Einführung in die Programmierung mit Java

Martin Wirsing

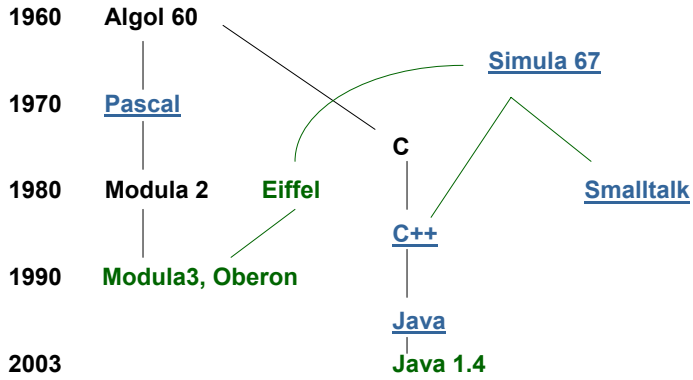
in Zusammenarbeit mit
Matthias Hölzl, Piotr Kosiuczenko, Dirk Pattinson

08/04/03

Ziele

- Geschichte der OO-Programmiersprachen
- Warum Java als Programmiersprache verwenden?
- Ein einfaches Java-Programm erstellen, übersetzen und ausführen
- Gut dokumentierte Java-Programme erstellen können

Entwicklung objektorientierter Programmiersprachen



Java

- Entwickelt von J. Gosling, u.a.
- Erste plattform-unabhängige OO-Sprache, insbesondere zur Programmierung von Internet-Applikationen
- Erste Version 1.0 1995, heute Java 1.4.
- Ursprünglicher Name: OAK.

Aspekte von Java

- Objektorientiert: Klassenkonzept, strenge Typisierung
- Unabhängig von Plattform: Durch Übersetzung in Virtuelle Maschine (JVM)
- Netzwerkfähig, nebenläufig
- Sicherheitskonzept

- **Nachteile:** Laufzeithandicap durch Interpretation der JVM
  (aber z.T. ausgeglichen durch Just-in-Time Übersetzung)
- **Vorteile:**
  - Verteilte Anwendungen, Web-Anwendungen
  - Rechnerunabhängigkeit von Graphikanwendungen

Grober Aufbau eines Java-Programms

- **Java-Programm** besteht aus Menge von **Klassen**

- Eine **Klasse** besteht aus
  - **Attributen** („fields“): Beschreiben **Zustand eines Objekts**
  - **Methoden**: Beschreiben die **Operationen**, die ein Objekt ausführen kann

Einfaches imperatives Java-Programm

- Ein **imperatives** Java-Programm besteht aus
  - **Klassendeklaration** mit einer
  - **einzigen Methode** namens **“main”:**

```
public class <KlassenName>
{
        public static void main(String[] args)
        {
                <Anweisungen>
        }
}
```


Beispiel: Hallo

```
public class Hallo
{
        public static void main(String[] args)
        {
                System.out.println(„Hallo!“);
        }
}
```

Methodenaufruf

- **Methodenaufruf allgemein:**

  object.methodName(parameters);

- **Beispiel:**

  `System.out.println(„Hallo!“);`

Konventionen

- Klassennamen beginnen mit großen Buchstaben

  Bsp. Klasse Hallo
- Methodennamen und Variablennamen beginnen mit kleinen Buchstaben

  Bsp. println, out
- Konstantennamen bestehen nur aus großen Buchstaben.

  Bsp.
- Zusammengesetzte Namen werden zusammengeschrieben,

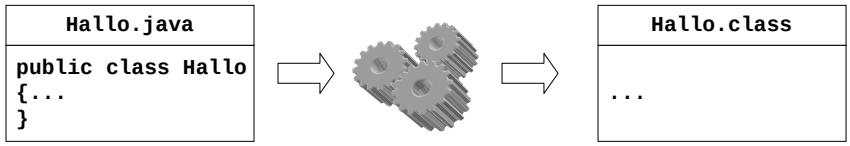
  jeder (innere) Teilname beginnt mit einem großen Buchstaben

  Bsp. Klasse HalloWelt, Methoden getName, getMyObject

Übersetzung und Ausführung von Java-Programmen

Übersetzung in Bytecode

- Aus einer Textdatei mit Endung „.java“ erzeugt der Compiler javac eine Datei mit gleichem Namen, aber Endung „.class“
- Diese enthält den Bytecode für die JVM



