Objektorientierte Datenbanken

Vorlesung 12 vom 08.07.2004

Dr. Sebastian Iwanowski
FH Wedel

Inhalt heute: Finale

JDBC: Java Database Connectivity

Touristeninformationssystem als Beispiel für objektorientiert modellierte Daten, die in einer relationalen Datenbank abgespeichert sind

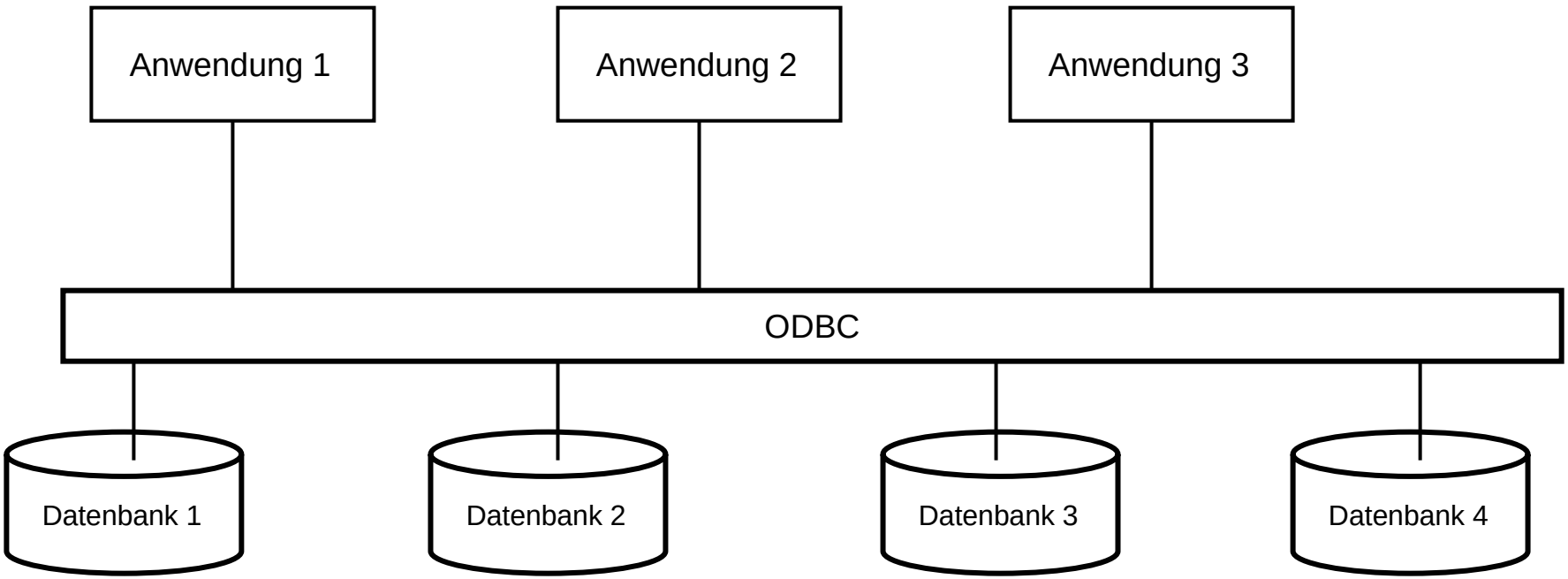
