ORM Fundamentals

The ORM Concept, Config, CRUD Operations

Relational DB

Object

```
[Table(Name = "Customers")]
public class Customer
{
}
```
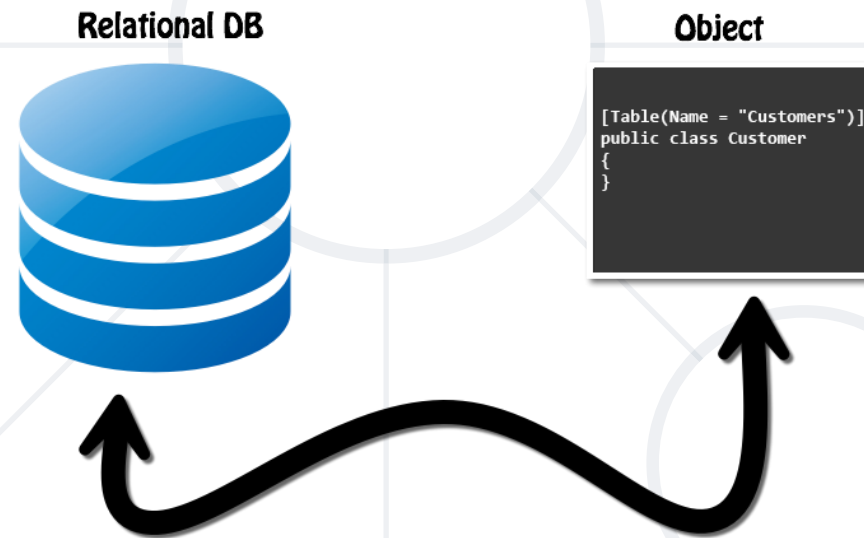
SoftUni

