Bibliotheksklassen II

Lernziele

- Sie können die Objektsammlung HashMap zur Speicherung und Verwaltung von Schlüssel-Wert Paaren einsetzen.
- Sie verstehen die Bedeutung des Schlüsselwortes `final`.
- Sie können erkennen, ob eine Klasse das Geheimnisprinzip verletzt und eine geeignete Verbesserung vorschlagen.
- Sie setzen die für den jeweiligen Zweck passende Objektsammlung ein.
- Falls nötig können Sie gezielt Informationen in Klassendokumentationen nachschlagen um die jeweilige Funktionalität anschliessend in Ihrem Programm verwenden zu können.

Aufgabe 1 (auf Papier!)

Wenn Sie die Ausgabe der Noten aus Praktikum 5-1 Aufgabe 3 so erweitern müssten, dass die Noten zusätzlich in Text ausgegeben werden, wie würden Sie die String Zuweisung zu den nummerischen Notenwerten am besten realisieren?

```
Herzliche Gratulation Max Muster! Sie haben an der Pruefung eine 5.5 (fünf
punkt fünf) erzielt und somit bestanden!
Susi Mueller, Sie haben an der Pruefung eine 3.5 (drei punkt fünf) erzielt und
sind somit durchgefallen!
Herzliche Gratulation Heinz Moser! Sie haben an der Pruefung eine 4.0 (vier
punkt null) erzielt und somit bestanden!
```

```
Map<Double, String> notenMitWorten = new HashMap<>();

notenMitWorten.put(1.0, "eins punkt null");
notenMitWorten.put(1.5, "eins punkt fünf");
notenMitWorten.put(2.0, "zwei punkt null");
notenMitWorten.put(2.5, "zwei punkt fünf");
notenMitWorten.put(3.0, "drei punkt null");
notenMitWorten.put(3.5, "drei punkt fünf");
notenMitWorten.put(4.0, "vier punkt null");
notenMitWorten.put(4.5, "vier punkt fünf");
notenMitWorten.put(5.0, "fünf punkt null");
notenMitWorten.put(5.5, "fünf punkt fünf");
notenMitWorten.put(6.0, "sechs punkt null");
```

Aufgabe 2 (auf Papier!)

Gegeben sei die folgende Klasse:

```
public class MeinObjekt {
	private int nummer;

	public MeinObjekt(int nummer) {
		setNummer(nummer);
	}

	public void setNummer(int nummer) {
		this.nummer = nummer;
	}
}
```

Ihr Kollege hat soeben die Bedeutung von `final` gelernt. Zum Testen schreibt er folgende Klasse:

```
public class FinalVerstehen {
	private final int nummer = 1;
	private final MeinObjekt meinObjekt = new MeinObjekt(1);

	public void eineMethode() {
		nummer = 10;
		meinObjekt = new MeinObjekt(1);
		meinObjekt.setNummer(2);
	}
}
```

Er teilt Ihnen mit das Zeile 1 und 2 der Methode `eineMethode()` zu einem Fehler beim Kompilieren führt, Zeile 3 jedoch nicht. Er hätte jedoch erwartet, dass auch Zeile 3 zu einem Kompilierfehler führt. Erklären Sie ihm wieso dem nicht so ist:

Die 1 Zeile für zur ein Kompilierfehler; Die 2 Zeile wird auch ein Fehler ausgegeben,

da den Objekt meinObjekt schon initialisiert wurde; Die 3 Zeile wird 1 ausgegeben,

da den Objekt meinObjekt als final gesetzt wurde.

Aufgabe 3 (auf Papier!)

Die Klasse `Auto` verletzt das Geheimnisprinzip:

```
public class Auto {
	public String kennzeichen;
	...
	public boolean setAutokennzeichen(String autoKennzeichen) {
		if (istGueltigesKennzeichen(autoKennzeichen)) {
			kennzeichen = autoKennzeichen;
			return true;
		}
		return false;
	}
	...
}
```

Welches Geheimnisprinzip wird verletzt? Was ist das Problem, welches daraus entsteht? Wie kann es korrigiert werden?

Die String kennzeichen wird als public gesetzt. Da den zweiten Prinzip des Geheimnis-

prinzip verletzt ist, wird die Interna Kenntnis erlaubt. String kennzeichen sollte den

Zugriffsmodifikator private haben.

Nachdem Sie ihre Korrektur angebracht haben kriegen Sie den Auftrag, die in `setAutokennzeichen` erhaltene Zeichenkette zu trennen und das Kantonskennzeichen und die Nummer separat abzuspeichern. Das Kantonskennzeichen soll zudem über eine sondierende Methode abgefragt werden können. Ist es möglich die Änderung durchzuführen, ohne dass andere Klassen, welche momentan diese Klasse verwenden, geändert werden müssen? Begründen Sie Ihre Antwort

Am bestens wäre die Methode für den Kantonskennzeichen separat in eine andere Klasse

zu schreiben, sodass es die Modularisierung respektiert und daher die Geheimnisprinzipien.

