Objektorientierte Datenbanken

Vorlesung 7
Sebastian Iwanowski
FH Wedel

JDO: 5. Teil: **JDOQL (Abschluss)** *(siehe OODB 6)*
Datenidentitätskonzepte
Lebenszykluszustände (Anfang)

JDO-Datenidentitätskonzepte

Wozu benötigt man die genaue Definition der Identität ?

- für die Suche nach einem bestimmten Objekt in der Datenbank
- für Verweise auf andere Objekte in der Datenbank
- für die Arbeit mit persistenten Objekten in der Java Virtual Machine
- für den Abgleich zwischen Objekten in der JVM und der Datenbank

JDO-Datenidentitätskonzepte

Wie bestimmt man im Allgemeinen die Identität eines Objekts ?

- **durch Gleichheit der Adresse**

- **durch Beantwortung bestimmter Äquivalenzmethoden mit `true`**

- **durch Verwendung eines eindeutigen Namens**

JDO-Datenidentitätskonzepte

JDO-Konzept:

- **Jedes First-Class-persistente Objekt hat eine JDO-spezifische ID**
- **Jede JDO-spezifische ID gehört zu einer ID-Klasse, die folgende Eigenschaften erfüllt:**
  - **Die ID-Klasse ist `public`**
  - **Die ID-Klasse implementiert das Interface `Serializable`**
  - **Die ID-Klasse hat einen Konstruktor ohne Parameter**
  - **Die ID-Klasse überschreibt `toString()` für Vergabe eines eindeutigen Namens.**
  - **Die ID-Klasse hat einen Konstruktor, der einen `String` als Parameter hat.**
- **Jede persistenzfähige Klasse hat eine eindeutige ID-Klasse**
- **Die ID-Klasse einer persistenzfähigen Klasse wird in den XML-Metadaten festgelegt:**

1) entweder automatisch 2) oder individuell gewählt

JDO-Datenidentitätskonzepte

1) Automatische Festlegung der ID-Klasse: Datastore-Identity

Konzept: Identität wird durch eindeutigen Namen festgelegt

- Der Identitätsname kann aber nicht selbst gewählt werden, sondern wird durch die Persistenzverwaltung automatisch festgelegt.
- Ebenso wird die Klasse und Instanz der ObjektId durch die Persistenzverwaltung automatisch festgelegt.
- Zugriff über Objektname (als String) oder ObjektId (als Object)

Vorteile ?

JDO-Datenidentitätskonzepte

2) Individuelle Festlegung der ID-Klasse: Application-Identity

Konzept: Identität wird durch Primärschlüsselfelder festgelegt

- **Klasse und Instanz der ObjektId wird durch die Applikation festgelegt.**
- **Ein eindeutiger Objektname, der sich aus den Primärschlüsseln zusammensetzen muss, kann ebenfalls durch die Applikation festgelegt werden.**
- **Zugriff über Objektname (als String) oder ObjektId (als Object)**

Jede Application-ID-Klasse muss folgende zusätzliche Eigenschaften erfüllen:

- **Die ID-Klasse hat für jedes Primärschlüsselfeld ein `public` Feld gleichen Namens**
- **Die Methoden `equals(obj)` und `hashCode()` müssen von jedem einzelnen Primärschlüsselfeld abhängen**
- **Die Datentypen der Primärschlüsselfelder müssen zu denen gehören, die per default persistenzfähig sind (**außer **`HashSet`**, siehe OODB 3, Folie 18**)**

Vorteile ?

JDO-Datenidentitätskonzepte

3) Spezialfall: nondurable Identity

- **Für Datenbankobjekte, die in der Datenbank keine ID erhalten sollen**
- **Beispiele: log files, history files, etc.**

Vorteile ?

JDO-Datenidentitätskonzepte

Zusammenfassung:

JDO kennt folgende Datenidentitätskonzepte:

