DB Apps Introduction

Connecting via JDBC, Executing Statements, SQL Injection, Advanced Concepts

SoftUni Team

Technical Trainers

Software University

http://softuni.bg

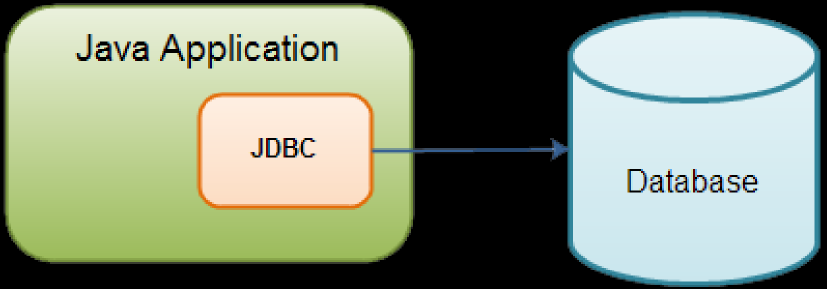
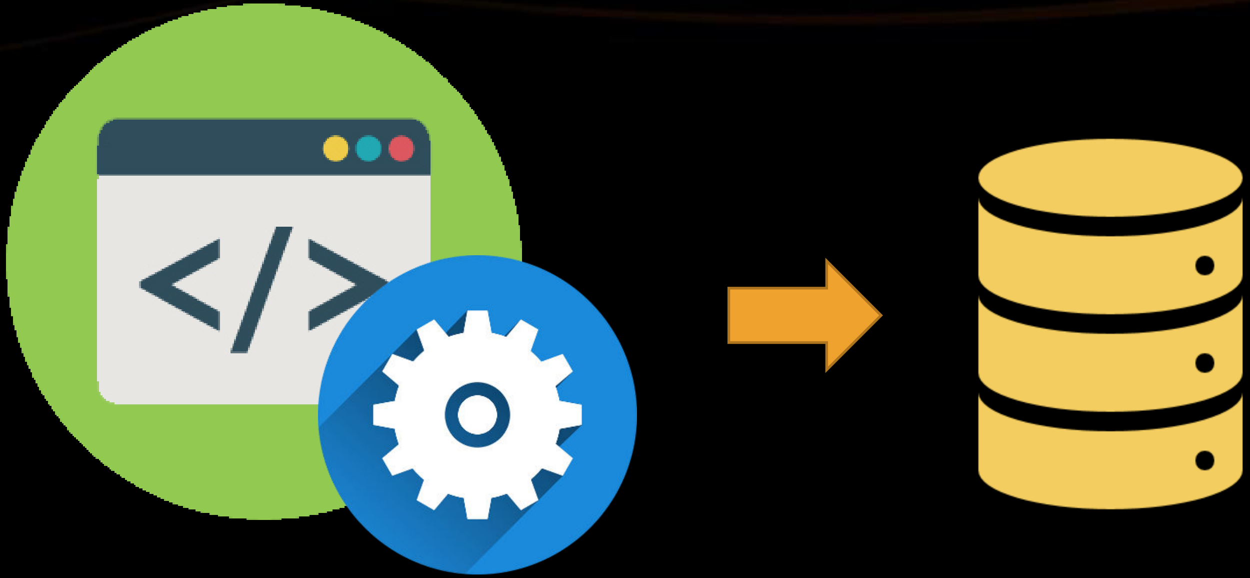


