**Instructions For Compilation:**

1- Go to the “Application.java” file (Server).

2- Run its main method.

3- The server will be waiting for clients to connect.

4- Run “Client.java”, it will run two clients just to show threading.

5- If you logged in as a teacher, you would navigate to the teacher menu.

5- If you logged in as a student, you would navigate to the student menu.

6- The Teacher menu has options that will direct you to the Quiz menu.

**List Of Students Who Submitted:**

We tried to send an invite in Vocareum to Troy and Anushka; however, they do not receive our invites for some reason. So, we are separated.

First group - (Zuhair and Anish) Second group - (Anushka and Troy)

Zuhair submitted on Vocareum. Anushka submitted on Vocareum.

Zuhair submitted the presentation.

Zuhair submitted the report.

**Description of classes:**

**1-** **Application.java:**

**Functionality:** Displays a menu for sign-in and log-in. Teachers and students can sign-in and register. It will direct them to their menus depending on who logged in. The class saves the log-in information in a notepad. Also, it reads all information needed from the database when you run it. It is also the server that clients connect to. It offers multiple sockets for different clients to enforce threading.

**Testing:** We ran the compiler and confirmed that it accepts clients and create a thread for each one of them; it assigns students and teachers and directs both to their respective menus.

**Relationship:** When it directs to other menus, it connects to other Teachers and Students classes.

**2-** **Course.java:**

**Functionality:** Saves course information

**Testing:** When we created a course, a teacher was assigned with the specified course and number of student enrollment.

**Relationship:** It is used by all Courses classes.

**3-** **CourseArchive.java:**

**Functionality:** Collects all the courses in an ArrayList and modifies that list.

**Testing:** It returns all the courses created when we ran the compiler. Teachers are able to get access to the courses when created.

**Relationship:** It is used by teachers and other Course classes.

**4-** **InvalidCourseException.java:**

**Functionality:** Throws an exception whenever a course can’t possibly be created.

**Testing:** It throws an exception when appropriate and indicates a descriptive error.

**Relationship:** Used by all Course classes.

**5-** **InvalidQuizException.java:**

**Functionality:** Throws an exception whenever a quiz can’t possibly be created.

**Testing:** It throws an exception when appropriate and indicates a descriptive error.

**Relationship:** Used by all Quiz classes.

**6-** **PrintInformation.java:**

**Functionality:** Used to write and read all quiz information from/to text files. Also, it reads quizzes that are imported externally and add them to the quizzes arraylist.

**Testing:** Ran the compiler and created quizzes, and they are written in a text file. Deleted the existing quizzes and then read the quizzes inside the text file to extract their information. And they were saved in the QuizArchive.java correctly. Also, it adds imported quizzes to the program and writes its information in a txt file to save it.

**Relationship:** It is used by TheQuizFunction.java. Every time TheQuizFunction runs, it reads the quizzes from the text files and when it finishes modifying the quizzes, it writes them again. Also, Application uses it to read quizzes from the database.

**7-** **Quiz.java:**

**Functionality:** It creates a quiz object and saves its questions, studentAnswers, the correctAnswers, quiz name, calculates the score, randomizes the questions and options for the quiz, modifies the questions and options, and generates a string the contains all quiz information that are ready to be printed to a file.

**Testing:** Using the compiler, it adds questions, assigns a name for the quiz, modified the name, questions, and options upon request, randomizes the quiz questions and options upon teacher request, and the quiz information is printed appropriately.

**Relationship:** It is used by all Quiz classes.

**8-** **QuizArchive.java:**

**Functionality:** Stores all quizzes in one ArrayList.

**Testing:** It retrieves the needed quizzes upon request.

**Relationship**: Used by all Quiz classes.

**9-** **TheQuizFunction.java:**

**Functionality:** Displays a menu that enables Teachers to create a quiz, modify a quiz, randomize a quiz, view student submissions, list available quizzes, delete a quiz, and import a quiz externally.

**Testing:** Tested the menu in the compiler, it creates a quiz and store it in QuizArchive, modify a quiz from the Quiz class, randomized all questions and options appropriately, displayed student submissions, listed quizzes, deleted quizzes, and imported quizzes.

**Relationship:** It uses Quiz.java, QuizArchive.java, PrintInformation.java, InvalidQuizException.java. Teachers, TheCourseFunction.java, and the main Application uses this class to navigate to its menu.

**10- Client.java:**

**Functionality:** Connects to our server (Application.java). It creates two clients to connect simultaneously to enforce threading.

**Testing:** Both clients connected to the server successfully. Therefore, if another machine that wants to connect to the server, it will work.

**Relationship:** Not used by any class. Connects to the server.

**11-** **Student.java:**

**Functionality:** A user is able to create, modify or delete a student account and all the information of all registered accounts is saved in a txt file (allows user to log back in if program is closed but account is already registered)

**Testing:** Created a main method with just this implementation of code where we tried logging in, creating student accounts and then modified and deleted them while checking the txt files to make sure they were updated

**Relationship:** Displayed and used in application.java

**12-** **StudentAnish.java:**

**Functionality:** Displays a menu that only students see. It allows a student to access a specific course and take quizzes in specific courses. After they take quizzes, they can view the grade they got for the quiz. It includes the “startQuiz” method, “getScore” method, and “getModifiedScore” method. It writes and reads student submissions. Also, it can attach answers from a file externally.

**Testing:** It created a menu for students only. Students took quizzes, got their timestamps, attached their answers, and viewed their grades. All information needed were saved in the StudentQuizzes.txt file.

**Relationship:** Used when someone logs in as a student and when the student wants to take a quiz, access course, or view a quiz grade. It is interconnected with the Teacher.java, TheQuizFunction.java, and Quiz.java, CourseArchive.java, and especially Student.java.

**13-** **Teacher.java:**

**Functionality:** A user is able to create, modify or delete a teacher account and all the information of all registered accounts is saved in a txt file (allows user to log back in if program is closed but account is already registered)

**Testing:** Created a main method with just this implementation of code where we tried logging in, creating teacher accounts and then modified and deleted them while checking the txt files to make sure they were updated

**Relationship:** Displayed and used in application.java

**14-** **TheCourseFunction.java:**

**Functionality:** Displays a menu for courses, which enable you to: create a course, navigate TheQuizFunction.java menu, add students to a course, modifies course information like its name, enrollment, or assigned teachers to the course, and deletes a course.

**Testing:** The compiler used the menu, and each option works as specified. It created a course, modified a course, added students to a course, assigned teachers to their courses, deleted courses, and navigated TheQuizFunction.java menu successfully.

**Relationship:** It uses Course.java, CourseArchive.java, InvalidCourseException.java, and Teacher and Student classes. Application.java uses this class to navigate to its menu.

**15-** **TimeStamp.java:**

**Functionality:** Creates a timestamp when the students finished taking a quiz.

**Testing:** Everytime a student completed a quiz, it provides a timestamp for it.

**Relationship:** It is used in the startAQuiz method.

**16-** **TestCases.java:**

**Functionality:** Simulates the test cases used for project 4.

**Testing:** It prints true for all passed test cases.

**Relationship:** It uses the whole program to test every method in it.