Fakultät Wirtschaft - Wirtschaftsinformatik

Aufgaben:	Fortgeschrittene Programmierung (SS 2024)
Studienrichtung:	EG/EH

1. Wiederholung Klassen!

(6) (a) Erstellen Sie die Klasse `Character` nach folgendem Klassendiagramm.

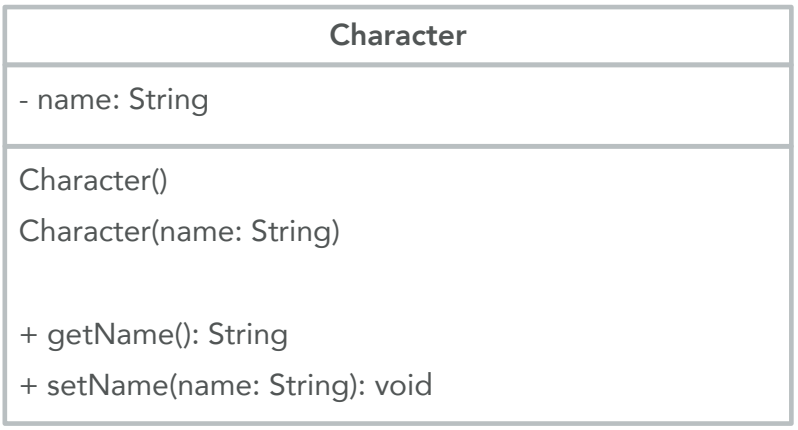

Abbildung 1: Klassendiagramm `Character`

(5) (b) Legen Sie zwei Objekte dieser Klasse an. Geben Sie einem Objekt den Namen über den Konstruktor mit. Dem zweiten Objekt über die Setter Methode. Lassen Sie sich dann von beiden Objekten den Namen in der Konsole ausgeben.

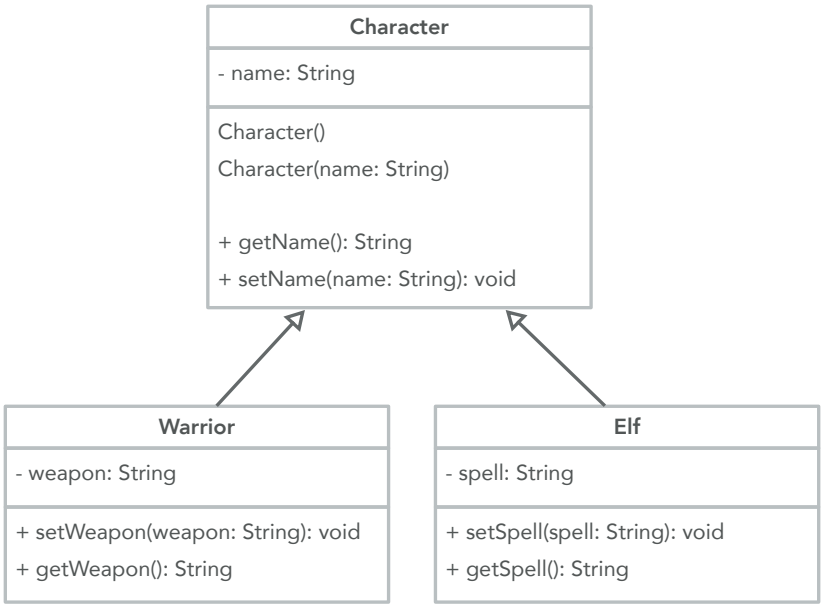

Abbildung 2: Klassendiagramm `Character` mit Kindklassen

(5) (c) Legen Sie eine zweite Klasse `Warrior` an, die von der Klasse `Character` abgeleitet ist (s. Abb. 2).

(5) (d) Legen Sie eine dritte Klasse `Elf` an, die von der Klasse `Character` abgeleitet ist (s. Abb. 2).

(10) (e) Legen Sie jeweils ein Objekt der Klasse `Elf` und eines der Klasse `Warrior` an. Vergeben Sie für beide Objekte Namen. Das Objekt der Klasse `Warrior` soll dann ein `Schwert` als Waffe erhalten. Das Objekt `Elf` Zauberspruch `Feuerball`. Lassen Sie sich anschließend den Namen der Objekte und ihrer jeweiligen Waffen auf der Konsole ausgeben.