Table of Content

Questions

sli.do

#db-advanced

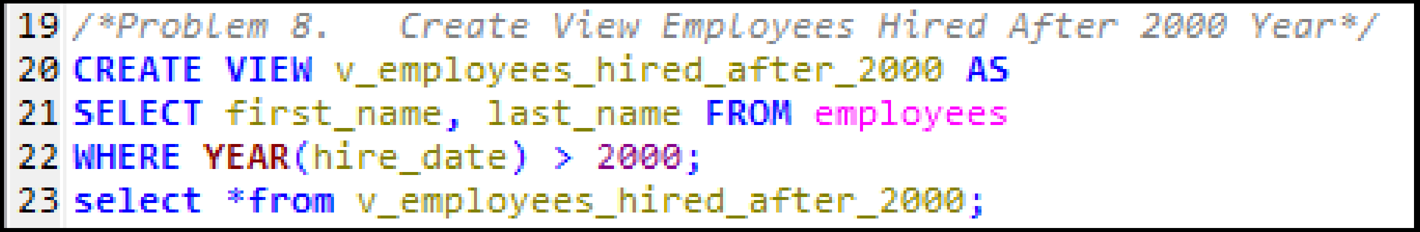


Application to Database Connection

Accessing data via client application

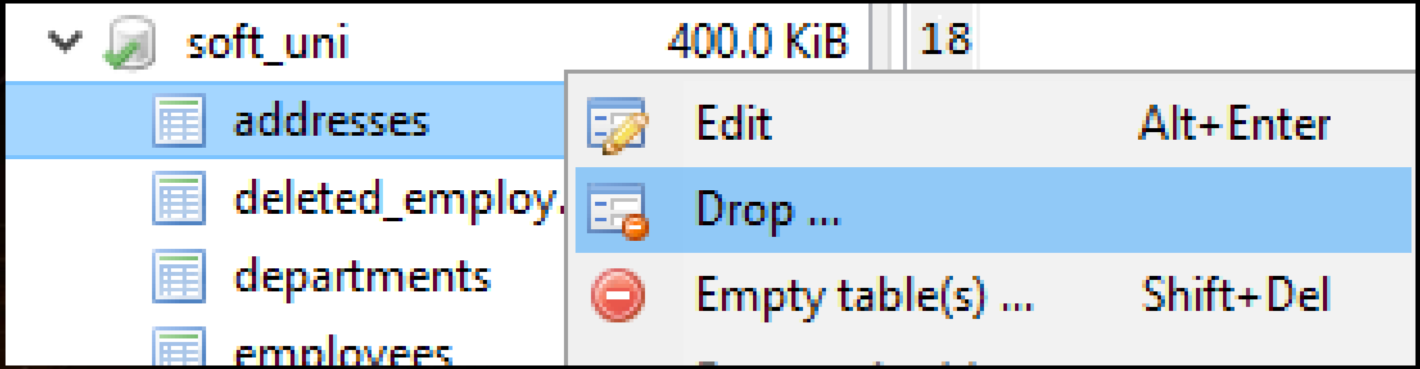
DB Basics

- So far we've been retrieving data by:
  - Writing and executing SQL queries



```
/*Problem 8.   Create View Employees Hired After 2000 Year*/
CREATE VIEW v_employees_hired_after_2000 AS
SELECT first_name, last_name FROM employees
WHERE YEAR(hire_date) > 2000;
select *from v_employees_hired_after_2000;
```

  - Using the GUI (HeidiSQL) functionalities



ORM Frameworks Overview

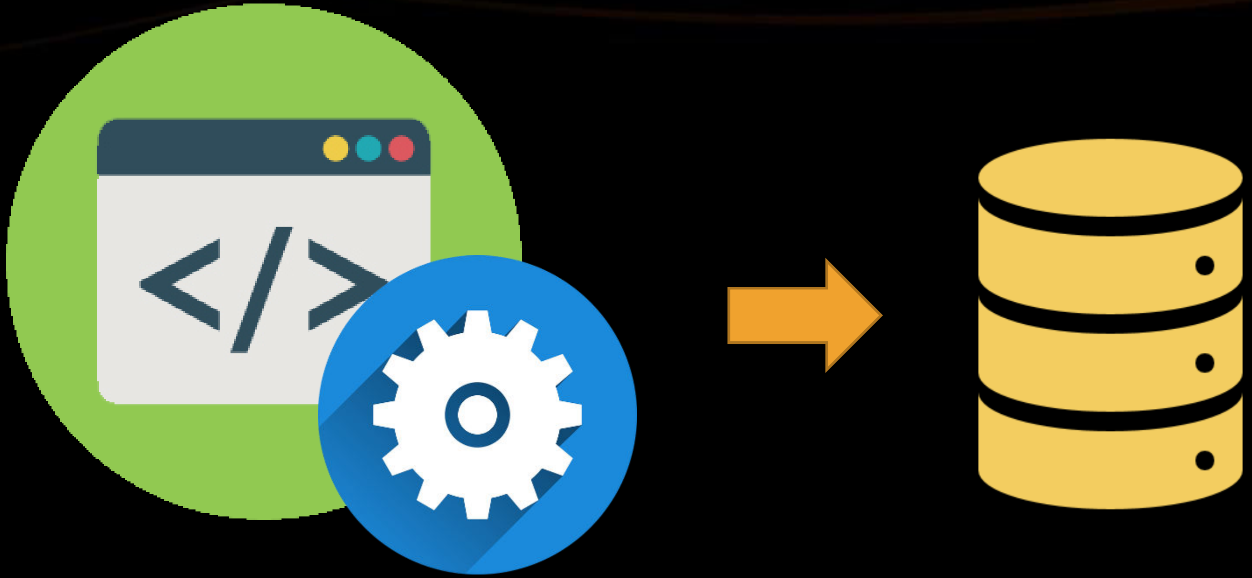
- In development programmers use object relational mapping frameworks
  - Mapping Java classes and data types to DB tables and SQL data types
  - Generate SQL calls and relieves the developer from the manual handling
    - E.g. (pseudo-code)

```
User user = new User("Peter", 25);
dbManager.saveToDB(user);
```

