

Quiz Manager - Console

Object Oriented programming

Selasi Gborglah & Vijay Yeruva | Java Fundamentals | 19 Feb 2019

Master of Sciences Data Science Analytics-EPITA

TABLE OF CONTENTS

0 Preface 2

1 Introduction 3

1.1 Purpose

1.2 Scope

1.3 Definitions, Acronyms and Abbreviations

1.4 References

1.5 Overview

2 System Overview 5

2.1 System Characteristics

2.2 System Architecture

2.3 Infrastructure Services

3 System Context 6

4 System Design 7

4.1 Design Method and Standards

4.2 Documentation Standards

4.3 Naming convention

4.4 Programming Standards

4.5 Software development tools

5 SQl Samples 8

0 Preface

0.1 Purpose of this document

1. This document is a generic Technical Design Document made by Vijay & Selasi, MS EPITA Students. It provides guidance and template material which is intended to assist for prospects, in producing a project‑specific Technical Design Document. It is also useful background reading for anyone involved in developing or monitoring the Quiz (MS-DSA Fundamental).

0.2 Use of this document

1. This Preface is addressed to the users of this generic document and is not meant to be retained in any project‑specific Technical Design Document documents based on it.
2. The template should be used pragmatically, that is - where a section is not relevant it should be omitted. Conversely, the material contained in this document is not necessarily exhaustive

# Introduction

The Project is made on regards to the directions provided with a minimal prototype design to ensure the functionalities

The Quiz app has only two users; the admin and the student. When a user launches the application, they must input their name and password.

When the details are entered, the student would be directed to a page where they can attempt questions. However, the admin is directed to a page where he can either create, read, update and delete contents of a table. So, the admin can perform these functionalities on the Question and MCQ Choice table. If the admin decided to Create a table, they would have to input the table they want to edit,

## 

## Purpose

The quiz app is written in Java and the IDE used was eclipse. We employed the use of the H2 database. In the database.java file we created all our queries to link the program to the database.

## Scope

Scope of the project defines the working set of the prototype build over for the following parameters

identify the difficulty level of the questions and classify according to the level.

validation of the takers results.

Organising the quizzes with in intervals.

The system handles all the operations, and generates reports

as soon as the test is finish, that includes name, mark, time

spent to solve the exam.

Allow students to see or display his answers after the exam is finish.

The type of questions is only multiple choice right.

Manage the students, the admin/professors can add/edit/delete

student and review their scores

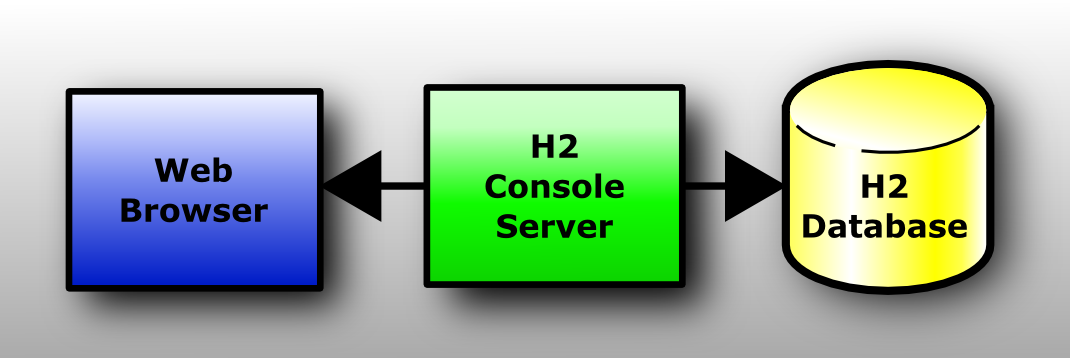
## References

- <https://github.com/thomasbroussard/2018s2-fund-group3.git>

- <https://github.com/Selasi-Afi/JavaProject>

- <http://www.h2database.com/html/main.html>

- <https://www.eclipse.org/>



OverView

To handle most of the possible cases, there are several types of question to consider.

* MCQ Questions
* Open Questions
* Associative Questions

Software Requirements

BACKEND/WEBSERVER FRONTEND DATABASE

JAVA, H2, ECLIPSE IDE

## System Overview

## Characteristics

* The usual problem while preparing an evaluation, are to:
* • Constitute an appropriate evaluation regarding of the required level
* • Reuse former questions
* • Organize sample evaluations
* • Correct automatically the MCQ questions.
* The implementation of this project has been done on Console

Types of question

The MCQ questions

The MCQ questions are composed of a question text and a set of possible choices, each choice can be right or wrong.  
It can also be interesting to add extra content, like some code extract, some picture or some other kind of media (video, music etc.).

