Zwischenbericht
Großer Beleg

Aspektorientierte Programmierung und EJB im Vergleich

Aufgabenstellung

EJB und AOP im Vergleich:

- Trennung technischer Funktionalität und Geschäftslogik bei Entwicklung von verteilten Anwendungen
- Untersuchung, ob AOP eine gleichwertige bzw. sogar effektivere Lösung

Gliederung

- AOP und AspectJ
- EJB
  - Wichtige Funktionseinheiten
- Beispielanwendung
- Realisierung mit EJB
- Realisierung in AOP ohne EJB
- Vergleich der Lösungen

Gliederung

- **AOP und AspectJ**
- EJB
  - Wichtige Funktionseinheiten
- Beispielanwendung
- Realisierung mit EJB
- Realisierung in AOP ohne EJB
- Vergleich der Lösungen

Gründe für den Einsatz von Aspektorientierter Programmierung

- Verschmutzung des funktionalen Codes
- Beeinflussung mehrer Klassen im Programm - **crosscuts**

→ Kapselung dieser zusätzlichen, technischen Anforderungen in Aspekten

- Erhöhte Modularisierung
- Verringerung des Codes der Geschäftslogik
- Wartbarkeit und Wiederverwendbarkeit

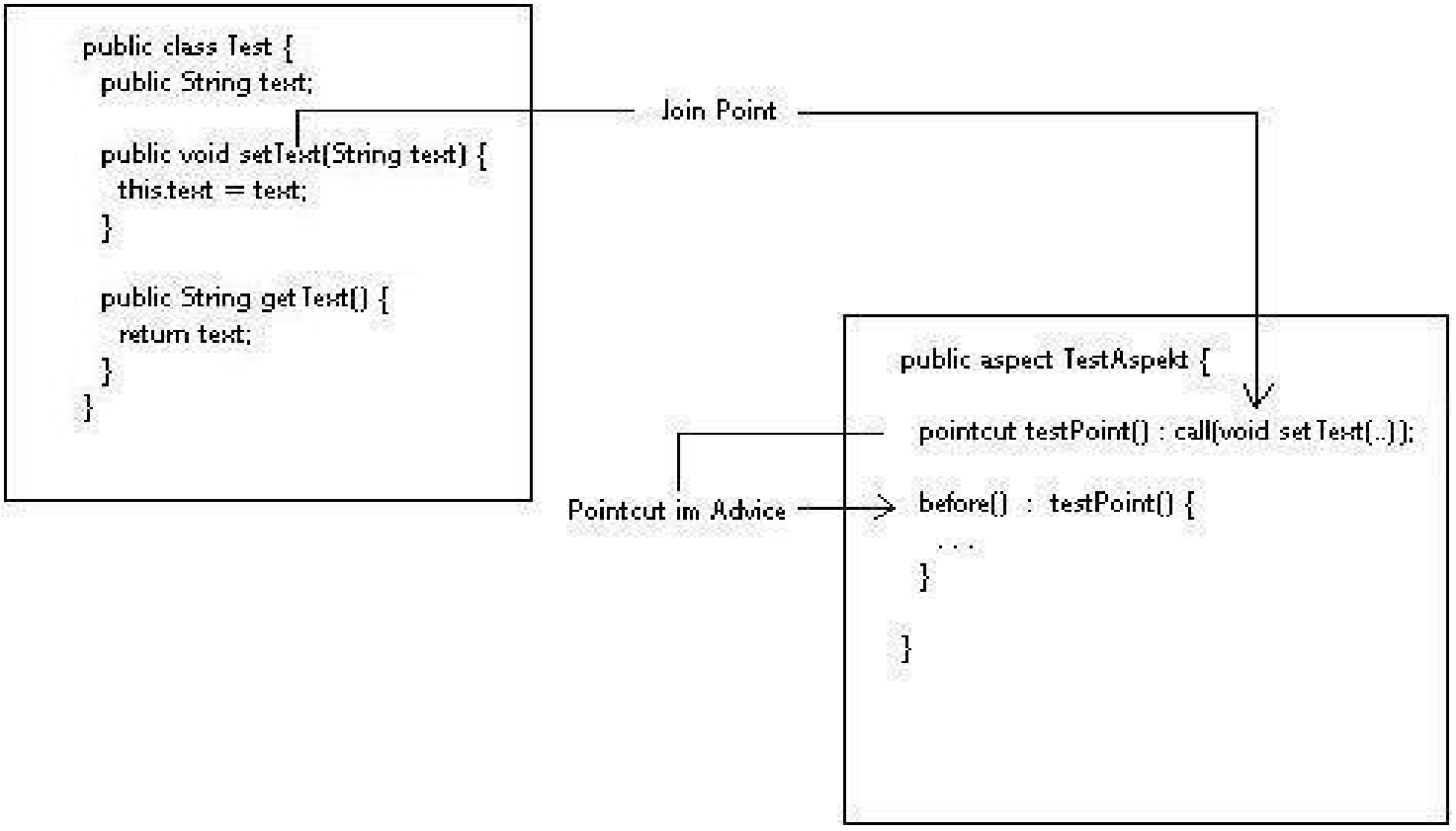
Typische Anwendungsgebiete

- Tracing / Logging
- Fehlerbehandlung und Fehlertoleranz
- Synchronisation / Thread-Sicherheit
- Caching-Strategien
- Resource-Sharing
- Transaktionen
- Test
- ...

Funktionsweise der AOP

- Trennung in Aspekte und Klassen
- Klassen: Geschäftslogik
- Aspekte: zusätzliche Information (crosscutts)
- AspectJ Compiler
  1. Zusammenfügen von Klassen und Aspekten durch Aspect Weaver
  2. normaler Java Compiler
- Ergebnis
  - Erweiterte Funktionalität, wie von Hand eingefügt
  - Klassen sieht man nichts an

Zusammenspiel



Pointcut (I)

Überwachung des Zugriffs auf Join Points unter bestimmten Bedingungen:

- **`call`, `execution`, `initialization`**
  - `call(public String Test.getString());`
  - `initialization(Test.new());`
- **`get`, `set`** – Zugriff auf statische Attribute
  - `set(String Test.staticVar);`

  Zugriff auf neuen Wert mit **`args`**
- **`handler`** – Ausführung einer Exception
  - `handler(java.lang.Exception)`

Pointcut (II)

- Komposition: `||` `&&` `!`
- **`this,target,args`** – Zugriff auf Objekte
  - `this()`: das aufrufende Objekt
  - `target()`: das Zielobjekt
  - `args()`: Parameter beim Methodeaufruf
- Namensbasiertes Crosscutting
  - `execution(public void Test.setString(String));`
- Eigenschaftsbasiertes Crosscutting
  - `execution(public * *.setString(..))`

Advice

Auszuführen, wenn Pointcut erfüllt:

before – Nach Betreten des Methodenkörpers

```
before() : Pointcut
```

after – Vor Verlassen des Methodenkörpers

after returning – Bei erfolgreichem Beenden

after throwing – Beim Beenden mit Exception

around – Anstatt des Methodenkörpers

– Eigentlichen Inhalt mit **proceed**

thisJoinPoint – Zugriff auf Signatur, Argumente

Statisches Crosscutting (I)

- Definieren eines Feldes
  - `private int Test.id = 0;`
- Definieren einer Methode
  - `public int Test.getId() {...}`
- Definieren einer abstrakten Methode
  - `abstract public int Test.setId (int id);`
- Definieren eines Konstruktors

```
public Test.new(String text,int id) {
  this.text = text;
  this.id = id;
}
```

Statisches Crosscutting (II)

`declare parent :`

- Deklarieren von Vererbungsstruktur

  `Test extends Superclass;`

- Deklarieren von Interface Implementation

  `Test implements Interface;`

Gliederung

- AOP und AspectJ
- **EJB**
  - **Wichtige Funktionseinheiten**
- Beispielanwendung
- Realisierung mit EJB
- Realisierung in AOP ohne EJB
- Vergleich der Lösungen

EJB–Funktionseinheiten

Einschränkungen:

- Keine Message Driven Beans
- Nur Containermanagement

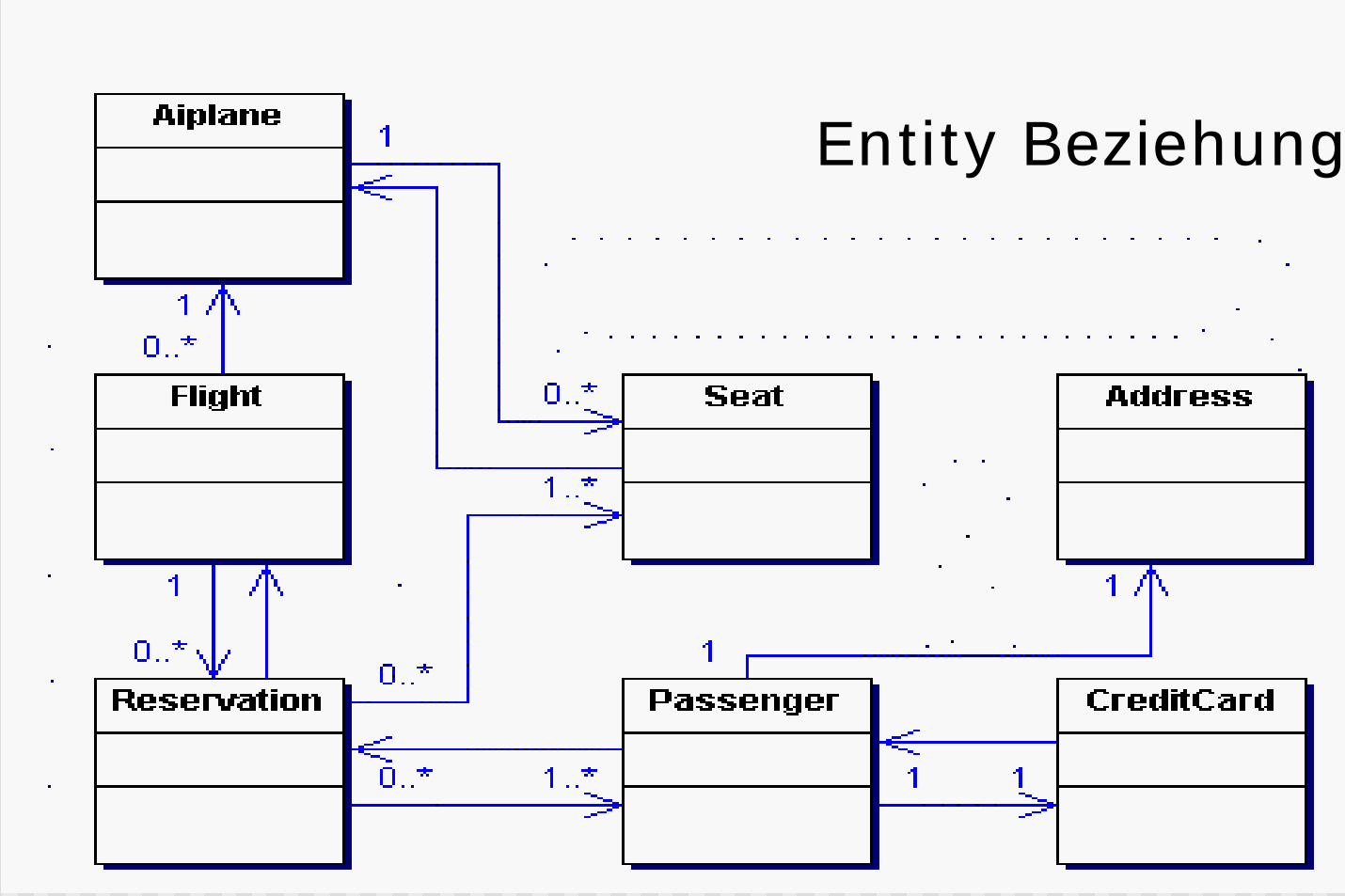
Funktionenseinheiten:

- Zugriff über Netzwerk
- Aufbau Schnittstelle
- Eintity und Session Beans
- Persistenzhaltung
- Transaktionsverwaltung
- Sicherheit und Zugangskontrolle
- ResourceManagement

Gliederung

- AOP und AspectJ
- EJB
  - Wichtige Funktionseinheiten
- **Beispielanwendung**
- Realisierung mit EJB
- Realisierung in AOP ohne EJB
- Vergleich der Lösungen

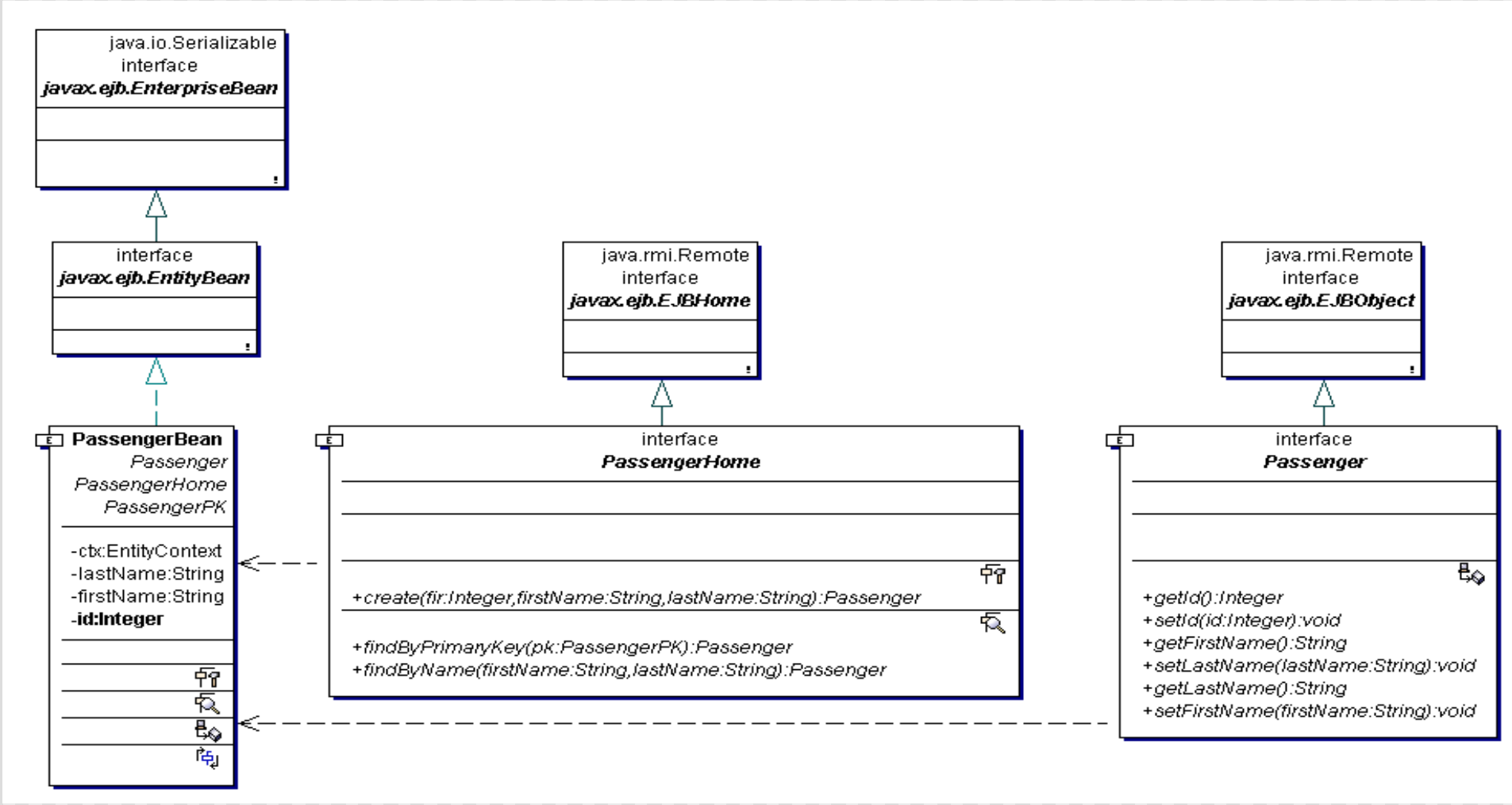
Beispiel: Flugbuchungs- & Verwaltungssystem



Gliederung

- AOP und AspectJ
- EJB
  - Wichtige Funktionseinheiten
- Beispielanwendung
- ➢ **Realisierung mit EJB**
- Realisierung in AOP ohne EJB
- Vergleich der Lösungen

Klassenstruktur in EJB



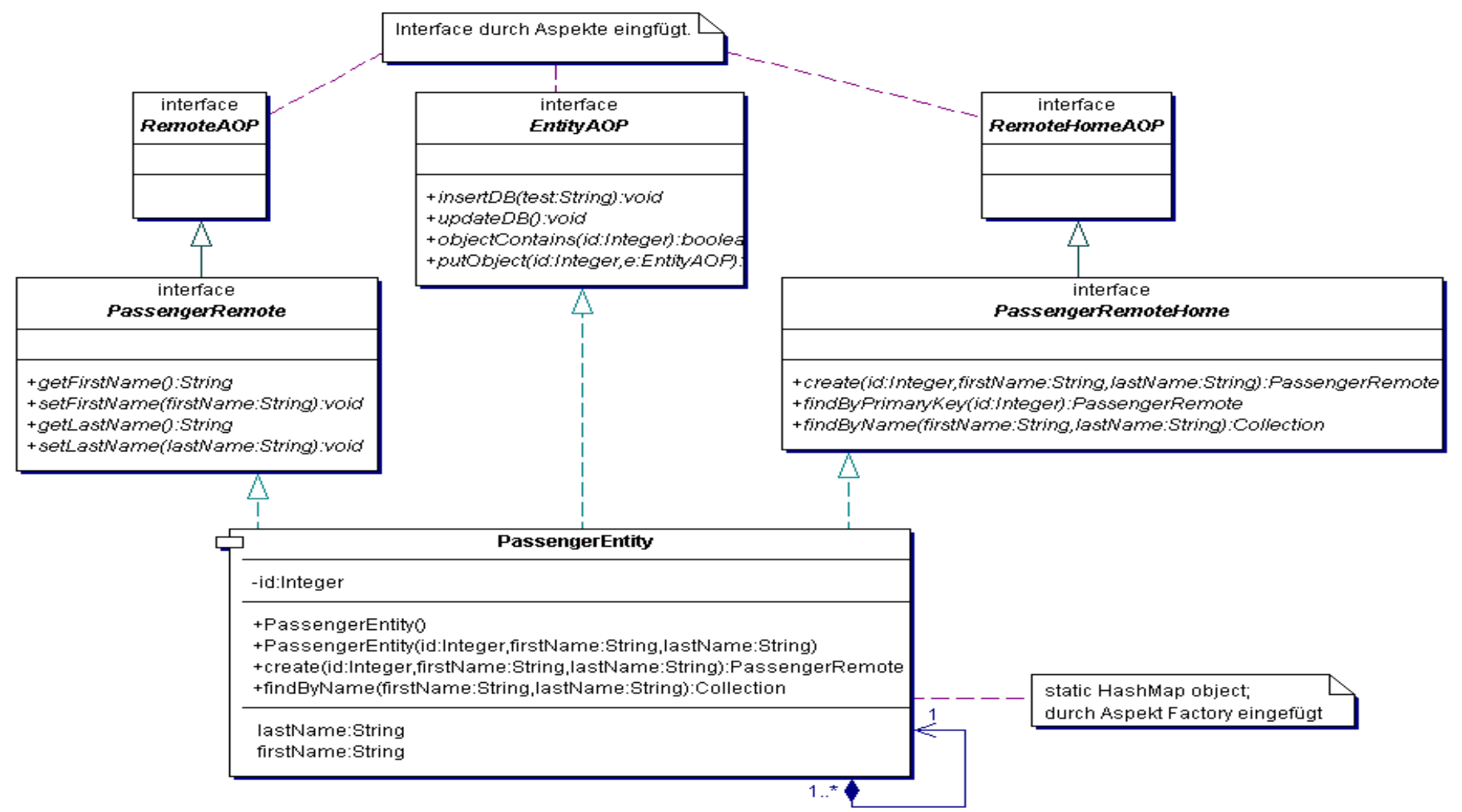
Gliederung

- AOP und AspectJ
- EJB
  - Wichtige Funktionseinheiten
- Beispielanwendung
- Realisierung mit EJB
- **Realisierung in AOP ohne EJB**
- Vergleich der Lösungen

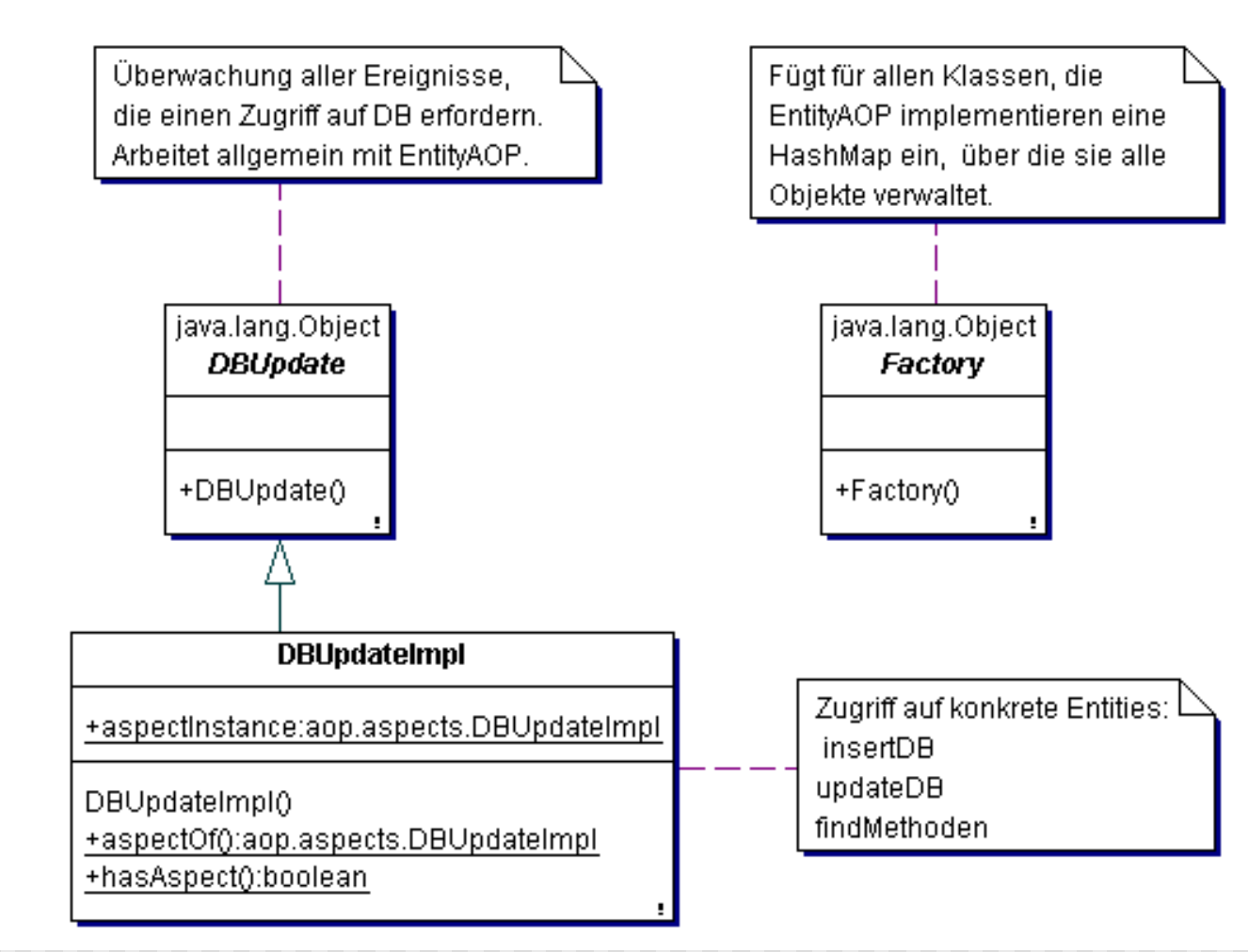
Betrachteter Ausschnitt

- Passenger EntityBean
  - Schnittstellengestaltung (RemoteHome / Remote)
  - Zugriff über Netzwerk
  - Persistenzhaltung der Daten
    - Zugriff auf Datenbank
  - Verwaltung der Objekte

Klassen und Schnittstellen in AOP



Aspekte in AOP



Weitere Planungen

- Momentan in Bearbeitung
  - Allgemeine Funktion für DB Zugriff
- Eventuell Verwendung von XML
  - Definition der Persistenzattribute (Primary Key)
  - Definition von Beziehungen
- Umsetzung der anderen Funktionseinheiten in EJB

Gliederung

- AOP und AspectJ
- EJB
  - Wichtige Funktionseinheiten
- Beispielanwendung
- Realisierung mit EJB
- Realisierung in AOP ohne EJB
- **Vergleich der Lösungen**

Vergleichs- und Bewertungsansätze

- Bewertung der Realisierbarkeit
- Aufwand für Anpassung der Anwendung
- Vergleich der Container Anhand von bestimmten Kriterien
  - Modularisierung
    - Bessere Modularisierung
    - Wiederverwendung der Module
    - Kombination von Modulen
    - Anpassung des Containers an speziellen Funktionsanforderungen