SQL Encapsulated in method

ORM Frameworks Overview (2)

- ORM frameworks do not drop the need to write SQL!
  - At some point you might need some manual query optimization
- ORM Frameworks examples:
  - Java – Hibernate, EclipseLink, TopLink…
  - .NET – Entity Framework, NHibernate…
  - PHP – Doctrine, Laravel(Eloquent)...



Application to Database Connection

Demo

Connection to DB via Java app Demo

- Download the demo from course instance
- You are given a simple application that:
  - Establishes connection with the "soft_uni" DB
  - Executes simple MySQL statement to retrieve the names of employees by given salary criteria

Connection to DB via Java app Demo (1)

- Lets analyze the program:
  - Connection to DB is established by asking the user to give credentials:

```
System.out.print("Enter username default (root): ");
String user = sc.nextLine();
user = user.equals("") ? "root" : user;
…

System.out.print("Enter password default (empty):");
String password = sc.nextLine().trim();
…
```

Connection to DB via Java App Demo (1)

- Using an external library (MySQL Connector/J) we make a connection via a DriverManager and a Connection class

```
Properties props = new Properties();
        props.setProperty("user", user);
        props.setProperty("password", password);

Connection connection =
DriverManager.getConnection("jdbc:mysql://localhos
t:3306/soft_uni", props);
```

Connection to DB via Java App Demo (2)

- We retrieve a result by the ResultSet and PreparedStatement class

```
PreparedStatement stmt =
connection.prepareStatement("SELECT * FROM employees
WHERE salary > ?");


String salary = sc.nextLine();
stmt.setDouble(1, Double.parseDouble(salary));
ResultSet rs = stmt.executeQuery();
```

SQL Query

Salary criteria by user input

Runs the SQL statement and returns retrieved result

Connection to DB via Java App Demo (3)

- Iterating over result

Retrieved data

```
while(rs.next()) {
        System.out.printf("%s  %s",
        rs.getString("first_name")
        rs.getString("last_name"));
}
```

The ResultSet is a set of table rows

Demo Conclusion

- We can access databases on a programmer level
  - No manual actions needed
- In a bigger applications we can:
  - Encapsulate custom SQL logic in methods
  - Achieve database abstraction

