

Space Station Recruitment



Now that Stephen successfully established his own Space Station, he has to recruit some astronauts to work there. You are going to help him by building a system for that.

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 **Astronaut** with the following properties:

- **Name:** string
- **Age:** int
- **Country:** string

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

"Astronaut: {name}, {age} ({country})"

Next, write a C# class **SpaceStation** that has **data** (a collection, which stores the entity **Astronaut**). All entities inside the repository have the **same properties**. Also, the **SpaceStation** 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 astronauts
- Method **Add(Astronaut astronaut)** - **adds** an **entity** to the data **if there is room** for him/her.
- Method **Remove(string name)** - removes an astronaut by **given name**, if such **exists**, and **returns bool**.
- Method **GetOldestAstronaut()** - returns the **oldest** astronaut.



- Method **GetAstronaut(string name)** – returns the astronaut with the **given name**.
- Getter **Count** – returns the **number** of astronauts.
- **Report()** – returns a **string** in the following **format**:
 - "Astronauts working at Space Station {spaceStationName}:
 {Astronaut1}
 {Astronaut2}
 (...)"

Constraints

- The **names** of the astronauts will be **always unique**.
- The **age** of the astronauts will always be with **positive values**.
- You will always have an astronaut added before receiving methods manipulating the Space Station's astronauts.

Examples

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

Sample code usage

```
//Initialize the repository
SpaceStation spaceStation = new SpaceStation("Apolo", 10);
//Initialize entity
Astronaut astronaut = new Astronaut("Stephen", 40, "Bulgaria");
//Print Astronaut
Console.WriteLine(astronaut); //Astronaut: Stephen, 40 (Bulgaria)

//Add Astronaut
spaceStation.Add(astronaut);
//Remove Astronaut
spaceStation.Remove("Astronaut name"); //false

Astronaut secondAstronaut = new Astronaut("Mark", 34, "UK");

//Add Astronaut
spaceStation.Add(secondAstronaut);

Astronaut oldestAstronaut = spaceStation.GetOldestAstronaut(); // Astronaut with name Stephen
Astronaut astronautStephen = spaceStation.GetAstronaut("Stephen"); // Astronaut with name Stephen
Console.WriteLine(oldestAstronaut); //Astronaut: Stephen, 40 (Bulgaria)
Console.WriteLine(astronautStephen); //Astronaut: Stephen, 40 (Bulgaria)

Console.WriteLine(spaceStation.Count); //2

Console.WriteLine(spaceStation.Report());
//Astronauts working at Space Station Apolo:
//Astronaut: Stephen, 40 (Bulgaria)
//Astronaut: Mark, 34 (UK)
```

Submission

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

