MINT-KOLLEG BADEN-WÜRTTEMBERG

Universität Stuttgart

Vorkurs Informatik

Dr. Sebastian Müller
Dr. Frank Schweiner

---

5. Tutoriumsblatt

Bearbeiten Sie dieses Übungsblatt im Projekt `Day5` und legen Sie für jede Aufgabe ein Paket `task01`, `task02`, ... an.

Aufgaben Vormittag

1. Aufgabe

Verkehr / Traffic

Auf einer Durchgangsstraße wurden eine Woche lang die durchfahrenden Autos gezählt. Dabei ergab sich folgende Tabelle:

Tag	1	2	3	4	5	6	7
#Autos	3465	3768	4067	3976	4782	2057	2954

Hinweis: Sie benötigen für diese Aufgabe nur eine Klasse `Test` mit der `main`-Operation, in der Sie alle Aufgabenteile bearbeiten.

a) Speichern Sie die Anzahl Autos in einer Liste. Beachten Sie, dass der erste Index der Liste immer 0 ist, d.h. das Element mit dem Index 0 enthält die Auto-Anzahl von Tag 1.

b) Geben Sie die Größe der Liste aus.

c) Geben Sie jedes Element der Liste einzeln aus. Nutzen Sie hierfür einmal eine `for`-Schleife und einmal eine for-each-Schleife.

d) Berechnen Sie die durchschnittliche Anzahl Autos während der sieben Tage.

e) Lassen Sie Ihr Programm den Tag mit maximalem Verkehr finden und ausgeben.

f) Beim Erstellen der Tabelle ist etwas schief gegangen. Tauschen Sie die Werte der Tage 4 und 5. (Tipp: Zusätzliche Variable als Zwischenspeicher)

g) Erzeugen Sie eine zweite Liste und speichern Sie in dieser die Werte aus der ersten Liste in umgekehrter Reihenfolge.

h) Sortieren Sie die Liste mit dem Bubblesort-Algorithmus.

Bitte wenden!

Aufgaben Nachmittag

2. Aufgabe

Bauarbeiter / Worker

Auf einer Baustelle gibt es unterschiedliche Arbeiter (`Worker`): Maler (`Painter`), Maurer (`Bricklayer`) und Tischler (`Carpenter`). Allen gemeinsam ist, dass jeder von Ihnen ein Honorar erhält (das aber ganz unterschiedlich ausfallen kann) und mit einem bestimmten Material arbeitet.

a) Legen Sie eine Enumeration `WorkingMaterial` an mit folgenden Werten: `WOOD`, `PAINT`, `BRICKS`.

b) Schreiben Sie eine abstrakte Klasse `Worker`. Diese soll die Attribute für Honorar `double salary` und Arbeitsmaterial `WorkingMaterial workingMaterial` deklarieren. Schreiben Sie den Konstruktor sowie Getter-Methoden. Schreiben Sie außerdem eine Setter-Methode für das Honorar. Denken Sie an Exceptions!

c) Schreiben Sie eine Operation `void performAction()`, die einfach nur den Text `"Working ..."` ausgibt. Diese soll später in den Unterklassen geeignet überschrieben werden.

d) Legen Sie nun die Unterklassen für Maler (`Painter`), Maurer (`Bricklayer`) und Tischler (`Carpenter`) an. Da Konstruktoren in Java nicht vererbt werden, müssen Sie in jeder Klasse einen eigenen Konstruktor schreiben. Dieser soll mittels `super(..)` den Konstruktor der Oberklasse aufrufen und die Attribute initialisieren. Ein Maler arbeitet dabei mit Farbe, ein Maurer mit Steinen und ein Tischler mit Holz. Tischler führen immer eine bestimmte Anzahl Nägel mit sich – mindestens 100, aber mehr als 1000 passen nicht in den Beutel. Ergänzen Sie ein geeignetes Attribut.

e) Alle Unterklassen sollen die Operation `void performAction()` der Oberklasse überschreiben. Im Methodenrumpf soll die Operation der Oberklasse aufgerufen werden und anschließend ein auf den Bauarbeiter angepasster Text ausgegeben werden: `"painting a wall!"`, `"laying bricks!"`, `"hammering nails!"`. Das Einschlagen von Nägeln soll fünfmal geschehen. Dabei soll sich die Anzahl der Nägel reduzieren. Stellen Sie sicher, dass der Tischler genügend Nägel hat, um die Aktion ausführen zu können.

f) Legen Sie eine Klasse `Test` mit der `main`-Methode an. Setzen Sie in dieser folgendes Szenario um:

> Auf einer Baustelle arbeiten ein Maler namens Paul (Grundgehalt: 23 Euro), eine Tischlerin namens Paula (Grundgehalt: 25 Euro) und ein Maurer names Ronnie (Grundgehalt: 18 Euro). Um zu zeigen, was sie können, sollen alle Bauarbeiter Ihre Aktion ausführen. Anschließend erhält Ronnie eine Gehaltserhöhung von 5 Euro.

g) *Zusatzaufgabe:* In Java darf als statischer Typ von Objekten der Bezeichner der Oberklasse verwendet werden. Beispielsweise kann man also bei Paul auch folgendes schreiben: `Worker paul = new Painter(23);`. Setzen Sie obiges Szenario so um, dass alle Arbeiter den statischen Typ `Worker` besitzen. Fügen Sie sie dann zu einer Liste hinzu und lassen Sie mittels einer for-each-Schleife die Aktionen ausführen.