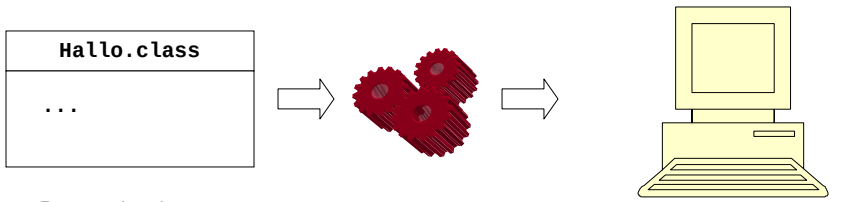
Quellprogramm als Textdatei Hallo.java

Compiler javac.exe

Bytecode als Datei Hallo.class

Übersetzung und Ausführung von Java-Programmen

- Die Datei mit dem Bytecode wird der JVM übergeben und von der JVM ausgeführt (d.h. interpretiert).



Bytecode als Datei Hallo.class

Java Virtual Machine java.exe

Ergebnisausgabe (von Hallo auf Bildschirm)

Übersetzung und Ausführung von Hallo.java (unter Windows)

Übersetzung von Hallo.java:

```
C: > javac Hallo.java
```

Interpretation von Hallo.class:

```
C: > java Hallo
```

Gibt auf Bildschirm zurück:

```
Hallo!
```


Kommentare in Java

„The view that documentation is something that is added to a program after it has been commissioned seems to be

wrong in principle, and counterproductive in practice.

Instead, documentation must be regarded as an integral part of the process of design and coding. „

C. A. R. Hoare:

Hints on Programming Language Design (1973)

Darstellungen für Kommentare in Java

- Durch

  ```
  // bla, bla}
  ```

  wird eine Zeile oder ein Rest einer Zeile zum Kommentar.

- Zur Erzeugung von Kommentaren zu Klassen und Methoden werden die Klammern

  `/**` und `*/`

  verwendet.

  Solche Kommentare werden in den mit dem Befehl **`javadoc`** erzeugten Report mit aufgenommen.

Die Klasse Hallo dokumentiert

```
 /**
Diese Klasse dient nur zum Ausdrucken des Strings "Hallo,
Welt!„ auf den Bildschirm
 */
public class HalloDoc
{      /**    Die Methode main druckt …
        */
       public static void main (String[] args)
       {      System.out.println("Hallo, Welt!");
       }
}
```

Erzeugung der Dokumentation

- Mit dem Befehl

```
javadoc Hallo.java
```

wird automatisch eine Beschreibung der Klasse `Hallo` erzeugt und in die Datei

```
Hallo.html
```

geschrieben.

Spezielle Variablen bei javadoc

- `@see` für Verweise
- `@author` für Namen des Autors
- `@version` für die Version
- `@param` für die Methodenparameter

Hallo Welt Applet - erweitert dokumentiert

```
 /** Diese Klasse ist die Applet-Version von Hallo zur
Demonstration von javadoc.
@see java.applet.Applet
@author Martin Wirsing
@version 1.1
 */
public class HalloApplet extends Applet
{       /**  Diese Methode dient nur zur Illustration der
        Parameterbehandlung durch javadoc.
        @param value ist ein Eingabeparameter
         */
        public void m (int value) {   ...   }
}
```


Zusammenfassung

- **Geschichte:**
  - Objektorientierte Programmiersprachen seit 1967: Simula
  - OO-Programmierung populär seit Ende der 80er Jahre mit Smalltalk und C++.
  - Heute vor allem C++ und Java
- **Java**
  - OO-Programmiersprache,
  - vor allem zur Programmierung im Internet eingesetzt
  - Java ist plattformunabhängig, interpretierend, unterstützt Sicherheitskonzepte und besitzt eine reichhaltige Klassenbibliothek (API, engl. "Application Programming Interface").

Zusammenfassung (2)

- Ein **Java-Programm** besteht aus einer oder mehreren **Klassen**.
  - Klassen enthalten **Attribute** und die Definitionen von **Methoden**.
  - Eine Methode besteht aus einer **Sequenz von Anweisungen**, die den Berechnungsablauf festlegen.
- Jede **selbstablaufende** Java-Anwendung enthält eine Methode „**main**“.
- Ein Java-Programm wird mit einem **Übersetzer in Byte-Code** übersetzt, der dann mit einem **Interpreter**, der **JVM** , ausgeführt wird.
- Java-Programme sollten gut dokumentiert werden. Mit **`javadoc`** kann automatisch eine übersichtliche Dokumentation erzeugt werden.