**README**

**Instructions:**

For Project 5 you will have to first compile all the classes in the repository. The repository will contain a Server file and a Client file. The Server file will need to be run first. The Client file contains a main method which has a socket set to localhost by default, to run the client on a different PC, the host will have to be changed to the IP address of the server PC.

For the file imports you will have to adhere to the following formats:

Quiz file imported by teacher:

Format:

/\*\*

Quiz title

Question 1

Option 1

Option 2

Option 3

Option 4

…

\*/

Example:

/\*\*

Quiz03

What year is it?

2019

2020

2021

2022

How is your semester?

Good

Fine

Not good

Bad

/\*

Response file imported by student:

Format:

/\*\*

Response for Question 1

Response for Question 2

Response for Question 3

\*/

Example:

/\*\*

A

b

c

\*/

Correct answers file imported by teacher:

Format:

/\*\*

Answer for Question 1

Answer for Question 2

Answer for Question 3

\*/

Example:

/\*\*

B

A

C

\*/

Submission record:

- Adree Das - submitted report on Brightspace

- “Jerry Mann”- submitted Vocareum workspace

**Classes**:

**PERSON**

**Fields:**

Name Description

username The person’s username

password The person’s password

teacher Boolean denoting whether they are a teacher or not

**Methods:**

Name Description

getters Returns the value of a field

setters Sets the value of a field

toString Returns the string output of the object

AddToFile Adds a person object’s credentials to the

Accounts.txt file

The Person class lays down the framework for creating a teacher and a student. Whenever a user logs in, a

person is created with their specific credentials and authority.

**STUDENT EXTENDS PERSON**

**Fields:**

Name - Description

username - String containing the student’s username

password - String containing the student’s password

teacher - Boolean set to false

Methods:

Name – Description

Constructors – Creates a Student object

-The Student class is one of the two types of users which will use our program. The student has access to

all the available quizzes in a course. They can choose to take a quiz from the list of available quizzes and

then submit it. Each submission gets recorded in the QuizSubmissions.txt file. The student also has the

option to attach a file with their responses instead of typing them in. Once a student has submitted their

quiz and a teacher has graded it, the student can view the graded submission with points assigned to each

question and a final point total.

**TEACHER EXTENDS PERSON**

**Fields:**

Name - Description

username - String containing the teacher’s username

password - String containing the teacher’s password

teacher - Boolean set to true

**Methods:**

Name - Description

Constructors – Creates a Teacher object

-The Teacher class is the second of the two types of users who will access the program. Once logged in as

a teacher, one can choose to make a quiz. This can be done manually or by importing a file with the title

and questions in correct format, after which each quiz is added to the Quiz.txt file. The teacher can add

any number of questions to a quiz and any number of quizzes to a course. The teacher can also choose to

randomize the order of questions and their choices in a quiz. They can also choose to edit or delete any

created quiz. The teacher can view submissions made by students to a quiz and then grade them. The

graded submissions are then stored in the GradedQuizzes.txt file.

**COURSE**

**Fields:**

Name - Description

name - String containing name of the course

quizNames - String arraylist containing names of all the quizzes in the course

**Methods:**

Name - Description

getters -Returns the value of a field

addQuiz - Adds a quiz title to the course

deleteQuiz - Deletes a quiz title from the course

toString - Returns the string output of the object

-The Course class is used as a means to track the courses to which each individual quiz is associated. The

course object stores only the name of the course and the names of the quizzes it has. Each course can have

as many quizzes as necessary.

**QUESTION**

**Fields:**

Name - Description

question - String containing the question

answers - String array containing 4 options

**Methods:**

Name - Description

getters - Returns the value of a field

setters - Sets the value of a field

toString - Returns the string output of the object

-The Question class is used to create a question which is the building block of a quiz. It consists of the

actual question itself along with its 4 options. Questions can be added to quizzes and be edited by a

teacher. They are displayed to a student when they take a test.

**QUIZ**

**Fields:**

Name - Description

quizTitle - String containing title of the quiz

timeLimit - Int containing time limit for the quiz in minutes

question - Question arraylist containing all the questions

for a quiz

**Methods:**

Name Description

getters - Returns the value of a field

writeQuiz - Writes the quiz title, time limit and questions to

the Quiz.txt file

-The Quiz class is used to create a quiz object which is the center of our program. A teacher can create,

edit and delete a quiz. The quiz can have as many questions as the teacher likes. All quizzes are stored in

the Quiz.txt file. Students can view and take a quiz.

**SERVER**

**Methods**

Name - Description

main - Creates a server socket and a socket to connect to the client, and then creates a new thread for handling the client

run - Contains functioning of the quiz program

accountList - Sorts Accounts.txt into an arraylist of persons

writeAccountList - Writes the toString for persons from a person arraylist into Accounts.txt

readRandomizationFile - Writes the Randomization.txt file into a String arraylist

randomizeQuiz -Returns a randomized list of integers from 1 to the number of questions

randomizeChoices -Shuffles the choices in a quiz’s questions

writeSubmission -Writes the quiz to the QuizSubmissions.txt file

decodeResponses -Decodes multiple choice responses to match the unscrambled text

makeQuizArray(String filename) - Reads the Quiz.txt file and converts the text

into quiz objects and returns a quiz arraylist

makeQuizArray(String filename, Course course) - Reads the Quiz.txt file and converts the text

into quiz objects, runs them thorugh a filter for the specified course and returns a quiz arraylist

makeCourseArray - Returns an arraylist with the current courses

writeCourse - -Writes all the courses to the Course.txt file

Server - Creates the server socket object and a socket object which connects to a client. It creates a new ClientHandler object which runs the Client thread. The server receives information from the Client and does the required processing, updating the files etc. and then sends the output to the Client. The run method includes all the code for the quiz program. On start, a login window pops up. If the user doesn’t have an account, they can make one or edit an existing account. The user can also choose to delete their existing account. The user must specify themselves as a student or a teacher while creating the account.

When a teacher logs in, they are presented with the teacher window. The teacher must first select or create a course before accessing any of the teacher options. Once selected, the teacher can create, edit, randomize, or delete quizzes and view and grade submissions through the buttons displayed on the left of the window.

When a student logs in, they are presented with the student window. The student must also first select a course. They can then choose to take a quiz or view their graded submissions by the buttons on the left side of the Student window.

**CLIENT EXTENDS JCOMPONENT IMPLEMENTS RUNNABLE**

**Methods**:

run - contains GUI and client-side functioning of the Login window

run2 -contains GUI and client-side functioning of the student and teacher windows

main -creates socket object which connects to the server and then runs the GUI

-The client class contains everything that the user interacts with while using the program. It creates a socket object connecting to the server and then runs the GUI. The run method contains the GUI which handles input and output for the Login window. The run2 methods contains the GUI which handles input and output for the Teacher and Student windows. The GUIs include JFrames with panels, buttons, fields, comboBoxes etc. The run methods implement action listeners for all the buttons which send and receive information from the Server and display it to the user.