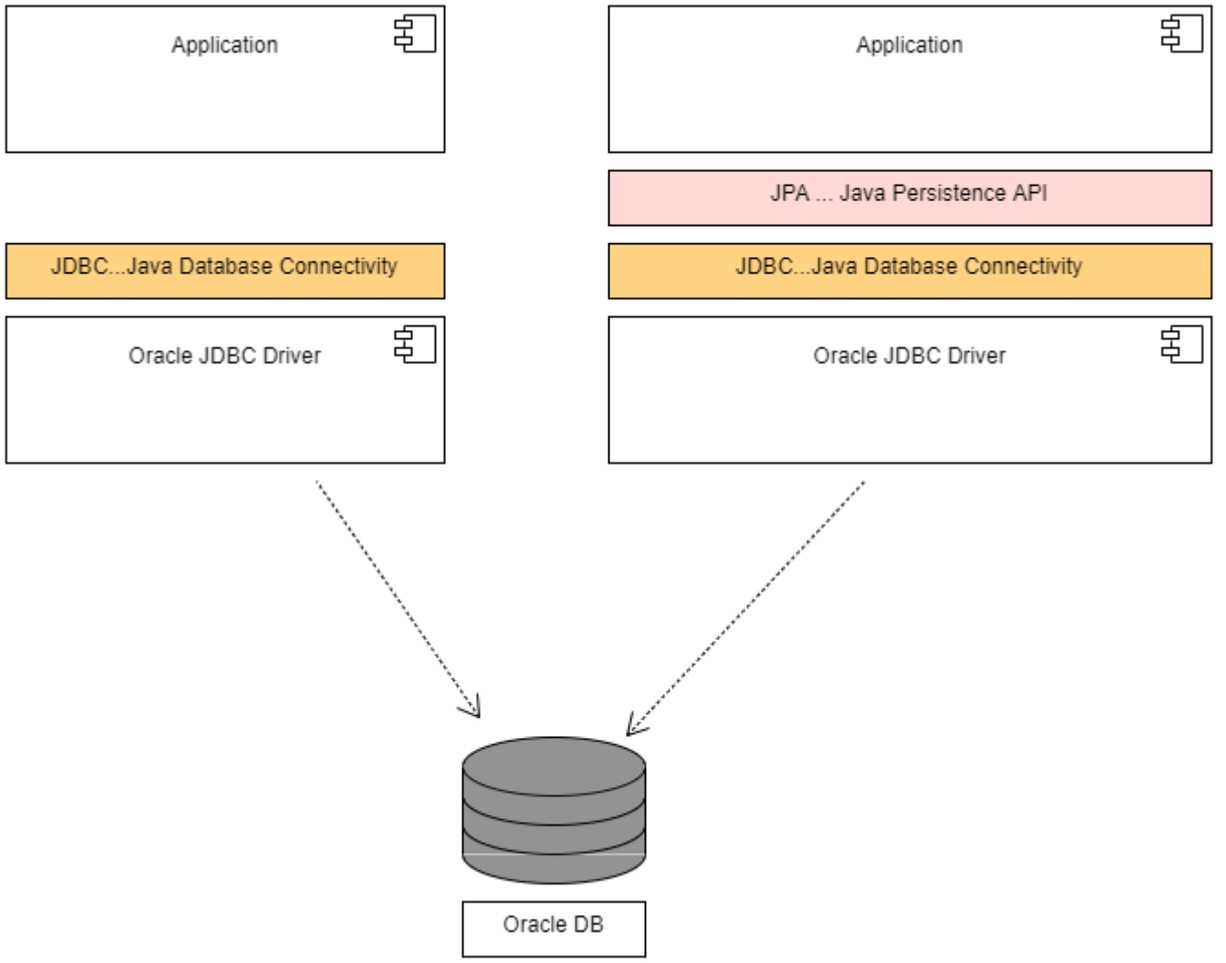
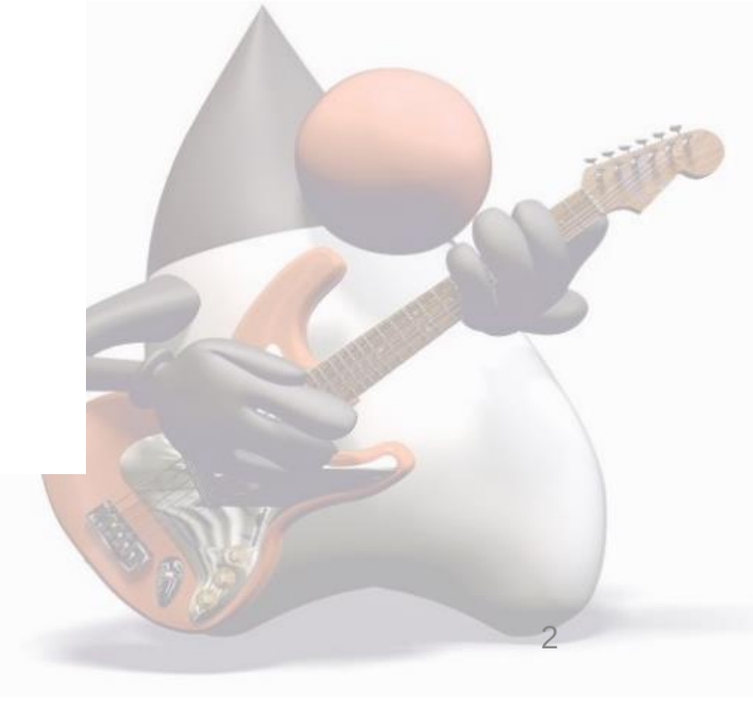
Java Programmierung

