**JAVA LAB PROGRAMS:**

**1 a. Design and create a class named RetailItem that holds data about an item in a retail store. The class should have the following fields:**

* **Description - The description field references a String object that holds a brief description of the item.**
* **Units - The units field is an int variable that holds the number of units currently in inventory.**
* **Price - The price field is a double that holds the item’s retail price.**

**Write a constructor that accepts arguments for each field, appropriate mutator methods that store values in these fields, and accessor methods that return the values in these fields. Write the main method which creates three RetailItem objects and invokes appropriate methods.**

import java.util.\*;

class RetailItem

{

private String Description;

int units; float price;

RetailItem()

{

}

RetailItem(String a,int u,float p)

{

Description=a; units=u; price=p;

}

String retdes()

{

return Description;

}

int retu()

{

return units;

}

float retp()

{

return price;

}

void display()

{

System.out.println("Product name and description: "+this.retdes()); System.out.println("Number of units: "+retu()); System.out.println("Price of each unit: "+retp()); System.out.println("Total price: "+retu()\*retp());

}

}

public class onea

{

public static void main(String args[])

{

Scanner s=new Scanner(System.in); System.out.println("Enter the number of objects: "); int n=s.nextInt();

RetailItem obj[]=new RetailItem[n]; for(int i=0;i<3;i++)

{

");

Scanner sc=new Scanner(System.in);

System.out.println("Enter the name of the product along wih the description:

String a = sc.nextLine(); System.out.println("Enter the number of units: "); int u=sc.nextInt();

System.out.println("Enter the price of each product: "); float p=sc.nextFloat();

obj[i]= new RetailItem(a,u,p);

}

for(int i=0;i<3;i++)

obj[i].display();

}

}

**OUTPUT:**

Enter the number of objects:

3

Enter the name of the prduct along wih the description: lifebuoy: soap

Enter the number of units:

3

Enter the price of each product:

20

Enter the name of the prduct along wih the description: nicip: medicine

Enter the number of units:

4

Enter the price of each product:

10

Enter the name of the prduct along wih the description: classmate: notebook

Enter the number of units:

5

Enter the price of each product:

45

Product name and description: lifebuoy: soap Number of units: 3

Price of each unit: 20.0 Total price: 60.0

Product name and description: nicip: medicine Number of units: 4

Price of each unit: 10.0 Total price: 40.0

Product name and description: classmate: notebook Number of units: 5

Price of each unit: 45.0 Total price: 225.0

**1.b.Write a class named Car that has the following data members:**

* **model. The model field is an int that holds the car’s year.**
* **make. The make field references a String object that holds the make of the car.**
* **speed. The speed field is an int that holds the car’s current speed. The class should have the following constructor and other methods.**
* **The constructor should accept the car’s year model, make and speed as arguments.**
* **Accessor methods should get the values stored in an object’s year, Model, make, and speed fields.**
* **Accelerate method should add 5 to the speed field each time it is called.**
* **Brake method should subtract 5 from the speed field each time it is called.**

**Demonstrate the class in a program that creates a Car object, and then calls the accelerate method five times.**

**After each call to the accelerate method, get the current speed of the car and display it. Call the brake method five times. After each call to the brake method,**

**get the current speed of the car and display it.**

import java.util.\*; class car

{

int model; String make; int speed; car()

{}

car(int m,String b,int s)

{

model=m; make=b; speed=s;

}

int retm()

{

return model;

}

String ma()

{

return make;

}

int rets()

{

return speed;

}

void Accelerate()

{

speed+=5;

}

void Break()

{

speed-=5;

}

}

public class oneb

{

public static void main(String args[])

{

Scanner s=new Scanner(System.in); System.out.println("Enter the make (brand): "); String b=s.nextLine(); System.out.println("Enter the model (year): "); int y=s.nextInt();

