Zoo Keeper

General Introduction

The challenge exercises are intended for students who are done with the regular weekly homework exercises and seek an extra challenge. Only those who passed the challenge program entry test will be considered participants of the challenge program.

Your submission for all the parts of the exercise should be in Java. All the challenge exercises must be done individually. Feel free to send an e-mail to course2ip90+CP@gmail.com if you have any questions.

Assignment description

You are a zookeeper. You keep carnivals: *lions*, *tigers* and *leopards*. You have *zebras*, *wildebeests* and *giraffes*. You have *bears*. You need to accommodate all these animals. For this, your zoo has open *enclosures* and *cages*. Zebras, wildebeests and giraffes can live only in open enclosures. The rest can live in cages as well.

Surely, you can't keep grass-eaters and carnivals together. Besides, carnivals don't like to live with the other species of carnivals. Neither do bears. On the other hand, tigers and leopards are lonely creatures but lions can live together.

Each animal home type has its limited capacity. You need to be able to move animals between homes.

You need to feed your animals. For this you have *hey*, *corn*, *grain* and *carrots*. This is for herbivore animals, except carrots are forbidden for wildebeests for dietary reasons. You have *meat* and chicken for carnivals. Bears eat meat, chicken and carrots. You have limited sources of each type of food. Sometimes you need to replenish each source.

Finally, animals can, sadly, die or can be borrowed/sold to another zoo. So, you need to be able to remove an animal from the zoo.

Exercise

Design and implement a program which would help you to manage your zoo.

Requirements and Constraints

This assignment is about object-oriented programming. Use appropriate abstract and concrete classes with inheritance and polymorphism.
Your client program must be of class `ZooKeeper` and has to look like this:

```
public class ZooKeeper {
  public static void main(String[] args) {
    Zoo myZoo = new MyZoo();
    myZoo.accomodateAnimal("Leopard", "Bagheera", "Cage", 2);
    myZoo.accomodateAnimal("Tiger", "Tom", "Cage",1);
    // ...other Zoo management tasks
  }
}
```

That is, after creating your `Zoo`, the main should contain a sequence of zoo management commands in form of the `Zoo` interface calls.

`Zoo` is an interface which is the only client's access point to all management tasks. You cannot change this interface and you **must not access any other class by your client**. `MyZoo` must be a class, which implements this interface.

This interface is the following:

```
public interface Zoo {
  boolean accomodateAnimal (
String animalSpecies,String animalNickname, String homeType, int homeNumber
);

  boolean removeAnimal(String animalNickname);

  boolean relocateAnimal (
String animalNickname, String homeType, int homeNumber
);

  boolean feedAnimals (
String homeType, int homeNumber, int amountOfFood, String animalFoodType
);

  boolean buyFood (String animalFoodType, int amount);

  void listOfTenants();

}
```

All the methods, except the last one, return *true* if the task is successful, otherwise they return *false*. Successful attempt means that the methods fulfills the corresponding management task in accordance with the assignment description above.

It is the responsibility of `MyZoo` to fulfil the tasks or specify the reason of an unsuccessful attempt. An accommodation attempt fails, if the first restricted pair of animals is found. On the other hand, you should be informed about each unsatisfied animal you were trying to feed. You feed all animals in a certain home with some amount of predefined food in one go. If at least one animal in the home can eat the offered food, it eats all amount of this food.

Method `listOfTenants()` should print lists of animals in each animal home which is not empty.

You program **must not crash** if any combination of interface calls is constructed as a sequence in the `ZooKeeper's main`().

Some numerical constraints:
There are *20* cages in the Zoo numbered from 1 to 20 and *5* enclosures numbered the same way. Each cage has the capacity of *4* animals and each enclosure can accommodate up to *20* animals. Once created, you cannot change these capacities.

Each animal has its unique nickname. That is, accommodation of yet another animal with this name should not be allowed regardless the species type.

You have a single storage for each type of food, that is one for meat, one for grain, etc. Each storage can be initially filled with a certain amount of food.

Hints

In addition to the flawless execution, your submission will be assessed by the quality of your class design. Think carefully what abstract and concrete classes you use. What instance variables and methods should they have?

A good indication of a proper design:
What if you need to extend your Zoo with an aquapark: add pools, fish tanks and terrariums to keep crocodiles, snakes and tropical fish, each animal type with yet another own habits? (You don't need to do this, but you can, if you want.)
If, to be able to do this, you don't need to change your current classes (except, maybe a little bit, the implementation class `MyZoo`) but just need to add new ones, then your design is correct and solid.

