Einführung in die Informatik

Jochen Hoenicke

Software Engineering
Albert-Ludwigs-University Freiburg

Sommersemester 2014

Teil I

Organisation

Organisation

- Die Vorlesung ist Mittwochs 14–16 (c.t).
- Die Übungen sind Dienstag 16–18, Mittwoch 16–18, Freitags 12–14
- Die Übungen starten in der nächsten Woche

Anmeldung zu den Übungen

- Übungen können in Gruppen von zwei Personen bearbeitet werden.
- Bitte Gruppe in der Liste eintragen.
- Außerdem muss sich jeder mit seinem Rechenzentrumszugang auf der Daphne-Seite anmelden: `https://daphne.informatik.uni-freiburg.de/ss2014/EinfuehrungInformatikSS2014/`

Die Klausur findet am Freitag, den 5. September, 14 Uhr statt.

Übungsgruppen

In den Übungsgruppen

- sollen Fragen zur Vorlesung geklärt werden,
- Übungsaufgaben/Musterlösung besprochen werden,
- Fragen zum neuen Übungsblatt gestellt werden,
- Probleme mit den Aufgaben besprochen und gelöst werden.

Eine Sprache lernt man nicht durch Theorie; man muss sie benutzen.

Übungen sind nicht verpflichtend, aber ohne sie kann man nicht Programmieren lernen.

Daphne und Subversion

Die Übungen werden über Daphne verwaltet.

- Abgabe geschieht elektronisch über Subversion.
- Programme werden durch Jenkins auf Syntaxfehler geprüft.
- Abgabezeitpunkt wird automatisch überwacht.

Empfohlene Literatur

Die Vorlesung folgt dem Buch:

Dietmar Ratz, Jens Scheffler, Detlef Seese, Jan Wiesenberger, Grundkurs Programmieren in Java, Hanser Verlag, 2011.

`http://www.grundkurs-java.de/`

Teil II

Unser erstes Programm

Unser erstes Programm

Unser erstes Java-Programm soll $3 + 4$ ausrechnen.

```
public class Berechnung {

    public static void main(String[] param) {

        int i;

        i = 3 + 4;

        System.out.println(i);

    }
}
```

Schreiben, Kompilieren, Ausführen

- Das Programm kann mit einem Editor (z.B. notepad++, nano, vim, emacs, eclipse, idea) geschrieben werden.
- Es muss unter dem Namen `Berechnung.java` gespeichert werden.
- Anschließend kann man auf der Konsole (unter Windows startet man dazu cmd) das Programm kompilieren und ausführen:

```
> cd Verzeichnis
> javac Berechnung.java
> java Berechnung
7
```

Achtung

JDK muss installiert sein und der Pfad gesetzt sein (siehe Forum-Link).

Rahmen

Unser eigentliches Programm wird in einen Rahmen gepackt:

```
public class Berechnung {
    public static void main(String[] param) {
        ...
    }
}
```

- Die erste Zeile enthält den Namen des Programms (Berechnung).

Achtung

Der Name muss mit dem Namen der .java-Datei übereinstimmen.

- Die zweite Zeile ist notwendig, damit das Programm ausführbar ist.
- Die genaue Bedeutung dieser Zeilen wird später erklärt. Für das erste, können Sie die Zeilen einfach übernehmen.

Variablen

Mit der Zeile

```
int i;
```

wird ein Speicherbereich für eine Ganzzahl-Variable mit dem Namen `i` reserviert.

- Variablen dienen zum Speichern von Zwischenergebnissen.
- Eine Variable kann unterschiedliche Typen haben: ganze Zahlen (`int`), Kommazahlen (`double`), Zeichenketten (`String`), usw.
- `int` steht für "integral number" (Ganzzahl)
- Die Grösse ist begrenzt: `int` kann nur Zahlen zwischen -2147483648 und 2147483647 speichern.
  Statt `int` kann man auch `long` schreiben für Zahlen zwischen -9223372036854775808 und 9223372036854775807.

Zuweisungen

Mit der Zeile

```
i = 3 + 4;
```

wird $3 + 4$ „berechnet“ und in der Variablen `i` gespeichert.

