Hochschule Bonn-Rhein-Sieg
Fachbereich Informatik
M.Sc. Andre Kless
Dipl. Inf. Sigrid Weil

Übung 11

Einführung in die
Programmierung
SoSe 2023

Themen:
Definition von Interfaces, Implementierung von Interfaces, Nutzung von Interfaces

Aufgabe 11.1

a) Definieren Sie ein Interface `Vergleichbar`. Objekte dieses Typs verfügen über eine Methode `boolean kleiner(Object o)`
die angibt, ob das gegebene Objekt „kleiner" als das übergebene Objekt ist.
Was genau „kleiner" bedeuten soll, hängt von der implementierenden Klasse ab.

b) Leiten Sie eine Unterklasse `PersonAlter` von `Person` ab, die das Interface implementiert. Eine Person soll „kleiner" als eine andere sein, wenn sie jünger ist.

c) Modifizieren Sie die Klasse `Punkt2D` so, dass auch sie das Interface implementiert (oder leiten Sie eine Unterklasse von `Punkt2D` ab, die dies tut). Ein Punkt soll „kleiner" als ein anderer sein, wenn er näher am Ursprung liegt.

d) Im Code-Download-Bereich ist eine Testklasse zur Verfügung gestellt, die Sie nutzen können.
Definieren Sie in der Testklasse eine statische Methode `min()`, die ein Array von einem *vergleichbaren Typ* als Parameter erhält und das *kleinste* Objekt des Arrays zurückliefert.

Hinweis:
Bedenken Sie bei der Implementierung, dass die Methode `min` als Eingabe ein Objekt der Klasse `Object` erwartet. Um spezielle Methoden der implementierenden Klassen darauf anwenden zu können, muss das Objekt auf die passende Klasse gecastet werden.

Aufgabe 11.2

Gegeben sei das Unter-Interface `Turtle` von `Beweglich`, deren Codes Sie im Download-Bereich des LEA-Kurses finden. Nehmen Sie an, die Klasse `TurtleRoboter` implementiere dieses Interface.
Die Klasse verfüge über einen parameterlosen Konstruktor, der einen Roboter mit Startposition $(0, 0)$ und Blickrichtung „oben" erzeugt.
Implementieren Sie eine Unterklasse `MyRoboter` von `TurtleRoboter`, die verschiedene Instanzmethoden anbietet:

a) `void squareDance(int n)`
lässt das Objekt n-viele Quadrate mit den Eckpunkten $(n, n), (-n, n), (-n, -n), (n. - n)$ ablaufen. Der Roboter startet im Punkt $(1, 1)$ und durchläuft das erste Quadrat bis zum Punkt $(1, -1)$, von dort zurück zum Punkt $(1, 1)$, von dort zum Punkt $(2, 2)$, von hier aus das nächste Quadrat usw.

b) `void treppenLauf(Punkt2D start, int weglaenge)`
lässt den Roboter einen treppenförmigen Weg nehmen:
vom Startpunkt aus einen Schritt nach oben, dann einen nach rechts, wieder nach oben usw. Dabei ist jede Treppenstufe allerdings um 1 höher und breiter als die vorherige Stufe.
Der Roboter soll so lange laufen, bis seine insgesamt zurückgelegte Wegstrecke den Wert `weglaenge` erreicht bzw überschreitet.

Bemerkung:
Zum Testen Ihrer Lösung ist auch eine Implementierung der Klasse `TurtleRoboter` im Download-Bereich hinterlegt. Als Klausuraufgabe, die auf dem Papier zu lösen wäre, müssten Sie die Aufgabe *ohne Kenntnis* der Klasse `TurtleRoboter` lösen können.

Aufgabe 11.3
Ergänzen Sie das Projekt „Mittelerde“ aus Übung 9 wie folgt:

- Wir wollen nun auch Gegenstände modellieren. Gegenstände haben eine Bezeichnung vom Typ String.
- Manche Gegenstände (zB Waffen) sind tragbar. Tragbare Gegenstände haben ein Gewicht vom Typ double.
- Wesen können tragbare Dinge nehmen, wobei
  - Hobbits nur Dinge tragen können, deren Gewicht höchstens 20 (kg) beträgt.
  - Magier können beliebig schwere Dinge tragen.
  - Jedes Wesen kann nur ein Ding zur gleichen Zeit tragen.

a) Definieren Sie eine Klasse `Gegenstand` mit einem Konstruktor und einer get-Methode für die Bezeichnung.

b) Definieren Sie ein Interface `Tragbar`, in dem eine Methode `double gewicht()` definiert wird.

c) Implementieren Sie das Interface durch eine Klasse `Waffe`, die von `Gegenstand` abgeleitet wird.

d) Zeichnen Sie das UML-Diagramm neu.

e) Fügen Sie der Klasse `Wesen` ein Attribut `ding` vom Typ `Tragbar` und eine abstrakte Methode `void nehmen(Tragbar)` hinzu und eine (nicht-abstrakte) Methode `tragen()`, die eine Referenz auf das getragene Objekt liefert.

f) Passen Sie die Klassen `Hobbit` und `Magier` entsprechend an, sodass die Methode `nehmen` dort implementiert wird.

Erweitern Sie die Aufgabenstellung nach eigener Lust und Laune, zB:

- Fügen Sie (auch nicht-tragbaren) Gegenständen einen Preis und Wesen einen „Geldbeutel“ hinzu.
- Fügen Sie Methoden hinzu, mit denen Wesen Gegenstände kaufen können (sofern sie genügend Geld haben).
- Überlegen Sie, wie es implementiert werden kann, dass Wesen mehrere (wieviele? beliebig viele?) Gegenstände in ihrem Besitz haben können.
- Lassen Sie Wesen Gegenstände *anderen Wesen abkaufen* können.
- Fügen Sie eine neue Art von Wesen hinzu, nämlich Pferde.
- Pferde können Dinge oder Wesen tragen - sofern diese „tragbar“ sind. Hobbits und Magier sollten dann tragbar sein, andere Wesen sind nicht tragbar.
- . . . eigene weitere Ideen (?)