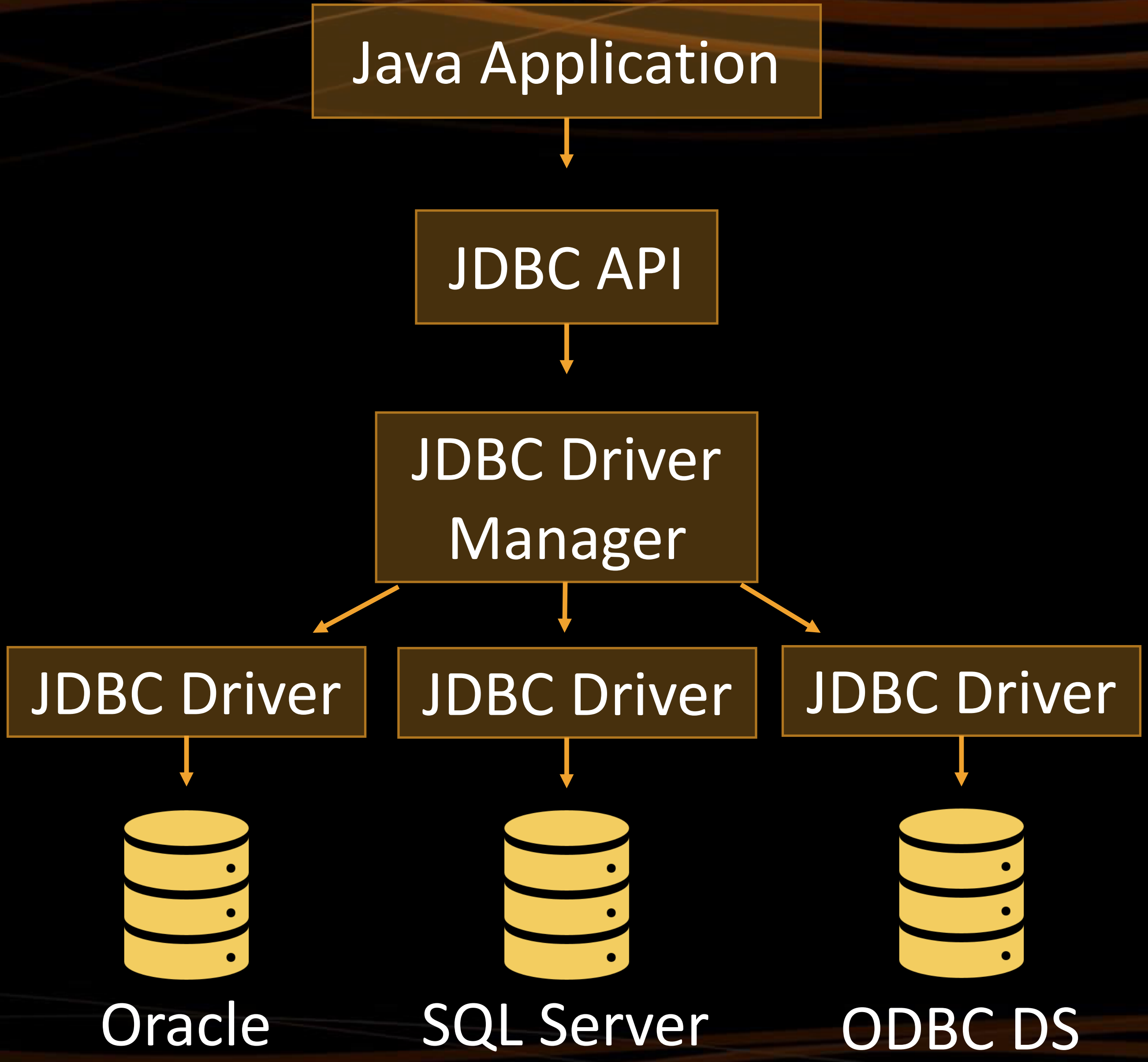
Java Database Connection

Client access to a database

Java Database Connectivity (JDBC)

- JDBC is a standard Java API for database-independent connectivity
- Includes APIs for:
  - Making a connection to a database
  - Creating and executing SQL queries in the database
  - Viewing & Modifying the resulting records

JDBC Architecture



JDBC Architecture (2)

- JDBC API – provides the connection between the application and the driver manager
- JDBC Driver Manager – establishes the connection with the correct driver
  - Supports multiple drivers connected to different types of databases
- JDBC Driver - handles the communications with the database

JDBC API

- JDBC API provides several interfaces and classes:
  - DriverManager – matches requests from the application with the proper DB driver
  - Driver – handles the communication with the DB server
  - Connection – all methods for contacting a database
  - Statement – methods and properties that enable you to send SQL
  - ResultSet – retrieved data (set of table rows)
  - SQLException

JDBC API – ResultSet Class

- ResultSet maintains a cursor pointing to its current row of data
  - Not updatable
  - Iterable only once and only from the first row to the last row
- Provides getter methods for retrieving column values from the current row
  - E.g. from previous demo:

```java
while(rs.next()) {
        System.out.printf("%s %s",
rs.getString("first_name"), rs.getString("last_name"));}
```

