Java Intro

Frank Kargl
Chaos Computer Club Ulm
frank.kargl@informatik.uni-ulm.de

Überblick

- Features
- Java Historie
- Die Java 2 Plattform
  - » Standard Edition J2SE
  - » Enterprise Edition J2EE
  - » Micro Edition J2ME
- Sprache und Programmierung
- Zusammenfassung / Bewertung

The Network is the Computer

- Java - Hype
- Zunächst nur zur Animation von Web-Pages mit Applets
- Heute ganze IDEs in Java
- The Java Revolution by 

- Unterstützung von allen namhaften Herstellern (exkl. M$)

The Network is the Computer

- Heute
  - » Standalone PCs/große Applikationspakete
  - » Separate Installationen pro PC
  - » Hoher Administrationsaufwand
- Morgen ?
  - » Network Computer (NC) + zentraler Server
  - » Software (in Java) nur auf Server
  - » Download ‚just-in-time‘

Portabel

Schnell

Robust

Einfach

Sicher

Dynamisch

Objektorientiert

Portabel

Schnell

Robust

Einfach

Sicher

Dynamisch

Objektorientiert

Portabel

- Java-Virtual-Machine interpretiert plattformunabhängigen Bytecode
- Java Datentypen und Standard-APIs sind nicht maschinenspezifisch

Portabel

Schnell

Robust

Einfach

Sicher

Dynamisch

Objektorientiert

Robust

- keine Pointers
- automatisches Memory-Management mit Gargabe Collection
- automatisches Array-Bounds-Checking

Portabel

Schnell

Robust

Einfach

Sicher

Dynamisch

Objektorientiert

Sicher

- Bytecode-Verifier
- Run-Time Memory Layout
- Klassen haben getrennte Adreßbereiche
- Sandbox - Prinzip
- Sicherheitssystem
- Signaturen

Portabel

Schnell

Robust

Einfach

Sicher

Dynamisch

Objektorientiert

Objektorientiert/Dynamisch

- Ähnlichkeit zu C++ / Objective C
- komplett Objektorientiert
- dynamisches Laden von Klassen zur Laufzeit
- eigene Classloader

Portabel

Schnell

Robust

Einfach

Sicher

Dynamisch

Objektorientiert

Einfach

- einfach zu lernen, da Syntax ähnlich C++
- einfach zu programmieren, da viele Fehlerquellen eliminiert
- einfach zu programmieren, da Vielzahl von Bibliotheken verfügbar
- einfache Fehlerbehandlung

Portabel

Schnell

Robust

Einfach

Sicher

Dynamisch

Objektorientiert

Schnell ?

- Eingebautes Multithreading
- Effizienter Bytecode
- Just-In-Time Compilation / HotSpot
- Native Code

Java Historie

April 1991	Green Project von Sun
	Oak - Plattform-unabhängige Sprache für Consumer Electronics
	James Gosling
1993	First Person Inc.
	Set-Top-Boxes und Video-on-Demand
1994	First Person Inc. aufgelöst
Sep. 1994	Naughton und Payne - WebRunner
	(später HotJava)