Achtung

Das Gleichheitszeichen ist in Java nicht symmetrisch. Die Zuweisung

```
3 + 4 = i;
```

führt zu einem Syntaxfehler.
Links muss eine Variable stehen (genauer: ein Name für ein Speicherfeld), rechts muss eine Berechnung stehen.

Ausgabe

Mit der Zeile

```
    System.out.println(i);
```

wird das Ergebnis auf die Konsole ausgegeben.
Man kann auch Zeichenketten ausgeben:

```
    System.out.print("Das Ergebnis ist: ");
    System.out.println(i);
```

oder

```
    System.out.println("Das Ergebnis ist: " + i);
```

gibt die Zeile

```
Das Ergebnis ist: 7
```

aus.

Teil III

Java Syntax

Lexikographische Struktur

Lexikographische Struktur

Ein Java-Programm besteht aus vielen Zeichen.
Mehrere Zeichen bilden zusammen eine Einheit, ähnlich einem Wort in einer natürlichen Sprache. Wir unterscheiden:

- Bezeichner: z.B. `i`, `main`, `param`.
  Damit werden Variablen, Klassen, Methoden, Felder bezeichnet.
- Literale: `3`, `3.14`, `'a'`, `"Hallo Welt"`, `true`, `null`
- Reservierte Wörter: `class`, `int`, `public`, `static`, `void`
- Sonderzeichen und Operatoren: `{`, `}`, `[`, `]`, `(`, `)`, `+`, `=`
  Es gibt auch zusammengesetzte Operatoren: `!=`, `<=`, `++`, `>>>=`
- Kommentare und Leerräume
  Diese haben keine Bedeutung für den Computer.

Bezeichner

Syntax

$$\begin{aligned} \textit{Bezeichner} &::= \textit{JavaLetter}\,\textit{JavaLetterOrDigit}^* \\ \textit{JavaLetter} &::= \texttt{A} \mid \ldots \mid \texttt{Z} \mid \texttt{a} \mid \ldots \mid \texttt{z} \mid \texttt{\_} \mid \texttt{\$} \mid \text{any Unicode letter} \\ \textit{JavaLetterOrDigit} &::= \textit{JavaLetter} \mid 0 \mid \ldots \mid 9 \mid \text{any Unicode digit} \end{aligned}$$

Bezeichner bestehen auf Buchstaben, Ziffern, Unterstrich oder Dollar und dürfen nicht mit einer Ziffer beginnen. Z.B.

- `i`, `Uebung01`, `Uebung_01`
- `Übung01` (gefährlich, da Kodierung kaputt gehen kann)
- `\u00dbbung01` (sichere Variante, aber nicht lesbar)
- `Uebung$01` (aber Dollarzeichen sind für interne Zwecke reserviert)

Keine Bezeichner sind:

- `1Uebung`, `Uebung 1`, `Uebung-1`
- `class`, `int` (reservierte Wörter)

Literale

Literale sind Konstanten wie ganze Zahlen, Zeichen, Zeichenketten

- 0, 123, -3421 (ganze Zahlen)
- `3.14159`, `6.62606957e-34` (Fließkommazahlen)
- `'a'`, `'@'` (Zeichen)
- `"Das Ergebnis ist:"` (Zeichenketten)
- `true`, `false` (Boolesche Konstanten)
- `null` (Null-Objekt)

Mit \ man Zeichen wie ", \ und ' in Zeichenketten ausdrücken:

- `'\''`
- `'\\'`
- `"Ich sagte: \"Hallo\""`

Reservierte Wörter

Folgende Schlüsselwörter sind in Java reserviert:

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

Wenn der Compiler für Ihr Programm merkwürdige Fehlermeldungen liefert, überprüfen Sie, ob Sie eine Variable nach einem Schlüsselwort benannt haben.
Die Schlüsselwörter `const` und `goto` sind zwar reserviert, aber nicht in Java eingebaut.

