Kapitel 02

Java – was, wann, warum, wieso

Fachgebiet Knowledge Engineering
Prof. Dr. Johannes Fürnkranz

TECHNISCHE UNIVERSITÄT DARMSTADT

Java, eine objektorientierte Programmiersprache

Java ist eine objektorientierte Programmiersprache und als solche ein eingetragenes Warenzeichen der Firma Sun Microsystems. Sie ist eine Komponente der Java-Technologie.

- Programmiersprache Java
- Java-Plattform
  - Java Virtual Machine (JVM)
  - Java Runtime Environment (JRE)
  - Java Development Kit (JDK)

Java Logo

Basisdaten	
Entwickler:	Sun Microsystems
Aktuelle Version:	6.0 (11. Dezember 2006)
Betriebssystem:	Windows, Linux, Mac OS X, Solaris
Website:	java.sun.com

Geschichte von Java

Entwicklung seit 1991 von der Firma Sun Microsystems

Ziel: Programmierung von elektronischen Geräten

- Konsumgüter sollten zur Interaktivität befähigt werden
- Arbeitstitel während der Entwicklung: OAK (Object Application Kernel)
- Wegen namensrechtlicher Probleme Umbenennung in Java
  - Java ist eine starke Kaffeebohnensorte

Geschichte von Java

Java lehnt sich in den Äußerlichkeiten stark an die weit verbreitete und ältere Programmiersprachen C und C++ an.

C/C++-Programmierern soll der Umstieg erleichtert werden.

- Höhere Akzeptanz in der Wirtschaft
- Aber es werden auch „Altlasten“ aus C/C++ übernommen
  - Beispiel: Java verwendet genau wie C/C++ für Zuweisungen ‘=‘ und für Gleichheit ‘==‘
  - Andere Programmiersprachen verwenden die intuitivere Variante mit ‘:=‘ für Zuweisungen und ‘=‘ für Gleichheit

Geschichte von Java

Allgemeine Java-Philosophie

In C/C++ hatte der Programmierer maximale Freiheit

- Alles, was nicht ausdrücklich verboten ist, ist erlaubt, egal ob es Sinn macht oder nicht.
- Der Programmierer hat so einen hohen Grad an Selbstverantwortung.

Beim Entwurf von Java ist man davon ausgegangen, dass diese Selbstverantwortung für die meisten Programmierer zu hoch ist.

- Alles, was aller Erfahrung nach keinen Sinn macht, ist von vornherein verboten.

Compiler vs. Interpreter

Ein Programmierer entwickelt ein Programm.

Die Entwicklung findet in einer menschenlesbaren Programmiersprache statt (z. B. Java).

- Der Programmierer erstellt ein oder mehrere Source-Files.
- Diese müssen dann in maschinenlesbaren Code umgewandelt werden.
  - Maschinen verstehen nur Maschinencode (Binärcode).
- Die Umwandlung kann durch einen Compiler oder Interpreter geschehen.

Compiler vs. Interpreter

Compiler

Die Ausführung eines kompilierten Maschinencodeprogramms ist wesentlich schneller als die Abarbeitung durch einen Interpreter.

- Einsatz des Programms in der Praxis.

Kompilieren: Das Source-File wird in ein Maschinenprogramm übersetzt, das danach jederzeit als ein normaler Prozess des Betriebssystems aufgerufen werden kann.

Interpreter

Die Zeit fürs Kompilieren wird eingespart, was bei großen Programmen mitunter sehr lang dauert.

- Verwendung während der Entwicklung des Programms.

Interpretieren: Das Source-File wird ohne vorherigen Arbeitsgang vom Interpreter Schritt für Schritt ausgeführt.

Compiler vs. Interpreter

Im Prinzip kann jede Sprache sowohl kompiliert als auch interpretiert werden.

Aber verschiedene Sprachen sind für das eine oder andere besser geeignet.

- HTML wird interpretiert.
  - Wenn eine HTML-Seite im Browser geladen wird, startet der Browser einen HTML-Interpreter.
  - Dieser Interpreter stellt die Texte und Bilder gemäß den enthaltenen Formatierungsbefehlen dar.
- C/C++ wird kompiliert.
  - C/C++ wird für größere und komplexere Programme verwendet.

Compiler vs. Interpreter

Für Java ist die Frage kompilieren vs. interpretieren strikt geregelt.

Ein salomonisches sowohl als auch.

- Ein Java-Source-File wird nicht in Maschinencode, sondern in sogenannten Java-Bytecode kompiliert.
  - Zum Kompilieren wird das Programm „javac“ verwendet.
    - z. B. `javac Programmierprojekt.java`
  - Zum Abarbeiten des Programm wird dann der Java-Bytecode interpretiert.
    - z. B. `java Programmierprojekt`

Java-Bytecode und die Java-Virtual-Machine

Durch diese Abstraktion von konkreter Hardwareplattform und Betriebssystem kann im Prinzip jedes kompilierte Java-Bytecode-Programm auf jedem System laufen.

Zusätzlich enthält Java-Bytecode noch Informationen, die es z. B. erlauben zu prüfen, ob das Programm nur auf Daten und Ressourcen zugreift, die ihm gestattet sind.

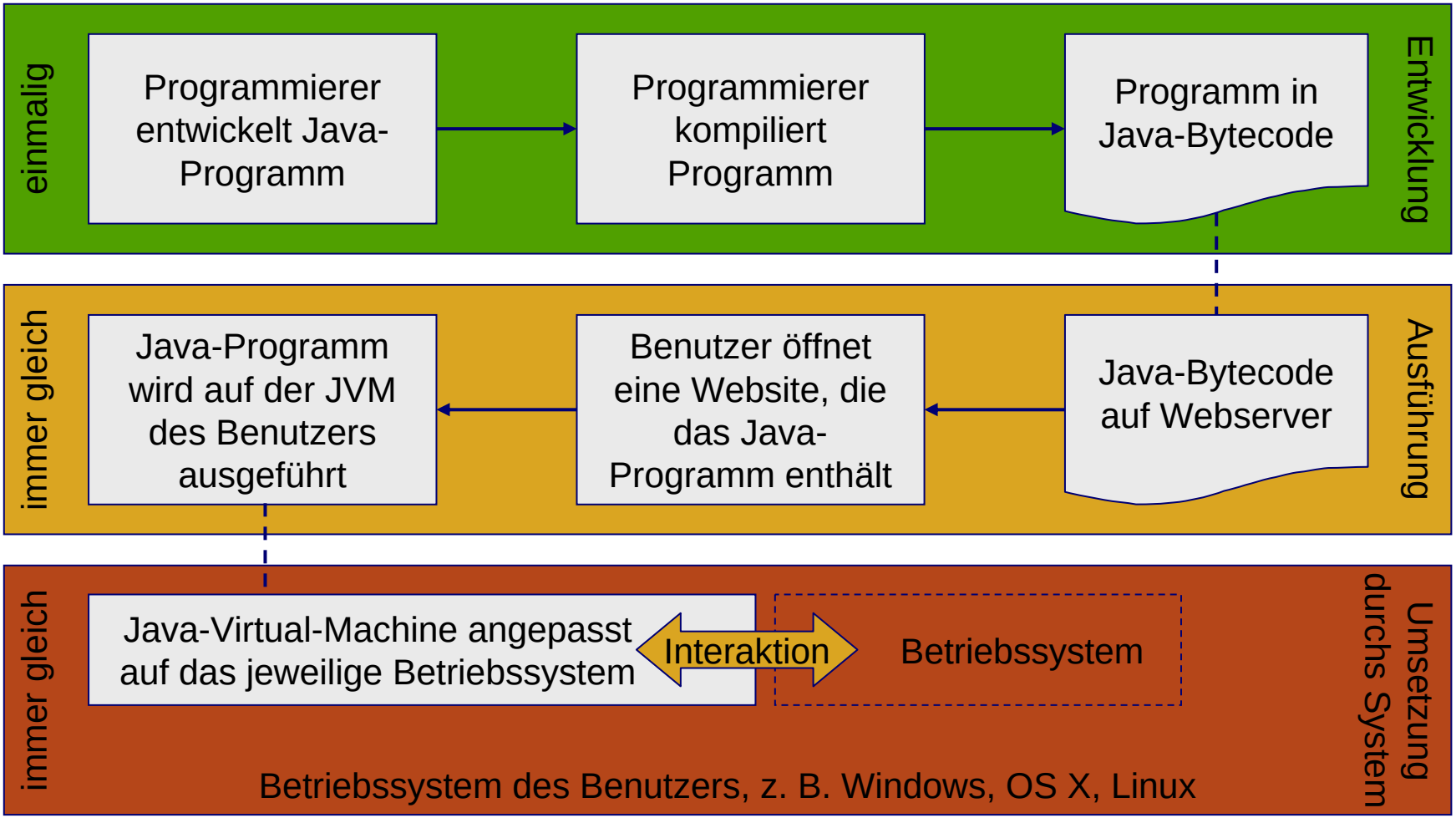
Java-Bytecode

Java-Bytecode ist eine idealisierte und standardisierte Variation von Maschinencode. Diese ist speziell für die Java-Virtual-Machine angepasst.

Java-Virtual-Machine

Die Java-Virtual-Machine ist ein virtuelles Rechnersystem, das auf dem Betriebssystems des Computers läuft und innerhalb dessen der Java-Byte-Code ausgeführt wird.

Java-Bytecode und die Java-Virtual-Machine



Java-Laufzeitsystem

Während der Ausführung eines Java-Programms passieren einige Dinge im Hintergrund.

Das Java-Laufzeitsystem übernimmt Aufgaben, die nicht explizit im Quellcode gestellt wurden, z. B.

- Zugriffsprüfungen
- Abfangen von Ausnahmen wie Division durch 0
- automatische Speicherbereinigung

Java-Laufzeitsystem

Das Java-Laufzeitsystem stellt Dienste zur Verfügung, die während der Programmausführung im Hintergrund automatisch vom Interpreter (hier speziell: der JVM) ausgeführt werden, ohne explizit im Quellcode erstellt worden zu sein und dafür sorgen, dass das Programm sinnvoll und sicher ausgeführt werden kann.

Einsatz von Java-Programmen

Java ist nicht einfach zu erlernen für Programmieranfänger. Warum also Java?

Java ist heute und zukünftig eine der wichtigsten Programmiersprachen wegen ihrer Möglichkeiten der Internetprogrammierung.

- Java-Applets und Java-Servlets
- SAP Netweaver

Bekannte Programme aus AI I und dem Studium

- KarelJ
- BlueJ
- Eclipse

Was braucht man zum Arbeiten mit Java?

Sun Microsystems stellt folgende Bestandteile zur Verfügung:

Eine Ausführungsumgebung zum Ausführen von Java-Programmen.

- Java Virtual Machine (JVM)
- Java Runtime Environment (JRE)

Eine Entwicklungsumgebung zum Erstellen von Java-Programmen.

- Java Development Kit (JDK)

Was braucht man noch zum Arbeiten mit Java?

Zum Erstellen von Java-Programmen benötigt man mindestens einen Editor und die Konsole.

Im Editor schreibt man den Quellcode.

- Auf den Linuxsystemen der RBG startet man einen solchen z. B. durch `gedit MeineKlasse.java`

Anschließend muss der Quellcode kompiliert und ausgeführt werden.

- Der Compiler wird gestartet über `javac MeineKlasse.java` und erzeugt die Datei `MeineKlasse.class`
- Das Programm kann nun mit `java MeineKlasse` ausgeführt werden

Was braucht man noch zum Arbeiten mit Java?

Der etwas elegantere und einfachere Weg führt über eine IDE.

Bekannte IDEs für Java

- KarelJ
- BlueJ
- Eclipse

Die wohl verbreitetste IDE für Java ist Eclipse. Eclipse ist eine sehr mächtige Software für den professionellen Einsatz. Für das Lernen von Java eignet sich BlueJ sehr gut.

IDE

Integrated Development Environment (IDE): ist ein Anwendungsprogramm zur Entwicklung von Software, das in der Regel über folgende Komponenten verfügt: Texteditor, Compiler bzw. Interpreter, Linker, Debugger, Quelltextformatierungsfunktion.

Was braucht man noch zum Arbeiten mit Java?

Um die vielen Möglichkeiten von Java nutzen zu können benötigt man Informationen über die zur Verfügung stehenden Klassen und Funktionen.

Der klassische Weg für erfahrene Programmierer führt über die Java-API (Application Programming Interface).

- Diese ist einsehbar unter http://java.sun.com/javase/6/docs/api/

Ein etwas angenehmerer Weg, Informationen über Klassen und Funktionen von Java zu erhalten führ über Java-Referenz, also Bücher.

- Eine gute Referenz ist „Java ist auch eine Insel“.
- Es gibt eine freie HTML-Version auch zum Herunterladen unter http://www.galileocomputing.de/openbook/javainsel6/