

# **EINI**

# **LogWing/WiMa/MP**

**Einführung in die Informatik für  
Naturwissenschaftler und Ingenieure**

**Vorlesung      2 SWS      WS 23/24**

**Dr. Lars Hildebrand**  
**Fakultät für Informatik – Technische Universität Dortmund**  
**[lars.hildebrand@tu-dortmund.de](mailto:lars.hildebrand@tu-dortmund.de)**  
**<http://ls14-www.cs.tu-dortmund.de>**

## Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

### In diesem Kapitel:

- **Prolog**
- Kontrollstrukturen
  - Sequenz
  - Block
  - Alternative
  - Iteration

## ▶ Kapitel 3

Basiskonstrukte imperativer (und objektorientierter) Programmiersprachen

## ▶ Unterlagen

- ▶ Echte, Klaus und Michael Goedicke: *Lehrbuch der Programmierung mit Java*. Heidelberg: dpunkt-Verl, 2000. (→ ZB)
- ▶ Gumm, Heinz-Peter und Manfred Sommer: *Einführung in die Informatik*, 10. Auflage. München: De Gruyter, 2012. (Kap. 2) (→ Volltext aus Uninetz)

# Übersicht

EINI LogWing /  
WiMa

## Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

- ✓ Variablen
- ✓ Zuweisungen
- ✓ (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

### In diesem Kapitel:

- **Prolog**
- Kontrollstrukturen
  - Sequenz
  - Block
  - Alternative
  - Iteration

# Kontrollstrukturen: Sequenz

## ▶ Sequenz

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

## ❖ Anmerkungen:

- ▶ Einrückung beibehalten
- ▶ (möglichst) nur **eine** Anweisung pro Zeile
- ▶ Wichtig: Lesbarkeit des Codes!

### In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - Block
  - Alternative
  - Iteration

# Kontrollstrukturen: Block

Ein Block ist eine zusammengehörende Anweisungsfolge.

- ▶ Zusammengehörigkeit wird durch `{ . . . }` markiert.
- ▶ Anwendung sinnvoll z.B. bei Verzweigungen, um mehr als eine Anweisung pro Situation angeben zu können
- ▶ Blöcke erlauben in vielen Sprachen die Deklaration von Variablen, die nur innerhalb dieses Blockes zur Verfügung stehen.
- ▶ Begrenzungssymbole können je nach Sprache unterschiedlich sein (`begin`, `end`), die Idee ist jedoch immer dieselbe.

EINI LogWing /  
WiMa

## Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - **Block**
  - Alternative
  - Iteration



# Kontrollstrukturen: Sequenz und Block

Artikel im EINI-Wiki:

- **Kontrollstrukturen**
- **Block**
- **Anweisungssequenzen**

## Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

## In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - **Block**
  - Alternative
  - Iteration

# Kontrollstrukturen: Alternativen

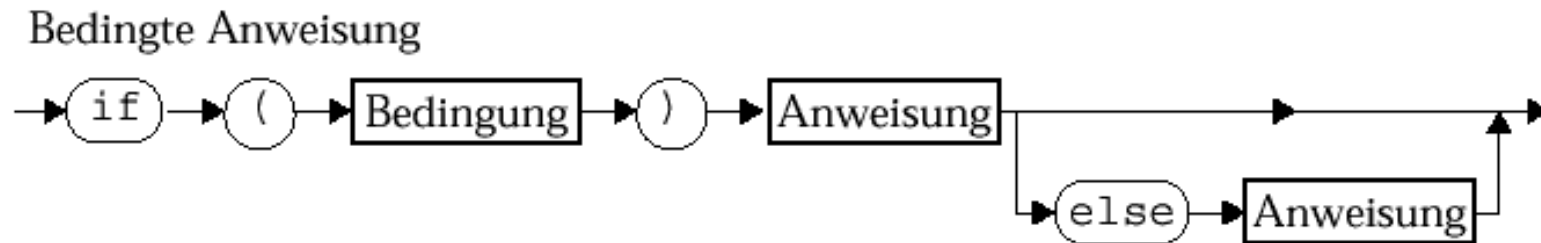
Alternativen beruhen auf Fallunterscheidungen.

EINI LogWing /  
WiMa

## Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

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



Echtle/Goedicke, Heidelberg: *Abb. 2–7*, S. 48 © dpunkt 2000.