Aufgabe 4

Im Buch haben Sie gelernt was statische Variablen sind. Manchmal ist es nötig eine statische Variable zu initialisieren, welche mehr als eine Zuweisung benötigt. Die Lösung zum Problem nennt man „statische Initialisierung“. Lesen Sie die folgenden Artikel zur statischen Initialisierung:

- http://www.dpunkt.de/java/Die_Sprache_Java/Objektorientierte_Programmierung_mit_Java/8.html
- https://docs.oracle.com/javase/tutorial/java/javaOO/initial.html

Aufgabe 5

Forken Sie für diese Aufgabe das Projekt https://github.engineering.zhaw.ch/prog1-kurs/05_Praktikum-2_Wortstatistik. Nutzen Sie BlueJ um die eigene Projektkopie auf Ihren Computer zu holen und zu bearbeiten.

Schreiben Sie ein Programm zum Analysieren der Worthäufigkeit in Texten. Schreiben Sie dazu ein Programm, welchem Sie mehrere Texte übergeben können. Dabei soll jederzeit die momentane Statistik (also alle Wörter und deren Häufigkeit) ausgegeben werden können.

Hinweis: Reguläre Ausdrücke oder eine Objektsammlung mit den zu entfernenden Satzzeichen könnten hier von Nutzen sein. Wenn Sie sich für die Objektsammlung entscheiden, dann fügen Sie der Sammlung die Werte mit einem Static Initialization Block (s. Aufgabe 4) hinzu. Überlegen Sie sich aus welchem Grund sich hier die statische Initialisierung anbietet.

```
Worthaeufigkeitsanalyse hauefigkeitsanalyse = new Worthaeufigkeitsanalyse();
hauefigkeitsanalyse.verarbeiteText("Fritz sagt: \"Die Softwareentwicklung ist
meine Leidenschaft!\"");
hauefigkeitsanalyse.verarbeiteText("Hans meint, er teile die Leidenschaft mit
Fritz.");
hauefigkeitsanalyse.verarbeiteText("John fuegt hinzu, dass die
Softwareentwicklung nicht nur aus Programmieren bestehe, sondern es sich dabei
um einen komplexen Prozess, bestehend aus vielen kleinen Komponenten,
handelt.\"");
hauefigkeitsanalyse.druckeStatistik();
```

Bei der Implementierung müssen Sie dabei auf folgende Dinge Rücksicht nehmen:

- Satz-[1] und Leerzeichen sollen nicht gewertet werden.
- Gross- und Kleinschreibung soll beim Zählen nicht berücksichtigt werden. **D**ie und **d**ie werden also z.B. als dasselbe Wort betrachtet.

Die obigen Probleme können Sie allesamt mit der Klasse `java.lang.String` lösen. Nehmen Sie unbedingt die Dokumentation zur Hilfe.

Gehen Sie schrittweise vor („teile und herrsche“):

- Teilen Sie den Text in Worte auf.
- Entfernen Sie die Satzzeichen.
- Behandeln Sie das Problem der Gross- und Kleinschreibung.
- Aktualisieren Sie den Zähler des entsprechenden Wortes. Überlegen Sie sich gut, welche Datenstruktur sich hierzu am besten eignet.

Hinweis: Wenn Sie mit einer for-each Schleife alle Elemente in einer Map verarbeiten möchten, dann müssen Sie die Elemente in der Map zuerst in ein Set überführen, welche dann in einer for-each Schleife verwendet werden kann. Studieren Sie dazu die Methoden `entrySet()` bzw. `keySet()` der Klasse `HashMap`.

Die Ausgabe des Programmes sollte in etwa wie folgt strukturiert sein:

---

1 Sie müssen lediglich die Satzzeichen `.,?!":;` beachten.

```
hans1
prozess      1
nicht  1
john   1
kleinen      1
meine  1
softwareentwicklung       2
handelt      1
meint  1
sagt   1
sondern      1
die    3
komponenten  1
fritz  2
dabei  1
mit    1
teile  1
bestehend    1
einen  1
ist    1
um     1
komplexen    1
leidenschaft 2
fuegt  1
vielen 1
hinzu  1
aus    2
programmieren      1
dass   1
es     1
er     1
bestehe      1
nur    1
sich   1
```

Buchstabenhäufigkeit (Optional)

Erweitern Sie Ihr Programm so, dass auch die Buchstabenhäufigkeit erhoben wird. Verarbeitens sie grosse Texte und vergleichen Sie Ihre Resultate mit jenen von Wikipedia.[2]

Hintergrund: Die Buchstabenhäufigkeitsanalyse findet praktische Anwendung. Im weiteren Verlauf Ihres Studiums werden Sie in der Kryptographie bzw. Kryptoanalyse sicherlich nochmals darauf stossen.

Einlesen von Datei (Optional)

Lesen die Zeichenketten aus einer oder mehrere Dateien ein.

[2] http://de.wikipedia.org/wiki/Buchstabenhäufigkeit