# The Race

*Your friend Vin Benzin decided to organize a real street race. He called you and asked you to develop a platform where the racers can sign up.*

## Preparation

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

## Problem description

Your task is to create a repository, which stores departments by creating the classes described below.

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

* **Name: string**
* **Speed: int**

The class **constructor** should receive **name,** and **speed.**

Next, write a C# class **Racer** with the following properties:

**Name: string**

**Age: int**

**Country: string**

**Car: Car**

The class **constructor** should receive **name, age, country and Car** and override the **ToString()** method in the following format:

**"Racer: {name}, {age} ({country})"**

**Next**, write a C# class **Race** that has **data** (a collection, which stores the entity **Racer**). All entities inside the repository have the **same properties**. Also, the Race class should have those properties:

* **Name: string**
* **Capacity: int**

The class **constructor** should receive **name** and **capacity** (the maximum allowed number of racers), also it should initialize the **data** with a new instance of the collection**.** Implement the following features:

* Field **data** – **collection** that holds added Racers
* Method Add(Racer Racer) – **adds** an **entity** to the data **if** **there** **is** **room** for him/her.
* Method Remove(string name) – removes a Racer by **given name,** if such **exists**, and **returns bool**.
* Method GetOldestRacer() – returns the **oldest** Racer.
* Method **GetRacer(string name)** – returns the Racer with the **given name**.
* Method **GetFastestRacer()** – returns the Racer whose car has the highest speed.
* Getter Count – **returns** the **number** of Racers.
* **Report()** – **returns** a **string** in the following **format**:
  + **"Racers participating at {RaceName}:  
    {Racer1}  
    {Racer2}  
    (…)**"

## Constraints

* The **names** of the Racers will be **always unique**.
* The **age** of the Racers will always be with **positive values**.
* You will always have a Racer added before receiving methods manipulating the Race’s data.

## Examples

This is an example how the **Race** class is **intended to be used**.

|  |
| --- |
| Sample code usage |
| //Initialize the repository  Race race = new Race("Indianapolis 500", 10);  //Initialzie cars  Car car1 = new Car("ferrari", 150);  Car car2 = new Car("lambo", 170);  //Initialize racer1  Racer racer1 = new Racer("Stephen", 40, "Bulgaria",car1);  //Print Racer  Console.WriteLine(racer1); //Racer: Stephen, 40 (Bulgaria)  //Add Racer  race.Add(racer1);  //Remove Racer  race.Remove("Vin Benzin"); //false  Racer racer2 = new Racer("Mark", 34, "UK",car2);  //Add Racer  race.Add(racer2);  Racer oldestRacer = race.GetOldestRacer(); // Racer with name Stephen  Racer racerStephen = race.GetRacer("Stephen"); // Racer with name Stephen  Racer fastestRacer = race.GetFastestRacer(); // Racer with name Mark  Console.WriteLine(oldestRacer); //Racer: Stephen, 40 (Bulgaria)  Console.WriteLine(racerStephen); //Racer: Stephen, 40 (Bulgaria)  Console.WriteLine(fastestRacer); // Racer: Mark, 34 (UK)  Console.WriteLine(race.Count); //2  Console.WriteLine(race.Report());  //Racers working at Indianapolis 500:  //Racer: Stephen, 40 (Bulgaria)  //Racer: Mark, 34 (UK) |

## Submission

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