## In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - Block
  - **Alternative**
  - Iteration

# Bedingungen

## Kapitel 3

### In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
- Sequenz
- Block
- **Alternative**
- Iteration

- ▶ Eine **Bedingung** ist durch einen **Booleschen Ausdruck** gegeben:
  - ▶ Einfache oder komplexe Boolesche Ausdrücke:
    - $a \leq b$
    - $a > 3 \ \&\& \ v \ || \ b \ != \ c \ \&\& \ !w$
  - ❖ Beachte die Typisierung der Variablen!
- ▶ Die Bedingung wird **stets** in runde Klammern eingeschlossen:  
(  $a \leq b$  ) ....
- ▶ **Bedeutung:**
  - ▶ Falls die Auswertung *wahr* ergibt, wird die nächste Anweisung ausgeführt. Ein möglicher **else**-Zweig wird nicht ausgeführt.
  - ▶ Falls die Auswertung der Bedingung *falsch* ergibt, wird die erste Anweisung nicht ausgeführt, sondern – falls vorhanden – die Anweisung nach dem Schlüsselwort **else**.



# Beispiel: Fallunterscheidungen (1)

```
int g = 5;
```

```
int k = 1;
```

```
if (g > k)
```

```
    System.out.println("g ist größer");
```

```
else
```

```
    System.out.println("g ist kleiner oder gleich");
```

EINI LogWing /  
WiMa

## Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

### In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - Block
  - **Alternative**
  - Iteration

## Beispiel: Fallunterscheidungen (2)

```
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 ?

- Prolog
- **Kontrollstrukturen**
- Sequenz
- Block
- **Alternative**
- Iteration

# Beispiel: Fallunterscheidungen (3)

## Anweisungsfolge - Block

```
double winkel = ...;
```

```
if (winkel > 90.0 && winkel < 180.0)
```

```
{  
    System.out.println ("stumpfer Winkel");  
    winkel = 180.0 - winkel;  
}
```

EINI LogWing /  
WiMa

### Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

### In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - Block
  - **Alternative**
  - Iteration

# Verschachtelung von `if`-Anweisungen

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

- ▶ Hier tritt die Frage auf, wie ein **else** gebunden wird, wenn es im Prinzip zu mehreren **ifs** gehören könnte.
- ▶ In Java, wie in vielen anderen Programmiersprachen auch, gilt:
  - ❖ Bindung immer an das innere **if**, es sei denn, es werden `{ }` gesetzt!

## Kapitel 3

### In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - Block
  - **Alternative**
  - Iteration

# Beispiel für verschachtelte ifs I

```
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");
```

EINI LogWing /  
WiMa

## Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

### In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - Block
  - **Alternative**
  - Iteration

# Beispiel für verschachtelte ifs II

```
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");
```

EINI LogWing /  
WiMa

## Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

### In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - Block
  - **Alternative**
  - Iteration

# Einrücken ist sinnvoll I

## 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");
```

