Große Übung
Praktische Informatik 1

2005-11-03

Holger Füßler
fuessler@informatik.uni-mannheim.de
http://www.informatik.uni-mannheim.de/pi4/people/fuessler

1: Announcements / Orga

Die Mailingliste

- Mail an pi4studforum-l-request@pi4.informatik.uni-mannheim.de mit dem Wort subscribe im Body
- oder http://tinyurl.com/bevl4
- Announcements, die Vorlesung und Übung, oder einfach auch nur alle PI1-Hörer betreffen

Die Übungsklausur

- vorauss. am 2005-12-22 in der großen Übung
- vorauss. 60min
- Stoff der Übung und Vorlesung bis dato
- Punkte zählen zum Übungs-Konto

Blatt-Abgabe

- um Punkte gutgeschrieben zu bekommen, müssen sowohl die physikalische als auch die elektronische Abgabe erfolgen!
- elektronische Abgabe:
  - Abgabe der .java Dateien (falls gefordert auch .class)
  - Mail muss Vor- und Nachnamen enthalten
  - Dateien direkt an die Mail anhängen. Keine Archive!

2: Relevantes in Java

Zeichenketten

- komplexer Datentyp (Objekte der String-Klasse)
- deshalb laufen Vergleiche nicht mit Standard-Operatoren
- werden von Java derart gesondert behandelt, dass “Zeichenkette” ein String-Objekt erzeugt
- Der Plus Operator, auf einen String angewendet, verknüpft mehrere Strings zu einem neuen

Zeichenketten: Strings

```
String s = “Zeichenkette”;
System.out.println(s); // Ausgabe der Zeichenkette

String t = s + “ Anhang”; // Anhängen einer anderen Zeichenkette

String s1 = “Hallo”, s2 = “hallo”;
if (s1.equals(s2))
    System.out.println(“Gleich”);
else
    System.out.println(“Ungleich”);
// Ungleich

if (s1.equalsIgnoreCase(s2))
    System.out.println(“Gleich”);
else
    System.out.println(“Ungleich”);
// Gleich
```

System.out.printf(...)

```
System.out.printf("Format-Zeichenkette", Argument, Argument, ...);
```

- Die Format-Zeichenkette kann Platz-Halter für die Argumente enthalten
- Die Platzhalter geben an, was für ein Typ kommt und wie er formatiert wird

System.out.printf(...)