Zusammenfassung der Vorlesung OODB

JDBC: Java Database Connectivity

ODBC: Open Database Connectivity

Ziel

- **Standardisierter Zugriff auf beliebige Datenquellen mit SQL-Befehlen**

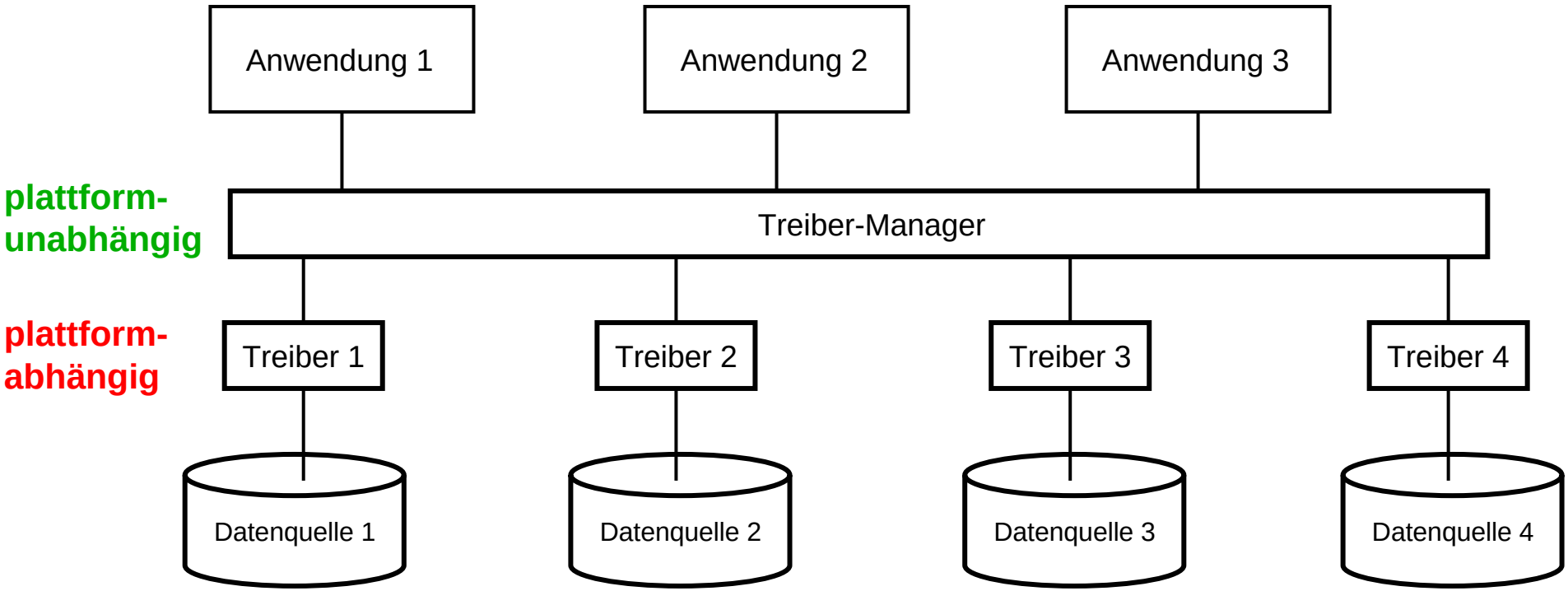


Geschichte

- **entwickelt Anfang der 90‘er Jahre mit Microsoft-Unterstützung**
- **Anwendungsfokus: C- / C++- Anwendungen**