EINI LogWing /  
WiMa

### Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

### In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - Block
  - **Alternative**
  - Iteration

# Einrücken ist sinnvoll II

## 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");
```

EINI LogWing /  
WiMa

### Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

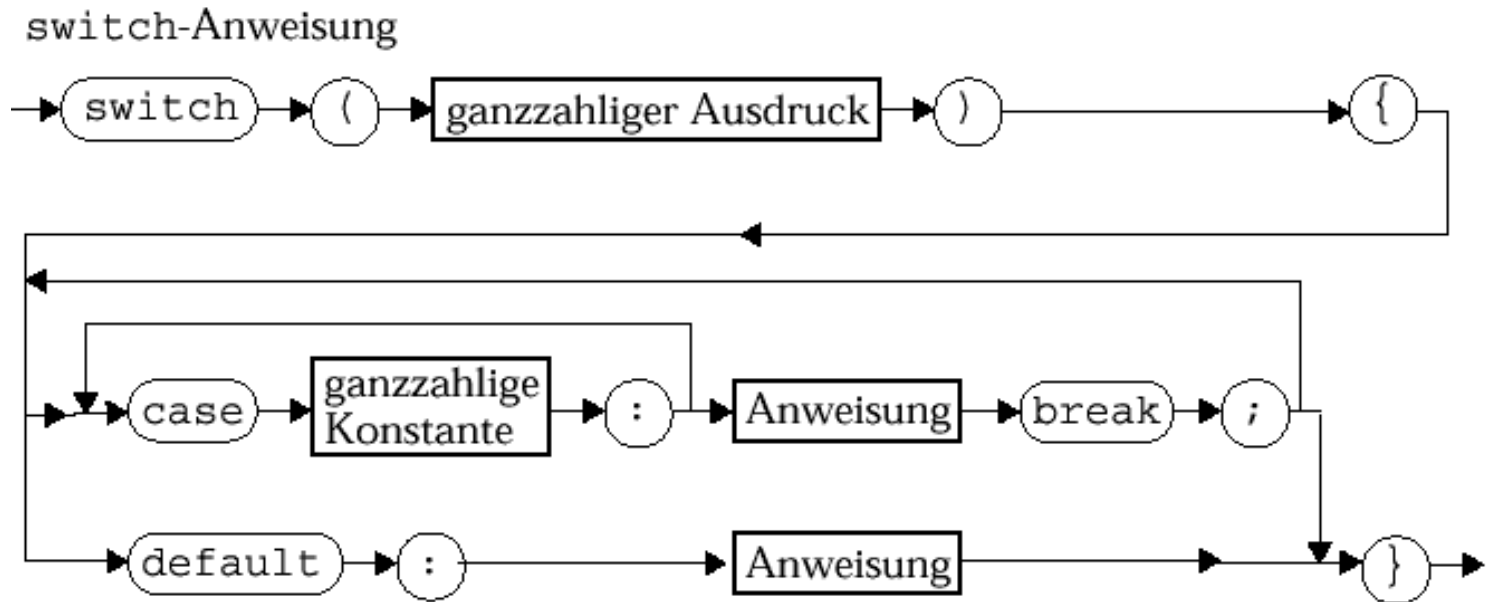
### In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
- Sequenz
- Block
- **Alternative**
- Iteration



# Das switch - Statement

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



Echtle/Goedicke, Heidelberg: *Abb. 2–8*, S. 51 © dpunkt 2000.

EINI LogWing /  
WiMa

## Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

### In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
- Sequenz
- Block
- **Alternative**
- Iteration

# Beispiel für switch I

```
int monat = 4;  
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;  
}
```

- Prolog
- **Kontrollstrukturen**
- Sequenz
- Block
- **Alternative**
- Iteration

# Beispiel für switch II

```
int monat = 4;
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;
}
```

► Bedingungen dürfen mehrfach vorkommen.

- Prolog
- **Kontrollstrukturen**
- Sequenz
- Block
- **Alternative**
- Iteration

# Beispiel für switch III

```
int monat = 4;
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;
}
```

- ▶ **Default** wird erreicht, wenn keine der Bedingungen zutrifft.

- Prolog
- **Kontrollstrukturen**
- Sequenz
- Block
- **Alternative**
- Iteration

# Beispiel für switch IV

```
char buchstabe = 'x';
```

```
switch (buchstabe)
```

```
{
```

```
    case 'a':
```

```
    case 'e':
```

```
    case 'i':
```

```
    case 'o':
```

```
    case 'u': System.out.println("Vokal!"); break;
```

```
    default: System.out.println("Konsonant!");
```

```
}
```

EINI LogWing /  
WiMa

## Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

### In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
- Sequenz
- Block
- **Alternative**
- Iteration

- ▶ Jeder primitive Datentyp, der implizit in einen `int`-Wert umgewandelt werden kann, eignet sich als Kriterium.

# Der Vorteil von switch

```
int zahl = 2;
char zeichen;

if(zahl==1) zeichen = '1';
else if(zahl==2) zeichen = '2';
else if(zahl==3 || zahl==4) zeichen = 'x';
else zeichen = '?';
```