Bitte wenden!

3. Aufgabe

Datenbank / Database

Teil 1: Student

a) Erstellen Sie eine Klasse `Student`, in der Name `name`, Alter `age`, Matrikelnummer `matriculationNumber` und Studiengang `course` gespeichert werden sollen. Schreiben Sie einen Konstruktor sowie Getter-Methoden. Auf Exceptions dürfen sie hier verzichten.

b) Schreiben Sie eine Methode `equals(Student student)`, die überprüft, ob die Werte des aufrufenden Objektes inhaltsgleich sind mit den Werten des Argumentes. Beispiel: Angenommen, es existieren zwei Studierende `paul = new Student("Paul",20,1234,"informatics");` und `paula = new Student("Paula",20,1235,"informatics");`. Beim Methodenaufruf `paul.equals(paula);` sollen die Attributwerte verglichen werden. In diesem Beispiel müsste der Methodenaufruf `false` zurückliefern.

c) Schreiben Sie eine Methode `toString()`, die die Daten als String zurückgibt. Beispiel: Für Paul soll `"Student: (Paul,20,1234,informatics)"` zurückgegeben werden.

d) Testen Sie die Klasse `Student` und deren Methoden mit Paul und Paula (siehe oben).

Teil 2: Database

a) Schreiben Sie eine Klasse `Database` mit dem privaten Attribut `List<Student> students` und einem Konstruktor, in dem eine leere(!) ArrayList erzeugt wird.

b) Schreiben Sie eine Methode `addStudent(Student student)`, mit der ein weiteres `Student`-Objekt zu Liste hinzugefügt wird.

c) Schreiben Sie eine Methode `studentIsPresent(Student student)`. Diese soll die Liste durchgehen und mittels `equals` überprüfen, ob das `Student`-Objekt bereits vorhanden ist.

d) Passen Sie die Methode `addStudent(Student student)` so an, dass das `Student`-Objekt nur dann zur Liste hinzugefügt wird, wenn es nicht bereits vorhanden ist. Ansonsten soll eine Exception geworfen werden.

e) Schreiben Sie eine Methode `removeStudent(Student student)`. Diese soll die Liste durchgehen und das betreffende `Student`-Objekt aus der Liste entfernen. Ist das `Student`-Objekt nicht vorhanden, soll einfach nichts passieren.

f) Schreiben Sie eine weitere Methode, welche das Durchschnittsalter der Personen in der Datenbank zurückgibt.

g) Schreiben Sie eine Methode, über welche die gesamte Liste auf der Konsole ausgegeben wird.

h) Schreiben Sie eine Methode, welche die Liste nach den Namen der `Student`-Objekt über den Bubblesort-Algorithmus sortieren soll. Verwenden Sie hierzu die Methode `int compareTo(String str2)` der Klasse `String`.

Teil 3: Test

a) Testen Sie ihre Datenbank! Erzeugen Sie hierzu fünf `Student`-Objekte Ihrer Wahl. Erstellen Sie eine Datenbank und fügen Sie die `Student`-Objekte hinzu.

b) Versuchen Sie ein `Student`-Objekt doppelt hinzuzufügen.

c) Sortieren Sie die Liste und lassen Sie sich die Liste ausgeben.

Zusatzaufgaben

4. Aufgabe

Primzahlzwillinge / Twin prime

Bei einem Primzahlzwilling handelt es sich um zwei benachbarte Primzahlen, deren Differenz 2 ist (Zum Beispiel 3 und 5 oder 11 und 13). Schreiben Sie ein Programm, dass Ihnen alle Primzahlzwillinge im Zahlenbereich von 2 bis 200000 ausgibt in der Form

```
1.  3, 5
2.  5, 7
...
2160. 199931, 199933
```

a) Wir wollen zunächst möglichst schnell alle Primzahlen zwischen 2 und 200000 finden. Gehen Sie dabei folgendermaßen vor: Schreiben Sie eine Schleife über alle Zahlen $i = 2 \ldots 200000$. Um zu testen, ob eine beliebige Zahl i eine Primzahl ist, überprüfen Sie, ob sie durch eine beliebige Primzahl $p < i$ teilbar ist. Jede neu gefundene Primzahl wird dann zu einer Liste hinzugefügt.

b) Sollte sich zeigen, dass i durch eine Zahl teilbar ist, können Sie die Rechnung für diese Zahl abbrechen, um die Programmlaufzeit zu verringern.

c) Gehen Sie Ihre Liste durch und lassen Sie sich alle Primzahlzwillinge ausgeben.

5. Aufgabe

Leute / People (sehr schwer)

10000 Leute stehen durchnummeriert im Kreis. Nacheinander muss jeder Dritte gehen. Welche 2 Personen bleiben übrig? (Richtige Lösung: Person Nummer 2692 und 8923.)