

# ESOP - Verzweigungen & Schleifen

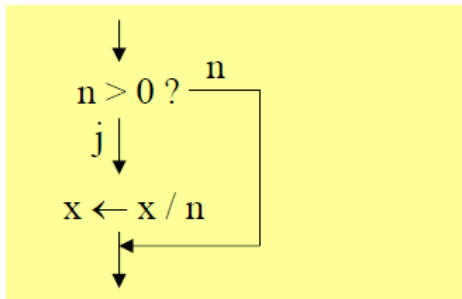
Assoc. Prof. Dr. Mathias Lux  
ITEC / AAU

# Agenda



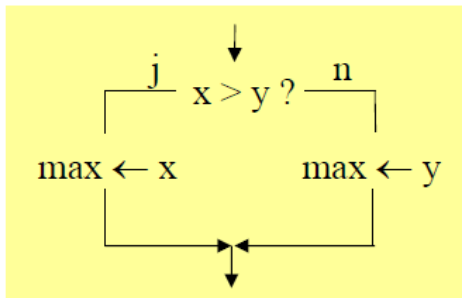
- Verzweigungen
  - If – Else, Switch
- Schleifen
  - While, Do-While, For

# If-Anweisung



```
if (n > 0) x = x / n;
```

ohne else-Zweig



```
if (x > y)
    max = x;
else
    max = y;
```

mit else-Zweig

Syntax

```
IfStatement = "if" "(" Expression ")" Statement ["else" Statement].
```

Expression: Ausdruck,  
der zu einem einzelnen  
Wert evaluiert!

# Anweisungsblöcke



Wenn if-Zweig oder else-Zweig aus mehr als 1 Anweisung bestehen, müssen sie durch { ... } geklammert werden.

Statement = Assignment | IfStatement | Block | ... .  
Block = "{ {**Statement**} }".

# Anweisungsblöcke



- Beispiel

```
if (x < 0) {  
    negNumbers++;  
    System.out.print(-x);  
} else {  
    posNumbers++;  
    System.out.print(x);  
}
```

Einrückung

Best Practice:  
{...} auch bei einzelnen  
Statements

# Einrückungen



- Erhöhen die Lesbarkeit
  - machen Programmstruktur besser sichtbar
- Einrückungstiefe
  - 1 Tabulator oder 2 Leerzeichen
- Kurze If-Anw. auch in einer Zeile:
  - `if (n != 0) x = x / n;`
  - `if (x > y) max = x; else max = y;`

# Dangling Else



```
if (a > b)
    if (a != 0) max = a;
else
    max = b;
```

```
if (a > b)
    if (a != 0) max = a; else max = b;
```

- Zu welchem if gehört das else?
- In Java: else gehört immer zum unmittelbar vorausgegangenen if
- Alternative: Anweisungsblöcke verwenden!

# Kurze If-Anweisungen



- `(Expression) ? Statement : Statement`

```
int x = 3;  
int y = 4;  
int max = (x < y) ? y : x;
```

```
println(max);
```

Expression: Ausdruck,  
der zu einem einzelnen  
Wert evaluiert!



# Vergleichsoperatoren



- Vergleich zweier Werte
- Liefert *true* oder *false*

	<i>Bedeutung</i>	<i>Beispiel</i>
<code>==</code>	gleich	<code>x == 3</code>
<code>!=</code>	ungleich	<code>x != y</code>
<code>&gt;</code>	größer	<code>4 &gt; 3</code>
<code>&lt;</code>	kleiner	<code>x+1 &lt; 0</code>
<code>&gt;=</code>	größer oder gleich	<code>x &gt;= y</code>
<code>&lt;=</code>	kleiner oder gleich	<code>x &lt;= y</code>

# Zusammengesetzte Vergleiche

## Boolesche Operatoren



### && Und-Verknüpfung

x	y	x && y
true	true	true
true	false	false
false	true	false
false	false	false

### || Oder-Verknüpfung

x	y	x    y
true	true	true
true	false	true
false	true	true
false	false	false

### ! Nicht-Verknüpfung

x	!x
true	false
false	true

- Beispiel

– `if (a >= 0 && a <= 10 || a >= 100 && a <= 110) b = a;`

# Boolesche Operatoren

## Bindung



- ! bindet stärker als && bzw ||
- && bindet stärker als ||
- Klammerung möglich
  - `if (a > 0 && (b==1 || b==7)) ...`

# Datentyp boolean



- Datentyp (wie z.B. `int`)
  - mit den beiden Werten *true* und *false*
- Beispiel

```
boolean p, q;  
p = false;  
q = x > 0;  
p = p || q && x < 10;
```

# DeMorgan'sche Regeln



- $\neg (a \ \&\& \ b) \Leftrightarrow \neg a \ || \ \neg b$
- $\neg (a \ || \ b) \Leftrightarrow \neg a \ \&\& \ \neg b$

```
if (x >= 0 && x < 10) {  
    ...  
} else { // ! (x >= 0 && x < 10)  
    ...  
}
```

$\Rightarrow \neg (x \geq 0) \ || \ \neg (x < 10)$

$\Rightarrow x < 0 \ || \ x \geq 10$

# Beispiele boolean & if

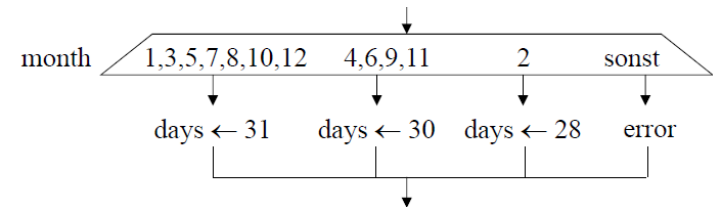


- Expression: Bedingung wird auf true oder false ausgewertet
  - `if (true)`
  - `if (!true)`
  - `if ((x >=1) == true)`

# Switch-Anweisung



- Mehrfachverzweigung
- In Java



```
switch (month) {  
    case 1: case 3: case 5: case 7: case 8: case 10: case 12:  
        days = 31; break;  
    case 4: case 6: case 9: case 11:  
        days= 30; break;  
    case 2:  
        days = 28; break;  
    default:  
        System.out.println("error");  
}
```

# Switch-Anweisung



- **Bedingungen**

- Ausdruck ganzzahlig, char oder String
- Case-Marken sind Konstante
- Case-Marken Typ muss zu Ausdruck passen
- Case-Marken müssen verschieden sein

- **Break-Anweisung**

- Spring ans Ende der Switch-Anweisung
- Fehlt break, wird alles danach ausgeführt  
=> häufiger Fehler!

## Switch-Ausdruck

```
switch (month) {  
    case 1: case 3: case 5: case 7: case 8: case 10: case 12:  
        days = 31; break;  
    case 4: case 6: case 9: case 11:  
        days = 30; break;  
    case 2:  
        days = 28; break;  
    default:  
        System.out.println("error");  
}
```



# Switch-Syntax

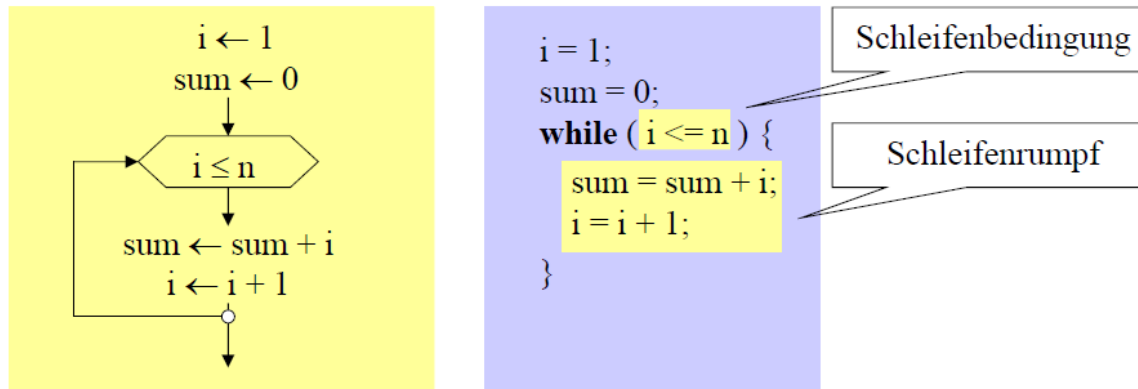


```
Statement = Assignment | IfStatement | SwitchStatement | ... | Block.  
SwitchStatement = "switch" "(" Expression ")" "{" {LabelSeq StatementSeq} "}".  
LabelSeq = Label {Label}.  
StatementSeq = Statement {Statement}.  
Label = "case" ConstantExpression ":" | "default" ":".
```