ODBC: Open Database Connectivity

Architektur

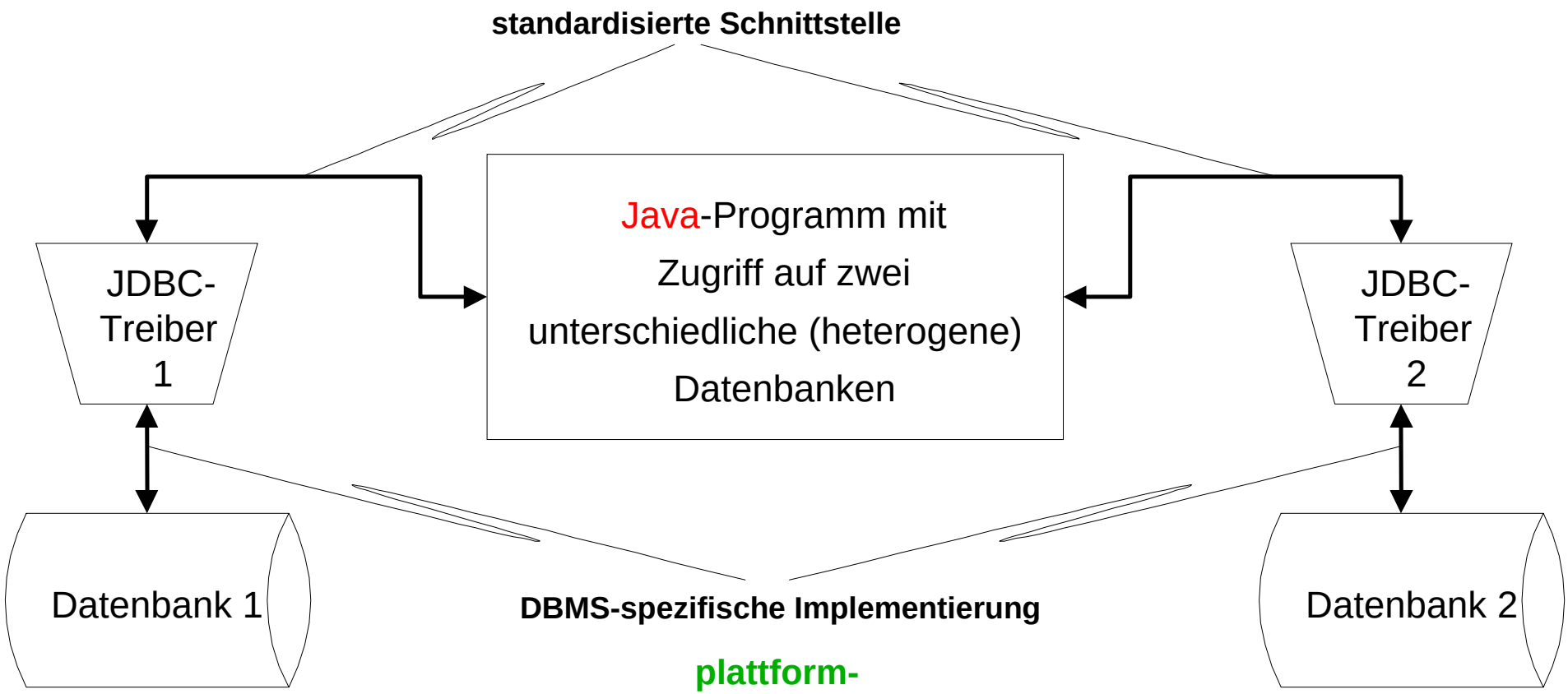


Mögliche Datenquellen:

- Desktop-Datenbanken (für Einzelbenutzer)
- Server-Datenbanken (für parallelen Mehrbenutzerbetrieb)
- Textdateien in beliebigem Format
- Tabellenkalkulationen

Datenquellen müssen keine richtigen Datenbanken sein

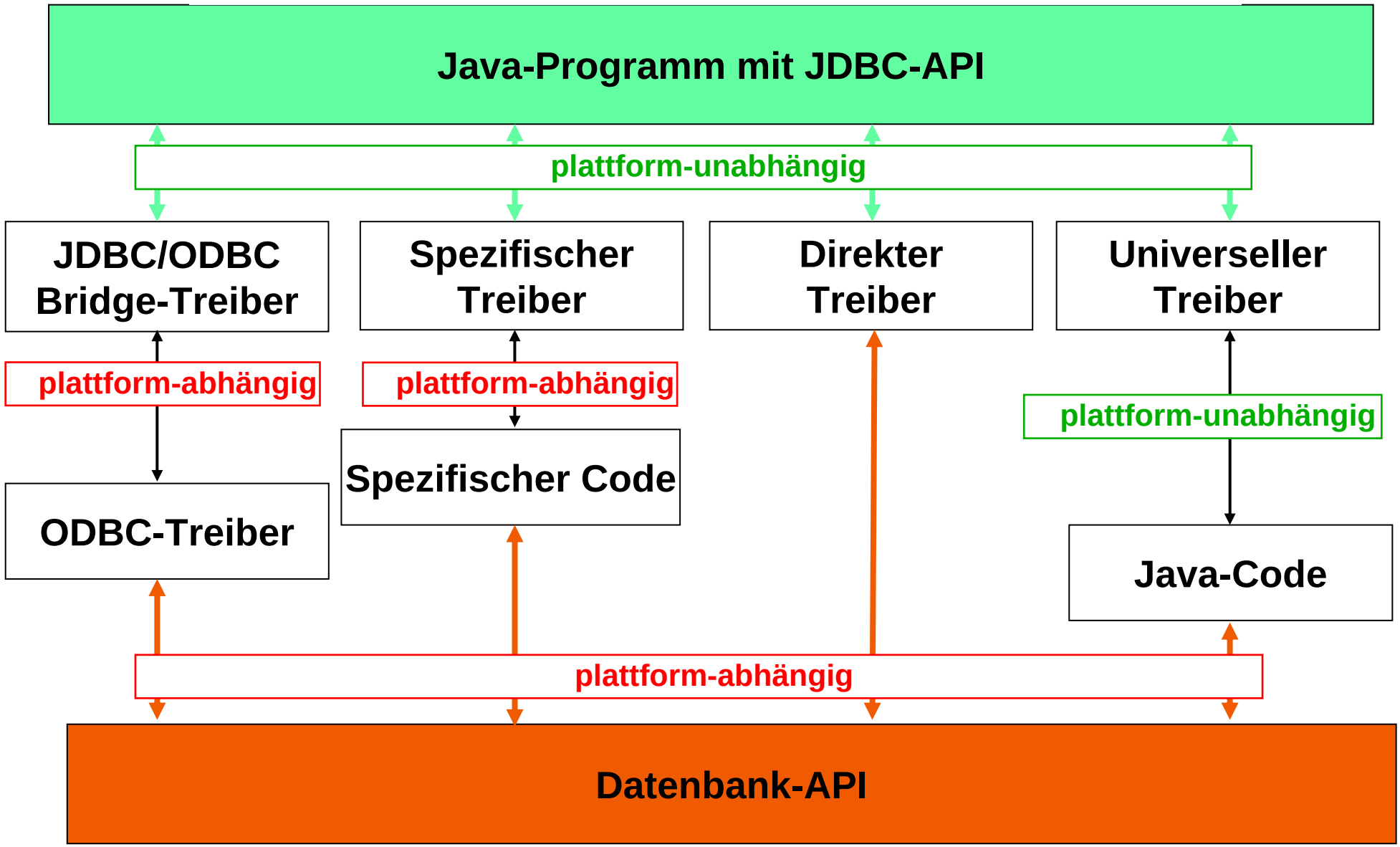
JDBC: Java Database Connectivity



plattform-unabhängig

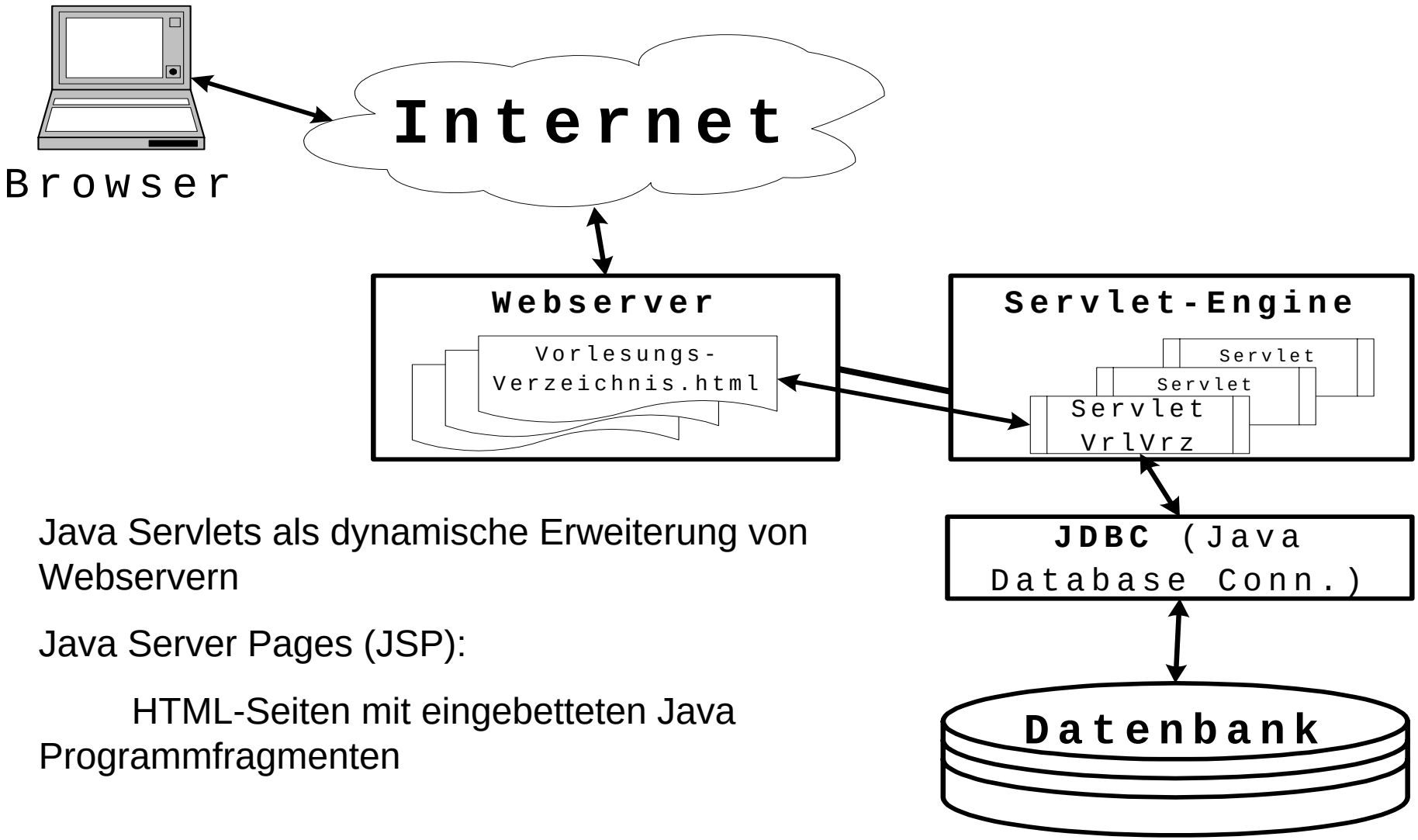
(im Prinzip …)

Anbindungsmöglichkeiten für JDBC



Aktuelle Verwendung von JDBC:

Anbindung von Datenbanken an das Internet



- Java Servlets als dynamische Erweiterung von Webservern
- Java Server Pages (JSP):

  HTML-Seiten mit eingebetteten Java Programmfragmenten

Java-API für JDBC

- **Laden des JDBC-Treibers**

  dynamisch:

  ```
  static Class Class.forName (String drivername)
  ```

  statisch: durch Spezifikation im properties-File

- **Verbindungsaufbau zur Datenbank**

  ```
  static Connection DriverManager.getConnection (String url)
  ```

- **Generierung eines Anfrageobjekts**

  ```
  Statement Connection.createStatement ()
  ```

  Hierdurch werden Eigenschaften der Antwort festgelegt.

- **Formulierung und Stellen der Frage**

  ```
  ResultSet Statement.executeQuery (String sqlQuery)
  ```

  generiert Datenbanktabelle (mit Zeilen und Spalten)

Java-API für JDBC

- **Navigieren in der Antwort:**

```
boolean ResultSet.next ()
boolean ResultSet.previous ()
```

navigiert zur nächsten bzw. vorigen Zeile der Antwort

```
String ResultSet.getString (int index)
int ResultSet.getInt (int index)
```

liest das Element in Spaltenposition index in der gegenwärtigen Zeile

```
String ResultSet.getString (String name)
int ResultSet.getInt (String name)
```

liest das Element in Spalte name in der gegenwärtigen Zeile

`ResultSet` bietet viele weitere nützliche Methoden.

Wichtig: Alle Zugriffe auf Datenbank mit `try ... catch` !