Getter method

Column name

JDBC API – ResultSet Class

- Retrieved information is reached by getter methods:
  - E.g.:
    - getString('column_name')
    - getDouble('column_name')
    - getBoolean('column_name') etc.
- The driver converts the underlying data to the Java type

java.sql* and MySQL Driver

- The java.sql package provides all previously mentioned JDBC classes
- In order to work with JDBC we need to download a MySQL Driver – Connector/J
  - It can be found on the following webpage: https://dev.mysql.com/downloads/connector/j/

MySQL Driver Connection

- Connection with the database is established via connection string
  - jdbc:<driver protocol>:<connection details>
  - E.g. connection from previous demo:

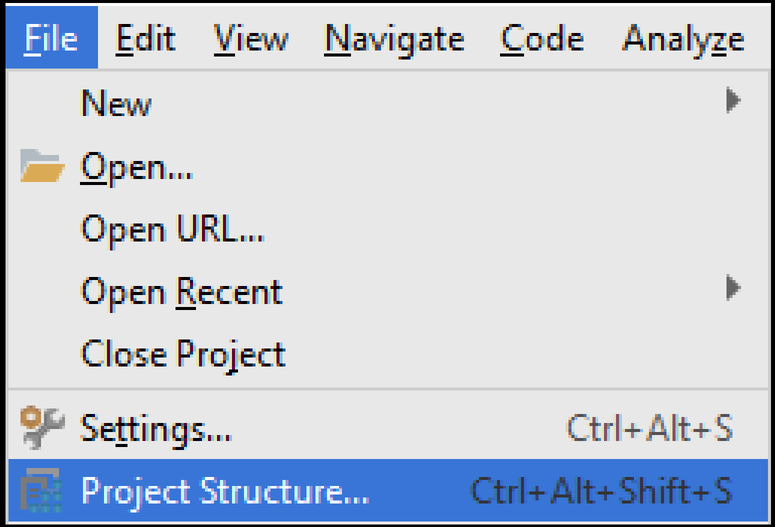
```
Connection c = DriverManager.getConnection(
"jdbc:mysql://localhost:3306/soft_uni", props);
```

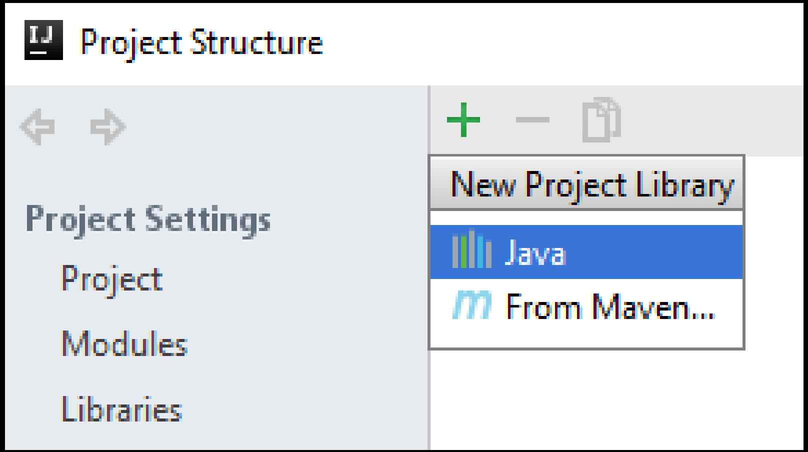
Database name

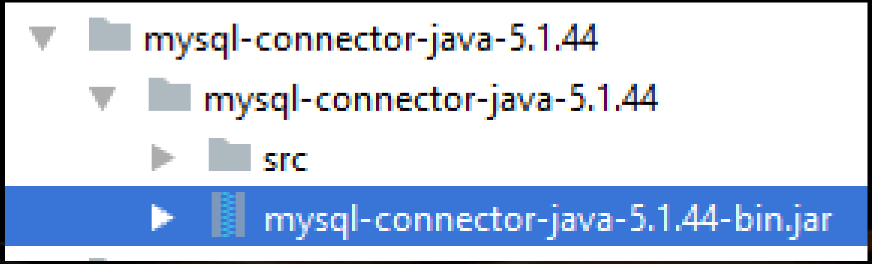
Credentials

Setting up the Driver in IntelliJ IDEA

- Add the driver as an external library:
  - "File" -> "Project Structure" -> "Libraries"






mysql-connector-java-5.1.44
mysql-connector-java-5.1.44
src
mysql-connector-java-5.1.44-bin.jar

JDBC Statements

Statement, PreparedStatement, CallableStatement

Statements

- The JDBC Statement interface defines the methods and properties that enable you to send SQL commands to the database

Interfaces	Recommended use
Statement	For general-purpose access to your database and static SQL statements at runtime. Cannot accept parameters.
PreparedStatement	For SQL statements used many times. Accepts parameters.
CallableStatement	Used for stored procedures. Accepts parameters.

