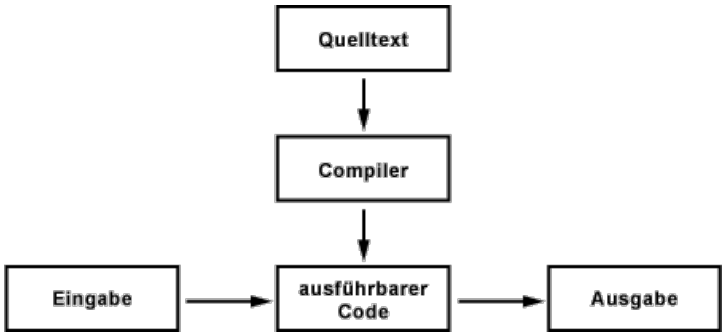
Java Grundlagen

Einstieg in die Grundlagen der Programmierung

1. Compiler und Interpreter

Compiler

Der Compiler übersetzt vor Programmausführung den Quelltext einmal und wird danach zur Programmausführung **nicht** mehr benötigt.



Vorteile:

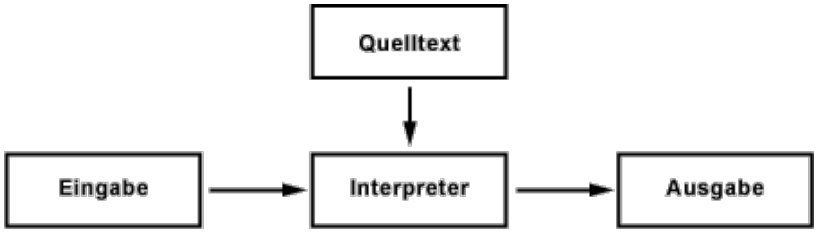
- Hohe Ausführungsgeschwindigkeit
- Sicherheit vor Eingriffen

Nachteile:

- Plattformabhängigkeit
- Bei Programmänderung muss neu kompiliert werden

Interpreter

Der Interpreter übersetzt den Quelltext jedes Mal zur Laufzeit des Programms und wird **immer** zur Programmausführung benötig.



Vorteile:

- Plattformunabhängigkeit
- Leichteres Debugging

Nachteile:

- Langsamer
- ineffizienter

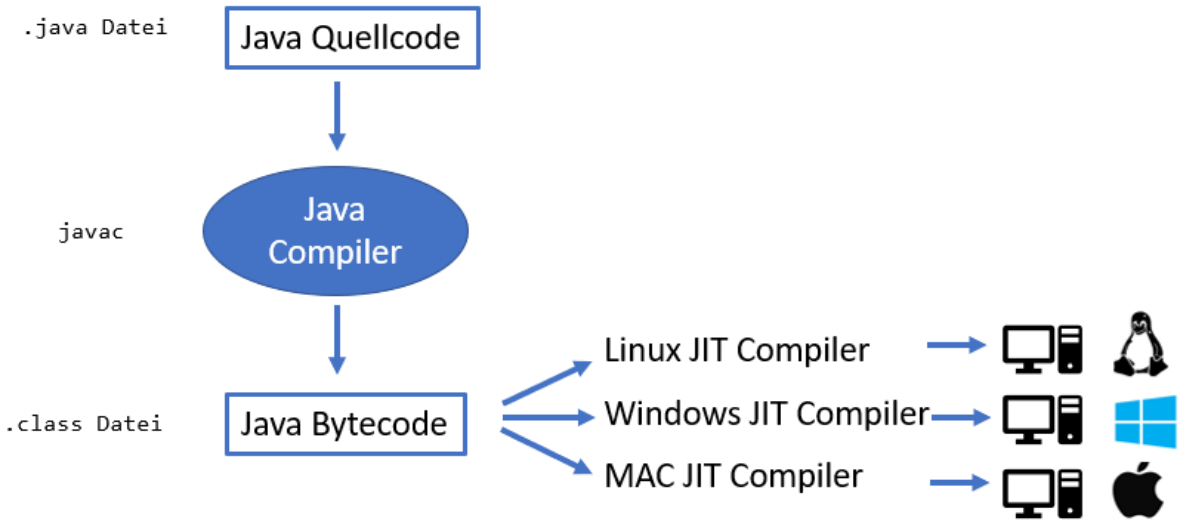
JIT-Compiler (JIT = Just in time)

Just-in-time-Kompilierung ist ein Verfahren, um Programme zur Laufzeit in Maschinencode zu übersetzen. Ziel ist es dabei, die Ausführungsgeschwindigkeit gegenüber einem Interpreter zu steigern.

Wie beim Interpreter wird der Quelltext in eine von der Hardware unabhängigen Bytecode übersetzt. Bei der Ausführung dieser Anweisungen durch den JIT-Compiler werden die Anweisungen in Maschinencode übersetzt. Der JIT-Compiler berücksichtigt dabei die verschiedenen Eigenheiten des Prozessors.

Beim JIT-Compiler wird aus dem Bytecode der Programmiersprache der Maschinencode für den Prozessor generiert, weil das schneller ist, als wenn der Interpreter den Bytecode ausführt. Der Bytecode wird nach der Ausführung verworfen und muss bei nochmaliger Ausführung erneut übersetzt werden.

Typischerweise arbeitet man mit einem JIT-Compiler, wenn es um das Prinzip "write once, run anywhere" geht. Wenn es also darum geht, den Quelltext auf unterschiedlichen Betriebssystemen und Hardware einzusetzen. Eine Software wird einmal geschrieben und ist dann auf verschiedenen Plattformen lauffähig (Portabilität).



2. Einfache Ausgaben in Java

- Beschreiben Sie den Unterschied zwischen `System.out.println("xxx")` und `System.out.print("xxx")`?

println fängt nach der Ausgabe eine neue Zeile an

- Beschreiben Sie folgende Ausgaben!

`System.out.print("\n");` ? Geht in die nächste Zeile

`System.out.print("\t");` ? Entspricht der TAB-Taste bei Text-Editoren

- Welche Ausgabe liefern die Zeilen dargestellten Programmzeilen?

  Versuchen Sie die Lösung erst **ohne Programmierung** zu lösen! Überprüfen Sie anschließend Ihre Ergebnisse am Rechner!

Aufgabe 1

```
System.out.println("1");
System.out.print("2");
System.out.print("3");
System.out.println("4");
```

1
234
–

Aufgabe 2

```
System.out.print("1\n");
System.out.print("2");
System.out.print("3\n");
System.out.print("4\n");
```

1
23
4
–

Aufgabe 3

```
System.out.println("1\n");
System.out.print("2\n");
System.out.print("3\n");
System.out.println("4\n");
```

1
2
3
4
–

Aufgabe 4

```
System.out.println("1");
System.out.println("2\n");
System.out.print("3");
System.out.println("4");
```

1
2
34
–

3. Datentypen

Variablendeklaration und Variableninitialisierung

Bei Java muss jede Variable vor der Verwendung eingeführt werden (Deklaration). Variablen müssen einen **Typ** und einen **Namen** besitzen. Als Initialisierung bezeichnet man die Wertzuweisung zu einer Variablen.

Deklaration: Typ name ;

Initialisierung: name = WERT ;

Primitive Datentypen

Typ	Bit-Tiefe	Wertebereich
boolean und char		
boolean	JVM-spezifisch	*true* oder *false*
char	16 Bit	Unicode Zeichen
numerisch (alle vorzeichenbehaftet)		
Ganzzahlen		
byte	8 Bit	- 128 bis 127
short	16 Bit	- 32768 bis + 32767
int	32 Bit	- 2 147 483 648 bis + 2 147 483 647
long	64 Bit	-9 223 372 036 854 775 808 bis +9 223 372 036 854 775 807
Fließkommazahlen		
float	32 Bit	variabel
double	64 Bit	variabel

camelCase

Camel case is the practice of writing compound words or phrases such that each word or abbreviation in the middle of the phrase begins with a capital letter, with no intervening spaces or punctuation. Example: iPhone, eBay

Variablen schreiben wir beginnend mit einem **kleinen Buchstaben**,
Klassen mit einem **Großbuchstaben**.
Bsp.: `boolean headlessChickenMonster = true;` [1]

[1] Die Seegurke Enypniastes eximia, auch "headless chicken monster" genannt, ist eine äußerst seltene Spezies. Sie erreicht Körperlängen von 6 bis 25 cm und halten sich in Meeren in Tiefen zwischen 300 und 6000 Metern auf.

Wert und Variable

```
int x = 24;
byte b = x; // funktioniert nicht!!
```

Begründen Sie, warum das nicht funktioniert!

Der reservierte Speicher des Integers passt nicht in das byte rein

Verbessern Sie falls nötig die Anweisungen! Die Quellcodeanweisungen sind zusammenhängend!

```
1. int x = 34.5;
2. boolean boo = 20;
3. int g = 17;
4. int y = g;
5. byte b = 3;
6. byte v = b;
7. short n = 32768;
```

```
8.  float wert = 32.5 f ;
9.  char test = "a";
10. long bla = 2147483647;
11. long blabla = 2147483648L ;
12. int 4zahl = 129;
13. int erg = 34__12;
14. double wert = 14,3;
```

4. String

Ein String ist eine Zeichenkette. Der String zählt nicht zu den primitiven Datentypen und hat als Wertebereich die Menge aller Zeichenketten. Strings werden in Anführungszeichen gesetzt. Beispiele für Strings sind "abba", "Hallo Fritz!", "@%#&$", "x".
Das folgende Programmstück zeigt die Deklaration einer Variablen ***s*** vom Typ String gefolgt von einer Wertzuweisung und anschließender Ausgabe des Strings.

```
String s1 = "Hallo ";
String s2 = "Welt!";
System.out.println(s1);
System.out.println(s2);
System.out.println(s1 + s2);
```

Programmierauftrag – Steckbrief

Erstellen Sie ein Programm, mit Variablen für Name, Alter und Geschlecht (m/w). Initialisieren Sie die Variablen mit Werten und geben Sie diese über die Ausgabe am Bildschirm aus! Erweitern Sie das Programm um weitere Werte bzw. persönliche Daten des Steckbriefes!

5. Berechnung von Ganz- und Fließkommazahlen

Arbeitsauftrag: Notenberechnung

Schreiben Sie nachfolgendes Programm ab, kompilieren Sie es und testen Sie es aus.

```
1  class Noten
2    {
3    public static void main(String[] args)
4      {
5      int schul1, schul2, ex1, ex2, anzahl;
6      double notenschnitt;
7      schul1 = 2;
8      schul2 = 3;
9      ex1 = 4;
10     ex2 = 2;
11     anzahl = 6;
12     notenschnitt = (schul1*2 + schul2*2 + ex1 + ex2)/anzahl;
13     System.out.println("1.Schulaufgabe: " + schul1);
14     System.out.println("2.Schulaufgabe: " + schul2);
15     System.out.println("1.Ex:           " + ex1);
16     System.out.println("2.Ex:           " + ex2);
17     System.out.println("_____________________");
18     System.out.println("Notenschnitt = " + notenschnitt);
19     }
20   }
```

Überprüfen Sie die Berechnung mit einem Taschenrechner! Welches Problem tritt auf?

- Ändern Sie das Programm so ab, dass das aufgetretene Problem beseitigt wird und speichern Sie die neue Variante unter gleichen Namen ab.

  (Die Deklarationen in den Zeilen 5 und 6 dürfen nicht verändert werden!)
  Notieren Sie die Änderung:

  _____

  _____

  _____

Das Programm soll zudem um die Ausgabe der Zeugnisnote (ganze Zahl) erweitert werden.

- Fügen Sie dazu die folgenden Programmzeilen an den entsprechenden Stellen ein:

  ```
  int note;

  note = (int)notenschnitt;

  System.out.println("Zeugnisnote =  " + note);
  ```

- Ermöglichen die eingefügten Zeilen das Berechnen der Zeugnisnote?
  Was bewirkt die Zeile: `note = (int)notenschnitt;`?

  _____

  _____

  _____

- Auf welche Weise lässt sich dieses Problem (Ermittlung der Zeugnisnote) mit ihrem bisherigen Wissen lösen?
  Ergänzen Sie das Programm dahingehend!

- Die Angabe des Notenschnitts auf 16 Nachkommastellen ist nicht effektiv. Daher soll der Notenschnitt auf zwei Stellen nach dem Komma richtig gerundet angegeben werden. Lösen Sie dieses Problem mit den bisher zur Verfügung stehenden Mitteln.