Object-based Programming

Week 12 – JDBC

Java Database Connectivity (JDBC)

- An Java API which defines how a client may access a database

- Needs JDBC driver for specific DBMS

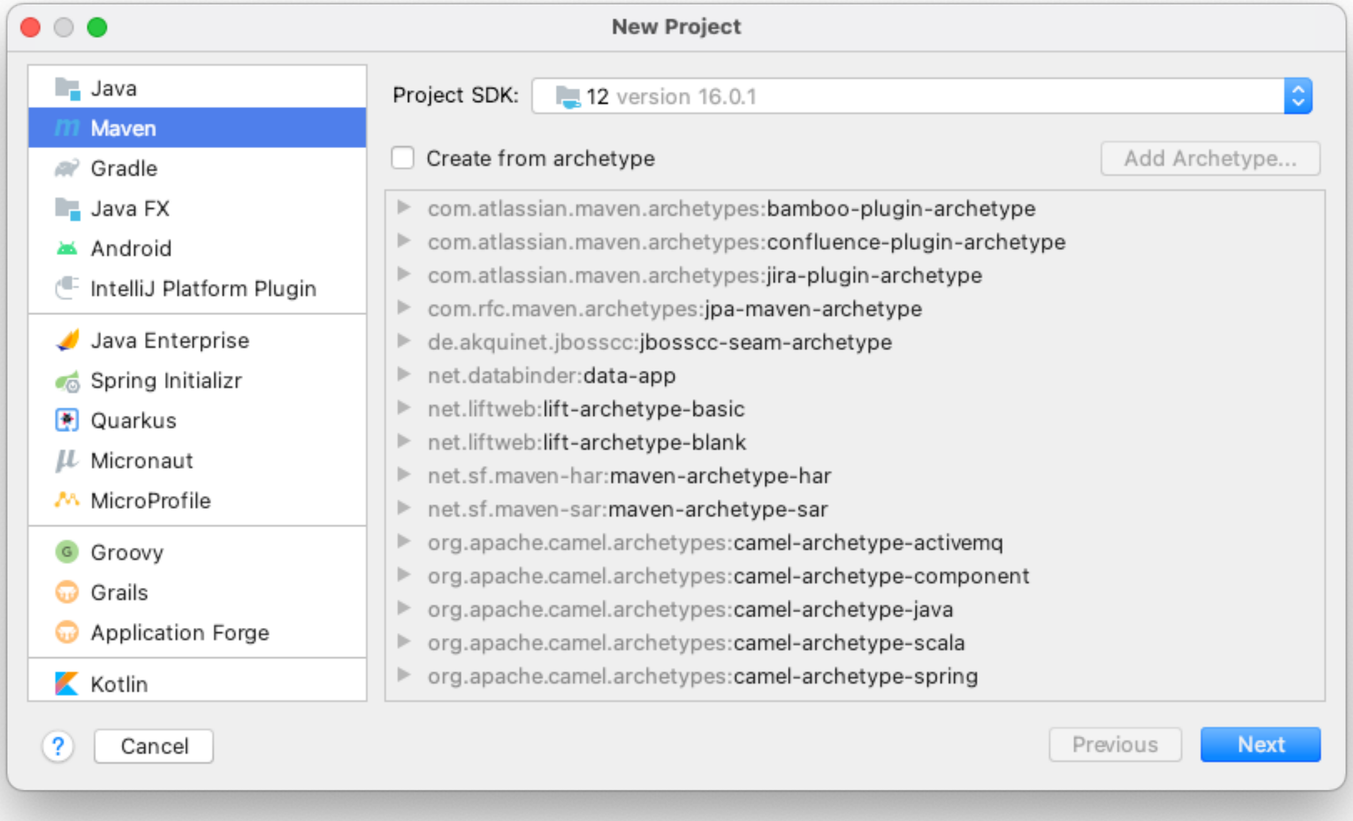
JDBC Drivers

- MySQL
  - https://mvnrepository.com/artifact/mysql/mysql-connector-java

- Oracle DB
  - https://www.oracle.com/database/technologies/maven-central-guide.html

Install via Maven (IntelliJ)