Statements Example

- Example(PreparedStatement) from previous demo:

```
PreparedStatement stmt =
connection.prepareStatement("SELECT * FROM employees
WHERE salary > ?");

String salary = sc.nextLine();
stmt.setDouble(1, Double.parseDouble(salary));
```

SQL Query

Statements are created via the connection

Query parameter

Parameter Index

Parameter value

SQL Injection

How to prevent it?

What is SQL Injection?

- Placement of malicious code in SQL Statements
  - Usually done via user input
- To protect our data we can place parameters in our statements
  - We can do it by using PreparedStatement

SQL Injection Example: Login form input by user

- Ask the user to input username and password in fields
  - If we don't secure our statements, we risk SQL Queries to be written as an input
  - E.g. :
    - username: 'example_user'
    - password: '12345'
    - The following query will be built and executed to the data source:

```sql
SELECT id FROM users
WHERE username = 'example_user' AND password = '12345';
```

SQL Injection Example: Login form input by user (2)

- In result the id of the user will be returned
  - User will be authenticated to do actions in the application
- Without validating and securing our statements information might get exposed:
  - Value for password: ''1' OR username = 'admin';'
  - The following query will be executed:

```
SELECT id FROM users
WHERE username = 'pesho'
AND password = '1' OR username = 'admin';
```

SQL Injection Example: Login form input by user (3)

- In result the id an admin will be returned
  - Will permit actions to the user that can harm our application and database
- We can validate the input by setting rules
  - Length, special characters, digits etc.
  - Set up validation in our code in different layers (front-end, back-end etc.)

Advanced Concepts

Transactions and DAO Pattern

JDBC Transaction Pattern

- Every JDBC Connection is set to auto-commit by default
  - SQL statements are committed on completion
- In bigger applications we want greater control
  - If and when changes are applied to the database
- Turn off auto-commit:

```
connection.setAutoCommit(false);
```

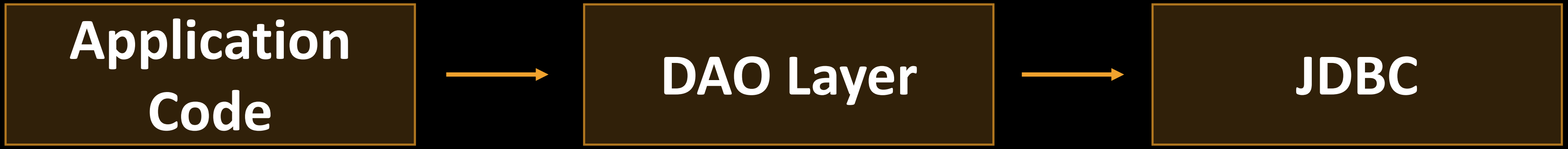
JDBC Transaction Pattern (2)

- Example (pseudo code):

```
try {
   connection.setAutoCommit(false);
   Statement stmt = conn.createStatement();
   String sql = "…";
   stmt.executeUpdate(sql);
   // If there is no error
   connection.commit();
} catch(SQLException se){
   // If there is any error
   conn.rollback();
}
```

