

1 Hintergründiges

1.1 Überblick über den Sprachstandard

Die Programmiersprache Java wurde im Jahre 1995 von der Firma Sun offiziell veröffentlicht (Auf den Internetseiten der Firma Sun finden sich auch die entsprechenden frei verfügbaren Programmpakete:

`http://java.sun.com/j2se`

Ziel war es eine einfache, leistungsfähige und möglichst universell einsetzbare, so genannte objektorientierte, Programmiersprache zu schaffen.

Ein zentraler Punkt bei der Entwicklung von Java war und ist die Systemunabhängigkeit, d.h. ein Java-Programm soll auf allen Rechnersystemen ohne Änderung ausführbar sein. Neben der eigentlichen Programmiersprache enthält der JavaStandard daher auch Festlegungen zur Gestaltung geeigneter Java-Laufzeitsysteme bzw. Ausführungsumgebungen. Diese Kombination aus Programmiersprache und passender Programmablaufumgebung stellt die sog. JavaPlattform dar. Die Basis von Java ist die Beschreibung der zulässigen Sprachkonstrukte in der Sprachspezifikation. Die Java-Spezifikation liefert eine ausführliche Beschreibung der einzelnen Sprachkonstrukte. Außerdem existieren verbindliche Beschreibungen aller Bestandteile des Application Programming Interface (API) von Java. Das API ist eine Sammlung von Standardbibliotheken, die alle zur Programmierung der Java Plattform benötigten Routinen beinhalten. Aufgrund vieler Erweiterungen der Java-Plattform ist die API-Beschreibung wiederholt aktualisiert worden. Die aktuelle Fassung von Java bzw. der JavaPlattform ist die Version 1.5.

Für viele Rechner bzw. Betriebssysteme existieren inzwischen Realisierungen der Java-Plattform. Im Rahmen dieses Kurses wird das kostenlos verfügbare Java Development Kit (JDK) verwendet. Das JDK ist als Referenz-Implementation der jeweils aktuellen Java-Version und der zugrundeliegenden Java-Plattform konzipiert und existiert für eine Reihe unterschiedlicher Betriebssysteme.

Bestandteile des JDK

Das JDK ist eine komplette Implementierung der Java-Plattform. Daher beinhaltet es neben allen für das API benötigten Bibliotheken auch einen Compiler (`javac`, siehe 1.2 und eine Laufzeitumgebung zur Ausführung der übersetzten Programme (`java`, siehe 1.3). Zusätzlich enthält das JDK einen einfachen Debugger namens `jdb`, der für die Analyse von Abläufen in Java-Programmen entwickelt wurde. Er wird jedoch hier nicht weiter angesprochen werden.

1.2 Compilierung von Java-Programmen

Die Übersetzung von Java-Programmen erfolgt in zwei Stufen: Zunächst wird der Programmtext mit dem Compiler `javac` in einen Zwischencode, den sog. Bytecode, übersetzt. Dieser Bytecode ist dann mit Hilfe des Java-Interpreters `java` ausführbar (siehe (1.3)).

Die Grobstruktur von Java-Programmen ist durch so genannte Klassen vorgegeben (dazu später mehr in Abschnitt (4)). Die Deklarationen von Java-Klassen erfolgen in Form von Textdateien, die durch die Dateiendung `.java` als Java-Quelltexte identifizierbar sind. Zwar kann eine solche Quelltextdatei prinzipiell mehrere Klassendeklarationen enthalten, es empfiehlt sich jedoch aus Gründen der Übersichtlichkeit für jede Klasse eine eigene Quelltextdatei zu verwenden. Die Quelltextdatei wird bei der Compilierung mit `javac` komplett übersetzt. Ein Aufruf des Java-Compilers `javac` hat folgende Form:

```
javac classname.java
```

Wichtig: Die Datei-Endung `.java` darf beim Aufrufen des Compilers nicht weggelassen werden! Die Verwendung einer anderen Datei-Endung als `.java` wird vom Compiler nicht akzeptiert. Da ein Java-Programm im allgemeinen aus mehreren Klassen besteht und diese Klassen voneinander abhängen, können mittels `javac` auch mehrere Java-Quelltextdateien innerhalb eines Compileraufrufs übersetzt werden. Aufrufe der Art

```
javac class1.java class2.java class3.java
```

sind ebenso zulässig wie die Benutzung von sog. Wildcards bzw. Jokerzeichen: `javac class*.java`.

Beispiel: Die folgende Deklaration der so genannten Klasse `HelloWorld` ist bereits ein vollständiges, korrektes Java-Programm: Bei Ausführung des Programms wird mit Hilfe der Methode `System.out.println()` aus einer der Java-Klassenbibliotheken der Text `Hello World!` auf dem Bildschirm ausgegeben:

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

Listing 1: "Hello World"

Wenn man diesen Java-Programmtext in eine Datei `HelloWorld.java` geschrieben hat, kann er mit folgendem Aufruf compiliert werden:

```
javac HelloWorld.java
```

Bei der Compilierung einer Quelltextdatei wird für jede in der Datei enthaltene Klassendeklaration eine eigene Klassendatei erzeugt, die den für die jeweilige Klasse generierten Bytecode enthält. Der Name einer Klassendatei setzt sich aus dem Klassennamen und der Endung `.class` zusammen. Die Compilierung von `HelloWorld.java` liefert demnach die Bytecode-Datei `HelloWorld.class`.

1.3 Ausführung von Java-Programmen

Wie zuvor gezeigt, entstehen bei der Compilierung eines Programms eine oder mehrere Bytecode-Dateien. Um diese Dateien auszuführen, benötigt man den Java-Interpreter. Für den im JDK enthaltenen Interpreter `java` erfolgt ein solcher Aufruf durch

```
java classname
```

Ein Aufruf des in obigem Beispiel gezeigten Programms `HelloWorld` sähe also so aus:

```
java HelloWorld
```

Wichtig: Die Datei-Endung `.class` wird nicht angegeben! Werden zur Programmausführung noch weitere Klassen benötigt, so werden diese vom Interpreter automatisch dazugeladen und müssen daher nicht explizit angegeben werden!

1.4 Entwicklungsumgebungen für Java

Für Java gibt es zahlreiche Entwicklungsumgebungen, so genannte IDE (Integrated Development Environments). Sie sind zum Teil frei im Netz verfügbar. Es empfiehlt sich, zunächst auf Entwicklungsumgebungen zu verzichten und die Compilierung und Ausführung direkt über die Befehlszeile zu starten. Spätestens bei der Implementierung von Java-Programmen mit mehreren Klassen ist die Verwendung einer entsprechenden Umgebung jedoch hilfreich. Folgende Tabelle gibt, natürlich ohne Anspruch auf Vollständigkeit, Anregungen für verschiedene Werkzeuge:

Name	Größe	Status	Adressat
Eclipse	ca. 60 MB	frei	erfahrene Entwickler
JBuilder	ca 50 MB	kommerziell (Teilversion frei)	erfahrene Entwickler
NetBeans	ca. 100 MB	frei, in J2SE 5.0 integriert	erfahrene Entwickler
JCreator	ca. 3 MB	frei („Amateur-version“)	am ehesten für Anfänger geeignet

1.5 Bytecode-Portabilität

Java-Interpreter wie `java` können als Bytecode-Prozessoren angesehen werden, d.h. die Menge aller gültigen Bytecode-Befehle entspricht dem Befehlssatz eines Java-Interpreters. Das Laufzeitsystem von Java emuliert also einen Rechner mit Bytecode-Prozessor. Die Java-Spezifikation bezeichnet diesen Rechner bzw. seine Emulation als Java Virtual Machine (JVM). Mit der Spezifikation der JVM ist auch der Bytecode eindeutig festgelegt. Damit lässt sich der Bytecode unverändert auf allen Rechnern ausführen, auf denen eine JVM (ein Java-Interpreter) existiert. Eine erneute Compilierung des Programmtexts ist nicht erforderlich.

Die JVM ist nicht für einen speziellen Prozessortyp entworfen, lässt sich aber einfach auf verschiedenen Prozessortypen emulieren. Das erleichtert die Implementierung von Java-Interpretern.