EINI LW

Einführung in die Informatik für Naturwissenschaftler und Ingenieure

Vorlesung 2 SWS WS 11/12

Dr. Lars Hildebrand
Fakultät für Informatik – Technische Universität Dortmund
lars.hildebrand@udo.edu
http://ls1-www.cs.uni-dortmund.de

technische universität dortmund

Dr. Lars Hildebrand

- **Kapitel 3**
  Basiskonstrukte imperativer (und objektorientierter) Programmiersprachen

- **Unterlagen**
  - Gumm/Sommer, Kapitel 2
  - Echtle/Goedicke, Einführung in die Programmierung mit Java, dpunkt Verlag

Zwischenstand

Dr. Lars Hildebrand

- Variablen
  - Bezeichner, Datentyp, Speicherort, Wert
- Zuweisungen
  - Zuweisungen müssen typverträglich sein
- (Einfache) Datentypen und Operationen
  - Zahlen **`(integer, byte, short, long; float, double)`**
  - Wahrheitswerte **`(boolean)`**
  - Zeichen **`(char)`**
  - Zeichenketten **`(String)`**
  - Typkompatibilität
- Kontrollstrukturen
  - Sequentielle Komposition, Sequenz
  - Alternative, Fallunterscheidung
  - Schleife, Wiederholung, Iteration
- Verfeinerung
  - Unterprogramme, Prozeduren, Funktionen
  - Blockstrukturierung
- Rekursion

Kontrollstrukturen: Sequenz

technische universität dortmund

Dr. Lars Hildebrand

- Sequenz
  - „einfachste“ Kontrollstruktur
  - Trennzeichen zwischen Anweisungen: ;
  - Zuweisungen können aneinandergereiht werden:
  - z.B. a = 3 ; b = a + 4 ;

- Anmerkungen
  - Einrückung beibehalten
  - möglichst **nicht** mehrere Anweisungen in eine Zeile
  - Lesbarkeit ist wichtig!

Kontrollstrukturen: Fallunterscheidung

technische universität dortmund

Dr. Lars Hildebrand

- Verzweigungen steuern den Programmablauf abhängig von Bedingungen:
  - Zuweisungen werden von Bedingungen abhängig sein
  - Bedingungen werden am häufigsten in Form der **`if`**-Anweisungen formuliert:

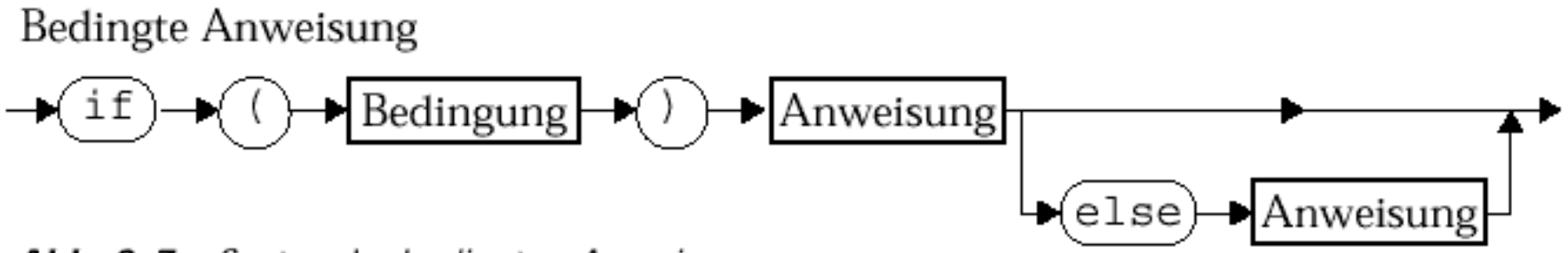


Abb. 2–7 *Syntax der bedingten Anweisung*

Lehrbuch der Programmierung mit Java, Echtle Goedicke, Heidelberg, © dpunkt 2000

Bedingungen

- Eine **Bedingung** ist durch einen **Booleschen Ausdruck** gegeben:
  - Einfache oder komplexe Boolesche Ausdrücke
    - a<=b
    - a> 3 && v || b != c && !w
  - Beachte die Typisierung der Variablen!

- Die Bedingung wird **stets** in runde Klammern eingeschlossen (a<=b) ....

