Übungen zu Kapitel 6 Programmieren 2

Aufgabe 6.1 - Formenverwaltung

Schreiben Sie ein Programm zur Verwaltung von Formen und Bäumen.

Dabei verwenden Sie bitte folgende Formen:

a) Klasse: Rectangle

Attribute: `width (private), height (private)`

Funktionen: `calculateScope() (public)`

`calculateArea() (public)`

`setter-Methoden (protected)`

`getter-Methoden (public)`

b) Klasse: Square – erbt von Rechteck

Überlegen Sie was Sie zusätzlich implementieren müssen.

c) Klasse: Tree

Attribute: `age (private), height (private)`

Funktionen: `harvest()`

d) Klasse: FruitTree

Attribute: `fruits(String)`

Funktionen: `harvest()`

Bei einem normalen Bauen kann man nichts ernten, bei einem Obstbaum die Früchte

Übungen zu Kapitel 6 Programmieren 2

Beachten Sie, dass Sie die jeweiligen Funktionen abändern müssen. Zeichnen Sie zuerst ein schematisches Klassendiagramm! Schreiben Sie ein Verwaltungsprogramm der verschiedenen Formen und Bäume. Legen Sie verschiedene Instanzen an und berechnen Sie jeweils den Umfang und die Fläche bzw. geben Sie Alter und Höhe aus und ernten Sie den Baum. Verwenden Sie bei ihrer Umsetzung das Konzept der Vererbung

Hinweis: Benutzen Sie die Getter und Setter für den Zugriff auf die "private" Attribute.

Aufgabe 6.2 - Personenbeziehungen

Schreiben Sie ein Programm, das für jede der im folgenden Text genannten Personen angibt, wen sie liebt, mag und hasst:

```
Jim likes Larry and Jean, but hates Kim. Bob loves Jean, and
likes Larry and Kim.
Jean loves Bob, likes Jim, but hates Kim.
Kim hates Jim, likes Larry and Bob.
Larry loves Martin, and hates Karl and Jean.
```

Die Aufgabe besteht darin, den obigen Text im Programm fest zu hinterlegen und den Text an den entscheidenden Stellen zu splitten. Für jede Person soll eine Liste der likes / loves / hates Beziehungen ausgegeben werden, wie zum Beispiel (erster Satz):

```
Jim: [ likes: Larry, Jean] [loves: ] [ hates: Kim]
```

Das Programm soll so gestaltet sein, dass der obige Text gegen einen vergleichbaren anderen Text ausgetauscht bzw. erweitert werden kann.

Hinweis: Nutzen Sie auch hierfür die Funktionen der Java-Klasse String (vgl. Java-API-Doc).

Aufgabe 6.3 - Buchstabenmixer

Schreiben Sie ein Java-Programm, das ein beliebiges Wort (ein String ohne Leerzeichen) vom Benutzer abfragt. Anschließend sollen die einzelnen Buchstaben des Wortes so umgestellt werden, dass die vordere Worthälfte am Ende des Wortes steht.

Beispiel: Eingabe „Haus“, Ausgabe „usHa“

Hinweis: Nutzen Sie die Funktionen der Java-Klasse String (vgl. Java-API-Dokumentation). Beispielsweise liefert der Aufruf “myString“.substring(2,3) einen String mit dem Buchstaben „*S*“.

Übungen zu Kapitel 6 Programmieren 2

Aufgabe 6.4 - Konstruktor-Varianten anhand einer Person

Schreiben Sie eine Klasse „**Person**“ mit den Attributen „**firstName**“ und „**lastName**“ mit folgenden Konstruktoren:

a) Keine Übergabeparameter

Ausgabe: „Created new person“

b) Übergabeparameter: „firstName“

Ausgabe: „Created person“ + firstName
Weisen Sie zudem den Klassenvariablen die Werte der Übergabeparameter zu.

c) Übergabeparameter: „firstName” und „lastName”

Ausgabe “Created person” + firstName + lastName
Weisen Sie zudem den Klassenvariablen die Werte der Übergabeparameter zu.

Aufgabe 6.5 - Static Student

ID Generator für Personen

Ziel: Verwendung von statischen Variablen

Aufgabe 1

Erstelle eine Klasse `Student` mit folgenden Attributen:

- `firstName`: soll den Vornamen als Text speichern (`private`)
- `lastName`: soll den Nachnamen als Text speichern (`private`)

Weiteres

- Erstelle `getter` und `setter` für alle privaten Attribute!
- Erstelle einen leeren Konstruktor `Student()`, welcher keine Übergabeparameter besitzt und beide Attribute mit einem leeren Wert initialisiert.

Aufgabe 2

Erstelle eine Klasse `program.java`, welche eine `main()` beinhaltet und drei Studenten anlegt:

- `tomStudent { firstName: "Tom", lastName: "Teuer" }`

- `reginaStudent { firstName: "Regina", lastName: "Reich" }`
- `leaStudent { firstName: "Lea", lastName: "Lustig" }`

Sie müssen die Objekte hierbei nicht auf der Konsole/Terminal ausgeben.

Aufgabe 3

Nachdem wir jetzt drei Studenten angelegt haben, interessiert uns zusätzlich die `ID` der Studenten. Für die `ID` gilt folgendes:

- jede `ID` ist eindeutig
- sie ist eine ganze Zahl
- sie beginnt bei 1 und wird automatisch bei Erstellung eines Studenten hochgezählt
- erstelle hierfür eine private Variable `id` im Studenten, welche die `ID` für den Studenten speichert (getter und setter nicht vergessen)
- und erstelle einer statische Variable `static_id` im Studenten, welche die aktuell höchste `ID` für alle Studenten speichert
- ändere den Konstruktor der Studentenklasse, damit die Änderungen vollzogen werden können
- geben Sie dann in der main nach der Erstellung jedes Studenten folgendes aus: `firstName + ": " + id` (mit getter auf das Objekt zugreifen). Das Ergebnis sollte lauten:
  - `Tom: 1`
  - `Regina: 2`
  - `Lea: 3`

Hinweise

- Statische Variablen oder Methoden werden über den KlassenNAMEN aufgerufen
- In einer statischen Methode habt ihr KEIN Objekt der Klasse zur Verfügung, d.h. 'this' wird nicht funktionieren. Ihr könnt nur auf normale Methoden zugreifen, wenn ihr ein Objekt der Klasse erstellt (verwendet man sehr selten)

Beispiel

Klasse `Person` hat die statische Variable `numberOfEyes`, weil egal welches Objekt der Klasse `Person` immer die gleiche Anzahl an Augen besitzen wird. Daher wäre der Aufruf für diese statische Variable `Person.numberOfEyes`.