# While-Schleife

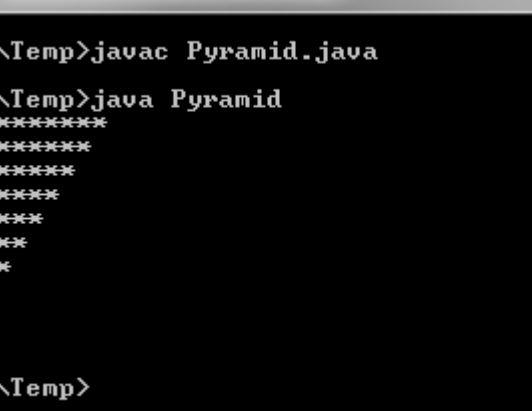


- Führt eine Anweisungsfolge aus
- Solange eine bestimmte Bedingung gilt



Statement = Assignment | IfStatement | SwitchStatement | WhileStatement | ... | Block.

WhileStatement = **"while"** **"(" Expression ")"** Statement .



```
C:\Windows\system32\cmd.exe
E:\Temp>javac Pyramid.java
E:\Temp>java Pyramid
*****
*****
*****
*****
*****
*****
***
**
*

E:\Temp>
```

# Termination

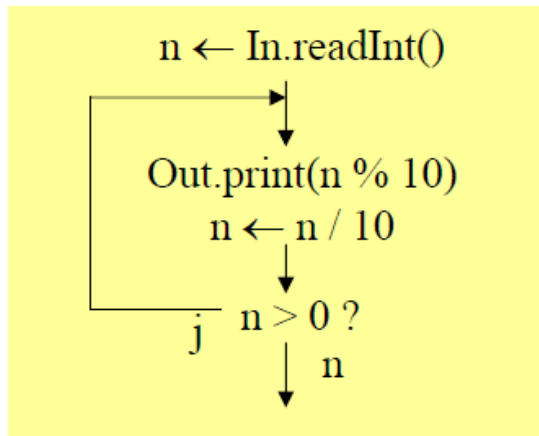


- Schleifen sollten ein Ende haben
  - kein `while (true) { ... }`
- Problem: Endlosschleifen
  - Abgefragte Variable wird nicht verändert
  - Abbruchbedingung wird nicht erreicht
    - z.B. `while (x!=0) { x -= 5; }`
- Lösung: Modellierung möglicher Probleme

# Do-While-Schleife



- Abbruchbedingung wird am Ende der Schleife geprüft
- Schleife wird mind. 1x durchlaufen



```
int n = In.readInt();
do {
    Out.print(n % 10);
    n = n / 10;
} while ( n > 0 );
```

*Schreibtischtest*

n	n % 10
<del>123</del>	3
<del>12</del>	2
<del>1</del>	1
0	

Statement = Assignment | IfStatement | WhileStatement |  
DoWhileStatement | ... | Block.

DoWhileStatement = **"do" Statement "while" "(" Expression ")" ";"**.

# Review: While-Schleife



- Mehrfache Ausführung desselben Programmteils
  - Mit eventuellen Änderungen in Variablen

# Beispiele While-Schleife



- Rechner mit Scanner -> Endlosschleife

# For-Schleife (Zählschleife)



- Falls Anzahl im Vorhinein bekannt ist

```
sum = 0;  
for ( i = 1 ; i <= n ; i++ )  
    sum = sum + i;
```

- 1) Initialisierung der Laufvariablen
- 2) Schleifenabbruchbedingung
- 3) Ändern der Laufvariablen

Kurzform für

```
sum = 0;  
i = 1;  
while ( i <= n ) {  
    sum = sum + i;  
    i++;  
}
```



# For-Schleife: Beispiele



for (i = 0; i < n; i++)	i: 0, 1, 2, 3, ..., n-1
for (i = 10; i > 0; i--)	i: 10, 9, 8, 7, ..., 1
for (int i = 0; i <= n; i = i + 1)	i: 0, 1, 2, 3, ..., n
for (int i = 0, j = 0; i < n && j < m; i = i + 1, j = j + 2)	i: 0, 1, 2, 3, ... j: 0, 2, 4, 6, ...
for (;;) ...	Endlosschleife