(5) (f) Erweitern Sie die Klasse Warrior so, dass es möglich ist, ihr über den Konstruktor einen Namen zuzuweisen. Es gibt dafür zwei Möglichkeiten. Schreiben Sie beide auf.

2. Wiederholung Abstrakte Klassen!

(6) (a) Erstellen Sie die abstrakte Klasse `Monster` nach folgendem Klassendiagramm.

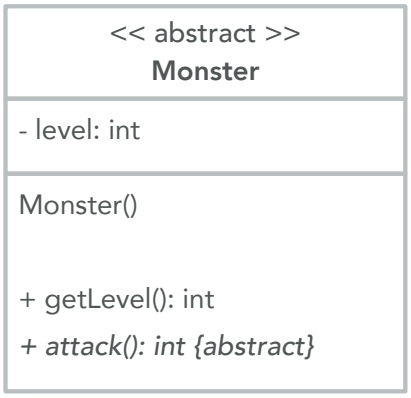

Abbildung 3: Klassendiagramm `Monster`

(4) (b) Legen Sie nun eine Kindklasse `Goblin` an, in der Sie die abstrakte Methode `attack()` auscodieren (s. Abb. 4). Als Rückgabewert soll hierbei das Level multipliziert mit 10 genommen werden. Legen Sie dann ein Objekt dieser Klasse an. Lassen Sie sich das Level ausgeben, bevor und nachdem der Goblin einmal gelevelt hat. Lassen Sie sich dann den Attack-Wert anzeigen.

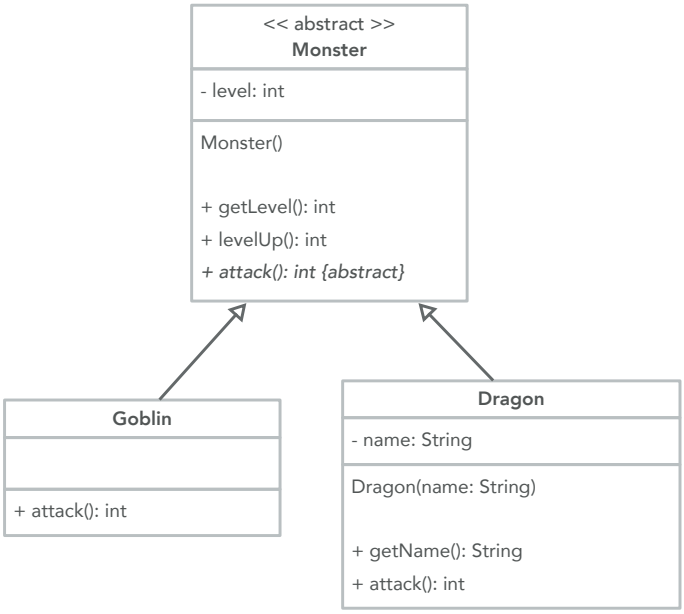

Abbildung 4: Klassendiagramm `Monsters`

(4) (c) Legen Sie nun die Kindklasse `Dragon` an, die über die Eigenschaft `name` verfügt. Legen Sie einen eigenen Konstruktor an und codieren Sie die Methode `attack()` aus. Diese soll den 200-fachen Wert des Levels zurückgeben (s. Abb. 4). Legen Sie dann ein Objekt dieser Klasse an und lassen sich den Namen, das Level und den Attack-Wert anzeigen.

3. Interfaces

(10) (a) Implementieren Sie die Klassen `Monster` und `Dragon` aus dem Klassendiagramm 5. Legen Sie drei Objekte der Klasse `Dragon` an und geben Sie jedem Drachen einen Namen und eine Startposition. Lassen Sie sich dann von allen Drachen den Namen und die Position in der Konsole ausgeben.