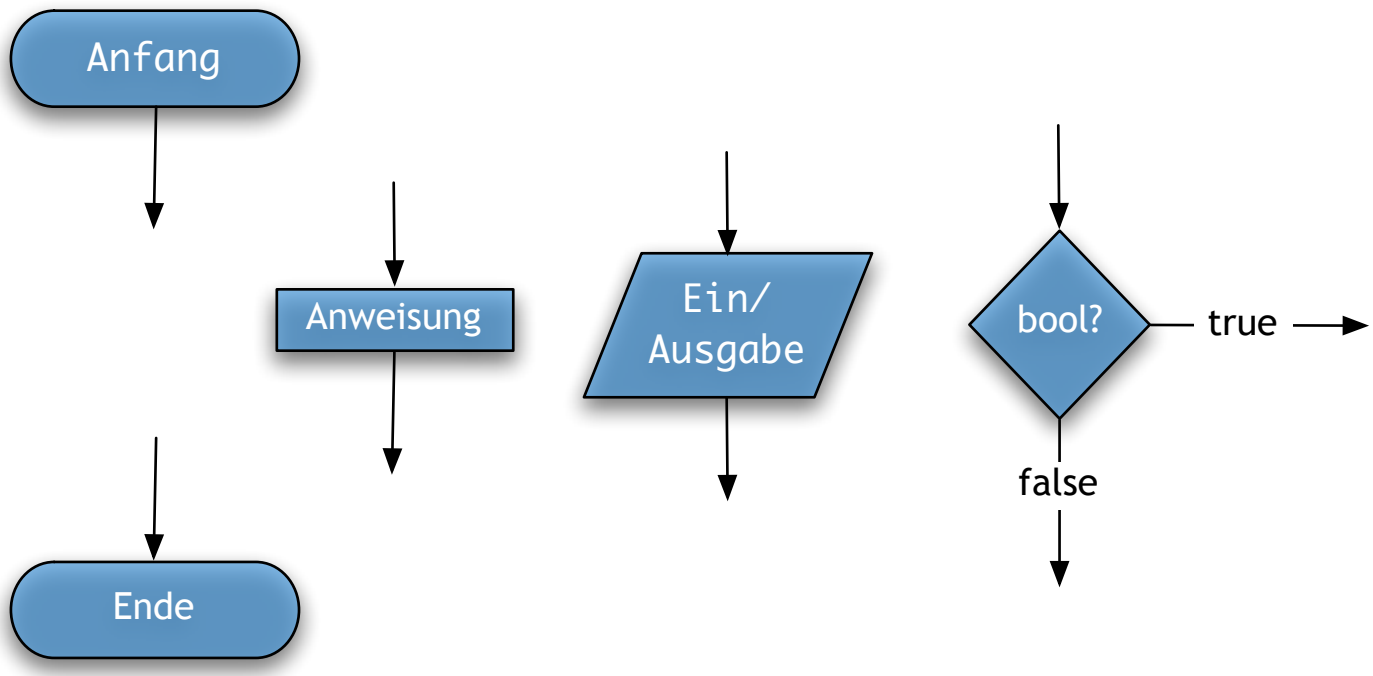
```
double d = 13.2;
int i = 9;
System.out.printf("d: %f, i: %d\n", d, i);
System.out.printf("d '.2f': %.2f, D '08.2f': %08.2f\n", d, d);
System.out.printf("Hexa: 0x%x, Octa: 0%o\n\n", i, i);

double tinyDouble = .00000000000000000003;
double largeDouble = 298934E09;
System.out.printf("tiny: %f, large: %f\n", tinyDouble, largeDouble);
System.out.printf("tiny: %g, large: %g\n", tinyDouble, largeDouble);
System.out.printf("d: %g\n", d);
```

```
d: 13,200000, i: 9
d '.2f': 13,20, d '08.2f': 00013,20
Hexa: 0x9, Octa: 011

tiny 'f': 0,000000, large 'f': 298934000000000,000000
tiny 'g': 3.00000e-23, large 'g': 2.98934e+14
d 'g': 13.2000
```

Blöcke

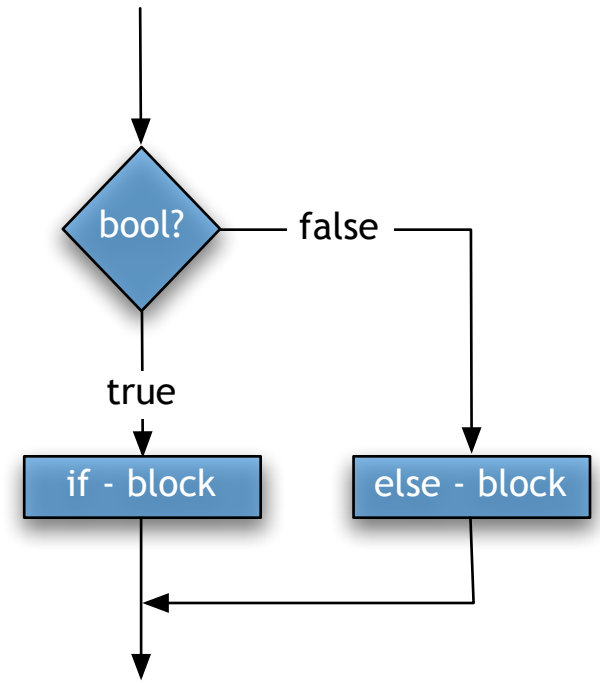
- durch “balancierte” { } gekennzeichnet
- wichtig für:
  - bedingte Anweisungen
  - Schleifen
- Blöcke begrenzen den Gültigkeitsbereich lokaler Variablen

