Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Einführung in die Informatik
Sommersemester 2018

Prof. Dr. Wolfram Burgard
Andreas Kuhner
Daniel Büscher

Übungsblatt 6

Abgabe bis Sonntag, 03.06.2018, 23:59 Uhr

Hinweis:
Aufgaben immer per E-Mail (eine E-Mail pro Blatt und Gruppe) an den zuständigen Tutor schicken (bei Programmieraufgaben Java Quellcode und evtl. benötigte Datendateien).

Aufgabe 6.1

Beantworten Sie folgende Frage:

a) Was bedeuten die folgenden Zeichen: <, >, ==, <=, >=, !=?

b) In welcher Reihenfolge werden der &&-Operator, ||-Operator und !-Operator ausgeführt?

c) Wozu dienen `while` Ausdrücke?

d) Ist es nötig den Rückgabewert bei einem `next()`-Aufruf zu casten?

e) Worin liegt der Vorteil von generischen Klassen?

f) Was ist der Unterschied zwischen `while(...)` und `for(...)`?

Aufgabe 6.2

Auf der letzten Seite des Übungsblattes finden Sie einige Konventionen für die Formatierung von Java-Code. Betrachten Sie folgendes Programm und korrigieren Sie die Stellen, die nicht mit den Konventionen übereinstimmen.

```
import java.net.*;
import java.io.*;

class webpageread {
  public static void main(String[] arg) throws Exception{
    URL u = new URL("http://www.informatik.uni-freiburg.de/");
    InputStream ins =  u.openStream();
    InputStreamReader isr = new InputStreamReader(ins);
    BufferedReader WebPageBuffer = new BufferedReader(isr);


    int NoOfLines =10;

    if(NoOfLines==0){ return; }

    if(NoOfLines == 1){
    System.out.println("Read " + NoOfLines + " line of " + u );
    }
    else{
```

```
    System.out.println("Read " + NoOfLines + " lines of " + u );
    }

    for(int i=1; i<=NoOfLines ; i++)
    {
      System.out.println("line #" +i+ ": \""+ WebPageBuffer.readLine() +"\"");
    }
  }
}
```

Aufgabe 6.3

Die Fakultät $n!$ ist eine Funktion, die jeder natürlichen Zahl n das Produkt aller natürlichen Zahlen kleiner und gleich dieser Zahl zuordnet:

$$\begin{aligned} n! &= \prod_{k=1}^{n} k \\ &= 1 \cdot 2 \cdot \ldots \cdot n \end{aligned}$$

Zusätzlich gilt $0! = 1$.

Schreiben Sie ein Java-Programm, das den Benutzer zur Eingabe einer ganzen Zahl n auffordert und anschließend den Wert von $n!$ ausgibt. Für eine negative ganze Zahl soll der Wert -1 ausgegeben werden.

Aufgabe 6.4

Schreiben Sie eine Klasse `Measurements` zum Auswerten von Messdaten. Gehen Sie davon aus, dass Ihr Datenfile in jeder Zeile eine `Double`-Zahl enthält. Benutzen Sie Iteratoren, um die folgenden Methoden zu implementieren.

1. Implementieren Sie eine Methode, die einen Dateinamen als Argument erhält und die in der Datei enthaltenen `Double`-Zahlen in einer `ArrayList`-Instanzvariable speichert.

2. Implementieren Sie die Methode `double sum()`, die die Summe aller Zahlen in dem Array berechnet.

3. Implementieren Sie die Methoden `double min()` und `double max()`, die das Minimum bzw. das Maximum der Zahlen zurückgeben.

4. Implementieren Sie die Methode `double mean()`, die den Durchschnitt der Zahlen berechnet.

5. Implementieren Sie die Methode `ArrayList normalize()`, die eine ArrayList zurückgibt, die aus den normalisierten Messungen besteht. Normalisieren bedeutet, dass die Zahlen durch die Summe aller Zahlen geteilt werden. Nach der Normalisierung summieren sich alle Werte zu 1.

6. Schreiben Sie eine Methode, die eine Liste von Zahlen einliest und die eben implementierten Methoden ausführt. Verwenden Sie dafür die Datei `data.dat` auf der Vorlesungshomepage.

Codestyle - Konventionen

Ihre Programme sollten folgende Konventionen einhalten:

1. Variablen- und Methodennamen: $[a-z][a-zA-Z0-9\_]^*$
   (d.h. erstes Zeichen Kleinbuchstabe, folgende Zeichen beliebige Buchstaben oder Unterstriche). Die Bezeichnung der Variablen bzw. Methoden sollte möglichst klar ihre Bedeutung im Programm beschreiben.

2. Klassennamen: $[A-Z][a-zA-Z0-9\_]^*$
   (d.h. erstes Zeichen Großbuchstabe, folgende Zeichen beliebige Buchstaben oder Unterstriche).

3. Leerzeichen nach ",".

4. Leerzeichen um zweistellige Operatoren, wie z.B. "+","−","<" oder "=".

5. If-Blöcke in der Form:

   ```
   if (i < j) {
     System.out.println("i < j");
   } else {
     System.out.println("j <= i");
   }
   ```

   mit Leerzeichen nach `if` und `else` sowie Leerzeichen vor geschweiften Klammern.

6. For-Schleifen in der Form:

   ```
   for (int i = 0; i < 10; ++i) {
     System.out.println("i");
   }
   ```

   mit Leerzeichen nach `for` sowie Leerzeichen vor geschweiften Klammern.

7. While-Schleifen in der Form:

   ```
   while (i < 10) {
     System.out.println("i");
     ++i;
   }
   ```

   mit Leerzeichen nach `while` sowie Leerzeichen vor geschweiften Klammern.