

Programmierkurs für Erstsemester

Arrays

- Arrays erstellen
- Arrayzugriff

Schleifen

- While-Schleifen
- Do-While-Schleifen
- Endlosschleifen
- For-Schleifen
- Break und Continue

Debugging

Arrays

- Arrays erstellen
- Arrayzugriff

Schleifen

- While-Schleifen
- Do-While-Schleifen
- Endlosschleifen
- For-Schleifen
- Break und Continue

Debugging

Arrays

Arrays erstellen

Arrayzugriff

Schleifen

While-Schleifen

Do-While-Schleifen

Endlosschleifen

For-Schleifen

Break und Continue

Debugging

- ▶ Ein Array fasst mehrere Variablen des gleichen Typs zusammen.

Beispiel: Ein Array von Integern enthält Ganzzahlen:

{ 4, 8, 15, 16, 23, 42 }

- ▶ Alle Werte müssen vom gleichen Typ sein.

Falsch: { 3, 18, 3.14, 'r' }

Arrays

Arrays erstellen

Arrayzugriff

Schleifen

While-Schleifen

Do-While-Schleifen

Endlosschleifen

For-Schleifen

Break und Continue

Debugging

- ▶ Um ein Array vom Typ *type* zu deklarieren:

```
type [] arrayName;
```

- ▶ Um ein Array vom Typ *type* und Größe *n* zu deklarieren und initialisieren:

```
type [] arrayName = new type[n];
```

Das Array wird dann mit Standardwerten gefüllt (bei Zahlen mit 0).

Arrays

Arrays erstellen

Arrayzugriff

Schleifen

While-Schleifen

Do-While-Schleifen

Endlosschleifen

For-Schleifen

Break und Continue

Debugging

- ▶ Um ein Array mit Werten zu initialisieren:

```
type [] arrayName = new type [] { w1, w2 };
```

Die Größe eines Arrays kann nachträglich nicht mehr geändert werden.

Zum Vergrößern oder Verkleinern muss ein neues Array angelegt werden.

Alternativen zu Arrays kommen in der Vorlesung.

- ▶ Zugriff auf das i -te Arrayelement:

```
arrayName [ i ]
```

Achtung: Der Index geht von 0 bis $n - 1$!

- ▶ Die Größe des Arrays (n) kann mit

```
arrayName . length
```

bestimmt werden.

Beispiele:

```
System.out.println ( arrayName [ 3 ] );  
arrayName [ arrayName . length - 1 ] = 5;
```

Arrays

Arrays erstellen

Arrayzugriff

Schleifen

While-Schleifen

Do-While-Schleifen

Endlosschleifen

For-Schleifen

Break und Continue

Debugging

Arrays

Arrays erstellen

Arrayzugriff

Schleifen

While-Schleifen

Do-While-Schleifen

Endlosschleifen

For-Schleifen

Break und Continue

Debugging

- ▶ Schleifen führen einen Programmteil mehrfach aus.
- ▶ Sie werden so lange ausgeführt, wie ihre Schleifenbedingung wahr ist (bzw. bis ihre Abbruchbedingung erfüllt ist).
- ▶ Es gibt verschiedene Schleifentypen, die aber alle untereinander austauschbar sind.

Syntax:

```
while (Bedingung) {  
    Anweisung1;  
    Anweisung2;  
    // ...  
}
```

Arrays

Arrays erstellen

Arrayzugriff

Schleifen

While-Schleifen

Do-While-Schleifen

Endlosschleifen

For-Schleifen

Break und Continue

Debugging

Arrays

Arrays erstellen

Arrayzugriff

Schleifen

While-Schleifen

Do-While-Schleifen

Endlosschleifen

For-Schleifen

Break und Continue

Debugging

Beispiel:

```
int zaehler = 0;
while (zaehler < 10) {
    System.out.println("Hallo □ Welt");
    zaehler++;
}
```

Arrays

Arrays erstellen

Arrayzugriff

Schleifen

While-Schleifen

Do-While-Schleifen

Endlosschleifen

For-Schleifen

Break und Continue

Debugging

Syntax:

```
do {  
    Anweisung1;  
    Anweisung2;  
    // ...  
} while (Bedingung);
```

Anders als While-Schleifen wird eine Do-While-Schleife immer mindestens einmal durchlaufen.

