Vorkurs WS 2014/15 der Fachschaft 07

Eine Einführung in die Softwareentwicklung anlässlich des Vorkurses für die Erstsemester im WS2014/15

Zusammengestellt von der Fachschaft 07

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 von 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 dieser mit Java sehr gut verdeutlicht werden.

1. Quellcode

```
|public class HelloWorld {
|    public static void main(String[] args) {
|         System.out.println(„Hello World“);
|    }
|}
```

2. Kompilieren

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

3. Ausführen

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

Anmerkung: System.out.println() wird genutzt, um Text auf der Kommandozeile auszugeben.

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.

Vorkurs WS 2014/15 der Fachschaft 07

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, mit ausgeliefert.

Downloaden könnt ihr das unter oracle.com.
Googelt im einfachsten Fall einfach nach „Java SE Development Kit 8“.

Da in der Windows-Konsole nur Programme ausgeführt werden können, die in bestimmten Verzeichnissen liegen, ist es nötig, die sogenannte Path-Variable anzupassen, damit später die Befehle javac und java von jedem Verzeichnis ausgehend 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

Verwendet ihr einen Mac, müsst ihr keine Umgebungsvariablen manuell setzen.

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.

Typ	Von	Bis	Beispiel
byte	-128	127	101010
short	-32768	32767	4711
int	-2147483648	2147483647	
long	-9.223.372.036.854.780.000	9.223.372.036.854.780.000	
float	+/- $1{,}4*10^{-45}$	+/- $3{,}4*10^{+38}$	-4,2
double	+/-4,9*10-324	+/-1,7*10+308	
boolean	false	true	21 > 42 (= false)
char	0	65535	'a'

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

Ausgabe: Ergebnis: 3

3. Kontrollstrukturen

Kontrollstrukturen dienen in einer Programmiersprache dazu, Programmteile unter bestimmten Bedingungen auszuführen.
Dabei können Programmteile auch wiederholt ausgeführt werden.
Hier solltet ihr aber darauf achten, dass ihr nicht zu viel redundanten Code produziert.
Merken: DON'T REPEAT YOURSELF!

Nachfolgend sind einige wichtige Kontrollstrukturen aufgelistet.

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);
|              }
|         }
|    }
```

Ausgabe: 3

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 die Aussage wahr und die Ausgabe erfolgt.

If-Else-Anweisung

Hiermit kann man die Bedingungen abfangen.
Ergibt die if- Anweisung ein „false“, wird der else-Teil ausgeführt.
Bei „true“ wird wie im oberen Beispiel der if-Teil ausgeführt.

```
|    public class HelloWorld {
|         public static void main(String[] args) {
|              int operator1 = 1;
|              int operator2 = 2;
|
|              if (operator2 < 1) {
|                   System.out.println(operator1 + operator2);
|              } else {
|                   System.out.println("Ich bin im Else-Zweig.");
|              }
|         }
|    }
```

Ausgabe: Ich bin im Else-Zweig.

While-Schleife

In einer while-Schleife können Programmteile unter einer bestimmten Bedingung wiederholt ausgeführt werden.

```
|    public class HelloWorld {
|         public static void main(String[] args) {
|              int operator1 = 1;
|
|              while(operator1 <= 5) {
|                   System.out.println("Durchlauf" + operator1);
|                   operator1++;
|              }
|         }
|    }
```

Ausgabe: Durchlauf 1
Durchlauf 2
Durchlauf 3
Durchlauf 4
Durchlauf 5

For-Schleife

Die for-Schleife ist eine andere Schleifenvariante.

Jede for-Schleife kann in einer while-Schleife formuliert werden, aber nicht umgekehrt.

```
|    public class HelloWorld {
|         public static void main(String[] args) {
|
|              for (int i = 1; i <= 3; i++) {
|                   System.out.println("Durchlauf" + i);
|              }
|         }
|    }
```

Ausgabe: Durchlauf 1
Durchlauf 2
Durchlauf 3

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

- Initialisierung der Zählvariable → `int i = 1`
- Bedingung → `i <= 3`
- Hochzählen/Runterzählen der Zählvariable → `i++ (i+1)`

In der for-Schleife werden die Anweisung in folgender Reihenfolge ausgeführt:

1. Initialisierung der Zählvariable (nur einmal)
2. Bedingung wird geprüft
3. Ausführung des Programmcodes
4. Hoch- oder Runterzählen der Zählvariable
5. Fortsetzung ab Schritt 2 bis Bedingung false ergibt

BTW: Am Ende jeder Anweisung ist ein Semikolon zu setzten!

4. Kommentare

Kommentare sind ein wichtiger Bestandteil von Programmcode.
Sie dienen zur Dokumentation und können hilfreich sein, schwierigen Programmcode zu verstehen.
Es gibt mehrere Varianten, Kommentare zu schreiben.

```
// Dies ist ein Zeilenkommentar

/*
 *  Dies ist ein Blockkommentar
 *  mit zwei Zeilen.
 */

/**
 * Das ist ein Javadoc-Kommentar.
 */
```

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.

Literatur

- Java ist auch eine Insel openbook.galileocomputing.de/javainsel
- Java von Kopf bis Fuß ISBN-13: 978-3897214484