Einfache Rechenstrukturen und Kontrollfluß I

Martin Wirsing

in Zusammenarbeit mit
Matthias Hölzl, Piotr Kosiuczenko, Dirk Pattinson

10/04/03

Allgemeines

- Bitte zur Übung anmelden bis Freitag, 15 Uhr, auf
  http://www.pst.informatik.uni-muenchen.de/lehre/SS03/infoII/
  unter Anmeldung

Literaturhinweise

K. Arnold, J. Gosling. *The Java Programming Language.* Addison-Wesley, 1996.
David J. Barnes, Michael Kölling. *Objects First With Java, A Practical Introduction Using BlueJ.* Harlow: Pearson Education, 2003.
...
C. Horstmann. *Computing Concepts with Java Essentials.* 3rd Edition, Wiley, 2003.
D. Lea. *Concurrent Programming in Java.* Addison-Wesley, 1997.
...
Johannes Link. *Unit Tests mit Java.* Heidelberg, dpunkt.Verlag, 2002
...

Ziele

- Verstehen der Grunddatentypen von Java
- Verstehen von Typkonversion in Java
- Lernen lokale Variablen und Konstanten zu initialisieren
- Verstehen der Speicherorganisation von lokalen Variablen
- Wiederholen der Regeln des Hoare-Kalkül

Grunddatentypen in Java

- Ganze Zahlen
- Gleitpunktzahlen
- Zeichen
- Boole'sche Werte
- und Felder (später)

Ganze Zahlen

- byte 1 Byte -128 bis 127
  -2^7 bis 2^7-1
- short 2 Byte -32768 bis 32767
- int 4 Byte -2,147,483,648 bis 2,147,483,647
  -2^{31} bis $2^{31}-1$
- long 8 Byte
  -9,223,372,036,854,775,808 bis 9,223,372,036,854,775,807

Gleitpunktzahlen

- float 4 Byte bis ca. 10^{32}
- double 8 Byte bis ca. 10^{308}
  nach IEEE-754--Standard (1985)

Beispiele:

- double: 6.22, 622E-2 , 62.2e-1
- float: 6.22F, 622E-2F, 62.2e-1f

Arithmet. Operationen und Vergleichsoperationen

* Multiplikation, / Division, % Modulo (Rest)

\+ Addition, - Subtraktion

> größer, >= größer oder gleich
< kleiner, <= kleiner oder gleich

== gleich, != nicht gleich (Achtung: „!" fehlt auf Ausdruck)

(= Zuweisung wird als Gleichheit geschrieben)

Typkonversion

„Kleiner-Beziehung“: zwischen Datentypen

`byte` < `short` < `int` < `long` < `float` < `double`

Beispiele:

1 + 1.7	ist vom Typ **`double`**
1.0f + 1	ist vom Typ **`float`**
1.0f + 1.0	ist vom Typ **`double`**

Typkonversion

Type Casting:

Erzwingen der Typkonversion durch Voranstellen von „(**`type`**)“

Beispiele:

(**`byte`**) 3	ist vom Typ **`byte`**
(**`int`**) (2.0 + 5.0)	ist vom Typ **`int`**
(**`float`**) 1.3e-7	ist vom Typ **`float`**

Bei der Typkonversion kann Information verloren gehen.

Beispiel:

(**`int`**) 5.2 == 5
(**`int`**) –5.2 == -5

Zeichen

- Typ **`char`** (für character)
- bezeichnet Menge der Zeichen aus dem Unicode-Zeichensatz
- **`char`** umfaßt ASCII-Zeichensatz mit kleinen und großen

  Buchstaben, Zahlen und verschiedenen Sonderzeichen
- Darstellung von Zeichen durch Umrahmung mit Apostroph

  Beispiel: `‘a‘ , ‘A‘ , ‘1‘ , ‘9‘`
- Zeichenketten: werden mit Doppelapostroph umrahmt und sind

  vom Typ `String` (eine Klasse): „Wirsing“, „Info2“

Boole‘sche Werte

Der Typ **`boolean`** hat genau zwei Werte, **`true`** und **`false`**.

Boole‘sche Operatoren

!	strikte Negation	(Achtung: „!“ fehlt auf Ausdruck)
&	strikte Konjunktion („und“, auch bitweise Addition)	
^	strikte Disjunktion („entweder-oder“)	
|	strikte Adjunktion („oder“)	
&&	sequentielle Konjunktion	(**`andalso`** in SML)
||	sequentielle Adjunktion	(**`orelse`** in SML)

Boole'sche Werte

„entweder-oder“ **und** **„oder“**

^	true	false
true	false	true
false	true	false

|,||	true	false
true	true	true
false	true	false

Boole'sche Werte

Beispiel für die strikte/sequentielle Konjuktion

```
int teiler = 0;
(teiler != 0)  &&  (100/teiler > 1) == false  // Ok
(teiler != 0)  &  (100/teiler > 1)  == false
                                // Laufzeitfehler
```

Beispiel für die strikte/sequentielle Adjunktion

```
true || (1/0 == 1) == true;          // Ok
true |  (1/0 == 1)                   // Laufzeitfehler
```


Korrespondenz SML - Java

	Java	SML
Gleitpunktzahlen	`float, double`	`real`
unäres Minus	`-`	`~`
Division	`/`	`/`
Modulo (Rest)	`%`	nicht vorhanden
Konversion nach ganze Zahl	`(int)`	`truncate`
Ganze Zahlen	`int`	`int`
Ganzzahldiv.	`/`	`div`
Modulo (Rest)	`%`	`mod`
Boole'sche Werte	`boolean`	`bool`
strikte Konj.	`&`	nicht vorhanden
sequ. Konj.	`&&`	`andalso`
strikte Adj.	`|`	nicht vorhanden
sequ. Adj.	`||`	`orelse`
strikte Disj.	`^`	nicht vorhanden
Worte	`String`	`string`
Konkatenation	`+`	`^`

Deklaration lokaler Variablen und Konstanten

Eine einfache Deklaration lokaler Variablen hat die Form

```
<Type>  <VarName> = <Expression>;
```

//Deklaration mit Initialisierung

Beispiel:

```
int total = -17;
int quadrat = total * total;
boolean aussage = false;
```

Deklaration lokaler Variablen und Konstanten

Andere Möglichkeit: Deklaration ohne Initialisierung der Form

```
<Type>  <VarName>;
```

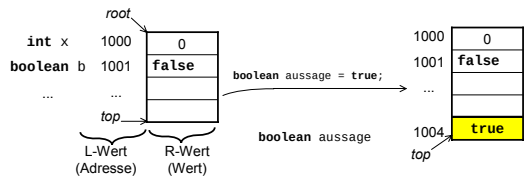
Beispiel:

```
int total;
int quadrat;
boolean aussage;
```

Ist nur erlaubt, wenn zur Übersetzungszeit nachgewiesen werden kann, dass die Variable initialisiert wird bevor sie benutzt wird

Deklaration lokaler Variablen

Lokale Variablen werden im „Keller" (Stack) gespeichert. Durch die Deklaration wird eine neue Speicherzelle für die lokale Variable reserviert und mit ihrem Initialwert belegt.



Iterierte Deklaration lokaler Variablen

Beispiel:

```
int total = 17, max = 100, i, j;
```

ist eine Abkürzung für

```
int total = 17;
int max = 100;
int i;
int j;
```


Deklaration lokaler Konstanten

- Eine Konstante wird durch Angabe des „Modifiers" **`final`** deklariert.
  Beispiel: `final int TOTAL = 100;`
- Konstanten werden i.a. mit Großbuchstaben geschrieben
- Konstanten sollten (wie auch Variablen) „sprechende" Namen besitzen
- Nie „Magic Numbers" verwenden

  Beispiele:

- Anstelle von 365 im Programm für „Anzahl der Tage im Jahr" verwende man besser `final int TAGE_PRO-JAHR = 365;`
- Für die mathematischen Größen π und e verwende man anstelle von 3.14159 und 2.7182 besser `Math.Pi` bzw. `Math.E`

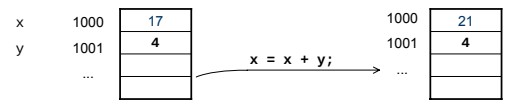
Zuweisung

Eine **Zuweisung** hat die Form

```
<VarName> = <Expression>;
```

(in Java mit „**=**“ geschrieben im Gegensatz zu Pascal, Modula etc., wo „**:=**“ verwendet wird).

Beispiel:



Zuweisung: Abkürzende Schreibweisen

Abkürzungen

`x++;`	steht für `x = x + 1;`
`x--;`	steht für `x = x - 1;`
`x op= <Ausdruck>;`	steht für `x = x op <Ausdruck>`

Beispiele

`x += y;`	steht für `x = x + y;`
`x &&= c;`	steht für `b = b && c;`
`x += 3*y;`	steht für `x = x + 3*y;`

Zuweisung: Hoare-Regeln (Wiederholung)

Zuweisungsaxiom

```
{P[exp/x]} x = exp; {P}
```

(Gilt auch für Deklaration lokaler Variablen)

Abschwächungsregel

$$\frac{P1 \Rightarrow P,\ \{P\}\ S\ \{Q\},\ Q \Rightarrow Q1}{\{P1\}\ S\ \{Q1\}}$$

Zuweisung: Hoare-Regeln (Wiederholung)

Beispiel:

- `{max – 10 > 50} max = max – 10; {max > 50}`
- `{max – C == 35} max = max – C; {max == 35}`
- Aus `max == 100` folgt

  wegen `max == 100` $\Rightarrow$ `max – 10 > 50`

  mit der Abschwächungsregel

  `{max == 100} max = max – 10; {max > 50}`

Zusammenfassung

- Java besitzt
  - 4 Grunddatentypen für ganze Zahlen (**byte, short, int, long**) und
  - 2 Grunddatentypen für Gleitpunktzahlen (**float, double**).
- Dazu kommen noch **boolean** und **char**.
- String ist **kein** Grunddatentyp.
- Java hat eine **automatische Konversion in den "größeren"** Grunddatentyp.
- Konversion in einen **"`kleineren" Datentyp** geschieht explizit durch **Typcasting**.

Zusammenfassung (2)

- Grundlegende **imperative Konstrukte** von Java sind:
  - Deklaration lokaler Variablen, Zuweisung, Sequ. Komposition
- Eine lokale Variable besitzt einen **L-Wert** (Adresse) und einen **R-Wert**. Lokale Variablen werden im **Keller** gespeichert.
- Lokale Variablen müssen **vor Benutzung initialisiert** werden.
- Hoare-Regeln dienen zum Beweis der partiellen u. totalen Korrektheit.