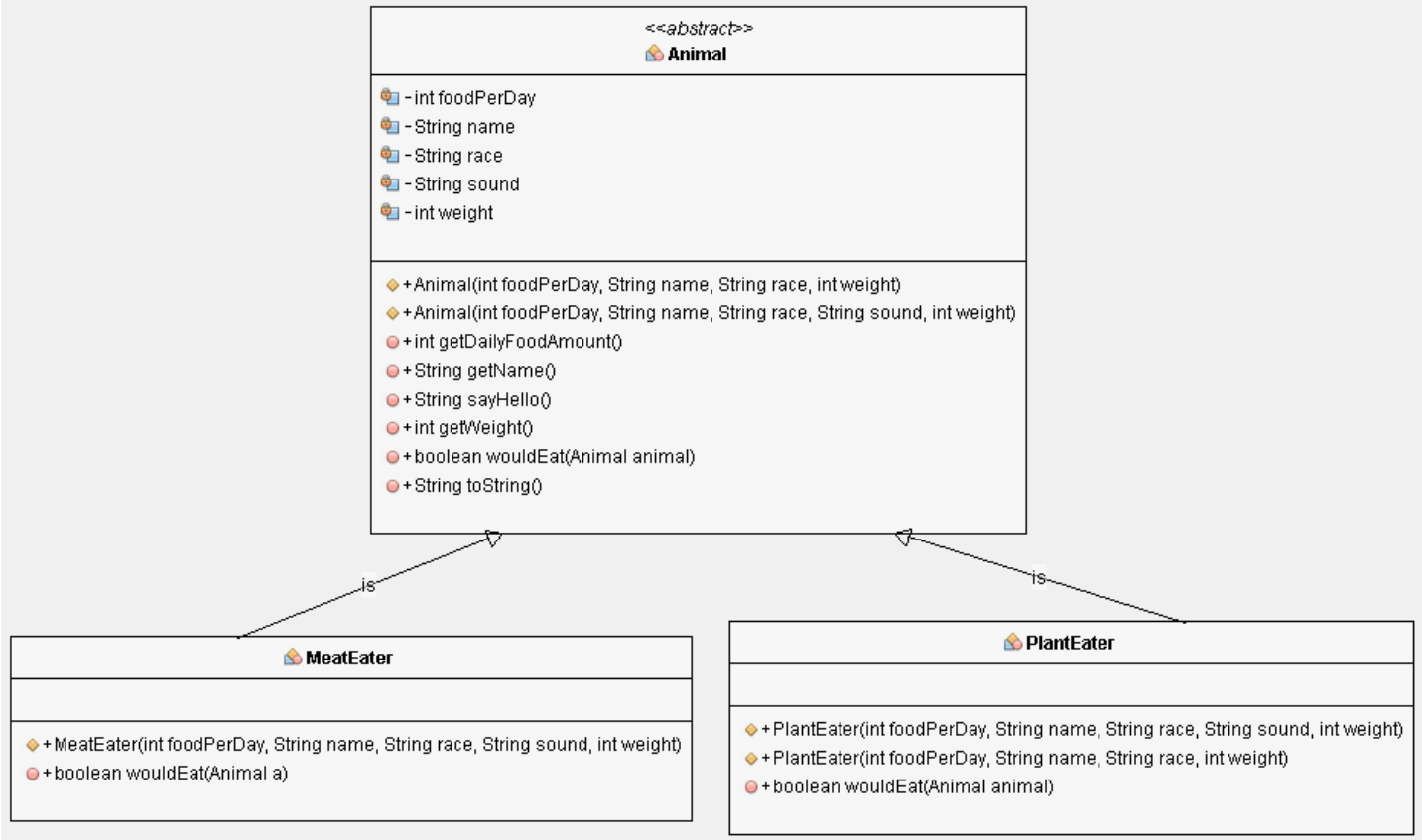
Programmieren und Software-Engineering II
Übung 18

Name: _____ **Klasse:** _____ **Datum:** _____

Aufgabe 1: Klasse `Animal`, Klasse `MeatEater`, Klasse `PlantEater`

Bei Tieren kann man u.a. zwischen fleischfressenden und pflanzenfressenden Tieren unterscheiden. Pflanzenfressende Tiere sind in der Regel nicht gefährlich für andere Tiere. Bei Fleischfressern hingegen stehen andere Tiere am täglichen Speiseplan.

