

# Einführung in die Programmierung für NF

Übung 01

23.10.2013

---

# Inhalt

- Übungen und Übungsblätter
- Uniworx und Abgabeformate
- Plagiarismus
- Algorithmen und Pseudocode
- Variablen und Methoden
- Boolesche Operatoren

# Übungen

- Aktuelle Informationen immer auf [www.medien.ifi.lmu.de/eipnf](http://www.medien.ifi.lmu.de/eipnf)
- Übungen sind eine wichtige Vorbereitung für die Klausur
  - > Teilnahme wird dringend empfohlen
- Übungsblätter geben einen Notenbonus für die Klausur

# Übungsblätter

- Erscheinen immer montags
- Abgabe spätestens bis übernächsten Mittwoch auf Uniworx
- Korrektur ebenfalls auf Uniworx
- Pro Blatt 20 Punkte
- Maximaler Klausurbonus: 10%  
(wenn alle Übungsblätter zu 100% bestanden)

# Uniworx und Abgabeformate

- Auf Uniworx für Vorlesung und Übung anmelden: <https://uniworx.ifi.lmu.de>
- Lösungen **NUR im gewünschten Format**
- Lösungsdateien in zip-Archiv packen
- zip-Archiv auf Uniworx hochladen
- Achtung: Eine neue Abgabe überschreibt die alte Abgabe, es kann nur ein einzelnes zip-Archiv als Abgabe aktiviert werden

# Plagiarismus

- Abschreiben
  - von Kommilitonen
  - von anderen Quellen (z.B. Wikipedia)ist in **keiner Weise** erlaubt!
- Wenn Sie erwischt werden wird das ganze Blatt mit 0 Punkten bewertet
- Im wiederholten Fall können alle Blätter mit 0 Punkten bewertet werden

# Plagiarismus

- Abschreiben
  - von Kommilitonen
  - von anderen Quellen (z.B. Wikipedia)ist in **keiner Weise** erlaubt!
- Wenn Sie erwischt werden wird das ganze Blatt mit 0 Punkten bewertet
- Im wiederholten Fall können alle Blätter mit 0 Punkten bewertet werden

# Algorithmen

- Algorithmen sind Handlungsvorschriften (Anleitungen) zur Lösung eines Problems
- Sie bestehen aus einzelnen Schritten (Anweisungen)
- Diese sind eindeutig und ausführbar
- Algorithmen treten ständig im Alltag auf



# Algorithmen

- Wir nutzen Algorithmen, um einem Computer zu sagen, was er tun soll
- Dazu gibt es einige Regeln, die beachtet werden müssen:
  - Zu einem Zeitpunkt wird nur ein Schritt ausgeführt
  - Jeder Schritt wird genau einmal ausgeführt
  - Alle Schritte werden ausgeführt
  - Die Schritte werden nacheinander ausgeführt

# Algorithmen

- Computer sind „dumm“, d.h. sie tun nur, was ihnen gesagt wird
- Algorithmen für Computer können aber extrem komplex sein
- Darum gibt es einige Bestandteile, die die Abfolge von Anweisungen erleichtern und strukturieren

# Fallunterscheidungen

- Fallunterscheidungen

```
wenn  Bedingung
dann  Anweisung 1
sonst Anweisung 2
```

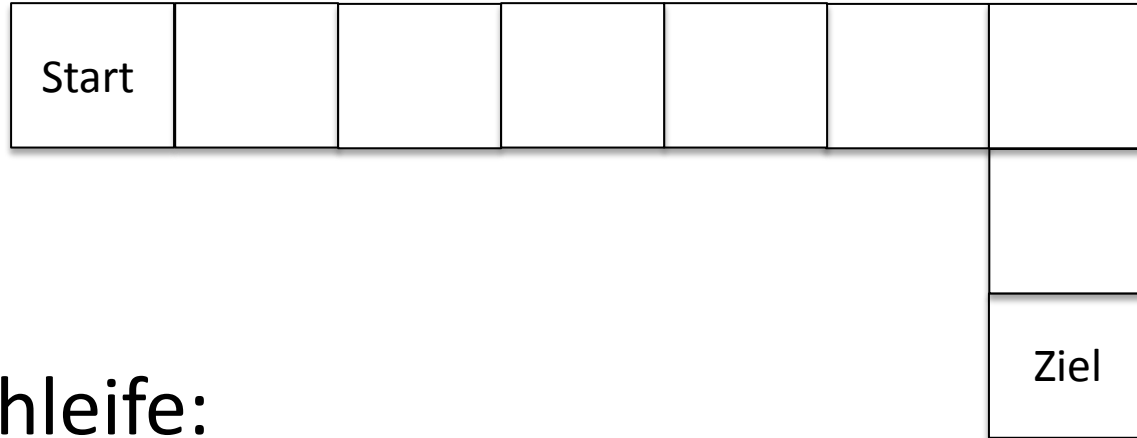
- In Java

```
if (bedingung) {
    anweisung1;
} else {
    anweisung2;
}
```

