

Softwareentwicklung

1. Einführung

Was ist Programmieren eigentlich?

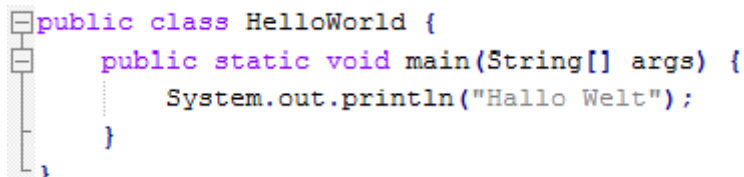
Programmiersprachen sind die Schnittstelle zwischen Mensch und Computer. Aber was bedeutet programmieren genau? Folgender Ablauf beschreibt es etwas besser.

1. Schreiben: Hierbei schreibt der Programmierer alle Teilfunktionen eines Programms in Form einer sehr logisch strukturierten Sprache. Das Geschriebene bezeichnet man als Quellcode bzw. source code.
2. Kompilieren: Hier wird der Quellcode vom einem Programm namens Compiler in Maschinencode umgewandelt. Computer verstehen grundsätzlich nur Maschinencode (also 1en und 0en).
3. Ausführen: Beim Ausführen des Maschinencodes wird das Programm in den Arbeitsspeicher geladen und dann mit Maschinenbefehl für Maschinenbefehl ausgeführt.

Die Programmiersprache Java

Java weicht zwar leicht vom obigen Ablauf ab, trotzdem kann er mit Java sehr gut verdeutlicht werden.

1. Quellcode



```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hallo Welt");  
    }  
}
```

2. Kompilieren

```
D:\Programmierung\PureJava>javac HelloWorld.java
```

```
D:\Programmierung\PureJava>_
```

3. Ausführen

```
D:\Programmierung\PureJava>java HelloWorld  
Hallo Welt
```

```
D:\Programmierung\PureJava>_
```

Viele Programme, die täglich von vielen Anwendern genutzt werden, sind in Java geschrieben. Darunter OpenOffice, LibreOffice, JDownloader und Eclipse. Außerdem werden die meisten mobilen Anwendungen für Android mit Java entwickelt.

JDK (Java Development Kit) installieren

Um in Java entwickeln zu können bietet Oracle das JDK an. Darin werden wichtige Funktionen, wie z.B. der Java-Compiler mitausgeliefert.

Download: <http://www.oracle.com/technetwork/java/javase/downloads/jdk7u7-downloads-1836413.html>

Da in der Windows-Konsole nur Programme ausgeführt werden können, die in bestimmten Verzeichnissen liegen, ist es nötig, die sogenannte Path-Variable anzupassen. Das ist nötig, damit

später die Befehle `javac` und `java` vom jedem Ort ausgeführt werden können.

1. Windows-Taste + Pause
2. „Erweiterte Systemeinstellungen“ (linke Seite)
3. „Umgebungsvariablen“
4. In der unteren Liste die Variable `Path` auswählen => Bearbeiten => Je nach System, folgendes hinten an den Wert dranhängen
`;C:\Program Files\Java\jdk1.6.0_21\bin\`
 (Der genaue Pfad ist systemabhängig)
5. Erstellt eine neue Systemvariable.
 Name: `JAVA_HOME`
 Wert: `C:\Program Files\Java\jdk1.6.0_21`
6. Testen, ob alles korrekt konfiguriert wurde
 Konsole öffnen und eintippen: `java -version`

2. Variablen

Damit man als Programmierer Berechnungen durchführen kann, braucht man Hilfsmittel. Dazu gibt es in Java Variablen. Es gibt für verschiedene Anwendungsfälle verschiedene Typen.

<i>Typ</i>	<i>Von</i>	<i>Bis</i>
byte	- 128	+127
short	- 32.768	+32.767
int	- 2.147.483.648	2.147.483.647
long	-9.223.372.036.854.775.808	9.223.372.036.854.775.807
float	+/- 1,4*10 ⁻⁴⁵	+/- 3,4*10 ⁺³⁸
double	+/-4,9*10 ⁻³²⁴	+/-1,7*10 ⁺³⁰⁸
boolean	false	true
char	0	65.535

Beispiel

```
public class HelloWorld {
    public static void main(String[] args) {
        int operator1 = 1;
        int operator2 = 2;
        int ergebnis = operator1 + operator2;
        System.out.println("Ergebnis: " + ergebnis);
    }
}
```

Konstanten

Wenn eine Variable einen festen Wert darstellt, sollte es nicht mehr möglich sein diesen Wert abzuändern.

Beispielsweise ist die mathematische Konstante Pi fest definiert mit 3,14159... und ist universell gültig. Um in einem Programm diesen Wert unveränderbar zu machen, definiert man die Konstante mit `final`.

```
final double pi = 3,14159;
```

Steht danach z. B. diese Anweisung, ...

```
pi = 42;
```

... so würde bereits der Compiler eine Fehlermeldung liefern.

3. Kontrollstrukturen

Kontrollstrukturen dienen in einer Programmiersprache dazu, Programmteile unter bestimmten Bedingungen auszuführen. Dazu gehört nicht nur, das Ausführen eines Programmteils unter einer bestimmten Voraussetzung, sondern auch das wiederholte Ausführen von Teilen.

Nachfolgend sind einige wichtige Kontrollstrukturen gelistet, die Java kennt.

If-Anweisung

Die Anweisung besteht aus dem Schlüsselwort „if“ und einem Ausdruck des Typs boolean (zwingend), der in Klammern folgt.

```
public class HelloWorld {  
    public static void main(String args []) {  
        int operator1 = 1;  
        int operator2 = 2;  
  
        if (operator2 <= 2) {  
            System.out.println(operator1 + operator2);  
        }  
    }  
}
```

Der Programmteil zwischen den geschweiften Klammern, nach der if-Anweisung, wird nur dann ausgeführt, wenn das Ergebnis der if-Anweisung „true“ ist. In diesem Fall ist das der Fall und die Ausgabe erfolgt.

If-Else-Anweisung

Bei dieser Form der Kontrolle gibt es eine echte „Entweder-Oder“-Abfrage. Ergibt die if-Anweisung ein „false“, wird der Else-Teil ausgeführt. Bei „true“, der if-Teil.

```
if (operator2 <= 1) {  
    System.out.println(operator1 + operator2);  
}  
else {  
    System.out.println("Ich bin im Else-Zweig");  
}
```

While-Schleife

Die while-Schleife ist eine Schleife, die vor jedem Schleifeneintritt die Schleifenbedingung prüft.

```
while(operator1 <= 5) {  
    System.out.println("Ich bin in der Schleife");  
    operator1++;  
}
```

For-Schleife

Die for-Schleife ist eine spezielle Variante einer while-Schleife und wird typischerweise zum Zählen benutzt.

```
for ( int i = 1; i <= 10; i++ ) {  
    System.out.println(i);  
}
```

Hinter der for-Anweisung kommen (in dieser Reihenfolge) folgende Parameter:

- Initialisierung der Zählvariablen
- Bedingung
- Hochzählen der Variablen

4. Kommentare

Kommentare sind ein wichtiger Bestandteil von Programmcode. Sie erleichtern es z.B. schwierigen Code zu verstehen. Besonders. Wenn der Code durch verschiedene Hände geht, ist es hilfreich, wenn man sofort versteht, warum ein Programm in der vorhandenen Art und Weise implementiert wurde.

Hierzu gibt es 3 verschiedene Arten von Kommentaren.

- Zeilenkommentare
//Ich bin ein Zeilenkommentar. Ich bin nur eine Zeile lang.
- Blockkommentare
/*
* Ich bin ein Blockkommentar. Ich habe beliebig viele Zeilen.
*/

5. Funktionen

Um die Übersichtlichkeit und Wartbarkeit einer Software zu verbessern, lagert man für gewöhnlich

Logik in sogenannte Funktionen aus. Vor Allem komplexere Algorithmen sollten in separaten Funktionen behandelt werden.

Beispiel 1: Es existiert eine Software, die eine mathematische Kurvendiskussion durchführen soll. Hier sollten alle Teilschritte der Kurvendiskussion (also z. B. Differentiation oder Integration) in verschiedene Funktionen ausgelagert werden.

Beispiel 2: Wir übergeben einem Programm als Parameter eine Anzahl an Tagen. Dieses Programm soll die Tage in Sekunden umrechnen. Um die Komplexität der main-Methode zu senken, lagern wir die Umrechnung in eine Funktion aus. Die main-Methode enthält nur noch den Aufruf der Funktion.

```
public static void main(String[] args) {  
    System.out.println(daysToSeconds(25));  
}
```

Die definierte Funktion sieht so aus:

```
public static long daysToSeconds(long days) {  
    long hours = days * 24;  
    long minutes = hours * 60;  
    long seconds = minutes * 60;  
    return seconds;  
}
```

Dieses Beispiel zeigt zwei unterschiedliche Funktionen. Die main-Methode ist eine Funktion ohne Rückgabewert. Bei ihr ist es nicht nötig, Werte zurückzugeben, weil sie ein von der Umwelt unabhängiges Konstrukt darstellt.

Die Funktion `daysToSeconds` zeigt eine Funktion **mit** Rückgabewert (in diesem Fall die Gesamtsekunden der übergebenen Tage).

Der Variablentyp des Rückgabewerts wird vor dem Namen der Methode angegeben. Genau das gleiche gilt auch für die Parameter der Methode. Deren Typen und Namen werden in den Klammern definiert.

6. Objektorientierung (Klassen, Methoden)

Java gehört zu den objektorientierten Programmiersprachen. Das bedeutet, dass man in Java, Objekte definieren und Operationen auf Ihnen ausführen kann. Hauptgrund für die Objektorientierung von Java ist die Unterstützung von Wiederverwendbarkeit und dass die reale Welt auch aus Objekten besteht.

Eine Klasse definiert einen neuen Typ, beschreibt die Eigenschaften der Objekte und gibt somit den Bauplan an.

Das Konzept der Objektorientierung beruht größtenteils darauf, dass man die reale Welt auf sein Programm überträgt.

7. Literatur

- Java ist auch eine Insel (Galileo <openbook>)
<http://openbook.galileocomputing.de/javainsel/>
- Java von Kopf bis Fuß
ISBN-10: 3897214482; ISBN-13: 978-3897214484