- **Bedeutung**:

  Falls die Auswertung der Bedingung falsch ergibt, wird die erste Anweisung nicht ausgeführt, sondern – falls vorhanden – die Anweisung nach dem Schlüsselwort **`else`**

Beispiel: Fallunterscheidung 1

technische universität dortmund

Dr. Lars Hildebrand

```
int g = 5;
int k = 1;


if (g > k)
  System.out.println("g ist größer");
else
  System.out.println("g ist kleiner oder gleich");
```

Beispiel: Fallunterscheidung 2

technische universität dortmund

Dr. Lars Hildebrand

```
int g,k = 0;
g = ...;
if (g == 1)
    k = k + 100;
else if (g == 2)
    k = k + 1000;
System.out.println("k = " + k);
```

- Frage: Was wird ausgegeben für die Eingabe
  - 1 ?
  - 2 ?
  - 3 ?

technische universität dortmund

Dr. Lars Hildebrand

Block: zusammengehörende Anweisungsfolge

- wird durch `{ . . . }` als zusammengehörend gekennzeichnet.
- sinnvoll z.B. bei Verzweigungen, um mehr als eine Anweisung je Situation angeben zu können.
- Blöcke erlauben in vielen Sprachen die Deklaration von Variablen, die nur innerhalb des Blockes zur Verfügung stehen.
- Begrenzungssymbole können je nach Sprache unterschiedlich sein (begin, end), die Idee ist jedoch stets gleich.

Beispiel: Fallunterscheidung 3

Dr. Lars Hildebrand

Anweisungsfolge

```
double winkel = ...;

if (winkel > 90.0 && winkel < 180.0)
   {
     System.out.println ("stumpfer Winkel");
     winkel = 180.0 - winkel;
   }
```

Verschachtelung von `if`-Anweisungen

technische universität dortmund

Dr. Lars Hildebrand

```
if (...)
    if (...)
    else if ( ... )
```

- Hier tritt das Problem auf, wie das **`else`** gebunden wird, wenn es im Prinzip zu mehreren **`if`**s gehören könnte.
  - In Java (und auch in vielen anderen Programmiersprachen) gilt:
  - Bindung immer an das innere **`if`** , es sei denn, es werden { } gesetzt.

Beispiele zu verschachtelten Ifs ...

technische universität dortmund

Dr. Lars Hildebrand

```
int  i = ...,  j = ...;
if (i == 5)
  if (j == 5)
    System.out.println ("i und j sind 5");

  else
    System.out.println ("nur i ist 5");
else
  if (j == 5)
    System.out.println ("nur j ist 5");
```

Beispiele zu verschachtelten Ifs ...

technische universität dortmund

Dr. Lars Hildebrand

```
int  i = ...,  j = ...;
if (i == 5)
 { if (j == 5)
    System.out.println ("i und j sind 5");
 }
else
  System.out.println ("i ist nicht 5");
```

```
if (j == 5)
    System.out.println ("j ist 5");
```

technische universität dortmund
Dr. Lars Hildebrand

Unübersichtliche Variante

```
double winkel = ...;
if (winkel <   90.0)
  System.out.println ("spitzer Winkel");
else if (winkel ==  90.0)
  System.out.println ("rechter Winkel");
else if (winkel <  180.0)
  System.out.println ("Stumpfer Winkel");
else if (winkel == 180.0)
  System.out.println ("gestreckter" +"Winkel");
```

Einrücken ist sinnvoll

Dr. Lars Hildebrand

Übersichtliche Variante

```
double winkel = ...;
if (winkel <   90.0)
    System.out.println ("spitzer Winkel");
else if (winkel ==  90.0)
        System.out.println ("rechter Winkel");
    else if (winkel <  180.0)
            System.out.println ("Stumpfer Winkel");
        else if (winkel == 180.0)
                System.out.println("gestr. Winkel");
```

Das `switch` - Statement

technische universität dortmund
Dr. Lars Hildebrand

Auswahl aus einer gegebenen Menge von Alternativen mittels eines int-Wertes:

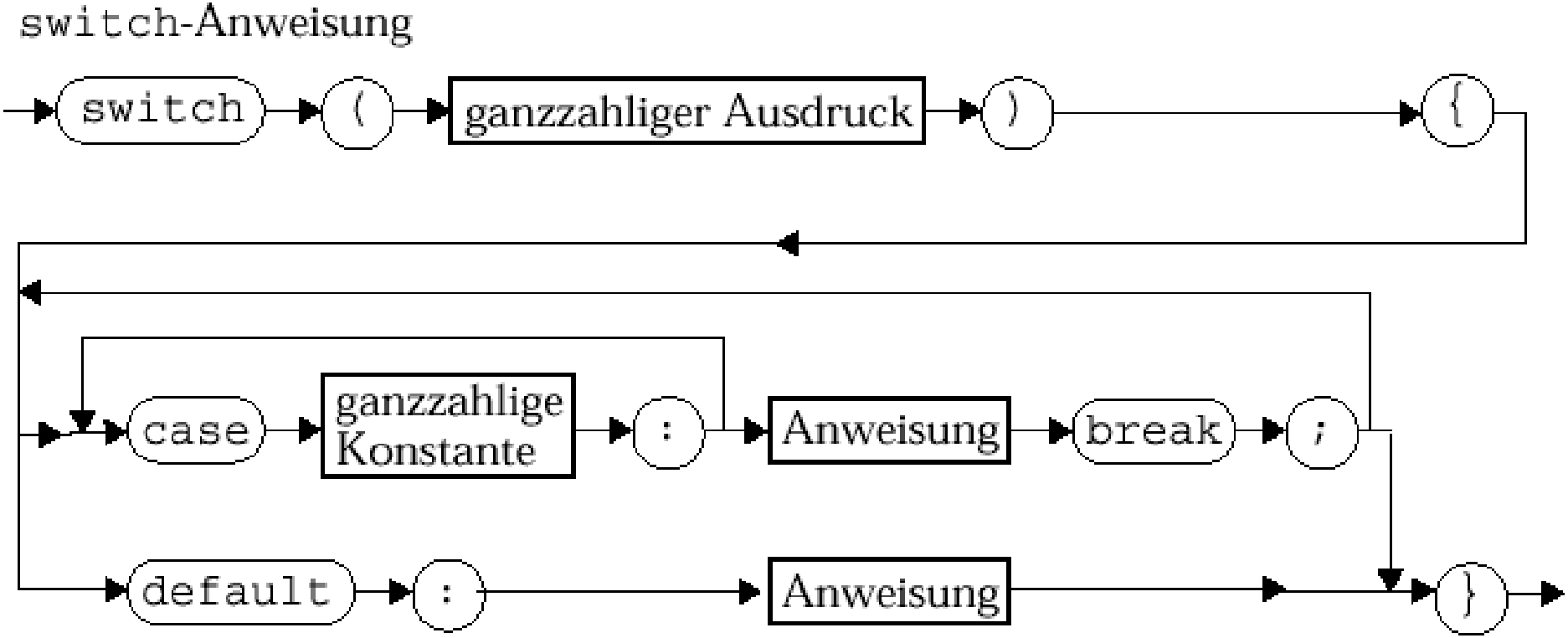


Abb. 2–8 *Syntax der switch-Anweisung*

Lehrbuch der Programmierung mit Java, Echtle Goedicke, Heidelberg, © dpunkt 2000

Beispiel für `switch`


```
int monat;
int quartal;

switch (monat)
{
 case 1: quartal = 1; break;
 case 2: quartal = 1; break;
 case 3: quartal = 1; break;
 case 4: quartal = 2; break;
  . . .
  . . .
 case 11: quartal = 4; break;
 case 12: quartal = 4; break;
}
```

Beispiel für `switch`

Dr. Lars Hildebrand

```
int monat;
int quartal;

switch (monat)
{
 case 1:  case 2:  case 3:  quartal = 1; break;
 case 4:  case 5:  case 6:  quartal = 2; break;
 case 7:  case 8:  case 9:  quartal = 3; break;
 case 10: case 11: case 12: quartal = 4; break;
}
```

Beispiel für `switch`

Dr. Lars Hildebrand

```
int monat;
int quartal;

switch (monat)
{
 case 1:  case 2:  case 3:  quartal = 1; break;
 case 4:  case 5:  case 6:  quartal = 2; break;
 case 7:  case 8:  case 9:  quartal = 3; break;
 default: quartal = 4;
}
```

Beispiel für `switch`


```
char buchstabe;

switch (buchstabe)
{
 case 'a':
 case 'e':
 case 'i':
 case 'o':
 case 'u': System.out.println("Vokal!"); break;
 default:  System.out.println("Konsonant!");
}
```