Das nachfolgende UML-Diagramm zeigt die Klassen, die dazu implementiert werden müssen:

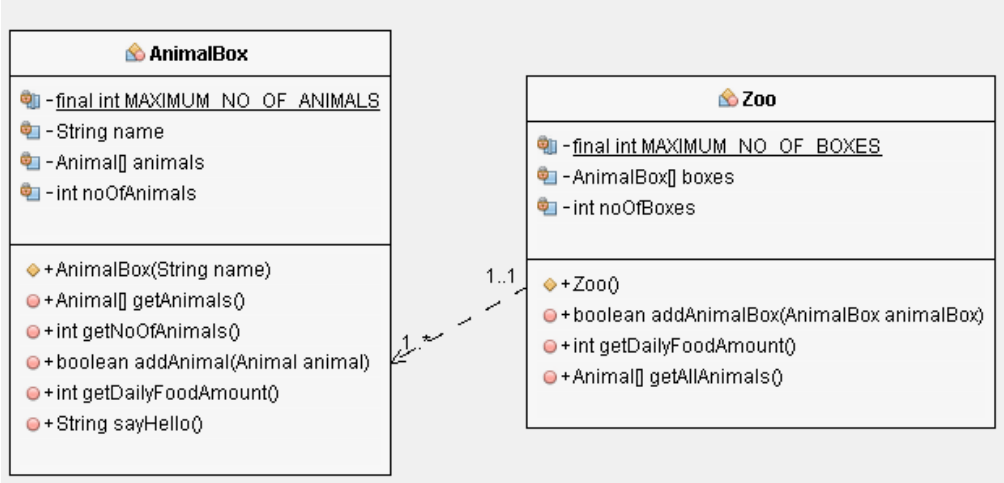


HINWEISE:

- Erstelle die angegebenen **Konstruktoren**
- Erstelle nur die angegebenen `get()` – Methoden
- Die Klasse `Animal` ist als **abstrakte Klasse** zu realisieren.
  Die darin enthaltene Methode `boolean wouldEat(Animal a)` ist eine **abstrakte Methode**.
- Durch die Methode `sayHello()` kann der Wert des Attributs `sound` ausgelesen werden.
- Die Klasse `MeatEater` ist abgeleitet von der Klasse `Animal`.
  Die darin enthaltene Methode `boolean wouldEat(Animal a)` stellt fest ob ein Tier gefressen werden kann. Ein Tier wird gefressen wenn:
  - `Animal a` ebenfalls ein Fleischfresser ist UND wenn das Gewicht von `Animal a` kleiner ist als das eigene Gewicht
- Die Klasse `PlantEater` ist ebenfalls abgeleitet von der Klasse `Animal`.
  Die darin enthaltene Methode `boolean wouldEat(Animal a)` ist so zu implementieren, dass KEIN Tier gefressen werden kann.

Programmieren und Software-Engineering II
Übung 18

Aufgabe 2: Klasse `Zoo`, Klasse `AnimalBox`
Die in Aufgabe 1 erstellten Tiere werden in einem Zoo beherbergt. Die Aufgabe besteht nun darin, verschiedenste Tiere nach vorgegebenen Regeln in mehrere Gehege aufzuteilen, sodass sich die Tiere gegenseitig nicht gefährden. Außerdem soll die tägliche Futtermenge berechnet werden und eine sortierte Liste aller Tiere im Zoo ausgedruckt werden können.



HINWEISE:

- Die Größe der Arrays wird mit einer Konstante (`MAXIMUM_NO_OF_BOXES` bzw. `MAXIMUM_NO_OF_ANIMALS`) festgelegt.