Prozessablauf-Diagramme



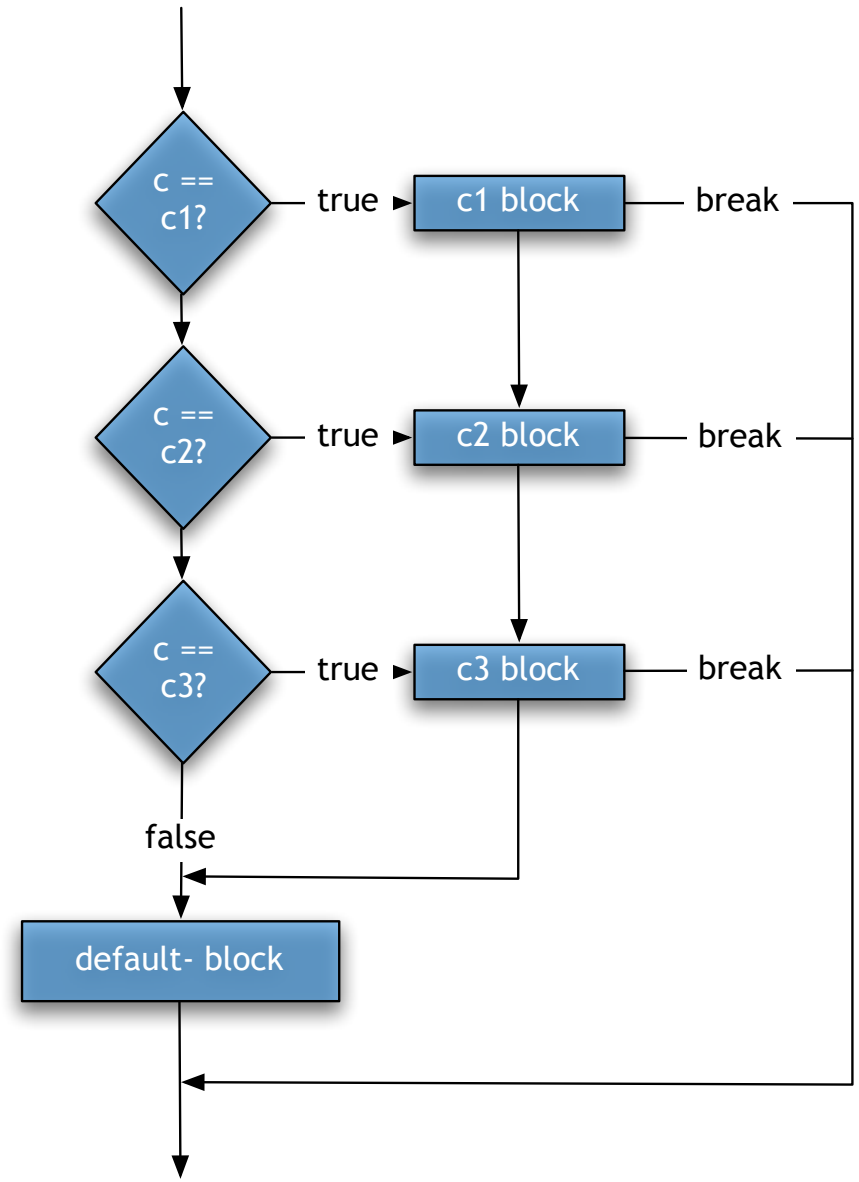
Die if-Anweisung

```
if ( <boolean-Ausdruck> )
    Anweisung;
else
    Anweisung;

if ( <boolean-Ausdruck> ) {
    <Block>
} else {
    <Block>
}
```