Select Maven when creating a new project



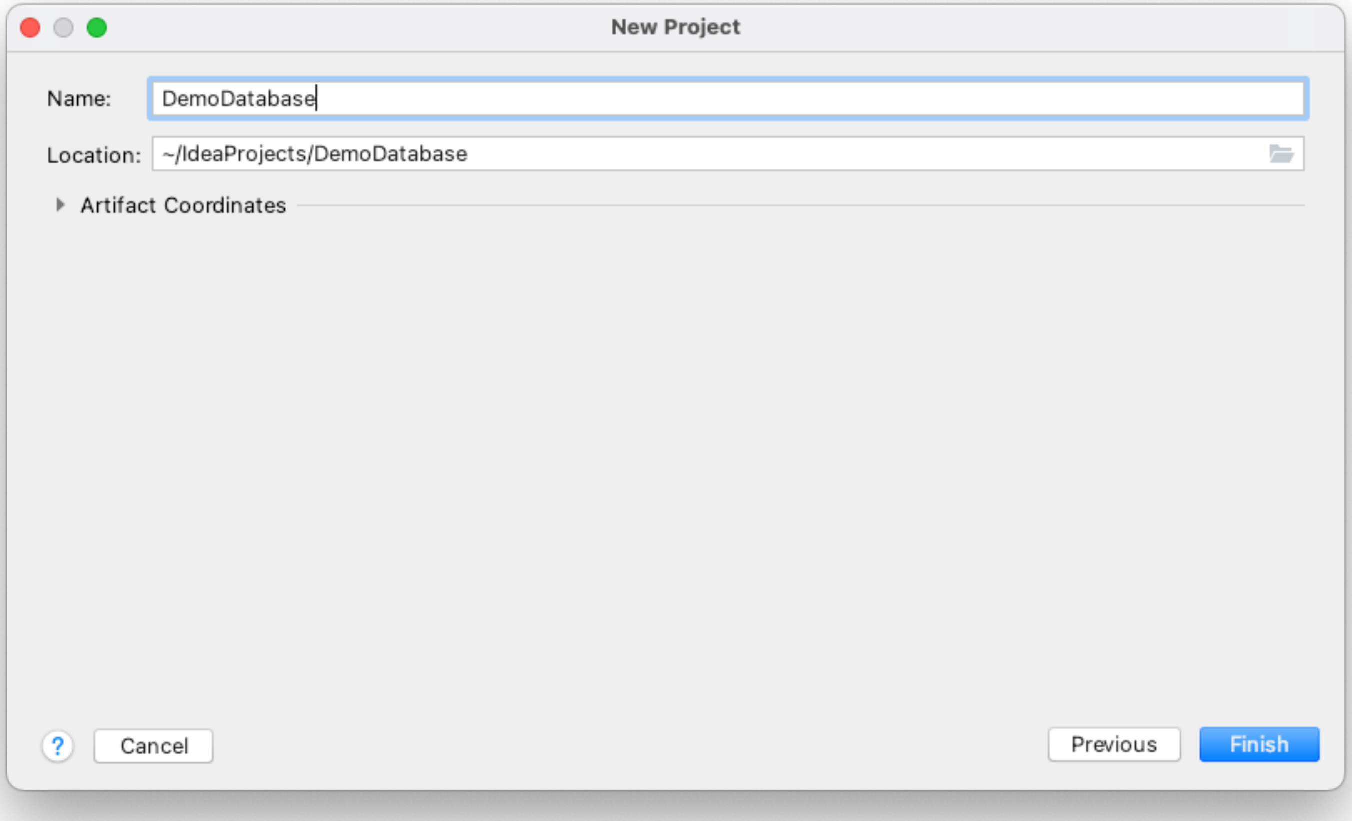
Project SDK: 12 version 16.0.1

□ Create from archetype

```
▶ com.atlassian.maven.archetypes:bamboo-plugin-archetype
▶ com.atlassian.maven.archetypes:confluence-plugin-archetype
▶ com.atlassian.maven.archetypes:jira-plugin-archetype
▶ com.rfc.maven.archetypes:jpa-maven-archetype
▶ de.akquinet.jbosscc:jbosscc-seam-archetype
▶ net.databinder:data-app
▶ net.liftweb:lift-archetype-basic
▶ net.liftweb:lift-archetype-blank
▶ net.sf.maven-har:maven-archetype-har
▶ net.sf.maven-sar:maven-archetype-sar
▶ org.apache.camel.archetypes:camel-archetype-activemq
▶ org.apache.camel.archetypes:camel-archetype-component
▶ org.apache.camel.archetypes:camel-archetype-java
▶ org.apache.camel.archetypes:camel-archetype-scala
▶ org.apache.camel.archetypes:camel-archetype-spring
```

Install via Maven (IntelliJ)

Enter the project name as usual



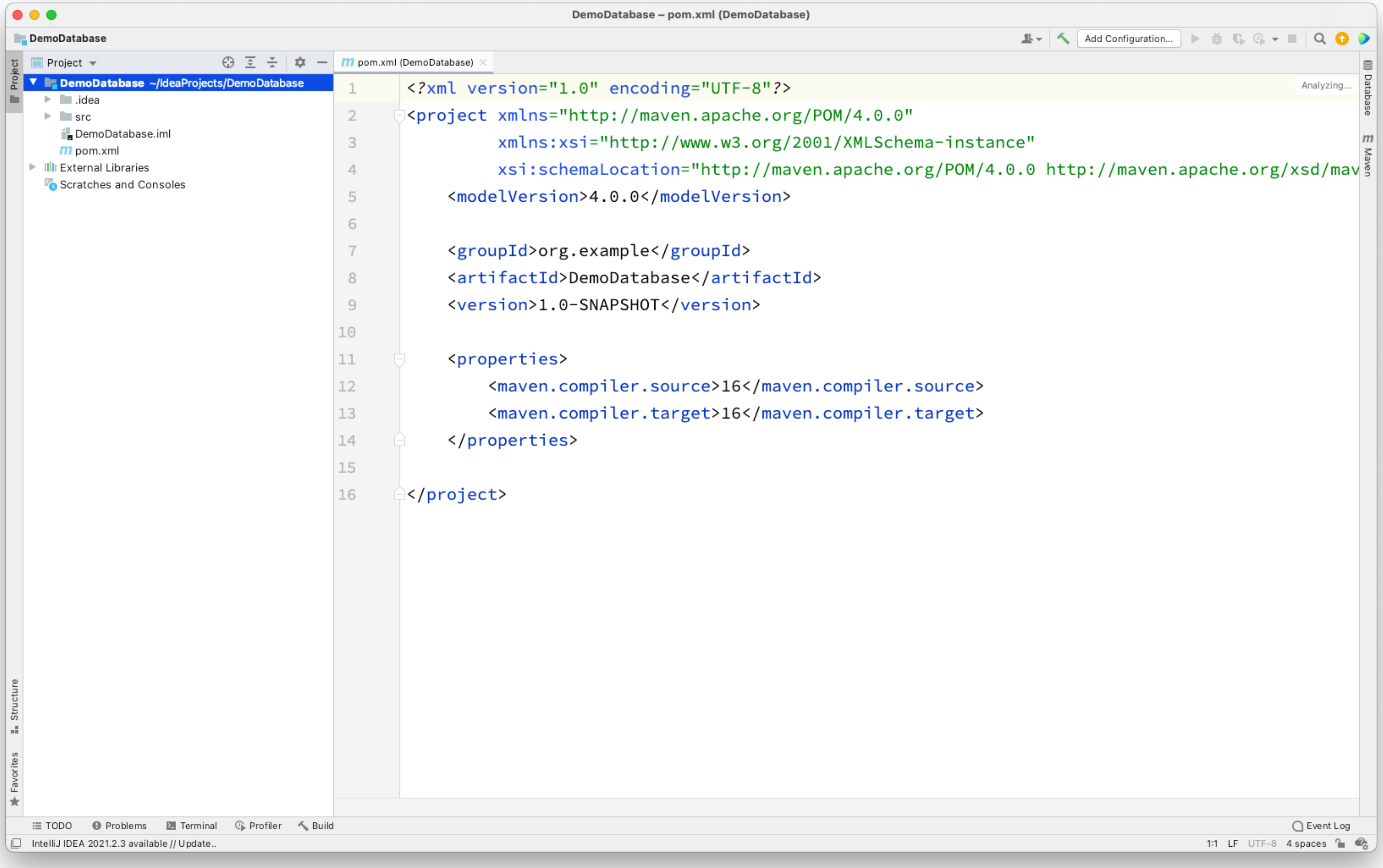
New Project

Name: DemoDatabase

Location: ~/IdeaProjects/DemoDatabase

Install via Maven (IntelliJ)

