Databases SS22

Lab 4 preparation

Group 2

Alisa Khrustaleva

Celestine Machuca

Submitted: 27.06.2022

Task 1

Task 1.1

Get the Java project provided in EMIL running for you (e.g., using an IDE like Eclipse). Try out the program by running it and describe what it can be used for.

The program was initialized using IntelliJ IDE. In the program a GUI displaying students was hardcoded, so no proper connection to database was implemented. The program set a good base for developing a project involving a connection with the student database.

Task 1.2

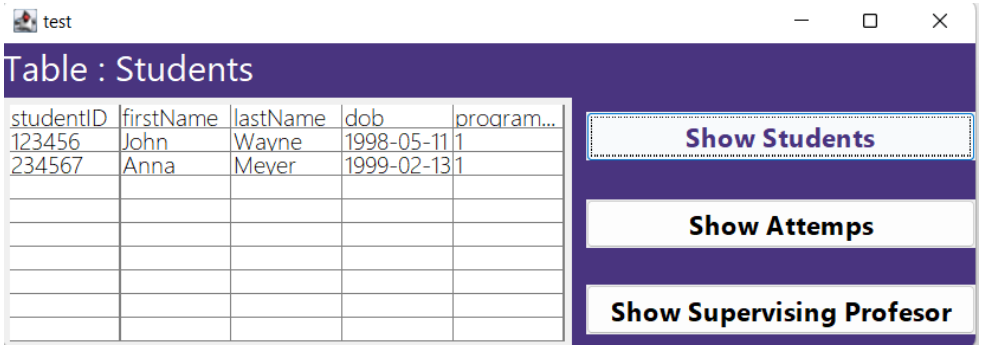
Take a look at the JDBC documentation and figure out, how you connect the prototype to your SQL database (either your local MySQL or the Oracle database at HAW).

The database was successfully connected with the local MySQL server by adding the JDBC file to the IntelliJ project and entering the correct credentials to the code.

Task 1.3

Now you need to implement the first Query to the Database. Take a look at the getAllStudents Method and fill the variable result with a SQL query.

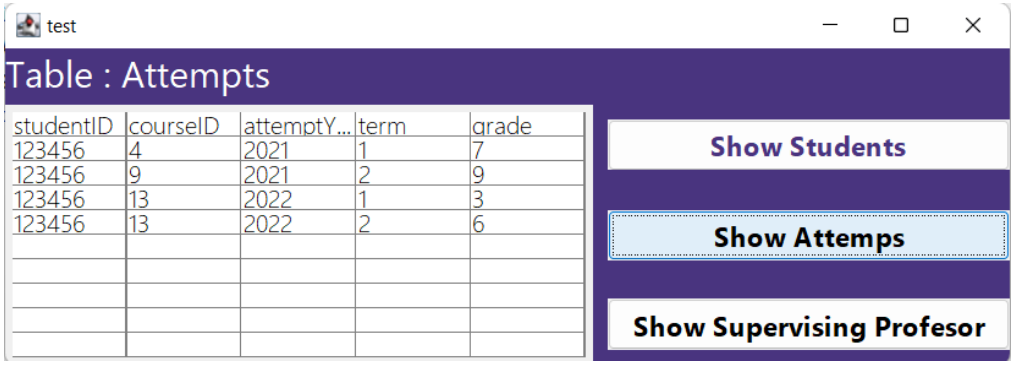
```
public ArrayList<Student> getStudents() {
    ArrayList<Student> students = new ArrayList<>();
    String query = "select * from Student;";
    try {
        Statement statement = connection.createStatement();
        ResultSet resultSet = statement.executeQuery(query);
        while(resultSet.next()){
            String studentID = resultSet.getString( columnLabel: "studentID");
            String firstName = resultSet.getString( columnLabel: "firstName");
            String lastName = resultSet.getString( columnLabel: "lastName");
            String dob = resultSet.getString( columnLabel: "dob");
            String programID = resultSet.getString( columnLabel: "programID");
            students.add(new Student(studentID, firstName, lastName, dob, programID));
        }
    }  catch (SQLException e) {
        e.printStackTrace();
    }
    return students;
}
```



Task 1.4

Implement the second Query to the Database. Implement the Method getAttemptsForStudents accordingly. Query the database for all attempts of the handed over parameter Student and put them into result.

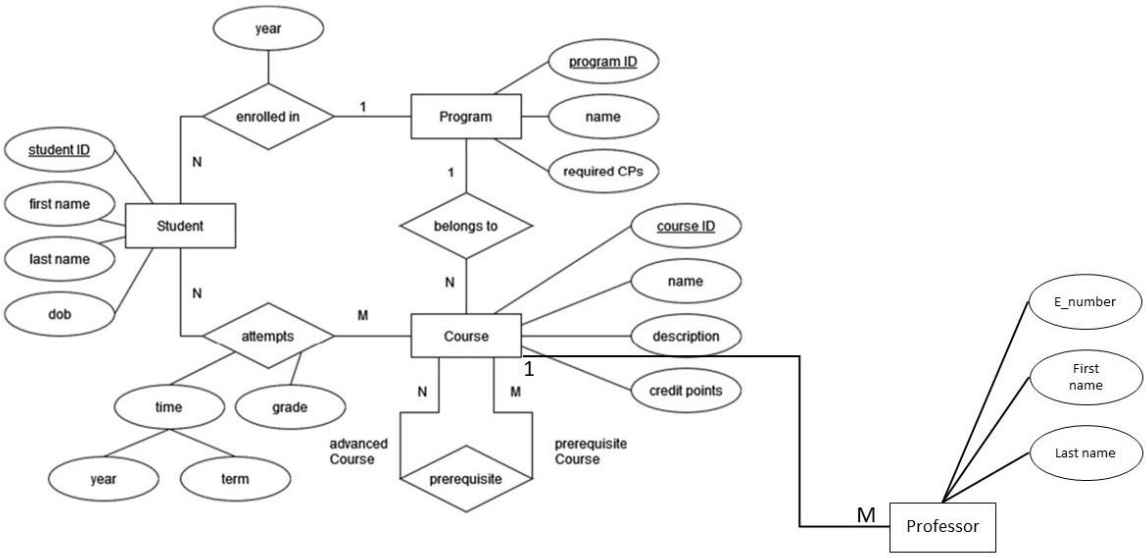
```
public ArrayList<Attempt> getAttemptsForStudent(Student student){
    ArrayList<Attempt> results = new ArrayList<>();
    String query = "select * from ATTEMPT where studentID = " + String.valueOf(student.studentID) + ";";
    try {
        Statement statement = connection.createStatement();
        ResultSet resultSet = statement.executeQuery(query);
        while (resultSet.next()){
            String studentID = resultSet.getString( columnLabel: "studentID");
            String courseID = resultSet.getString( columnLabel: "courseID");
            String term = resultSet.getString( columnLabel: "term");
            String attemptYear = resultSet.getString( columnLabel: "attemptYear");
            String grade = resultSet.getString( columnLabel: "grade");
            results.add(new Attempt(studentID, courseID, term, attemptYear, grade));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return results;
}
```



Task 2

Task 2.1

Alter the Student Information System ER-Diagram accordingly.



Task 2.2

Create a relational schema from the altered ER-Diagram.

PROFESSOR	
PK, FK	courseID INT
PK	e_number INT NOT NULL
	firstName VARCHAR(20) NOT NULL
	lastName VARCHAR(20) NOT NULL

STUDENT	
PK	studentID INT NOT NULL
	firstName VARCHAR(15) NOT NULL
	lastName VARCHAR(20) NOT NULL
	dob DATE
FK	programID INT

PREREQUISITE	
PK, FK	advancedCourseID INT NOT NULL
PK, FK	prerequisiteCourseID INT NOT NULL

ATTEMPT	
PK, FK	courseID INT NOT NULL
FK	studentID CHAR(6) NOT NULL
	attemptYear CHAR(4) NOT NULL
	term INT NOT NULL
	grade INT NOT NULL

COURSE	
PK, FK	courseID INT NOT NULL
	courseName VARCHAR(3) NOT NULL
	courseDescription VARCHAR(30)
	creditPoints INT NOT NULL
FK	programID INT

PROGRAM	
PK	programID INT NOT NULL
	programName VARCHAR(30) NOT NULL
	requiredCPs INT NOT NULL

Task 2.3

Write SQL statements that creates/alters the tables of the SQL database for the Student Information System to take account for the newly introduced Professors.

```sql
create table if not exists PROFESSOR (
    e_number INT NOT NULL,
    firstName VARCHAR(20) NOT NULL,
    lastName VARCHAR(20) NOT NULL,
    courseID INT,
    PRIMARY KEY (e_number)
    );
-- connects profesors table to course table
alter table PROFESSOR
    add constraint FOREIGN KEY (courseID) references COURSE(courseID);


-- create fake data for the table profesors
-- available course id are 4, 9, 13, 15
insert into PROFESSOR values (1, 'William', 'Shakespeare', 4);
insert into PROFESSOR values (2, 'Abraham', 'Lincoln', 9);
insert into PROFESSOR values (3, 'Winston', 'Churchill', 13);
insert into PROFESSOR values (4, 'John', 'Keats', 15);
```

Task 2.4

Update the Java program to be capable of displaying the supervisor in the attempts window. You need to change up multiple classes for that. Have a look at classes that have the name "Attempt" in them.

```java
public Professor getProfessorFromAttempt(Attempt attempt){
    Professor professor = null;
    String query = "select * from professor where courseID = " + String.valueOf(attempt.courseID) + ";";
    try {
        Statement statement = connection.createStatement();
        ResultSet resultSet = statement.executeQuery(query);
        while (resultSet.next()){
            String e_number = resultSet.getString( columnLabel: "e_number");
            String firstName = resultSet.getString( columnLabel: "firstName");
            String lastName = resultSet.getString( columnLabel: "lastName");
            String courseID = resultSet.getString( columnLabel: "courseID");
            professor =  new Professor(e_number, firstName, lastName, courseID);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return professor;
}
```