

# JDBC

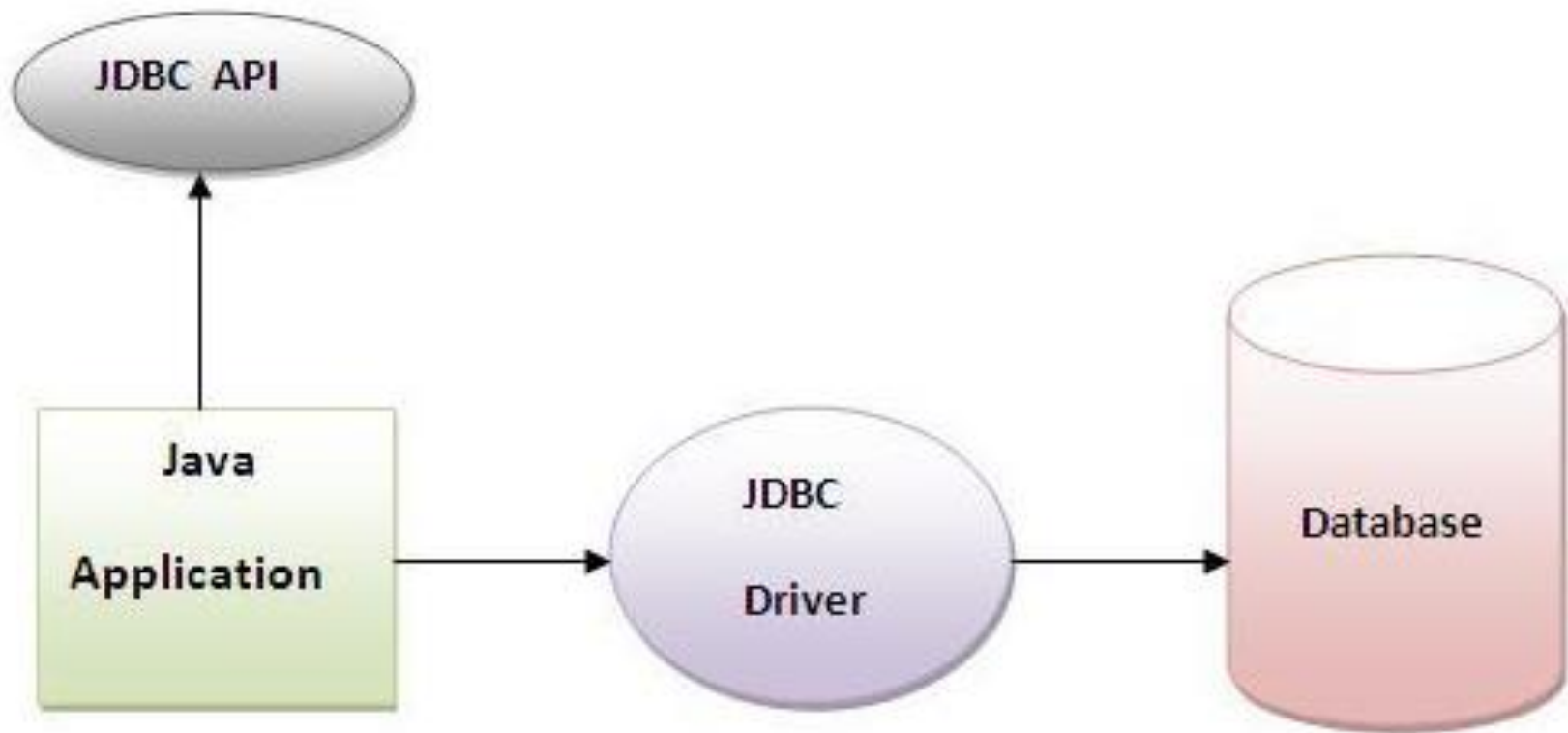
# Mål

- Kunne anvende JDBC til at
    - få forbindelse til MySql
    - afvikle forespørgsler
    - indsætte og ændre data
- fra et Java-program.

# Litteratur

- <https://www.tutorialspoint.com/jdbc/jdbc-introduction.htm>

# JDBC



# JDBC driveren

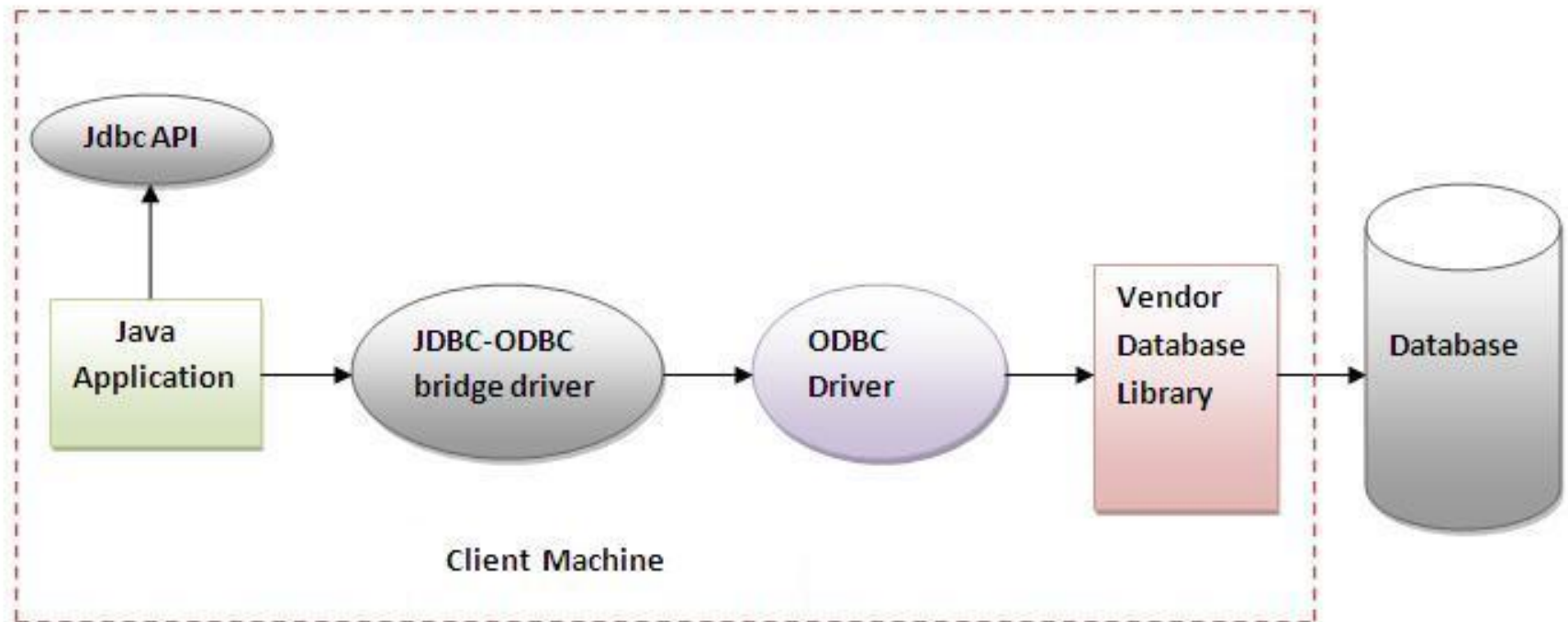
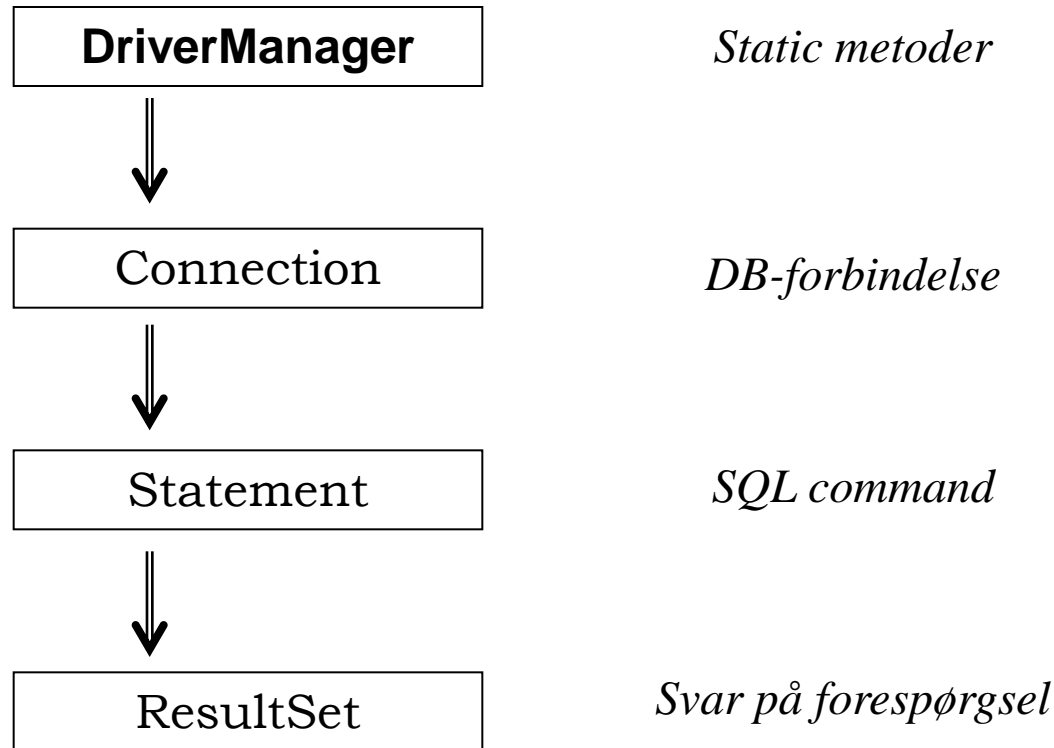


Figure- JDBC-ODBC Bridge Driver

# 5 Steps

1. Register the driver class
  2. Creating connection
  3. Creating statement
  4. Executing queries
  5. Closing connection
- <http://www.javatpoint.com/steps-to-connect-to-the-database-in-java>

# Elementer i JDBC



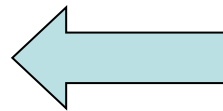
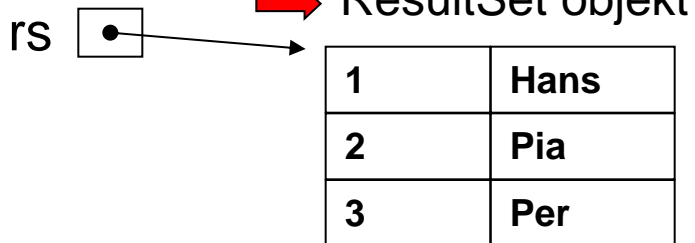
# Bruge JDBC

- Importer JDBC klasserne (java.sql.\*)
- Load JDBC driveren ind i memory
- Opret forbindelse til databasen
  - få et **Connection**-Objekt
  - få et **Statement**-Objekt
- Afvikl SQL
  - executeUpdate()**
  - executeQuery()**
    - modtag et **ResultSet**-Objekt med svaret
- Luk **Statement** og **Connection**



# ResultSet

```
String query = "SELECT * FROM perons";  
rs = statement.executeQuery(query);  
ArrayList<Person> pl = new ArrayList<Person>();  
while (rs.next()) {  
    pl.add(new Person (rs.getInt(1),  
                       rs.getString(2))  
};
```



