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

Übung 9

Einführung in die
Programmierung
SoSe 2023

Themen:
Vererbung, Ober- und Unterklassen, Abstrakte Klassen

Aufgabe 9.1
Definieren Sie drei Klassen `Viereck, Rechteck` und `Quadrat` mit folgenden Eigenschaften:

- Klasse `Viereck`:
  - Ein Viereck besteht aus 4 Punkten, die bei Erzeugung übergeben werden.
  - Die Methode `ecken()` liefert ein Array, das die 4 Eckpunkte des Vierecks enthält.
  - Die Methode `umfang()` berechnet den Umfang (= Summe der Länge aller 4 Seiten) des Vierecks. Nutzen Sie hierzu eine Methode der Klasse `Punkt2D` (in LEA unter dem Reiter „Code“ zum Download).
  - Die Methode `toString()` gibt die String-Darstellung der 4 Eckpunkte zurück.

- Klasse `Rechteck` :
  - Jedes Rechteck ist natürlich auch ein Viereck. Wir betrachten nur achsenparallele Rechtecke.
  - Ein Rechteck ist durch einen Eckpunkt (links unten), die Breite und die Höhe (Ausdehnung in x- bzw. in y-Richtung) bestimmt. Diese Werte werden dem Konstruktor übergeben.
  - Die Methoden `hoehe()` und `breite()` liefern die entsprechenden Werte.
  - Die Methode `flaeche()` berechnet den Flächeninhalt des Rechtecks (aus $breite * hoehe$). Die Berechnung des Umfangs kann ggü. allgemeinen Vierecken vereinfacht werden: Der Umfang eines Rechtecks ist $2 * (breite + hoehe)$.

- Klasse `Quadrat` :
  - Ein Quadrat ist ein Rechteck, bei dem Breite und Höhe gleich sind. Zur Erzeugung eines Objektes genügt also die Angabe eines Eckpunktes und der Kantenlänge.
  - Die Methode `kantenlaenge()` liefert diesen Wert.

Hinweise und Bemerkungen:

- Nutzen Sie die Möglichkeiten zur Vererbung, wo immer dies möglich ist.
- Um bei der Erzeugung eines Rechtecks den Konstruktor der Oberklasse aufrufen zu können, müssen Sie aus den Angaben über „linke untere Ecke, Breite und Höhe“ die Koordinaten der drei anderen Eckpunkte berechnen. Sie können dazu die Methode `verschiebeUm(double dx, double dy)` der Klasse `Punkt2D` nutzen.
- Sie brauchen an keiner Stelle zu prüfen, ob die übergebenen Parameter sinnvolle Werte enthalten. Wir gehen davon aus, dass die 4 angebenen Punkte tatsächlich ein konvexes Viereck bilden und dass die Werte für Breite, Höhe und Kantenlänge nicht negativ sind.

Aufgabe 9.2
In J.R.R. Tolkiens Mittelerde leben Wesen, die über unterschiedliche Lebenserwartungen und Fähigkeiten verfügen.
So kann zB. bei Hobbits die individuelle Lebenserwartung durch Verwundung sinken, während Magier durch Heilung einem Hobbit seine ursprüngliche Lebenserwartung zurückgeben können.

Überlegen Sie sich - **bevor** Sie Java-Code implementieren - eine passende Klassenstruktur, die mindestens die Klassen `Wesen`, `Hobbit` und `Magier` umfasst. Nutzen Sie die Konzepte der Vererbung und - wenn es Ihnen sinnvoll erscheint - der Abstraktion. Zeichnen Sie eine UML-Diagramm!

- Wesen haben i.A. einen Namen (String) und ein Alter (int).
- Manche Wesen (zB Hobbits, Orks, Zwerge) sind sterblich: sie haben eine beschränkte Lebenserwartung (int). Andere Wesen (Magier, Elben) sind unsterblich, dh sie haben keine beschränkte Lebenserwartung.
- Bei Erzeugung eines Objektes sollen der Name, das Alter und ggf. die Lebenserwartung als Parameter übergeben werden.
- Die Instanzmethode `lebendig()` gibt bei sterblichen Wesen an, ob das Alter des Wesens kleiner als die Lebenserwartung ist. Unsterbliche Wesen sind natürlich immer lebendig.
- Die Methode `gruss` liefert die Grussformel eines Wesens. Hobbits grüßen zB. mit „Hallo!“, Magier mit „Sei gegruesst!“.
- Die Instanzmethode `wirdVerwundet(int)` senkt die Lebenserwartung eines sterblichen Wesens um die übergebene Zahl. Je nach Schwere der Verwundung kann ein Wesen auch tödlich verletzt werden. Die Methode soll auf jeden Fall eine entsprechende Meldung auf dem Monitor ausgeben.
- Überschreiben Sie die `toString()`-Methode in geeigneter Weise. Der zurückgegebene String sollte Name, Alter und Lebenserwartung des Wesens bzw. die Meldung „bereits verstorben“ enthalten.
- Objekte der Klasse `Hobbit` haben grundsätzlich eine *maximale* Lebenserwartung von 150 Jahren. Bei Erzeugung eines Hobbit-Objektes braucht die Lebenserwartung also nicht explizit angegeben werden. Die individuelle Lebenserwartung kann aber auch hier - wie bei sterblichen Wesen im Allgemeinen - durch Verwundung sinken.
  Durch Heilung wird die Lebenserwartung wieder auf den maximalen Wert zurückgesetzt.
- Magier verfügen über eine Methode `heilt(Hobbit)`, die dem Hobbit-Objekt die maximale Lebenserwartung zurückgibt - sofern der Hobbit noch einen Funken Leben in sich trägt. Bereits gestorbene Hobbits kann auch ein Magier nicht wiederbeleben. Die Methode soll ebenfalls eine entsprechende Meldung auf dem Monitor erzeugen.
- Schreiben Sie eine Testklasse mit einer `main`-Methode, in der ein Array von Wesen erzeugt wird. Definieren Sie eine Hilfsmethode `status()`, die Auskunft über die erzeugten Wesen und deren jeweiliges Alter und die Lebenserwartung gibt.
  Zum Test können Sie die im Download-Bereich angegebene `main`-Methode verwenden. Die Bildschirmausgabe dieses Tests ist ebenfalls als Kontrollergebnis im Download-Bereich hinterlegt.
- Definieren Sie je nach persönlicher Lust und Laune weitere Klassen von Wesen (Zwerge, Elben, Orks, . . .) mit weiteren zusätzlichen oder einschränkenden Fähigkeiten und Eigenschaften.

Eine Testdatei und die dadurch erzeugte Ausgabe finden Sie im Code-Download-Bereich im LEA-Kurs.