

if (cres==true){

System.out.println("Key element "+ckey+" is found in the char array.");

}

else{

System.out.println("Key element "+ckey+" is not found in the char array.");

}

}

}

======================================================================================================

1. **TERM WORK☹CAR SALES)**

**1.1) It is required to store and analyze data about 6 car manufacturer’s sales data in all the 12 months of a year. Demonstrate how you would store the data in a two dimensional matrix and do the following**

**Write a function to Find for a given car manufacturer, the month in which, maximum no. of cars are sold.**

**Write a function to Find the average number of cars sold for each car manufacturer**

**Write a function to Find the total number of cars sold for each car manufacturer.**

**Write a function to find standard deviation for a given car manufacturer**

**Assume – row index 0 - ‘Maruti Suzuki’, 1 – ‘Hundai’ 2 – ‘Tata Motors’ 3-‘KIA’ 4 – ‘BMW’ 5 – ‘Renault’**

**Col index 0 –‘Jan’, 1-‘Feb’………………………………….11 –‘Dec’**

**Demonstrate the working of the program with appropriate values for each car manufacturer and the months.**

import java.util.Scanner;

class MyCarSales{

int sales[][]= new int[6][12];

void read\_sales(){

Scanner s=new Scanner(System.in);

System.out.println("Enter the sales for cars: ");

for(int i=0;i<6;i++){

for(int j=0;j<12;j++){

sales[i][j]=s.nextInt();

}

}

}

void display\_sales(){

System.out.println("The car sales for all mfgrs are as below..");

for(int i=0;i<6;i++){

for(int j=0;j<12;j++){

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

}

System.out.println();

}

}

void avg\_Sales(){

int total;

for(int i=0;i<6;i++){

total=0;

for(int j=0;j<12;j++){

total=total+sales[i][j];

}

System.out.println("Average sales for manufacturer "+(i+1)+ " is: "+(total/12));

}

}

void total\_Sales(){

int total;

for(int i=0;i<6;i++){

total=0;

for(int j=0;j<12;j++){

total=total+sales[i][j];

}

System.out.println("Total sales for manufacturer "+(i+1)+ " is: "+total);

}

}

void month\_Sales(){

int msales;

for(int j=0;j<12;j++){

msales=0;

for(int i=0;i<6;i++){

msales=msales+sales[i][j];

}

System.out.println("Sales in month " +(j+1)+ " are:"+msales);

}

}

void max\_sales(){

int max=0,i,j,month=0;

Scanner sc=new Scanner(System.in);

System.out.println("Enter the car for which u want to find the max sales:");

int mg= sc.nextInt();

for( i=mg,j=0; j<12;j++){

if(max<sales[i][j]){

max=sales[i][j];

month=j;

}

}

System.out.println("Max. Sales:" +max+ " in the month "+(month+1));

}

}

public class Car\_Sales\_Demo {

public static void main(String args[]){

MyCarSales MS = new MyCarSales();

MS. read\_sales();

System.out.println();

MS.display\_sales();

System.out.println();

MS. avg\_Sales();

System.out.println();

MS.total\_Sales();

System.out.println();

MS.month\_Sales();

System.out.println();

MS.max\_sales();

}

}

1. **TERM WORK😊TRIANGLE PROGRAM**

**2.1) Design a class by name myTriangle to model a triangle geometrical object with three sides. Include functions to:**

**Initialize the three sides of triangle.**

**Determine the type of triangle represented by the three sides (Equilateral/ Isosceles/ Scalene triangle).**

**Compute and return the area of the triangle.**

**Note:**

**When three sides are given we use the following formula:**

**s=(a+b+c)/2;**

**area=sqrt(s\*(s-a)(s-b)(s-c));**

import java.util.Scanner;

class init{

double s1,s2,s3;

void init(){

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

Scanner s=new Scanner(System.in);

s1=s.nextDouble();

s2=s.nextDouble();

s3=s.nextDouble();

}

void type(){

if((s1==s2)&&(s2==s3)&&(s3==s1))

System.out.println("Equilatreal");

else if((s1==s2)||(s2==s3)||(s3==s1))

System.out.println("Isoceles");

else

System.out.println("Scalene");

}

void Area(){

double area;

System.out.println("s1="+s1);

System.out.println("s2="+s2);

System.out.println("s3="+s3);

if((s1+s2>s3)||((s2+s3>s1)||(s3+s1>s2)))

{

double S= (s1+s2+s3)/2;

area=Math.sqrt(S\*(S-s1)\*(S-s2)\*(S-s3));

if(area>0){

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

}

else

System.out.println("Area cannot be calculated");

}

}

}

public class TriangleDemo {

public static void main(String [] args){

init t1=new init();

t1.init();

t1.type();

t1.Area();

init t2=new init();

t2.init();

t2.type();

t2.Area();

init t3=new init();

t3.init();

t3.type();

t3.Area();

}

}

1. **TERM WORK☹BOOK PROGRAM)**

**Define a class Book with the following attributes: Title, Author, Publisher and Price. Initialize the data members using parameterised constructor. Implement a method discountPrice() which calculates discount price with 10% discount rate on price of the books.**

**Demonstrate inheritance by creating 3 subclasses for the categories of the books KidsStory, Scientific and GeneralKnowledge. Demonstrate overriding by redefining the discountPrice() method in each subclass where the discount rate is 15%, 20% and 25% respectively.**

**Implement the display() method to display all the attribute values along with the original price and the discounted price of the book.**

class BOOK {

String Title;

String Author;

String Publisher;

double Price;

double dprice;

// Constructor for BOOK

BOOK(String T, String A, String P, double Pr) {

this.Title = T;

this.Author = A;

this.Publisher = P;

this.Price = Pr;

}

// Method to calculate discounted price

void discountPrice() {

double dp = 0.1 \* Price; // 10% discount

dprice = Price - dp;

}

// Method to display details

void Display() {

discountPrice(); // Calculate discount before displaying

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

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

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

System.out.println("Original Price: " + Price);

System.out.println("Discounted Price: " + dprice);

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

}

}

// Subclass KidsStory

class KidsStory extends BOOK {

KidsStory(String T, String A, String P, double Pr) {

super(T, A, P, Pr);

}

@Override

void discountPrice() {

double dp = 0.15 \* Price; // 15% discount

dprice = Price - dp;

}

}

// Subclass Scientific

class Scientific extends BOOK {

Scientific(String T, String A, String P, double Pr) {

super(T, A, P, Pr);

}

@Override

void discountPrice() {

double dp = 0.2 \* Price; // 20% discount

dprice = Price - dp;

}

}

// Subclass GK

class GK extends BOOK {

GK(String T, String A, String P, double Pr) {

super(T, A, P, Pr);

}

@Override

void discountPrice() {

double dp = 0.25 \* Price; // 25% discount

dprice = Price - dp;

}

}

// Main Class

public class BookDemo {

public static void main(String[] args) {

// Creating objects of different categories

BOOK b = new BOOK("Data Structure Using C", "Sumitra Arora", "Thakur Publications Lucknow", 2000);

BOOK kb = new KidsStory("Fairy Tales", "Balachandran", "Dream Books", 1500);

BOOK sf = new Scientific("The Earth", "John Smith", "SciTech Publishers", 3000);

BOOK gk = new GK("General Knowledge", "Quiz Master", "Knowledge Press", 1000);

// Displaying details

b.Display();

kb.Display();

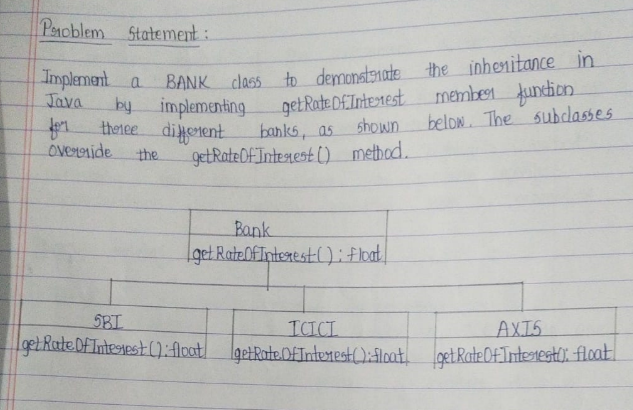
sf.Display();

gk.Display();

}

}

1. **TERM WORK(bank)**



class Bank {

String name;

String IFSC;

String Address;

double ROI;

float getRateOfInterest() {

return 6.5f;

}

// Default constructor

Bank() {

this.name = "Unknown";

this.IFSC = "Unknown";

this.Address = "Unknown";

this.ROI = 0.0;

}

// Parameterized constructor

Bank(String name, String IFSC, String Address) {

this.name = name;

this.IFSC = IFSC;

this.Address = Address;

this.ROI = 6.5;

}

void display() {

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

System.out.println("IFSC Code: " + IFSC);

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

System.out.println("Rate of Interest: " + ROI);

}

}

class SBI extends Bank {

@Override

float getRateOfInterest() {

return 7.0f;

}

// Constructor

SBI(String name, String IFSC, String Address) {

super(name, IFSC, Address);

this.ROI = 7.0;

}

}

class AXIS extends Bank {

@Override

float getRateOfInterest() {

return 7.5f;

}

// Constructor

AXIS(String name, String IFSC, String Address) {

super(name, IFSC, Address);

this.ROI = 7.5;

}

}

class ICICI extends Bank {

@Override

float getRateOfInterest() {

return 8.0f;

}

// Constructor

ICICI(String name, String IFSC, String Address) {

super(name, IFSC, Address);

this.ROI = 8.0;

}

}

public class BankDemo {

public static void main(String[] args) {

// Bank object

Bank bank = new Bank();

bank.getRateOfInterest();

bank.display();

// SBI object

SBI sbi = new SBI("SBI", "SBI00123", "Belagavi");

sbi.getRateOfInterest();

sbi.display();

// AXIS object

AXIS axis = new AXIS("AXIS", "AXI500123", "Mysuru");

axis.getRateOfInterest();

axis.display();

// ICICI object

ICICI icici = new ICICI("ICICI", "ICICI00123", "Bengaluru");

icici.getRateOfInterest();

icici.display();

}

}

1. **TERM WORK**

Write a menu-driven Java Program to create a HashMap to store key-value pairs of login credentials. The menu options to be provided are for: adding a key-value pair, retrieve the "value" for a given "key" (first check if the specified key is present), retrieve all the keys, retrieve all the values, retrieve all the key-value pairs, change the value associated with a key in a HashMap, remove a HashMap element given the key, remove a HashMap entry with Key and Value, check if a given "value" exists in the Hashmap and display the HashMap. Read user input where required and display suitable error messages where applicable.

import java.util.HashMap;

import java.util.Scanner;

public class LoginCredentialsManager {

// Method to display menu options

public static void displayMenu() {

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

System.out.println("1. Add key-value pair");

System.out.println("2. Retrieve value for a given key");

System.out.println("3. Retrieve all keys");

System.out.println("4. Retrieve all values");

System.out.println("5. Retrieve all key-value pairs");

System.out.println("6. Update value for a given key");

System.out.println("7. Remove entry by key");

System.out.println("8. Remove entry by key-value pair");

System.out.println("9. Check if a value exists");

System.out.println("10. Display entire HashMap");

System.out.println("0. Exit");

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

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

HashMap<String, String> loginCredentials = new HashMap<>();

int choice;

do {

displayMenu();

choice = sc.nextInt();

sc.nextLine(); // Consume newline

switch (choice) {

case 1: // Add key-value pair

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

String username = sc.nextLine();

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

String password = sc.nextLine();

loginCredentials.put(username, password);

System.out.println("Credentials added successfully!");

break;

case 2: // Retrieve value for a given key

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

username = sc.nextLine();

if (loginCredentials.containsKey(username)) {

System.out.println("Password: " + loginCredentials.get(username));

} else {

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

}

break;

case 3: // Retrieve all keys

System.out.println("Keys: " + loginCredentials.keySet());

break;

case 4: // Retrieve all values

System.out.println("Values: " + loginCredentials.values());

break;

case 5: // Retrieve all key-value pairs

System.out.println("Key-Value pairs: " + loginCredentials.entrySet());

break;

case 6: // Update value for a given key

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

username = sc.nextLine();

if (loginCredentials.containsKey(username)) {

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

password = sc.nextLine();

loginCredentials.put(username, password);

System.out.println("Password updated successfully!");

} else {

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

}

break;

case 7: // Remove entry by key

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

username = sc.nextLine();

if (loginCredentials.remove(username) != null) {

System.out.println("Username removed successfully!");

