13.5 Übungsblatt 5

Dieses Blatt vertieft die Kenntnisse des Programmierens mit Zeichenketten. Außerdem wird die Objektorientierung vertieft.

Aufgabe 5.1: *Lesen*

Lesen Sie sich zunächst die folgenden Kapitel im [Javabuch] durch:

- 6.7.1 Konstante Zeichenketten
- 2.3 Individuelle und geteilte Daten von Objekten in Java
- 2.5 Erstes Programmbeispiel mit Objekten
- 10.2 Klassenvariablen und Klassenmethoden
- 10.3 Die this-Referenz
- 10.4.4 Konstruktoren zur Initialisierung
- 10.4.5 Aufruf eines Konstruktors im Konstruktor

Sie sollten Fragen zu diesen Inhalten beantworten können.

13.5.1 Zeichenkettenprogrammierung

Gegeben sei folgendes Kinderlied:

```
static final String text =
                  "Drei Chinesen mit dem Kontrabass\n"
                + "saßen auf der Straße und erzählten sich was.\n"
                + "Da kam ein Mann: Ja was ist denn das?\n"
                + "Drei Chinesen mit dem Kontrabass.\n\n"
                ;
```

Es dient der spielerischen Vermittlung des Unterschieds zwischen Konsonanten und Vokalen. Dazu wird das Lied mehrfach gesungen, ab der zweiten Strophe werden aber *alle* Vokale durch *einen* festen Vokal ersetzt. Wer einen Fehler (also meist den Originalvokal) singt, darf nicht mehr mitsingen. Die letzte Sängerin bzw. der letzte Sänger hat gewonnen.

Aufgabe 5.2: *Text zentriert ausgeben*

In dieser Aufgabe soll *irgendein* Text durch eine Klassenmethode `printCentered` zentriert ausgegeben werden, etwa so:

```
      Drei Chinesen mit dem Kontrabass
saßen auf der Straße und erzählten sich was.
    Da kam ein Mann: Ja was ist denn das?
      Drei Chinesen mit dem Kontrabass.
```

Nur die breiteste Zeile fängt ganz links an. Alle anderen werden entsprechend ihrer Breite eingerückt. Dazu ist es sinnvoll, den Text zunächst in eine Array mit den einzelnen Zeilen zu zerlegen. Das geht mit der Methode `String.split()` recht einfach. Danach bestimmt man das Maximum der Zeichen je Zeile von allen Zeilen (wenn Sie wollen, extra Methode bei Ihnen) und rückt dann alle Zeilen beim Ausgeben ein.

Es genügt eine Lösung für nichtproportionale Schriftarten. Für proportionale Schriftarten (Helvetica oder Arial beispielsweise, hier ist ein "M" viel breiter als ein "i") ist diese Aufgabe wesentlich komplexer ...

1. Bauen Sie bitte eine Klasse `DreiChinesen`, die den obigen Text als Klassenvariable speichert.
2. Implementieren Sie die Klassenmethode `printCentered`.
3. Testen Sie bitte Ihre neue Methode in der `main`-Methode ausgiebig (gar kein Text, langer Text, nur eine Zeile, ...).
4. Dokumentieren Sie Ihre Methoden mit Dokumentationskommentaren.

Aufgabe 5.3: *Vokale im Text ersetzen*

In dieser Aufgabe soll wieder bei *irgendeinem* Text durch eine Klassenmethode `changeVowels` die Vokale ersetzt werden. Die (zentrierte) Ausgabe sieht dann so aus:

```
      Drii Chinisin mit dim Kintribiss
sißin iif dir Strißi ind irzihltin sich wis.
    Di kim iin Minn: Ji wis ist dinn dis?
      Drii Chinisin mit dim Kintribiss.
```

Implementieren Sie die weitere Klassenmethode `changeVowels`.

1. Nutzen Sie möglichst *vorhandene* Methoden der `String`-Klasse, wie beispielsweise `length(), substring(), replace()`.
2. Testen Sie bitte Ihre neue Methode in der `main`-Methode ausgiebig (Umlaute, leerer Text, langer Text, ...).
3. Dokumentieren Sie Ihre Methoden mit Dokumentationskommentaren.

13.5.2 Objektorientierung

In den folgenden Aufgaben soll der Umgang mit Klassen, Instanziierungen, Referenzen, Gettern und Settern geübt werden.

Aufgabe 5.4: *Eigene Klassen und Instanzen erstellen*

Als Einstieg in die Objektorientierung soll in dieser Aufgabe ein Bauer mit Hühnern modelliert werden.

a) Erstellen Sie zunächst eine neue Klasse „`Huhn`“. Diese Klasse soll die Instanzvariablen „`name`“ und „`hungrig`“ bekommen. Beide Instanzvariablen sollen den Zugriffsmodifikator `private` erhalten.

   Während der Name des Huhns durch einen Übergabeparameter im Konstruktor der Klasse gesetzt wird, ist jedes Huhn zu Beginn standardmäßig hungrig.

b) Damit sich jemand um die Hühner kümmern kann, erstellen Sie bitte eine weitere Klasse „`Bauer`“ mit einer einzigen Instanzvariablen „`name`“. Wie auch bei der Klasse „`Huhn`“, soll auch der Name eines Bauern im Konstruktor übergeben werden.

c) Ein Bauer unserer Modellierung kann immer nur ein einzelnes Huhn füttern. Hierzu erhält die Klasse „Bauer“ die Methode `fuettern()`. Dieser Methode soll eine Referenz auf ein Objekt der Klasse „Huhn“ übergeben werden können und dessen Attributwert „hungrig“ auf `false` gesetzt werden.