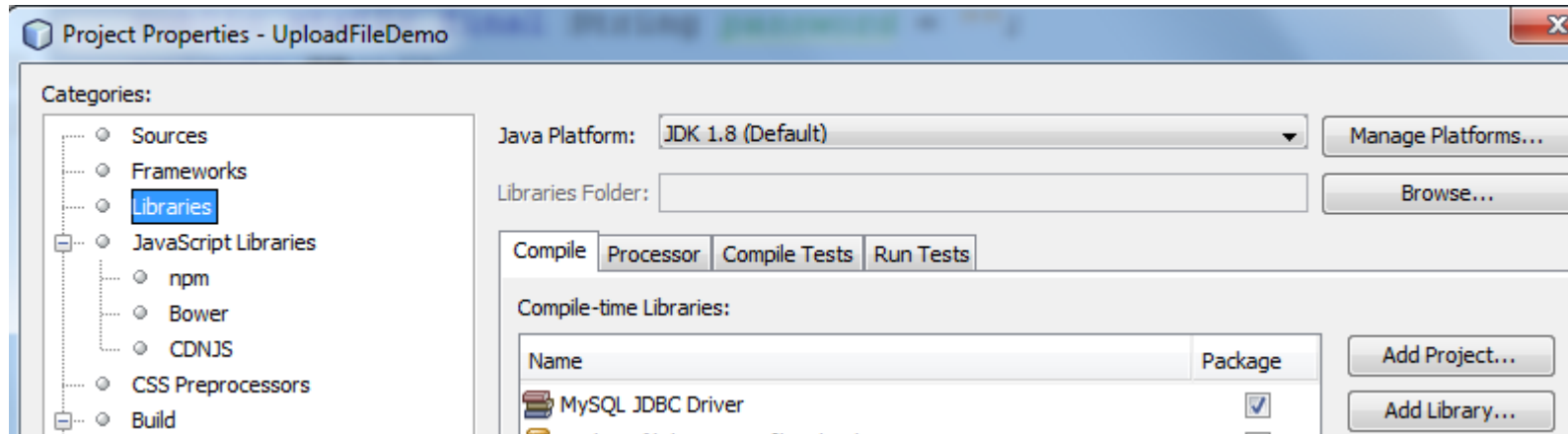
Persons

id	name
1	Hans
2	Pia
3	Per

# PreparedStatement

- adskiller SQL kode fra data-værdier
  - øget sikkerhed ift. SQL-injection
- muliggør performance gevinst, når samme SQL instruktion skal udføres **mange gange** blot med udskiftning af nogle parametre

# Load driver



```
try
{
    Class.forName ("oracle.jdbc.driver.OracleDriver");
}
catch (ClassNotFoundException e)
{
    System.out.println ("Can't find driver");
}
```

# Connection

- En forbindelse til databasen oprettes ved kald til `static Connection getConnection (String url, String user, String password)` i DriverManager klassen.
- Herefter sker al kommunikation, som oprettelse af Statement's, commit osv, med databasen via det returnerede **connection-objekt**

```
public class DB {
    public static final String driver =
"com.mysql.jdbc.Driver";
    public static final String url =
"jdbc:mysql://localhost/test";
    public static final String username = "root";
    public static final String password = "root";
    private DB(){}
    public static Connection getConnection(){
        Connection conn = null;
        try{
            Class.forName(driver);
            conn =
DriverManager.getConnection(DB.url,DB.username,DB.password);

        } catch (ClassNotFoundException se){
            se.printStackTrace();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
        return conn;
    }
}
```

# Statement

- Et statement-objekt repræsenterer den SQL instruktion, man vil have udført
- Der skelnes specielt mellem
  - **Queries** ( SELECT )
  - **Non Queries** ( INSERT, UPDATE, DELETE )

# Statement

- Et statement-objekt oprettes af Connection objektet:
  - `createStament()` // Statement
  - `prepareStatement()` // PreparedStatement

# Statements

- En forespørgsel afvikles med metoden

```
ResultSet rs =  
    statement.executeQuery(..)
```

- En ændring afvikles med metoden

```
int numberChangedRows =  
    statement.executeUpdate(...)
```



# ResultSet

- Resultatet af en forespørgsel (executeQuery) returneres som et **ResultSet**
- Et **ResultSet** er et objekt med en liste af elementer
- Hvert element indeholder data fra én tuple
- Metoden next() "bladrer" ét trin frem i listen
- En celle i tuplen hentes med **getXXX()**

ex:

```
ResultSet rs = statement.executeQuery(SELECT ..);  
rs.next();  
String name = rs.getString(1);
```

# Resultset 2

- Et resultset løbes igennem med en løkke, typisk

```
while(rs.next())  
{  
    id    = rs.getInt(1);  
    name  = rs.getString(2);  
    Person p = new Person(id, name);  
    personListe.add(p);  
}
```

# PreparedStatement

- PreparedStatement opererer med såkaldte **pladsholdere**, der angives med et **spørgsmålstegn**

– **ex:**

```
SELECT * FROM persons  
WHERE id = ? AND name = ?
```

- Pladsholderne nummeres fra venstre og starter med 1

# PreparedStatement (fortsat)

- Aktuelle værdier indsættes på de enkelte pladsholdere med
  - **setXXX()**, hvor XXX er datatype

ex:

```
setInt(1,17)
```

```
setString(2,"Donald Duck")
```