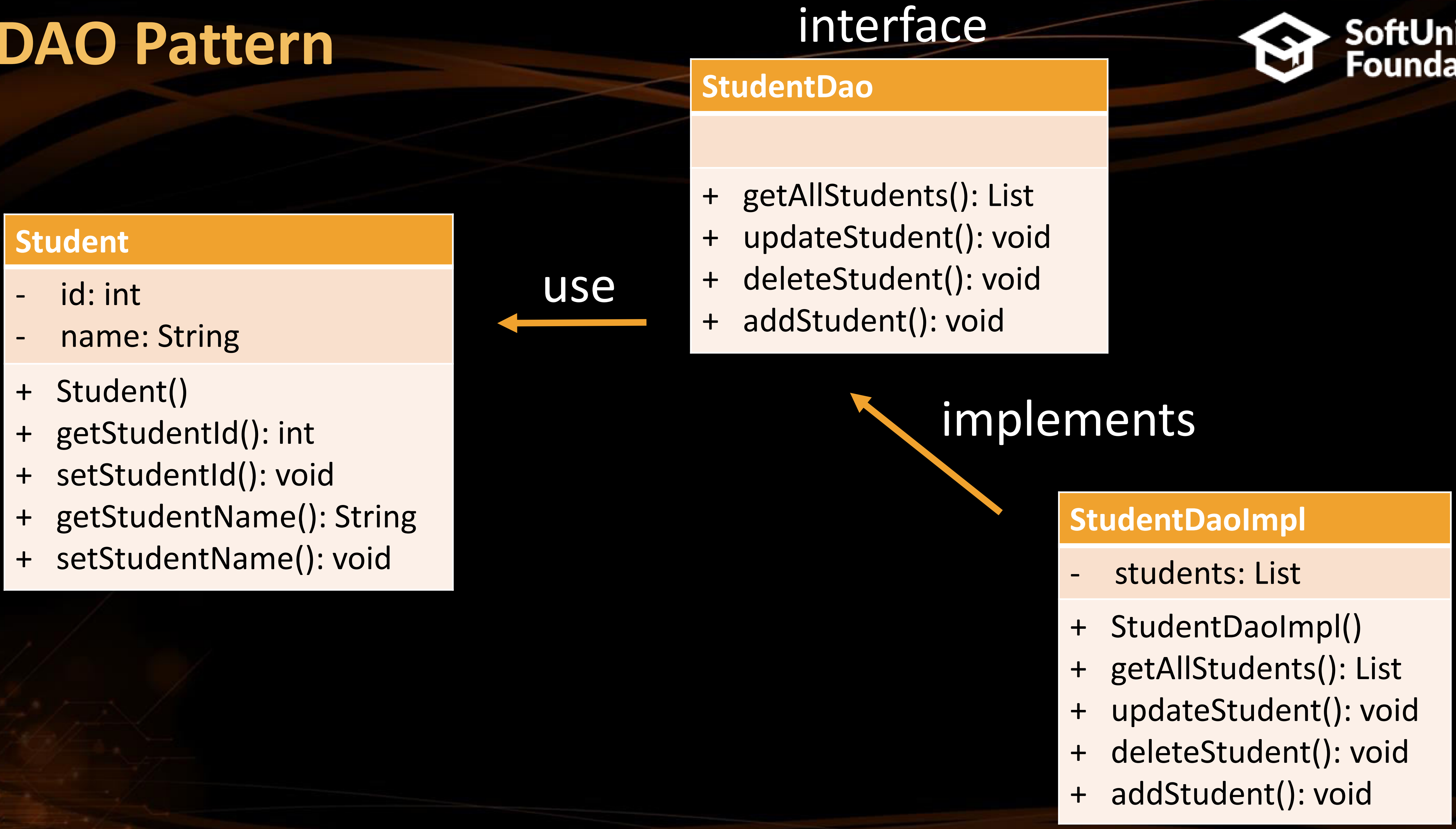
DAO Pattern

- Data Access Object design pattern is based on abstraction and encapsulation



- Why implement it:
  - Keeping data access code away from business logic
  - In result it can be changed without affecting other layers of the app
  - Improves testing with Mock objects

DAO Pattern



interface
StudentDao
+ getAllStudents(): List
+ updateStudent(): void
+ deleteStudent(): void
+ addStudent(): void
use
Student
- id: int
- name: String
+ Student()
+ getStudentId(): int
+ setStudentId(): void
+ getStudentName(): String
+ setStudentName(): void
implements
StudentDaoImpl
- students: List
+ StudentDaoImpl()
+ getAllStudents(): List
+ updateStudent(): void
+ deleteStudent(): void
+ addStudent(): void

Summary

- ORM Frameworks map Java objects to SQL entities
- JDBC provides us classes for operating with a database
- SQL Injection can seriously harm our data source or expose it
  - Our application should secure the statements being sent

Databases Advanced – DB Apps Intro

Questions?

https://softuni.bg/courses/databases-advanced-hibernate

License

Free Trainings @ Software University

- Software University Foundation – softuni.org
- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg