**C# OOP-Basic**

**1.Правя 2 папки**

**- Core - тук стои контролният клас**

**- Entities – разделя се на папки по класовете които ще имам и си правя и класовете(абстрактни и нормални),добавям и наследяването.Пускам в Judge да се тества структурата.Ако получа грешка – описано е кой клас липсва(в нулевият тест)!Много е важно имената на класовете да са същите както в заданието.**

**2.Отивам в папка Core/Контролен клас и копирам всички методи,които той трябва да има от заданието.Методите,които връщат нещо им давам 0 или string.Empty!Всички методи трябва да са Public!!!**

**3.В папка Core,правя клас Engine,с който ще чета данните.В Main(StartUp):**

Engine engine=new Engine();

engine.Run();

**4.В контролният клас правя основните структури+базов конструктор**

**Пример:**

private Dictionary<int, Car> cars;

private Dictionary<int, Race> races;

private Garage garege;

public CarManager()

{

this.cars=new Dictionary<int, Car>();

this.races=new Dictionary<int, Race>();

this.garege=new Garage();

}

**5.Да прочета много внимателно Constrains-Описани са какви ще бъдат всички числа и string-ове!**

**6.В класа Engine правя поле и конструктор за контролният клас и започвам да чета командите и ги разделям по вид:**

public class Engine

{

private CarManager manager;

public Engine()

{

this.manager=new CarManager();

}

public void Run()

{

while (true)

{

string command = Console.ReadLine();

if(command== "Cops Are Here") break;

string[] cmdArgs = command.Split();

ExecuteCommand(cmdArgs);

}

}

public void ExecuteCommand(string[] cmdArgs)

{

switch (cmdArgs[0])

{

case "register":

break;

case "check":

break;

case "open":

break;

case "participate":

break;

case "start":

break;

case "park":

break;

case "unpark":

break;

case "tune":

break;

}

}

}

**7.Така вече може да достъпваме методите от контролният клас:** manager.Register();

**Важно е да подаваме данните,точно както е описано в заданието!**

**8.Подаваме данни за register:**

case "register":

int id = int.Parse(cmdArgs[1]);

string type=cmdArgs[2];

string brand = cmdArgs[3];

string model = cmdArgs[4];

int yearOfProduction=int.Parse(cmdArgs[5]);

int horsepower = int.Parse(cmdArgs[6]);

int acceleration = int.Parse(cmdArgs[7]);

int suspension = int.Parse(cmdArgs[8]);

int durability = int.Parse(cmdArgs[9]);

manager.Register(id,type,brand,model,yearOfProduction,horsepower,acceleration,suspension,durability);

break;

**Отиваме в класа Car и правим нейните полета+конструктор:**

public abstract class Car

{

private string brand;

private string model;

private int yearOfProduction;

private int horsepower;

private int acceleration;

private int suspension;

private int durability;

protected Car(string brand, string model, int yearOfProduction, int horsepower, int acceleration, int suspension, int durability)

{

this.brand = brand;

this.model = model;

this.yearOfProduction = yearOfProduction;

this.horsepower = horsepower;

this.acceleration = acceleration;

this.suspension = suspension;

this.durability = durability;

}

}

**Добавяме базов конструктор от наследяването на ShowCar и PerformanceCar и в CarManger пишем:**

if (type== "Performance")

{

this.cars.Add(id,new PerformanceCar(brand,model,yearOfProduction,horsepower,acceleration,suspension,durability));

}

else

{

this.cars.Add(id,new ShowCar(brand, model, yearOfProduction, horsepower, acceleration, suspension, durability));

}

using System.Collections.Generic;

public class PerformanceCar:Car

{

private List<string> addOns;

public PerformanceCar(string brand, string model, int yearOfProduction, int horsepower, int acceleration, int suspension, int durability)

: base(brand, model, yearOfProduction, horsepower, acceleration, suspension, durability)

{

this.addOns=new List<string>();

}

}

public class ShowCar:Car

{

private int stars;

public ShowCar(string brand, string model, int yearOfProduction, int horsepower, int acceleration, int suspension, int durability)

: base(brand, model, yearOfProduction, horsepower, acceleration, suspension, durability)

{

}

}

**9.За да тествам метода регистер,трябва да направя метода check:**

public string Check(int id)

{

return cars[id].ToString();

}

**Правя override на базовият клас Car:**

public override string ToString()

{

StringBuilder sb=new StringBuilder();

sb.AppendLine($"{brand} {model} {yearOfProduction}")

.AppendLine($"{horsepower} HP, 100 m/h in {acceleration} s")

.AppendLine($"{suspension} Suspension force, {durability} Durability");

return sb.ToString();

}

**След това override и на PerformanceCar :**

public override string ToString()

{

StringBuilder sb=new StringBuilder(base.ToString());

if (this.addOns.Count>0)

{

sb.AppendLine($"Add-ons: " + string.Join(", ", this.addOns));

}

else

{

sb.AppendLine("Add-ons: None");

}

return sb.ToString().Trim();

}

**След това override и на ShowCar :**

public override string ToString()

{

StringBuilder sb = new StringBuilder(base.ToString());

sb.AppendLine($"{stars} \*");

return sb.ToString().Trim();

}

В **Engine пишем:**

Console.WriteLine(manager.Check(int.Parse(cmdArgs[1])));

**И тестваме някой пример**

**10.Правим метода за Open.В Engine декларираме стойностите**

id = int.Parse(cmdArgs[1]);

type = cmdArgs[2];

int length = int.Parse(cmdArgs[3]);

string route = cmdArgs[4];

int prizePool = int.Parse(cmdArgs[5]);

manager.Open(id,type,length,route,prizePool);

**и отваряме с F12 manager.Open:**

public void Open(int id, string type, int length, string route, int prizePool)

{

switch (type)

{

case "Casual":

this.races.Add(id,new CasualRace());

break;

case "Drag":

this.races.Add(id, new DragRace());

break;

case "Drift":

this.races.Add(id, new DriftRace());

break;

}

}

**Отиваме в Race и правим конструктор и пропертита:**

using System.Collections.Generic;

public abstract class Race

{

public int Length { get; set; }

public string Route { get; set; }

public int PrizePool { get; set; }

public List<Car> Participants { get; set; }

protected Race(int length, string route, int prizePool)

{

Length = length;

Route = route;

PrizePool = prizePool;

Participants = new List<Car>();

}

}

Накрая добавяме липсващият конструктор към 3-те вида състезания.

public class CasualRace:Race

{

public CasualRace(int length, string route, int prizePool)

: base(length, route, prizePool)

{

}

}