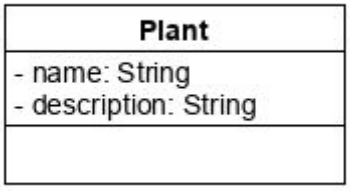
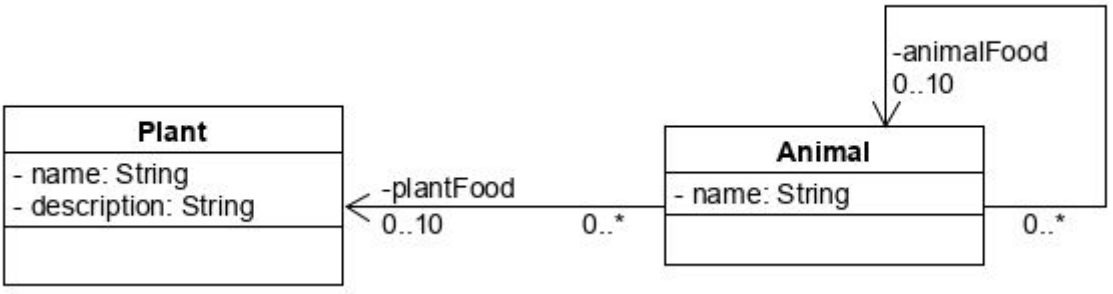
1a)
Die Pflanze und das Tier sollten hier ziemlich leicht als die Klassen zu identifizieren sein. Dabei gilt als Daumenregel, dass meistens Nomen, Gegenstände und Objekte aus der echten Welt als Klassen infrage kommen. Wenn diesen Gegenständen Eigenschaften zugewiesen werden, sind diese meist als Attribut in einem UML Diagramm umzusetzen. Konkrete Beispiele aus dieser Aufgabe sind: *"Zu jeder Pflanze und zu jedem Tier möchte er den Namen speichern können."* und *"Zu jeder Pflanze möchte er eine kurze textuelle Beschreibung speichern können."*

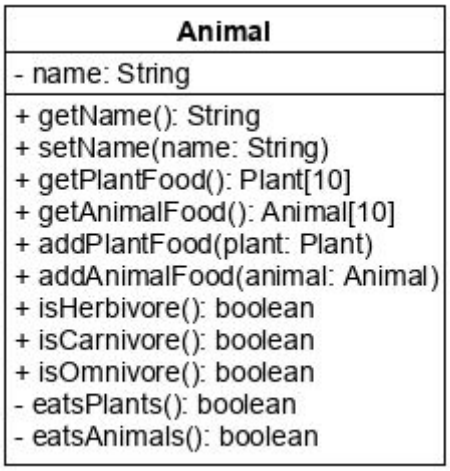


Wichtig dabei sind Attribute, die Beziehung von Objekten darstellen, beziehungsweise andere Objekte speichern. Diese werden in Form von **Assoziationen** (Pfeile zwischen Objekten) dargestellt. Es gibt mehrer Hinweise in der Aufgabe auf 2 verschiedene Assoziationen: *"Zu jedem Tier möchte er gerne speichern können, was es frisst."* und *"Ein Array für alle Pflanzen, die das Tier frisst, und eins für alle Tiere, die es frisst.".* Aus diesen beiden Hinweisen ist zu erschließen, dass Tier eine Assoziation zu sich selbst haben muss und eine zu Pflanze.
Die **Rollenbezeichner** werden so genannt wie das dazugehörige Attribut. Genauso wie bei Methoden und Attributen darf bei Rollenbezeichnern die **Sichtbarkeit** nicht fehlen. Die **Multiplizität**, die in diesem Fall bei beiden Assoziationen 0..1 und 0..* ist, lässt sich dabei aus der in der Aufgabe vorgegebenen Arraylänge von 10 erschließen. Schwerer ist dagegen die Multiplizität von 0..*. Dabei muss die Assoziation Rückwärts gedacht werden. Sprachlich könnte man sagen: "Eine Pflanze / ein Tier ist für wie viele andere Tiere das Futter?". Theoretisch kann ein Tier / eine Pflanze von beliebig vielen, aber auch von keinem anderen Tier gegessen werden, daher die Multiplizität 0..*.



Zuletzt fehlen nur noch die Methoden. Da in dieser Aufgabe kein Konstruktor benötigt ist und der Standardkonstruktor nicht im UML Diagramm aufgeführt werden muss, ist in der vollständigen Klasse keiner eingezeichnet. Nicht zu vergessen sind hierbei die Getter und Setter Methoden für die einzelnen Attribute der Klassen. Außerdem haben die beiden Hilfsmethoden die Sichtbarkeit `private` (-), da sie nur aus der Klasse zugegriffen werden.

Alle anderen Methoden sind `public` (+). Rückgabewerte und Parameter jeder Methode sollten angegeben werden, dabei kann der Rückgabetyp `void` ausgelassen werden.



1c)
Die Klasse Animal muss hier nur um die beiden privaten Methoden `eatsPlants():boolean` und `eatsAnimals():boolean` erweitert werden.