```
int zahl = 2;
char zeichen;

switch (zahl)
{
    case 1: zeichen = '1'; break;
    case 2: zeichen = '2'; break;
    case 3: case 4: zeichen = 'x'; break;
    default: zeichen = '?';
}
```

EINI LogWing /  
WiMa

## Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
- Sequenz
- Block
- **Alternative**
- Iteration

# Regeln für `switch`

EINI LogWing /  
WiMa

## Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - Block
  - **Alternative**
  - Iteration

- ▶ Die Bedingung wird vollständig ausgewertet.
- ▶ Zur Fallunterscheidung dienen nur Konstanten.
- ▶ Wird eine Übereinstimmung mit einer Konstanten gefunden, wird bei der zugehörigen Anweisung fortgesetzt.
- ▶ Wird keine Übereinstimmung gefunden, wird bei `default` fortgesetzt.
- ▶ Die Reihenfolge von `case` und `default` ist beliebig.
- ▶ `break` beendet die `switch`-Anweisung sofort.
- ▶ Für Bedingung und Konstanten sind nur folgende Datentypen zugelassen:
  - ▶ `byte`, `short`, `int`
  - ▶ `char`



# Kontrollstrukturen: Alternative

Artikel im EINI-Wiki:

- **Alternative**
- **Fallunterscheidung**
- **Boolean**
- **Boolescher Ausdruck**
- **Schleife**
  - Schlüsselwörter
  - break-Statement

## Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

## In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - Block
  - **Alternative**
  - Iteration



# Zwischenstand

EINI LogWing /  
WiMa

## Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

- ✓ Variablen
- ✓ Zuweisungen
- ✓ (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

### In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - Block
  - **Alternative**
  - Iteration

# Iteration

## Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

### In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - Block
  - Alternative
  - **Iteration**

- ▶ Bisher sind die besprochenen Programme nur 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  $2, 3, \dots, \sqrt{n}$  Teiler von  $n$  sind.

# Schleifen

▶ Drei Varianten:

▶ **while** (Bedingung) { Anweisungsfolge }

▶ **do** { Anweisungsfolge } **while** (Bedingung)

▶ **for** (Initialisierung; Bedingung; Fortsetzung)  
{ Anweisungsfolge }

▶ Diese Vielfalt ist „nur“ durch Komfort begründet.

▶ Die allgemeinste Form ist die **while**-Schleife:

while-Schleife



Echtle/Goedicke, Heidelberg: *Abb. 2–9*, S. 53 © dpunkt 2000.

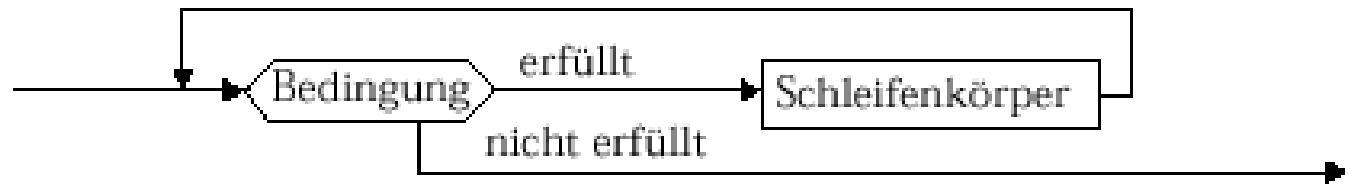
In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - Block
  - Alternative
  - **Iteration**

# while-Schleife

```
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**):



Echtle/Goedicke, Heidelberg: *Abb. 2–10*, S. 53 © dpunkt 2000.

EINI LogWing /  
WiMa

## Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

### In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - Block
  - Alternative
  - **Iteration**

# Beispiel: Reihenberechnung

```
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.

EINI LogWing /  
WiMa

## Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
- Sequenz
- Block
- Alternative
- **Iteration**

# Beispiel: Primzahltest I

## Algorithmus-Idee:

- ▶ Teste, ob  $2, 3, \dots, \sqrt{n}$  Teiler von  $n$  sind (kann natürlich optimiert werden!).

## Umsetzung:

- ▶ Wir prüfen ein konkretes  $n$ .
- ▶ Solange **kein Teiler gefunden** und **die Grenze nicht erreicht** ist: 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++;
```

## Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

### In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
- Sequenz
- Block
- Alternative
- **Iteration**

# Beispiel: Primzahltest II

```
01 int n, wurzel, teiler = 2;
02 boolean istPrimzahl = true;
03
04 n = 42;
05
06 wurzel = (int) java.lang.Math.sqrt((float) n);
07
08 while ( (teiler <= wurzel) && (istPrimzahl == true) )
09     if (n % teiler == 0)
10         istPrimzahl = false;
11     else
12         teiler++;
13
14 System.out.println (n + " prim: " + istPrimzahl);
```

# Schleifen: Schwieriges Programmkonstrukt

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

n	Wurzel	Teiler	istPrimzahl
21	4	2	true
21	4	3	<b>false</b>

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	<b>true</b>

5 Iterationen

- ▶ Auch hier: Ggfs. Klammern und Einrückung zur Erhöhung der Lesbarkeit verwenden.
- ▶ Schleifen sind schwierig, aber ohne sie kommt man nicht aus!
  - ▶ Warum schwierig?
  - ▶ Warum notwendig?

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - Block
  - Alternative
  - **Iteration**



# do-while-Schleife

```
do {Anweisungsfolge} while (Bedingung);
```

EINI LogWing /  
WiMa

## Kapitel 3

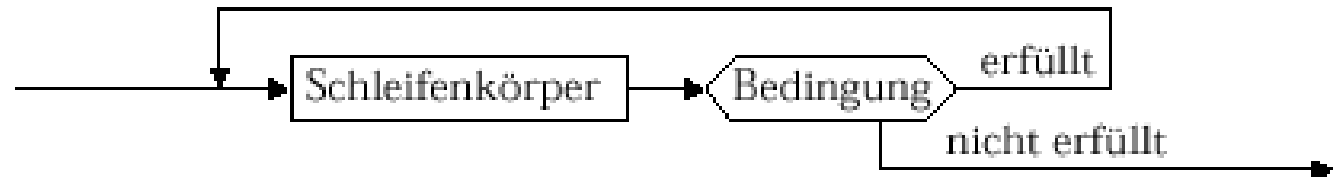
Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

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

do-while-Schleife



Echtle/Goedicke, Heidelberg: *Abb. 2-12*, S. 56 © dpunkt 2000.



Echtle/Goedicke, Heidelberg: *Abb. 2-13*, S. 56 © dpunkt 2000.