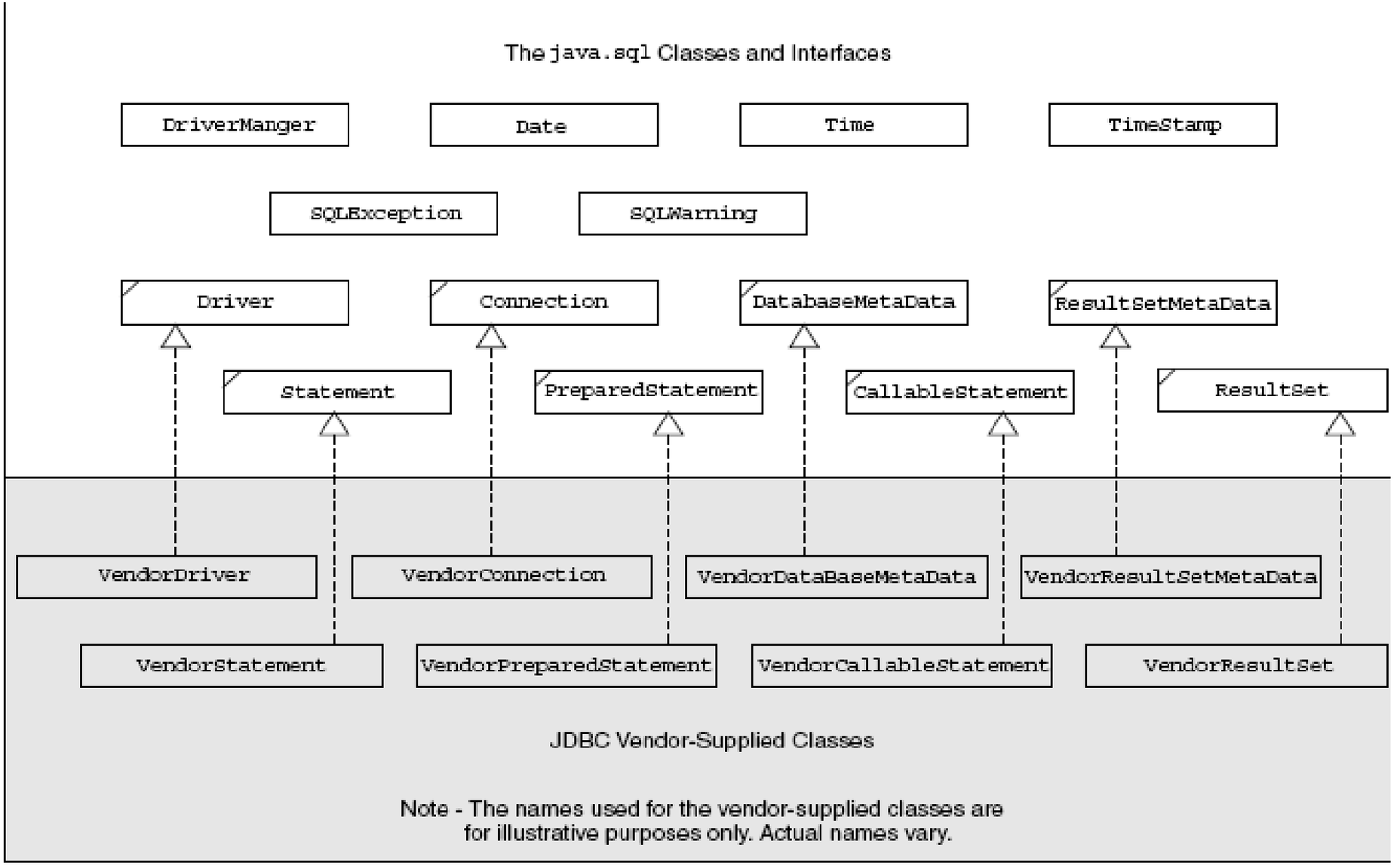
Datenbankzugriff mit JDBC


Application
Application
JPA ... Java Persistence API
JDBC...Java Database Connectivity
JDBC...Java Database Connectivity
Oracle JDBC Driver
Oracle JDBC Driver
Oracle DB



JDBC … Java Database Connectivity

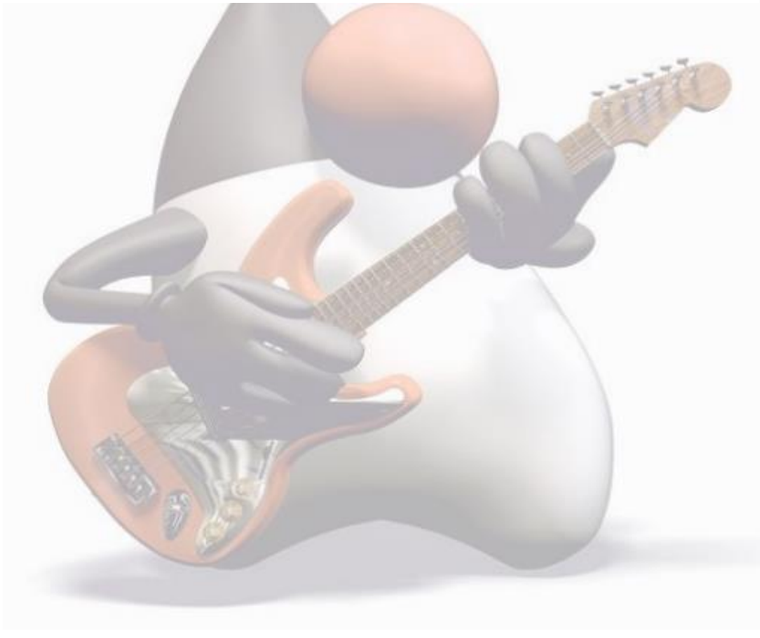
- einfachste Möglichkeit des DB Zugriffs in Java
- von allen DB Herstellern unterstützt
- vereinheitlicht Verbindungen, Authentifizierung, Transaktionen
- Statements werden weiterhin im SQL-Dialekt des DB-Herstellers abgesetzt


The java.sql Classes and Interfaces
DriverManger
Date
Time
TimeStamp
SQLException
SQLWarning
Driver
Connection
DatabaseMetaData
ResultSetMetaData
Statement
PreparedStatement
CallableStatement
ResultSet
VendorDriver
VendorConnection
VendorDataBaseMetaData
VendorResultSetMetaData
VendorStatement
VendorPreparedStatement
VendorCallableStatement
VendorResultSet
JDBC Vendor-Supplied Classes
Note - The names used for the vendor-supplied classes are
for illustrative purposes only. Actual names vary.

