Willkommen

Willkommen

Letztes Mal

- Klasse und Objekt
- Attribute
- Elementare Datentypen

Willkommen

Letztes Mal

- Klasse und Objekt
- Attribute
- Elementare Datentypen

Dieses Mal

- Methoden
- Variablen
- Einfache Operationen

Willkommen

Letztes Mal

- Klasse und Objekt
- Attribute
- Elementare Datentypen

Dieses Mal

- Methoden
- Variablen
- Einfache Operationen

Nächstes Mal

- Konstruktoren
- Kontrollfluss

Wie ist das 1. Übungsblatt gelaufen?

Einschub: Hello World! – Reloaded

Warum?

Wir können jetzt HelloWorld.java compilieren und ausführen...

Einschub: Hello World! – Reloaded

Warum?

Wir können jetzt HelloWorld.java compilieren und ausführen...
...nun schauen wir es uns mal ein bisschen genauer an!

Einschub: Hello World! – main

HelloWorld.java

```
1  /**
2   * This class implements a little  sample program that prints "Hello World!"
3   */
4  class HelloWorld {
5      /**
6       * This method prints "Hello World!"
7       *
8       * @param args unused
9       */
10     public static void main(String[] args) {
11         System.out.println("Hello World!");
12     }
13 }
```

Einschub: Hello World! – main

HelloWorld.java

```java
/**
 * This class implements a little  sample program that prints "Hello World!"
 */
class HelloWorld {
    /**
     * This method prints "Hello World!"
     *
     * @param args unused
     */
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

public static void main(String[] args)

- public static Schlüsselwörter
- void Rückgabetyp
- main Methodenname
- String[] args Parameter

Einschub: Hello World! – Schlüsselwörter

`public`

- **Methoden (allgemein)** Erklärung kommt im Laufe des Semesters
- **main (speziell)** Muss immer `public` sein!

Einschub: Hello World! – Schlüsselwörter

`public`

- **Methoden (allgemein)** Erklärung kommt im Laufe des Semesters
- **main (speziell)** Muss immer `public` sein!

`static`

- **Methoden (allgemein)** "normale" Methoden (ohne `static`) beziehen sich auf ein konkretes Objekt dessen Attribute sie lesen (und gegebenfalls verändern) können. Im Gegensatz dazu beziehen sich `static` Methoden auf kein Objekt und werden deswegen auch als Klassenmethoden bezeichnet
- **main (speziell)** Muss immer `static` sein! (kein Objekt vorhanden)

Einschub: Hello World! – Rückgabetyp und Methodenname

Rückgabetyp

Dient der Rückgabe von Informationen an den Aufrufer
Beispiel: Signalisierung von Fehlern

- **Methoden (allgemein)** Grundsätzlich kann jeder Typ, wie wir sie von Attributen kennen, als Rückgabetyp verwendet werden (also unter anderem auch Klassen). Zusätzlich gibt es noch `void`, was bedeutet, dass nichts zurückgegeben wird
- **main (speziell)** Muss immer `void` sein! (Festlegung)

Einschub: Hello World! – Rückgabetyp und Methodenname

Rückgabetyp

Dient der Rückgabe von Informationen an den Aufrufer
Beispiel: Signalisierung von Fehlern

- **Methoden (allgemein)** Grundsätzlich kann jeder Typ, wie wir sie von Attributen kennen, als Rückgabetyp verwendet werden (also unter anderem auch Klassen). Zusätzlich gibt es noch `void`, was bedeutet, dass nichts zurückgegeben wird
- **main (speziell)** Muss immer `void` sein! (Festlegung)

Methodennamen

- **Methoden (allgemein)** Es gilt das gleiche wie für Attributnamen
- **main (speziell)** Muss natürlich `main` sein ;)

Einschub: Hello World! – Parameter (allgemein)

Parameterübergabe

Mit Parametern kann der Aufrufer das Verhalten der Methode beeinflussen
Beispiel: Zahl zu einer anderen addieren (Parameter: zu addierende Zahl)

Einschub: Hello World! – Parameter (allgemein)

Parameterübergabe

Mit Parametern kann der Aufrufer das Verhalten der Methode beeinflussen
Beispiel: Zahl zu einer anderen addieren (Parameter: zu addierende Zahl)

Parameter

Eine Methode kann 0, 1, ... Parameter haben, die jeweils aus Typ (wie die Typen bei Attributen) und Name (auch 'formaler Parameter' genannt) bestehen
Der Wert den ein Parameter zur Ausführung der Methode annimmt nennt man auch 'aktueller Parameter' und kann mittels des 'formalen Parameters' (Parametername) innerhalb der Methode verwendet werden
Beispiele:

```
int power(int base, int exponent)

void printError(String message)
```

Einschub: Hello World! – Parameter (speziell)

Parameter bei `main`

Bis auf den Parameternamen, der nur innerhalb der Methode eine Rolle spielt, ist es immer `String[] args`
`args` ist ein Array (später in der Vorlesung) von Strings, die java mit übergeben werden

Einschub: Hello World! – Parameter (speziell)

Parameter bei `main`

Bis auf den Parameternamen, der nur innerhalb der Methode eine Rolle spielt, ist es immer `String[] args`
`args` ist ein Array (später in der Vorlesung) von Strings, die java mit übergeben werden

Beispiele

```
$ java ParameterOut Hallo
args[0]: Hallo

$ java ParameterOut Hello World
args[0]: Hello
args[1]: World

$ java ParameterOut "Dies ist ein Beispiel"
args[0]: Dies ist ein Beispiel
```

Einschub: Hello World! – Ausgabe

`System.out.println("Hello World!")`

`System`	Eine Klasse die Systemfunktionalität wie Ein- und Ausgabe zur Verfügung stellt
`out`	Ein Attribut der Klasse `System` vom Typ `PrintStream`
`println`	Methoden der Klasse `PrintStream` die verschiedene Datentypen ausgeben können `print` verursacht anders als `println` keinen Zeilenvorschub

Detailierte Informationen findet ihr in der Java API!

Einschub: Hello World! – Ende

HelloWorld.java

```java
/**
 * This class implements a little  sample program that prints "Hello World!"
 */
class HelloWorld {
    /**
     * This method prints "Hello World!"
     *
     * @param args unused
     */
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Habe ich alles erklärt?

Variablen

Wir kennen bereits Attribute in Klassen...

Variablen

Wir kennen bereits Attribute in Klassen...
...Variablen in Methoden funktionieren fast genau so!

Variablen

Wir kennen bereits Attribute in Klassen...
...Variablen in Methoden funktionieren fast genau so!

Zum Beispiel zum Zwischenspeichern von Ergebnissen

Variablen

Wir kennen bereits Attribute in Klassen...
...Variablen in Methoden funktionieren fast genau so!

Zum Beispiel zum Zwischenspeichern von Ergebnissen

Beispiel 1: Deklaration und Zuweisung

```
double mittelwert;
mittelwert = 1 / 2.0 * (x0 + x1);
```

Beispiel 2: Initialisierung

```
double flaeche = (xRechts - xLinks) * (yUnten - yOben);
```

Einfache Operationen

Einleitung

Im Folgenden werden wir eine Menge Operationen kennenlernen die ihr in Java benutzen könnt. In der Regel funktionieren die meisten davon wie man es intuitiv erwarten würde.

Einfache Operationen

Einleitung

Im Folgenden werden wir eine Menge Operationen kennenlernen die ihr in Java benutzen könnt. In der Regel funktionieren die meisten davon wie man es intuitiv erwarten würde.

Anmerkung

Ihr müsst mir das nicht alles glauben, mittlerweile solltet ihr in der Lage sein es mit eigenen Programmen zu überprüfen ;)

Zuweisung

Zuweisungsoperator =

Dem Attribut oder der Variablen auf der linken Seite wird der Wert/Referenz der rechten Seite zugewiesen

Verhält sich nicht symmetrisch! (`a = b` $\not\leftrightarrow$ `b = a`)

Anmerkung: Entspricht in der (Uni) Mathematik eher := als =

Zuweisung

Zuweisungsoperator =

Dem Attribut oder der Variablen auf der linken Seite wird der Wert/Referenz der rechten Seite zugewiesen

Verhält sich nicht symmetrisch! (`a = b` $\not\leftrightarrow$ `b = a`)

Anmerkung: Entspricht in der (Uni) Mathematik eher $:=$ als $=$

Beispiele

```
double a = 5.0;
double b;
a = 3.5;
b = a;

Author author = new Author();
```

Vergleichsoperatoren (1)

Gleichheit ==

- **Bei elementaren Datentypen** prüft ob die Werte identisch sind
- **Bei Klassen** prüft ob die Referenzen (Identität) identisch sind
  Insbesondere können also zwei Objekte mit dem selben Zustand trotzdem bzgl. == verschieden sein
  In der Regel möchte man eher das Verhalten der `equals` Methode

Vergleichsoperatoren (1)

Gleichheit ==

- **Bei elementaren Datentypen** prüft ob die Werte identisch sind
- **Bei Klassen** prüft ob die Referenzen (Identität) identisch sind
  Insbesondere können also zwei Objekte mit dem selben Zustand trotzdem bzgl. == verschieden sein
  In der Regel möchte man eher das Verhalten der `equals` Methode

Beispiele

```
int yRes = 768;
System.out.println(yRes == 1024); // false
System.out.println(yRes == 768); // true

String hello1 = new String("World"):
String hello2 = new String("World");
System.out.println(hello1 == hello1); // true
System.out.println(hello1 == hello2); // false
```

Vergleichsoperatoren (2)

<, <=, >=, >

Operieren auf Ganzzahlen und Gleitkommazahlen
Liefern das (gewohnte) Ergebnis als `boolean`

Vergleichsoperatoren (2)

<, <=, >=, >

Operieren auf Ganzzahlen und Gleitkommazahlen
Liefern das (gewohnte) Ergebnis als `boolean`

Beispiele

```
5 < 4 :
```

Vergleichsoperatoren (2)

<, <=, >=, >

Operieren auf Ganzzahlen und Gleitkommazahlen
Liefern das (gewohnte) Ergebnis als `boolean`

Beispiele

```
5 < 4 : false
4 < 5 :
```

Vergleichsoperatoren (2)

<, <=, >=, >

Operieren auf Ganzzahlen und Gleitkommazahlen
Liefern das (gewohnte) Ergebnis als `boolean`

Beispiele

```
5 < 4 : false
4 < 5 : true
-23 <= 450 :
```

Vergleichsoperatoren (2)

<, <=, >=, >

Operieren auf Ganzzahlen und Gleitkommazahlen
Liefern das (gewohnte) Ergebnis als `boolean`

Beispiele

```
5 < 4 : false
4 < 5 : true
-23 <= 450 : true
-21.5 >= 600 :
```

Vergleichsoperatoren (2)

<, <=, >=, >

Operieren auf Ganzzahlen und Gleitkommazahlen
Liefern das (gewohnte) Ergebnis als `boolean`

Beispiele

```
5 < 4 : false
4 < 5 : true
-23 <= 450 : true
-21.5 >= 600 : false
1E-30 > 1E15 :
```

Vergleichsoperatoren (2)

<, <=, >=, >

Operieren auf Ganzzahlen und Gleitkommazahlen
Liefern das (gewohnte) Ergebnis als `boolean`

Beispiele

```
5 < 4 : false
4 < 5 : true
-23 <= 450 : true
-21.5 >= 600 : false
1E-30 > 1E15 : false
999 > 780 :
```

Vergleichsoperatoren (2)

<, <=, >=, >

Operieren auf Ganzzahlen und Gleitkommazahlen
Liefern das (gewohnte) Ergebnis als `boolean`

Beispiele

```
5 < 4 : false
4 < 5 : true
-23 <= 450 : true
-21.5 >= 600 : false
1E-30 > 1E15 : false
999 > 780 : true
```