The Open Questions

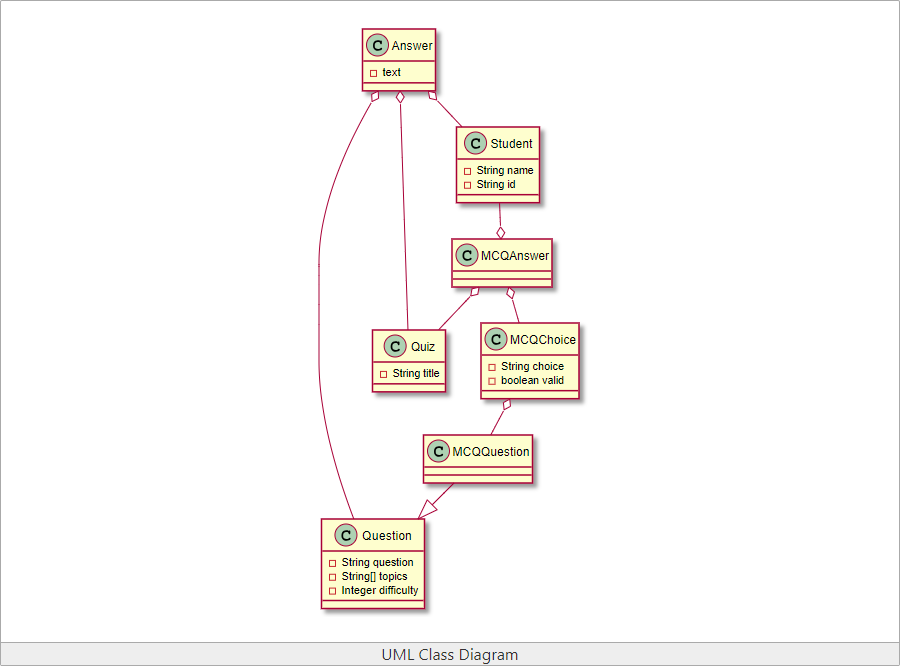
The open questions are composed only by a question, and some hints, additionally they can be completed by extra media content.

#### **Associative questions**

The associative questions are questions where it necessary to assign some propositions to some descriptions, like in the following.

#### **Common questions attributes**

Each question has a some extra attributes to describe the **topic** (tag) and the **difficulty** of the question. Those two fields help to balance the overall exam complexity, and the topics coverage.  
Those attributes can be taken in account for automatic exam assembly.

**

Testing

To satisfy the requirements and check whether every build service in the application are executed as expected, one of the earliest, testing efforts performed on the code is Unit Testing. To quickly test any new code or changes to existing code without the overhead and additional time involved in tasks such as server configuration, services setup and application deployment, we integrated the popular testing framework JUnit Framework to execute the unit test on every small code module.

Our objectives are in writing and executing the tests

- We want to code and run the tests without leaving the IDE (Eclipse).

- There should be no special deployment of the code required

- We should be able to exploit other code analysis tools such as Metrics and find bugs right from within

System Hierarchey

This section gives a functional requirement that applicable to the Quiz Application. There are two sub modules in this phase.

• Admin module.

• User module.

The functionality of each module is as follows:

SYSTEM DESIGN  
User module: The candidate will logon to the application and take his quiz. He can check the list of question type, name and then attend the quiz as per his preference. The candidate will get result immediately after the completion of the examination.

Admin module: The professor/admin will logon to the application and take his/her quiz. They can manage the student identities, create quiz – MCQ type, ASSOCIATIVE type, OPEN type, and view the marks of every student.

Major Features

Login/Sign Up Authentication   
Manage Identities   
Manage Questions  
Add Questions  
Update Questions  
Delete Questions  
Search Questions  
View Questions

Conception

Authentication

### The web application cannot move forward without the login. There are two logins for student and admin. And a register page for the student, in-order to login.