Die Interfaces `Connection`, `Statement`, `ResultSet` und die Klasse `DriverManager` befinden sich im package `java.sql`

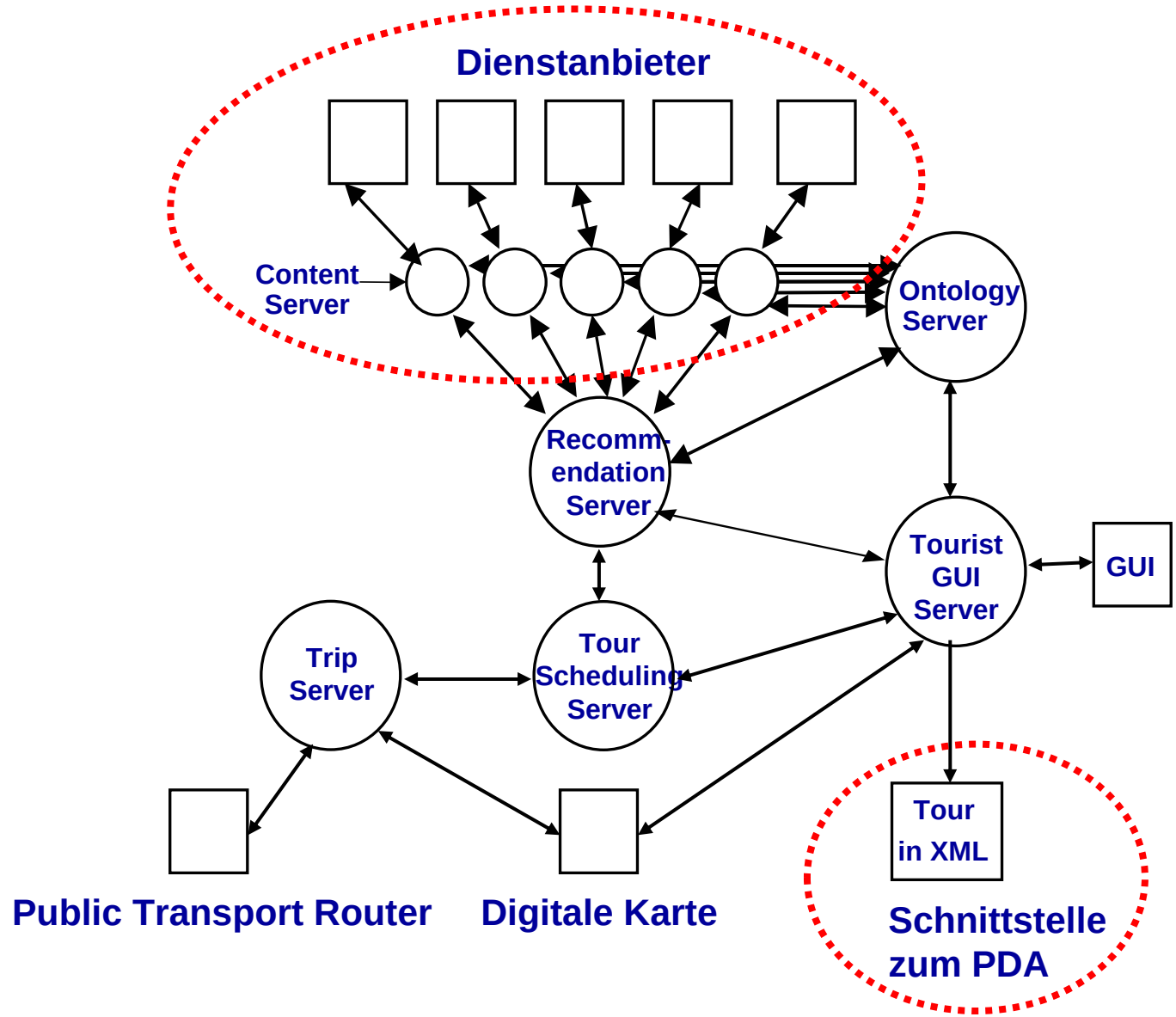
Java-API für JDBC: Einfaches Beispiel

```
try {
     Class.forName("oracle.jdbc.driver.OracleDriver");
     conn = DriverManager.getConnection
               ("jdbc:oracle:oci8:@lsintern-db", "nobody", "Passwort");
     sql_stmt = conn.createStatement();
     }
catch (Exception e) {
     System.err.println("Folgender Fehler ist aufgetreten: " + e);
     System.exit(-1);  }
try {
     ResultSet rset = sql_stmt.executeQuery(
       "select Name, Raum from Professoren where Rang = 'C4'");
     System.out.println("C4-Professoren:");
     while(rset.next())  {
          System.out.println
               (rset.getString("Name") + " " +  rset.getInt("Raum"));
                              }
     rset.close();
     }
catch(SQLException e) {System.out.println ("Error: " + e); }
 try {
     sql_stmt.close(); conn.close();
     }
catch (SQLException e) {
     System.out.println("Fehler beim Schliessen der DB: " + e);
                               }
```

Professoren			
PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Datenmodellierung und Datenbankanbindung im Touristeninformationssystem

Touristeninformationssystem



Touristeninformationssystem

zu beachtende Prinzipien:

- **Daten werden in Kategorien eingeteilt, die in Klassifikationsbäumen hierarchisch beschrieben sind (mit Attributen und Vererbung).**

  „Ontologien“

  objektorientiertes Modellierungskonzept

- **Die Kategorien dürfen von verschiedenen Content-Servern verschieden verfeinert werden.**

  ➔ Ontologien können nicht mit Java-Klassen modelliert werden !

- **Die Content-Server legen die Kategorien fest. Die konkreten Daten sind in Datenbanken abgelegt. Es bleibt der Implementierung eines Content-Servers überlassen, wie sie die Kategorieinformationen aus den Datenbanken herausziehen.**

  ... und das kann sehr mühsam sein wegen des Impedance Mismatch

  Objektorientierte Datenbanken wären hier sehr hilfreich !

- **Eine Datenbank ist ein lokales Hilfsmittel für einen Content-Server. Alle anderen Server des Touristeninformationssystems kommunizieren nur über plattformweit festgelegte Schnittstellen mit dem Content-Server, niemals mit der Datenbank.**

Touristeninformationssystem: Datenmodellierung

Beispiel einer Ontologie für Essensmöglichkeiten:

(in KQML, einer Spezifikationssprache für Agentensoftware)

```
<POI> ::= (POI (DictEntry POI)
    :subClasses (<SightseeingPOI> <DiningPOI> <ShoppingPOI> <EntertainmentPOI>)
    :name (DictEntry Name) <string>
    :address (DictEntry Address) <AddressPlace>
    :description (DictEntry Description) <DictEntry>
    :place (DictEntry Dummy) ({<GeoPlace>}+)
    :events (DictEntry EventsGeneral) ({<Event>}+))

<DiningPOI> ::=
    (DiningPlace (DictEntry DiningPlace)
        :subClasses
            ((Bistro (DictEntry Bistro)
                :subClasses ()
                :seatingFacilities (DictEntry SeatingFacilitiesExist) <boolean>
            (Restaurant (DictEntry Restaurant)
                :subClasses ()
                :seats (DictEntry NumberOfSeats) <number>
                :stars (DictEntry CookingHats) <number>))
        :cuisine (DictEntry Cuisine) <DictEntry>
```

Touristeninformationssystem: Datenmodellierung

XML-Schnittstelle für das PDA

⟨*OntologyTree*⟩ ::=

<ontology-tree language = ”⟨*String*⟩”>

<category-name> ⟨*String*⟩ </category-name>

[<sub-categories> {⟨*OntologyTree*⟩}+ </sub-categories>]

[<special-attributes> {<key> ⟨*String*⟩ </key>}+ </special-attributes>]

</ontology-tree>

Touristeninformationssystem: Datenmodellierung

XML-Schnittstelle für das PDA

⟨POI*OntologyTree*⟩ ::=

<ontology-tree language = ”⟨*String*⟩”>

<category-name> ⟨*String*⟩ </category-name>

[<sub-categories> {⟨POI*OntologyTree*⟩}+ </sub-categories>]

[<special-attributes> {<key> ⟨*String*⟩ </key>}+ </special-attributes>]

[<pois> ⟨*POIReference*⟩+ </pois>]

</ontology-tree>

Touristeninformationssystem: Datenmodellierung

XML-Schnittstelle für das PDA

⟨Event*OntologyTree*⟩ ::=

<ontology-tree language = ”⟨*String*⟩”>

<category-name> ⟨*String*⟩ </category-name>

[<sub-categories> {⟨Event*OntologyTree*⟩}+ </sub-categories>]

[<special-attributes> {<key> ⟨*String*⟩ </key>}+ </special-attributes>]

[<events> ⟨*EventReference*⟩+ </events>]

</ontology-tree>

XML-Schnittstelle: POIs

⟨*POIReference*⟩ ::= <poi-reference>
<poi-name> ⟨*String*⟩ </poi-name>
<file> ⟨*Filename*⟩ </file>
{⟨*GeoPlace*⟩}+
</poi-reference>

⟨*POI*⟩ ::= <poi language = "⟨*String*⟩">
<poi-name> ⟨*String*⟩ </poi-name>
{⟨*GeoPlace*⟩}+
{<timing-event> ⟨*Event*⟩ ⟨*TimeSchedule*⟩ </timing-event>}+
{<info-set>
[<text> ⟨*Filename*⟩ </text>]
[<image> ⟨*Filename*⟩ </image>]
[<video> ⟨*Filename*⟩ </video>]
[<audio> ⟨*Filename*⟩ </audio>]
</info-set>}
[<street> ⟨*String*⟩ </street>]
[<number> ⟨*String*⟩ </number>]
[<district> ⟨*String*⟩ </district>]
[<post-code> ⟨*String*⟩ </post-code>]
[<city> ⟨*String*⟩ </city>]
[<state> ⟨*String*⟩ </state>]
[<country> ⟨*String*⟩ </country>]
[<phone> ⟨*String*⟩ </phone>]
[<fax> ⟨*String*⟩ </fax>]
[<special-attributes>
{<key> ⟨*String*⟩ </key> <value> ⟨*String*⟩ </value>}+
</special-attributes>]
[<categories> {<name> ⟨*String*⟩ </name>}+ </categories>]
</poi>

Touristeninformationssystem: Datenbankanbindung

Realisierung mit vorhandenen Datenbanken verschiedener Anbieter:

- **Bertelsmann**
- **YellowPages**
- **iPublish**

Kleiner Spezialanbieter für Berlin

benutzte als einziger hierarchisch geordnete Kategorien

Benutzte Datenbank: Microsoft Access

keine JDBC-Anbindung

daher Benutzung einer ODBC-JDBC-Bridge

Zusammenfassung der Vorlesung OODB

Zusammenfassung: OODB

Vorlesung 1: Grundlegende Prinzipien relationaler und objektorientierter Datenmodellierung

Gemeinsamkeiten und Unterschiede, Impedance Mismatch

Vorlesung 2: Synthese von Vorlesung 1: Objektorientierte Datenbanken

ODMG als erste erfolgreiche Standardisierung: Wurzeln Smalltalk und C++, Objektmodell, verschiedene Persistenzkonzepte

Vorlesung 3: ODMG, 2. Teil

Transaktionskonzept, Gewährleistung verschiedener Integritätsbedingungen, Java-Spezifikationen der ODMG, ~~Realisierung in Fast-Objects~~

Vorlesung 4: ODMG-Anfragesprache OQL

Warum eine Anfragesprache, Ähnlichkeit zu SQL, Navigieren in Objektpfaden, ~~Direkte Anfragen, FastObjects-OQL~~

Zusammenfassung: OODB

Vorlesung 5-10: JDO

Der Kern dieses Vorlesungszyklus, Zusammenfassung am Ende von Vorlesung 10

Vorlesung 11: Datenbanken in betriebswirtschaftlichen Anwendungen

SAP und Datawarehouselösungen: Grundlegende Zielsetzungen, Unterschiede

Erklärung der wichtigsten Begriffe und Probleme von Datawarehouselösungen

Vorlesung 12: JDBC

Unterschied zu ODBC, verschiedene Datenbankanbindungsmöglichkeiten, minimal notwendige Operationen zur Anbindung, Nachteile zu OODBs

Das war die Vorlesung OODB:

Reaktion auf die Kritik und Anregungen