Komplexes Beispiel für `switch`

technische universität dortmund

Dr. Lars Hildebrand

```
char ziffer;   int wert;
ziffer = ...;
switch (ziffer)
{
  case '0': case '1': case '2': case '3': case '4':
  case '5': case '6': case '7': case '8': case '9':
    wert = ziffer - '0';
    break;

  case 'a': case 'b': case 'c': case 'd': case 'e':  case 'f':
    wert = ziffer - 'a' + 10;
    break;

  case 'A': case 'B': case 'C': case 'D': case 'E':  case 'F':
    wert = ziffer - 'A' + 10;
    break;

  default: System.out.println(Ziffer + " ist ungültig");
}
```

Bemerkungen

- Im Beispiel auf der vorherigen Folie
  - **`char`** ist typkompatibel zu **`int`**
  - Subtraktion ist daher definiert
- Es gibt Formen des **`switch`**-Statements, die gelegentlich ohne **`break`** auskommen
  - "durchrutschen" ist dann gewünscht
  - ist aber unübersichtlich.
- Auch hier gilt: Klammern erhöhen in der Regel die Übersicht und Lesbarkeit.

Zwischenstand

Dr. Lars Hildebrand

- Variablen
  - Bezeichner, Datentyp, Speicherort, Wert
- Zuweisungen
  - Zuweisungen müssen typverträglich sein
- (Einfache) Datentypen und Operationen
  - Zahlen **`(integer, byte, short, long; float, double)`**
  - Wahrheitswerte **`(boolean)`**
  - Zeichen **`(char)`**
  - Zeichenketten **`(String)`**
  - Typkompatibilität
- Kontrollstrukturen
  - Sequentielle Komposition, Sequenz
  - Alternative, Fallunterscheidung
  - Schleife, Wiederholung, Iteration
- Verfeinerung
  - Unterprogramme, Prozeduren, Funktionen
  - Blockstrukturierung
- Rekursion

Wiederholungen

Dr. Lars Hildebrand

- Bisher sind die besprochenen Programme einmal durchgelaufen
  - jede Anweisung wurde höchstens einmal ausgeführt

- Beispiele für Wiederholungen:
  - Mathematische Folgen und Reihen
  - Verarbeitung wiederkehrender Vorgänge (Buchungen...)
  - Primzahltest:
    - Ist die Zahl n eine Primzahl?
    - teste ob 2,3,..., $\sqrt{n}$ Teiler von n sind.

Schleifen

- Drei Varianten
  - **`while`** (Bedingung) { Anweisungsfolge }
  - **`do`** { Anweisungsfolge } **`while`** (Bedingung)
  - **`for`** (Initialisierung, Bedingung, Fortschritt) { Anweisungsfolge }
- Diese Vielfalt ist „nur“ durch Komfort begründet
- Die allgemeinste Form ist die **`while`** -Schleife

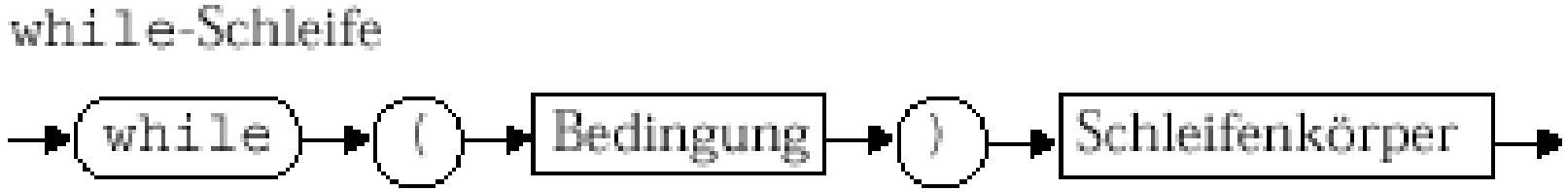


Abb. 2–9 *Syntax der while-Schleife*

Lehrbuch der Programmierung mit Java, Echtle Goedicke, Heidelberg, © dpunkt 2000

Wirkung von Schleifen

technische universität dortmund
Dr. Lars Hildebrand

`while` (Bedingung) { Anweisungsfolge }

- Grundsätzlich gilt, dass der Schleifenkörper solange wiederholt wird, wie die Bedingung wahr ist (auch 0-mal).
- Die Bedingung wird zu true oder false ausgewertet.
- Die Bedeutung kann auch durch ein Diagramm dargestellt werden (**Kontrollflussgraph**)