Vorbereitung Datenbankzugriff:

- Herunterladen JDBC Treiber von der Download Seite des DB-Herstellers
- JDBC Treiber jar File als Bibliothek dem Projekt hinzufügen
- JDBC Connection String aus der Dokumentation des Treibers entnehmen
- JDBC Connection String mit einem JDBC Client testen
  (z.b. Eclipse Data Source Explorer)

```
public class KundenDAO {
    // Definition der SQL Statements (können auch DB-proprietäre Statements
    // sein)
    private static final String SQL_INSERT =
        "INSERT INTO KONTEN.KUNDE(KUNDENNUMMER,NAME,ADRESSE,GEBDATUM) VALUES(?,?,?,?))";
    private static final String SQL_SELECT_ALL =
        "SELECT KUNDENNUMMER,NAME,ADRESSE,GEBDATUM FROM KONTEN.KUNDE";

    public List<Kunde> getAlleKunden() {
        List<Kunde> kunden = new LinkedList<Kunde>();
        try (Connection connection = DriverManager.getConnection(
                    "jdbc:derby://localhost:1527/konten","username","password")) {
            try (PreparedStatement stmt = connection.prepareStatement(SQL_SELECT_ALL)) {
                // Ausführen des SQL Statements
                try (ResultSet rs = stmt.executeQuery()) {
```



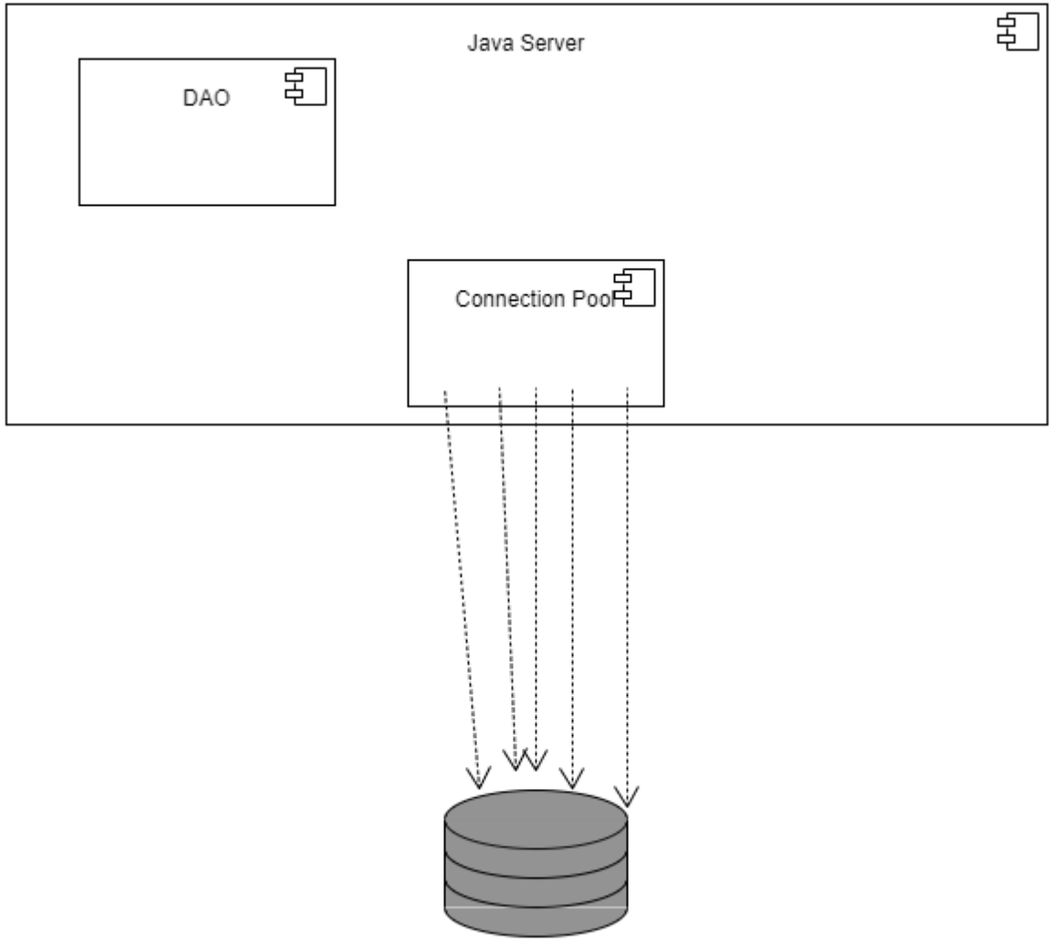
```
// Auslesen des ResultSets (wird während des Auslesens
// schrittweise von der DB geholt)
while (rs.next()) {
          // Auslesen der Spalte Kundennummer (SQL-BIGINT //-> Java Long)
    Long kundennummer = rs.getLong("KUNDENNUMMER");
    // Auslesen der Spalte Name
          //(SQL-VARCHAR -> Java String)
    String name = rs.getString("NAME");
    // Auslesen der Adresse
    String adresse = rs.getString("ADRESSE");
    // Auslesen der Spalte Adresse
          //(SQL-DATE -> Java Date)
    Date gebDatum = rs.getDate("GEBDATUM");
    // Anlegen eines neuen Kundenobjektes
    Kunde k = new Kunde(kundennummer,name,adresse,gebDatum);
    // Hinzufügen des Kunden zur Ergebnisliste
    kunden.add(k);
}
```

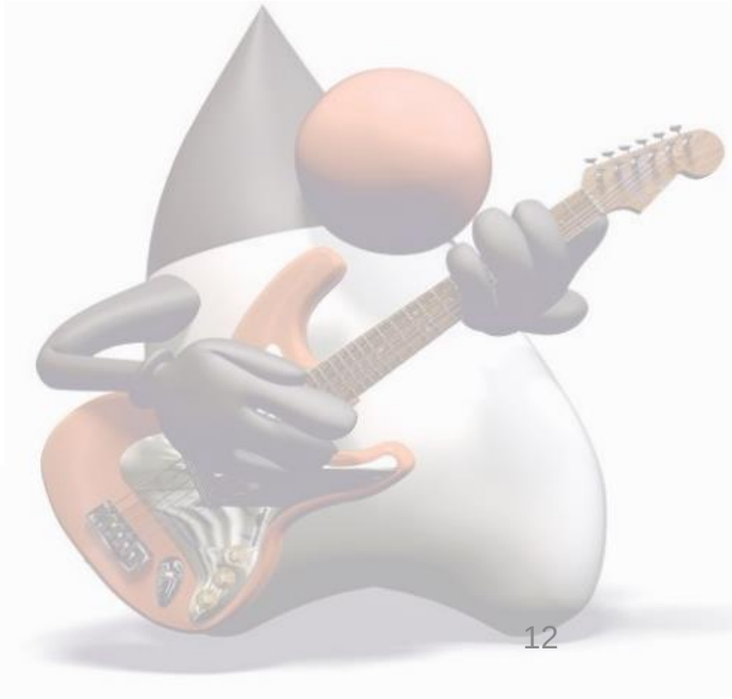
```
                }
            }
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return kunden;
    }
}
```

```
private static final String SQL_INSERT =
"INSERT INTO KONTEN.KUNDE(KUNDENNUMMER,NAME,ADRESSE,GEBDATUM) VALUES(?,?,?,?))";

    public void insertKunde(Kunde kunde) {
        List<Kunde> kunden = new LinkedList<Kunde>();
        ResultSet rs = null;
        try (Connection connection = DriverManager.getConnection(
                "jdbc:derby://localhost:1527/konten","username","password")) {
            try (PreparedStatement stmt = connection.prepareStatement(SQL_INSERT)) {
                // automatisches Commit ausschalten
                connection.setAutoCommit(false);
```

```
            // Setzen der Parameter
            stmt.setLong(1, kunde.getKundenummer());
            stmt.setString(2, kunde.getName());
            stmt.setString(3, kunde.getAdresse());
            java.sql.Date sqlDatum = new java.sql.Date(kunde.getGebDatum().getTime());
            stmt.setDate(4, sqlDatum);
            // Ausführen des SQL Statements
            int anzahlGeaendeterDatensaetze =
                        stmt.executeUpdate();
            //Commit
            connection.commit();
        }
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        KundenDAO dao = new KundenDAO();

        try {
            SimpleDateFormat format = new SimpleDateFormat("dd.MM.yyyy");
            Date gebDatum = format.parse("06.09.1932"); //throws ParseException
            Kunde kunde = new Kunde(9991, "Frank Stronach", "Kanada", gebDatum);
            dao.insertKunde(kunde);
        } catch (ParseException e) {
            e.printStackTrace();
        }

        List<Kunde> kunden = dao.getAlleKunden();
        System.out.println(kunden);
    }
}
```

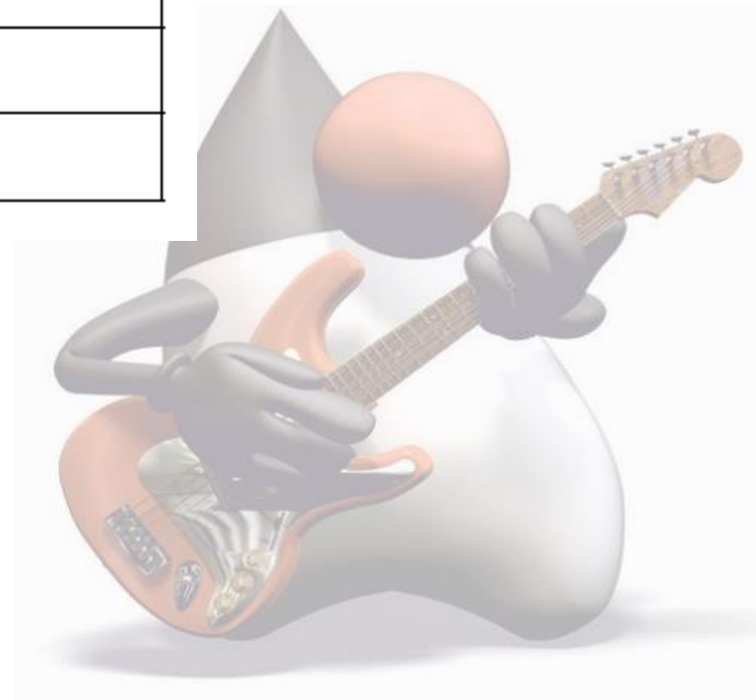

Java Server
DAO
Connection Pool



SQL Type	Java Technology Type
CHAR	String
VARCHAR	String
LONGVARCHAR	String
NUMERIC	java.math.BigDecimal
DECIMAL	java.math.BigDecimal
BIT	boolean
TINYINT	byte
SMALLINT	short
INTEGER	int
BIGINT	long
REAL	float
FLOAT	double

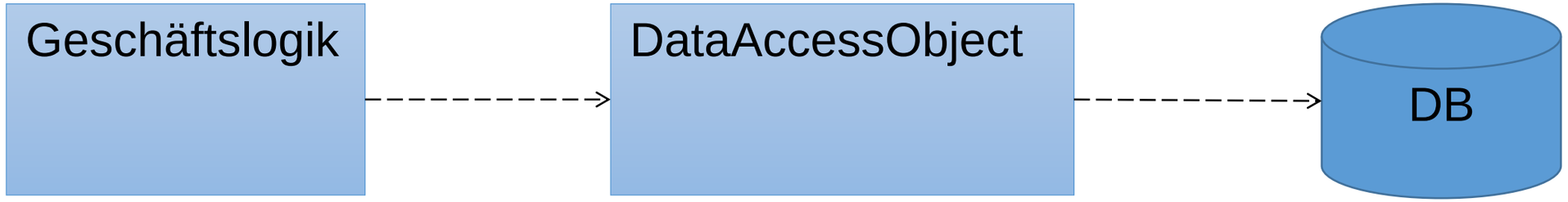
Umwandlung Datentypen Java ←→ SQL

SQL Type	Java Technology Type
DOUBLE	double
BINARY	byte[]
VARBINARY	byte[]
LONGVARBINARY	byte[]
DATE	java.sql.Date
TIME	java.sql.Time
TIMESTAMP	java.sql.Timestamp



Das Data Access Object (DAO) Design Pattern:

- Der Datenbankzugriff sollte in spezialisierten Klassen gekapselt werden



Beispielschema für Übung (Derby DB)

```
CREATE SCHEMA KONTEN;

CREATE
    TABLE KONTEN.KONTEN
    (
        nummer BIGINT NOT NULL,
        inhaber VARCHAR(40),
        saldo BIGINT,
        typ VARCHAR(10),
        PRIMARY KEY (nummer)
    );
```

Übung

- Schreiben Sie eine Klasse AccountDAO, die Accounts in die Datenbank schreiben und von ihr lesen kann
- AccountDAO soll die Methoden save, findById, findAll, remove implementieren
- Schreiben Sie Unit Tests, um Ihren DAO zu testen

Java Programmierung

Ende Datenbankzugriff