Open `pom.xml` from the project



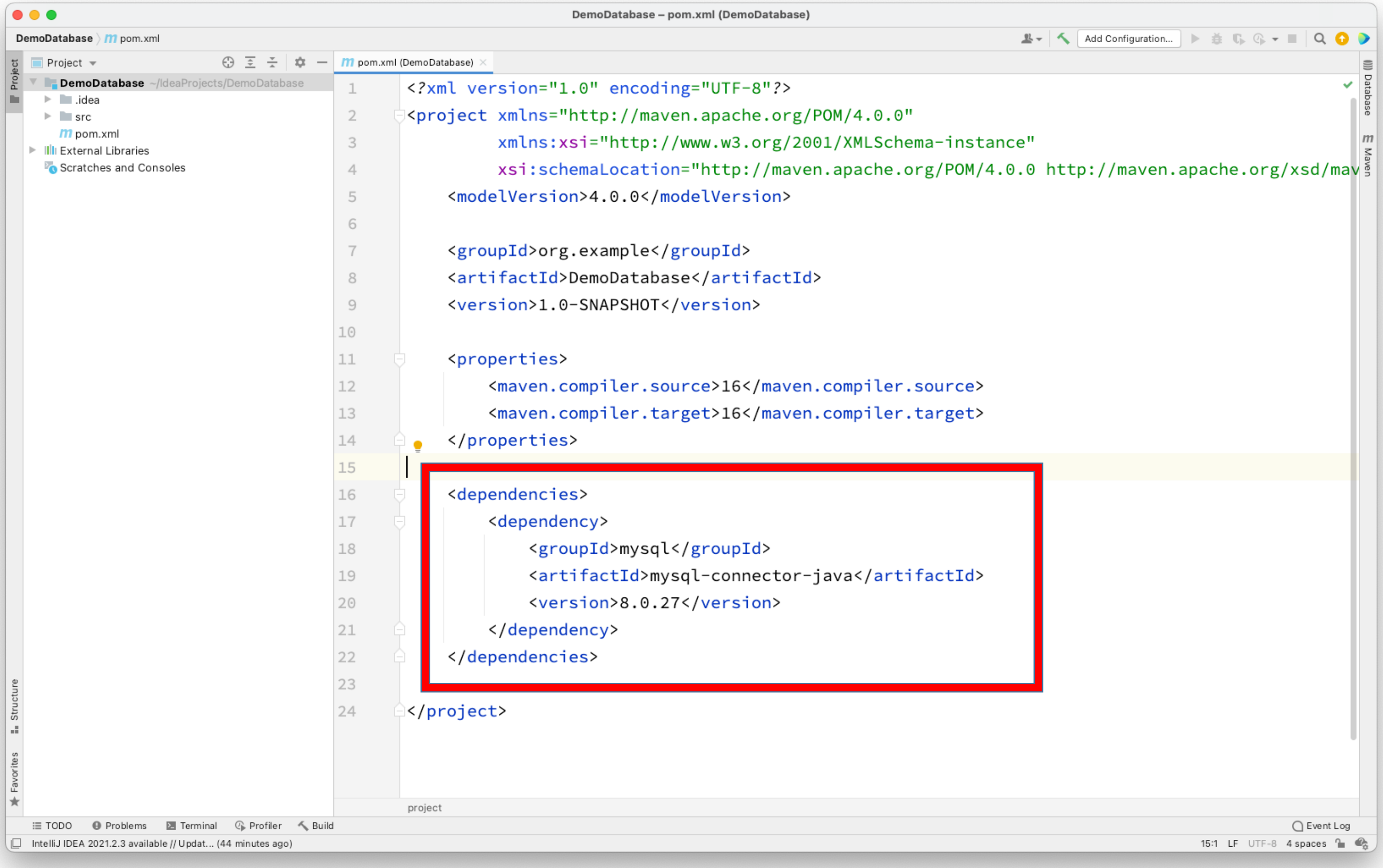
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/mav
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>DemoDatabase</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>16</maven.compiler.source>
        <maven.compiler.target>16</maven.compiler.target>
    </properties>

</project>
```

Install via Maven (IntelliJ)

Insert the following dependencies (e.g. MySQL)

If not auto reload, then:

- right-click
- select "*Maven*"
- select "*Reload Project*"



```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/mav
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>DemoDatabase</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>16</maven.compiler.source>
        <maven.compiler.target>16</maven.compiler.target>
    </properties>

    <dependencies>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.27</version>
        </dependency>
    </dependencies>

</project>
```

UNTAR
Universitas Tarumanagara

UNTAR untuk INDONESIA

Connecting to the database

