README

Who Submitted What:

- Report: Luke Cooley
- Vocareum Workspace: Logan Snelling
- Presentation: Anushka Gupta

Instructions on running our project (pre-loaded user details below instructions):

- Run the Server.java file
- Run the ClientSession.java file
- GUI application is launched with option “Log In” or “Create account” buttons
    - Select the “Log in!” Button
        - Enter the “Test Teacher” for teacher role or “Test Student” under Full Name section
        - Enter “teach1” or “stud1” based on roles. (Ex: expected “Name”, received “name”)
        - Enter “password” for both accounts
        - Click on "Log In" button corresponding to user status (teacher or student).
    - User selects the “Create account” button
        - Enter a full name for the new user.
        - Enter a username for the new user.
        - Enter a password for the new user.
        - Click on the “Create Account” button based on the user status.
- Landing page/Main Menu is visible.
    - For Student
        - Take Quiz
            - Click on “Take Quiz” button
            - Click on CS18000 under the select course drop down
            - Click on New Quiz under the select quiz drop down menu
            - Attempt questions including Multiple Choice, True/False and Fill in the blanks.
            - Submit quiz
        - Edit Account
            - Click on “Edit Account” button
            - Enter the new credentials
            - Click on Change with respect to the status (teacher in this case)
        - Delete Quiz

            - Click on “Delete Account” button
            - Re-enter the password, “password”
        - Log out
            - Click on “Logout” button
            - On the logging out window, click OK to finish the process
- For Teacher
    - To create a new course
        - Click on “Create Course” button
        - Enter the text field with a name of choice
        - Dialog pops up with message “Course Successfully created”
    - To modify an existing course
        - Click on “Modify Course” button
        - Select the course you want to modify
        - To create a Quiz manually
            - Select the create quiz button
            - In the quiz creator window, write a question in the insert question field
            - Select the type of question
            - Click on add question to add new questions
            - Click on delete to remove undesired questions
            - Click on “Save Quiz” button to save progress
            - Window for quiz name pops up, fill the quiz name requirement
        - To create a Quiz using File Import
            - Select the create quiz button
            - In the quiz creator window, write a question in the insert question field
            - Click on import quiz button
            - Select the file “Example Quiz.txt” from the local storage
            - Click on “Save Quiz” button to save progress
            - Window for quiz name pops up, fill the quiz name requirement
        - To edit a quiz
            - Click on “Edit Quiz” button
            - Select the quiz you want to edit
            - In the quiz creator window, write a question in the insert question field
            - Select the type of question
            - Click on add question to add new questions
            - Click on delete to remove undesired questions

- To delete a quiz
    - Click on “Delete Quiz” button
    - Select the quiz you want to delete
    - Quiz is deleted
- To View/Grade Submissions
    - Click on “View/Grade Submissions”
    - Select the submission from the drop down menu which includes student username and the name of the quiz within the course
    - All the details of the attempted quiz are now visible
    - Edit Grades
        - Click on “Edit Grades” button
        - Select the question number you want to edit grade for
        - Enter a value for maximum points that could be earned with the question and click OK
        - Enter a grade point for the student you are evaluating and click OK
        - View the updated changes in the submission window
    - Close the window with “Close button”
- Click on “Exit” button to exit out of modify course window

- Edit Account
    - Click on “Edit Account” button
    - Enter the new credentials
    - Click on Change with respect to the status (teacher in this case)
- Delete Quiz
    - Click on “Delete Account” button
    - Re-enter the password, “password”
- Log out
    - Click on “Logout” button
    - On the logging out window, click OK to finish the process

Files and their formats:

Quiz Input File Format: Example in Repository Filename = “QuizInputExample.txt”

quiz1 //(this is the quiz name)

--Question Start--

MC //(question type)

mcanswer //correct answer
mcquestion // the question
optA // option A
optB // option B
optC // option C
optD // option D
--Question Start--
TF
tfquiz1correctanswer
tfquiz1question
--Question Start--
FIB
fibcorrectanswer
fibquestion
--Quiz End--
--Quiz Start--
quiz2
--Question Start--
TF
tfquiz2correctanswer
tfquiz2question
--Quiz End--

