

Willkommen

Letztes Mal

- Verzweigung
- Schleifen
- Tests

Letztes Mal

- Verzweigung
- Schleifen
- Tests

Leider nicht!

Letztes Mal

- Verzweigung
- Schleifen
- Tests

Leider nicht!

Dieses Mal

- Verzweigung
- Schleifen
- Tests
- Arrays

Wer übt auch außerhalb des Vorlesungs- und Übungsbetriebs?

Wer übt auch außerhalb des Vorlesungs- und Übungsbetriebs?

⇒ **Ihr sollt und müsst ggf. den Stoff nacharbeiten!**

Wer übt auch außerhalb des Vorlesungs- und Übungsbetriebs?

⇒ **Ihr sollt und müsst ggf. den Stoff nacharbeiten!**

Ihr wollt doch auch nicht, dass der Stoff für das Übungsblatt erst **nach** der Abgabe des Übungsblattes besprochen wird!

Wer übt auch außerhalb des Vorlesungs- und Übungsbetriebs?

⇒ **Ihr sollt und müsst ggf. den Stoff nacharbeiten!**

Ihr wollt doch auch nicht, dass der Stoff für das Übungsblatt erst **nach** der Abgabe des Übungsblattes besprochen wird!

Vertiefungstutorium

Wann? Dienstags, 9:45-11:15

Wo? morgen: 50.34, Raum 148

Auch dort sollt ihr die Aufgaben möglichst selbstständig lösen


```
if (A) {B} else {C}
```

Wenn A (Boolscher Ausdruck) zu true evaluiert, wird der Codeblock B ausgeführt, andernfalls wird der Codeblock C ausgeführt

Hinweis: Das else {C} ist optional

```
1 public static double max(double a, double b) {  
2     double max;  
3  
4     if (a > b) {  
5         max = a;  
6     } else {  
7         max = b;  
8     }  
9  
10    return max;  
11 }
```

Noch mehr Verzweigung

Kaskadierung

```
1  if (A1) {  
2      ...  
3  } else if (A2) {  
4      ...  
5  } else {  
6      ...  
7  }
```

Verschachtelung

```
1  if (A) {  
2      if (A1) {  
3          ...  
4      } else {  
5          ...  
6      }  
7  } else {  
8      ...  
9  }
```

Schleifen – Warum?

```
1  /* Summiere von 1 bis 4 */
2  int summe = 0;
3  summe = summe + 1;
4  summe = summe + 2;
5  summe = summe + 3;
6  summe = summe + 4;
```

Schleife statt copy-and-paste

```
1  /* Summiere von 1 bis 4 */
2  int summe = 0;
3  for (int i = 1; i <= 4; i = i + 1) {
4      summe = summe + i;
5  }
```

Flexibel/Dynamisch zur Laufzeit

```
1  /* Summiere von 1 bis n */
2  int summe = 0;
3  for (int i = 1; i <= n; i = i + 1) {
4      summe = summe + i;
5  }
```

```
for (I; C; A) {B}
```

Bei der `for`-Schleife (auch Zählschleife genannt) wird zu Beginn `I` ausgeführt

Vor jedem Schleifendurchlauf wird der Boolesche Ausdruck `C` evaluiert und nur wenn er `true` ist, wird die Schleife durchlaufen, ansonsten geht die Programmausführung unterhalb der Schleife weiter

Bei jedem Schleifendurchlauf wird zuerst `B`, dann `A` ausgeführt und anschließend wieder `C` evaluiert um zu entscheiden ob die Schleife nochmal durchlaufen werden soll

Hinweis: Normalerweise steht die Anzahl der Durchläufe schon vor dem ersten Schleifendurchlauf fest

Schleifen – while

```
while (C) {B}
```

Die `while` Schleife wird durchlaufen (d.h. B ausgeführt) solange der Boolesche Ausdruck C zu `true` evaluiert

Hinweis: Wenn C von einer Variable abhängt die in B bei jedem Durchlauf inkrementiert/dekrementiert wird, könnte eine `for`-Schleife besser sein

Beispiel

```
1  /* success could be already false due to previous errors */
2  while (success) {
3      token = nextToken();
4      success = token.isValid();
5  }
```

Schleifen – do-while

```
do {B} while (C)
```

Sehr ähnlich zur `while`-Schleife, nur das bei der `do-while` Schleife B mindestens einmal ausgeführt wird

Beispiel

```
1  /* do the first run regardless of whether success is true or false */
2  do {
3      token = nextToken();
4      success = token.isValid();
5  } while (success);
```

Primzahlen: Aufgabe 1, 2

Hinweis: Formal sind alle Schleifen äquivalent

Warum überhaupt testen?

Warum überhaupt testen?

„Ein Feature das nicht getestet wurde existiert auch nicht!“

Erst das Testen macht das spezifizierte Verhalten von Methoden und Klassen verlässlich!

Testen (1)

Warum überhaupt testen?

„Ein Feature das nicht getestet wurde existiert auch nicht!“

Erst das Testen macht das spezifizierte Verhalten von Methoden und Klassen verlässlich!

Was testen?

- **Normalfall** : Das Verhalten bei "richtigen" Eingabedaten
- **Randfälle** : Übergangsbereiche (Normalfall ↔ Fehlerfall)
- **Spezialfälle** : Beispiel: 0!
- **Fehlerfall** : Das Verhalten bei "falschen" Eingabedaten

Manuelles Testen

Ausgabe von Testwerten und Vergleichen mit erwartetem Ergebniss

- mühsam
- fehleranfällig

Automatisches Testen

Programme überprüfen die Testwerte mit erwartetem Ergebniss

- + ohne Probleme nach jeder Änderung durchführbar
- manche Sachen sind schwierig durch Programme zu überprüfen

Was ist ein Array?

Was ist ein Array?

- Ein Objekt
- Eine Ansammlung von Objekten/Werten des gleichen Typs

Hinweis: Arrays können also insbesondere selbst wieder Arrays enthalten

Arrays – Verwendung (1)

Deklaration

```
1 int[] primzahlen;  
2 int[][] table;
```

Zuweisung

```
1 primzahlen = new int[42];  
2 table = new int[2][2];
```

Initialisierung

```
1 String[] fehlermeldungen = {"Datei existiert nicht", "Datei ist kaputt"};
```

Arrays – Verwendung (2)

Zugriff auf i-tes Element

```
1 primzahlen[0] = 2;  
2 table[0][0] = 1;
```

Hinweis: Ein Array a mit n Elementen, enthält nur die Elemente $a[0], \dots, a[n-1]$, **nicht** $a[n]$

Länge ermitteln

```
1 int length = primzahlen.length;
```

Arrays und Schleifen

TODO

- ① Einreichen einer Lösung für das 3. Übungsblatt im Praktomat bis
6.12.2010, 13:00
- ② Anmelden für den Übungsschein auf <https://studium.kit.edu/> bis
31.3.2011

Vielen Dank für die Aufmerksamkeit!

...und viel Spaß beim Programmieren :)