# Café

# 

*Now that you successfully saved money for your Café, 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 the name of the package cafe, 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 cafe, 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} from {country}"**

**Next**, write a Java class **Cafe** that has **employees** (a collection, which stores the entity **Employee**). All entities inside the repository have the **same properties**. Also, the Cafe class should have those properties:

* **name: String**
* **capacity: int**

The class **constructor** should receive the **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 addEmployee(Employee employee) – **adds** an **entity** to the data **if** **there** **is** **room** for him/her.
* Method removeEmployee(String name) – removes an employee by **given name,** if such **exists**, and **returns** a **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 Cafe {cafeName}:  
    {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 Employees.

## Examples

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

|  |
| --- |
| Sample code usage |
| //Initialize the repository  Cafe cafe = new Cafe("Costa", 15);  //Initialize entity  Employee employee = new Employee("Alexander", 30, "Bulgaria");  //Print Employee  System.*out*.println(employee); // Employee: Alexander, 30 from Bulgaria  //Add Employee  cafe.addEmployee(employee);  //Remove Employee  System.*out*.println(cafe.removeEmployee("Employee")); //false  Employee secondEmployee = new Employee("Sara", 24, "UK");  Employee thirdEmployee = new Employee("Anna", 22, "Germany");  //Add Employee  cafe.addEmployee(secondEmployee);  cafe.addEmployee(thirdEmployee);  Employee oldestEmployee = cafe.getOldestEmployee();  Employee employeeStephen = cafe.getEmployee("Sara");  System.*out*.println(oldestEmployee); // Employee: Alexander, 30 from Bulgaria System.*out*.println(employeeStephen); //Employee: Sara, 24 from UK  System.*out*.println(cafe.getCount()); // 3  System.out.println(cafe.removeEmployee("Anna")); // true  System.*out*.println(cafe.report());  //Employees working at Cafe Costa:  //Employee: Alexander, 30 from Bulgaria  //Employee: Sara, 24 from UK |

## Submission

Submit a **single .zip file**, containing the **cafe package, with the classes inside (Employee, Cafe, 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 your program there. However, there should be a **main(String[] args)** method inside.