# SoftUni Parking

## 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 cars by creating the classes described below.

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

* **Make: string**
* **Model: string**
* **HorsePower: int**
* **RegistrationNumber: string**

|  |
| --- |
| **public class** Car  {   *//* ***TODO: implement this class*** } |

The class **constructor** should receive **make, model, horsePower and registrationNumber** and override the **ToString()** method in the following format:

**"Make: {make}"**

**"Model: {model}"**

**"HorsePower: {horse power}"**

**"RegistrationNumber: {registration number}"**

Write a C# class **Parking** that has **cars** (a collection which stores the entity **Car**). All entities inside the class have the **same properties**.

|  |
| --- |
| **public class** Parking  {  *//* ***TODO: implement this class*** } |

The class **constructor** should initialize the **cars** with a new instance of the collection and accept capacity as parameter**.** Implement the following features:

* Field **cars** – **collection** that holds added cars
* Field **capacity** – accessed only by the base class (responsible for the parking capacity)
* Method AddCar(Car car) – first checks if there is already a car with tha provided car registration number and if there is the method returns the following message:

"Car with that registration number, already exists!"

Next checks if the count of the cars in the parking is more than the capacity and if it is returns the following message:

"Parking is full!"

Finally if nothing from the previous conditions is true it just adds the current car to the cars in the parking and returns the message:

"Successfully added new car {Make} {RegistrationNumber}"

* Method RemoveCar(string registrationNumber) – removes a car with the givven registration number. If the provided registration number does not exist returns the message:

"Car with that registration number, doesn't exists!"

Otherwise, removes the car and returns the message:

"Successfully removed {registrationNumber}"

* Method GetCar(string registrationNumber) – returns the **Car** with the provided registration number
* Method RemoveSetOfRegistrationNumber(List<string> registrationNumbers) – removes all cars having the provided registration numbers and procceeds the same way as the **RemoveCar()** method
* Getter Count – returns the number of stored cars.

## Examples

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

|  |
| --- |
| Sample code usage |
| var car = new Car("Skoda", "Fabia", 65, "CC1856BG");  var car2 = new Car("Audi", "A3", 110, "EB8787MN");  Console.WriteLine(car.ToString());  //Make: Skoda  //Model: Fabia  //HorsePower: 65  //RegistrationNumber: CC1856BG  var parking = new Parking(5);  Console.WriteLine(parking.AddCar(car));  //Successfully added new car Skoda CC1856BG  Console.WriteLine(parking.AddCar(car));  //Car with that registration number, already exists!  Console.WriteLine(parking.AddCar(car2));  //Successfully added new car Audi EB8787MN  Console.WriteLine(parking.GetCar("EB8787MN").ToString());  //Make: Audi  //Model: A3  //HorsePower: 110  //RegistrationNumber: EB8787MN    Console.WriteLine(parking.RemoveCar("EB8787MN"));  //Successfullyremoved EB8787MN  Console.WriteLine(parking.Count); //1 |

## Submission

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