# 03. Street Racing

*You and your friends decide to organize some illegal street races. Since you are a programmer, you have the task to develop software for organizing safe and fair races.*

## Preparation

Download the skeleton provided in Judge. **Do not** change the **StartUp** class or its **namespace**.

## Problem description

Your task is to create a race in which participate different cars.

### Car

First, write a C# **class**, called **Car** with properties:

* **Make: string**
* **Model: string**
* **LicensePlate: string**
* **HorsePower: int**
* **Weight: double**

The **constructor** of Car class should receive **make, model, licensePlate, horsePower and weight**.

The class should also have the following methods:

* Override **ToString()** method in the format:

**"****Make: {Make}**

**Model: {Model}**

**License Plate: {LicensePlate}**

**Horse Power: {HorsePower}**

**Weight: {Weight}"**

### Race

Next step is to write **Race** class that has a **collection** of type **Car** with corresponding **unique** **License Plate** of a Car. The name of the collection should be **Participants**. All the entities of the **Participants** collection have the **same** properties. The **Race** has also some additional properties:

* **Name: string**
* **Type: string**
* **Laps: int**
* **Capacity: int -** the maximum allowed number of participants in the race
* **MaxHorsePower: int -** the maximum allowed Horse Power of a Car in the Race

The **constructor** of the Race class should receive **name, type, laps, capacity** and **maxHorsePower**.

Implement the coming features:

* Getter **Count** - returns the count of the currently enrolled participants
* Method **Add(Car car)** - adds the entity **if** there **isn't** a Car with the same **License plate** and **if** there is enough space in terms of race **capacity** and if the car meets the **maximum horse power** requirment of the race.
* Method **Remove(string licensePlate)** - removes a Car from the race with the given **License plate**, if such **exists** and returns **bool** if the deletion is successful.
* Method **FindParticipant(string licensePlate)** - returns a **Car** with the given License plate. If it doesn't exist, return **null**.
* Method **GetMostPowerfulCar() –** returns the **Car** with most **HorsePower**. If there are no Cars in the Race, method should return null.
* Method **Report()** - returns information about the Race and the Cars participating it in the following format:

**"****Race: {Name} - Type: {Type} (Laps: {Laps})**

**{Car1}**

**{Car2}**

**… "**

## Constraints

* The License plate of each Car in the race will always be unique
* The HorsePower of a Car and the MaxHorsePower of the Race will always be positive numbers
* There will not be a case where two Cars have the same HorsePower
* You will always be given Car added before receiving method for its manipulation

### Examples

|  |
| --- |
| Sample code usage |
| //Sample Code Usage:  //Initialize Race  Race race = new Race("RockPort Race", "Sprint", 1, 4, 150);  //Initialize Car  Car car = new Car("BMW", "320ci", "NFS2005", 150, 1450);  //Print car  Console.WriteLine(car.ToString());  //Make: BMW  //Model: 320ci  //License Plate: NFS2005  //Horse Power: 150  //Weight: 1450  //Add car  race.Add(car);  //Remove car  Console.WriteLine(race.Remove("NFS2005")); // True  Car carOne = new Car("BMW", "320cd", "NFS2007", 150, 1350);  Car carTwo = new Car("Audi", "A3", "NFS2004", 131, 1300);  //Add cars  race.Add(carOne);  race.Add(carTwo);  //GetMostPowerfulCar  Console.WriteLine(race.GetMostPowerfulCar());  //Make: BMW  //Model: 320cd  //License Plate: NFS2007  //Horse Power: 150  //Weight: 1350  //Print Race report  Console.WriteLine(race.Report());  //Race: RockPort Race - Type: Sprint (Laps: 1)  //Make: BMW  //Model: 320cd  //License Plate: NFS2007  //Horse Power: 150  //Weight: 1350  //Make: Audi  //Model: A3  //License Plate: NFS2004  //Horse Power: 131  //Weight: 1300 |

## Submission

Zip all the files in the project folder except **bin** and **obj** folders.