Gruppe: Ladies Night
Autoren: Marius Birk
Pieter Vogt

Tutorium: Gruppe B/G

Punkte:

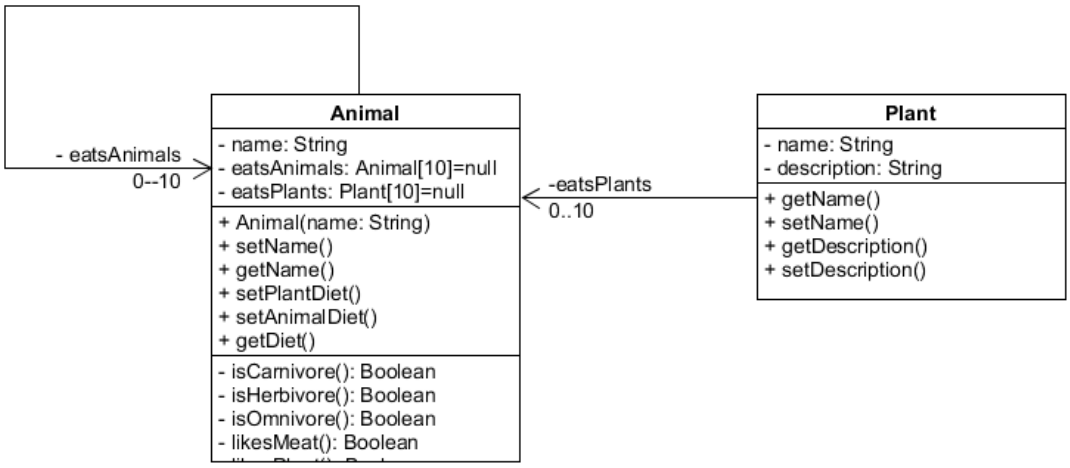
A1	$\sum$

Objektorientierte Modellierung und Programmierung
Übung 01

ANMERKUNG: Marius Birk(Tutorium B) und Pieter Vogt (Tutorium G) arbeiten zusammen und geben gemeinsam den bearbeiteten Zettel ab. Parallel wurde von jedem Teilnehmer der Gruppe ein Programm geschrieben, sodass es in der Abgabe zu Kollisionen in den Objektbezeichnungen kommen kann. Jeder Teilnehmer kann aber im Notfall eine funktionierende Variante seines Programmes vorweisen.

Aufgabe 1

UML-Architektur



Implementierung

Der Code für die Klasse 'Animal' sieht aus wie folgt:

```
1 public class Animal {
2    //FIELDS
3    private String name;
4    private Animal[] eatsAnimals = new Animal[10];
5    private Plant[] eatsPlants = new Plant[10];
6    //CONSTRUCTOR
7    public Animal(String name) {
8       this.name = name;
9    }
```

```
10      //GETTER SETTER
11      public void setName(String name) {
12          this.name = name;
13      }
14      public String getName() {
15          return name;
16      }
17      public void setPlantDiet(Plant plant) {
18          for (int i = 0; i < eatsPlants.length; i++) {
19              if (eatsPlants[i] == null) {
20                  eatsPlants[i] = plant;
21                  return;
22              }
23              return;
24          }
25      }
26      public void setAnimalDiet(Animal animal) {
27          for (int i = 0; i < eatsAnimals.length; i++) {
28              if (eatsAnimals[i] == null) {
29                  eatsAnimals[i] = animal;
30                  return;
31              }
32              return;
33          }
34      }
35      //PUBLIC METHODS
36      public void getDiet() {
37          if (isCarnivore()) {
38              System.out.println(this.name + " is a carnivore");
39          } else if (isHerbivore()) {
40              System.out.println(this.name + " is a herbivore");
41          } else System.out.println(this.name + " is a omnivore")
                ;
42      }
43      //HELPER METHODS
44
45      private Boolean isCarnivore() {
46          if (!likesPlant()) {
47              return true;
48          } else return false;
49
50      private Boolean isHerbivore() {
51          if (!likesMeat()) {
52              return true;
53          } else return false;
54      }
55      private Boolean isOmnivore() {
56          if (likesMeat() && likesPlant()) {
57              return true;
```

```
58          } else return false;
59      }
60      private boolean likesMeat() {
61          for (int i = 0; i < eatsAnimals.length; i++) {
62              if (eatsAnimals[i] != null) {
63                  return true;
64              }
65          }
66          return false;
67      }
68      private boolean likesPlant() {
69          for (int i = 0; i < eatsPlants.length; i++) {
70              if (eatsPlants[i] != null) {
71                  return true;
72              }
73          }
74          return false;
75      }
76  }
```

Der Code für die Klasse 'Plant' sieht aus wie folgt:

```
1   public class Plant {
2       //FIELDS
3       private String name;
4       private String description;
5       //GETTER SETTER
6       public String getName() {
7           return name;
8       }
9       public String getDescription() {
10          return description;
11      }
12      public void setName(String name) {
13          this.name = name;
14      }
15      public void setDescription(String description) {
16          this.description = description;
17      }
18  }
```

Der Code für das Programm 'Biotest' sieht aus wie folgt:

```
1   public class Bio_Test {
2       public static void main(String[] args){
3           Pflanzen_Tiere.Pflanzen Gras = new Pflanzen_Tiere.
                Pflanzen();
4           Gras.setBeschreibung("Gras ist Gruen");
5
6           Pflanzen_Tiere.Pflanzen Beeren = new Pflanzen_Tiere.
                Pflanzen();
```

```
7           Beeren.setBeschreibung("Beeren sind rot");
8
9           Pflanzen_Tiere.Tiere Zebra = new Pflanzen_Tiere.Tiere
                ();
10          Zebra.setName("Zebra");
11          Zebra.setPf_futter("Gras");
12
13
14          Pflanzen_Tiere.Tiere Loewen = new Pflanzen_Tiere.Tiere
                ();
15          Loewen.setName("Loewen");
16          Loewen.setF_futter("Zebras");
17
18          Pflanzen_Tiere.Tiere Baeren = new Pflanzen_Tiere.Tiere
                ();
19          Baeren.setName("Baeren");
20          Baeren.setPf_futter("Beeren");
21          Baeren.setF_futter("Fische");
22
23          Zebra.ausgabe();
24          Loewen.ausgabe();
25          Baeren.ausgabe();
26      }
27  }
```

Die UML-Architektur der Objektebene nach Ausführung von 'Biotest' sieht wie folgt aus: