

## Übung 02: Verzweigungen

Abgabetermin: 27. 10. 2009

Name: \_\_\_\_\_ Matrikelnummer: \_\_\_\_\_

Gruppe: G1 (Prähofer)  G2 (Wolfinger)  G3 (Wolfinger)  G4 (Jahn)

Aufgabe	Pkte	Abzugeben	gelöst	Erreichte Pkte	Kommentar
Aufgabe 02.1	12	Prosabeschreibung Ablaufdiagramm Java-Programm Testfälle und Ergebnisse	<input type="checkbox"/>		
Aufgabe 02.2	12	Prosabeschreibung Java-Programm Testfälle und Ergebnisse	<input type="checkbox"/>		

### Aufgabe 2.1: Einlesen und Prüfen von Email-Adressen

Die Email-Adressen einer Organisation sollen nach folgender Grammatik aufgebaut sein:

```
EMailAddr = name [ "." name ] "@" name [ "." name ] "." ( "at" | "com" | "net" )
```

Zuerst steht ein Name. Dieser kann von einem Punkt und einem weiteren Namen gefolgt sein. Dann kommt das @-Zeichen. Danach auf jeden Fall ein Domain-Namen, eventuell gefolgt von einem weiteren und abschließend "at", "com" oder "net", jeweils mit "." getrennt. Ein name ist aufgebaut wie ein Java-Identifizier und kann mit `In.readIdentifizier` eingelesen werden.

Schreiben Sie ein Java-Programm, das eine potentielle Email-Adresse einliest und überprüft, ob diese korrekt aufgebaut ist.

#### Hinweise:

Verwenden Sie dazu folgende Methoden der `In`-Klasse:

- `readIdentifizier()` liest das nächste Wort ein (liefert Wert von Datentyp `String`), z.B.:  
`String name = In.readIdentifizier();`
- `read()` liest das nächste Zeichen (Datentyp `char`; Achtung, liefert auch Leerzeichen!). z.B.:  
`char ch = In.read();`
- `readCahr()` liest das nächste Zeichen, das kein Leerzeichen ist (Datentyp `char`). z.B.:  
`char ch = In.readChar();`
- `peek()` liefert das nächste Zeichen (Datentyp `char`) ungleich dem Leerzeichen, ohne es zu lesen. (Achtung, Leerzeichen werden überlesen!)  
`char ch = In.peek();`
- `done()` überprüft, ob die letzte read-Anweisung erfolgreich war (liefert `true` oder `false`)  
`if (In.done()) { ...`

Um Strings auf Gleichheit zu vergleichen, verwenden Sie `equals` (und nicht `==`), wie z.B.:

```
if (name.equals("at")) { ...
```

welches `true` liefert, wenn in `name` genau die Zeichenkette "at" gespeichert ist.

#### Abzugeben sind:

- Die Beschreibung des Programms in Prosa
- Ablaufdiagramm
- Das Java-Programm
- Testfälle und die Ergebnisse

## Aufgabe 2.2: Wochentag zu einem Datum

Vor einiger Zeit hat bei der Fernsehsendung „Wetten dass“ ein kleiner Junge durch die Fähigkeit beeindruckt, für ein beliebiges Datum den Wochentag bestimmen zu können. Wir können zwar nicht so gut Kopfrechnen, dafür können wir Java-programmieren. Schreiben Sie ein Java-Programm, das ein beliebiges Datum in der Form von 3 ganzen Zahlen (Tag, Monat, Jahr) einliest und den Wochentag für dieses Datum ausgibt.

Im Folgenden sehen Sie einen Beispieldialog mit dem Programm:

```
Berechnen des Wochentages für ein beliebiges Datum.
=====
Tag: 24
Monat: 12
Jahr: 2009
Der 24.12.2009 ist ein Donnerstag
```

Um die Aufgabe lösen zu können, sind folgende Informationen wichtig:

- Für die Berechnung muss man von einem fixen Datum ausgehen. Wir wählen dafür den 1.1.1900, der ein Montag war. Um die Aufgabe einfach zu halten, wollen wir nur Tage nach dem 1.1.1900 berechnen.
- Wir können die Aufgabe lösen, indem wir die Tage seit dem 1.1.1900 zählen und dann die Anzahl der Tage modulo 7 rechnen. Das Ergebnis der Rechnung ergibt den Wochentag (0 für Montag, 1 für Dienstag, .. 6 für Sonntag).
- Die Anzahl der Tage ergibt sich aus
  - Anzahl der vollen Jahre \* Tage im Jahr
  - Die Tage im Monat vor dem gegebenen Monat (31 für Jänner, 28 oder 29 für Februar, ...).
  - Die Tage im gegebenen Monat minus 1 (warum?).

### Beispiel:

Das gegebene Datum sei der 12. 4. 1903.

Für dieses Datum zählen wir die Tage:

```
3 volle Jahre * 365 + Tage im Jänner + Tage im Februar + Tage im März + 12 Tage im April - 1 =
1095 + 31 + 28 + 31 + 11 (!) = 1196
1196 % 7 = 6 und somit war der 12. 4. 1903 ein Sonntag.
```

- Schwierigkeiten bereiten jetzt nur mehr die Schaltjahre, bei denen das Jahr ja 366 Tage und der Februar 29 Tage hat. Schaltjahre sind alle 4 Jahre, nicht aber zu den durch 100 teilbaren Jahren (z.B. nicht 1900, 2100), aber dann doch wieder zu den durch 400 teilbaren Jahren (d.h. 2000 und 2400 sind doch Schaltjahre). Diese Schaltjahre muss man besonders berücksichtigen. Wir können folgend rechnen:
  - mit folgenden Booleschen Ausdruck kann man bestimmen, ob ein Jahr `year` ein Schaltjahr ist:
 

```
year % 4 == 0 && !(year % 100 == 0 && year % 400 != 0)
```
  - die Anzahl der vollen Schaltjahre vor einem Jahr `year` seit 1900 berechnet sich mit:
 

```
((year-1) / 4 - 474) - ((year-1) / 100 - 18) + ((year-1) / 400 - 4)
```

 Zur Erklärung:  $1899 / 4 = 474$ ,  $1899 / 100 = 18$ ,  $1899 / 400 = 4$ ;

### Hinweise:

- Verwenden Sie für die Aufgabe Integerdivision (`/`) und Restwertbildung (`%`).
- Verwenden Sie `if`-Abfragen (oder `switch`-Anweisung) beim Zählen der Tage der Monate.
- Verwenden Sie eine `switch`-Anweisung für die Unterscheidung des berechneten Wochentags (0 bis 6) und die Ausgabe des entsprechende Namens für diesen Wochentag ("Montag" – "Sonntag").
- Einen `DayOfWeek`-Rechner zum Nachrechnen findet man auf <http://www.searchforancestors.com/utility/dayofweek.html>.

### Abzugeben sind:

- Prosabeschreibung
- Java-Programm
- Ausgaben des Programms bei unterschiedlichen Eingaben (mind. 3 unterschiedliche Eingaben)

Achtung: Kein Ablaufdiagramm verlangt!!!!