SQL Schema

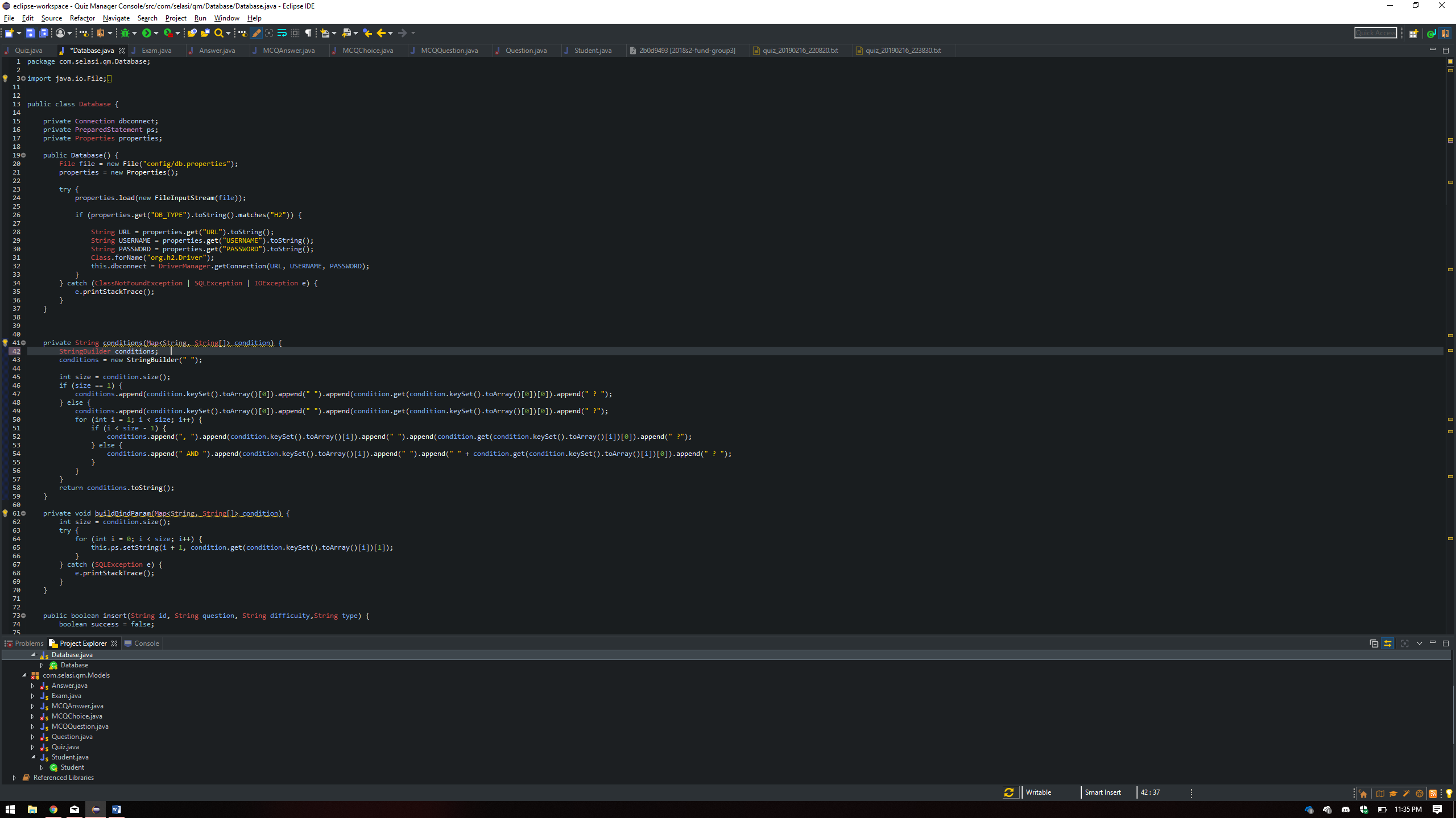
**USERS** Attributes

PK User \_ID Int  
Mail Varchar  
Password Varchar  
Username Varchar

**Questions Attributes**

PK ID Int  
Correctanswer Varchar  
Answer 1 Varchar  
Answer 2 Varchar  
Answer 3 Varchar  
Answer 4 Varchar  
Question Type Varchar  
QuizName Varchar

### 



SQL Structures

The quiz project has the table schemas run the quiz and get the answers from the user.  
All the answers should be recorded and at the end of the quiz the global result is displayed to the user.

SQL STATEMENTS IMPLEMENTED TO CREATE TABLES

CREATE TABLE QUESTIONS (  
    ID int NOT NULL,  
    QUESTIONS varchar(255) NOT NULL,  
    DIFFICULTY int,  
    TYPE varchar(255),   
    CHOICE1 varchar(255) NOT NULL,  
    CHOICE2 Varchar(255) NOT NULL,  
    CHOICE3 varchar(255) NOT NULL,   
    CHOICE4 Varchar(255) NOT NULL,  
   PRIMARY KEY (ID)  
);

CREATE TABLE ANSWER (  
    QID int NOT NULL,  
    TEXT varchar(255) NOT NULL,  
    PRIMARY KEY (QID), FOREIGN KEY (ID) REFERENCES Questions(ID)  
);

CREATE TABLE LOGIN (  
    NAME varchar(255),  
    PASSWORD Varchar(255),  
    TYPE Varchar(255),  
    PRIMARY KEY (NAME)  
);