**Assignment – 2**

1. **Car having property modelname,color,price. Create constructor to set thevalues. Override the method toString() method of Object class. Override equals() method of object class which check if two car objects have sameprice then it is equal.**

import java.lang.String;

class Car {

String name;

String color;

double price;

Car(String n, String c, double p) {

name = n;

color = c;

price = p;

}

public boolean equals(Object o) {

if (this == o)

return true;

if (o == null || getClass() != o.getClass())

return false;

Car c = (Car) o;

return Double.compare(c.price, price) == 0;

}

public String toString() {

return "Name: " + name + ", color: " + color + ", price: " + price;

}

}

public class ass0201 {

public static void main(String args[]) {

Car c1 = new Car("alto", "white", 400000);

Car c2 = new Car("scorpio", "blue", 400000);

System.out.println(c1.equals(c2));

System.out.println(c2.toString());

}

}

G:\Java> java ass0201

true

Name: Bolero, color: blue, price: 600000.0

1. **Create class Employee with property name, basicsalary, noofleave. Createconstructor and method void calbudget() which calculate and the display thesalary. salary = (30-noofleave)\*basicpay. Create class TeamLead with property projectname, noofemployee ,noofprojectdays, rateperday. Create constructor and override method calbudget() which calculate the budget of the project. (budget= noofemployee\*noofprojectdays\*rateperday). Create an object of Employee and display salary. Create object of Teamleadand display project budget**

import java.lang.String;

class Employee {

String name;

double basicSalary;

int noOfLeave;

Employee(String n, double b, int nl)

{

name = n;

basicSalary = b;

noOfLeave = nl;

}

void calbudget() {

System.out.println((30 - noOfLeave) \* basicSalary);

}

}

class TeamLead {

String projectName;

int noOfEmployee, noOfProjectDays;

double ratePerDay;

TeamLead(String n, int ne, int np, double r) {

projectName = n;

noOfEmployee = ne;

noOfProjectDays = np;

ratePerDay = r;

}

void calbudget() {

System.out.println(noOfEmployee \* noOfProjectDays \* ratePerDay);

}

}

public class ass0202 {

public static void main(String[] args) {

Employee e = new Employee("x", 50000, 3);

e.calbudget();

TeamLead t = new TeamLead("compiler", 150, 10, 1.1);

t.calbudget();

}

}

G:\Java> java ass0202

1350000.0

1650.0

1. **Create an Interface called Ticket. It contain the method void booktickets(). Create class Movie which implements interface Ticket. Create constructor for set values noofticket, priceperticket . Calculate anddisplay the total cost in booktickets() method. Create class Train which implements the interface Tickets. Create constructor which set values noofticket, ticketclass, priceperticket. calculate and display the total cost into booktickets() method. Take price of ticket based on ticketclass. Create object of Ticket and Movie and calculate and display total amount. Note : required values can be pass into the constructor of respective class.**

import java.lang.String;

interface Ticket {

void bookTickets();

}

class Movie implements Ticket {

int noOfTickets;

double pricePerTicket;

public Movie(int noOfTickets, double pricePerTicket) {

this.noOfTickets = noOfTickets;

this.pricePerTicket = pricePerTicket;

}

public void bookTickets() {

System.out.println(noOfTickets \* pricePerTicket);

}

}

class Train implements Ticket {

int noOfTickets;

String ticketClass;

double pricePerTicket;

public Train(int noOfTickets, String ticketClass, double pricePerTicket) {

this.noOfTickets = noOfTickets;

this.ticketClass = ticketClass;

this.pricePerTicket = pricePerTicket;

}

public void bookTickets() {

System.out.println(noOfTickets \* pricePerTicket);

}

}

public class ass0203 {

public static void main(String[] args) {

Ticket t1 = new Movie(3, 12.5);

t1.bookTickets();

Ticket t2 = new Train(2, "First Class", 50.0);

t2.bookTickets();

}

}

G:\Java> java ass0203

37.5

100.0

1. **Discuss the important point of Dequeue collection class. Create a collection class of String type. And perform following task based on choice 1. add element in last 2. pop the element 3. search the element and remove( if element is present then delete otherwise display message element does not present**

//Java Deque Interface is a linear collection that supports element insertion and removal at both ends. Deque is an acronym for "double ended queue". A deque can perform both the insertion and deletion using the LIFO (Last In First Out) principle

import java.util.\*;

import java.lang.String;

public class ass0204 {

public static void main(String args[]) {

Deque<String> d = new ArrayDeque<String>();

Scanner sc = new Scanner(System.in);

int c;

do {

System.out.println("1. add 2. pop 3. search");

c = sc.nextInt();

switch (c) {

case 1:

String s = sc.next();

d.add(s);

break;

case 2:

System.out.println(d.poll());

break;

case 3:

String searchStr = sc.next();

if (d.contains(searchStr)) {

d.remove(searchStr);

System.out.println("removed: " + searchStr);

} else {

System.out.println("element not there");

}

break;

default:

break;

}

} while (c >= 1 && c <= 3);

}

}

G:\Java> java ass0204

1. add 2. pop 3. search

1

hi

1. add 2. pop 3. search

2

hi

1. add 2. pop 3. search

3

bye

element not there