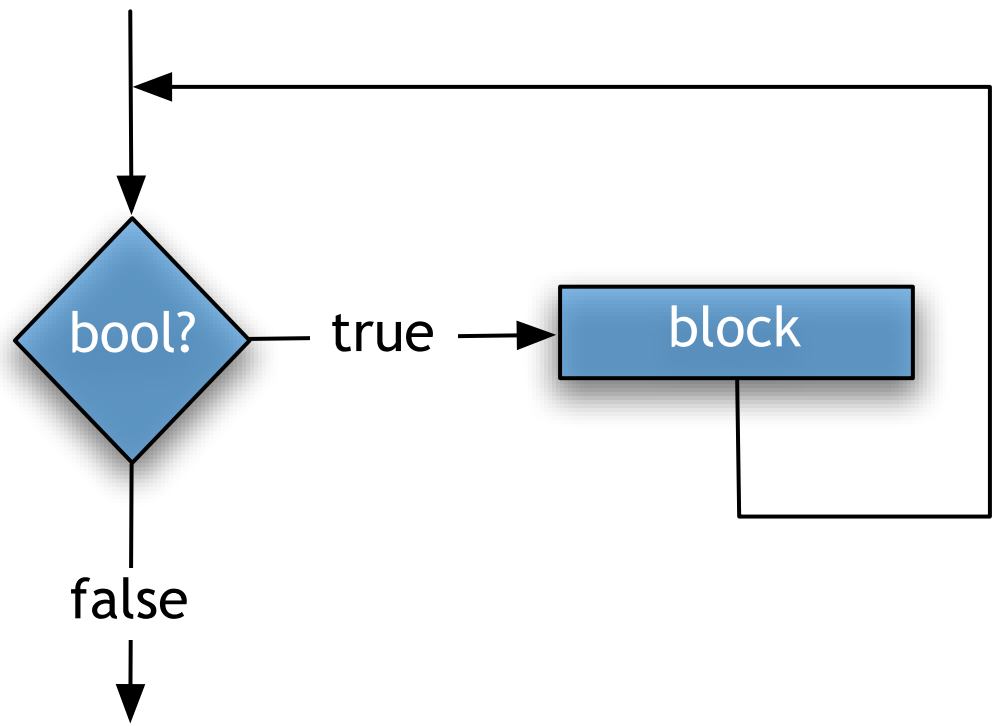
Die switch-Anweisung

```
switch (Ausdruck) {
c1:
    break;
c2:
    break;
c3:
    break;
default:

}
```



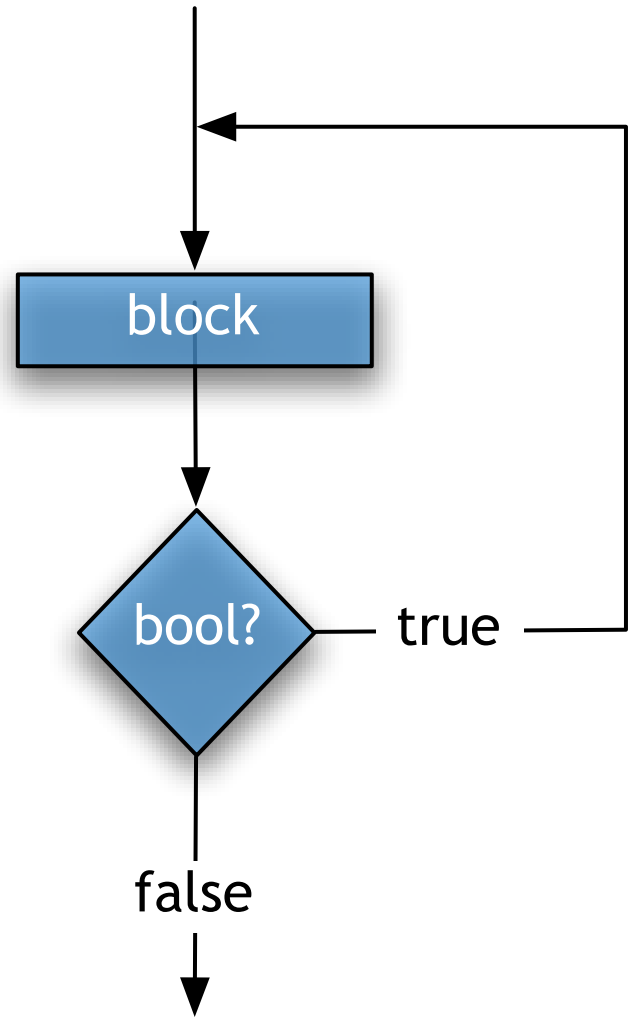
Die while-Schleife

```
while (bool-Ausdruck) {


}
```



