

Willkommen

Wie ist das 1. Übungsblatt gelaufen?

Warum?

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

Warum?

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

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 }
```

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 }
```

public static void main(String[] args)

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

`public`

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

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 gegebenenfalls 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)

Rückgabotyp

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 Rückgabotyp 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)

Rückgabotyp

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 Rückgabotyp 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 formalen Parameter, 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 formalen Parameter, 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
```

```
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

Detaillierte Informationen findet ihr in der Java API!

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 }
```

Habe ich alles erklärt?

Wir kennen bereits Attribute in Klassen...

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

Wir kennen bereits Attribute in Klassen...

...Variablen in Methoden funktionieren fast genau so!

Zum Beispiel zum Zwischenspeichern von Ergebnissen

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);
```

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 angegebenen Rückgabebetyp passen

Bei einem `return` endet auch die Ausführung der Methode

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

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

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