# 03. Hotel



## Preparation

Download the skeleton provided in Judge. **Do not** change the **packages**!

**Pay attention to the name of the package hotel, 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 repository that stores people by creating the classes described below.

**Person**

First, write a Java class **Person** with the following fields:

* **name: String**
* **id: int**
* **age: int**
* **hometown: String – "n/a" by default**

The class **constructor** should receive the **name, id, age, and hometown**. You need to create the appropriate **getters and setters**. Override the **toString()** method in the following format:

**"Person {name}: {id}, Age: {age}, Hometown: {hometown}"**

**Hotel**

**Next**, write a Java class **Hotel** that has **a roster** (a collection that stores **Person** entities). All entities inside the repository have the **same fields**. Also, the **Hotel** class should have those **fields**:

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

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

* Method add(Person person) - **adds** an **entity** to the roster **if** **there** **is** **room** for it
* Method remove(String name) - removes a person by **given name,** if such **exists**, and **returns boolean**
* Method **getPerson(String name, String hometown)** – returns the people with the **given name** and **hometown** or **null** if there is no such person.
* Getter getCount() – **returns** the **number** of people.
* **getStatistics()** – **returns** a **String** in the following **format**:
  + **"The people in the hotel {name} are:  
    {Person1}  
    {Person2}  
    (…)**"

## Constraints

* The **names** of the people will be **always unique**.
* You will always have a person added before receiving methods manipulating the Hotel's people.

## Examples

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

|  |
| --- |
| Sample code usage |
| package hotel;  public class Main {  public static void main(String[] args) {  //Initialize the repository (hotel)  Hotel hotel = new Hotel("Hilton", 30);  //Initialize entity  Person person = new Person("Mark", 11111, 41, "Sofia");  //Print person  System.*out*.println(person);  //Person Mark: 11111, Age: 41, Hometown: Sofia   //Add person  hotel.add(person);  System.*out*.println(hotel.getCount()); //1  System.*out*.println(hotel.remove("Anna")); //false   Person firstPerson = new Person("Alice", 22121, 18, "London");  Person secondPerson= new Person("Lizzy", 31311, 24, "Varna");  Person thirdPerson = new Person("Lucy", 54122, 31, "Birmingham");  Person fourthPerson = new Person("Maria", 66611, 41, "Sofia");    //Add people  hotel.add(firstPerson);  hotel.add(secondPerson);  hotel.add(thirdPerson);  hotel.add(fourthPerson);  // Get person  Person personForGet = hotel.getPerson("Lucy", "Birmingham");  Person personForGet1 = hotel.getPerson("Anna", "Burgas");  System.out.println(personForGet); // Person Lucy: 54122, Age: 31, Hometown: Birmingham  System.out.println(personForGet1); // null  //Count  System.out.println(hotel.getCount()); // 5  //Remove Person  System.*out*.println(hotel.remove("Alice")); //true  // Get Statistics  System.out.println(hotel.getStatistics());  // The people in the hotel Hilton are:  // Person Mark: 11111, Age: 41, Hometown: Sofia  // Person Lizzy: 31311, Age: 24, Hometown: Varna  // Person Lucy: 54122, Age: 31, Hometown: Birmingham  // Person Maria: 66611, Age: 41, Hometown: Sofia  } } |

## Submission

Submit a **single .zip file**, containing the **hotel package, with the classes inside (Person, Hotel, 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.