## In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - Block
  - Alternative
  - **Iteration**

# Beispiel: do-while (1)

```
import java.util.Scanner;

public class Beispiel {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int summe = 0, anzahl = 0;

        do
        {
            summe = summe + scan.nextInt();
            anzahl++;
        }
        while (summe <= 100);

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

EINI LogWing /  
WiMa

## Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
- Sequenz
- Block
- Alternative
- **Iteration**

# Beispiel: do-while (2) I

## Beispiel: einfache Numerik-Funktionen

- ▶ Berechnung der Quadratwurzel `sqrt` für  $n > 0$
- ▶ Nützlichkeit klar,
  - ▶ da 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 dem Teilintervall fort, das das Resultat enthält.

### Kapitel 3

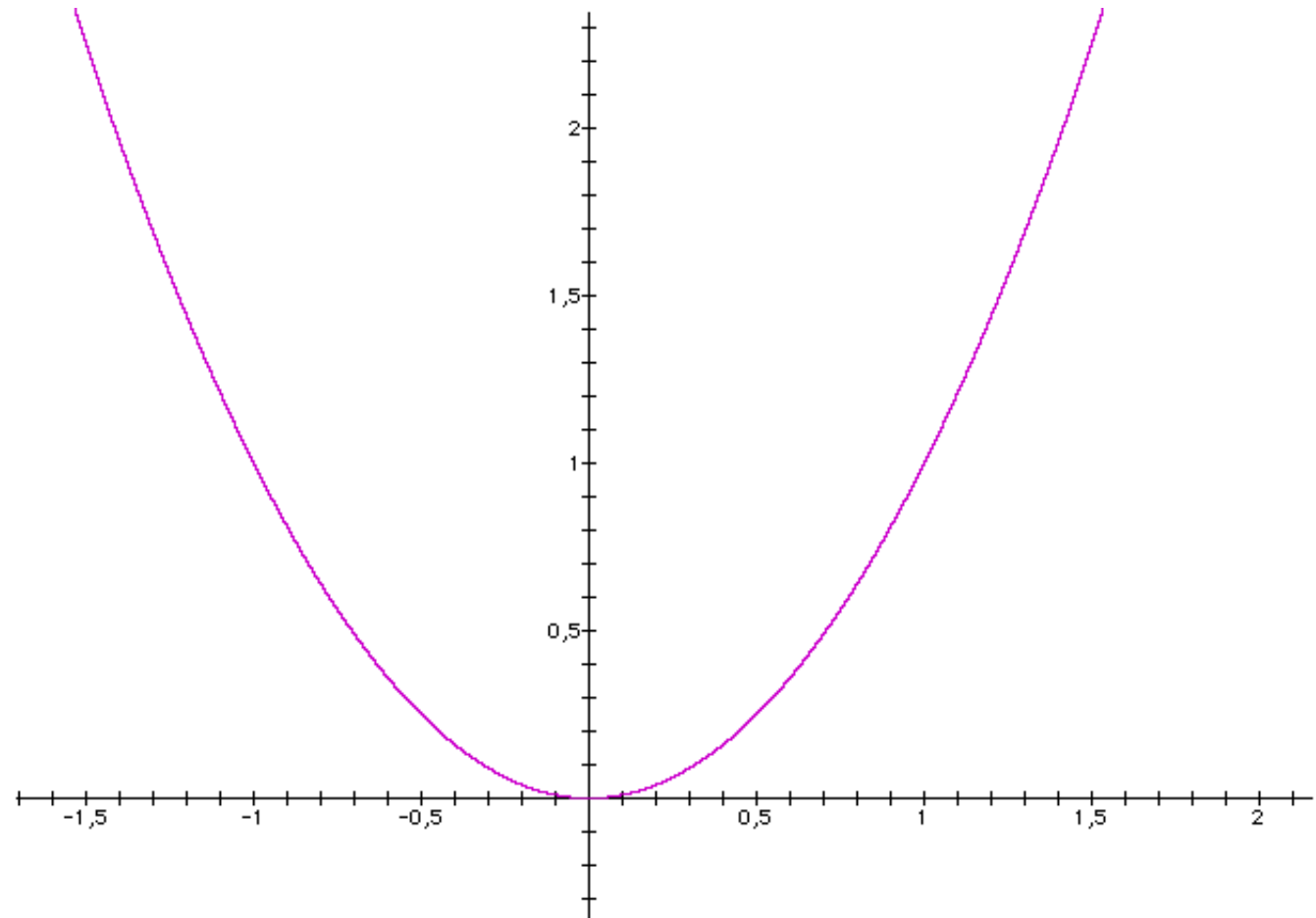
Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

#### In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - Block
  - Alternative
  - **Iteration**

# Beispiel: do-while (2) II

## Quadratwurzel-Berechnung mittels Intervallschachtelung



EINI LogWing /  
WiMa

### Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

### In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - Block
  - Alternative
  - **Iteration**

# Beispiel: do-while (2) III

- ▶ Quadratwurzel-Berechnung mittels Intervallschachtelung
- ▶ Rückführung der Berechnung auf Quadrierung
- ▶ Start: Intervall  $[0, x+1]$ ,
  - ▶  $uG = 0$ ;
  - ▶  $oG = 3$ ;
  - ▶ 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  $oG - uG < \varepsilon$

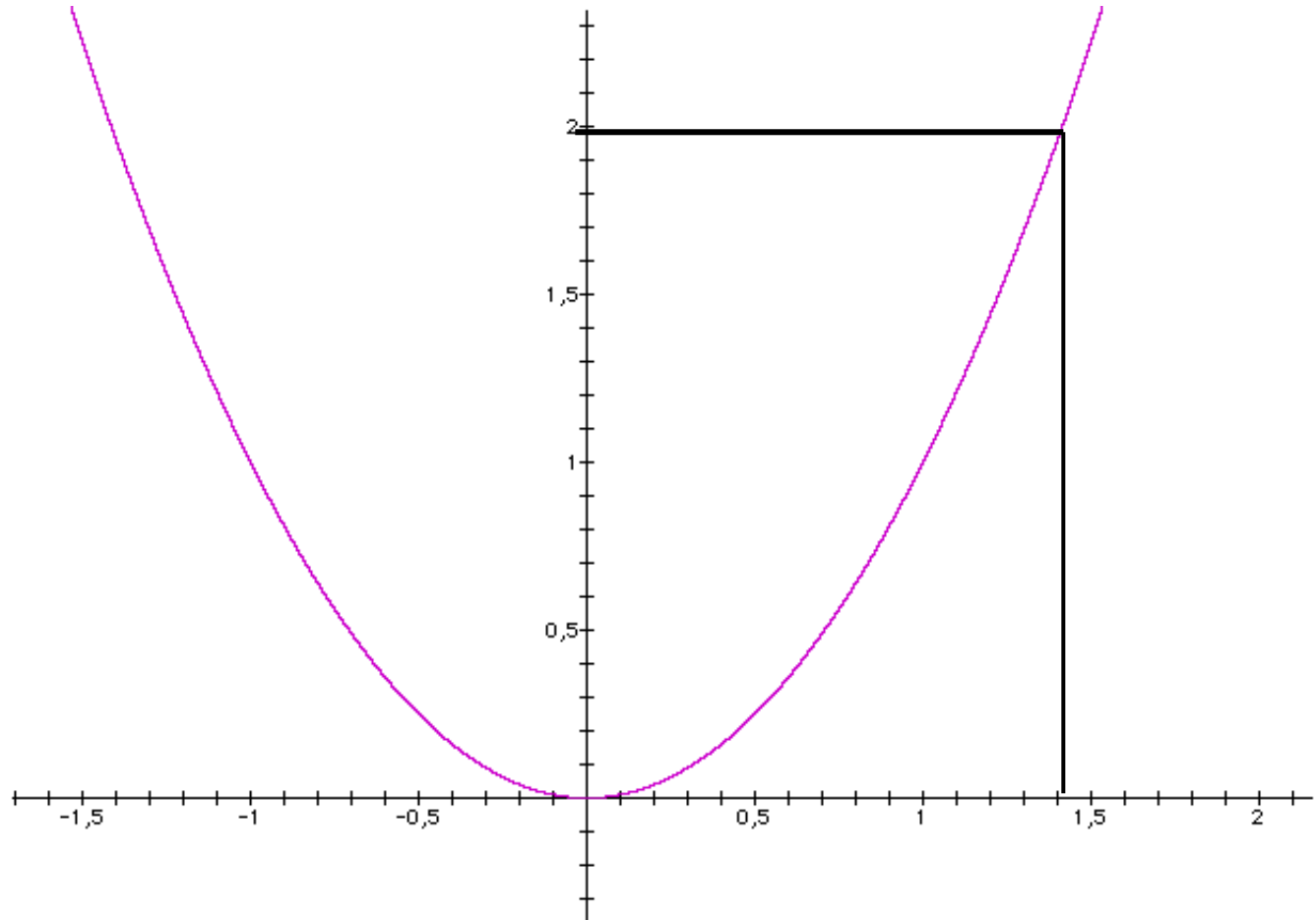
## Kapitel 3

### In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - Block
  - Alternative
  - **Iteration**

# Beispiel: do-while (2) IV

## Quadratwurzel-Berechnung mittels Intervallschachtelung



uG m oG

EINI LogWing /  
WiMa

### Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

### In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
- Sequenz
- Block
- Alternative
- **Iteration**

# Beispiel: do-while (2) V