System.out.println("Enter the current speed: "); int sp=s.nextInt();

car c=new car(y,b,sp); System.out.println("Model: "+c.retm()); System.out.println("Make: "+c.retm());

for(int i=0;i<5;i++)

{

1. ccelerate();

System.out.println("Current speed is: "+c.rets());

}

for(int i=0;i<5;i++)

{

1. reak();

System.out.println("Current speed is: "+c.rets());

}

}

}

OUTPUT:

Enter the make (brand):

Lamborghini

Enter the model (year):

2016

Enter the current speed:

256

Model: 2016

Make: 2016

Current speed is: 261 Current speed is: 266 Current speed is: 271 Current speed is: 276 Current speed is: 281 Current speed is: 276 Current speed is: 271 Current speed is: 266 Current speed is: 261 Current speed is: 256

**2a. Identify the type of inheritance in the given diagram. Create a class A with two integer member**

**variables that are private, two float variables that are protected and two integer variables that are public.**

**Let class B inherit class A and class C and Class D are inherited from class B. Write appropriate**

**methods to illustrate the following**

**i) Usage of super keyword**

**ii) Function overriding**

**iii) Default constructors**

**iv) Parameterized constructors**

**v) How to we make a method not to be over ridden and a class not be inherited further**

class A {

private int a;

private int b;

protected float c;

protected float d;

public int e;

public int f;

int g = 10;

A() {// default constructor

a = 0;

b = 0;

c = 0.0f;

d = 0.0f;// this problem contains hybrid

inheritance

e = 0;

f = 0;

}

A(int p, int q, float r, float s, int t, int u) {

// parameterized constructor

a = p;

b = q;

c = r;

d = s;

e = t;

f = u;

}

void display() {

System.out.println(" a: " + a + " b: " + b + "

c: " + c + " d: " + d + " e: " + e + " f: " + f);

}

/\* final \*/ void read() { // using final keyword to

stop method overriding

System.out.println(" value of g : " + g);

}

}

/\* final\*/ class B extends A { // using final keyword

to stop inheritance

int g = 100;

void read() { // method overriding

System.out.println(" value of g : " + g);

System.out.println(" value of g : " +

super.g);// displays (parent's class) g value of A , as

super keyword refers to immediate parent's class

instance when it has same variable as child class

}

}

class C extends B {

}

class D extends B {

}

class Lab3 {

public static void main(String args[]) {

A a = new A(10, 20, 30.5f, 40.5f, 50, 60);// c

and d are of float type

a.display();

B b = new B();

b.read();

}

}

**2b.Define one class A in package apack. In class A, four variables are defined of access modifiers default, protected,private and public.**

**Define class B in package bpack which extends A and write display() method which access variables of class A.Define class C in package cpack which has one method display() in that create one object of class A and display its variables.**

**Define class ProtectedDemo in package dpack which contains the main () method. Create objects of class B and C and call dis display method for both these objects. Analyze the program by**

**interpreting the access modifiers and provide valid conclusion.**

Package apack

package apack;

import java.util.\*;

public class A {

public int a=1;

protected int b=2;

int c=3;

private int d=4;

public void read()

{

Scanner o=new Scanner(System.in);

System.out.println("input value of a");

a=o.nextInt();

System.out.println("input value of b");

b=o.nextInt();

System.out.println("input value of c");

c=o.nextInt();

System.out.println("input value of d");

d=o.nextInt();

}

}

package bpack

package bpack;

import apack.\*;

public class B extends A {

public void display()

{

System.out.println("value of a: "+a);

System.out.println("value of b: "+b);

// System.out.println("value of c: "+c);

// System.out.println("value of d: "+d);

}

}

package cpack

package cpack;

import apack.\*;

public class C {

public void display()

{

A ob=new A();

System.out.println("Value of a: "+ob.a);

// System.out.println("Value of b: "+ob.b)''

// System.out.println("Value of c: "+ob.c);

// System.out.println("Value of d: "+ob.d);

}

}

package dpack

package dpack;

import bpack.\*;

import cpack.\*;

import apack.\*;

public class protecteddemo {

public static void main(String[] args) {

B obj =new B();

C obj1=new C();

obj.display();

obj1.display();

}

}

**2c.Design the given model**

import java.lang.Math;

import java.util.\*;

interface IPoint

{

int a=20,b=10;

void message();

}

abstract class Shape1

{

double area;

abstract void draw();

abstract void area();

}

class Triangle extends Shape1 implements IPoint

{

public void message()

{

System.out.println("Hello Traiangle ");

}

void draw()

{

System.out.println("A triangle drawn");

}

void area()

{

area=0.5\*(a\*b);

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

}

}

class Circle extends Shape1

{

int r;

Circle(int ra)

{

r=ra;

}

void draw()

{

System.out.println("A circle drawn");

}

void area()

{

area=3.14\*r\*r;

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

}

}

class ThreeDCircle extends Circle

{

ThreeDCircle(int rd)

{

super(rd);

}

void draw()

{

System.out.println("A sphere drawn");

}

void area()

{

area=4\*3.14\*r\*r;

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

}

}

class Hexagon extends Shape1 implements IPoint

{

public void message()

{

System.out.println("Hello Hexagon");

}

void draw()

{

System.out.println("A hexagon has been drawn");

}

void area()

{

area=(1.5)\*Math.sqrt(3)\*a;

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

}

}

class Demo

{

public static void main(String args[])

{

Triangle shape1=new Triangle();

shape1.message();

shape1.draw();

shape1.area();

Circle shape2=new Circle(5);

shape2.draw();

shape2.area();

ThreeDCircle shape3=new ThreeDCircle(6);

shape3.draw();

shape3.area();

Hexagon shape4=new Hexagon();

shape4.message();

shape4.draw();

shape4.area();

}

}

**3a.Consider a student examination database system that prints the mark sheet of the students.**

**Input the following from the command line student name and marks in 6 subjects.**

**These marks should be in between 0 and 50 if the marks are not in the specified range raise a Range Exception else find the total marks and print the percentage of the student.**

import java.util.InputMismatchException;

public class studentexception {

public static void main(String[] args) {

double totalmarks = 0, percentage;

String name = args[0];

int marks[] = new int[6];

try {

for (int i = 1; i <= 6; i++) {

marks[i - 1] = Integer.parseInt(args[i]);

if (marks[i - 1] >= 0 && marks[i - 1] <= 50) {

totalmarks += marks[i - 1];

} else

throw new InputMismatchException("Exception: Marks must be in

the range of 0 to 50");

}

percentage = (totalmarks / (6 \* 50)) \* 100;

System.out.println("Student Details-: ");

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

System.out.print("Marks in 6 subjects: ");

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

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

System.out.println("\nTotal Marks: " + totalmarks);

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

} catch (InputMismatchException e) {

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

}

}

}

**3b. Create a class temperature with member variable temp. Implement exception handling to test if temperature is equal to zero.**

import java.util.Scanner;

public class Temperature {

float temp;

Temperature() {

Scanner in = new Scanner(System.in);

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

temp = in.nextFloat();

in.close();

}

void check() {

try {

if (temp == 0) {

throw new

NullPointerException("Exception: Temperature can not be

ZERO");

}

System.out.println("Temperature = " +

temp);

} catch (NullPointerException e) {

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

}

}

}

class DemoTemperature {

public static void main(String[] args) {

Temperature ob = new Temperature();

ob.check();

}

}

**4a. Consider a Bus reservation system that allows onlinereservations to its customers.**

**Suppose there are two transactions of reservation for a particular seat simultaneously which**

**leads to race condition. Develop a solution to avoid the unpredictable situation with a program.**

class TicketBooking implements Runnable {

int ticketsAvailable = 1;

public void run() {

System.out.println("Waiting to book ticket for

: " + Thread.currentThread().getName());

synchronized (this) {

if (ticketsAvailable > 0) {

System.out.println("Booking ticket for

: " + Thread.currentThread().getName());

try {

Thread.sleep(1000);

} catch (Exception e) {

}

ticketsAvailable--;

System.out.println("Ticket BOOKED for :

" + Thread.currentThread().getName());

System.out.println("currently

ticketsAvailable = " + ticketsAvailable);

} else {

System.out.println("Ticket NOT BOOKED

for : " +

Thread.currentThread().getName());

}

}//end of synchronization block

}

}

public class BusReserve {

public static void main(String[] args) {

TicketBooking obj = new TicketBooking();

Thread customer1 = new Thread(obj, "customer1

");//Thread(obj);

Thread customer2 = new Thread(obj, "customer2

");//Thread(obj);

customer1.start();

customer2.start();

}

}

**4 b. Create a class called Library. Write a program to manipulate the book information from files by using FileInputStream and FileOutputStream.**

import java.io.FileInputStream;

import java.io.FileOutputStream;

public class Library {

public static void main(String[] args) {

System.out.println("Writing in File");

try {

FileOutputStream fout = new

FileOutputStream("demo.txt");

String str = "Name of the book :A suitable

boy ,Author:Vikram Seth, Total\_Page:1500";

byte b[] = str.getBytes();

fout.write(b);

fout.close();

System.out.println("successful write.");

} catch (Exception e) {

System.out.println(e);

}

System.out.println("Reading From File");

try {

FileInputStream fin = new

FileInputStream("demo.txt");

int i = 0;

while ((i = fin.read()) != -1) {

System.out.print((char) i);

}

fin.close();

} catch (Exception e) {

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

}

}

}

**5 a. Write a program to implement dynamic growable queue using generics.**

class Queue<E>

{

E[] a;

int front,rear;

Queue()

{

a=(E[])new Object[10];

front =0;

rear=0;

}

void enqueue(E m)

{

if(rear==10)

{

growable();

}

a[rear]=m;

rear++;

}

E dequeue()

{

E item;

item=a[front];

front++;

return item;

}

boolean haselement()

{

return front!=rear;

}

void growable()

{

int n= 10\*2;

E newarr[];

newarr=(E[])new Object[n];

int tmpfront=front;

int tmprear=0;

while(true)

{

newarr[tmprear++]=this.a[tmpfront];

tmpfront++;

if(tmpfront==10)

{

tmpfront=0;

break;

}

}

this.a=newarr;

this.front=0;

this.rear=tmprear;

}

}

class student

{

String name;

int id;

student(String m,int n)

{

name=m;

id=n;

}

public String toString()

{

return name+" "+id;

}

}

public class Genericqueue

{

public static void main (String[] args)

{

Queue<Integer> obj=new Queue<Integer>();

Queue<Double> obj1=new Queue<Double>();

Queue<student> obj2=new Queue<student>();

obj.enqueue(1);

obj.enqueue(2);

obj.enqueue(3);

obj.enqueue(4);

obj.enqueue(5);

obj.enqueue(6);

obj.enqueue(7);

obj.enqueue(8);

obj.enqueue(9);

obj.enqueue(10);

obj.enqueue(11);

obj.enqueue(12);

obj.enqueue(13);

obj.enqueue(14);

obj.enqueue(15);

obj1.enqueue(1.2);

obj1.enqueue(2.4);

obj1.enqueue(3.6);

obj1.enqueue(4.4);

obj1.enqueue(5.55);

obj2.enqueue(new student("Ram",1));

obj2.enqueue(new student("Krish",2));

obj2.enqueue(new student("Ganesh",3));

obj2.enqueue(new student("Kiran",4));

while(obj.haselement())

{

System.out.println(obj.dequeue());

}

while(obj1.haselement())

{

System.out.println(obj1.dequeue());

}

while(obj2.haselement())

{

System.out.println(obj2.dequeue());

}