Kapitel 04

Wichtige Regeln für die Programmierung in Java

Fachgebiet Knowledge Engineering
Prof. Dr. Johannes Fürnkranz

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Der Sprachaufbau von Java – Abstraktionsebenen

Spezifikatorische Ebene	Übereinstimmung der Ergebnisse von Programm-aufrufen mit den ursprünglich gesetzten Zielen
Logische Ebene	Ergebnisse von Programmaufrufen ohne Fehler auf darunter liegenden Ebenen
Semantische Ebene	Bedeutung von Texten, die lexikalisch und syntaktisch korrekt formuliert sind
Syntaktische Ebene	Gruppierung von lexikalischen Einheiten und trennenden Elementen zu Ausdrücken
Lexikalische Ebene	Unterste Ebene, zusammengesetzt aus lexikalischen Einheiten und trennenden Elementen

Lexikalische Ebene

Unterste Ebene, zusammengesetzt aus lexikalischen Einheiten und trennenden Elementen

z. B. `int i = 7;`

- Schlüsselwörter: Zeichenketten mit besonderer Bedeutung, z. B.
  - `int, if, for, class, return`
  - in Java immer aus Kleinbuchstaben
- Bezeichner: Namen von Variablen, Methoden, Klassen, etc.
  - beliebige Zeichenketten aus Buchstaben, Ziffern und Underscore
  - dürfen nicht mit einer Ziffer anfangen
- Operatoren: Zuweisungen und Rechenzeichen
  - Grundrechenarten `+, -, *, /`
  - Vergleichsoperatoren `<, >, etc.`
  - Logische Operatoren `!, &&, ||`
- Klammerzeichen
  - `( ), { }, [ ]`
- Literale: explizite Werte
  - `4711; 3,14159; 'a'; "Hello"`
- Separatoren: einer oder mehrere sind gleichbedeutend
  - Leerzeichen, Tab, Umbruch

Syntaktische Ebene

Korrekte Gruppierung von lexikalischen Einheiten und trennenden Elementen zu Ausdrücken

Beispiele

- Öffnende und schließende Klammern dürfen nur in Paaren auftreten.
- Die Öffnende steht vor der Schließenden
  - hintereinander `(..)..(..)..(..)`
  - geschachtelt `(..((..)(..)..))`

- Ein Schlüsselwort ist Teil eines Programmkonstrukts, darf nur dort und in festgelegter Reihenfolge auftreten

```
1   int i = 3;
2   while(i < 100)
3   {
4           if(i%2 == 0)
5                   i--;
6           else
7                   i*=2;
8   }
```

Anweisung 1

Anweisung 2

Semantische Ebene

Der Begriff Semantik einer Sprache bezieht sich auf die Bedeutung von Texten, die lexikalisch und syntaktisch korrekt formuliert sind.

Beispiel Bindungsstärke bei Operatoren

- Verschiedene Operatoren haben verschiedene Bindungsstärken, vgl. Punkt vor Strich in der Mathematik
- `int i = 3 - 10/2*4 + 1;`
- Das Ergebnis ist `i = -16;`
- Aber durch Klammerung kann die Bindungsstärke beeinflusst werden
- `int j = 3-(10/(2*(4+1)));`
- Das Ergebnis ist `j = 2;`

Das Ergebnis verändert sich, weil sich die Bearbeitungsreihenfolge geändert hat und damit die Bedeutung des Ausdrucks.

Logische Ebene

Ergebnisse von Programmaufrufen ohne Fehler auf darunter liegenden Ebenen

Was heißt dann noch, dass ein Programm korrekt bzw. nicht korrekt ist?

- Ein Programm ist dann korrekt, wenn es
  - alles das tut, was es soll
  - mit allen erwünschten Effekten
  - und darüber hinaus keine Effekte hat, die es nicht haben soll.

Beispiel: Fencepost Syndrom

- Wie viele Zaunpfähle braucht man, wenn
  - der Zaun 20 Meter lang werden soll
  - alle 2 Meter ein Pfahl stehen muss

Programmierfehler auf der logischen Ebene werden nicht beim Kompilieren mit Fehlermeldungen abgefangen.

Sie können nur durch ausführliches Testen entdeckt und beseitigt werden.

