



Java – Erster Überblick

Stephan Karrer

Autor: Stephan Karrer

Designziele von Java, Teil 1

- **Einfach und vertraut:**
Orientiert an C/C++, aber drastisch gesäubert! Konzepte aus Smalltalk, Ada, ...
- **Objektorientiert:**
Kapselung, Geheimnisprinzip, Vererbung, Polymorphismus
- **Basis für verteilte Client-Server-Systeme:**
Bereitstellung umfangreicher Klassen-Bibliotheken und Mechanismen für die Erstellung verteilter Anwendungen
- **Robust:**
Strenge Typprüfung zur Übersetzungszeit durch den Java-Compiler, keine Zeigerarithmetik, automatische Speicherbereinigung (Garbage Collection), Laufzeitprüfung (Ausnahmebehandlung)
- **Sicher:** Zugriff auf Ressourcen außerhalb der Laufzeitumgebung ist steuerbar

Designziele von Java, Teil 2

■ Architekturneutral und portabel:

- verteilt wird Byte-Code, der an der Client-Maschine interpretiert wird (architekturneutral)
- Ein integer in Java ist auf jedem System eine 32-Bit-Zahl (in C nicht ...)
- Unicode für die Codierung von Zeichen, ...

■ Leistungsfähig:

- Interpreter verlässt sich auf gewisse Prüfungen und wird damit schneller
- Automatische Speicherbereinigung läuft als Hintergrund-Thread mit niedriger Priorität: Speicher ist vorhanden, wenn man ihn braucht
- Schnittstelle zu Maschinencode vorhanden (in C)

■ Multi-Threading-fähig:

Java unterstützt Threads auf Sprachebene (Thread-Klasse), im Laufzeitsystem (Bausteine zur Synchronisation) und in den Bibliotheken (Thread-sichere Routinen).

Java Technologie: Spezifikation

- seit 1998 werden die Spezifikationen im sogenannten Java Community Process (JCP) entwickelt

- Es werden 3 verschiedene Plattformen (Frameworks mit bestimmten Umfang) definiert:
 - **JSE** (Java Platform Standard Edition)
 - **JEE** (Java Platform Enterprise Edition)
 - **JME** (Java Platform Micro Edition)

- Sun liefert Umsetzungen der Spezifikation als sog. Java Development Kits (JDKs) nur für wenige Plattformen:
 - Solaris
 - Linux (bis Java 6)
 - WinXXX

- Alle anderen Hersteller implementieren die Spezifikation selbst in ihre Produkte

Entwicklung

| | |
|-------------|-------------------------------------|
| 1991 – 1995 | Entwicklung der ersten Java-Version |
| 1996 | JSE 1.0 |
| 1997 | JSE 1.1 |
| 1998 | JSE 1.2 |
| 2000 | JSE 1.3 |
| 2002 | JSE 1.4 |
| 2004 | Java 5.0 |
| 2006 | Java 6.0 |
| 2011 | Java 7 |
| 2014 | Java 8 |

Stand der Dinge

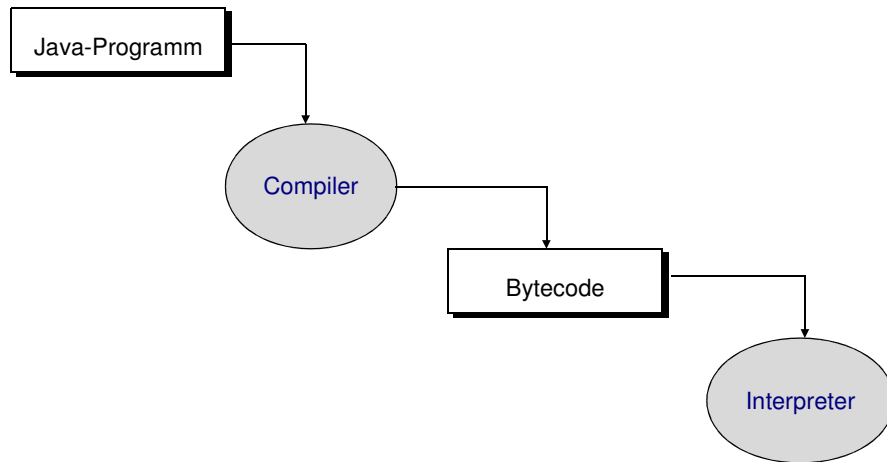
Aktuelle Java-Versionen: JSE 1.8.x (ab März 2014)
JSE 1.7.x (ab August 2011)
JSE 1.6.x (ab Dezember 2006)