SoftUni Team

Technical Trainers

Software University

Software University

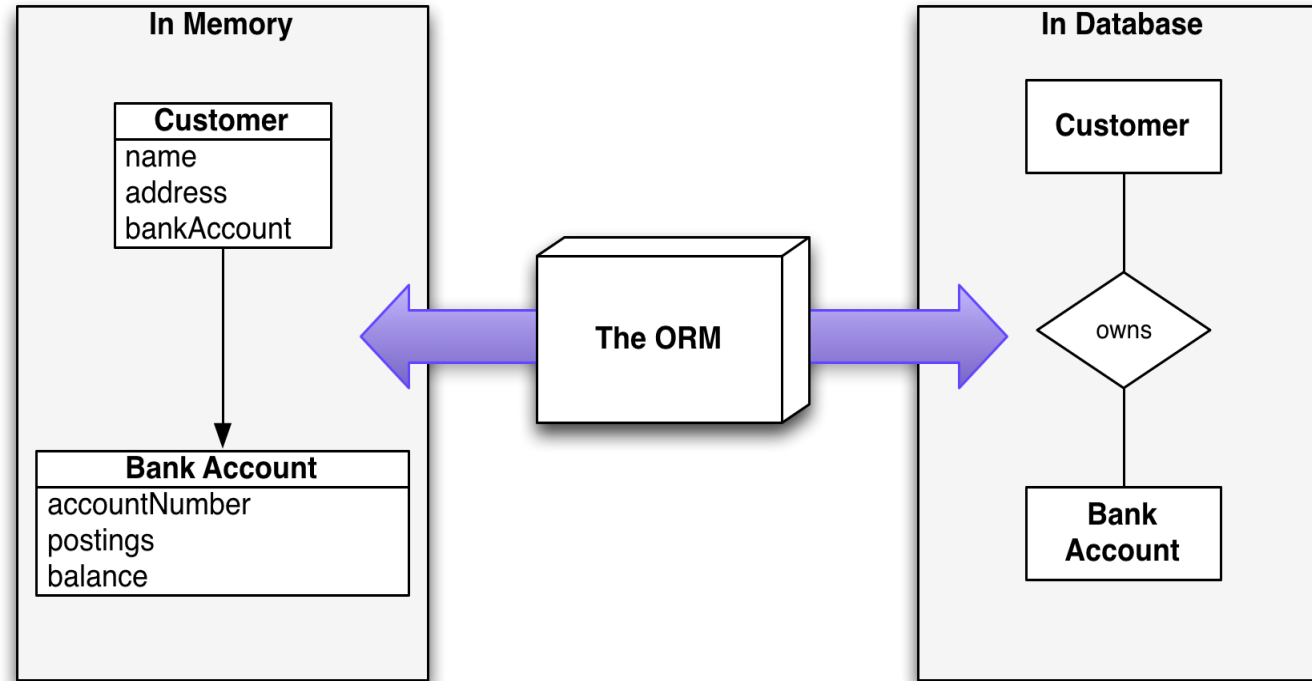# Table of Contents

1. Introduction to ORM
   - Architecture
   - Example
   - Approaches
2. ORM Advantages

sli.do

#java-db

ORM Introduction

Object-Relational Mapping

What is ORM?

- **Technique** for **converting data** between incompatible type systems using **object-oriented programming** languages

- **Object-Relational Mapping** (ORM) allows manipulating databases **using common classes and objects**

  - **Java/C#/etc. classes** ➔ **Database Tables**

  - **Database Tables** ➔ **Java/C#/etc. classes**

What is ORM? (2)

```java
public class Employee {
    public int id;
    public String firstName;
    public String middleName;
    public String lastName;
    public boolean isEmployed;
}
```