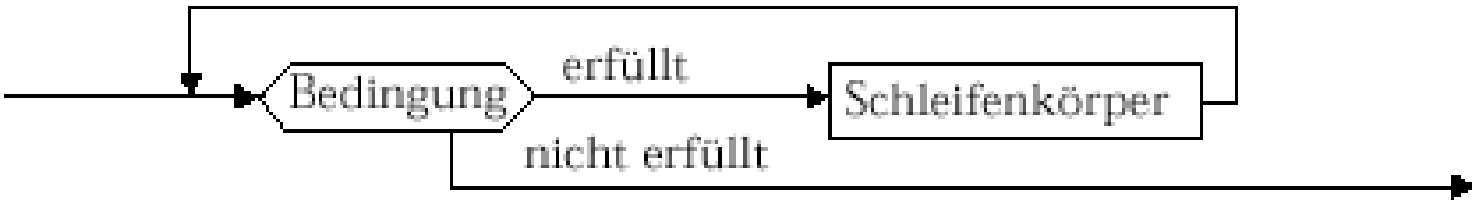


Abb. 2–10 Semantik der while-Schleife

Lehrbuch der Programmierung mit Java, Echtle Goedicke, Heidelberg, © dpunkt 2000

Beispiel: Reihenberechnung

Dr. Lars Hildebrand

```
int  i = 1,  a = 2;
while (i < 100)
{
  a = 4*a;
  System.out.println("i =" + i + "\t" + "a=" + a);
  i++;
};
```

- In 3 Zeilen werden 99 Ausführungen von Zeilen beschrieben.
- Kleine Fehler haben große Auswirkungen:
  z.B.: i statt i++
- Die häufigsten Fehler in Schleifen
  - Bedingung verändert sich nicht oder ist falsch.
  - Bedingung signalisiert falsches Ende.
  - Falsche Initialisierung.

Beispiel: Primzahl-Test (1)

Dr. Lars Hildebrand

Algorithmus-Idee

- Teste, ob 2,3,..., √n Teiler von n sind (kann natürlich optimiert werden!)

Umsetzung

- Wir prüfen ein konkretes n
- Solange ***kein Teiler gefunden*** und ***die Grenze nicht erreicht***: erhöhe den Teiler um eins
- Die Bedingung „kein Teiler gefunden“ wird in Boolescher Variablen `istPrimzahl` gespeichert

```
while ( teiler <= wurzel  &&  istPrimzahl == true )
  if (n % teiler == 0)
       istPrimzahl = false;
    else
       teiler++;
```

Beispiel: Primzahl-Test (2)

technische universität dortmund
Dr. Lars Hildebrand

```
int  n, wurzel,  teiler = 2;
boolean  istPrimzahl = true;

n = readInt();

wurzel = (int) java.lang.Math.sqrt((float) n);
```

```
while ( (teiler <= wurzel) && (istPrimzahl == true) )
  if (n % teiler == 0)
            istPrimzahl = false;
    else
            teiler++;
```

```
System.out.println (n + " prim: " + istPrimzahl);
```

```
public static int readInt() {/*...*/}
```

Schleifen sind schwieriges Programmkonstrukt

Dr. Lars Hildebrand

- Einfaches aber effizientes Verfahren ist die Darstellung der Werteverläufe über Tabellen:

n	Wurzel	Teiler	istPrimzahl
21	4	2	true
21	4	3	**false**

(2 Iterationen)

n	Wurzel	Teiler	istPrimzahl
37	6	2	true
37	6	3	true
37	6	4	true
37	6	5	true
37	6	6	**true**

(5 Iterationen)

- Auch hier: ggfs. Klammern und Einrückung zur Erhöhung der Lesbarkeit
- Schleifen sind schwierig, aber ohne sie kommt man nicht aus
  - Warum schwierig ?
  - Warum nötig ?

do-while-Schleife

- Durchlauf des Schleifenkörpers **mindestens 1 Mal**
- Syntax und Semantik durch Diagramme

