Auftrag Syntax

|  |  |
| --- | --- |
| Richtzeit  1,5 Tage |  |

# Einführung

Bei diesem Auftrag wird Step-by-Step die Syntax von Java erklärt und mittels Übungen angewendet.  
Das Lösen der Aufgaben in möglichst kurzer Zeit ist nicht das Ziel, sondern die intensive Auseinandersetzung mit dem Code anhand von Theorie, Beispielen und Übungen.   
Beim Programmieren gilt: „Verstanden hast du nur, was du angewendet hast!“  
Spiele mit dem Code herum und führe ihn möglichst oft aus. Überprüfe die Funktionsweise deiner Codezeilen mithilfe des **Debugger**.

Nütze Google. Nütze aber auch das Coaching, falls du länger nicht auf eine Lösung kommst, oder mit dem Verständnis Mühe haben solltest. Notiere dir allfällige Fragen.

Viel Spass!

# Erste Schritte in Java

Betrachte den Code. Wir werden noch sehr detailliert darauf eingehen. Zu Beginn begnügen wir uns mit einer groben Vorstellung davon, was wir vor uns haben.

public class SomeKlasse {  
 public static void main(String[] args) {  
   
 ***// Kommentar: Hier schreibst du deinen Code***}  
}

Auf der ersten Zeile wird eine Klasse SomeKlassedefiniert.

Eine **Klasse** kannst du dir als Container von Funktionen (und Daten) vorstellen. Bei Java werden Funktionen als **Methoden** bezeichnet. **Eine Datei (Sourcefile) enthält genau eine Klasse.**

Auf der zweiten Zeile wird eine Methode mit den Namen main() definiert. Innerhalb der runden Klammern werden die Parameter inkl. Datentyp definiert.

Begriffserklärung

Innerhalb von Methoden spricht man von **Parametern** für die Variablen in den runden Klammern.

Rufst du eine Methode auf und übergibst ein Wert, spricht man von einem **Argument,** welches du übergibst.



Bei Python wurden Codeblöcke mittels „**Einrücken“** definiert. Java verwendet geschweifte   
Klammern{}. Der Code wird zwar ebenfalls eingerückt, dies dient jedoch lediglich der Lesbarkeit.

Ein Programm besteht aus vielen **Klassen** mit vielen **Methoden,** jedoch nur aus **einer main()-Methode**. (Shortcut: psvm). Wenn du ein Python-Script ausführst, startet das Programm an der ersten Zeile. Bei Java ist dies anders, hier ist diemain()**-Methode der Einstiegspunkt** in deine Applikation.

**Achtung!**

Bei Python konntest du Befehle direkt ins Sourcefile schreiben. In diesem Auftrag befindet sich **jede Befehlszeile innerhalb einer Methode**.



Soweit die Theorie zu Klassen. In diesem Auftrag sollst du jetzt aber die Syntax von Java üben. Mehr von Klassen und Methoden erfährst du in der nächsten Präsi.

## Variablen & Datentypen

String s; *// Deklaration der Variable «s»*  
s = "Hallo"; *// Initialisierung von «s»*  
int i = 10;  
double d = 12.1; *// double ist das neue float*boolean b = true;

Bei Java muss jede Codezeile mit einem **Semikolon** enden. Es können beliebig viele Codezeilen auf einer Textzeile stehen.

Java ist **stark typisiert**. Im Gegensatz zum **schwach typisierten** Python, musst du an vielen Orten zusätzlich den Datentypen angeben zum Beispiel bei der Deklaration von Variablen.

Verwende in Java für Fliesskommazahlen den Datentyp double anstatt float.

## Kommentare

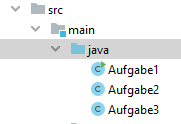
**Schreibe Kommentare im Code nur falls nötig.** Versuche durch gute Namensgebung den Code so zu schreiben, dass er auch ohne Kommentare verständlich ist. Falls dies nicht möglich ist, schreibe ein Kommentar.

