# Ski Rental

*Summer has come! Your friend who owns a Ski Rental business asked you to write a program to help him organize his ski storage in order to be prepared for the coming winter season.*

## 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 items by creating the classes described below.

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

* **Manufacturer: string**
* **Model: string**
* **Year: int**

The class **constructor** should receive the **manufacturer, model** and **year** and override the **ToString()** method in the following format:

**"****{manufacturer} - {model} - {year}"**

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

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

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

* Field **data** – **collection** that holds added Skis
* Method Add(Ski ski) – **adds** an **entity** to the data **if** **there** **is** an **empty slot** for the Ski.
* Method Remove(string manufacturer, string model) – removes the Ski by **given manufacturer and model,** if such **exists**, and **returns a bool**.
* Method GetNewestSki() – returns the newest Ski (by year) or null if there are no Skis stored.
* Method **GetSki(string manufacturer, string model)** – returns the Ski with the **given manufacturer** and **model** or null if there is no such Ski.
* Getter Count – **returns** the **number** of Skis.
* **GetStatistics()** – **returns** a **string** in the following **format**:
  + **"****The skis stored in {Ski Rental Name}:  
    {Ski1}  
    {Ski2}  
    (…)**"

## Constraints

* The **combinations** of **manufacturers** and **models** will be **always unique**.
* The **year** of the Skis will always be **positive**.
* There won't be Skis with the same years.

## Examples

This is an example of how the **Ski Rental** class is **intended to be used**.

|  |
| --- |
| Sample code usage |
| // Initialize the repository  SkiRental skiRental = new SkiRental("New Alpine ski rental", 5);  // Initialize entity  Ski firstSkiSet = new Ski("Rossignol", "XC70", 2017);  // Print Ski  Console.WriteLine(firstSkiSet); // Rossignol - XC70 - 2017  // Add Ski  skiRental.Add(firstSkiSet);  // Remove Ski  Console.WriteLine(skiRental.Remove("Rossignol", "XC90")); // False  Console.WriteLine(skiRental.Remove("Rossignol", "XC70")); // True  Ski secondSkiSet = new Ski("Fischer", "SpeedMax", 2018);  Ski thirdSkiSet = new Ski("Salomon", "TopLine", 2020);  skiRental.Add(secondSkiSet);  skiRental.Add(thirdSkiSet);  // Get Newest Ski  Ski newestSki = skiRental.GetNewestSki();  Console.WriteLine(newestSki); // Salomon - TopLine - 2020  // Get Ski  Ski salomonTopLine = skiRental.GetSki("Salomon", "TopLine");  Console.WriteLine(salomonTopLine); // Salomon - TopLine - 2020  // Count  Console.WriteLine(skiRental.Count); // 2  // Get Statistics  Console.WriteLine(skiRental.GetStatistics());  // The skis stored in New Alpine ski rental:  // Fischer - SpeedMax – 2018  // Salomon - TopLine - 2020 |

## Submission

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