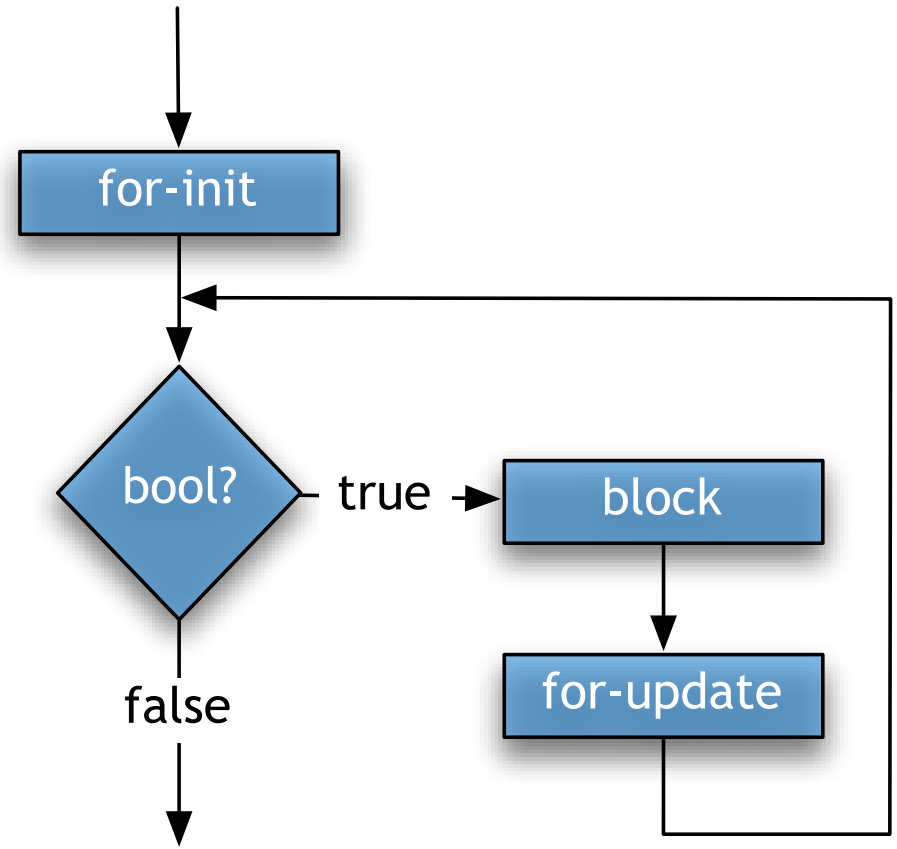
Die do-while-Schleife

```
do {

} while (bool-Ausdruck)
```



Die for-Schleife

```
for (for-init ; bool ; for-update) {

}
```



(static) Methoden

- Methoden/Funktionen sind im Prinzip benannte Blöcke mit einer definierten Schnittstelle, d.h. einer Menge von Parametern, die an den Block übergeben werden und einem Rückgabewert.
- Die main-Methode ist eine Methode ohne Objekt-Bezug (Schlüsselwort static). Aus dieser Methode können deshalb zunächst keine nicht-static Methoden aufgerufen werden (siehe OO-Einführung nächste Woche).
- Ergo beschränken wir uns erst einmal auf static Methoden.

(static) Methoden

```
static Rückgabetyp MethodenBezeichner(ArgTyp1 arg1, ArgTyp2 arg2 ...)
{
    ...
    return Rückgabewert;
}


static double getEURFromDEM(double dem) {
    final double faktor = 1.95583;
    return dem * faktor;
}

public static void main(String[] args) {
    System.out.printf("%g\n", getEURFromDEM(5.0));
}
```

3: Vorlesung

Lernziele bis jetzt...

- theoretisches Verständnis der Java-Syntax
- Variablen und Zuweisungen
- Typisierung und Typ-Umwandlung
- Blöcke, Kontrollstrukturen, Methoden

4: Übungsblatt 2

Hinweise

- Aufgabe 1: static-Methode schreiben
- Aufgabe 2: String-Vergleiche beachten; es reicht kleingeschriebene Strings umzuwandeln
- Aufgabe 3: Welche Programmteile können wie of erreicht werden. Tipp: Flussdiagramm zeichnen

Ende