   Da durch den Zugriffsmodifikator `private` außerhalb der Klasse „Huhn“ nicht auf die Instanzvariable „hungrig“ zugegriffen werden kann, verwenden Sie bitte eine Setter-Methode `setHungrig()` in der Klasse „Huhn“, um von außen Änderungen an dieser Instanzvariablen zuzulassen.

d) Für die zweite Instanzvariable der Klasse „Huhn“ benötigen wir keine Setter-Methode, da der Bauer den Namen des Huhns nicht verändern können soll. Falls er ein Huhn zu sich rufen möchte, soll er den Namen über die Methode `getName()`, welche durch die Klasse „Huhn“ bereitgestellt wird, auslesen können. In der Klasse „Bauer“ soll es dafür die Methode `rufeHuhn()` geben. Diese Methode nimmt wiederum eine Referenz auf ein Objekt der Klasse „Huhn“ entgegen und gibt den Namen des entsprechenden Huhns auf der Konsole aus.

e) Erstellen Sie nun in einer `main`-Methode den Bauern „Walter“, indem Sie ein Objekt der Klasse „Bauer“ erzeugen. Anschließend instanziieren Sie die Klasse „Huhn“ durch die drei Hühner „Heidrun“, „Heike“ und „Hannelore“.

   Lassen Sie Bauer Walter jedes der drei Hühner zu sich rufen und füttern. Am Ende der `main`-Methode prüfen Sie dann, mithilfe des Debuggers, ob alle Hühner satt geworden sind.

Aufgabe 5.5*: *Klassenvariablen*

a) Damit ein Bauer selbst prüfen kann, ob jedes Huhn satt geworden ist, soll er beim Füttern mitzählen. Erzeugen Sie hierfür die private Instanzvariable „anzahlGefuetterterHuehner“ in der Klasse „Bauer“ und inkrementieren Sie diese Variable innerhalb der Methode `fuettern()` um eins.

   Gleichzeitig muss bekannt sein, wie viele Huehner existieren. Dies soll durch eine private Klassenvariable „anzahlHuehner“ in der Huhn-Klasse realisiert werden. Diese Variable ist bei jedem Konstruktoraufruf der Klasse „Huhn“ um eins zu erhöhen und soll durch die Getter-Methode `getAnzahlHuehner()` von anderen Klassen aus auslesbar sein.

   Nun benötigt der Bauer noch eine Methode `berechneAnzahlHungrigerHuehner()`, welche die Anzahl der hungrigen Hühner auf der Konsole ausgibt. Testen Sie diese Methode in Ihrer `main`-Methode.

b) Was passiert, wenn Walter ein Huhn zweimal füttert? Beheben Sie den Fehler durch eine geeignete Abfrage in der `fuettern()`-Methode. Tipp: Implementieren Sie eine weitere Getter-Methode in der Klasse „Huhn“.

c) Was passiert, wenn ein weiterer Bauer Walter beim Füttern unterstützt? Wie können die beiden gemeinsam die bereits gefütterten Hühner zählen?

Aufgabe 5.6*: *Analyse und Ergänzung von gegebenen Klassen*

In ILIAS findet man im Verzeichnis „`Skript und Programmbeispiele`“ im Unterverzeichnis „`k04_types“ die Datei „class_examp.zip“` und „Person`.java“` aus dem Unterricht. Diese muss im Verzeichnis "k04_types" kopiert bzw. ausgepackt werden.

a) Übertragen Sie aus diesen Dateien bitte die Klassen „`Raum`“ „`Raumzuordnung`“ und „`Person`“ in das Paket „`blatt05`“. Die Klassen sollen dabei in jeweils einer eigenen Datei stehen (d.h. in „`Raum.java“, „Raumzuordnung.java`“ und „`Person.java“`). Bringen Sie bitte die `main`-Methode der Klasse „`Raumzuordnung“` zum Laufen.

> **TODO-Kommentare** sind beim Programmieren oft hilfreich, wenn es darum geht, nichts Wichtiges zu vergessen. Wenn man programmiert, fällt einem oft noch irgendetwas Wichtiges ein, was man nicht sofort erledigen kann, da man ja zurzeit auch etwas Wichtiges programmiert. Früher hat man so etwas auf den berühmten „Zettel“ notiert und durchgestrichen, wenn es erledigt war. Heute notiert man solche Gedankenblitze als TODO-Kommentare.
>
> Ein Vorteil ist, dass diese in Eclipse dargestellt werden (blaue Kästchen rechts des Scrollbars einer Datei, dort, wo auch die roten Fehlerkästchen dargestellt werden). Ein weiterer Vorteil ist, dass man im Team arbeiten kann. Wenn zum Beispiel ein Kollege krank wird, kann man dessen TODOs fertigmachen, bevor die Software raus geht.
>
> In Eclipse können Sie sich Ihre TODOs mit `Window->Show View->Tasks` in einer Liste anzeigen lassen.

b) In den Klassen „`Raum`“ und „`Raumzuordnung`“ befinden sich solche TODO-Kommentare. Bearbeiten Sie diese bitte. Beachten Sie dabei bitte auch, dass eine neue Instanzvariable für den „`Grund für die Zuordnung`“ wiederum einen Getter braucht und auch in der `print`-Methode und den Testbeispielen der `main`-Methode berücksichtigt werden müssen.

Die Ausgabe sollte am Ende in etwas so aussehen:

```
Bröckl, Ulrich nutzt den Raum: Kürzel: E 212A, Plätze: 2, Plätze bei Klausur: 0, ohne Audio als Büro
    Außerdem: Kürzel: E 201, Plätze: 77, Plätze bei Klausur: 25, mit Audio

Vogelsang, Holger nutzt den Raum: Kürzel: E 209, Plätze: 3, Plätze bei Klausur: 1, ohne Audio als Büro
```