# Openning



*Now that you successfully saved money for your own Bakery ,you need to recruit some employees to work there. You are You should build 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 **Employee** 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:

**"Employee: {name}, {age} ({country})"**

**Next**, write a C# class **Bakery** that has **data** (a collection, which stores the entity **Employee**). All entities inside the repository have the **same properties**. Also, the Bakery 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 Employees
* Method Add(Employee employee) – **adds** an **entity** to the data **if** **there** **is** **room** for him/her.
* Method Remove(string name) – removes an employee by **given name,** if such **exists**, and **returns bool**.
* Method GetOldestEmployee() – returns the **oldest** employee.
* Method **GetEmployee(string name)** – returns the employee with the **given name**.
* Getter Count – **returns** the **number** of employees.
* **Report()** – **returns** a **string** in the following **format**:
  + **"Employees working at Bakery {bakeryName}:  
    {Employee1}  
    {Employee2}  
    (…)**"

## Constraints

* The **names** of the employees will be **always unique**.
* The **age** of the employees will always be with **positive values**.
* You will always have an employee added before receiving methods manipulating the Space Station’s Employees.

## Examples

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

|  |
| --- |
| Sample code usage |
| //Initialize the repository  Bakery bakery = new Bakery("Barny", 10);  //Initialize entity  Employee employee = new Employee("Stephen", 40, "Bulgaria");  //Print Employee  Console.WriteLine(employee); //Employee: Stephen, 40 (Bulgaria)  //Add Employee  bakery.Add(employee);  //Remove Employee  Console.WriteLine(bakery.Remove("Employee name")); //false  Employee secondEmployee = new Employee("Mark", 34, "UK");  //Add Employee  bakery.Add(secondEmployee);  Employee oldestEmployee = bakery.GetOldestEmployee(); // Employee with name Stephen  Employee employeeStephen = bakery.GetEmployee("Stephen"); // Employee with name Stephen  Console.WriteLine(oldestEmployee); //Employee: Stephen, 40 (Bulgaria)  Console.WriteLine(employeeStephen); //Employee: Stephen, 40 (Bulgaria)  Console.WriteLine(bakery.Count); //2  Console.WriteLine(bakery.Report());  //Employees working at Bakery Barny:  //Employee: Stephen, 40 (Bulgaria)  //Employee: Mark, 34 (UK) |

## Submission

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