Große Übung
Praktische Informatik 1

2005-11-17

Holger Füßler
fuessler@informatik.uni-mannheim.de
http://www.informatik.uni-mannheim.de/pi4/people/fuessler

1: Announcements / Orga

Vorl. Leistungsbewertung

- Blatt 1: 84,2 %
- Blatt 2: 60,3 %
- insgesamt 231 “aktive” Studenten von 293
- 172 (ungef. 75%) davon > 50%

- Java ist in der Vorlesung fast durch!
- Ab jetzt ist Programmieren Üben und Stoffbegleiten

2: Relevantes in Java

Scanner

- java.util.Scanner
- http://java.sun.com/j2se/1.**5.0**/docs/api/java/util/Scanner.html
- in java 5.0 eingeführte Klasse, um aus einem Eingabestrom Tokens unterschiedlicher Typen herauszulesen

java.util

Class Scanner

```
java.lang.Object
  └ java.util.Scanner
```

All Implemented Interfaces:

Iterator<String>

```
public final class Scanner
extends Object
implements Iterator<String>
```

A simple text scanner which can parse primitive types and strings using regular expressions.

A `Scanner` breaks its input into tokens using a delimiter pattern, which by default matches whitespace. The resulting tokens may then be converted into values of different types using the various `next` methods.

For example, this code allows a user to read a number from `System.in`:

```
Scanner sc = new Scanner(System.in);
int i = sc.nextInt();
```

As another example, this code allows `long` types to be assigned from entries in a file `myNumbers`:

Scanner Example

```
import java.util.Scanner;
...
Scanner sc = new Scanner(System.in);
int i = sc.nextInt();                  // get next Integer
long l = sc.nextDouble();              // get next Double
String s = sc.next();                  // get next Token as String
...
```

```
# Simple File IO

java ScannerEx < infile.txt > outfile.txt

# feeds infile.txt to System.in and writes outfile.txt to System.out
```

Getting Numbers from Strings

- jeder primitive Datentyp hat eine sogenannte "Hüllklasse" oder "wrapper class"
- nützlich für
  - Konversionen
  - wichtige Konstanten
  - manchmal braucht man eine Hülle um einen primitiven Datentyp, z.B. bei Verwendung in Listen (nicht wichtig hier)

Wrapper Classes

primitiver Typ	wrapper class
char	java.lang.Character
byte	java.lang.Byte
short	java.lang.Short
int	java.lang.Integer
char	java.lang.Character
boolean	java.lang.Boolean
float	java.lang.Float
double	java.lang.Double

Konversion Methoden 1 + 2

```
String dString = “4,543”;
String bString = “true”;

Double d = Double.valueOf(dString); // .valueOf() Methode
System.out.printf(“%g\n”, d.doubleValue());

Boolean b = new Boolean(bString); // Konstruktor Methode
System.out.printf(“%b\n”, b.boolValue());

/* Diese Methoden funktionieren für alle Zahlentypen und Bool */
```

Konversion Methode 3

```
import java.util.Scanner;

String s = "3,52 hallo true";

Scanner sc = new Scanner(s);
double d   = sc.nextDouble()
String t   = sc.next();
boolean b  = sc.nextBoolean();

System.out.printf("%g / %s / %b \n", d, t, b);
```

Spielen Sie mit der Typumwandlung.
Schreiben Sie ein Programm pro Methode, dass doubles und booleans von der Eingabe einliest und gerundet, bzw. als ja/nein wieder ausgibt. Testen Sie mit Dateien.

3: Vorlesung

Warum Objektorientierung?

- früher (in C) wurden immer Funktionen auf Daten ausgeführt, dadurch können die Funktionen wenig Annahmen über die Daten machen.
- mit Objektorientierung werden Daten mit den Methoden verheiratet. Die Manipulation erfolgt über die Methodenschnittstelle.
- großer Vorteil ist die Möglichkeit, den Zustand der Objekte genau zu kontrollieren
- und und und

Beispiel Zeichenkette

- in C ist eine Zeichenkette ein char[] Vektor, bei dem nach dem letzten Zeichen ein ‘\0’ steht
- Ausgabe läuft also zeichenweise, bis ’\0’ erreicht wird
- was passiert?, wenn da kein ‘\0’ drinsteht? → Fehler

- In Java ist der Inhalt des Strings nicht direkt manipulierbar. So ein Fehler kann nicht auftreten.
- Weiterer Vorteil: Man findet die meisten Manipulationen, die mit Strings möglich sind, auf der entsprechenden Doku-Seite.

Beispiel Konto

- bei einem Programm, die mehrere Konten verwaltet, müsste man in C darauf achten, dass einer Abbuchung eine Gegenbuchung gegenüber steht.
- in OO kann man den Objektzustand kapseln und eine umbuchen() Funktion anbieten

Wichtig für die Übung

- OO-Konzepte verstehen, wenn wir sie sehen (z.B. Doku)

- ABER: im Wesentlichen bleiben wir bei der prozeduralen (static) Programmierung
- OO-Analyse und Design ist eine Kunst für sich (siehe Prof. Schader) - Ehrlich!
- Wir müssen nicht OO einsetzen können, um softwaretechnisch vollendete Programme schreiben zu können.

4: Übungsblatt 2/A2

Blatt 2 / A2a

Implementieren Sie folgende Aufgabestellungen, sofern möglich, mit dem switch-Konstrukt aus Java. An Stellen, an denen dies nicht möglich ist, weichen Sie auf ein Ihnen bekanntes allgemeineres Konstrukt aus.

Aufgabe 2 a) 4 Punkte
Implementieren Sie eine Methode, welche als Parameter eine Ziffer als Zahl (int) übergeben bekommt und diese dann alsWort (z.B.:”eins“) auf dem Bildschirm ausgibt.

Wird eine Zahl übergeben, die aus mehr als einer Ziffer besteht, soll eine Fehlermeldung ausgegeben werden.

Lösungsansatz

- Klasse und Methode deklarieren
- Kann man mit einem switch nach einer Integer-Zahl verzweigen?
- JA
- Kontrollfluss überlegen (Editor)
- main-Methode zum Testen schreiben

Blatt 2 / A2b

Implementieren Sie eine Methode, welche als Parameter eine Ziffer als Wort übergeben bekommt (z.B.: ”eins“) und diese dann als Zahl als Rückgabewert (int) der Methode zurückgibt. Wird ein ungültiges Wort übergeben, so soll –1 zurückgegeben werden.

- switch() ist hier nicht möglich (kein Integer), deshalb if

Blatt 2 / A2c

- Ich würde das immer gleich am Anfang mit machen, um Testen zu können
- Sinnvolles Testen
  - Wo kann es Fehler geben?
    - Tippfehler in den gemappten Strings
    - was passiert außerhalb des “sinnvollen” Wertebereichs der Parameter?
  - ALSO: mein Ansatz hier mit Schleife
    - Vorteile: Flexibilität, Vollständigkeit (Achtung, sollte noch übersichtlich sein)

5: Übungsblatt 4

Hinweise A1

- Programmaufruf mit Anzahl
- Methode Würfeln
  - Parameter Anzahl
- Methode ausgeben
  - Parameter Würfelverteilung

Hinweise A2

- keine Programmieraufgabe
- Begründung nicht vergessen, z.B. 1 → 39, da der lokale Parameter die Instanzvariable überdeckt.

Hinweis A3

- keine Programmieraufgabe
- a) Erklärung nicht vergessen, z.B. Fehler in Zeile 99, weil Instanzvariable p nicht zugreifbar
- b) Typkonversionen nicht vergessen...

Hinweise A4

- Die Doku zu Math.sin() und Math.cos() und Math.PI lesen
- Rest ist straightforward

Ende