Kommentare

Es gibt drei Arten von Kommentaren:

- Mit `//` wird der Rest der Zeile als Kommentar interpretiert

```
// ein einzeiliger Kommentar
```

- Alles zwischen `/*` und `*/` wird als Kommentar gelesen. Kommentare können nicht geschachtelt werden.

```
/* Dies ist ein Kommentar.
 * Er kann ueber mehrere Zeilen gehen */
/* /* // */ dies ist kein Kommentar
// dies aber */ und dies
```

- Alles zwischen `/**` und `*/` wird als `javadoc`-Kommentar gelesen.

```
/** Berechnung der Formel 3 + 4. */
public class Berechnung {
  /** Die Hauptmethode zur Berechnung von 3+4.  Sie wird
   *  beim Starten automatisch aufgerufen.
   *  @param params Kommandozeilenargumente (ignoriert).
   */
  public static void main(String[] params) {
```

Grundstruktur eines Programms

Klassen

Die kleinste Einheit eines Java-Programms ist eine Klasse.

```
public class Berechnung {
    ...
}
```

Das Schlüsselwort `public` besagt, dass die Klasse von fremden Programmteilen benutzt werden darf.
Eine als `public` definierte Klasse muss in einer Datei mit dem gleichen Namen und der Endung `.java` gespeichert werden.
Der Kompiler (`javac`) schreibt das Kompilat in eine Datei mit der Endung `.class`. Diese kann dann vom Interpreter (`java`) ausgeführt werden.

Package

Ein größeres Projekt wird gewöhnlich in Paketen organisiert. Es gibt auch Unterpakete, die mit Punkt getrennt werden.

```
package java.util;

public class ...
```

Jedes Paket bekommt sein eigenes Unterverzeichnis, das genau so heißen muss, wie das Paket.

Coding Conventions

Paketnamen sollen nur aus Kleinbuchstaben bestehen. Der Paketname soll mit dem umgekehrten Domainnamen beginnen, damit er weltweit eindeutig ist. Zum Beispiel:

```
package de.uni_freiburg.informatik.ultimate.smtinterpol;
```

Wenn das Programm nicht zur Veröffentlichung gedacht ist, reicht es aber lokal eindeutige Namen zu wählen.

Import

Klassen aus fremden Pakete können in eigenen Programmteilen benutzt werden, indem sie importiert werden. Man kann eine Klasse importieren, oder alle Klassen eines Pakets.

```
package ...;
import Prog1Tools.*;         // importiert alle Klassen
import Prog1Tools.IOTools; // importiert nur IOTools
public class ...
```

Es werden immer alle Klassen aus dem Paket `java.lang` inkludiert, z.B. `System`.

Main-Methode

Programmanweisungen stehen für gewöhnlich in Methoden (auch Prozeduren oder Funktionen genannt). Beim Starten eines Programms wird die `main`-Methode aufgerufen:

```
public class Berechnung {
    public static void main(String[] param) {
        // ...Anweisungen...
    }
}
```

Die Schlüsselwörter `static void` und das Argument `String[] param` werden wir später erklären.

Coding-Conventions

Programme sollen ordentlich eingerückt und umgebrochen werden.

```
public class Berechnung{public static void
main(String[]param){int i;i=3+4;System.out
.println(i);}}
```

Das Programm ist schwer zu lesen und zu warten.

Coding-Convention

- Maximal eine Anweisung pro Zeile
- Einrückung von Code in geschweiften Klammern um vier Zeichen
- Leerzeichen wie im Schreibmaschinenkurs
  Ausnahme: Üblicherweise kein Leerzeichen nach Punkt und zwischen Funktionsname und Parameter, z.B. `System.out.println(i);`

JavaDoc

Programme sollten immer mit JavaDoc-Kommentaren versehen werden: ein Kommentar für die Klasse und ein Kommentar für jede Methode.

```
/** Berechnung der Formel 3 + 4. */
public class Berechnung {
  /** Die Hauptmethode zur Berechnung von 3+4.  Sie wird
   *  beim Starten automatisch aufgerufen.
   *  @param params Kommandozeilenargumente (ignoriert).
   */
  public static void main(String[] params) {
```

Das Tool javadoc erzeugt aus diesen Kommentaren eine HTML-Dokumentation, die für andere Benutzer sehr hilfreich sein kann.
Siehe z.B. `http://docs.oracle.com/javase/7/docs/api/`
Die Kommentare müssen eine gewisse Konvention einhalten, z.B. sollte der erste Satz eine kurze und prägnante Beschreibung sein.