*// Der Kommentar geht bis zum Ende der Zeile*} **else** */\* Kommentar & Code auf einer Zeile \*/* {  
  
*/\*  
 \* Das ist ein Kommentar*

*\* über mehrere Zeilen  
 \*/*

# Vorbereitung

Erstelle ein neues Java-Gradle-Projekt. GroupId ch.bbcag und ArtifactId syntax.



**Hinweis**

Erstelle für **jede Aufgabe eine neue, sinnvoll benannte Klasse in deinem Projekt** syntax**.** So dass du jede Aufgabe separat ausführen kannst.



public class Aufgabe1 {  
 public static void main(String[] args) {  
   
 ***// Kommentar: Hier schreibst du deinen Code***}  
}

**Achtung!**

Schreibe den Lösungscode für diesen Auftrag ausschliesslich innerhalb der main()-Methoden!   
(Shortcut: psvm)



### Aufgabe 1 – Ergänze den Code

1. Kopiere den untenstehenden Code in eine neue Klasse Aufgabe1 und ersetze alle <TODO> mit den passenden Zeichen/Datentypen, so dass die Klasse kompiliert.
2. Die Ausgabe in der Konsole muss: Gratulation! sein.   
   Achtung: Du darfst nur die <TODO> ändern!

public class Aufgabe1 {

public static void main(String[] args) {

int a = **<TODO>**;

**<TODO>** b = 10.0;

**<TODO>** Dies ist ein Kommentar

double resultat = a + b**<TODO>**

String text = "Das Resultat ist: **<TODO>** + resultat;

if (resultat < 10.0) {

System.out.println("Gratulation!");

} **<TODO>** {

System.out.println(text);

}

}

}

### Aufgabe 2 – Viel Glück

#### Viel Glück

Schreibe ein Programm, das 10-mal Viel Glück! wünscht. Verwende dazu eine for-Schlaufe.

for (int x = **?**; x <= **?**; x++) {  
 ...  
}

Schreibe anschliessend 5-mal deinen Namen mithilfe einer while-Schlaufe und noch einmal 5-mal mit einer do-while**-**Schlaufe. Die main()-Methode soll also am Ende 20 x System.out.println() aufrufen.

### Aufgabe 3 – Dreieck

Zeichne ein Dreieck mithilfe einer for-Schlaufe.

for (int i = ???; ???; ???) {  
 *// Zeichnungscode*}

Ausgabe in der Konsole:  
  
\*  
\*\*  
\*\*\*  
\*\*\*\*  
\*\*\*\*\*

### Aufgabe 4 – Das kleine Einmaleins

Schreibe ein Programm, das mithilfe von zwei verschachtelten for-Schlaufen folgenden Output erzeugt.

1-er Reihe: 1 2 3 4 5 6 7 8 9 10

2-er Reihe: 2 4 6 8 10 12 14 16 18 20

3-er Reihe: 3 6 9 12 15 18 21 24 27 30

4-er Reihe: 4 8 12 16 20 24 28 32 36 40

5-er Reihe: 5 10 15 20 25 30 35 40 45 50

6-er Reihe: 6 12 18 24 30 36 42 48 54 60

7-er Reihe: 7 14 21 28 35 42 49 56 63 70

8-er Reihe: 8 16 24 32 40 48 56 64 72 80

9-er Reihe: 9 18 27 36 45 54 63 72 81 90

10-er Reihe: 10 20 30 40 50 60 70 80 90 100

Um diese Aufgabe zu lösen, benötigst du einen Cast von Integer nach String.

String text = Integer.*toString*(100);

**Hinweis**

Weitere Casts für andere Datentypen findest du bei Google.



## Package

Wie du bereits weisst, ist bei Java sämtlicher Code in Klassen. Sowohl dein Code wie auch der Code der Standardklassen im JDK. Bei Integer.toString(100) handelt es sich zum Beispiel um die Klasse Integer mit der Methode toString().

**Packages sind wie Ordner, welche Klassen beinhalten.** Alle JDK-Klassen befinden sich in Packages. Das Package java.lang steht dir implizit immer zur Verfügung. Es beinhaltet unter anderem die Klassen Integer, String und System.   
Fall du weitere Klassen aus der Java-SDK benötigst, muss du diese im IntelliJ mittels **Alt + Enter** explizit importieren: import javax.swing.\*;

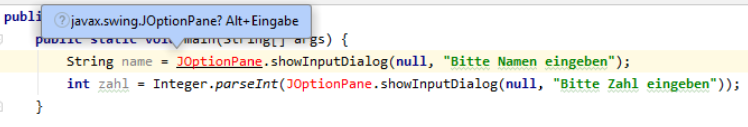


Abbildung 1: Automatisches Importieren von Bibliotheken mit IntelliJ

Kopiere den Code in eine neue Klasse um dies auszuprobieren und führe den Code aus.

public static void main(String[] args) {  
 String name = JOptionPane.*showInputDialog*(null, "Bitte Namen eingeben");  
 int zahl = Integer.*parseInt*(JOptionPane.*showInputDialog*(null, "Bitte Zahl eingeben"));  
}

Wenn zukünftig bei Aufträgen Benutzereingaben verlangt werden, kannst du das ab nun mit einem Dialog lösen.

### Aufgabe 5 - Eingabebedingte Ausgabe

Frage den Benutzer deines Programms nach einer Zahl. Je nach Eingabe soll folgende Meldung am Bildschirm erscheinen.

Positive Zahl: „Deine Zahl ist grösser 0“  
Negative Zahl: „Deine Zahl ist kleiner 0“  
0: „Deine Zahl ist die 0“

### Aufgabe 6 – Strings vergleichen mit equals()

Schreibe ein Programm, welches dich solange nach deinem Vornamen fragt, bis du deinen Vornamen richtig eingegeben hast.

**Achtung!**

Java vergleicht String mittels der equals()-Methode und nicht mittels Doppelgleich ==.



* Falls du deinen Vornamen richtig eingibst, soll der Text Das bin ich! ausgegeben werden und deine Applikation soll sich beenden
* Falls du deinen Vornamen falsch eingibst, soll der Text Das bin ich nicht! ausgegeben werden

### Auftrag 7 - Quadratzahlen

Gib auf der Konsole die ersten 25 Quadratzahlen aus.

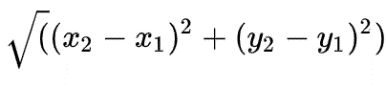
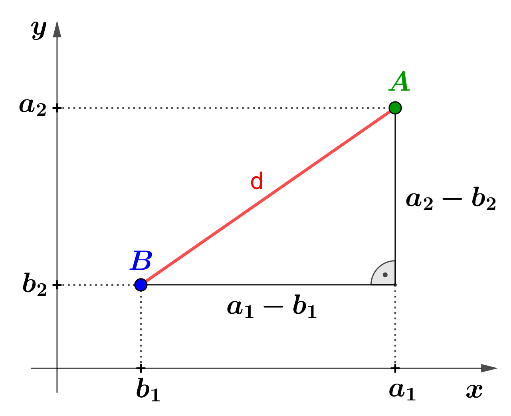
**Hinweis**

Mit Math.pow(x, 2) kannst du x² ausrechnen. Verwende eine Schlaufe.



// Ausgabe  
// 1^2 = 1  
// 2^2 = 4  
// 3^2 = 9  
// 4^2 = 16  
// ...

### Auftrag 8 – Berechne die Distanz zwischen zwei Punkten

Die Distanz zwischen zwei Punkten berechnet man mittels Satzes von Pythagoras.

Distanz =

Vervollständige den Code, so dass folgender Output erzeugt wird:



public static void main(String[] args) {  
 int x1 = 1;  
 int y1 = 1;  
 int x2 = 4;  
 int y2 = 4;  
 double distance = **<TODO>**  
 System.*out*.println("Distance: " + distance);  
}

**Hinweis**

Wir haben dir gezeigt, wie du die Math.pow()-Methode benutzen kannst, um hochzurechnen.

Findest du selbst heraus, wie du Quadratwurzeln berechnen kannst?



### Auftrag 9 - BMI

Frage den Benutzer nach seinem Gewicht, seinem Namen und seiner Körpergrösse. Das Programm soll den Bodymassindex mit folgender Formel berechnen:

Gewicht (kg)

BMI = ------------

Grösse² (m)

Gib den Namen und BMI auf der Konsole aus.

### Auftrag 10 - Kleiner Rechner

Der Benutzer kann zwei Zahlen und einen Operator eingeben. (+, \*, /, -, %)

Am Bildschirm soll nun die entsprechende Rechenaufgabe und das Resultat erscheinen:

System.out.println(zahl1 + " \* " + zahl2 + "ergibt " + resultat);

**Verwende dabei eine switch-case Anweisung!**

### Auftrag 11 - Einfache Methoden aus dem Paket java.lang

Teste folgende Methoden aus. Was machen sie? Beschreibe Zweck und Output mittels Kommentar.

System.out.println(Math.random());  
System.out.println(Math.PI);  
System.out.println(Math.sqrt(81));  
System.out.println(Math.round(3.14));  
System.out.println(Math.max(2343, 432));

## Arrays

**Arrays** bei Java beinhalten nur Werte **eines** Datentyps**,** die Länge des Arrays ist **unveränderbar** und muss beim Initialisieren gesetzt werden.   
Du kannst ein Array als Liste von bestimmter Länge erstellen, oder das Array auch gleich mit konkreten Werten initialisieren.

int[] meinArray = new int[3];  
int[] zahlen = { 10, 20, 13 };  
String[] farben = { "Rot", "Grün", "Blau" };

Das Lesen und Schreiben bei Arrays erfolgt über den **Index-Operator**:

int x = zahlen[0]; *// x = 10*zahlen[1] = 15; *// [10, 15, 13]*

x = zahlen.length; *// x = 3*

Für das Durchlaufen (iterieren) durch Arrays, existiert eine spezielle for**-Schlaufe**.

for (String farbe : farben) {  
 System.*out*.print(farbe);  
}

#### Task

Kopiere die Codezeilen in eine neue Klasse, spiele etwas herum und debugge einmal durch.

### Auftrag 12 – Arrays

Schreibe die Lösungen alle untereinander in eine main()-Methode.

1. Deklariere ein Array vom Datentyp int mit der Länge 5. Fülle die Zahlen 0 – 4 von Hand in das Array. Gib alle Werte des Arrays mithilfe einer Schlaufe in der Konsole aus.
2. Deklariere ein Array vom Datentyp int mit der Länge 100. Fülle die Zahlen 0 - 99 mit Hilfe einer Schlaufe in das Array. Gib alle Werte des Arrays in der Konsole aus.
3. Nimm das Array aus der vorherigen Aufgabe und addiere alle Werte des Arrays. Welches Resultat ergibt sich?
4. Iteriere nochmals durch dein Array und inkrementiere (+1) alle geraden Zahlen und dekrementiere (-1) alle ungeraden Zahlen.

## Casten von primitiven Datentypen

Auch Java kennt das Casten von Datentypen:

int i = Integer.*parseInt*("10");  
double d = Double.*parseDouble*("5.5");  
String text = Integer.*toString*(100);

Neu ist jedoch das Casten von Datentypen mit unterschiedlicher Wertigkeit. Gemeint ist hier Double ⬄ Float oder Long ⬄ Integer. Hier wird zwischen **implizitem und explizitem Cast unterschieden.**Nichts beachten musst du bei einem Cast von einem **niederwertigen zu** einem **höherwertigen Datentyp**, also Integer zu Long oder Float zu Double.

int wert = 10;  
long wert2 = 30;  
  
wert2 = wert;

Jedoch beim Casten eines **höherwertigen zu** einem **niederwertigen Datentyp,** musst du dies explizit angeben. Falls der Wert in der Long Variable zu gross ist, wird dein Code eine Exception werfen bzw. abbrechen.

int wert = 10;  
float wert2 = 30.5f;  
  
wert = **(int)** wert2;

Begriffserklärung

**Explizit** heisst, du als Programmierer musst etwas selbst hinschreiben.

**Implizit** heisst, es ist bereits etwas unsichtbares vorhanden, das du nicht mehr hinschreiben musst.



### Auftrag 13 – Explizites casten von Long nach Integer

public class Aufgabe13 {  
 public static void main(String[] args) {  
 long[] numbersAsLong = { 10, 20, 30};  
  
 int[] numberAsInteger = new int[numbersAsLong.length];  
  
 *// TODO*  
 for (int number : numberAsInteger){  
 System.*out*.println(number);  
 }  
 }  
}

Schreibe beim *//TODO* deinen Code. Nimm die Werte aus dem Array numbersAsLong und setze sie in der gleichen Rheinfolge in das numberAsInteger-Array.

**Hinweis**

Du wirst explizit casten müssen.



### Aufgabe 14 – Die Tanne

**Zusatz**

Diese Aufgabe ist **optional**.   
Behalte die Zeit im Auge.



Zeichne eine Tanne mithilfe von mehreren verschachtelten for-Schlaufen. Beachte, dass eine Schlaufe auch rückwärts gezählt werden kann:  
  
for (int i = 5; i > 0; i--) {  
 / /Zeichnungscode  
}

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

### Auftrag 15 – Weitere Aufgaben mit Arrays

Schreibe den Code möglichst selbst ohne die Verwendung von Java SDK Methoden.

**Zusatz**

Diese Aufgabe ist **optional**.   
Behalte die Zeit im Auge.



1. Generiere ein Array mit 10 Zufallsganzzahlen.
2. Gib alle Zahlen des Arrays auf der Konsole aus.
3. Gib die **kleinste** und die **grösste** Zahl auf der Konsole aus.
4. Gib die **Summe** aller Zahlen des Arrays auf der Konsole aus.
5. Gib den **Durchschnitt** aller Zahlen des Arrays auf der Konsole aus.
6. Erstelle ein neues Array, welches **nur die geraden** Zahlen des ersten Arrays beinhaltet.

### Auftrag 16 – Zahlenraten

**Zusatz**

Diese Aufgabe ist **optional**.   
Behalte die Zeit im Auge.



Erstelle ein Zahlenraten. Die gesuchte Zahl soll mittels Math.random() erzeugt werden.   
Der Benutzer muss solange eine Zahl eingeben bis er sie erraten hat. Natürlich sollte die gesuchte Zahl in einem bestimmten Intervall sein. Zum Beispiel zwischen 1 und 100.

Etwas einfacher für den Benutzer wird es, wenn der Computer nach jeder Eingabe eine Meldung herausgibt, welche aussagt, ob die Zahl grösser oder kleiner als die gesuchte Zahl ist.

Mögliche Erweiterungen:

* Gib am Ende aus, wie viele Versuche der Benutzer hatte, um die gesuchte Zahl zu erraten.
* Am Ende des Programms soll nachgefragt werden, ob der Benutzer ein neues Spiel mit neuer Zufallszahl starten will.
* Zu Beginn kann der Benutzer das Intervall wählen, aus welchem die Zufallszahl gewählt wird. (1-500 oder beliebig)
* Zwei Spieler können am gleichen PC das Zufallsraten abwechselnd spielen. Gewonnen hat wer eher die Zahl erraten hat.
* Die Resultate werden gespeichert, somit kann am Ende aller Spiele eine Statistik erstellt werden. (z.B.: Im Schnitt brauchte der Benutzer 3.5 Versuche, um die Zahl zu erraten)
* und viel mehr…

### Erweiterung

Weitere Übungen sind auf <http://codingbat.com/> vorhanden. Versuche Übungen zu lösen, welche deinem derzeitigen Niveau entsprechen.