Logik

Negation !

```
boolean a = !true;
System.out.println(!false);
System.out.println(!a);
```

Ausgabe:

Logik

Negation !

```
boolean a = !true;
System.out.println(!false);
System.out.println(!a);
```

Ausgabe:

```
true
true
```

Logik

Negation !

```
boolean a = !true;
System.out.println(!false);
System.out.println(!a);
```

Ausgabe:

```
true
true
```

Und &&, Oder ||

`a && b` : ist genau dann `true`, wenn `a` und `b` `true` sind

`a || b` : ist nur `true`, wenn mindestens `a` oder `b` `true` ist

Arithmetische Operationen – Vorwort

Manche Menschen behaupten, der Computer könne rechnen...

Arithmetische Operationen – Vorwort

Manche Menschen behaupten, der Computer könne rechnen...

...das stimmt aber nicht!

Arithmetische Operationen – Vorwort

Manche Menschen behaupten, der Computer könne rechnen...

...das stimmt aber nicht!

Der Computer macht gewisse Fehler die einem bewusst sein sollten!

Arithmetische Operationen – Vorwort

Manche Menschen behaupten, der Computer könne rechnen...

...das stimmt aber nicht!

Der Computer macht gewisse Fehler die einem bewusst sein sollten!

- Endliche Genauigkeit
- Überlauf
- Unterlauf

Zum Beispiel werden im Finanzbereich häufig Festkommazahlen (was eher Ganzzahlen entspricht) statt Gleitkommazahlen verwendet um keine Genauigkeit zu verlieren

Arithmetische Operationen – Vorwort

Manche Menschen behaupten, der Computer könne rechnen...

...das stimmt aber nicht!

Der Computer macht gewisse Fehler die einem bewusst sein sollten!

- Endliche Genauigkeit
- Überlauf
- Unterlauf

Zum Beispiel werden im Finanzbereich häufig Festkommazahlen (was eher Ganzzahlen entspricht) statt Gleitkommazahlen verwendet um keine Genauigkeit zu verlieren

Fazit

Gleitkommazahlen nie mit == vergleichen, sondern den Betrag der Differnz gegen ein kleines ϵ abschätzen!

Arithmetische Operationen

Operationen

- Addition: +
- Subtraktion -
- Multiplikation *
- Division /
- Modulo %

Hinweis: Wenn beide Argumente Ganzzahlen sind, wird eine Ganzzahloperation durchgeführt, was bei der Division bedeutet, dass das Ergebnis immer eine ganze Zahl ist und der Rest ignoriert wird!

Methoden – Ergänzungen

Bezugsobjekt

Alle nicht-`static` Methoden haben ein Bezugsobjekt
Man kann auf Attribute des Bezugsobjekts zugreifen, indem man einfach ihren Attributnamen schreibt
Eine Referenz auf das Bezugsobjekt ist via `this` verfügbar

Rückgabewerte

Man kann mittels `return` den Rückgabewert einer Methode festlegen
Der Rückgabewert sollte natürlich zu dem angegeben Rückgabetyp passen
Bei einem `return` endet auch die Ausführung der Methode

Übung: Statistik

Aufgabe

Schreiben Sie eine Klasse die Mittelwert und Standardabweichung für Datensätze berechnet. Die Daten werden einzeln in Form von Gleitkommazahlen an eine Methode `hinzufuegen` ihrer Klasse übergeben. Der Mittelwert soll zu jeder Zeit über die Methode `leseMittelwert` und die Standardabweichung über eine Methode `leseStandardabweichung` augelesen werden können.
Sie können hierbei davon ausgehen, dass die Methoden zum Auslesen nicht aufgerufen werden, bevor mindestens 2 Datensätze eingelesen wurden!

Hilfe (ohne Gewähr)

Mittelwert $E(X) = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1+x_2+\ldots+x_n}{n}$

Varianz $Var(X) = E(X^2) - (E(X))^2$

Standardabweichung $\sigma_X = \sqrt{Var(X)}$ (Java: `Math.sqrt`)