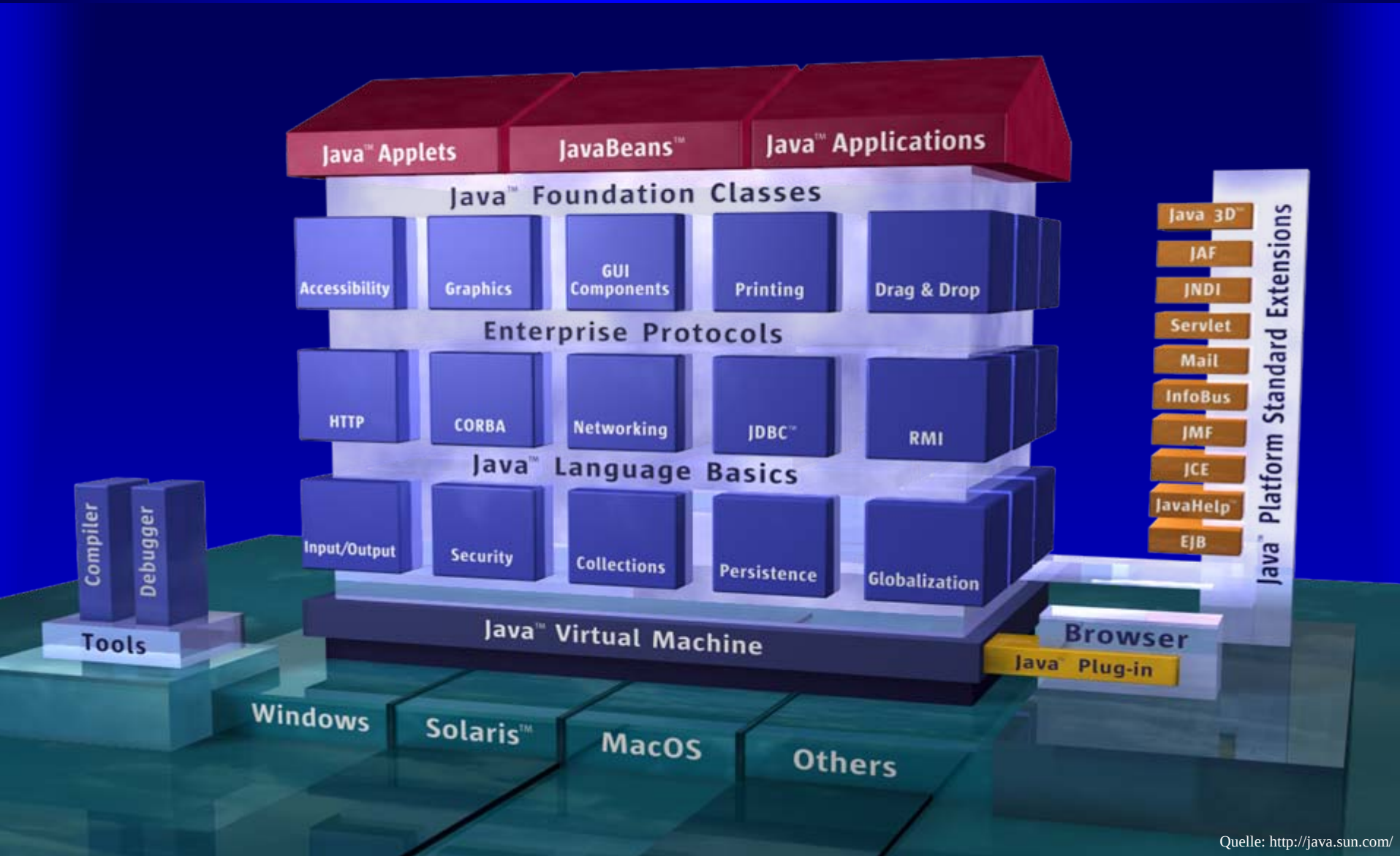
Java Historie

Anf. 1995	Arthur van Hoff implementiert Java-Compiler in Java
23.5.95	Java wird offiziell angekündigt
	Netscape und M$ bauen Java in ihre Web-Browser ein.
Anf. 1996	Java 1.0 released
Feb. 1997	Java 1.1
	Lokalisierung, Security API, AWT Erweiterungen, Beans, JAR, RMI, Serialisierung, Reflection, JDBC, JNI

Java Historie

Anf. 1999	JDK 1.2
	Security, Swing, 2D, Drag&Drop, Collections, Referenzen, Corba, Servlets ...
Mitte 1999	Java 2 Plattform

Java 2, Standard Edition

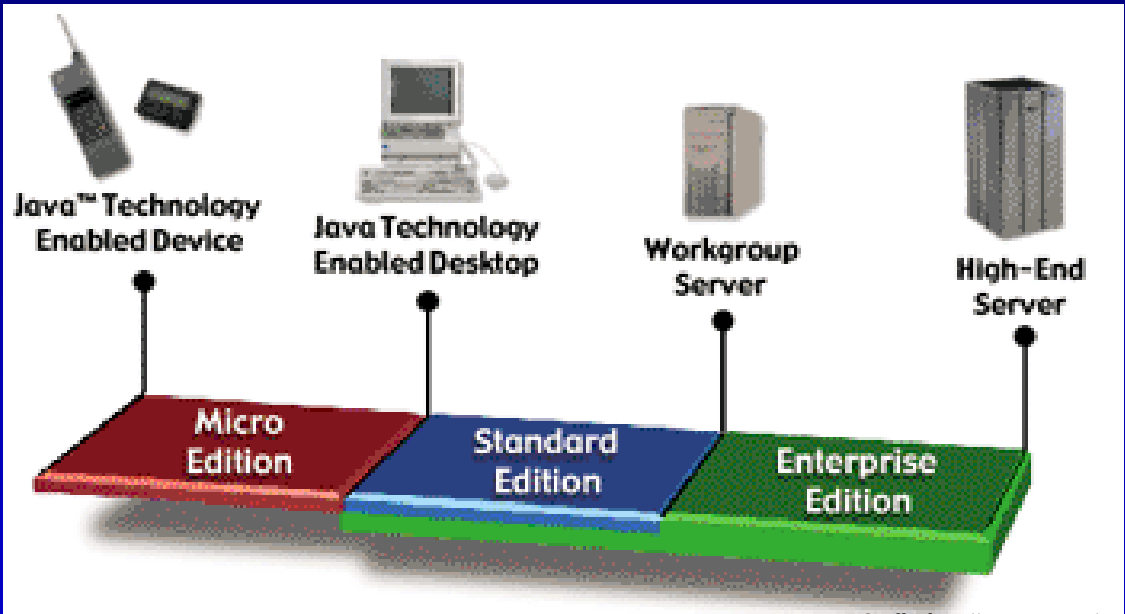


Quelle: http://java.sun.com/

Java 2 Plattform

„[...] the end result has left a veritable alphabet soup of acronyms to describe various features. Consequently, we are reorganizing the technologies to better reflect the markets they address.“

Alan Baratz, President of Software Products and Platforms at Sun Microsystems



Quelle: http://java.sun.com/

Java 2 Plattform

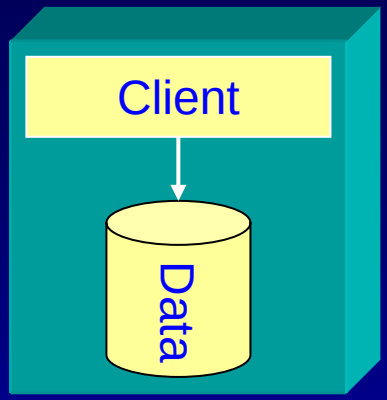
Jede Edition besteht aus:

- » Java Virtual Machine
- » Java Programming Language
- » Core Packages
- » Optional Packages
- » Profiles (z.B. Wireless Profile der J2ME)

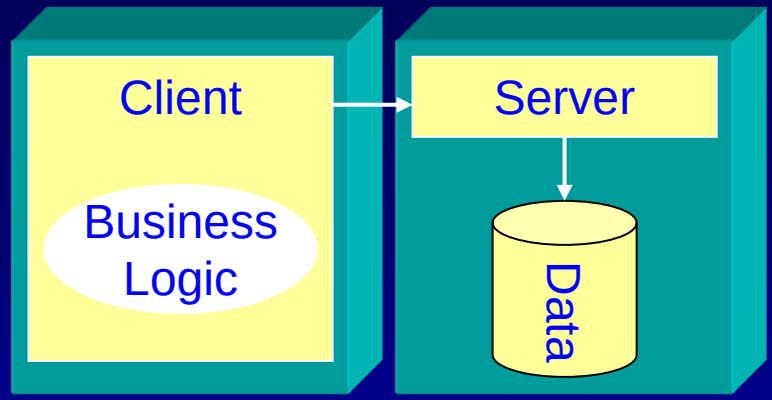
Java 2, Enterprise Edition

- Entwicklung von verteilten Business-Applikationen
  - Hochverfügbarkeit
  - Gute Wartbarkeit
  - Sicherheit
  - Zuverlässigkeit
  - Skarlierbarkeit
- Architekturmodelle
  - One-Tier
  - Two-Tier
  - Multi-Tier

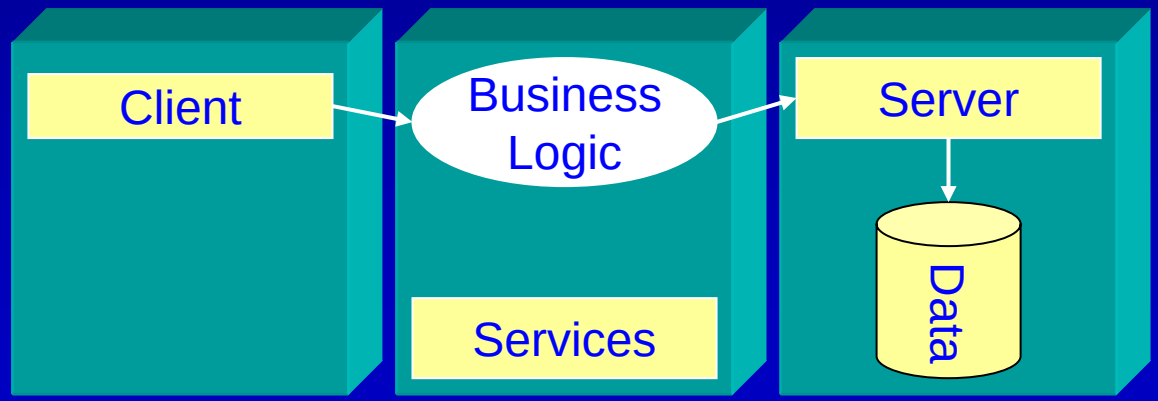
Architekturmodelle



One-Tier



Two-Tier



Multi-Tier

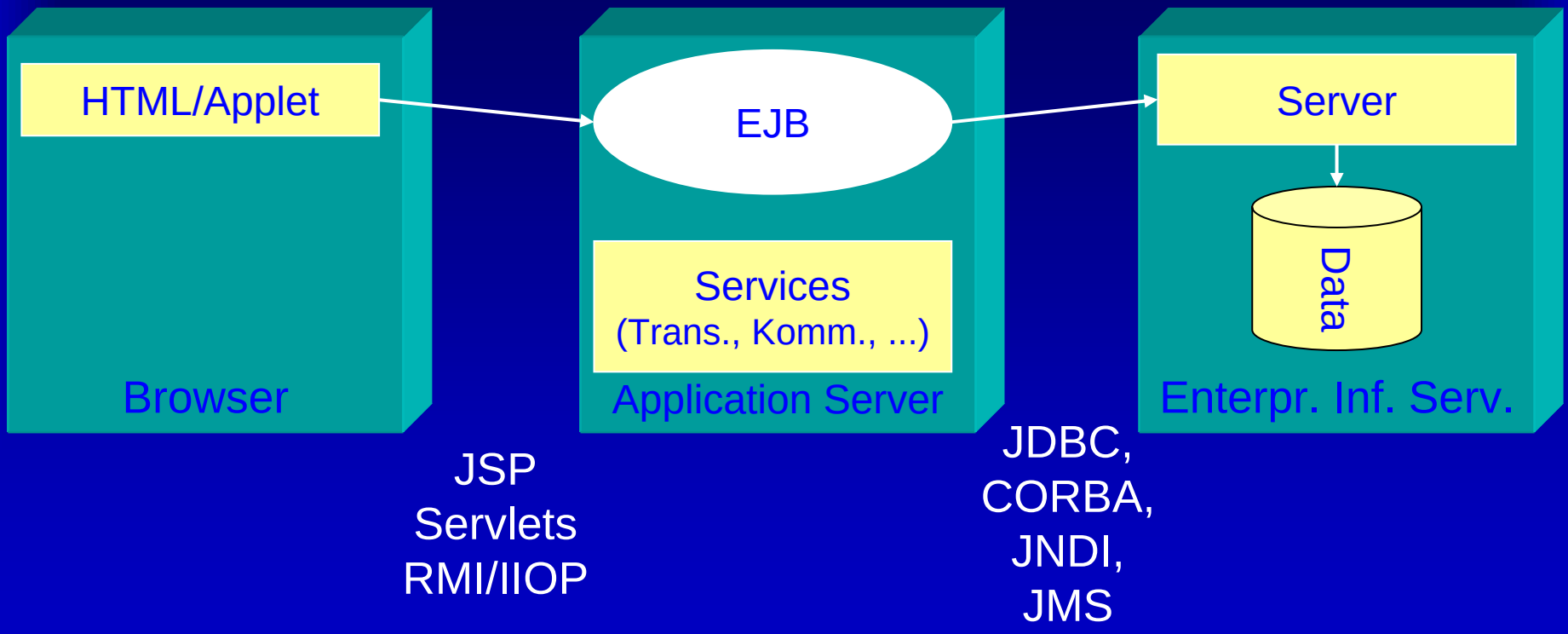
Multi-Tier

Probleme:

- » proprietärer Ansatz
- » hoher Integrations- und Portierungsaufwand

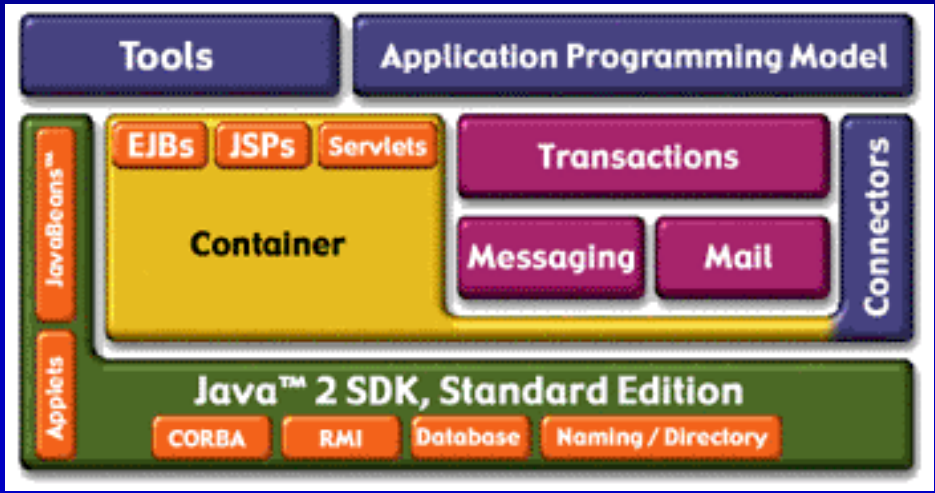
„The J2EE application model defines an architecture for implementing services as multi-tier applications that avoid these problems and deliver the scalability, accessibility, and manageability that is needed.“

J2EE Application Model



Bestandteile

- J2EE Application Programming Model
- J2EE Platform
- J2EE Reference Implementation
- J2EE Compatibility Test Suite



Quelle: http://java.sun.com/

Java 2, Micro Edition

- Ziel:
  - » Consumer-/Embedded-Applications
  - » Smart Card, Pager, Mobiltelefon, Digitale Set-Top-Box, Bordcomputer
- Probleme:
  - » Speicher
  - » Verteilung / Bandbreite

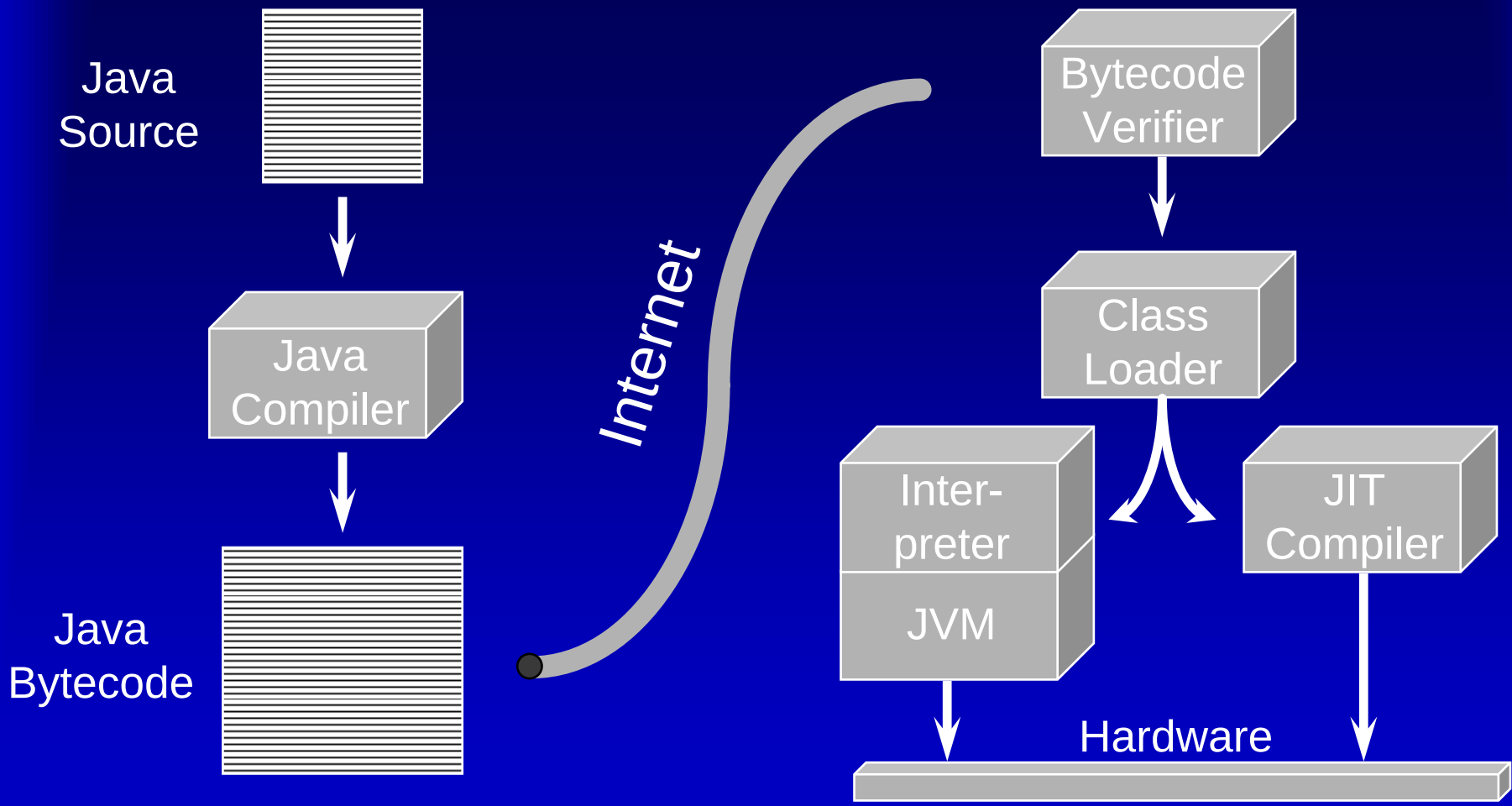
Java 2, Micro Edition

- Bestandteile
  - » Java VM
    (JavaCardVM, KVM, Standard VM)
  - » API Library
  - » Tools
    (Deployment, Configuration)
  - » Profiles
    (minimales API für Set-Top, Screenphone, Wireless, Car, ...)

Unterschiede C++/Java

- keine Header Files
- keine struct oder union
- kein Präprozessor
- keine Pointer
- keine Mehrfachvererbung
- keine alleinstehenden Funktionen (alles ist eine Klasse)
- kein goto
- kein Operator Overloading
- keine automatische Typumwandlung

Erstellung von Java-Programmen

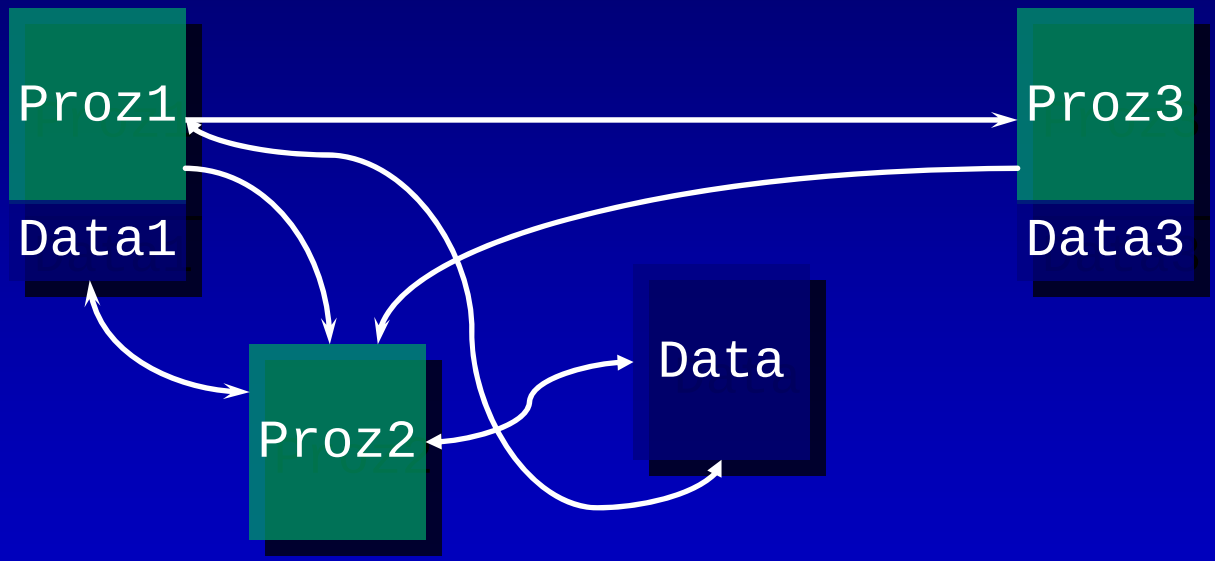


Java Programme ...
Applications, Applets, Beans, Servlets

- Applications
  - » stand alone, kein Browser
- Applets
  - » Ablauf im WWW-Browser
  - » Sicherheits-Restriktionen - Sand-Box-Prinzip
- Beans
  - » wiederverwendbare Komponenten
- Servlets
  - » Erweiterungen von WWW-Servern
- Enterprise Java Beans
  - » Business Components

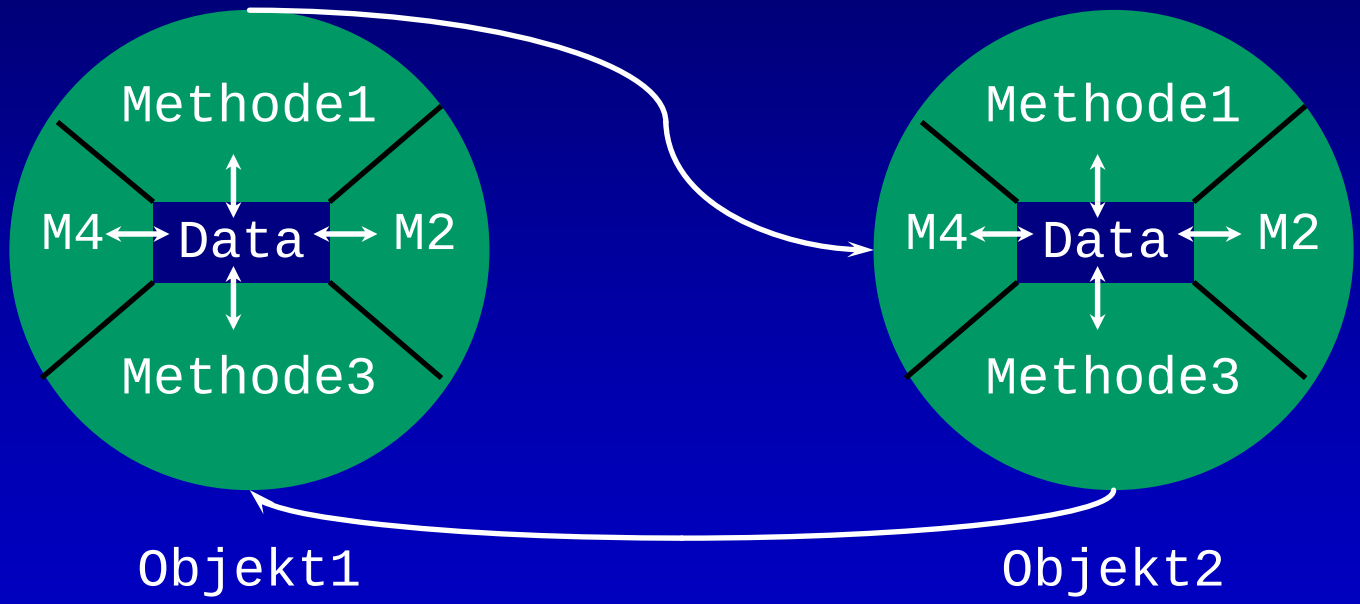
Objektorientiertes Programmieren (OOP)

- Bekannt:
  Prozedurorientieres Programmieren



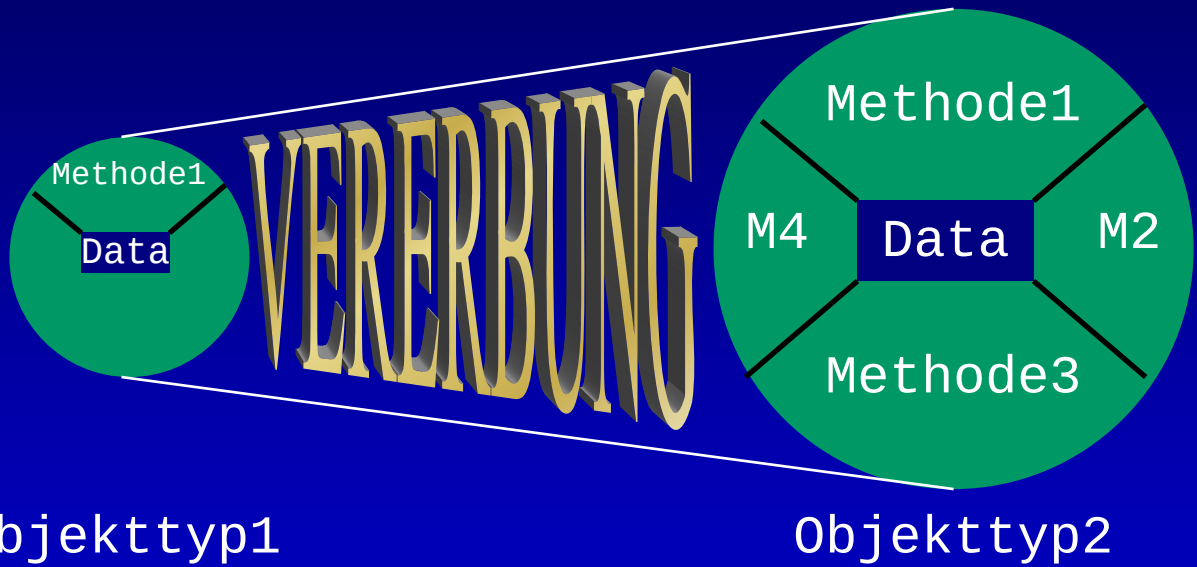
OOP

- Verbesserung:
  Abstrakte Datentypen



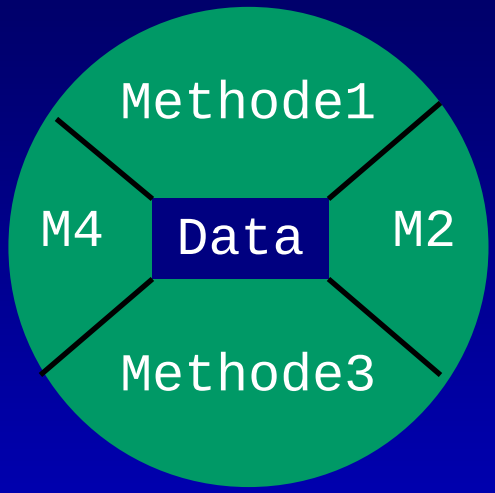
OOP

- Wiederverwendbarkeit und Vererbung

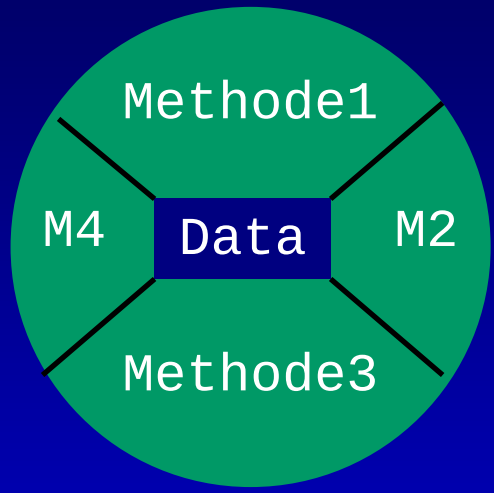


OOP

- Instanziierung



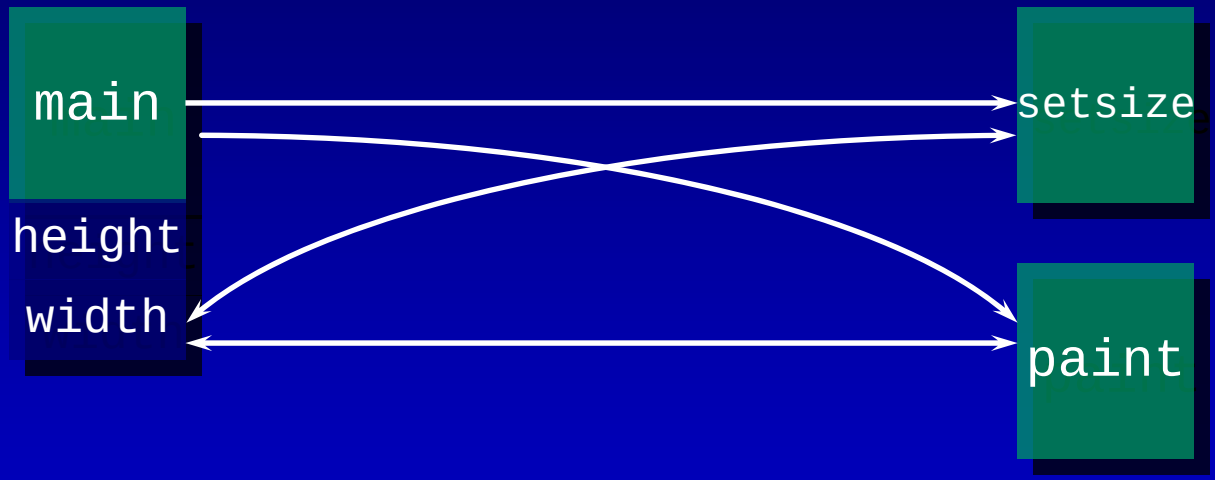
Instanz1



Instanz2

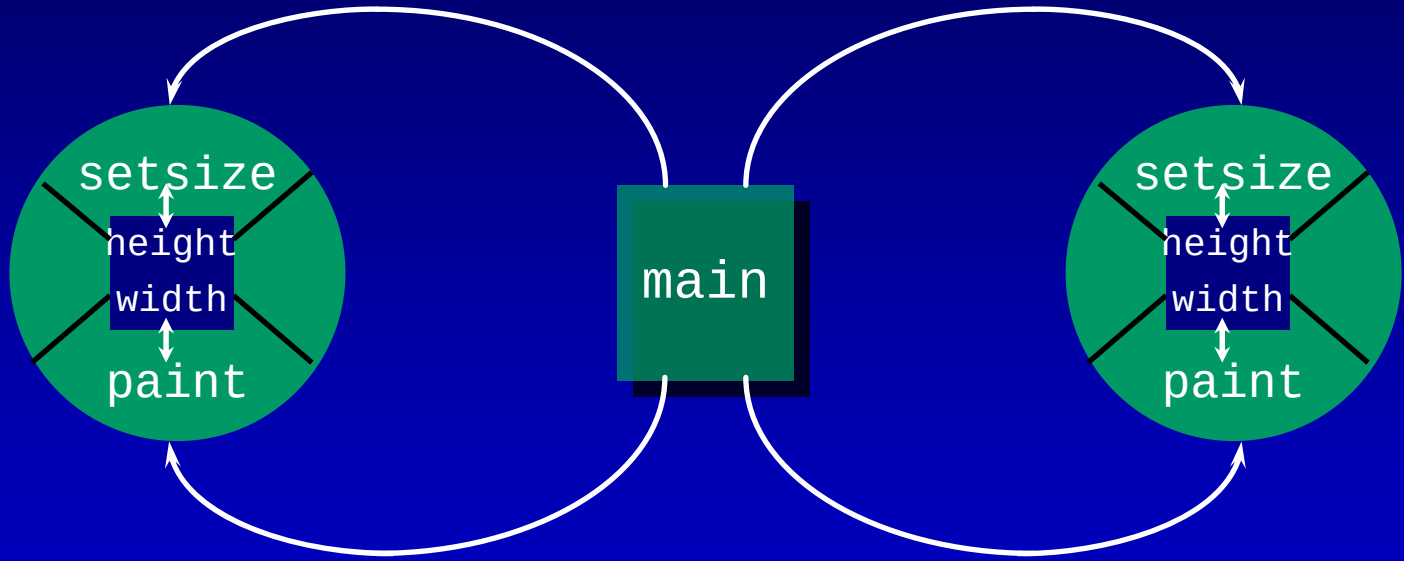
OOP

- Beispiel:
  rectangle - prozedurorientiert



OOP

- Beispiel:
  rectangle - objektorientiert



OOP

- Wiederverwendbarkeit und Vererbung

OOP - Begriffe

- *Klasse* - Zusammenfassung von Daten und Methoden
- *Instanz* - eine Realisierung einer Klasse zur Programm-Laufzeit
- *Methode* - Funktion in einer Klasse
- *Konstruktor* - Methode, die bei jeder Instanziierung aufgerufen wird
- *Destruktor* - Methode, die bei der Löschung eines Objekts (durch GC) aufgerufen wird
- *Overloading* - mehrere Methoden mit gleichem Namen aber unterschiedlicher Signatur

Die erste Applikation

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

Die erste Applikation

Definition der Klasse
HelloWorld

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

Die erste Applikation

```
public class HelloWorld {
  public static void main(String[]
  args){
      n(”HelloWorld!”);
  }
}
```

Die ***main()*** Funktion startet Applikationen

Die erste Applikation

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

System.out.println()
zur Textausgabe

Applets

- als Subklasse der Klasse Applet
- Überschreibt bestimmte (vorgegebene Methoden)
- wird in HTML-Text eingebunden

Demo !!!

Bemerkungen

- werden durch main() Funktion gestartet
- main() ist immer Teil einer Klasse
- jede Klasse in eigener Datei (Endung .java)
- keine Trennung von Deklaration und Implementierung wie bei C/C++

Zusammenfassung

- Java 2
  - » VM
  - » Sprache
  - » APIs
- Unterteilung in Editionen
  - » Standard (Desktop)
  - » Enterprise (Multi-Tier Business Applications)
  - » Micro (Consumer/Embedded)

Bewertung

- Java 2 adressiert breiten Bereich von Anwendungen
- Unterteilung notwendig, um minimale Funktionalität auf jeder Plattform festzulegen
- Gewährleistet Portabilität und Skalierbarkeit der Anwendungen
- Stabilisierung der „API Explosion“

Literatur

- Java Tutorial
  http://www.uni-ulm.de/admin/doku/javatutorial/index.html
- O'Reilly, Java in a Nutshell
- Addison-Wesley, Java Book Series
- http://www.javasoft.com/

The End

Fragen oder Anmerkungen?