Why do we need ORM?

- In OOP, data-management tasks act on **objects** that are almost always **non-scalar** values

- Many **database** can only store and manipulate **scalar** values, organized within **tables**

- We must **manually** convert values into groups of simpler values to store in DB and convert them back when we retrieve data

JDBC and ORM

- The main difference, between JDBC and ORM, is **complexity**

- **JDBC/SQL**
  - If the application is simple as to present data directly from the database

- **ORM**
  - If the application is domain driven and the relations among objects is complex

Application Architecture

Presentation Layer → Service Layer → Persistence Layer / ORM → Database

ORM Frameworks: Features
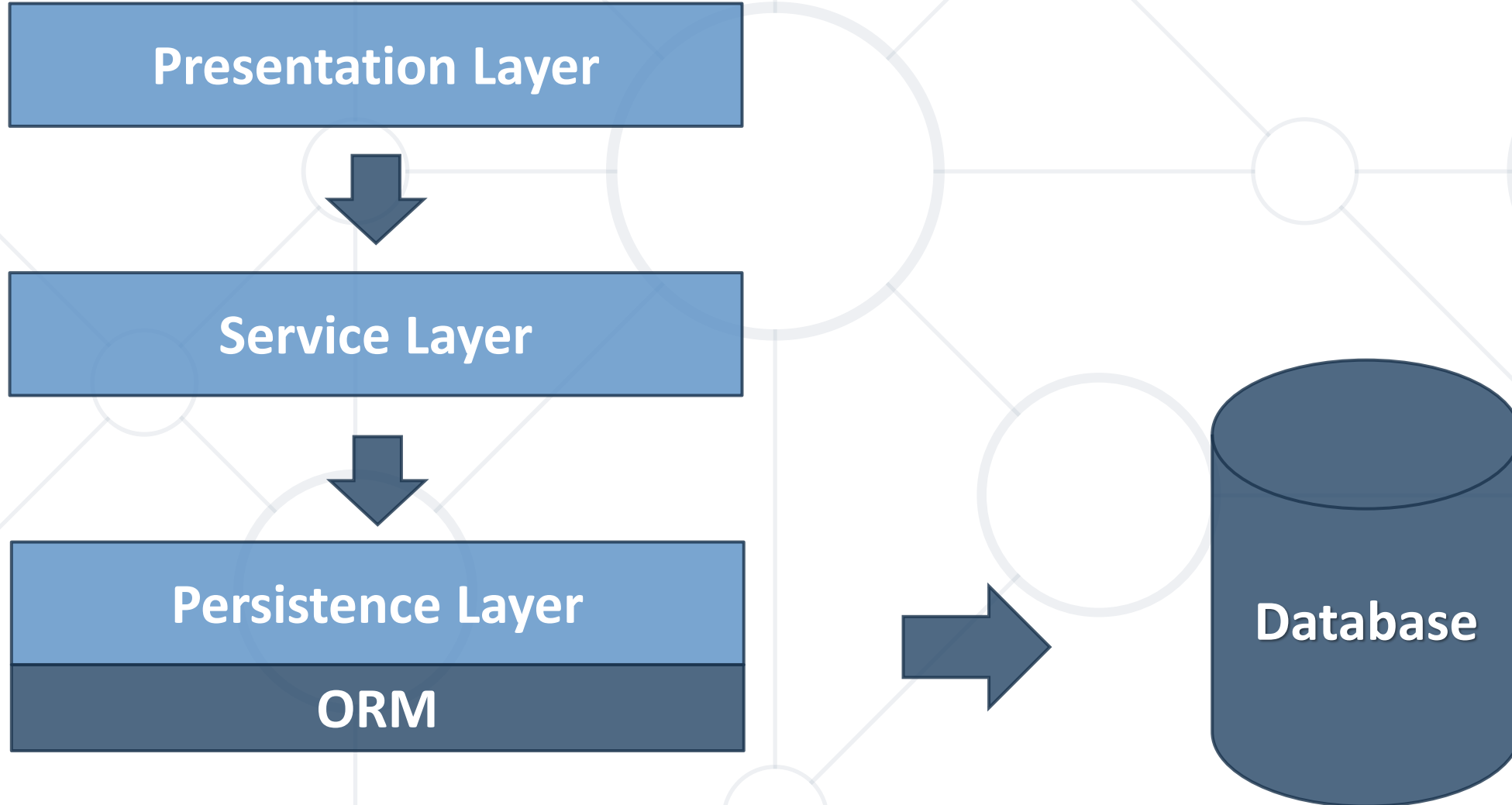
- **ORM frameworks** typically **provide** the following functionality:

  - **Automatically generate SQL** to perform
    data operations as:

    - persist, update, delete, merge, createQuery and so on.

  - **Object model from database schema** (DB First model)

  - **Database schema from object model** (Code First model)

Perform data operations with ORM (1)

- **Automatically generate SQL** to perform data operations

  - Save entity to DB

```
Student student = new
Student('George', 'Brown');
session.save(student);
```

➡

```
INSERT INTO students
(firstName, lastName)
VALUES
('George', 'Brown')
```

  - Retrieve data from DB

```
Student student = (Student)
session.get(Student.class, 1);
```

➡

```
SELECT * FROM students
WHERE id=1;
```

Perform data operations with ORM(2)

- We can use and specific ORM Query Language as **HQL** or **SQL**
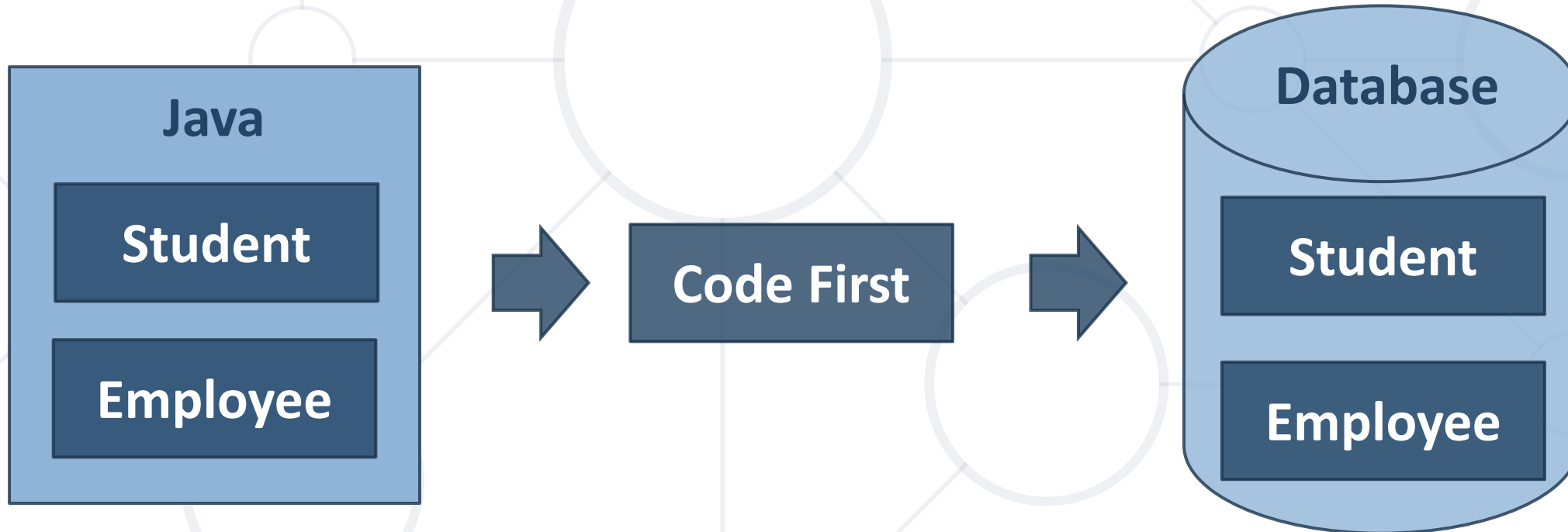
  - Using HQL

```
List<Student> studentList =
session.createQuery("FROM Student").toList();
```

