Javakurs SS03 - Übung 8

1 Read.java

Probiere das Programm Read.java (auf der Kursseite verlinkt) aus.

2 Ausnahmefläche

Hat eine Dreieck die Seitenlängen a, b und c, so ist die Fläche des Dreiecks $\sqrt{s(s-a)(s-b)(s-c)}$ mit $s = \frac{1}{2}(a+b+c)$. Schreibe ein Programm, das drei Zahlen von der Kommandozeile als Seitenlängen nimmt und die Fläche ausgibt. Behandle die Ausnahmen, die dabei auftreten können (es gibt Seitenlängen, für die kein Dreieck existiert).

3 Zahl von der Kommandozeile einlesen

Schreibe ein Programm, das den Benutzer nach einer Zahl fragt, eine Eingabe (mit return abgeschlossen) vom Benutzer einliest und die Eingabe in eine Zahl (`int`) umwandelt. Ist die Eingabe keine gültige Zahl, so teile dies dem Benutzer mit und lese erneut die Zahl ein, bis eine gültige Eingabe da ist.

Sieh Dir dazu an, welche Ausnahmen bei `Integer.parseInt(String)` auftreten können.

4 Unix grep

Schreibe ein Programm, das einen String `s` und einen Dateinamen als Kommandozeilenparameter nimmt und alle Zeilen der Datei ausgibt, die den String enthalten.

Hinweis: Benutze am besten `FileReader`, `BufferedReader` und ggf. `File`. Um festzustellen, ob ein String `s` innerhalb eines Strings `line` vorkommt, kannst Du den Aufruf `line.indexOf(s)` gebrauchen, siehe dazu API Dokumentation zu `java.lang.String`.

5 Datei kopieren

Schreibe ein Programm, das zwei Dateinamen als Parameter nimmt, eine Quelle und eine Ziele. Das Programm soll die Quelldatei zum Ziel kopieren (binär, also mit Streams).

Hinweis: Benutze am besten `FileInputStream`, `FileOutputStream` sowie `BufferedInputStream`, `BufferedOutputStream` und ggf. `File`.

6 Dateien vergleichen (Unix cmp)

Schreibe ein Programm, das zwei Dateien nach Unterschieden untersucht. Bei dem ersten Byte, das in beiden Dateien verschieden ist, soll die Position (byte count) ausgegeben werden und das Programm beendet werden.

Hinweis: Benutze am besten `FileInputStream`, `BufferedInputStream` und ggf. `File`.

7 Texteditor

Schreibe ein Fensterprogramm mit einem `javax.swing.JTextArea` (zum Text anzeigen und editieren) und einem Bedienelement (`JButton` oder Menue) zum Speichern des Textes in eine Datei. Das Programm sollte eine Textdatei als Kommandozeilenparameter nehmen und deren Inhalt bei Programmstart ins `JTextArea` laden.

8 Insel der Logik*

Die Insel der Logik hat drei Sorten von Einwohnern: himmlische Wesen (*devine*), die immer die Wahrheit sagen, böse (*evil*), die immer lügen und Menschen (*human*), die tagsüber die Wahrheit sagen und nachts lügen. Jeder Einwohner weiß von den anderen Einwohnern, von welcher Art diese sind. Ein Sozialwissenschaftler will die Insel besuchen. Da er nicht nach dem Aussehen schon zwischen den drei Einwohnerarten unterscheiden kann (fragt mich nicht, warum er offenbar auch nicht zwischen Tag und Nacht unterscheiden kann), bittet er Dich, ein Programm zu schreiben, daß die Gespräche mit den Einwohnern analysiert und daraus die Fakten extrahiert. Interessante Fakten sind die Arten der Einwohner und ob es Tag oder Nacht ist.

Die Eingabe Erfolgt über eine Datei. (Eine Beispieleingabe ist auf der Vorlesungsseite verlinkt.) Es enthält gleich mehrere Gesprächsbeschreibungen. Jede Gesprächsbeschreibung beginnt mit einer Zeile, in der eine ganzen Zahl *n*

(höchsten 50) steht, die Anzahl der Aussagen in der Gesprächsbeschreibung. Das Ende der Datei markiert eine Gesprächsbeschreibung mit n=0 Aussagen. Jede Aussage beginnt mit einem Sprecherpseudonym, ein Großbuchstabe von *A* bis *E*, gefolgt von einem Doppelpunkt und dahinter eine der folgenden Aussagen:

- `I am [not] ( divine | human | evil | lying ).`
- *X* `is [not] ( divine | human | evil | lying ).`
- `It is ( day | night ).`

Eckige Klammern bedeutet, daß der Inhalt optional ist, runde Klammern, daß genau eines der durch den geraden Strich getrennten Wörter vorkommt.

Als Ausgabe soll die Meldung herauskommen, daß der Fall unmöglich ist, das keine Fakten gefunden werden konnten oder alle Fakten, die herausgefunden werden konnten.

Beispiel:

- Eingabe:

```
1
A: I am divine.
1
A: I am lying.
1
A: I am evil.
3
A: B is human.
B: A is evil.
A: B is evil.
```

- Ausgabe

```
Conversation #1
No facts are deducible.

Conversation #2
This is impossible.

Conversation #3
A is human.
It is night.
```

```
Conversation #4
A is evil.
B is divine.
```

Warum kann man daraus Schlüsse ziehen? Sehen wir uns die dritte Beispieleeingabe an. Wenn A sagt *I am evil.*, so kann er nicht göttlich sein, denn dann müßte er die Wahrheit sagen. Er kann auch nicht böse sein, denn dann könnte er nicht verraten, daß er böse ist. Und da er gelogen hat, muß es Nacht sein.

Im vierten Gespräch, lügt A offenbar, denn er macht zwei sich widersprechende Aussagen. Also kann B weder menschlich noch böse sein, folglich ist er göttlich und damit seine Aussage wahr: A ist böse.

Lösungsansatz: Man kann das Problem „brute force“ lösen. Es gibt $2 * 3^5$ verschiedene Zustände für die drei möglichen Arten der fünf Personen bei Tag oder Nacht. Man macht sich eine Tabelle für diese Möglichkeiten, in der steht, ob diese Möglichkeit noch widerspruchsfrei ist. Für jede Aussage streicht man alle unmöglichen Kombinationen. Für die Ausgabe wertet man dann die Tabelle aus.

Zum „Parsen“ der Aussagen sind die `String`-Methoden

```
boolean contains(String)
boolean equals(String)
char charAt(String)
String substring(int, int)
int indexOf(String)
```

nützlich. Siehe auch API Doc zu `java.lang.String`.