# Renovators

*Despite your efforts and directions, Vanko totally destroyed Victoria's wall. Now she wants to choose another handyman, but this time she'll be more careful. That's why she asked you to help her and create a catalog with different renovators and their info.*

## Preparation

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

**Note**: The target framework of your project must be .NET Core 3.1.

## Problem description

Your task is to create a catalog, containing information for various renovators.

### Renovator

You are given a class **Renovator,** create the following fields:

* **Name: string**
* **Type: string**
* **Rate: double**
* **Days: int**
* **Hired: boolean - false by default**

The class **constructor** should receive **(name, type, rate, days)**.

The class should also have a method:

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

**"-Renovator: {name}**

**--Specialty: {type}**

**--Rate per day: {rate} BGN"**

### Catalog

Next, a class named **Catalog**is given that has a **collection**(**renovators**) of type **Renovator**. All the entities of the **renovators** collection have the **same** properties. The **Catalog** has also some additional properties:

* **Name: string**
* **NeededRenovators: int**
* **Project: string**

The **constructor** of the **Catalog** class should receive the **name, neededRenovators** and **project.**

Implement the following features:

* Getter **Count** - returns the count of the renovators in the catalog.
* string AddRenovator(Renovator renovator) - **adds** a renovator to the catalog's collection, **if** **renovators are still needed**. Before adding a renovator, check:
  + - * + If the **name** or **specialty** are **null or empty**, return **"Invalid renovator's information.".**
  + If renovators are no more needed, return "**Renovators are no more needed.". Renovators are needed when the count of the added renovators is less than the NeededRenovators property of the Catalog.**
  + If the **rate** is **above 350 BGN**,return **"Invalid renovator's rate.".**
  + Otherwise, return: **"Successfully added {renovatorName} to the catalog.".**
* bool RemoveRenovator(string name) - removes a renovator by a **given name.**
  + - If such **exists returns true**;
    - Otherwise, returns **false.**
* int RemoveRenovatorBySpecialty(string type) - removes **all renovators** by the given **specialty.**
  + - If such **exist(s), return the count of the removed renovators;**
    - Otherwise, returns **0.**
* Renovator HireRenovator(string name) method – **hire** (**set** their available **property** to **true** without removing them from the collection) the **renovator** with the **given name, if they exist**. As a result, **return** the **renovator, or null, if they don't exist.**
* List<Renovator> PayRenovators (int days) method – **return** a **list** with **all renovators** that have been working for **days** or more.
* **Report()** –returns a string with information about the catalog and renovators who are **not hired** in the following format:

**"**Renovators **available for Project {project}:  
{**Renovator**1}  
{**Renovator**2}  
{…}**"

**Note: Do not use** "\n\r" **for a new line.**

## Constraints

* The **names** of the renovators will be **always unique**.
* You will always have a renovator added before receiving methods manipulating the catalogs' renovators.

## Examples

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

|  |
| --- |
| Sample code usage |
| // Initialize the repository (Catalog)  Catalog catalog = new Catalog("Quality renovators", 5, "Kitchen");  // Initialize entity  Renovator renovator = new Renovator("Gosho", "Painter", 270, 7);  //Print Renovator  Console.WriteLine(renovator);  /\*  -Renovator: Gosho  --Specialty: Painter  --Rate per day: 270 BGN  \*/  // Add Renovator  Console.WriteLine(catalog.AddRenovator(renovator)); // Successfully added Gosho to the catalog.  Console.WriteLine(catalog.Count); // 1  // Remove Renovator  Console.WriteLine(catalog.RemoveRenovator("Pesho")); // False  Renovator secondRenovator = new Renovator("Pesho", "Tiles", 200, 9);  Renovator thirdRenovator = new Renovator("Ivan", "Tiles", 450, 7);  Renovator fourthRenovator = new Renovator("Velichko", "Painter", 120, 3);  Renovator fifthRenovator = new Renovator("Stamat", "Furniture", 300, 4);  Renovator sixthRenovator = new Renovator("Genadi", "Furniture", 80, 15);  Renovator seventhRenovator = new Renovator("Unufri", "Furniture", 80, 15);  // Add Renovators  Console.WriteLine(catalog.AddRenovator(secondRenovator)); // Successfully added Pesho to the catalog.  Console.WriteLine(catalog.AddRenovator(thirdRenovator)); // Invalid renovator's rate.  Console.WriteLine(catalog.AddRenovator(fourthRenovator)); // Successfully added Velichko to the catalog.  Console.WriteLine(catalog.AddRenovator(fifthRenovator)); // Successfully added Stamat to the catalog.  Console.WriteLine(catalog.AddRenovator(sixthRenovator)); // Successfully added Genadi to the catalog.  Console.WriteLine(catalog.AddRenovator(seventhRenovator)); // Renovators are no more needed.  // Hire renovator by name  Console.WriteLine(catalog.HireRenovator("Genadi"));  /\*  -Renovator: Genadi  --Specialty: Furniture  --Rate per day: 80 BGN  \*/  // Pay renovators  List<Renovator> renovators = catalog.PayRenovators(8);  foreach (var renovatorToBePaid in renovators)  {  Console.WriteLine(renovatorToBePaid);  }  /\*  -Renovator: Pesho  --Specialty: Tiles  --Rate per day: 200 BGN  -Renovator: Genadi  --Specialty: Furniture  --Rate per day: 80 BGN  \*/  // Remove renovators by specialty  Console.WriteLine(catalog.RemoveRenovatorBySpecialty("Painter")); // 2  Console.WriteLine("----------------------Report----------------------");  Console.WriteLine(catalog.Report());  /\*  Renovators available for Project Kitchen:  -Renovator: Pesho  --Specialty: Tiles  --Rate per day: 200 BGN  -Renovator: Stamat  --Specialty: Furniture  --Rate per day: 300 BGN  \*/ |

## Submission

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