Grooming Salon

Preparation

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

Pay attention to name the package groomingSalon, 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, which stores items by creating the classes described below.

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

- **`name: String`**
- **`age: int`**
- **`owner: String`**

The class **constructor** should receive **all fields.** You need to create the appropriate **getters and setters**. The class should override the **`toString()`** method in the following format:

`"{name} {age} - ({owner})"`

Next, write a Java class **`GroomingSalon`** that has **`data`** (a collection, which stores the Pets). All entities inside the repository have the **same fields**. Also, the **`GroomingSalon`** class should have those fields:

- **`capacity: int`**

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

- Field **`data`** – **`List`** that holds added pets
- Method **`add(Pet pet)`** – **adds** an **entity** to the data **if there is** an **empty place** in the grooming salon for the pet.
- Method **`remove(String name)`** – removes the pet by **given name,** if such **exists**, and **returns boolean**.
- Method **`getPet(String name, String owner)`** – returns the pet with the **given name** and **owner** or **`null`** **if no such pet exists**.
- Getter **`getCount`** – **returns** the **number** of pets.
- **`getStatistics()`** – **returns** a **String** in the following **format**:
    - **`" The grooming salon has the following clients:`**
      `{name} {owner}`
      `{name} {owner}`

(…)"

Constraints

- The **combinations** of **names** and **owners** will **always be unique**.
- The **age** of the pets will always be **positive**.

Examples

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

Sample code usage

```
// Initialize the repository
GroomingSalon salon = new GroomingSalon(20);

// Initialize entity
Pet dog = new Pet("Ellias", 5, "Tim");

// Print Pet
System.out.println(dog); // Ellias 5 - (Tim)

// Add Pet
salon.add(dog);

// Remove Pet
System.out.println(salon.remove("Ellias")); // true
System.out.println(salon.remove("Pufa")); // false

Pet cat = new Pet("Bella", 2, "Mia");
Pet bunny = new Pet("Zak", 4, "Jon");

salon.add(cat);
salon.add(bunny);

// Get Pet
Pet pet = salon.getPet("Bella", "Mia");
System.out.println(pet); // Bella 2 - (Mia)

// Count
System.out.println(salon.getCount()); // 2

// Get Statistics
System.out.println(salon.getStatistics());
// The grooming salon has the following clients:
//Bella Mia
//Zak Jon
```

Submission

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

Submit **single .zip file**, containing **groomingSalon package, with the classes inside (Pet, GroomingSalon 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.