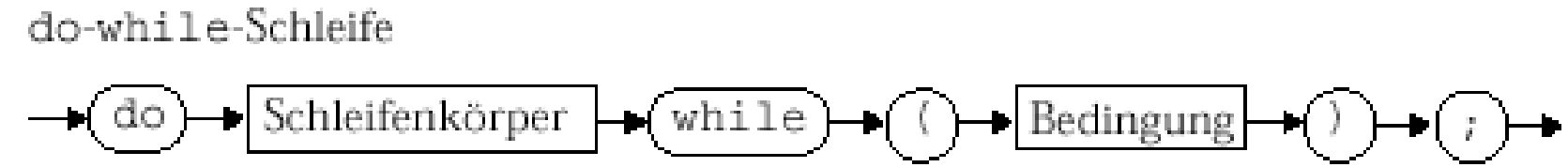


Abb. 2–12 *Syntax der do-while-Schleife*

Lehrbuch der Programmierung mit Java, Echtle Goedicke, Heidelberg, © dpunkt 2000

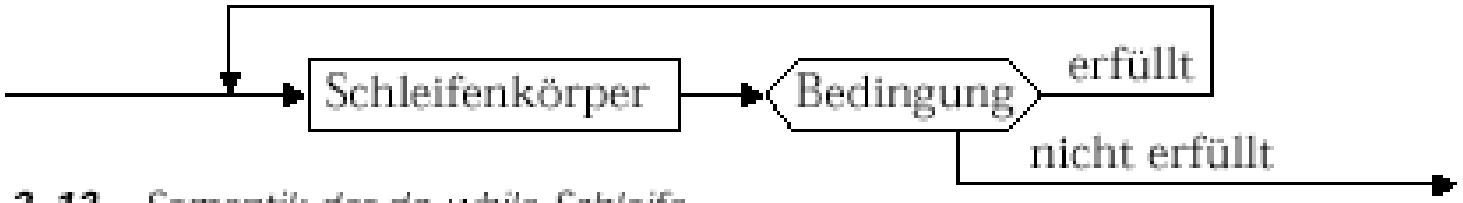


Abb. 2–13 *Semantik der do-while-Schleife*

Lehrbuch der Programmierung mit Java, Echtle Goedicke, Heidelberg, © dpunkt 2000

Beispiel do-while (1)

technische universität dortmund
Dr. Lars Hildebrand

```
int   summe = 0,   anzahl = 0;
```

```
do {
      summe = summe + Eingabe();
      anzahl++;
   }

while (summe <= 100);
```

```
System.out.println ("Summe: " + summe +
                    ", Anzahl: " + anzahl);
```

```
public static int Eingabe() {/*...*/}
```