# Iterationen / Schleifen

- Iterationen, bzw. Schleifen
- Es gibt zwei Arten von Schleifen:
  - **while-Schleife**  
Anzahl der Wiederholungen unbekannt,  
Abbruchbedingung wird mit der Schleife erreicht
  - **for-Schleife**  
Anzahl der Wiederholungen fix festgelegt

# Iterationen / Schleifen



- for-Schleife:  
gehe 6 Felder, drehe dich nach rechts, gehe 2 Felder
- while-Schleife:  
gehe bis zum Ende, drehe dich nach rechts, gehe bis zum Ende

# Iterationen / Schleifen

- Beispiel while-Schleife

```
int x = 0;
while(x < 5) {
    x = x + 1;
}
```

- läuft solange, bis x den Wert 5 erreicht
- Der Wert von x wird immer am Anfang überprüft, d.h. x hat am Ende den Wert 5

# Iterationen / Schleifen

- Beispiel for-Schleife

```
int x = 0;
for(int i=0; i<5; i++) {
    x = x + 1;
}
```

Startwert

Abbruchbedingung

Schlussanweisung

- läuft genau 5 mal durch, x hat am Ende den Wert 5
- i wird immer am Ende eines Durchlaufs erhöht, im ersten Durchlauf hat i den Wert 0

# Methoden / Funktionen

- Oft müssen gleiche Anweisungen an verschiedenen Stellen im Programm aufgerufen werden
- Um sich nicht zu wiederholen, können solche Anweisungen in Methoden ausgelagert werden
- An der Stelle im Code, an der die Anweisungen stehen würden, steht dann nur der Methodenaufruf
- Methoden können auch mit Paramtern aufgerufen werden



# Methoden / Funktionen

- Beispiel: längere Rechnerei mit einer Zahl  
„ $x + 5 - 3 + x/2$ “ möchte ich auf 5, 7 und 8 anwenden

- ohne Methode:

```
5 + 5 - 3 + 5/2;
```

```
7 + 5 - 3 + 7/2;
```

```
8 + 5 - 3 + 8/2;
```

- mit Funktion

```
function rechnerei(int x){  
    return x + 5 - 3 + x/2;  
}
```

```
rechnerei(5);
```

```
rechnerei(7);
```

```
rechnerei(8);
```

# Methoden / Funktionen

- Vorteile von Funktionen:
  - Weniger fehleranfällig
  - Ich muss nicht wissen, was in der Funktion passiert
  - Wenn sich etwas ändert, muss man es nur einmal ändern
  - Deutlich kürzerer und überschaubarer Code
- Wichtiges Prinzip in der Programmierung:  
**DRY = Don't Repeat Yourself**

# Pseudocode

- Jede Programmiersprache hat eine eigene Syntax – der Inhalt ist aber immer gleich
- Bevor man programmiert, überlegt man sich seine Lösungen
- Zur Verständnis können Lösungsansätze in Pseudocode geschrieben werden
- Hier geht es nur um den Inhalt, nicht darum, dass ein Computer den Code ausführen kann

# Variablen

- Variablen dienen im Programm zur Speicherung von Daten, z.B. Zahlen, Buchstaben oder Zeichenketten
- Variablen müssen angelegt, d.h. deklariert werden – erst danach kann ihnen ein Wert zugewiesen werden
- Beides kann auch gleichzeitig geschehen

# Datentypen

- In Java muss der Datentyp von Variablen manuell festgelegt werden

- z.B.     `int`           für einfache Zahlen  
          `double`       für Kommazahlen  
          `string`       für Zeichenketten  
          `boolean`     für Wahrheitswerte

- Im Code:

```
int x = 5;  
double y = 7.3;  
string hallo = "Hallo";  
boolean test = true;
```

boolean kann nur „true“  
oder „false“ sein

# Datentypen

- Wenn nicht gleich ein Wert angegeben ist, können Variablen trotzdem verwendet werden (nicht nur für Zuweisungen)
- Zahlen sind dann automatisch `0`
- Wahrheitswerte sind automatisch `false`
- Strings müssen allerdings initialisiert werden, z.B. `string test = "";`

# Boolsche Operatoren

- Um Bedingungen verschachteln zu können, gibt es die boolschen Operatoren **UND**, **ODER** und **NICHT**
- Mathematik:             $\wedge$              $\vee$              $\neg$
- Programmierung:     $\&\&$              $||$              $!$
  
- Boolsche Operatoren können nur auf boolsche Werte angewendet werden
- Außerdem             $==$              $!=$              $<=$              $>=$              $\wedge$

# Boolsche Operatoren

- Beispiele:

- Prüfe, ob Zahl größer 10:

```
x > 10
```

- Zahl zwischen 10 und 20:

```
x > 10 && x < 20
```

- Zahl zwischen 10 und 20:

```
x > 10 && x < 20
```

- Zahl ungleich 15:

```
x != 15
```

- Zahl zwischen 10 und 20 oder zwischen 30 und 40, aber nicht 15 und nicht 37:

```
((x > 10 && x < 20) || (x > 30 && x < 40)) &&  
x != 15 && x != 37
```



# Fragen zum Übungsblatt?