# For-Schleife: Definition



`ForStatement = "for" "(" [ForInit] ";" [Expression] ";"  
[ForUpdate] ")" Statement.`

`ForInit = Assignment {",", " Assignment} | Type VarDecl {", "  
VarDecl}.`

`ForUpdate = Assignment {",", " Assignment}.`

# For-Schleife: Beispiel



```
for (int i = 0; i < 10; i++) {  
    System.out.println(i);  
}  
int sum = 0;  
for (int i = 0; i < 10; i++) {  
    sum += i;  
}  
System.out.println("sum = " + sum);
```

# For-Schleife: Verschachtelt



```
class PrintMulTab {  
    public static void main (String[] arg) {  
        int n = 5;  
        for (int i = 1; i <= n; i++) {  
            for (int j = 1; j <= n; j++) {  
                System.out.print(i * j + "\t");  
            }  
            System.out.println();  
        }  
    }  
}
```

```
C:\Windows\system32\cmd.exe  
E:\Temp>javac PrintMulTab.java  
E:\Temp>java PrintMulTab  
1      2      3      4      5  
2      4      6      8      10  
3      6      9      12     15  
4      8      12     16     20  
5      10     15     20     25  
E:\Temp>_
```

# Schleifenabbrüche



- Abbruch mit Keyword *break*

```
while (In.done()) {  
    sum = sum + x;  
    if (sum > 1000) {  
        Out.println("zu gross");  
        break;  
    }  
    x = In.nextNumber();  
}
```

- Besser als Schleifenbedingung ...

```
while (In.done() && sum < 1000) {  
    sum = sum + x;  
    x = In.nextNumber();  
}  
if (sum > 1000)  
    Out.println("zu gross");
```

# Abbruch äußerer Schleifen



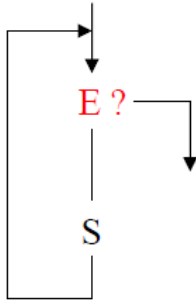
```
outer: // Marke!
for (;;) { // Endlosschleife!
    for (;;) {
        ...
        if (...) break; // verlässt innere Schleife
        else break outer; // verlässt äußere Schleife
        ...
    }
}
```

# Schleifenabbrüche



- Wann ist ein Schleifenabbruch mit `break` typischerweise vertretbar?
  - bei Abbruch wegen Fehlern (Performance!)
  - bei mehreren Aussprüngen an verschiedenen Stellen der Schleife
  - bei echten Endlosschleifen (z.B. in Echtzeitsystemen)

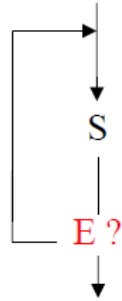
# Vergleich der Schleifenarten



**Abweisschleife**

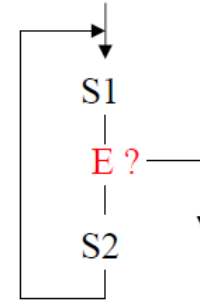
```
while (E)  
  S
```

```
for (I; E; U)  
  S
```



**Durchlaufschleife**

```
do  
  S  
while (E)
```



**allgemeine Schleife**

```
for (;;) {  
  S1;  
  if (E) break;  
  S2;  
}
```



# Welche Schleife Wann?



- Auswahl nach “Convenience”
- Auswahl nach Performance
  - (s.u. für Javascript, <http://jsperf.com/fun-with-for-loops/8>)

## Test runner

Done. Ready to run again.

Run again

Testing in Chrome 37.0.2062.124 32-bit on Windows Server 2008 R2 / 7 64-bit		
	Test	Ops/sec
FOR standard	<pre>for (var i; i &lt; a.length; i++) {   n++; }</pre>	329,591,795 ±0.23% fastest
FOR optimized	<pre>for (var i, imax = a.length; i &lt; imax; i++) {   n++; }</pre>	329,708,498 ±0.43% 0.16% slower
While Counting Down	<pre>var i = a.length + 1; while(--i) {   n++; }</pre>	29,620,863 ±19.14% 92% slower

# Beispiele for-Schleife



- Compute average of n numbers ...