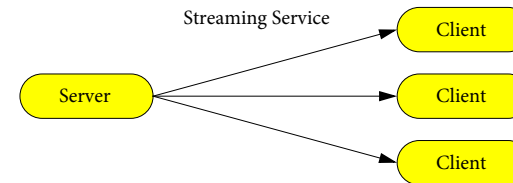


H FORMI: Fragmentierte RMI-basierte Objekte

- Motivation
- Fragmentierte Objektmodell
- Java RMI
- Fragmentierte Objekte und Java RMI
- Zusammenfassung

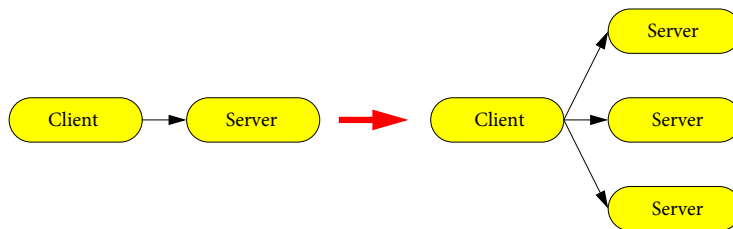
1 Motivation

- Standard Middleware für objektorientierte Anwendungen
 - ◆ CORBA, .NET-Remoting
 - ◆ Java Remote Method Invocation (RMI)
- **Problem:** Eine Vielzahl von Anwendungen mit nicht-funktionalen Anforderungen
 - ◆ Weiche Echtzeit
 - ◆ Fehlertoleranz
 - ◆ Hohe Verfügbarkeit



1 Motivation

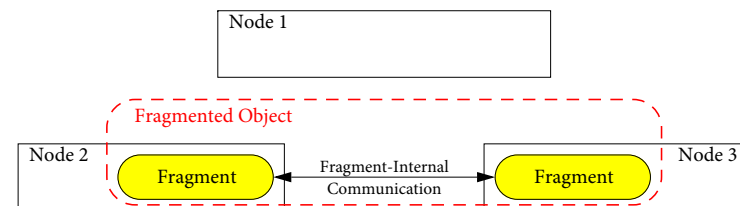
- Java RMI kann erweitert werden
 - ◆ Neue Aufrufsemantiken und Transportprotokolle
 - ◆ z.B. Realisierung von Objektgruppen für Fehlertoleranz



- Mögliche Anpassung sind jedoch eingeschränkt
 - ◆ Lösung: Fragmentiere Objektmodell [Shapiro94]

2 Fragmentierte Objektmodell

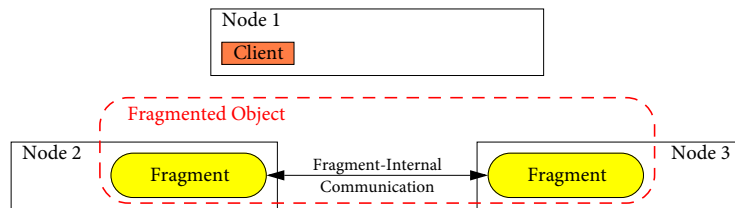
- Erweiterung des klassischen Stub-basierten verteilten Objektes
 - ◆ *Echt* verteiltes Objekt mit eindeutiger Id (Fragmente)
 - ◆ Verteilung von Zustand und Funktion sowie offene interne Kommunikation
 - ◆ Lokale Fragmente bieten die komplette Schnittstelle des FO an
 - ◆ Bietet Unterstützung für dynamische Adaption (z.B. Zustandsmigration etc.)
 - ◆ Implizite Bindung (Weitergabe einer Referenz erzeugt objektspezifisches Fragment)



2 Fragmentierte Objektmodell

H FORMI: Fragmentierte RMI-basierte Objekte

- Erweiterung des klassischen Stub-basierten verteilten Objektes
 - ◆ *Echt* verteiltes Objekt mit eindeutiger Id (Fragmente)
 - ◆ Verteilung von Zustand und Funktion sowie offene interne Kommunikation
 - ◆ Lokale Fragmente bieten die komplette Schnittstelle des FO an
 - ◆ Bietet Unterstützung für dynamische Adaption (z.B. Zustandsmigration etc.)
 - ◆ Implizite Bindung (Weitergabe einer Referenz erzeugt objektspezifisches Fragment)



Übungen zu Middleware

©Universität Erlangen-Nürnberg • Informatik 4, 2008

H-Formi-Fragment.fm 2008-12-11 11.27

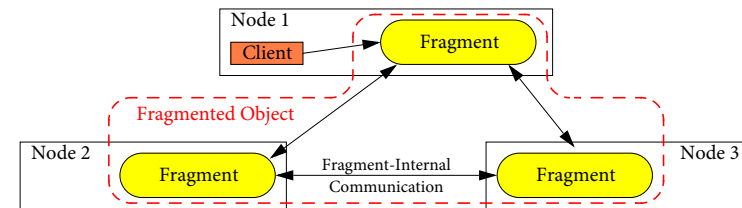
H.5

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Fragmentierte Objektmodell

H FORMI: Fragmentierte RMI-basierte Objekte

- Erweiterung des klassischen Stub-basierten verteilten Objektes
 - ◆ *Echt* verteiltes Objekt mit eindeutiger Id (Fragmente)
 - ◆ Verteilung von Zustand und Funktion sowie offene interne Kommunikation
 - ◆ Lokale Fragmente bieten die komplette Schnittstelle des FO an
 - ◆ Bietet Unterstützung für dynamische Adaption (z.B. Zustandsmigration etc.)
 - ◆ Implizite Bindung (Weitergabe einer Referenz erzeugt objektspezifisches Fragment)



Übungen zu Middleware

©Universität Erlangen-Nürnberg • Informatik 4, 2008

H-Formi-Fragment.fm 2008-12-11 11.27

H.6

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Idee

H FORMI: Fragmentierte RMI-basierte Objekte

- Fragmentiertes Objektmodell mit Java RMI
 - ◆ Design und Implementierung entstanden im Rahmen einer Masterarbeit
- Anforderungen
 - ◆ „friedliche“ Koexistenz von RMI- und fragmentierten FORMI-Objekten
 - ◆ Zugriffstransparenz für Aufrufer
 - kein Unterschied zwischen RMI- und FORMI-Objekt
 - ◆ Modelltransparenz bei Parameterübergabe
 - kein Unterschied bei Parameterübergabe
 - Übergabe von FORMI-Objekten an RMI-Objekte
 - Übergabe von RMI-Objekten an FORMI-Objekte
 - Übergabe von FORMI-Objekten an FORMI-ObjekteDesign
- Transparente Parameterübergabe
 - ◆ FORMI-Fragmente müssen sich wie RMI-Stubs verhalten
 - müssen mit RMI-Mechanismen serialisierbar sein

Übungen zu Middleware

©Universität Erlangen-Nürnberg • Informatik 4, 2008

H-Formi-Fragment.fm 2008-12-11 11.27

H.7

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Java RMI (Wiederholung)

H FORMI: Fragmentierte RMI-basierte Objekte

- Ziel von Java RMI
 - ◆ Erhaltung der Semantik Java Objektmodells in Kontext von verteilten Systemen
- Parameterübergabe Semantik
 - ◆ Primitive Datentypen, Java Objekte, nicht exportierte RMI-Objekte: call-by-value
 - ◆ Exportierte RMI-Objekte: call-by-reference
- Architektur



Übungen zu Middleware

©Universität Erlangen-Nürnberg • Informatik 4, 2008

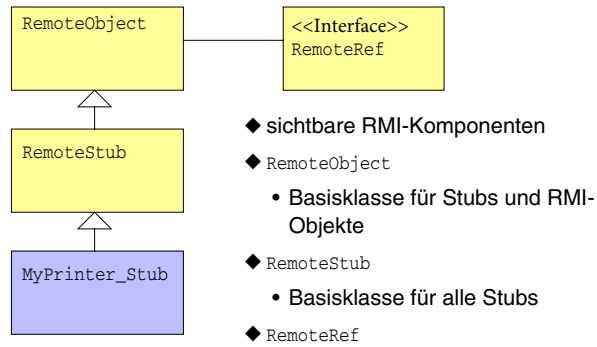
H-Formi-Fragment.fm 2008-12-11 11.27

H.8

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

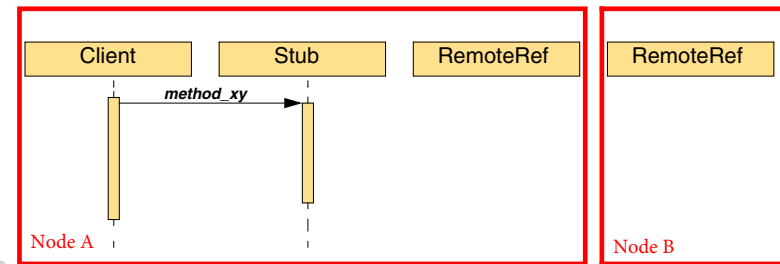
4 Java RMI

Zur Erinnerung: Aufbau der RMI-Stubs



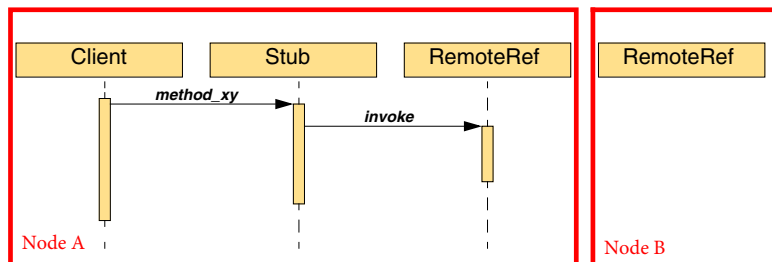
4 Java RMI

Entfernter Methodenaufruf in Java RMI



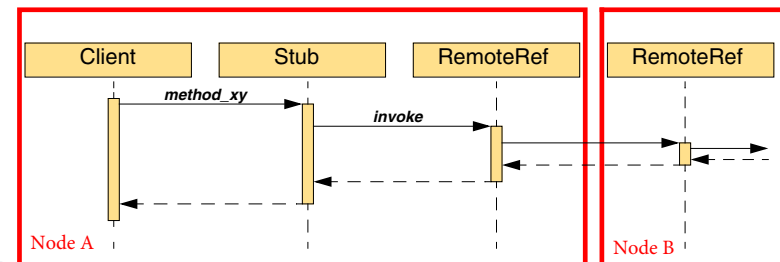
4 Java RMI

Entfernter Methodenaufruf in Java RMI



4 Java RMI

Entfernter Methodenaufruf in Java RMI



4 Design (2)

■ Zur Erinnerung: Abläufe in RMI

- ◆ Aufruf
 - Stubobjekt wandelt Aufruf in generischen Aufruf um
 - Stubobjekt ruft `invoke()`-Methode an der `RemoteRef` auf
- ◆ Marshalling
 - falls RMI-Objektreferenz vom Typ `RemoteStub`, dann serialisiere Referenz
 - falls RMI-Objektreferenz exportiertes Objekt, erzeuge Stub und verfare wie oben
- ◆ Unmarshalling
 - deserialisiere Stubobjekt

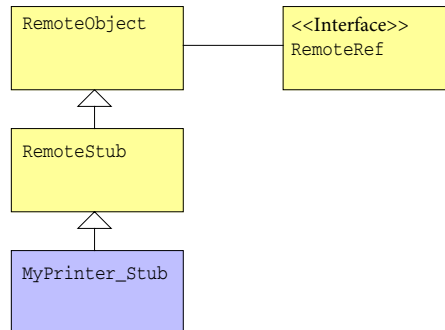
5 Aufbau eines Fragments

■ Anforderung an den Fragmentaufbau

- ◆ dynamischer Austausch der Implementierung
 - Indirektionsstufe (vgl. Aspectix in Kap. G)
 - Trennung von Fragmentimplementierung und Fragment-Interface
- ◆ dynamische Entscheidung über Fragmentimplementierung bei Parameterübergabe
 - Einsatz einer Fragment-Implementation-Factory

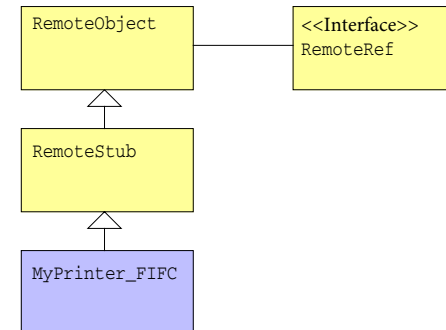
5 Aufbau eines Fragments (2)

■ Lösung in FORMI



5 Aufbau eines Fragments (2)

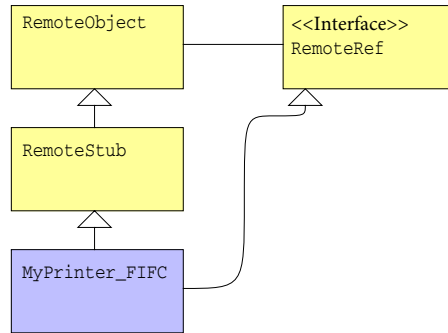
■ Lösung in FORMI



5 Aufbau eines Fragments (2)

H FORMI: Fragmentierte RMI-basierte Objekte

■ Lösung in FORMI

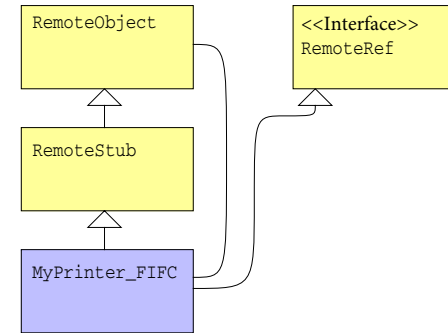


MW - Übung

5 Aufbau eines Fragments (2)

H FORMI: Fragmentierte RMI-basierte Objekte

■ Lösung in FORMI

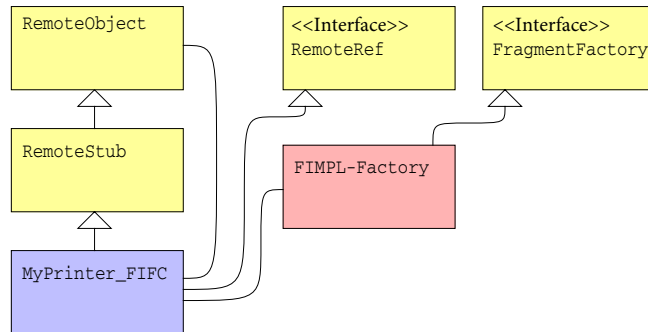


MW - Übung

5 Aufbau eines Fragments (2)

H FORMI: Fragmentierte RMI-basierte Objekte

■ Lösung in FORMI

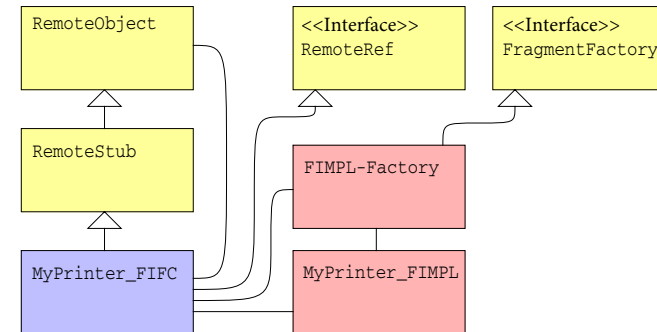


MW - Übung

5 Aufbau eines Fragments (2)

H FORMI: Fragmentierte RMI-basierte Objekte

■ Lösung in FORMI

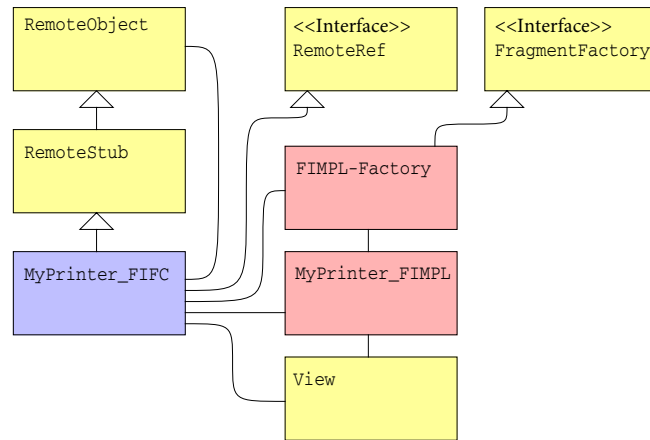


MW - Übung

5 Aufbau eines Fragments (2)

H FORMI: Fragmentierte RMI-basierte Objekte

■ Lösung in FORMI



Übungen zu Middleware

©Universität Erlangen-Nürnberg • Informatik 4, 2008

H-Formi-Fragment.fm 2008-12-11 11.27

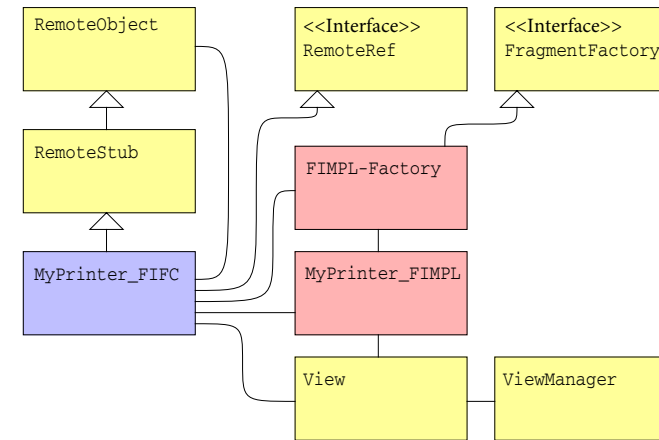
H.21

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Aufbau eines Fragments (2)

H FORMI: Fragmentierte RMI-basierte Objekte

■ Lösung in FORMI



Übungen zu Middleware

©Universität Erlangen-Nürnberg • Informatik 4, 2008

H-Formi-Fragment.fm 2008-12-11 11.27

H.22

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Aufbau eines Fragments (3)

H FORMI: Fragmentierte RMI-basierte Objekte

■ Lösung in FORMI (fortges.)

- ◆ Stub bildet Fragment-Interface
- ◆ Fragment-Interface implementiert RemoteRef und referenziert sich selbst
 - vermeidet Marshalling-Probleme
 - invoke()-Methode wird normalerweise nicht genutzt
 - invoke()-Aufrufe werden in lokale Aufrufe umgewandelt
- ◆ Fragment-Interface leitet Aufrufe an Fragmentimplementierung weiter
- ◆ Fragment-Interface referenziert eine Factory für Fragmentimplementierungen
 - Factory fällt Entscheidung welche Implementierung verwendet wird
 - dynamische Entscheidungen möglich (vgl. PDS in Kap. G)
- ◆ View-Objekt sorgt für Sharing der Fragmentimplementierung
 - View-Objekt wird in einem View-Manager verwaltet (enthält Tabelle aller lokalen Fragmente)

Übungen zu Middleware

©Universität Erlangen-Nürnberg • Informatik 4, 2008

H-Formi-Fragment.fm 2008-12-11 11.27

H.23

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

6 Aufrufe

H FORMI: Fragmentierte RMI-basierte Objekte

■ Weiterleitung aus Fragment-Interface in die Fragmentimplementierung

7 Marshalling und Unmarshalling

■ Marshalling

- ◆ Serialisierung des Fragment-Interfaces
 - View und Fragmentimplementierung sind als transient gekennzeichnet und werden nicht serialisiert

■ Unmarshalling

- ◆ Deserialisierung des Fragment-Interfaces
- ◆ Suche nach vorhandenem View
 - Verlinken des Views und entsprechender Fragmentimplementierung oder
 - Erzeugen einer neuen Fragmentimplementierung über Factory und Erzeugung eines neuen View-Objekts über View-Manager

Übungen zu Middleware

©Universität Erlangen-Nürnberg • Informatik 4, 2008

H-Formi-Fragment.fm 2008-12-11 11.27

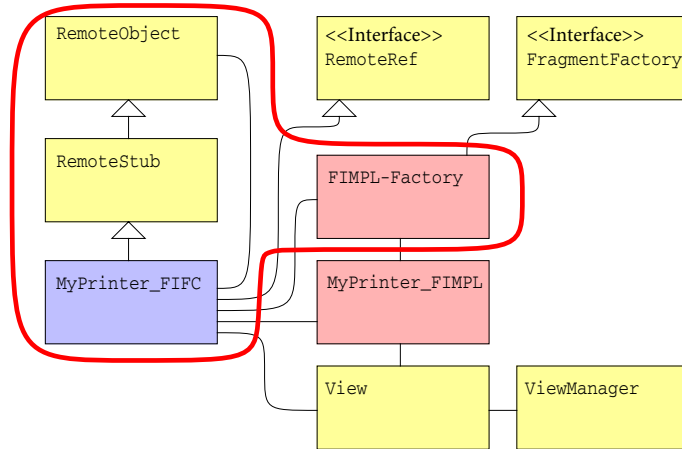
H.24

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

7 Marshalling und Unmarshalling (2)

H FORMI: Fragmentierte RMI-basierte Objekte

■ Serialisierungsbereich



8 Erzeugen von FORMI-Objekten

H FORMI: Fragmentierte RMI-basierte Objekte

■ Erzeugen eines ersten Fragments

- ◆ Aufruf einer speziellen Funktion mit Übergabe von
 - Fragment-Implementation-Factory
 - FORMI-Objektyp
 - erste Fragmentimplementierung
- ◆ erstes Fragment für Client als RMI-Stub sichtbar
 - bei Parameterübergabe: Marshalling wie oben beschrieben (kein Exportieren nötig)

9 Kommunikation zwischen Fragmenten

H FORMI: Fragmentierte RMI-basierte Objekte

- Beliebige Kommunikation
 - ◆ Java-Sockets stehen zur Verfügung
 - ◆ Einsatz von Standard-RMI für aufrufbasierte Kommunikation zwischen Fragmenten
- Finden der Fragmente
 - ◆ zur Zeit kein Ortsdienst verfügbar
 - kann selbst implementiert werden
 - Verankerung in der Fragment-Implementation-Factory
 - ◆ statische Kommunikationsadressen
 - Implantieren der Adressen in die Fragment-Implementation-Factory
 - ◆ Verwendung der Registry als Ersatz für Ortsdienst
 - z.B. zum Finden von Standard-RMI-Objekten, die anderes Fragment repräsentieren

10 Unterstützung bei der Objektentwicklung

H FORMI: Fragmentierte RMI-basierte Objekte

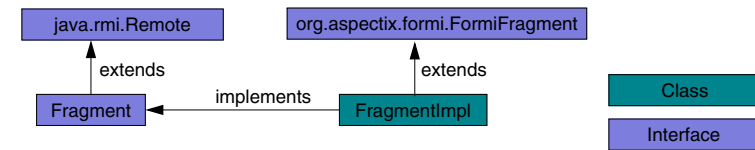
- Eigener RMI-Compiler
 - ◆ formic-Werkzeug
 - aus rmic-Implementierung von Sun abgeleitet
 - erzeugt Fragment-Interfaces (spezielle RemoteStubs für FORMI)
- Fragmentimplementierung
 - ◆ muss lediglich Remote-Interface implementierung und ...
 - ◆ ... von `FragmentImplementation` erben
 - ◆ keine Code-Erzeugung notwendig

11 Zusammenfassung

- FORMI hat den fragmentierten Ansatz
 - ◆ weltweit eindeutige **Objektbezeichner** zum Parametertransport zwischen Fragmenten
 - serialisiertes Fragment-Interface
 - RMI-kompatibel
 - **dynamisches Laden** von Code durch RMI-Classloader und Codebase-Angabe
 - Erzeugung lokaler Fragmente mit Fragment-Interface, Fragment-Implementierung, View-Objekt
 - ◆ **Erzeugung** neuer verteilter Objekte
 - ◆ spezielle Funktion zur Erzeugung der ersten Fragmentimplementierung

H.1 Aufgabe #5

■ Fragmented IM



1 Aufgabe #5

- Fragment Implementierung
 - ◆ `public interface Fragment extends Remote {}`
 - ◆ `public class FragImpl extends FormiFragment implements Fragment {}`
- Fragment Initialisierung und Erzeugung remote Referen
 - ◆ Erzeugen einer remote Referenz


```
Fragment frag = (Fragment) FragmentedObjectFactory.createObject(
    (FragImpl.class, DefaultFragImplFactory.class, (Object[]) new
    SIpAddress[] {comm}, null, null);
```
 - ◆ Veröffentlichen der Referenz an einem Namensdienst (z.B. RMIRegistry)


```
Naming.rebind("rmi://localhost/frag", frag);
```
- Referenz auf fragmentiertes Objekt
 - ◆ `MessengerInterface frag = (MessengerInterface) Naming.lookup("rmi://localhost/frag");`