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 **packages**!

Pay attention to name the package bakery, all the classes, their fields and methods the same way they are presented in the following document. It is also important to keep the project structure as described.

Problem description

Your task is to create a bakery, which stores employees by creating the classes described below.

First, write a Java 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 Java class **`Bakery`** that has **`employees`** (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 **employees** with a new instance of the collection**.** Implement the following features:

- Field **`employees`** – **List** 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 **`getCount()`** – **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
System.out.println(employee); //Employee: Stephen, 40 (Bulgaria)

//Add Employee
bakery.add(employee);
//Remove Employee
System.out.println(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
System.out.println(oldestEmployee); //Employee: Stephen, 40 (Bulgaria)
System.out.println(employeeStephen); //Employee: Stephen, 40 (Bulgaria)

System.out.println(bakery.getCount()); //2

System.out.println(bakery.report());
//Employees working at Bakery Barny:
//Employee: Stephen, 40 (Bulgaria)
//Employee: Mark, 34 (UK)
```

Submission

Submit **single .zip file**, containing **bakery package, with the classes inside (Employee, Bakery and the Main class)**, there is no specific content required inside the Main class e. g. you can do any kind of local testing of you program there. However there should be **main(String[] args)** method inside.