PRN:-2019033800120821

NAME:-PATEL PRIT SANJAYKUMAR

BATCH:- A

Git repo:- https://github.com/pritpatel179/Assignments

ASSIGNMENT 5

CODE :

Program.cs

using System;

using System.Collections.Generic;

using System.Text;

using carrentsystem;

class Program{

static void Main(string[] args) {

Indica i1 = new Indica("Diesel", 5, "GJ-05-CJ-6146",

VehicleInfo.TypeOfRent.Day, 1, 5, new DateOnly(2022, 01,28 ));

Indica i2 = new Indica("Petrol", 5, "MH-11-KJ-0490",

VehicleInfo.TypeOfRent.KiloMeter, 5, 8, new DateOnly(2021, 08,06 ));

Qualis q1 = new Qualis("Diesel", 7, "TN-10-OP-1923",

VehicleInfo.TypeOfRent.KiloMeter, 10, 10, new DateOnly(2019, 12,23));

MerBenEClass mb1 = new MerBenEClass("Diesel",2, "GJ-01-AA-1111",

VehicleInfo.TypeOfRent.KiloMeter, 3, 25, new DateOnly(2022, 01,15 ));

DavidHarley dh1 = new DavidHarley("Petrol",5, "MH-21-DK-7921",

VehicleInfo.TypeOfRent.Day, 4, 20, new DateOnly(2021, 03,06 ));

DavidHarley dh2 = new DavidHarley("Diesel",4, "GJ-06-WE-1234", VehicleInfo.TypeOfRent.KiloMeter, 6, 18, new DateOnly(2021, 05,26 ));

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

rv.AddVehicle(i1); rv.AddVehicle(i2); rv.AddVehicle(q1); rv.AddVehicle(mb1);

rv.AddVehicle(dh1);

rv.GiveForRent(i1, new DateOnly(2022, 02, 1), new DateOnly(2022, 02,

15), 100);

rv.GiveForRent(i2, new DateOnly(2022, 01, 8), new DateOnly(2022, 01,

10), 500);

rv.GiveForRent(dh2, new DateOnly(2022, 01, 15), new DateOnly(2022, 01,

28), 1000);

Console.Write("\nINFORMATION OF THIS INDICA CAR : \n ");

i2.getInfo();

Console.Write($"\n\n TOTAL RENT PER KILOMETER :

{rv.CalculateTotalRentToday(i2, 5):C2}");

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

Console.Write("\n\n VEHICLES AVAILABLE BEFORE 15-May-2021\n");

rv.ShowAvailabilityforbookingforgivendate(new DateOnly(2021, 03, 11));

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

Console.Write("\n\n VEHICLES WHICH ARE ON RENT CURRENTLY\n");

rv.GetCheckListREntedandAvailableVehicle();

}

}

Carrent.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace carrentsystem

{

internal interface VehicleInfo

{

internal void getInfo();

internal decimal CalculateRent(int Units); internal DateOnly getLastMaintenanceDate(); internal enum TypeOfRent

{

Day, KiloMeter

}

}

internal class Indica : VehicleInfo

{

internal string? type, number;

internal VehicleInfo.TypeOfRent renttype;

internal int age, rentperunit, seater;

internal DateOnly LastMaintainance;

internal Indica(string type, int seater, string number, VehicleInfo.TypeOfRent rentType, int age, int rentperunit, DateOnly LastMaintainance)

{

this.type = type; this.seater = seater; this.number = number; renttype = rentType; this.age = age;

this.rentperunit = rentperunit;

this.LastMaintainance = LastMaintainance;

}

public decimal CalculateRent(int Units)

{

return (decimal)rentperunit \* Units;

}

public void getInfo()

{

Console.Write("Company : Indica \n"); Console.Write($"Type : {type}\n"); Console.Write($"Rent per unit : {rentperunit}\n"); Console.Write($"Car NumberPlate : {number}\n"); Console.Write($"Number Of Seats : {seater}\n"); Console.Write($"Car age : {age}\n");

Console.WriteLine("\n");

}

public DateOnly getLastMaintenanceDate()

{

return LastMaintainance;

}

}

internal class Qualis : VehicleInfo

{

internal string? type, number;

internal VehicleInfo.TypeOfRent renttype;

internal int age, rentperunit, seater;

internal DateOnly LastMaintainance;

internal Qualis(string type, int seater, string number, VehicleInfo.TypeOfRent rentType, int age, int rentperunit, DateOnly LastMaintainance)

{

this.type = type; this.seater = seater; this.number = number; renttype = rentType; this.age = age;

this.rentperunit = rentperunit;

this.LastMaintainance = LastMaintainance;

}

public decimal CalculateRent(int Units)

{

return (decimal)rentperunit \* Units;

}

public void getInfo()

{

Console.Write("Company : Qualis \n"); Console.Write($"Type : {type}\n"); Console.Write($"Rent per unit : {rentperunit}\n"); Console.Write($"Car NumberPlate : {number}\n");

Console.Write($"Number Of Seats : {seater}\n"); Console.Write($"Car age : {age}\n");

Console.WriteLine("\n");

}

public DateOnly getLastMaintenanceDate()

{

return LastMaintainance;

}

}

internal class DavidHarley : VehicleInfo

{

internal string? type, number;

internal VehicleInfo.TypeOfRent renttype;

internal int age, rentperunit, seater;

internal DateOnly LastMaintainance;

internal DavidHarley(string type, int seater, string number, VehicleInfo.TypeOfRent rentType, int age, int rentperunit, DateOnly LastMaintainance)

{

this.type = type; this.seater = seater; this.number = number; renttype = rentType; this.age = age;

this.rentperunit = rentperunit;

this.LastMaintainance = LastMaintainance;

}

public decimal CalculateRent(int Units)

{

return (decimal)rentperunit \* Units;

}

public void getInfo()

{

Console.Write("Company : DavidHarley \n"); Console.Write($"Type : {type}\n"); Console.Write($"Rent per unit : {rentperunit}\n"); Console.Write($"Car NumberPlate : {number}\n"); Console.Write($"Number Of Seats : {seater}\n"); Console.Write($"Car age : {age}\n"); Console.WriteLine("\n");

}

public DateOnly getLastMaintenanceDate()

{

return LastMaintainance;

}

}

internal class MerBenEClass : VehicleInfo

{

internal string? type, number;

internal VehicleInfo.TypeOfRent renttype;

internal int age, rentperunit, seater;

internal DateOnly LastMaintainance;

internal MerBenEClass(string type, int seater, string number, VehicleInfo.TypeOfRent rentType, int age, int rentperunit, DateOnly LastMaintainance)

{

this.type = type; this.seater = seater; this.number = number; renttype = rentType; this.age = age;

this.rentperunit = rentperunit;

this.LastMaintainance = LastMaintainance;

}

public decimal CalculateRent(int Units)

{

return (decimal)rentperunit \* Units;

}

public void getInfo()

{

Console.Write("Company : MBenzEclass \n"); Console.Write($"Type : {type}\n"); Console.Write($"Rent per unit : {rentperunit}\n"); Console.Write($"Car NumberPlate : {number}\n"); Console.Write($"Number Of Seats : {seater}\n"); Console.Write($"Car age : {age} \n");

Console.WriteLine("\n");

}

public DateOnly getLastMaintenanceDate()

{

return LastMaintainance;

}

}

public class Car<T>

{

internal T carobj;

internal DateOnly startDate, endDate;

internal int Units;

internal decimal advancePayment;

internal int CalculateDays()

{

int day = endDate.Day - startDate.Day;

int month = endDate.Month - startDate.Month;

int year = endDate.Year - startDate.Year;

return day + month + year;

}

internal Car(T carobj, DateOnly startDate, DateOnly endDate, decimal advancePayment)

{

this.carobj = carobj; this.advancePayment = advancePayment; this.startDate = startDate; this.endDate = endDate;

}

internal Car(T carobj)

{

this.carobj = carobj;

}

}

internal class RentedVehicle<T>

{

List<Car<T>> Vehiclelist;

internal RentedVehicle()

{

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

}

internal void AddVehicle(T carobj)

{

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

}

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

{

Car<T> c = new Car<T>(carobj, startDate, endDate, adv\_pay); Vehiclelist.Add(c);

}

internal decimal CalculateRent(T carobj, int Units)

{

foreach (Car<T> c in Vehiclelist)

{

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

{

c.advancePayment;

}

c.Units = Units;

return ((VehicleInfo)carobj).CalculateRent(Units) -

}

return 0;

}

internal decimal CalculateTotalRentToday(T carobj, int TravelledUnits)

{

foreach (Car<T> c in Vehiclelist)

{

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

{

return (((VehicleInfo)carobj).CalculateRent(TravelledUnits) - c.advancePayment) / c.CalculateDays();

}

}

return 0;

}

internal void GetCheckListREntedandAvailableVehicle()

{

foreach (Car<T> c in Vehiclelist)

{

((VehicleInfo)c.carobj!).getInfo();

Console.Write($"THIS VEHICLE IS RENTED FROM DATE {c.startDate} TO {c.endDate} \n");

Console.WriteLine("\n");

}

}

internal bool GetCheckListvehilcesinmaintainence(T carobj)

{

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

foreach (Car<T> c in Vehiclelist)

{

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

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

car.getLastMaintenanceDate().CompareTo(today) > 0)

return true;

}

return false;

}

internal void ShowAvailabilityforbookingforgivendate(DateOnly date)

{

Console.Write($"\n\n VEHICALS ARE AVAILABLE ON {date} \n");

foreach (Car<T> c in Vehiclelist)

{

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

{

((VehicleInfo)c.carobj!).getInfo();

}

}

}

}

}

Output:





