**Roll no:: 412084**

**Prn No: 2019033800126574**

**Name:: Yagnik Mojidra**

**:Assignment-5:**

**Github::** <https://github.com/YagnikMojidra/Dot-Net-workspace>

**\*\*Geerics in C#\*\***

**Code:**

internal interface IVehicle {

internal enum Renttype {

KM,Day

}

internal decimal CalculateRent(int Units);

internal void getDetails();

internal DateOnly getLastMaintenanceDate();

}

internal class Indica : IVehicle {

internal string? type, number;

internal IVehicle.Renttype renttype;

internal int age, rentperunit, seater;

internal DateOnly last\_maintenance\_date;

internal Indica(string type, int seater, string number,IVehicle.Renttype rentType, int age, int rentperunit, DateOnly last\_maintenance\_date) {

this.type = type;

this.seater = seater;

this.number = number;

renttype = rentType;

this.age = age;

this.rentperunit = rentperunit;

this.last\_maintenance\_date = last\_maintenance\_date;

}

public decimal CalculateRent(int Units) {

return (decimal)rentperunit\*Units;

}

public void getDetails()

{

Console.Write("Brand : Indica \n");

Console.Write($"Car Number : {number}\n");

Console.Write($"Total Seats : {seater}\n");

Console.Write($"Type : {type}\n");

Console.Write($"Car age : {age}\n");

Console.Write($"Rent per unit : {rentperunit}\n");

}

public DateOnly getLastMaintenanceDate() {

return last\_maintenance\_date;

}

}

internal class Qualis : IVehicle {

internal string? type, number;

internal IVehicle.Renttype renttype;

internal int age, rentperunit, seater;

internal DateOnly last\_maintenance\_date;

internal Qualis(string type, int seater, string number,IVehicle.Renttype rentType, int age, int rentperunit, DateOnly last\_maintenance\_date) {

this.type = type;

this.seater = seater;

this.number = number;

renttype = rentType;

this.age = age;

this.rentperunit = rentperunit;

this.last\_maintenance\_date = last\_maintenance\_date;

}

public decimal CalculateRent(int Units) {

return (decimal)rentperunit\*Units;

}

public void getDetails()

{

Console.Write("Brand : Qualis \n");

Console.Write($"Car Number : {number}\n");

Console.Write($"Total Seats : {seater}\n");

Console.Write($"Type : {type}\n");

Console.Write($"Car age : {age}\n");

Console.Write($"Rent per unit : {rentperunit}\n");

}

public DateOnly getLastMaintenanceDate() {

return last\_maintenance\_date;

}

}

internal class HarleyDavid : IVehicle {

internal string? type, number;

internal IVehicle.Renttype renttype;

internal int age, rentperunit, seater;

internal DateOnly last\_maintenance\_date;

internal HarleyDavid(string type, int seater, string number,IVehicle.Renttype rentType, int age, int rentperunit, DateOnly last\_maintenance\_date) {

this.type = type;

this.seater = seater;

this.number = number;

renttype = rentType;

this.age = age;

this.rentperunit = rentperunit;

this.last\_maintenance\_date = last\_maintenance\_date;

}

public decimal CalculateRent(int Units) {

return (decimal)rentperunit\*Units;

}

public void getDetails()

{

Console.Write("Brand : HarleyDavid \n");

Console.Write($"Car Number : {number}\n");

Console.Write($"Total Seats : {seater}\n");

Console.Write($"Type : {type}\n");

Console.Write($"Car age : {age}\n");

Console.Write($"Rent per unit : {rentperunit}\n");

}

public DateOnly getLastMaintenanceDate() {

return last\_maintenance\_date;

}

}

internal class MercedesBenz : IVehicle {

internal string? type, number;

internal IVehicle.Renttype renttype;

internal int age, rentperunit, seater;

internal DateOnly last\_maintenance\_date;

internal MercedesBenz(string type, int seater, string number,IVehicle.Renttype rentType, int age, int rentperunit, DateOnly last\_maintenance\_date) {

this.type = type;

this.seater = seater;

this.number = number;

renttype = rentType;

this.age = age;

this.rentperunit = rentperunit;

this.last\_maintenance\_date = last\_maintenance\_date;

}

public decimal CalculateRent(int Units) {

return (decimal)rentperunit\*Units;

}

public void getDetails()

{

Console.Write("Brand : HarleyDavid \n");

Console.Write($"Car Number : {number}\n");

Console.Write($"Total Seats : {seater}\n");

Console.Write($"Type : {type}\n");

Console.Write($"Car age : {age}\n");

Console.Write($"Rent per unit : {rentperunit}\n");

}

public DateOnly getLastMaintenanceDate() {

return last\_maintenance\_date;