} else {

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

}

break;

case 8: // Remove entry by key-value pair

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

username = sc.nextLine();

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

password = sc.nextLine();

if (loginCredentials.remove(username, password)) {

System.out.println("Credentials removed successfully!");

} else {

System.out.println("Username and password do not match.");

}

break;

case 9: // Check if a value exists

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

password = sc.nextLine();

if (loginCredentials.containsValue(password)) {

System.out.println("Password exists.");

} else {

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

}

break;

case 10: // Display entire HashMap

System.out.println("HashMap contents: " + loginCredentials);

break;

case 0: // Exit

System.out.println("Exiting program...");

break;

default: // Invalid choice

System.out.println("Invalid choice. Please try again.");

break;

}

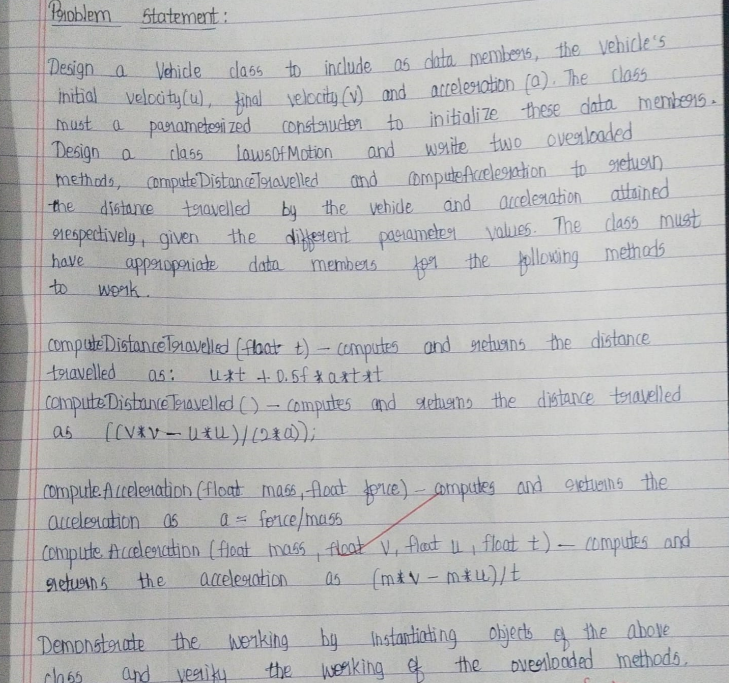
} while (choice != 0);

sc.close();

}

}

1. **TERM WORK**

****

import java.util.Scanner;

class Vehicle {

double u, v, a;

// Constructor to initialize values

Vehicle(double initial, double finalv, double acc) {

u = initial;

v = finalv;

a = acc;

}

// Method to compute distance traveled using the formula: s = ut + 0.5 \* a \* t^2

double computeDistanceTravelled(double t) {

double distance = u \* t + 0.5 \* a \* t \* t;

System.out.println("Distance travelled (using equation 1): " + distance);

return distance;

}

// Method to compute distance traveled using the formula: s = (v^2 - u^2) / (2 \* a)

double computeDistanceTravelled() {

double distance = (v \* v - u \* u) / (2 \* a);

System.out.println("Distance travelled (using equation 2): " + distance);

return distance;

}

// Method to compute acceleration using the formula: a = f / m

double computeAcceleration(double m, double f) {

double accn = f / m;

System.out.println("Acceleration (using equation 1): " + accn);

return accn;

}

// Method to compute acceleration using the formula: a = (v - u) / t

double computeAcceleration(double m, double v, double u, double t) {

double accn = (v - u) / t;

System.out.println("Acceleration (using equation 2): " + accn);

return accn;

}

}

public class LawsOfMotion {

public static void main(String[] args) {

Vehicle v = new Vehicle(0, 40, 20);

// Compute distance travelled using time (t = 10)

v.computeDistanceTravelled(10);

// Compute distance travelled using initial and final velocities

v.computeDistanceTravelled();

// Compute acceleration using force and mass

v.computeAcceleration(10, 100);

// Compute acceleration using mass, initial velocity, final velocity, and time

v.computeAcceleration(10, 20, 10, 10);

}

}