```
import java.sql.*;
...
try {
   // Load JDBC driver for MySQL database
   Class.forName(”com.mysql.jdbc.Driver”);

   // Establish connection
   Connection conn = DriverManager.getConnection(
      connString,
      dbUser,
      dbPass);
} catch (ClassNotFoundException | SQLException e) {
   e.printStackTrace();
}
```

Connecting to the database

```
String connString = “jdbc:mysql://localhost:3306/dbName”;

String dbUser = “user”;
String dbPass = “pass”;

// Establish connection
Connection conn = DriverManager.getConnection(
   connString,
   dbUser,
   dbPass);
```

Before the source code example…

- Create database "CorporateXYZ" in your DBMS
- Create table "employee" with the following columns:
  - id, int, primary key, auto_increment
  - fullname, varchar(50), not null
  - salary, int, not null

```
import java.sql.*;

public class Employee {
    private Connection conn;

    public Employee() throws Exception {
        Class.forName(“com.mysql.jdbc.Driver”);
        this.conn = DriverManager.getConnection(
            “jdbc:mysql://localhost:3306/CorporateXYZ”, “jansonh”, “jansonh”);
    }

    public boolean isConnected() { return (this.conn != null); }
    public void closeConnection() { this.conn.close(); }
}
```

```java
public class Main {
    public static void main(String[] args) {
        try {
            Employee e = new Employee();

            // Check if database is connected
            System.out.println(e.isConnected());
        }
        catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

SELECT query

```
public void selectEmployee() throws SQLException {
    String query = “SELECT * FROM employee”;

    Statement stmt = this.conn.createStatement();
    ResultSet rs = stmt.executeQuery(query);

    while (rs.next()) {
        System.out.println(“Full name: “ + rs.getString(“fullname”));
        System.out.println(“Salary: “ + rs.getInt(“salary”));
    }
}
```

SELECT query

```
public void selectEmployee(int id) throws SQLException {
   String query = “SELECT * FROM employee WHERE id = ?”;

   PreparedStatement stmt = this.conn.prepareStatement(query);
   stmt.setInt(1, id);

   ResultSet rs = stmt.executeQuery();

   while (rs.next()) {
      System.out.println(“Full name: “ + rs.getString(“fullname”));
      System.out.println(“Salary: “ + rs.getInt(“salary”));
   }
}
```

INSERT query

```
public void insertEmployee(String fullname, int salary) throws
SQLException {

   String query = “INSERT INTO employee(fullname, salary)
                   VALUES (?, ?)”;

   PreparedStatement stmt = this.conn.prepareStatement(query);

   stmt.setString(1, fullname);
   stmt.setInt(2, salary);

   stmt.execute();
}
```

Exercise! UPDATE query

```
public void updateEmployee(String id, String fullname, int salary)
throws SQLException {
    ...
}
```

Exercise! DELETE query

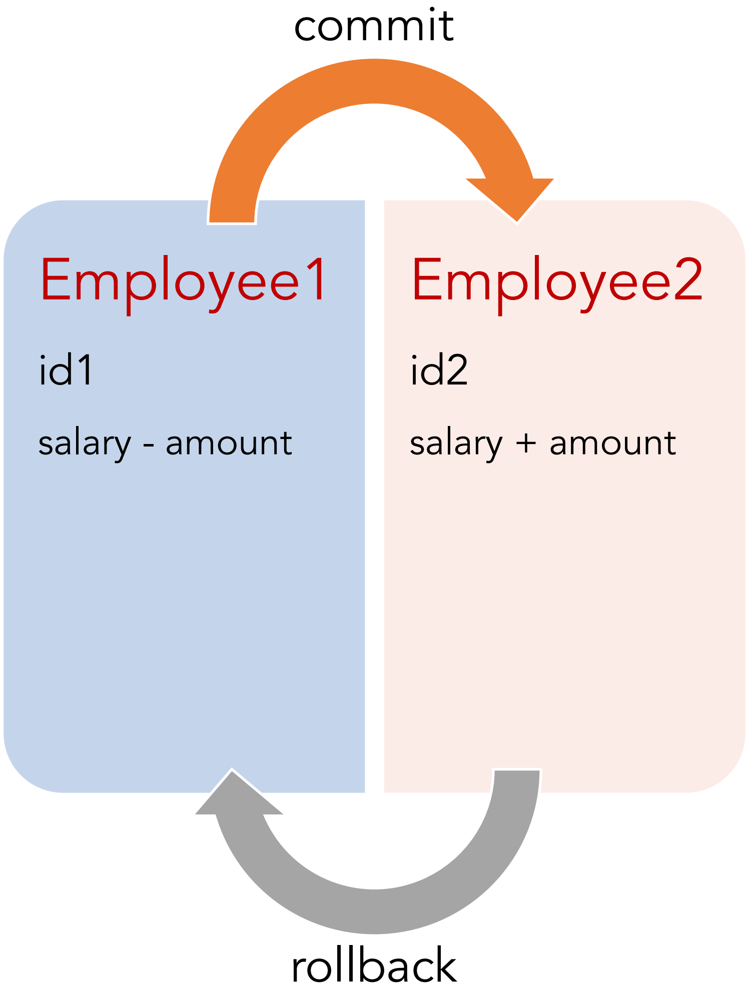
```
public void deleteEmployee(String id) throws SQLException {
   ...
}
```

