Übung 9

Themen: Vererbung, Abstrakte Klassen, Methodenbindung

Aufgabe 9.1
Definieren Sie eine Klasse zur Repräsentation von Dreiecken, die die abstrakte Klasse `GeoFigur` erweitert.

- Ein Dreieick besteht aus 3 Punkten, die bei Erzeugung übergeben werden.
- Die Methode `ecken()` liefert ein Array, das die 3 Eckpunkte des Dreiecks enthält.
- Die Methode `berechneUmfang()` berechnet den Umfang (= Summe der Länge aller 3 Seiten) des Dreiecks.

Aufgabe 9.2
Welche Ausgabe erzeugt folgendes Programm? Begründen Sie kurz.

```
public class A {

        public A() { }

        public static String getS() {
                return "getS in A";
        }

        public String get() {
                return "get in A";
        }
}

public class B extends A{

        public B() { }

        public static String getS() {
                return "getS in B";
        }

        public String get() {
                return "get in B";
        }
}

public class Test {
        public static void main(String[] args) {
                A a = new A();
                B b = new B();
                A ab = new B();

                System.out.println(a.getS());
                System.out.println(b.getS());
                System.out.println(ab.getS());

                System.out.println(a.get ());
                System.out.println(b.get ());
                System.out.println(ab.get ());
        }
}
```

Aufgabe 9.3
Wir wollen die Personalabtelung eines Unternehmens mit neuer Software unterstützen:

- Die Personalabteilung eines Unternehmens verwaltet - in stark vereinfachter Darstellung ;) - zu jedem Mitarbeitenden den Namen und das Geburtsjahr.
- Jeder Mitarbeiter bekommt außerdem eine (fortlaufend automatisch generierte) Personlanummer `persNr`, die durch eine get-Methode erfragt werden kann.
- Eine `toString`-Methode liefert einen String, der den Namen und die Personalnummer enthält.
- Außerdem gibt es für jeden Mitarbeiter eine Methode, die das Gehalt liefert.

Bei der Gehaltsberechung wird allerdings zwischen Tarifangestellten und Leitenden Angestellten unterschieden:

Für Tarifangestellte gilt:

- Tarifangestellte erhalten eine Gehaltsstufe (das ist ein ganzzahliger Wert zwischen 1 und 12), die bei der Erzeugung eine Objektes angegeben wird.
- Es gibt ein monatliches Grundgehalt, das für alle gleich ist (1250.00 Euro).
- Zum Grundgehalt werden monatlich pro Gehaltsstufe $300,00$ Euro aufgeschlagen.
- Grundgehalt und Stufenaufschlag können sich ändern (zB durch Tarifverhandlungen), die Veränderungen wirken sich dann natürlich auf alle Mitarbeiter des Unternehmens aus.
- Manche Mitarbeiter erhalten zusätzlich noch eine individuell vereinbarte Provision. Bei Einstellung eines neuen Mitarbeiters ist die immer Provision = 0.
- Definieren Sie getter- und setter für all diese Größen (Grundgehalt, Stufenaufschlag, Provision).
- Definieren Sie eine Methode `aufsteigen()`, durch die sich die Gehaltsstufe eines Mitarbeiters um 1 erhöht (falls er nicht bereits in der höchsten Gehaltsstufe 12 ist).

Für Leitende Angestellte gilt:

- Leitende Angestellte („Manager“ sind keiner Gehaltsstufe zugeordnet.
- Leitende Angestellte werden auch nicht nach Tarif bezahlt, sondern erhalten ein Gehalt, das bei der Einstellung (Erzeugung eines Objektes) frei vereinbart wird.
- Auch das Gehalt von Managern kann sich ändern.

Erstellen Sie Klassen `Mitarbeiter`, `TarifMA` (für die tariflich angestellten Mitarbeiter) und `Manager` (für die leitenden Angestellten), die die oben genannter Forderungen erfüllen.
Nutzen Sie die Konzepte von Vererbung und Abstraktion!

(Sie können davon ausgehen, dass alle Methoden nur mit zulässigen Parameterwerten aufgerufen werden. Eine Fehlerbehandlung ist hier nicht nötig.)

Aufgabe 9.4
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 die Lebenserwartung des Hobbit-Objekt auf den Maximalwert zurückgsetzt - 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 Mainklasse 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.