# 03. Parrots

# Flying parrot PNG images, free download

## Preparation

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

**Pay attention to name the package (parrots), 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.

# Parrot

First, write a Java class **Parrot** 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:

**"Parrot ({species}): {name}"**

# Cage

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

* **name: String**
* **capacity: int**
* **data**: **List<Parrot>** that holds added parrots

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(Parrot parrot) method - **adds** an **entity** to the data **if** **there** **is** **room** for it
* remove(String name) method - removes a parrot by **given name,** if such **exists**, and **returns boolean**
* sellParrot(String name) method - **sell** (**set** its available **property** to **false** without removing it from the collection) the **first parrot** with the **given name**, also **return** the **parrot**
* sellParrotBySpecies(String species) method - sells and returns **all parrots** from that **species as a List**
* count() - **returns** the **number** of parrots
* **report()** - **returns** a **String** in the following **format, including only not sold parrots**:
  + **"**Parrots **available at {cageName}:  
    {**Parrot **1}  
    {**Parrot **2}  
    (…)**"

## Constraints

* The **names** of the parrots will be **always unique**.
* You will always have a parrot added before receiving methods manipulating the Cage’s parrots.

## 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  Parrot parrot = new Parrot("Fluffy", "Loriinae");  //Print Parrot  System.out.println(parrot); // Parrot (Loriinae): Fluffy  //Add Parrot  cage.add(parrot);  System.out.println(cage.count()); //1  //Remove Parrot  cage.remove("Parrot Name"); //false  Parrot secondParrot = new Parrot("Bunny", "Cacatuidae");  Parrot thirdParrot = new Parrot("Jumpy", "Strigopoidea");  Parrot fourthParrot = new Parrot("Puffy", "Strigopoidea");  Parrot fifthParrot = new Parrot("Marlin", "Arinae");    //Add Parrots  cage.add(secondParrot);  cage.add(thirdParrot);  cage.add(fourthParrot);  cage.add(fifthParrot);    //Sell Parrot by name  System.out.println(cage.sellParrot("Bunny")); //Parrot (Cacatuidae): Bunny  //Sell Parrot by species  List<Parrot> soldSpecies = cage.sellParrotBySpecies("Strigopoidea");  soldSpecies.forEach(f-> {  System.out.println(f.getName()); });  //Jumpy  //Puffy  System.out.println(cage.report());  //Parrots available at Wildness:  //Parrot (Loriinae): Fluffy  //Parrot (Arinae): Marlin |

## Submission

Submit **single .zip file**, containing **parrots package, with the classes inside (Parrot, 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 youр program there. However, there should be **main(String[] args)** method inside.