Rabbits

Preparation

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

Pay attention to name the package (rabbits), 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 departments by creating the classes described below.

Rabbit

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

- **name: String**
- **species: String**
- **available: boolean - true by default**

The class **constructor** should receive **(name, species)**.

The class should also have the following methods:

- **`getName()`**
- **`getSpecies()`**
- **`isAvailable()`**
- **`setAvailable()`**
- Override the **`toString()`** method in the format:

  `"Rabbit ({species}): {name}"`

Cage

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

- **name: String**
- **capacity: int**
- **data**: **List<Rabbit>** that holds added rabbits

The class **constructor** should receive **(name**, **capacity)**, also it should initialize the **data** with a new instance of the collection.

Implement the following features:

- **`getName()`**
- **`getCapacity()`**
- **`add(Rabbit rabbit)`** method - **adds** an **entity** to the data **if there is room** for it
- **`removeRabbit(String name)`** method - removes a rabbit by **given name,** if such **exists**, and **returns boolean**
- **`removeSpecies(String species)`** method - removes **all rabbits** by given **species**
- **`sellRabbit(String name)`** method - **sell** (**set** its **`available`** **property** to **false** without removing it from the collection) the **first rabbit** with the **given name**, also **return** the **rabbit**
- **`sellRabbitBySpecies(String species)`** method - sells and returns **all rabbits** from that **species** **as a `List`**
- **`count()`** - **returns** the **number** of rabbits
- **`report()`** - **returns** a **String** in the following **format, including only not sold rabbits**:
    - **`"Rabbits available at {cageName}:`**
      `{Rabbit 1}`
      `{Rabbit 2}`
      `(…)"`

Constraints

- The **names** of the rabbits will be **always unique**.
- You will always have a rabbit added before receiving methods manipulating the Cage's rabbits.

Examples

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

Sample code usage

```
//Initialize the repository (Cage)
Cage cage = new Cage("Wildness", 20);
 //Initialize entity
Rabbit rabbit = new Rabbit("Fluffy", "Blanc de Hotot");
 //Print Rabbit
System.out.println(rabbit); // Rabbit (Blanc de Hotot): Fluffy

//Add Rabbit
cage.add(rabbit);
System.out.println(cage.count()); //1
//Remove Rabbit
cage.removeRabbit("Rabbit Name"); //false

Rabbit secondRabbit = new Rabbit("Bunny", "Brazilian");
Rabbit thirdRabbit = new Rabbit("Jumpy", "Cashmere Lop");
Rabbit fourthRabbit = new Rabbit("Puffy", "Cashmere Lop");
Rabbit fifthRabbit = new Rabbit("Marlin", "Brazilian");

//Add Rabbits
cage.add(secondRabbit);
cage.add(thirdRabbit);
cage.add(fourthRabbit);
cage.add(fifthRabbit);

//Sell Rabbit by name
System.out.println(cage.sellRabbit("Bunny")); //Rabbit (Brazilian): Bunny
//Sell Rabbit by species
List<Rabbit> soldSpecies = cage.sellRabbitBySpecies(("Cashmere Lop");

soldSpecies.forEach(f-> {
      System.out.println(f.getName());

  });
 //Jumpy
 //Puffy

System.out.println(cage.report());
//Rabbits available at Wildness:
//Rabbit (Blanc de Hotot): Fluffy
//Rabbit (Brazilian): Marlin
```

Submission

Submit **single .zip file**, containing **rabbits package, with the classes inside (Rabbit, Cage 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 youp program there. However, there should be **main(String[] args)** method inside.