Do-While-Schleifen

Beispiel:

```
int zaehler = 10;
while (zaehler < 10) {
    System.out.println("Hallo_Welt");
    zaehler++;
}
```

und

```
int zaehler = 10;
do {
    System.out.println("Hallo_Welt");
    zaehler++;
} while (zaehler < 10);
```

Arrays

Arrays erstellen

Arrayzugriff

Schleifen

While-Schleifen

Do-While-Schleifen

Endlosschleifen

For-Schleifen

Break und Continue

Debugging

Arrays

Arrays erstellen
Arrayzugriff

Schleifen

While-Schleifen
Do-While-Schleifen
Endlosschleifen
For-Schleifen
Break und Continue

Debugging

Vorsicht vor Endlosschleifen!

```
int i = 10;
while (i > 0) {
    System.out.println("Hilfe!");
    i = i/2 + 1;
}
```

Arrays

Arrays erstellen
Arrayzugriff

Schleifen

While-Schleifen
Do-While-Schleifen
Endlosschleifen

For-Schleifen

Break und Continue

Debugging

Syntax:

```
for(Initialisierung; Bedingung; Schritt) {  
    Anweisung1;  
    Anweisung2;  
    // ...  
}
```

Arrays

Arrays erstellen
Arrayzugriff

Schleifen

While-Schleifen
Do-While-Schleifen
Endlosschleifen
For-Schleifen
Break und Continue

Debugging

Beispiel:

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

Entspricht dieser While-Schleife:

```
int i = 0;  
while (i < 10) {  
    System.out.println("Hallo_Welt!");  
    i++;  
}
```

Arrays

Arrays erstellen
Arrayzugriff

Schleifen

While-Schleifen
Do-While-Schleifen
Endlosschleifen
For-Schleifen
Break und Continue

Debugging

- ▶ Zuerst wird die Initialisierungs-Anweisung ausgeführt. Meistens handelt es sich dabei um Laufvariablen-Deklaration und Initialisierung.
- ▶ Dann wird die Bedingung geprüft.
 - ▶ Ist die Bedingung falsch, wird die Schleife verlassen.
 - ▶ Ist die Bedingung wahr, werden die Anweisungen im Schleifenkörper ausgeführt.
- ▶ Anschließend wird die Schritt-Anweisung ausgeführt. Meistens wird die Laufvariable inkrementiert.
- ▶ Danach wird wieder die Bedingung geprüft.

Arrays

Arrays erstellen
Arrayzugriff

Schleifen

While-Schleifen
Do-While-Schleifen
Endlosschleifen
For-Schleifen
Break und Continue

Debugging

For-Schleifen werden häufig im Zusammenhang mit Arrays eingesetzt.

Beispiel:

```
char [] abc = new char [] { 'a', 'b', 'c' };  
for (int i = 0; i < abc.length; i++) {  
    System.out.println(abc[i]);  
}
```

break und *continue*

- ▶ *break* und *continue* sind alternative Möglichkeiten, eine Schleife zu verlassen.
- ▶ *break* verlässt die innerste Schleife sofort.
- ▶ *continue* beendet den aktuellen Schleifendurchlauf, aber nicht die ganze Schleife (es wird mit der Überprüfung der Bedingung weitergemacht).

Arrays

Arrays erstellen

Arrayzugriff

Schleifen

While-Schleifen

Do-While-Schleifen

Endlosschleifen

For-Schleifen

Break und Continue

Debugging

break und continue

Beispiel:

```
int [] werte = new int []
    { 10, 8, 14, 29, 38, 7, 21 };
```

```
for (int i = 0; i < werte.length; i++) {
    if (werte[i] >= 20) {
        System.out.println(werte[i]);
        break;
    }
}
```

```
for (int i = 0; i < werte.length; i++) {
    if (werte[i] >= 20)
        continue;
    System.out.println(werte[i]);
}
```

Arrays

Arrays erstellen
Arrayzugriff

Schleifen

While-Schleifen
Do-While-Schleifen
Endlosschleifen
For-Schleifen
Break und Continue

Debugging

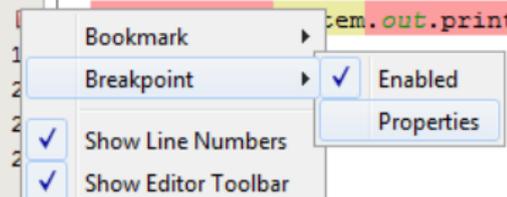
Beim Debugging von Schleifen sind Conditional Breakpoints nützlich.

