Java und intelligente Agenten

Rainer Stoll, Begzudin Omerović
Fachbereich Informatik

28. April 1997

Übersicht

- Was ist Java ?
- Java-Eigenschaften
- Das Java-API
- Java für Agenten ?
- Agenten Toolkits
  - FTP Software Agent Technology
  - MOLE
  - Java-To-Go
  - IBM Intelligent Agents

Was ist Java ?

Unter **Java** versteht man:

1. die **Programmiersprache**
   - C/C++ ähnlich
   - Objektorientiert

2. die **Programmierplattform**
   - Java Virtual Machine (JVM)
     – Virtueller Rechner
     – Offene Spezifikation
   - Standard Java-API
     – Ein-/Ausgabe
     – Netzwerkfunktionalität
     – Grafik/GUI

Java-Eigenschaften

Java: A simple, object-oriented, network-savvy, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, dynamic language

SIMPLE

- C/C++ ähnlich
- Vereinfachung durch Weglassen
  - keine Zeiger
  - Strings/Arrays sind richtige Objekte
  - kein Preprozessor
  - Freigabe von Speicher nicht mehr nötig: Garbage Collector
  - kein Operator-Overloading
  - keine Mehrfachvererbung

Java-Eigenschaften, Teil 2

OBJECT-ORIENTED

- Klassen ➠ Objekte (Instanzen)
- Methoden, Variablen
- Einfachvererbung
- Keine Mehrfachvererbung sondern Interfaces: Sammlung von Methodendeklarationen und Konstanten

NETWORK-SAVVY (savvy (engl.) – kapieren, begreifen)

Standard Netzwerk API `java.net`

Java-Eigenschaften, Teil 3

INTERPRETED, PORTABLE, ARCHITECTURE NEUTRAL

- Ausführung von Bytecode in der Java Virtual Machine (JVM)
- Unicode für Texte (Programmtext und Daten)
- Festgelegte Formate für primitive Datentypen: `byte, short, int, long, float, double`
- Standard API
- Standard GUI
  Abstract Windowing Toolkit (AWT)

Java-Eigenschaften, Teil 4

DYNAMIC

Klassen werden zur Laufzeit eingebunden.
Beispiel: Applets

ROBUST, SECURE

Applet $\leftrightarrow$ Application : *Sandkasten*modell.

4 stufiges Sicherheitskonzepts

1. Sprachdesign
2. Bytecode Verifier
   - Stack Over-/Underflow
   - Korrektheit der Operanden
   - Keine illegalen Klassenzugriffe
3. Class Loader
4. Security Manager

Weitere Sicherheitskonzepte

- Offenes System
- Klassen zur Verschlüsselung etc.

Java-Eigenschaften, Teil 5

High-Performance, Multi-Threaded

- Threads sind Bestandteil der JVM
- Einfaches Umwandeln der JVM Bytecodes in native Befehle
- Implementation der JVM in Hardware (picoJava, microJava etc.)

Das Java-API (Basis-Klassen)

`java.lang`
Thread, Number, Character, String, System

`java.util`
Random, Dictionary, Vector

`java.applet`
Applet

`java.io`
Ein-/Ausgabestreams

`java.net`
Socket, URL

`java.awt`
Button, Checkbox, Scrollbar, etc.

`java.awt.image`
Font, Image, Point, Polygon

Das Java-API (Erweiterungen)

`java.io.Serializable`
Interface: Serializable

`java.rmi`
Remote Method Invocation (RMI)

`java.sql`
Java DataBase Connectivity (JDBC)

`java.security`
Klassen für Message Digests,
Digitale Signaturen

Java für Agenten ?

- Mobilität
  - `java.net`
  - `java.rmi`
  - Serialisierung
- Heterogenes Umfeld
  Plattformunabhängigkeit von Java
- Sicherheit
  Sicherheitskonzept von Java

Toolkits für Java-Agenten

- FTP Software Agent Technology
- MOLE - A Java Based Mobile Agent System
- Java-To-Go
- IBM Intelligent Agent
  - Agent Building Enviroment (ABE)
  - Aglets Workbench (AWB)

FTP Software Agent Technology

- Konzepte
- IP-Auditor
- IP-Distributor

MOLE - A Java Based Mobile Agent System

- Konzepte

- Das Mole System

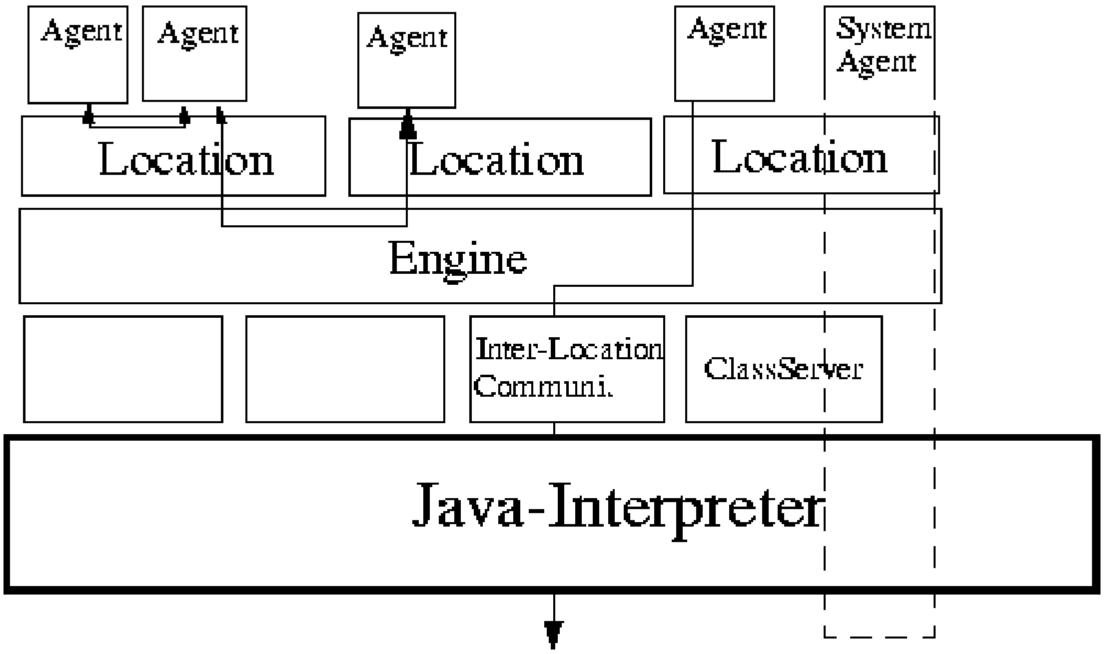
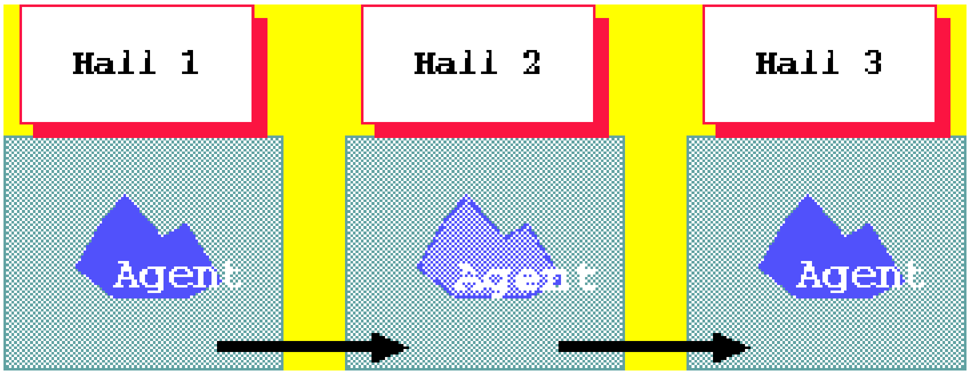


Figure 2: Architecture of the Mole system

Java-To-Go

- Infrastruktur



  - Agenten
  - Hall Server

IBM Intelligente Agenten

Beispiele von Agenten

- Webby
- World Avenue
- Intelligent Miner
- ...

IBM Intelligente Agenten

Agent Building Environment

- Architektur

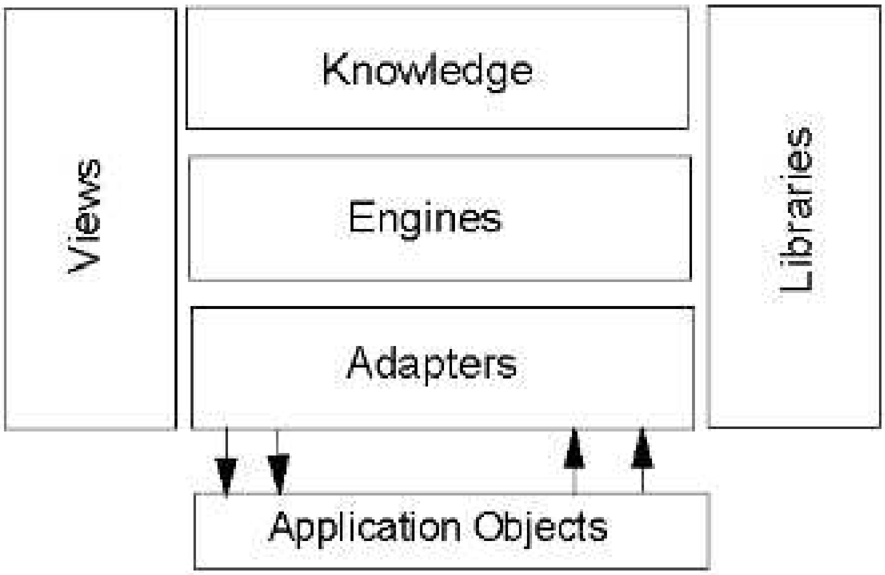


Figure 2: Agent Building Architecture

  - Engine
  - Knowledge
  - Adapters
  - Views

- Funktionen
  - Engine
  - Knowledge
  - KIF rule editor
  - Internet Adapters

Aglets Workbench

- J-AAPI
  - Wie werden Aglets programmiert ?
  - Beispiel

- Infrastruktur
  - Viewers
  - Servers
  - Resources
  - Boxes

Beispiel

Wie programmiert man ein Aglet ?

- Erweitere die `Aglet`-Klasse und überschreibe eine oder mehrere der folgenden Methoden:
  - `handleMessage(Message)`
  - `onCreation()`
  - `onClone()`
  - `onDispatching()`
  - `onArrival()`
  - `onReverting()`
  - `onDisposing()`
  - `onDeactivation()`
  - `onActivation()`
  - `run()`

Beispielcode

```
import aglet.*;

public class MyAglet extends Aglet {
  URL origin;
  public void onCreation(Object init) {
    origin = hostingURL();
  }
  public void onDispatching(URL destination) {
    System.out.println("Bye Bye...");
  }
  public void onArrival() {
    if (hostingURL().equals(origin))
       System.out.println("I‘m back...");
    else {
       doJob();
       dispatch(origin);
    }
  }
}
```