- Die Klasse `Zoo` verwaltet mehrere Gehege (Klasse `AnimalBox`) und implementiert folgende Methoden:
  `boolen addAnimalBox(AnimalBox animalBox)`: neues Gehege hinzufügen
  `int getDailyFoodAmount()`: tägliche Futtermenge aller Tiere berechnen
  `Animal[] getAllAnimals()`: liefert ein Array, das alle Tiere des Zoos beinhaltet. In einem ersten Schritt werden alle Tiere gezählt. Dann kann ein Array entsprechender Größe angelegt werden. Anschließend wird das Array mit den Tieren befüllt.

- Die Klasse `AnimalBox` beinhaltet ein Array von Tieren und implementiert folgende Methoden:
  `Animal[] getAnimals()`: liefert ein Array, das alle Tiere eines Geheges beinhaltet.
  `int getNoOfAnimals()`: liefert die Anzahl der Tiere in einem Gehege
  `boolean addAnimal(Animal a)`: ein Tier zum Gehege hinzufügen. Ein Tier kann nur hinzugefügt werden, wenn noch ein Platz im Gehege frei ist und wenn im Gehege kein Tier ist, das es fressen könnte.
  `int getDailyFoodAmount()`: die tägliche Futtermenge für ein Gehege berechnen
  `String sayHello()`: für jedes Tier im Gehege wird der Wert des Attributs `sound` ausgegeben.

Programmieren und Software-Engineering II
Übung 18

Aufgabe 3: Hauptprogramm

Das **Hauptprogramm** besitzt folgenden **Aufbau:**

```
public static void main(String[] args) {
        Zoo zoo = new Zoo();
        AnimalBox dangerousBox = new AnimalBox("Dangerous box");
        zoo.addAnimalBox(dangerousBox);

        Animal crocodile = new MeatEater(1000, "Croco", "Crocodile", "schnapp", 3000);
        dangerousBox.addAnimal(crocodile);

        Animal tiger = new MeatEater(2000, "Toni", "Tiger", "roaarrr", 8000);
        dangerousBox.addAnimal(tiger);

        Animal sheep = new PlantEater(500, "Cindy", "Sheep", "bäääääh", 800);
        dangerousBox.addAnimal(sheep);

        System.out.println(dangerousBox.sayHello());
        System.out.println();

        AnimalBox pinkBox = new AnimalBox("Pink box");
        zoo.addAnimalBox(pinkBox);

        pinkBox.addAnimal(sheep);
        System.out.println(pinkBox.sayHello());
        System.out.println();

        pinkBox.addAnimal(tiger);
        Animal rhino = new PlantEater(5000, "Albert", "Rhinoceros", 15800);
        dangerousBox.addAnimal(rhino);
        System.out.println(dangerousBox.sayHello());
        System.out.println();

        System.out.println("We are hungry! give us "+zoo.getDailyFoodAmount()+
                           " g of food every day!");
        System.out.println();

        Animal[] animals = zoo.getAllAnimals();
        System.out.println("All animals in the zoo (unsorted): ");
        print(animals);

        System.out.println("All animals in the zoo (sorted by name): ");
        sortByName(animals);
        print(animals);

       // print all PlantEaters and all MeatEaters
    }
```

HINWEISE:

- Implementiere eine statische Methode `print()`, die ein Array von Tieren zeilenweise ausgibt.
- Implementiere eine statische Methode `sortByName(animals)`, die ein Array von Tieren aufsteigend nach ihrem Namen sortiert.
- Implementiere eine Methode, welche zuerst alle Pflanzenfresser und danach alle Fleischfresser ausgibt.

Demoausgabe

adding Croco (Crocodile): succeeded
adding Toni (Tiger): succeeded
adding Cindy (Sheep): failed
Dangerous box: schnapp roaarrr

adding Cindy (Sheep): succeeded
Pink box: baa

removing Toni (Tiger): succeeded
adding Toni (Tiger): failed
adding Olaf (Rhinoceros): succeeded
Dangerous box: schnapp

We are hungry, give us 6500 g of food every day

Programmieren und Software-Engineering II
Übung 18

All animals in the zoo (sorted by name): Cindy (Sheep) Croco (Crocodile) Olaf (Rhinoceros)

Print all PlantEaters and all MeatEaters
PlantEater
Olaf (Rhinoceros)
Cindy (Sheep)

MeatEater
Croco (Crocodile)