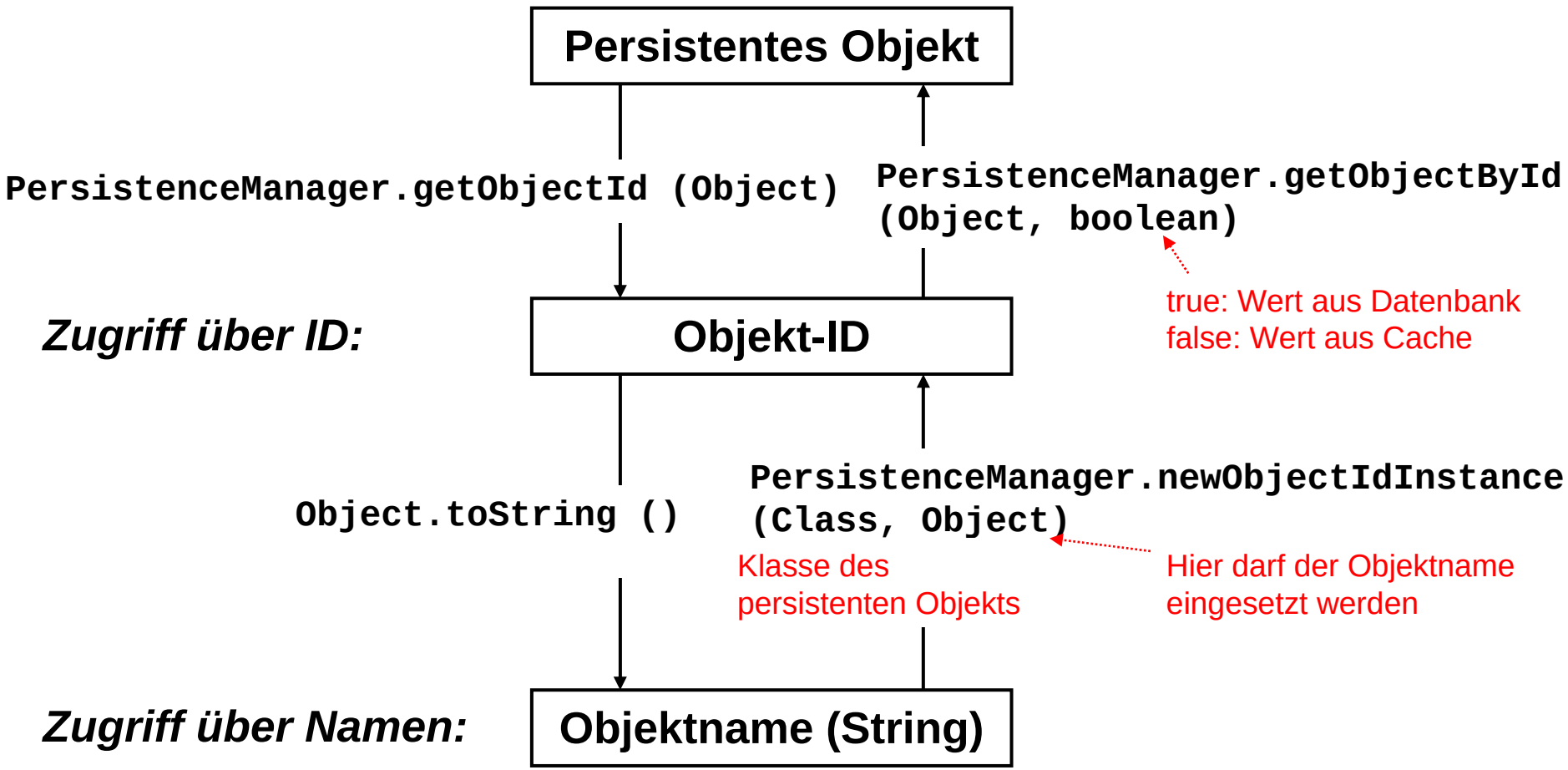
1) **Datastore identity** ☺ **wird von Versant voll unterstützt** ☺

2) **Application identity** ☹ **wird von Versant nicht unterstützt** ☹

3) **nondurable identity** ☹ **wird von Versant nicht unterstützt** ☹

JDO-Datenidentitätskonzepte

Möglichkeit des direkten Zugriffs auf ein Datenbankobjekt:



JDO-Datenidentitätskonzepte

JDO-Zugriffsmöglichkeiten auf Datenbankobjekte:

- **via Extent**
- **via JDO-Query**
- **via Object-ID / Objektnamen**

Konzept zum Management von Datenveränderungen: Lebenszykluszustände

Lebenszykluszustände eines Objekts

- **Jedes Objekt einer persistenzfähigen Klasse hat Zustand, der die Persistenz„geschichte“ beschreibt**
- **Die Zustände werden intern als flags abgespeichert**
- **Zustände werden durch bestimmte Aktionen implizit gesetzt**
- **Einige Zustände können abgefragt oder explizit gesetzt werden**
- **Gegenwärtige Zustände beeinflussen die Ausführung anderer Aktionen**

Lebenszykluszustände eines Objekts

Obligatorische Zustände:

- **transient**
- **persistent-new**
- **hollow**
- **persistent-clean**
- **persistent-dirty**
- **persistent-deleted**
- **persistent-new-deleted**

- **detached-clean**
- **detached-dirty**

} **nur in JDO 2.0**