  - Using SQL

```
String sql = "SELECT * FROM Employee";
SQLQuery query = session.createSQLQuery(sql);
query.addEntity(Employee.class);
List<Employee> results = query.list();
```

Code First Model

- **Models** the database after the entity classes

POJO + XML

- A bit old-fashioned, but very powerful

- Implemented in the "classical" ORM

```xml
...
    <description>Mapping file</description>
    <entity class="Employee">
        <table name="EMPLOYEETABLE"/>
        <attributes>
            <id name="id">
                <generated-value strategy="TABLE"/>
            </id>
            <basic name="name">
                <column name="EMP_NAME" length="100"/>
            </basic>
            <basic name="salary">
            </basic>
        </attributes>
    </entity>
...
```

POJO Mapped to DB Tables

- Based on Java annotations and XML

- Easier to implement and maintain

```java
@Entity
@Table(name = "employees")
public class Employee {
    @Id
    private int id;
    @Column(name = "name")
    private String name;
    @Column(name = "position")
    private String position;
}
```

ORM Advantages

And disadvantages

ORM Advantages (1)

- **Productivity**
  - Eliminates repetitive code
  - Generates database automatically
- **Maintainability**
  - Fewer lines of code
  - Easier to manage object model changes

ORM Advantages (2)

- **Performance**
  - Lazy loading
  - Caching
- **Database vendor independence**
  - The database is abstracted
  - Can be configured outside the application
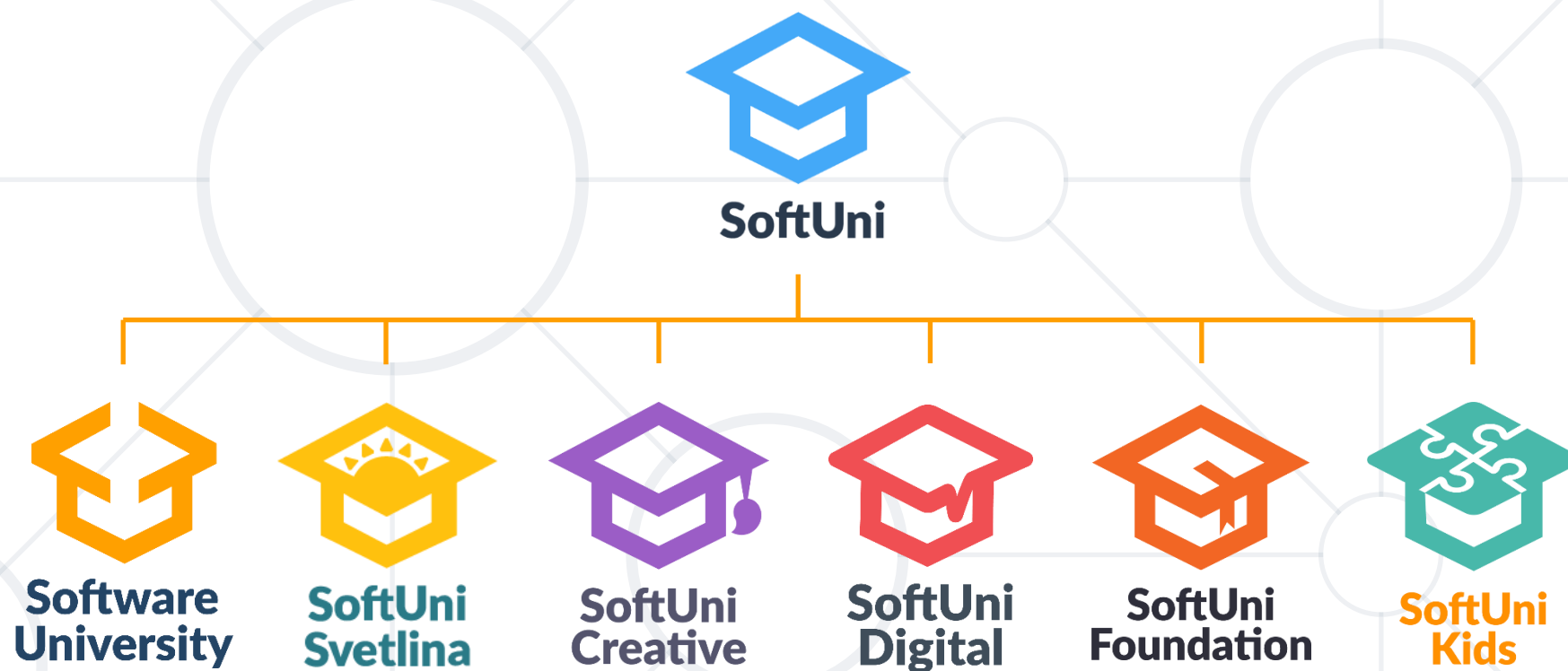
ORM Disadvantages

- **Reduced performance**
  - Due to overhead or auto generated SQL
- **Reduces flexibility**
  - Some operations are hard to implement
- **Lose understanding**
  - What the code is actually doing -
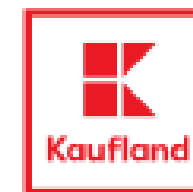   the developer is more in control using SQL

Summary

- **Object-Relational Mapping** (ORM) allows manipulating databases **using common classes and objects**
- The main difference, between JDBC and ORM, is **complexity**
- **POJO + XML** mapping
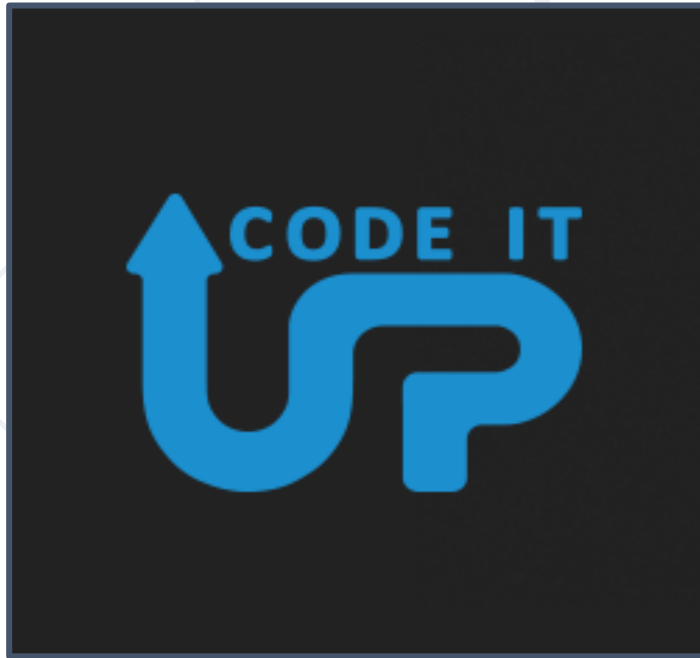- **POJO** mapped to **DB tables**

Questions?

SoftUni Diamond Partners

Educational Partners

Trainings @ Software University (SoftUni)

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

24

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**

- Unauthorized copy, reproduction or use is illegal

- © SoftUni – https://about.softuni.bg

- © Software University – https://softuni.bg