Elementare Konzepte von Programmiersprachen

Teil 2:

Anweisungen (Statements)

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

Anweisungen

- **Anweisungen** (*statements*) in Java
  - Berechnung (*expression statement*)
  - Zusammenfassung/Block, Verzweigung, Iteration (*control flow statement*)
  - Variable einrichten und initialisieren (*declaration statement*)
- Wird durch ein Semikolon ; abgeschlossen

Expression Statements

- *expression statements*
  - Zuweisungen (mit = oder += etc.),
  - Präfix- oder Postfixformen von ++ oder --
  - Methodenaufrufe
  - Objekterschaffungen mit new

Expression Statements

- Beispiele:
  - i++;
  - int i=5;
  - x += 1;
  - System.out.print("hello");
  - x = Math.sin(Math.PI);

Control-Flow Statement

- **Blöcke** (*block*)
  fassen durch Klammerung { }
  Anweisungssequenzen zu einer einzigen Anweisung zusammen
- **Verzweigungen** (*branch*)
  veranlassen bedingte Übergänge zu anderen Anweisungen im Kontrollfluss
- **Iterationen** (*loop*)
  organisieren strukturierte Wiederholungen (Schleifen) im Kontrollfluss (while, for)

Declaration Statement

- Führt neue Variablen ein und initialisiert sie
- Beispiele:
  - int i;
  - int i = 1;
  - String s;
  - int[] a;
  - int[] a = {1,2,3,4};

Declaration Statements und Blöcke

- Gültigkeitsbereich einer Deklaration ist der aktuelle Block

```
{ int x;      // Block 1
  { int y=1;   // Block 1.1
    // ...
  }
  { int y=2;   // Block 1.2
    // ...
  }
}
```

- Ein x (in Block 1)
- Zwei verschiedene y (in 1.1 bzw. 1.2)

Declaration Statements und Blöcke

- Wenn derselbe Block mehrfach durchlaufen wird, legen Deklarationen jeweils neue Variablen an

Bedingte Anweisungen

Die einfache **if-Anweisung** hat die Form

if (*condition*) *statement*

Die allgemeine **if-Anweisung** hat die Form

if (*condition*) *statement1* else *statement2*

Dabei ist *condition* ein Boolescher Ausdruck.

Falls *condition* zu true evaluiert, wird *statement1* ausgeführt, sonst *statement2*.

Jedes *statement* ist eine Anweisung, also evtl. auch ein Block oder wieder eine if-Anweisung, etc.

Schleifen-Anweisungen

Das Konstrukt

while (*condition*) *statement*

führt *statement* aus, solange der Ausdruck in *condition* zu true evaluiert.

Oftmals wird es sich bei *statement* um einen Block handeln.

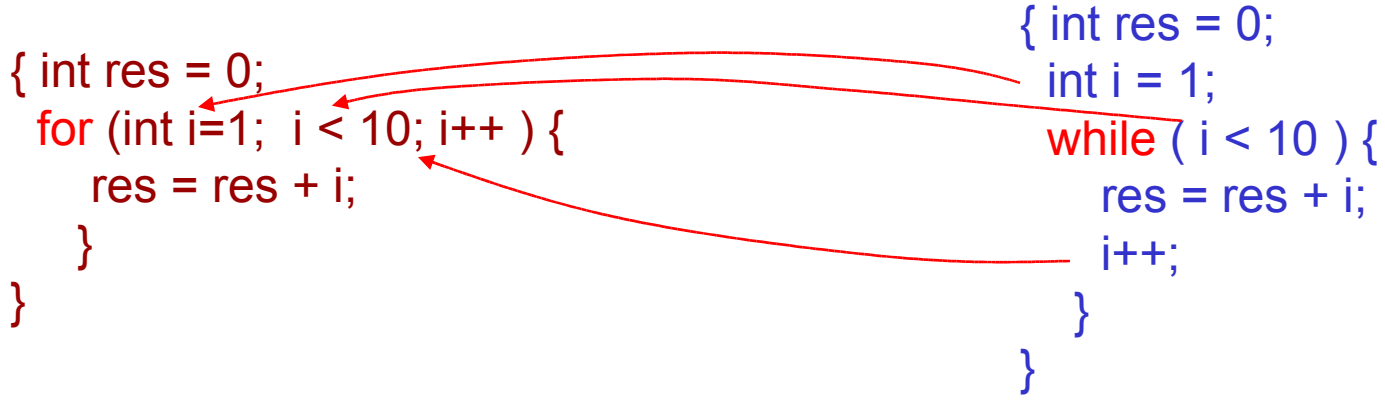
Schleifen-Anweisungen

Die Anweisung

for (init ; condition ; increment) statement

entspricht

{ init ; while(condition) {statement ; increment ; } }



Schleifen-Anweisungen

- Sowohl *init* als auch *increment* dienen vor allem zur Manipulation der Laufvariablen
- Jeder der drei Teile in einer for-Anweisung kann auch leer sein.

Teil 3:
Unterprogramme (Methoden)

Methoden

- Unterprogramme mit Namen heißen in Java **Methode**
- Methoden haben
  - Kopf
    - Name
    - Eingabeparameter (Typen + Namen)
    - Ergebnistyp
  - Rumpf

Methoden

- Beispiel

```
static int plus(int a, int b) { return a+b; }
```

- Dann:

```
System.out.println(plus(3,5));
```

gibt aus:

```
8
```

Methoden

- Methode kann mit passenden **aktuellen Parametern** über ihren Namen **aufgerufen** werden
  - An Stelle des Aufrufs werden die Anweisungen des Unterprogramms ausgeführt
  - Dabei erhalten die formalen Parameter die Werte der aktuellen Parameter
  - Am Ende wird der Aufruf durch das berechnete Ergebnis ersetzt

Methoden

- Methode kann sich selbst aufrufen (Rekursion)
- Methoden können sich gegenseitig aufrufen (p ruft q auf und q ruft p auf)
- Die Parameter verhalten sich im Rumpf wie lokale Variablen
- Rückgabewert kann wegfallen: leerer Typ void

```
void writeHelloWorld()
    {
        System.out.println(„Hello World“);
    }
```