Spezifikatorische Ebene

Auf der spezifikatorischen Ebene geht es um Diskrepanzen zwischen den eigentlichen Intentionen des Programms und seiner Spezifikation

Beispiel: Millenium Bug

- Für ein Datum sind Tag, Monat und Jahr abzuspeichern.
- Der Programmierer verwendet dafür zweistellige Zahlen und erzeugt Daten der Form 01.01.01
- Bis zum 31.12.1999 stimmen Spezifikation und Programm, aber danach …

Idealbild einer Spezifikation

- Ein übersichtliches und systematisch strukturiertes Schriftstück
- Verständlich und zu gleich exakt, eindeutig formuliert
- Deckt jeden möglichen Fall, der prinzipiell auftreten kann, auch ab.

In der Realität führt dies zu einem Dilemma der Softwareindustrie

- Spezifikation ist zu aufwändig, erfordert Zeit und Geld und wird doch nicht komplett gelesen.

Einige Grundregeln für das Programmieren in Java

In Java macht es keinen Unterschied, ob als Separat ein oder mehrere Leerzeichen oder Tabs verwendet werden.

Deshalb folgende Regeln, die das Lesen des Quellcodes erleichtern:

- Nach jedem Semikolon kommt ein Zeilenumbruch.
- Bei Anweisungen, denen ein Block folgt, steht `{` am Ende der Zeile.
  Anschließend kommt ein Umbruch.
- Nach einer `{` wird der Text eingerückt, nach `}` ausgerückt.

Codebeispiel

```
01   int quadratsumme(int n){
02
03     int summe = 0;
04     if(n==0){
05       return summe;
06     }
07     for(int i=0; i<=n; i++){
08       summe += i*i;
09     }
10
11     return summe;
12   }
```

Einige Grundregeln für das Programmieren in Java

In Java wird zwischen Groß- und Kleinschreibung bei Bezeichnern unterschieden.

- „`hallo`“, „`Hallo`“ und „`HALLO`“ sind drei verschiedene Bezeichner.
- „`IF`“, „`If`“ und „`iF`“ sind zulässige Bezeichner, obwohl „`if`“ ein Schlüsselwort ist.

Einige stilistische Regeln fürs Programmieren

- Wortanfänge innerhalb eines Bezeichners werden mit Großbuchstaben gekennzeichnet. Der Underscore sollte vermieden werden.
- Namen von Klassen beginnen mit Großbuchstaben, z. B. `UrRobot`
- Namen von Variablen und Methoden beginnen mit Kleinbuchstaben, z. B. `putBeepers(amount)`

Einige Grundregeln für das Programmieren in Java

Bezeichner als Kommentare

Damit ein Programm lesbar und verständlich bleibt, sollten die Bezeichner so gewählt sein, dass sie die Methode oder Variable beschreiben.

- `zeichneHausVomNikolaus` für eine Methode, die das Haus vom Nikolaus zeichnet
- Methoden aus den Javabibliotheken wie `drawString`, `drawLine`, etc.
- Dies wird sehr wichtig bei größeren und komplexen Programmen oder wenn mehrere Personen am gleichen Projekt arbeiten.

Einige Grundregeln für das Programmieren in Java

```
01 /**
02  * Dies ist ein Javadoc-Kommentar.
03  * @author David Thomas
04  * @version 9.3.98a
05  */
06 public class Hello {
07     /**
08      * Sprich einen Gruß aus.
09      *
10      * @return String
11      */
12     public String greet() {
13             System.out.println("Hallo, Welt!");
14             return "Hallo,Welt!";
15     }
16 } // class Hello
```

Einige Grundregeln für das Programmieren in Java

Richtig Kommentieren

Hilfreiche Regeln für das Kommentieren findet man in der Spezifikation von JavaDoc, das wir im späteren Verlauf noch ausführlicher behandeln werden.

- Als kurze Einführung dient http://de.wikipedia.org/wiki/JavaDoc
- Oben steht der Klassenkommentar, der Daten über Autor, Version, Datum, etc. enthalten kann und die Klasse kurz beschreiben sollte.
- Über jeder Methode steht ein Kommentar, der Parameter, Rückgabewert, Exceptions, etc. enthalten und die Methode kurz beschreiben sollte.
- Bei komplizierten Operationen können einzeilige Kommentare an Ort und Stelle angebracht werden.

Einige Grundregeln für das Programmieren in Java

Richtig Kommentieren

Einzeilige Kommentare sind der Form

- `// Hier steht ein Kommentar`

Mehrzeilige Kommentare werden Slash und Stern begrenzt wie folgt

- ```
  /**
  * Mehrzeilige Kommentare sind hilfreich,
  * wenn längere Beschreibungen zu Methoden
  * gegeben werden.
  */
  ```

- Bei mehrzeiligen Kommentaren ist der Stern am Anfang der Zeile zwar nicht zwingend notwendig, sollte aber vorhanden sein.
  ```