Examples

These examples do not exhaust all possible outputs which your Zoo methods have to produce. Neither they indicate all possible failures they must handle. However, if your program can handle them, then it's a good indication that you are on the right way.

Tasks in main():

```
System.out.println(myZoo.accomodateAnimal("Leopard","Bagheera","Cage",21));
System.out.println(myZoo.accomodateAnimal("Leopard", "Bagheera", "Cage",
11));
System.out.println(myZoo.accomodateAnimal("Lion", "Tom", "Cage",2));
System.out.println(myZoo.accomodateAnimal("Lion", "Tom1", "Cage",2));
System.out.println(myZoo.accomodateAnimal("Lion", "Tom2", "Cage",2));
System.out.println(myZoo.accomodateAnimal("Lion", "Tom3", "Cage",2));
System.out.println(myZoo.accomodateAnimal("Lion", "Tom4", "Cage",2));
System.out.println(myZoo.accomodateAnimal("Tiger", "King", "Cage",6));
System.out.println(myZoo.accomodateAnimal("Tiger", "Kid", "Cage",6));
System.out.println(myZoo.accomodateAnimal("Zebra", "Sid", "Cage",1));
System.out.println(myZoo.accomodateAnimal("Leopard", "Bagheera", "Enclosure",
1));
System.out.println(myZoo.accomodateAnimal("Zebra", "Sid", "Enclosure",1));
System.out.println(myZoo.accomodateAnimal("Zebra", "Vera", "Enclosure",1));
System.out.println(myZoo.accomodateAnimal("Leopard", "Bagheera", "Enclosure",
1));
System.out.println(myZoo.accomodateAnimal("Tiger", "Karl", "Enclosure",1));
System.out.println(myZoo.accomodateAnimal("Tiger", "Karl", "Cage",5));
System.out.println(myZoo.accomodateAnimal("Wildebeest", "Vega",
"Enclosure",1));
```

```
System.out.println(myZoo.accomodateAnimal("Giraffe", "Donald", "Enclosure",
4));
System.out.println(myZoo.accomodateAnimal("Lion", "King", "Enclosure",0));
System.out.println(myZoo.accomodateAnimal("Lion", "Kid", "Enclosure",0));
System.out.println(myZoo.feedAnimals("Enclosure", 0, 100, "Carrot"));
System.out.println(myZoo.feedAnimals("Enclosure", 0, 200, "Meat"));
System.out.println(myZoo.buyFood("Meat", -200));
System.out.println(myZoo.feedAnimals("Enclosure",0, 100, "Meat"));
System.out.println(myZoo.relocateAnimal("Bagheera", "Enclosure", 1));
System.out.println(myZoo.relocateAnimal("Bagheera", "Enclosure", 2));
System.out.println(myZoo.accomodateAnimal("Zebra", "Sid", "Enclosure",1));
System.out.println(myZoo.removeAnimal("Bagheera"));
System.out.println(myZoo.relocateAnimal("Bagheera", "Enclosure", 3));
System.out.println(myZoo.accomodateAnimal("Tiger", "Tom", "Cage",1));
System.out.println(myZoo.buyFood("Meat", 20));
System.out.println(myZoo.buyFood("Carrot", 20));
myZoo.listOfTenants();
```

Output

```
Our Zoo doesn't have Cage number 21
false
true
true
true
true
Cage is full, 4 animals already.
false
true
Tiger can't live with Tiger
false
Zebra doesn't live in Cage
false
true
Leopard can't live with Zebra
false
Leopard can't live with Zebra
false
Leopard with nickname Bagheera already lives in our Zoo!
false
Tiger can't live with Leopard
false
true
Leopard can't live with Wildebeest
false
true
Tiger with nickname King already lives in our Zoo!
false
true
Lion Kid doesn't eat Carrot
false
Not enough Meat, current amount = 100
false
You can't buy negative amount of Meat
false
```

```
true
true
true
true
true
There is no such an animal Bagheera in the Zoo
false
Lion with nickname Tom already lives in our Zoo!
false
true
true
Cage 2
Lion nickname Tom
Lion nickname Tom1
Lion nickname Tom2
Lion nickname Tom3
Cage 5
Tiger nickname Karl
Cage 6
Tiger nickname King
Enclosure 0
Lion nickname Kid
Enclosure 1
Zebra nickname Sid
Enclosure 4
Giraffe nickname Donald
```

Handing in the assignment

Upload into Canvas a zip file ZooKeeper.zip containing all the .java files of the project in one folder.