- ▶ Dazu erst wie gewohnt einen Breakpoint setzen.

```
16 public static void main(String[] args) {  
17     Line Breakpoint (int i = 0; i < 10; i++) {  
18         System.out.println("Durchlauf " + (i + 1));  
19     }
```

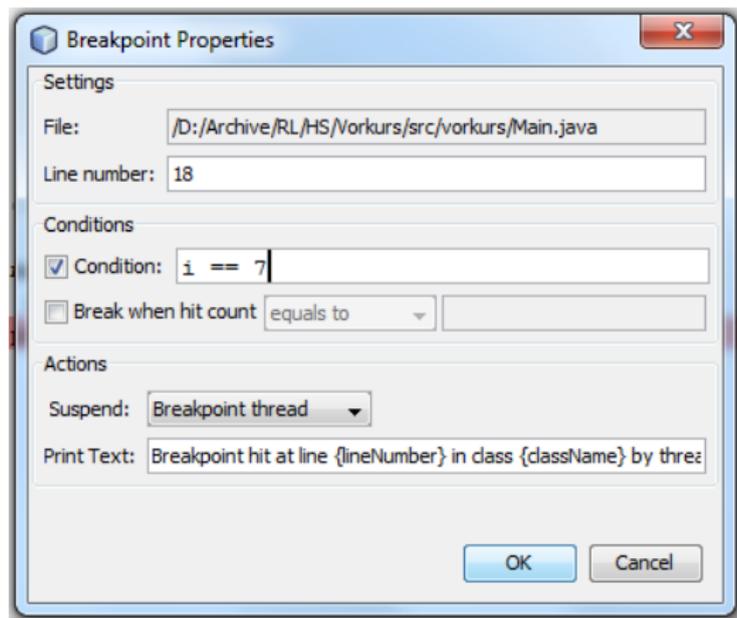
- ▶ Dann per Rechtsklick die Eigenschaften des Breakpoints öffnen.

```
17     for (int i = 0; i < 10; i++) {  
18         System.out.println("Durchlauf " + (i + 1));  
19     }
```



Debugging

Über *Condition* kann z.B. die Laufvariable auf einen bestimmten Wert überprüft werden. Nur wenn die Bedingung wahr ist, wird am Breakpoint angehalten.



Arrays

Arrays erstellen
Arrayzugriff

Schleifen

While-Schleifen
Do-While-Schleifen
Endlosschleifen
For-Schleifen
Break und Continue

Debugging

Debugging

Auch der *HitCount* kann nützlich sein. Damit kann man z.B. einstellen, dass erst ab dem 10. Durchlauf am Breakpoint gestoppt werden soll.

Der *HitCount* ist besonders nützlich, wenn die Schleife keine Laufvariable hat.

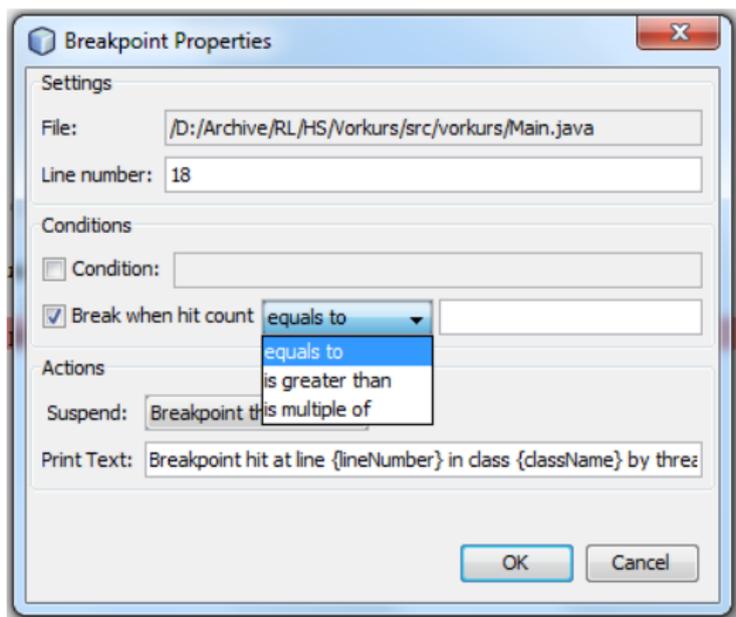
Arrays

Arrays erstellen
Arrayzugriff

Schleifen

While-Schleifen
Do-While-Schleifen
Endlosschleifen
For-Schleifen
Break und Continue

Debugging



Arrays

- Arrays erstellen
- Arrayzugriff

Schleifen

- While-Schleifen
- Do-While-Schleifen
- Endlosschleifen
- For-Schleifen
- Break und Continue

Debugging

Viel Spaß bei den Übungen.