```
double x = 2.0,  
       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 );
```

- Prolog
- **Kontrollstrukturen**
- Sequenz
- Block
- Alternative
- **Iteration**



# Kontrollstrukturen: Iteration

Artikel im EINI-Wiki:

- **Schleife**
- **Kopfgesteuerte Schleife**
- **Fußgesteuerte Schleife**

## Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

## In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - Block
  - Alternative
  - **Iteration**



# Zwischenstand

- ✓ Variablen
- ✓ Zuweisungen
- ✓ (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:
    - ✓ `while, do-while`
    - `for`
- ▶ Verfeinerung
  - ▶ Unterprogramme, Prozeduren, Funktionen
  - ▶ Blockstrukturierung
- ▶ Rekursion

EINI LogWing /  
WiMa

## Kapitel 3

Basiskonstrukte  
imperativer und  
objektorientierter  
Programmiersprachen

### In diesem Kapitel:

- Prolog
- **Kontrollstrukturen**
  - Sequenz
  - Block
  - Alternative
  - **Iteration**



**Vielen Dank für Ihre Aufmerksamkeit!**

## Nächste Termine

- ▶ Nächste Vorlesung – WiMa 16.11.2023, 08:15
- ▶ Nächste Vorlesung – LogWing 17.11.2023, 08:15