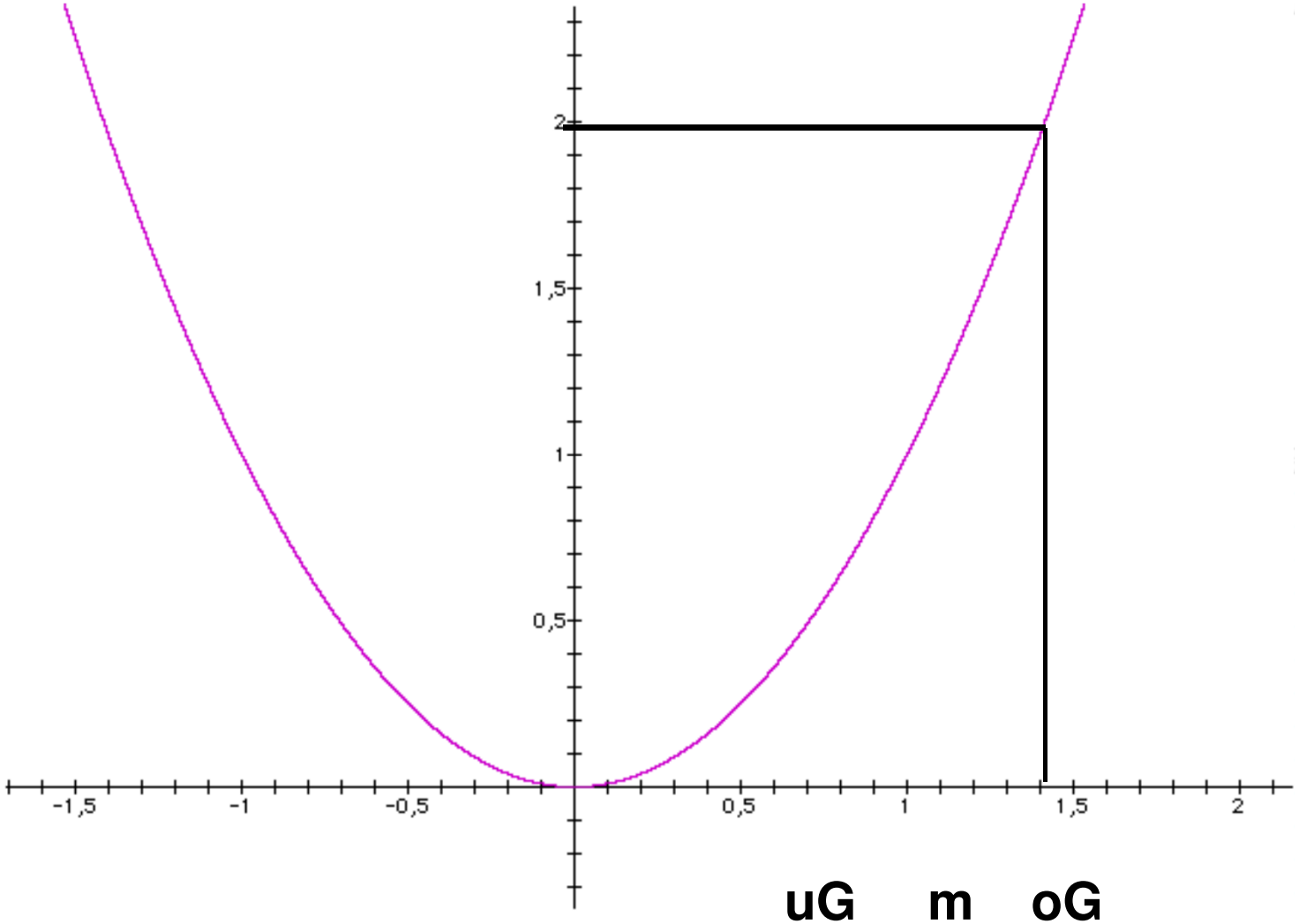
Beispiel: einfache Numerik Funktionen

- Berechnung der Quadratwurzel `sqrt( float n)` für n > 0:
- Nützlichkeit klar,
  - in vielen Programmen unabhängig vom Kontext verwendbar
  - daher auch in Bibliotheken (Libraries) stets verfügbar
- Eine Berechnungsidee: Intervallschachtelung
  - Finde eine untere Schranke.
  - Finde eine obere Schranke.
  - Verringere obere und untere Schranke, bis der Abstand hinreichend gering geworden ist.
  - Etwas konkreter: Halbiere Intervall, fahre mit demjenigen Teilintervall fort, das das Resultat enthält.

Beispiel do-while (2)

Dr. Lars Hildebrand

Quadrat-Wurzel Berechnung mittels Intervallschachtelung



Beispiel do-while (2)

Dr. Lars Hildebrand

- Quadrat-Wurzel Berechnung mittels Intervallschachtelung
- Rückführung der Berechnung auf Quadrierung

- Start: Intervall [0,x+1], Mitte m = 0,5*(uG+oG)

- Algorithmus:
  - Berechne neue Mitte m = 0,5*(uG+oG)
  - Falls $m^2 > x$: oG = m
    sonst: uG = m
  - Abbruch: falls oG - uG < ε

Beispiel do-while (2)

technische universität dortmund
Dr. Lars Hildebrand

```
double x = ...,
       uG = 0,  oG = x + 1,  m,
       epsilon = 0.001;
```

```
do { m = 0.5*(uG + oG);
     if (m*m > x)
          oG = m;
     else
          uG = m;
   }
while (oG - uG > epsilon);
```

```
System.out.println (  "Wurzel " + x
                    + " beträgt ungefähr "
                    + m);
```

Zwischenstand

Dr. Lars Hildebrand

- Variablen
  - Bezeichner, Datentyp, Speicherort, Wert
- Zuweisungen
  - Zuweisungen müssen typverträglich sein
- (Einfache) Datentypen und Operationen
  - `integer (byte, short, long; float, double)`
  - `boolean`
  - `char`
  - `String`
  - Typkompatibilität
- Kontrollstrukturen
  - Sequentielle Komposition, Sequenz
  - Alternative, Fallunterscheidung
  - Schleife, Wiederholung, Iteration (while, do while, for)
- Verfeinerung
  - Unterprogramme, Prozeduren, Funktionen
  - Blockstrukturierung
- Rekursion

Vielen Dank für Ihre Aufmerksamkeit!

Nächste Termine

- Nächste Vorlesung 18.11.2011, 08:30