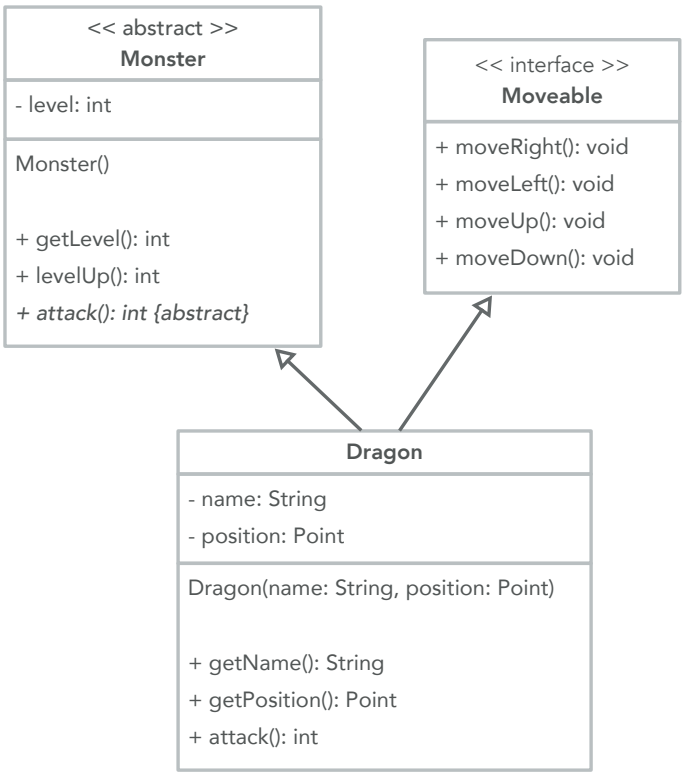

Abbildung 5: Klassendiagramm `Interface`

(5) (b) Erstellen Sie das Interface `Moveable` (s. Abb. 5).

(5) (c) Erweitern Sie die Klasse `Dragon` um das Interface `Moveable`. Implementieren Sie die vier übergebenen Methoden. Bewegen Sie dann die drei Drachen jeweils einen Schritt in eine beliebige Richtung. Lassen Sie sich die jeweiligen Schritte auf der Konsole ausgeben.

(10) (d) Erweitern Sie die Klasse `Dragon` um die private Eigenschaft `direction`. Diese hat den Datentyp String und enthält die Bewegungsrichtung (`w`, `a`, `s`, `d`). Erstellen Sie die zugehörige Getter- und Setter-Methode. Erweitern Sie das Interface `Moveable` um die Methode `move`.

Implementieren Sie diese in der Klasse `Dragon`. Die Methode soll abhängig vom Wert der Eigenschaft `direction` die entsprechende Methode (`moveUp()`, `moveLeft()`, `moveDown()`, `moveRight()`) aufrufen.

(5) (e) Erstellen Sie eine Klasse `Mover`. Diese enthält eine statische Methode `moveDragons`, an die als Parameter die `Dragon`-Objekte übergeben werden. Innerhalb der Methode werden dann die Drachen bewegt (Methode `move`).

(10) (f) Erstellen Sie eine Klasse `Character` (vgl. Abb. 1). Diese enthält analog zur Klasse `Dragon` die Methoden `moveUp()`, `moveLeft()`, `moveDown()`, `moveRight()` sowie die Methode `move`. Der `Character` darf pro Bewegung 2 Felder vorrücken. Erstellen Sie außerdem die Eigenschaft `direction` mit der zugehörigen Getter- und Setter-Methode.

(2) (g) Erweitern Sie die Klasse `Mover` um eine weitere statische Methode `moveAll`. Diese nimmt analog zur Methode `moveDragons` bewegliche Objekte entgegen und bewegt diese über die Methode `move`.

(4) (h) Legen Sie ein Objekt der Klasse `Character` an und initialisieren Sie dieses mit sinnvollen Werten. Rufen Sie dann die Methode `moveAll` der `Mover`-Klasse auf und übergeben an diese alle Drachen und das `Character`-Objekt. Lassen Sie sich alle Schritte zur Kontrolle auf der Konsole ausgeben.

4. Generics

(12) (a) Erstellen Sie die Klasse `Weapon` mit ihren beiden Kindern aus Klassendiagramm 6. Erstellen Sie außerdem die Klasse `Warrior`.