SQL Transaction

```
public void transfer(int id1, int id2, int amount) throws SQLException {
   String query = “UPDATE employee SET salary = salary + ?
                   WHERE id = ?”;

   PreparedStatement stmt1 = null;
   PreparedStatement stmt2 = null;

   try {
      this.conn.setAutoCommit(false);

      stmt1 = this.conn.prepareStatement(query);
      stmt2 = this.conn.prepareStatement(query);
   ...
```

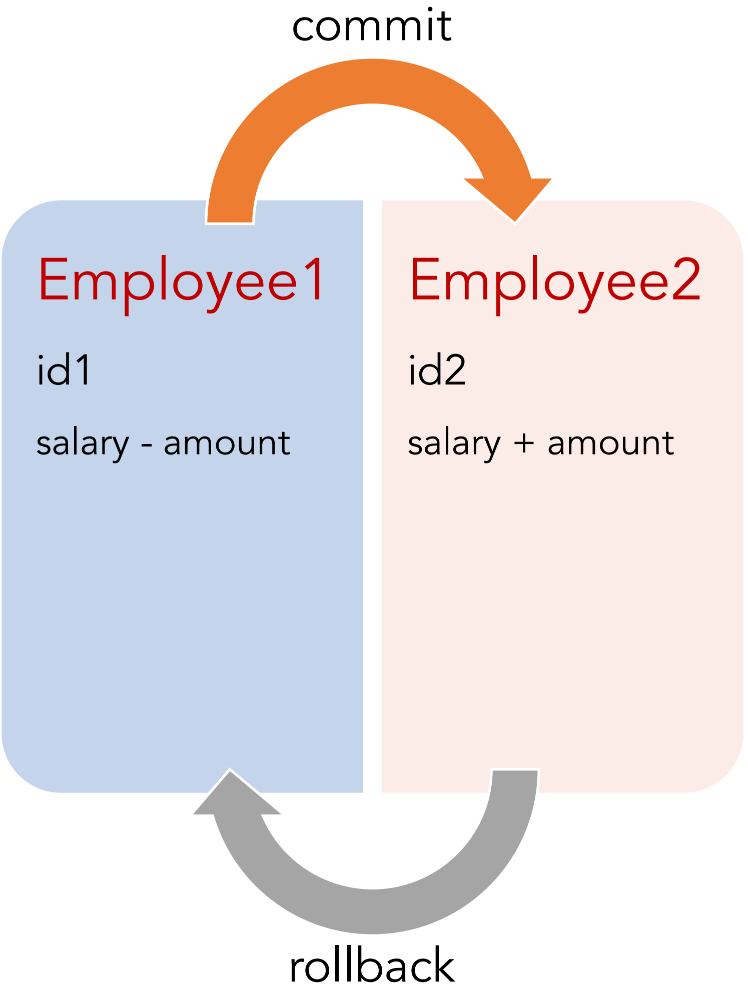
SQL Transaction

```
...
      stmt1.setInt(1, -amount);
      stmt1.setInt(2, id1);
      stmt1.executeUpdate();

      stmt2.setInt(1, amount);
      stmt2.setInt(2, id2);
      stmt2.executeUpdate();

      this.conn.commit();
...
```



SQL Transaction

```
...
   } catch (SQLException e) {
      try {
         this.conn.rollback();
      } catch (SQLException e) {
         e.printStackTrace();
      }
   } finally {
      this.conn.setAutoCommit(true);
   }
}
```


commit
Employee1
id1
salary - amount
Employee2
id2
salary + amount
rollback