Answer File Format: Example in Repository Filename = "AnswerExample.txt"
A // the correct answer

Submission List File Format: Example in Repository Filename = "SubmissionList.txt"

- This is one file that guarantees that the data persists (DO NOT EDIT)

stud1Example Quiz 1.txt
stud1Midterm1.txt

Course List File Format: Example in Repository Filename = "CourseList.txt"

- This is one file that guarantees that the data persists (DO NOT EDIT)

Example Course
CS18000

User_info.txt File Format: Example in Repository Filename - "User_info.txt"

- This is one file that guarantees that the data persists (DO NOT EDIT)

Test Teacher,teach1,password,true
Test Student,stud1,password,false

Pre-loaded Users:

- Teacher:
    - Full Name: Test Teacher
    - Username: teach1
    - Password: password
- Student:
    - Full Name: Test Student
    - Username: stud1
    - Password: password

This begins the description and methods of each class:

Course.java

Description:

This class is a representation of a course. It includes an array list of quizzes and a name. It is the center of the building and editing of a lot of courses and quiz methods.

Relationship to other classes

A course is made up of an array list of quizzes, which in turn are made up of an array list of questions. The session.java main method implements the course class.

Testing:

Testing of this class is included in TestCourse.java and in SessionTest.java

Fields

Field Name	Type	Access Modifier	Other modifiers	Description
name	String	private		The name of the course and course.txt file.
course	ArrayList <Quiz>	private		An arraylist of quizzes making up the course
quizStartString	String	private	static final	String used for course.txt file formatting
questionStartString	String	private	static final	String used for course.txt file formatting

quizEndString	String	private	static final	String used for course.txt file formatting
courseEndString	String	private	static final	String used for course.txt file formatting

Constructors

Parameters	**Access Modifiers**
String name*	public
String name, ArrayList<Quiz> quizzes	public

*This constructor reads from a course.txt file to create the course, using the name to identify the file.

Methods

Method Name	**Return Type**	**Access Modifier**	**Input Parameters**	**Description**
getName	String	public		Returns name of course
setName	void	public	String name	Sets parameter as name of course
getCourse	ArrayList<Quiz>	public		Returns ArrayList of course's quizzes.
setCourse	void	public	ArrayList<Quiz> course	Sets parameter as ArrayList of course's quizzes
saveCourse	void	public		Creates a file which saves a course and all of its information according to a specific format
removeQuiz	boolean	public	String quizName	Removes the quiz of specified name from the course

Question.java

Description:

The class is a singular instance of a question. It included the question itself, the type of question, the correct answer, and the options. It allows many other classes to work and provides a solid basis of a wide variety of information. The toString() method of the Question class is the groundwork for printing and taking a quiz.

Relationship to other classes:

This class is the basis of almost everything in this assignment. The question class allows the building of the quiz (an array list of questions) in Quiz.java. A course (Course.java) is made up of one or more quizzes. All of these are implemented and used in the Session.java class.

Testing:

Testing of this class is included in TestQuestion.java. This tests the methods of this class. I also used a separate main method to guarantee that everyone worked properly in this class. The testing of the Session.java class also tests many different aspects of Question.java.

Fields

Field Name	Type	Access Modifier	Description
type	String		The type of question (MC, TF, or FIB)
correctAnswer	String		The correct answer
question	String		The question or statement
optionA	String		If multiple choice, option A. Else, null
optionB	String		If multiple choice, option B. Else, null
optionC	String		If multiple choice, option C. Else, null
optionD	String		If multiple choice, option D. Else, null

Constructors

Parameters	Access Modifier(s)

String type, String correctAnswer, String question, String optionA, String optionB, String optionC, String optionD	public