JEE 7.x (ab Mai 2013)
JEE 6.x (ab Dezember 2009)
JEE 5.x (ab Mai 2006)

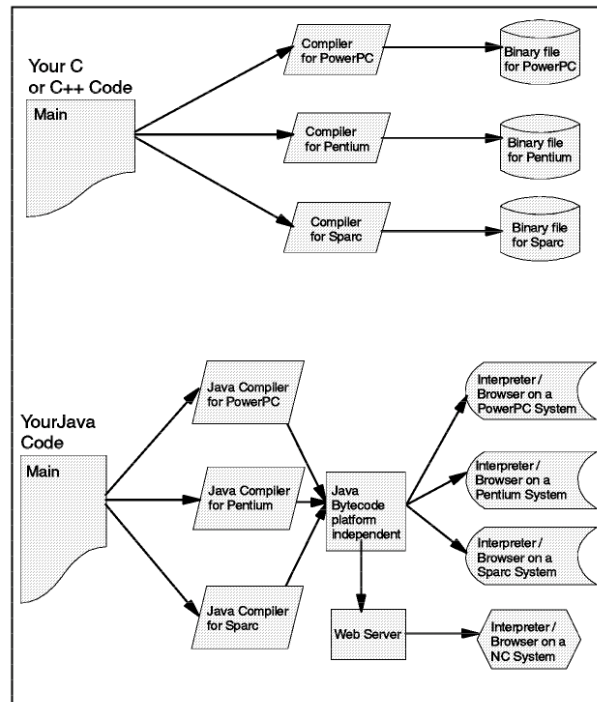
Plattformen:

- MacOS
- Winxxx
- Solaris (SPARC und Intel)
- AIX
- HP-UX
- Linux
-

Grundgedanke bei Java für die Übersetzung und Ausführung



Vergleich:
C- versus
Java-Compiler



Java Runtime Environment (JRE)

| Java-Software als Bytecode | | | |
|--|------------|-----------------------------------|-----|
| Java Laufzeitumgebung (Java Runtime Environment, JRE) | | Virtuelle Maschine (Interpreter) | |
| | | Standardisierte Klassenbibliothek | |
| Windows | Unix/Linux | Mobiles Gerät | ... |

Dynamisches Binden und Laden

- Statt Linker --> Klassenlader
- Laden ist inkrementell, leichtgewichtig
- Laden ist angepaßt an Netzwerkumgebung
- Laden ist ideal bei häufigen Änderungen
- Java-Compiler löst Referenzen nicht bis zu numerischen Werten (Offsets) auf (Java-Interpreter löst Referenzen einmalig zu Offsets auf beim Einbinden der Klasse)
- Speicheranordnung der Objekte bestimmt der Interpreter (nicht: Compiler). "Einspielen" neuer Klassen

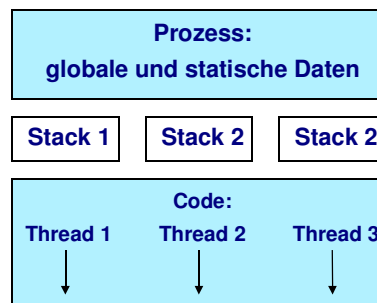
Informationen zur Laufzeit

- Jedes Objekt enthält Verweis auf ein Class-Objekt
- Zur Laufzeit abfragbar: Klassenname, -typ u.ä.
- Über das Reflection-API können weitere Informationen zu den Klassen und Methoden zur Laufzeit abgefragt werden
 - dies ermöglicht leistungsfähige Debugging- und Monitoring-Möglichkeiten
 - kann genutzt werden um dynamisch zur Laufzeit auf Methoden zuzugreifen (die vorab noch nicht bekannt waren)

Speicherverwaltung

- alle Objekte werden dynamisch angelegt (Heap)
- Daten der Basistypen werden ebenfalls dynamisch angelegt
- keine explizite Speicher-Allokierung (wie malloc, calloc o.ä. in C/C++) sondern Objekt-Instantiierung mit new
- Automatische Speicherbereinigung (Garbage Collection)
 - "Wenn keiner mehr darauf verweist: vernichten"
 - "Wenn eine Verweisvariable ihren Gültigkeitsbereich verläßt: Verweis löschen"
- Garbage Collector:
 - Thread niedriger Priorität

Thread-Konzept



Vorteile:

- effizienter Kontextwechsel
- gut für Serverprozesse
- gut abbildbar auf symmetrische Multiprozessor-Architektur
- effiziente Interthread-Kommunikation

Nachteile:

- weniger robust als Multiprozessansatz
- Synchronisation ist eigenes Thema
- Abbildung von "User Level Threads" auf "Kernel Threads" kann auch suboptimal sein
- löst auch nicht die Frage "Was ist parallelisierbar?"

Thread-Konzept

- Multithreading in die Sprache eingebaut
- Muß nicht notwendigerweise durch das Betriebssystem unterstützt werden, wäre aber günstig
- Zur Verfügung stehen:
 - Klassen Thread, ThreadGroup
 - Operationen start(), interrupt(), join(), yield(), setPriority(), ...
 - Schlüsselwort synchronized
 - Methoden notify() und wait()
- Programmierung vergleichsweise einfach

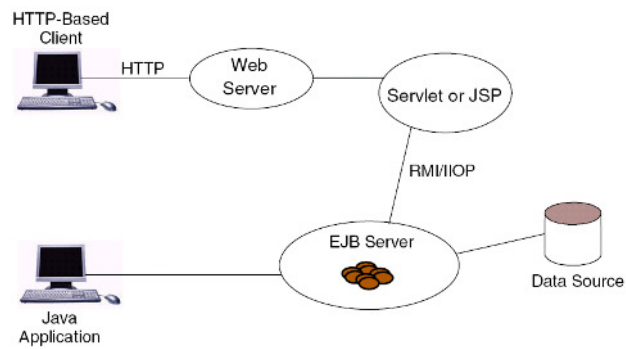
Sicherheitskonzept

Sicherheitsebenen:

- **"Dach"**: Java-Klassenbibliothek mit dem Sicherheitsmanager
- **"1. Obergeschoß"**: Klassenlader
- **"Erdgeschoß"**: Virtuelle Java-Maschine mit dem Bytecode-Verifizierer
- **"Keller"**: die Programmiersprache Java selbst

Java auf der Serverseite

- **Servlets:**
Serverseitiges Äquivalent zu Applets
- **Java Server Pages:**
Eigene Skriptsprache zur Ergänzung
- **Enterprise Java Beans:**
Applikationslogik-Bausteine die innerhalb eines J2EE-Containers laufen



- **Wichtige Server-Produkte:**
 - Apache Web-Server
 - Apache Tomcat: Servlet-Engine
 - RedHat JBoss, Oracle Weblogic, IBM Websphere: Komplette J2EE-Applikationsserver

Wichtige Entwicklungsumgebungen

- Eclipse (vormals IBM VisualAge, jetzt Open Source)
- NetBeans (von Sun/Oracle frei verfügbar)
- IntelliJ IDEA
- JDeveloper (Oracle)
- Xcode (Apple)
- Rational Application Developer (IBM)
- ...