Elementare Konzepte von Programmiersprachen

Teil 1:

Bezeichner, Elementare Datentypen, Variablen, Referenzen, Zuweisungen, Ausdrücke

Kapitel 6.3 bis 6.7 in Küchlin/Weber: Einführung in die Informatik

Bezeichner

- **Bezeichner** (*identifier*) in Programmiersprachen für viele Dinge notwendig
  - Sprachkonstrukte, wie while
  - Arithmetische Operatoren, wie +
  - Elemente elementarer (eingebauter) Datentypen, wie Zahlen
  - In der Programmiersprache definierbares, wie Variablen, Klassen, Funktionen, Methoden

Bezeichner

- **Alphabet**
  - geordnete Zeichenmenge (Zeichensatz)
  - i.a. bestehend aus Buchstaben, Zahlen, Sonderzeichen
  - Oft **ASCII**-Zeichensatz
  - In Java: **UNICODE**

Bezeichner

- Bezeichner sind **Wörtern** über **Alphabet**
  - Zeichenreihen, die gewissen Regeln entsprechen
  - Oftmals festgelegt als reguläre Ausdrücke (Begriff aus der theoretischen Informatik)
- **Syntax** ist die korrekte Art und Weise, sprachliche Elemente zusammen zu fügen und anzuordnen
  - Syntax von Programmen wird vom Übersetzer geprüft

Bezeichner

- **Schlüsselwörter** sind fest vordefinierte Bezeichner für Elemente der Sprache.
  - Dürfen nicht als Bezeichner für selbst definierbares benutzt werden
- **Operatoren** meist durch Kombination von Sonderzeichen
  - Etwa <=, ++, --, ==
- **Literale** bezeichnen Elemente eingebauter Datentypen
  - 123 für eine ganze Zahl
  - 12.3 für Gleitkommazahl,
  - `z` für ein Zeichen

Schlüsselwörter

- **Schlüsselwörter in Java**

abstract	double	interface	switch
assert	else	long	synchronized
boolean	extends	native	this
break	final	new	throw
byte	finally	package	throws
case	float	private	transient
catch	for	protected	try
char	goto	public	void
class	if	return	volatile
const	implements	short	while
continue	import	static	
default	instanceof	strictfp	
do	int	super	

goto und const momentan nicht verwendet;
assert und strictfp seit Java 2;
zusätzlich reservierte Wörter (Literale) true, false, null

Namenskonventionen

- Neben in Sprachdefinition fest vorgegebenen Regeln (die vom Compiler überprüft werden) gibt es auch **Namenskonventionen**
  - Weichen in verschiedenen Sprachen voneinander ab
  - Einhalten der Namenskonventionen **sehr sinnvoll!**
    - Keine „gesetzliche Vorschrift“, aber „guter Sprachgebrauch“
    - Erleichtert das Lesen von Programmen

Namenskonventionen

- In Java:
  - Namen beginnen mit Buchstaben
  - Namen für **Klassen groß** geschrieben
  - Namen für **Funktionen (Methoden)** und **Variablen klein** geschrieben
  - Bei zusammengesetzten Namen schreibt man angefügte Teile jeweils wieder groß
  - Sonderzeichen:
    - Alle UNICODE Buchstaben erlaubt, z.B. Umlaute
    - Kann missbraucht werden: verschiedene Buchstaben mit gleichem Druckbild

Namenskonventionen

- Beispiele
  - Gerät ist ein Klassenname
  - z ein Variablenname
  - getZ der Name eines Selektors für das Feld z
  - Weitere Methodennamen:
    - wähleKanal
    - bewegePunktZuPunk

Elementare Datentypen

- **Elementare Datentypen in Java**

`byte`	8-bit-Zahl in Zweierkomplement-Darstellung
`short`	16-bit-Zahl in Zweierkomplement-Darstellung
`int`	32-bit-Zahl in Zweierkomplement-Darstellung
`long`	64-bit-Zahl in Zweierkomplement-Darstellung
`float`	32-bit IEEE 754-1985 Gleitkommazahl
`double`	64-bit IEEE 754-1985 Gleitkommazahl
`char`	16-bit Unicode 2.0 Zeichen
`boolean`	Wahrheitswert, `true` oder `false`

Literaldarstellungen

- Literaldarstellung gibt es in Java für
  - boolean
  - char
  - int
  - long
  - float
  - double
  - Sonderfall: null-Literal

Literaldarstellungen

- **Ziffernfolgen** (evtl. mit Vorzeichen)
  - bezeichnen ein **int**
  - Beispiele: 23 -3 +53
  - werden im **Dezimalsystem** interpretiert, es sei denn, sie beginnen mit 0 oder 0x
  - Mit 0 beginnende Ziffernfolgen werden als **Oktalzahlen** interpretiert
    Beispiel: 027

Literaldarstellungen

- Mit 0x beginnende Ziffernfolgen werden als **Hexadezimalzahlen** interpretiert
  - a, b, c, d, e, f als Ziffern für 10,...,15
  - Beispiele: 027 0x17 0x1b 0xffff
- Nachgestelltes l oder L führt zu interpretation als **long**
  Beispiele: 23l -3L +53L 027L

Literaldarstellungen

- Ziffernfolgen mit **Dezimalpunkt** werden als **Fließkommazahl** (float) interpretiert
- Ziffern nach e werden als **Dezimalexponent** interpretiert
- Nachgestelltes d oder f führt zu Interpretation als **double** bzw. **float**
- Beispiele:
  - 2. -3.4 53e4
  - 0d 2d -3.4d 53e4d
  - 2f -3.4f 53e4f

Elementare Datentypen

- In Java stellt man Zeichenliteral dar durch
  - Zeichen in Hochkommata
    - Falls die Tastatur es erlaubt
    - Beispiel: `char c = ´A´;`
  - Zeichennummer in UNICODE als Hexadezimalzahl
    - Beispiel: `char c = ´\u0041´;`

Elementare Datentypen

- **Sonderliterale** (Escape-Sequenzen)

Escape-Sequenz	Unicode	Zeichen
\b	\u0008	*backspace*, BS
\t	\u0009	Tabulator, *horizontal tab*, HT
\n	\u000a	Zeilenvorschub, *linefeed*, *newline*, LF
\f	\u000c	Seitenvorschub, *form feed*, FF
\r	\u000d	Wagenrücklauf, *carriage return*, CR
\"	\u0022	Anführungszeichen, *double quote*, "
\'	\u0027	Hochkomma, *single quote*, '
\\	\u005c	*backslash*, \

Hüllklassen für elementare Datentypen

- Für alle elementaren Datentypen gibt es **Hüllklassen**
  - Elementare Datentypen beginnen alle mit Kleinbuchstaben, entsprechende Hüllklasse mit Großbuchstaben
    - Byte, Short, Integer, Long, Float, Double, Character, Boolean
  - Bündeln Konstanten und Methoden, die im Zusammenhang mit den elementaren Typen nützlich sind