# Guild

## 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 players by creating the classes described below.

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

* **Name: string**
* **Class: string**
* **Rank: string – "Trial" by default**
* **Description: string – "n/a" by default**

The class **constructor** should receive **name and class**. Override the **ToString()** method in the following format:

**"Player {Name}: {Class}**

**Rank: {Rank}**

**Description: {Description}"**

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

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

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

* Field **roster** - **collection** that holds added **players**
* Method AddPlayer(Player player) - **adds** an **entity** to the roster **if** **there** **is** **room** for it
* Method RemovePlayer(string name) - removes a player by **given name,** if such **exists**, and **returns bool**
* Method PromotePlayer(string name) - **promote** (**set his rank to "Member"**) the **first player** with the **given name.** If the player is **already** a "Member", **do nothing.**
* Method **DemotePlayer(string name)- demote (set his rank to "Trial")** the first player with the **given** name. If the player is **already** a "Trial",  **do nothing.**
* Method KickPlayersByClass(string class) - removes all the players by the given class and returns **all players** from that **class as an array**
* Getter Count - **returns** the **number** of players
* **Report()** - **returns** a **string** in the following **format:**
  + **"Players in the guild: {guildName}  
    {Player1}  
    {Player2}  
    (…)**"

## Constraints

* The **names** of the players will be **always unique**.
* You will always have a player added before receiving methods manipulating the Guild's players.

## Examples

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

|  |
| --- |
| Sample code usage |
| //Initialize the repository (guild)  Guild guild = new Guild("Weekend Raiders", 20);  //Initialize entity  Player player = new Player("Mark", "Rogue");  //Print player  Console.WriteLine(player); //Player Mark: Rogue  //  //Rank: Trial  //Description: n/a  //Add player  guild.AddPlayer(player);  Console.WriteLine(guild.Count); //1  Console.WriteLine(guild.RemovePlayer("Gosho")); //False  Player firstPlayer = new Player("Pep", "Warrior");  Player secondPlayer = new Player("Lizzy", "Priest");  Player thirdPlayer = new Player("Mike", "Rogue");  Player fourthPlayer = new Player("Marlin", "Mage");  //Add description to player  secondPlayer.Description = "Best healer EU";  //Add players  guild.AddPlayer(firstPlayer);  guild.AddPlayer(secondPlayer);  guild.AddPlayer(thirdPlayer);  guild.AddPlayer(fourthPlayer);  //Promote player  guild.PromotePlayer("Lizzy");  //RemovePlayer  Console.WriteLine(guild.RemovePlayer("Pep")); //True  Player[] kickedPlayers = guild.KickPlayersByClass("Rogue");  Console.WriteLine(string.Join(", ", kickedPlayers.Select(p => p.Name))); //Mark, Mike  Console.WriteLine(guild.Report());  //Players in the guild: Weekend Raiders  //Player Lizzy: Priest  //Rank: Member  //Description: Best healer EU  //Player Marlin: Mage  //Rank: Trial  //Description: n/a |

## Submission

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