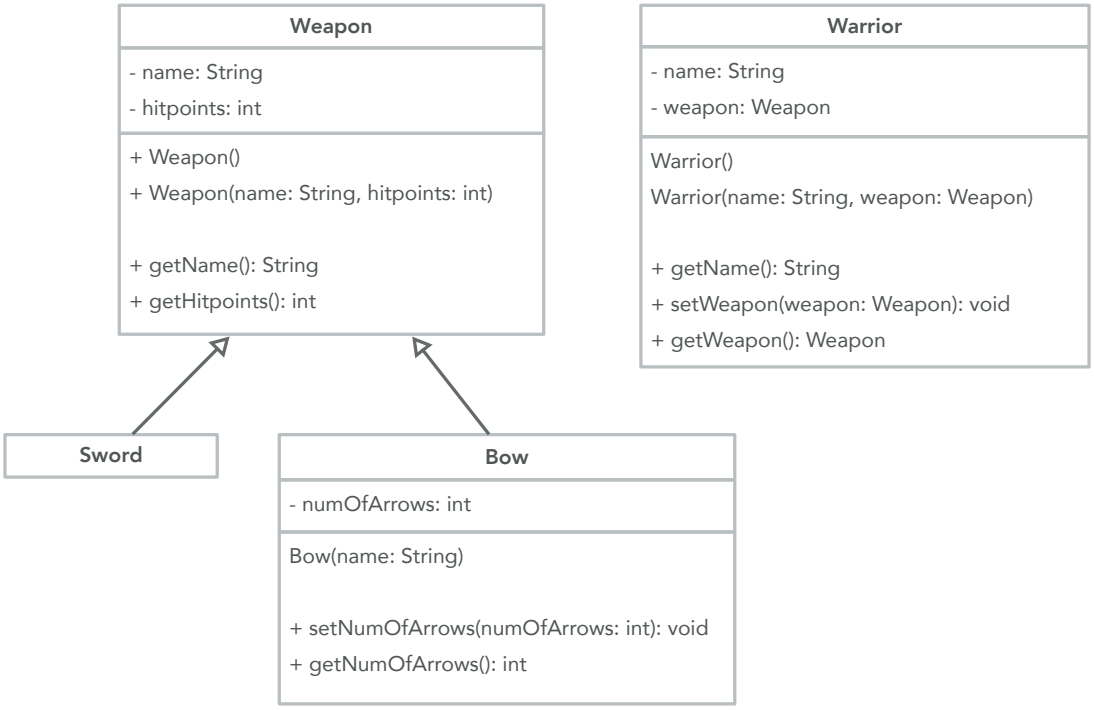

Abbildung 6: Klassendiagramm `Generics`

(6) (b) Legen Sie dann ein Objekt für einen Bogen, eines für ein Schwert und zwei für einen Krieger an. Ein Krieger erhält als Waffe den Bogen, der andere das Schwert. Lassen Sie sich den Namen des Kriegers und den Namen der jeweiligen Waffe in der Konsole ausgeben. Versuchen Sie sich die Anzahl der Pfeile des Bogens ausgeben zu lassen.

(5) (c) Legen Sie die Klasse `Goblin` an (s. Abb. 7). Diese wird mittels generischer Typen erstellt, d.h. die Eigenschaft `weapon` ist generisch und hat in der Klasse noch keinen Datentyp. Dieser wird erst zur Laufzeit festgelegt.

(5) (d) Legen Sie zwei Objekte der Klasse `Goblin` an und weisen einem den Bogen und dem anderen das Schwert aus Teilaufgabe 4.b zu. Lassen Sie sich dann den Namen der Goblins sowie ihrer Waffen auf der Konsole ausgeben. Lassen Sie sich dann auch die Anzahl der Pfeile des Bogens

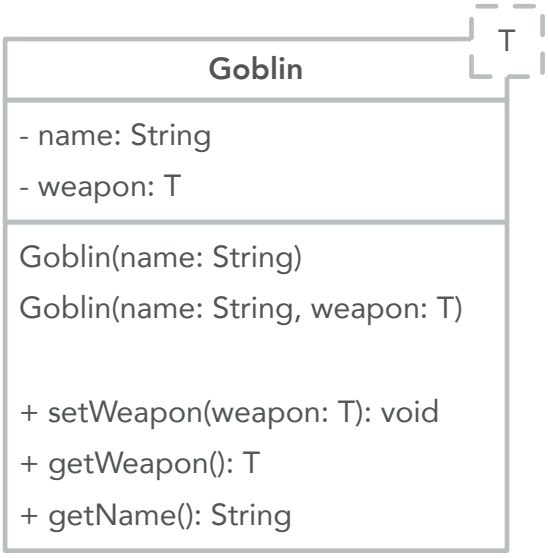

Abbildung 7: Klassendiagramm `Goblin`

ausgeben.