}

}

public class CarType<T> {

internal T carobj;

internal DateOnly startDate,endDate;

internal int Units;

internal decimal advPayment;

internal CarType(T carobj, DateOnly startDate,DateOnly endDate, decimal advPayment) {

this.carobj = carobj;

this.advPayment=advPayment;

this.startDate = startDate;

this.endDate = endDate;

}

internal CarType(T carobj) {

this.carobj = carobj;

}

internal int CalculateDays() {

int year = endDate.Year - startDate.Year;

int month = endDate.Month - startDate.Month;

int day = endDate.Day - startDate.Day;

return year + month + day;

}

}

internal class RentedVehicle<T> {

List<CarType<T>> Vehiclelist;

internal RentedVehicle() {

Vehiclelist = new List<CarType<T>>();

}

internal void AddVehicle(T carobj) {

CarType<T> c = new CarType<T>(carobj);

}

internal void GiveForRent(T carobj, DateOnly startDate, DateOnly endDate, decimal adv\_pay) {

CarType<T> c = new CarType<T>(carobj, startDate, endDate, adv\_pay);

Vehiclelist.Add(c);

}

internal decimal CalculateRent(T carobj, int Units) {

foreach(CarType<T> c in Vehiclelist) {

if(c.carobj!.Equals(carobj)) {

c.Units = Units;

return ((IVehicle)carobj).CalculateRent(Units) - c.advPayment;

}

}

return 0;

}

internal decimal CalculateTotalRentToday(T carobj, int TrvaelUnits) {

foreach(CarType<T> c in Vehiclelist) {

if(c.carobj!.Equals(carobj)) {

return (((IVehicle)carobj).CalculateRent(TrvaelUnits) - c.advPayment)/c.CalculateDays();

}

}

return 0;

}

internal void GetCheckListRentedVehicle() {

foreach(CarType<T> c in Vehiclelist) {

((IVehicle)c.carobj!).getDetails();

Console.Write($"\n Rented From {c.startDate} to {c.endDate}");

}

}

internal bool CheckVehiclesInMaintenance(T carobj) {

DateOnly today = DateOnly.FromDateTime(DateTime.Today);

foreach(CarType<T> c in Vehiclelist) {

IVehicle car = ((IVehicle)c.carobj!);

if(c.carobj!.Equals(carobj) && car.getLastMaintenanceDate().CompareTo(today) > 0)

return true;

}

return false;

}

internal void ShowAvailableByDate(DateOnly date) {

Console.Write($"\n\n Available Vehicles on {date}");

foreach(CarType<T> c in Vehiclelist) {

if(c.startDate.CompareTo(date) > 0) {

((IVehicle)c.carobj!).getDetails();

}

}

}

}

class Program {

static void Main(string[] args) {

// Console.Write("\n\n Adding Cars to the list");

Indica i1 = new Indica("Petrol", 5, "GJ-04-DA-0204", IVehicle.Renttype.Day, 10, 13, new DateOnly(2020, 12,16 ));

MercedesBenz mb1 = new MercedesBenz("Diesel",7, "GJ-11-AB-0786", IVehicle.Renttype.KM, 3, 17, new DateOnly(2021, 07,18 ));

Qualis q1 = new Qualis("Diesel", 7, "GJ-04-CR-5700", IVehicle.Renttype.KM, 7, 5, new DateOnly(2021, 11,21));

Qualis q2 = new Qualis("CNG", 4, "GJ-01-IJ-0001", IVehicle.Renttype.KM, 15, 14, new DateOnly(2022, 02,28 ));

MercedesBenz mb2 = new MercedesBenz("Petrol",7, "GJ-01-BA-7800", IVehicle.Renttype.KM, 3, 17, new DateOnly(2020, 10,26 ));

RentedVehicle<IVehicle> rv = new RentedVehicle<IVehicle>();

rv.AddVehicle(i1);

rv.AddVehicle(mb1);

rv.AddVehicle(q1);

rv.AddVehicle(q2);

rv.AddVehicle(mb2);

rv.GiveForRent(i1, new DateOnly(2021, 12, 20), new DateOnly(2021, 12, 29), 0);

rv.GiveForRent(q2, new DateOnly(2022, 07, 10), new DateOnly(2022, 07, 15), 500);

rv.GiveForRent(mb1, new DateOnly(2022, 09, 05), new DateOnly(2022, 09, 19), 1500);

Console.Write("\n Total rent per day for the given car : ");

q2.getDetails();

Console.Write($"\n\n Total rent per day : {rv.CalculateTotalRentToday(mb2, 5):C2}");

Console.Write("\n -------------------------------------------");

Console.Write("\n\n Show how many vehicles are available before 29-March-2022");

rv.ShowAvailableByDate(new DateOnly(2022, 03, 29));

Console.Write("\n -------------------------------------------");

Console.Write("\n\n Show how many vehicles are currently rented");

rv.GetCheckListRentedVehicle();

}

}

**Output::**

