

Java Data Objects im Kontext von J2EE

Dipl.-Wirtsch.-Inf. M. Merz
E-Mail: merz@rz.uni-mannheim.de

Agenda

1. Grundlagen
2. Java 2 Enterprise Edition (J2EE)
3. Java Data Objects (JDO)
4. Integration von JDO in J2EE
5. Performance-Messung
6. Fazit

1. Grundlagen Objektorientierter Programmierung

Grundlagen

■ Objekt

- Grundbaustein objektorientierter Anwendungssysteme
- Eigenständige Einheit, die über Daten (Attribute) und Verhalten (Methoden) verfügt
- Ist eindeutig identifizierbar

Beispiel:

- Objekt "Student"
- Attributen: Name, Vorname, Matrikelnummer
- Methode: Exmatrikulieren

Grundlagen

- Im Allgemeinen werden Objekte zur Laufzeit eines Anwendungsprogramms erzeugt und spätestens mit dessen Beendigung wieder gelöscht
- Was passiert mit Objekten die über die Lebenszeit dieses Programms dauerhaft gehalten werden sollen?

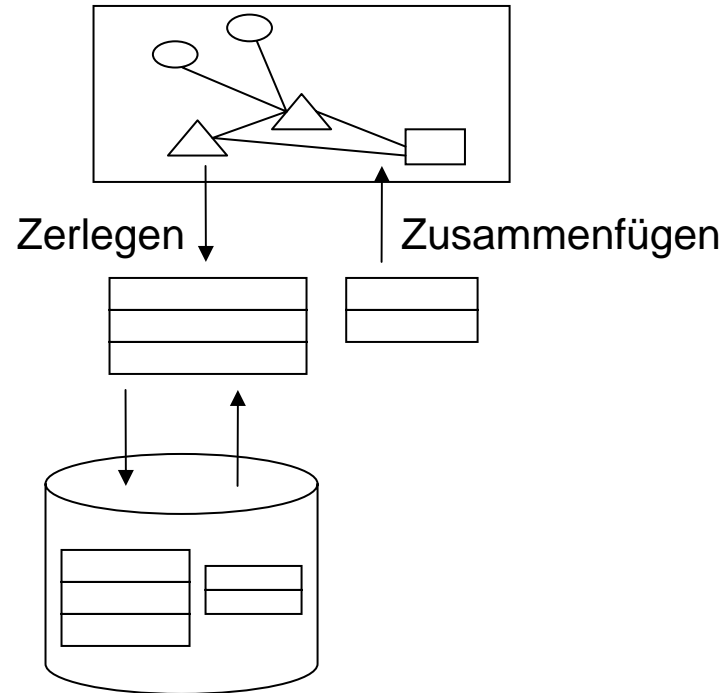
=> Persistentes Objekte können beispielsweise in Datenbanken abgelegt werden

Objekte und relationale Datenbanken

Objekte im
Anwendungsprogramm

Anwendung:
Relationale
Darstellung

RDBMS



Ineffizientes Mapping der Objekte auf Tupeln in der Datenbank

--> Impedance Mismatch

2. Java 2 Enterprise Edition (J2EE)

Java 2 Enterprise Edition (J2EE)

- J2EE ist eine Architektur zur Entwicklung Serverseitiger, mehrschichtiger Anwendungen in Java.
- Kommerzielle J2EE Server (= Application Server) sind derzeit von verschiedenen Herstellern verfügbar (Bea Weblogic, IBM etc.)
- Es gibt auch Open-Source Application Server z.B. JBoss (www.jboss.org)

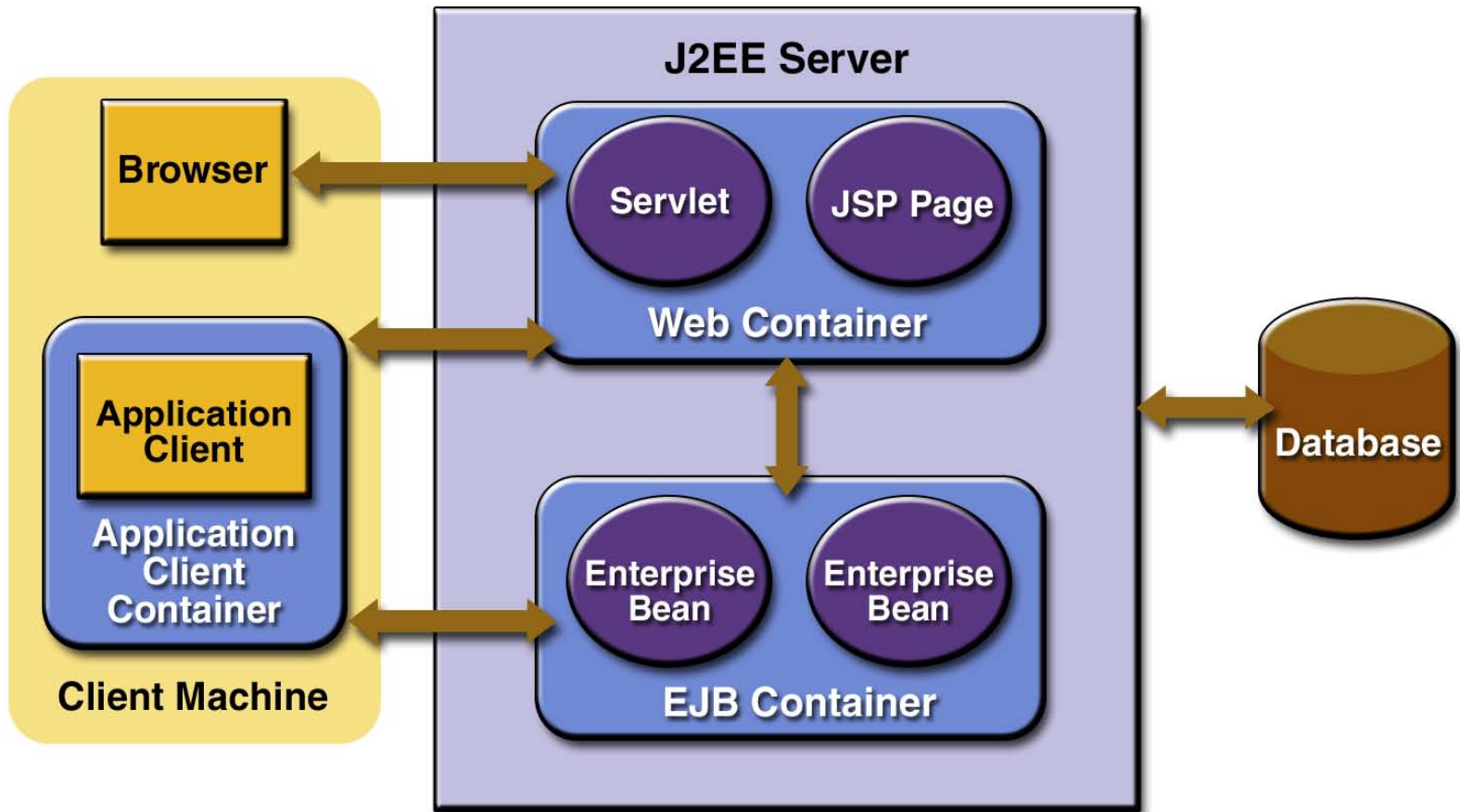
Java 2 Enterprise Edition (J2EE)

Was Vorteile bringt die J2EE-Plattform?

Es lassen sich verteilte, skalierbare, zuverlässige und sichere Anwendungen erstellen, bei denen die sich der Programmierer nicht um technische Details kümmern muss, wie z.B.:

- Sicherheitsmanagement (Rechteverwaltung)
- Transaktionsverwaltung
- Datenbankzugriff (Persistenz)
- Nebenläufigkeit
- Verwaltung von Ressourcen

Java 2 Enterprise Edition (J2EE)



Java 2 Enterprise Edition (J2EE)

Wesentliche Typen von Enterprise Java Beans:

- Session Beans
 - modellieren Prozesse und Workflows
 - sind einem Client zugeordnet
 - stehen in Beziehungen zu Entity Beans
- Entity Beans
 - modellieren Realweltobjekte
 - Werden persistent in einer Datenbank gespeichert

Java 2 Enterprise Edition (J2EE)

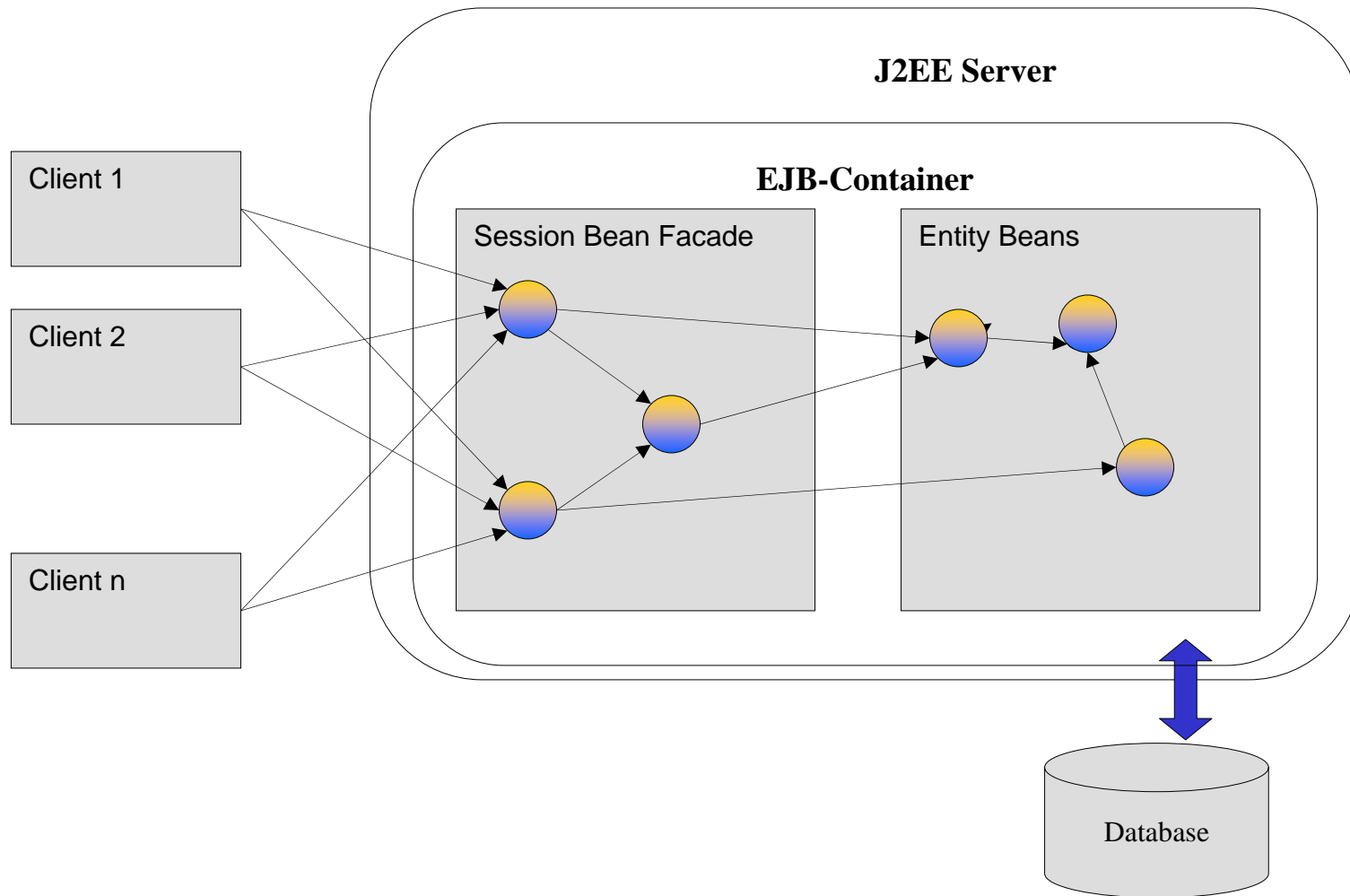
Woraus setzen sich Java Beans zusammen?

In einer Datei zusammengefasst werden

- Eine Java-Klassen
- Mehrere Interfaces
- Mindestens eine Konfigurationsdatei (Deployment Descriptor)

Der Vorgang der Installation einer Java Bean in einen J2EE-Server wird als Deployment bezeichnet.

Java 2 Enterprise Edition (J2EE)



Java 2 Enterprise Edition (J2EE)

- Persistenz mit EJB
 - EntityBeans müssen über spezielle Methoden verfügen:
z.B.: `ejbLoad()`, `ejbStore()`
 - Methoden werden durch den Container aufgerufen
- Zwei Alternativen für Mapping der Daten
 - Bean Managed Persistence (BMP)
 - Container Managed Persistence (CMP)

Java 2 Enterprise Edition (J2EE)

■ BMP

- Entwickler schreibt die Klassen `ejbLoad()` und `ejbStore()` selbst
- z.B. über SQL-Zugriff (JDBC) auf relationale Datenbanken

■ CMP

- Container generiert beim Deployment die Klassen `ejbLoad()` und `ejbStore()`
- Datenbank wird im Deployment Descriptor festgelegt

Java 2 Enterprise Edition (J2EE)

- Vorteile
 - J2EE mit CMP erspart dem Entwickler viel Arbeit

- Nachteile
 - "Overkill" für kleiner Anwendungen durch Aufwendige Konfiguration und Deployment
 - Kein Testen außerhalb des J2EE-Servers möglich
 - Keine Unterstützung von OODB
 - Keine Unterstützung von Vererbungen

3. Java Data Objects (JDO)

Java Data Objects (JDO)

- Stellt eine einheitliche Programmierschnittstelle zur dauerhaften Speicherung von Java-Objekten zur Verfügung
- Beliebige Datenbanken können verwendet werden (RDBMS, ODBMS, Dateisystem)
- Aktueller Stand: Version 1.0 (25. März 2002)
- Integration in die J2EE-Plattformen möglich
- „Transparente Persistenz“ (Entwickler müssen sich nicht um Persistenz kümmern)

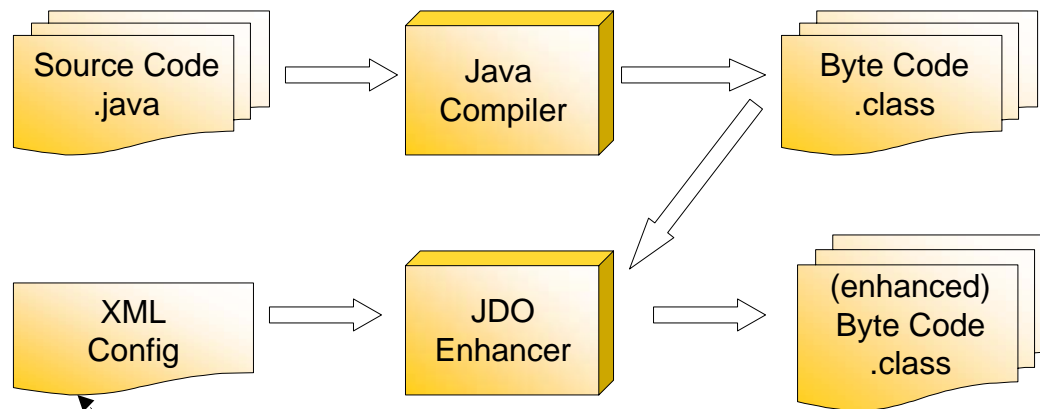
Java Data Objects (JDO)

Persistenz-Mechanismus:

- Alle Instanzen, die durch JDO verwaltet werden sollen, müssen das `PersistenceCapable` Interface implementieren
- Auf welche Weise die Implementation dieses Interface erfolgt, lässt die JDO-Spezifikation offen
- Theoretisch lässt sich die Implementierung manuell kodieren, üblicherweise wird ein Postprozessor (Enhancer) verwendet

Java Data Objects (JDO)

Enhancement-Prozess



- Persistence Descriptor:
 - Gibt an, welche Klassen persistenzfähig sein sollen
 - Es können aber auch detaillierte Angaben zu jedem einzelnen Feld einer Klasse gemacht werden

4. Integration von JDO in J2EE

Integration von JDO in J2EE

- JDO kann in J2EE-Umgebung eingebunden werden

=> Im Folgenden wird nur der Ansatz betrachtet, wonach Entity Beans komplett durch JDO-Objekte ersetzt werden

Integration von JDO in J2EE

■ Vorteil

- Datenbank unabhängig
- Testen auch außerhalb von Application Server
- Vererbung möglich

■ Nachteil

- Byte-Code Enhancement
- Keine Unterstützung von bidirektionalen Objekt-Beziehungen
- Neue Query-Language (JDO-QL)

=> Was ist mit der Performance?

5. Performance-Messung

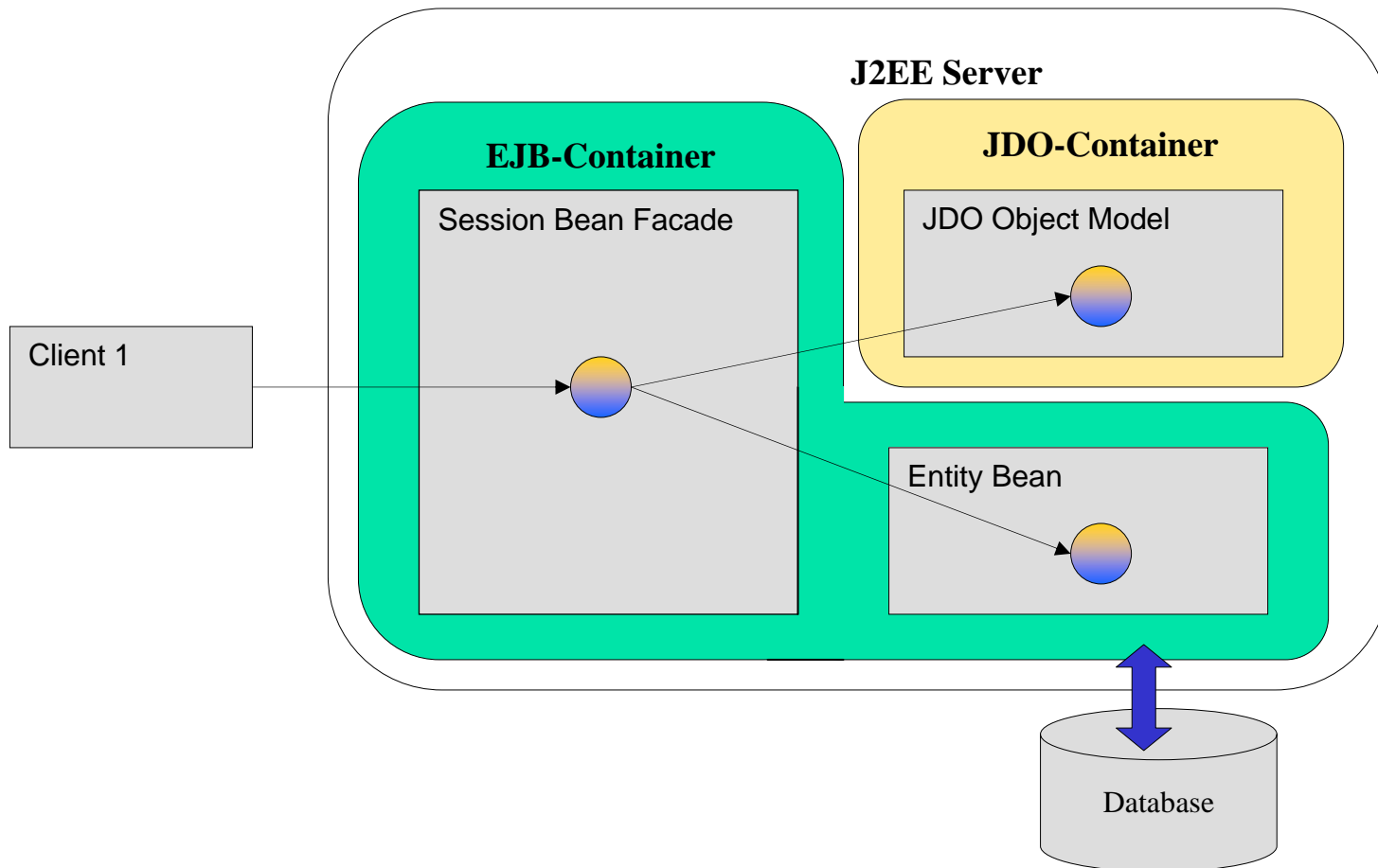
Performance-Messung

- Hard- und Software Ausstattung
 - Pentium IV 1,6 GHz; 512 MB Ram, Win2K
 - MySQL-Datenbank (3.23.53) + InnoDB Erweiterung
 - J2EE-Server: JBoss 3.2.0
 - intelliBO 3.1

Performance-Messung

- Vergleich von drei Alternativen:
 - J2EE mit BMP → Selbstgeschriebene SQL Anweisungen
 - J2EE mit CMP → Container erzeugt Anweisung
 - J2EE mit JDO → JDO sorgt für Persistenz
- Messungen in 2 Schritten:
 - Basis-Messungen (CRUD)
 - Komplexe Beispiel-Applikation

Performance-Messung

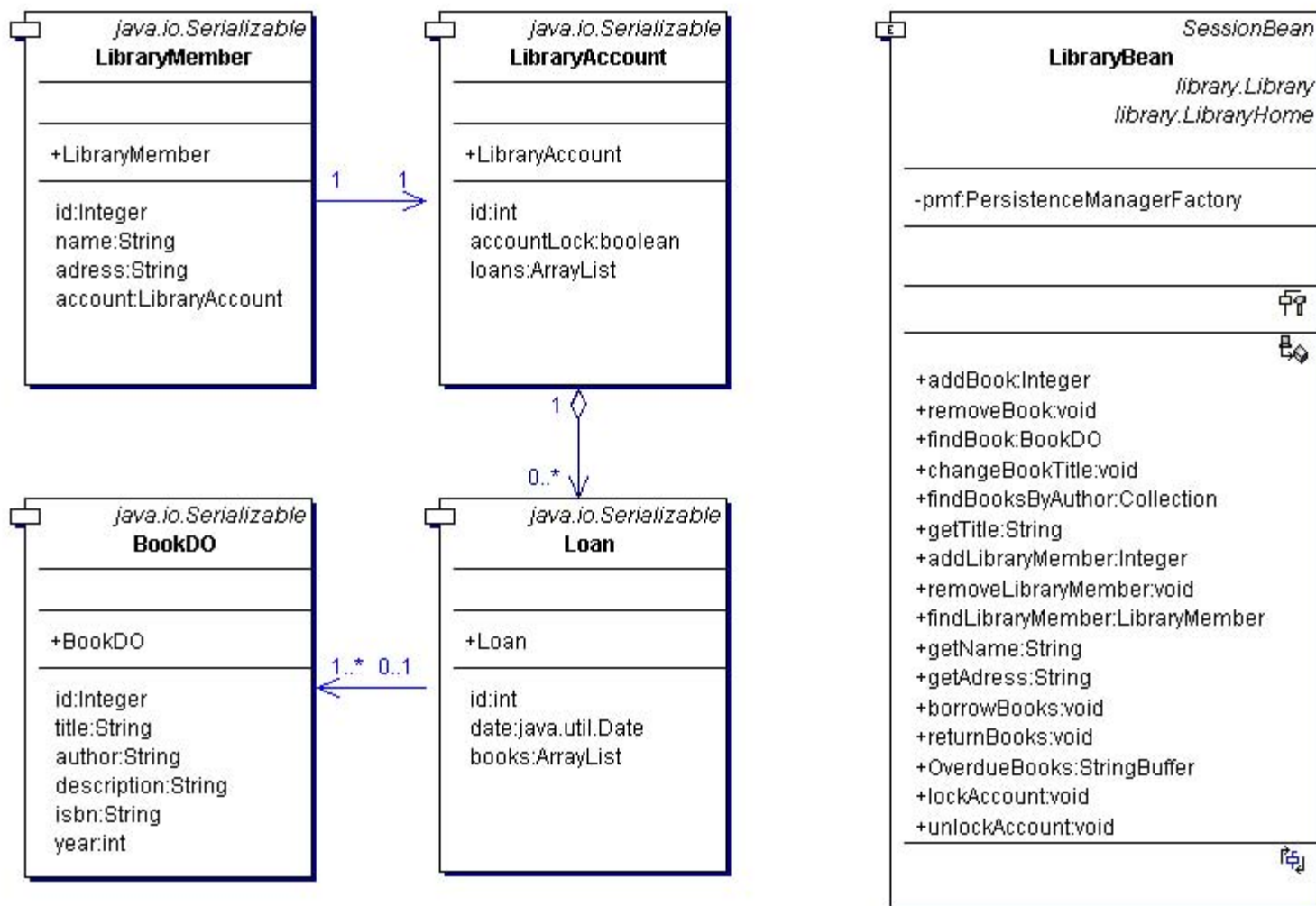


Performance-Messung

CRUD-Analyse mit 5.000 Objekten

Implementation	Create	Read	Update	Delete
JDO: average [sec]	24,5	22,5	25,9	25,3
variance[sec ²]	13,4	8,4	4,7	14,5
CMP: average [sec]	29,9	30,8	34,0	26,9
variance[sec ²]	3,9	0,1	0,1	0,0
BMP: average [sec]	29,7	27,6	23,4	23,3
variance[sec ²]	3,9	33,7	0,0	0,0

Performance-Messung



Performance-Messung

Ergebnisse

- JDO:
 - Ausführungszeit: 61,3 sec. (Varianz 11,8 sec²)
- CMP:
 - Ausführungszeit: 17,4 sec. (Varianz 0,2 sec²)
- ABER: Bei der wiederholten Ausführung:
 - JDO: Ausführungszeit nur noch 6 sec.
 - CMP: Keine Veränderung



6. Fazit

Fazit

- Kann auf bidirektionale Objektbeziehungen verzichtet werden und
- erfolgt der wiederholte Zugriff auf gleiche Objekte (Ausnutzung von Caching)
- Lohnt sich der Einsatz von JDO
- Zudem lassen sich die meisten JDO-Implementationen noch weiter tunen:
 - Query-Performance Optimizing
 - Stored-Procedures
 - Etc.

ENDE

Gibt es noch Fragen?