Methods

Method Name	**Return Type**	**Access Modifier**	**Input Parameters**	**Description**
getType()	String	public	None	Returns the type of the question
setType()	void	public	String type	Set the type to the new string you input
getCorrectAnswer()	String	public	None	Returns the correct answer of the question
setCorrectAnswer()	void	public	String correctAnswer	Set the correct answer to the new string you input
getQuestion()	String	public	None	Returns the question/statement of the question
setQuestion()	void	public	String Question	Set the question/statement to the new string you input
getOptionA()	String	public	None	Returns the option A of the question
setOptionA()	void	public	String optionA	Set the option A to the new string you input
getOptionB()	String	public	None	Returns the option B of the question
setOptionB()	void	public	String optionB	Set the option B to the new string you input
getOptionC()	String	public	None	Returns the option C of the question
setOptionC()	void	public	String optionC	Set the option C to the new string you input
getOptionD()	String	public	None	Returns the option D of the question

setOptionD()	void	public	String optionD	Set the option D to the new string you input
toString()	String	public	None	Return a string of the following format (used for printing): If MC: question + "\n" + "A. " + optionA + "\n" + "B. " + optionB + "\n" + "C. " + optionC + "\n" + "D. " + optionD + "\n"; If TF: question + "\n"; If FIB: question + "\n";
toString2()	String	public	None	Return a string of the following format (used for reading and writing a file): If MC: type + "\n" + correctAnswer + "\n" + question + "\n" + "A. " + optionA + "\n" + "B. " + optionB + "\n" + "C. " + optionC + "\n" + "D. " + optionD + "\n"; If TF: type + "\n" + correctAnswer + "\n" + question + "\n"; If FIB: type + "\n" + correctAnswer + "\n" + question + "\n";

Quiz.java

Description:

The class is a singular instance of a quiz. It is made up of an array list of questions and a name. This class however does much more than this. This class allows you to modify a quiz, read submissions, and save submissions.

Relationship to other classes:

Quiz is made up of an array list of one or more questions (Question.java). A course (Course.java) is made up of one or more quizzes. All of these are implemented and used in the Session.java class. Many of the methods below are used in the Session.java class for final implementation and printing purposes as well.

Testing:

Testing of this class is included in TestQuestion.java. This tests the methods of this class. I also used a separate main method to guarantee that everyone worked properly in this class. The testing of the Session.java class also tests many different aspects of Question.java.

Fields

Field Name	Type	Access Modifier	Description
quiz	ArrayList<Question>		An array list of questions that make up the quiz
name	String		The name of the quiz for a variety of internal purposes and identification

Constructors

Parameters	Access Modifier(s)
ArrayList<Question> quiz	public

Method Name	Return Type	Access Modifier	Input Parameters	Description
getQuiz()	ArrayList<Question> quiz	public	None	Returns the array list that represents the quiz
setQuiz()	void	public	ArrayList<Question> quiz	Set thequiz to the new array list you input
getName()	String	public	None	Returns the string of the name of the quiz
setName()	void	public	String name	Sets the name of the quiz to the input

saveSubmission()	void	public	String username, ArrayList<String> answers, double quizTime	Saves the submission of the current quiz with the answers in the array list, the username, and the quizTime. It allows for the teacher to look back at submissions and write the files of submissions.
readSubmission()	public	String	filename	Reads the submission given as the filename input. It returns the string representation of it in response to the teacher requesting to see it.
modifyQuiz()	void	public	int questionNumber, String type, String correctAnswer, String question, String optionA, String optionB, String optionC, String optionD	Sets a question object to the specified parameters. Allows quiz modification.
toString()	String	public	none	Returns a string representation of a quiz. It calls the question toString() method to do so.
modifyGrade()	void	public	double gradeValue, String fileName	Updates the grade of the student with the parameter gradeValue within the submission file given by fileName

User.java

Description:

The class is a singular instance of a user. It includes the user's name, user's username, and user password, and checks if the user is a student or a teacher. It allows many other classes to work and provides a solid basis for a wide variety of information.

Relationship to other classes:

This class is heavily in the Session class in implementing all the features surrounding the user.

Testing:

Testing of this class is included in TestUsers.java. This is a class that contains the main method which will allow the user to log in as either a teacher or student. Teachers will be able to share and edit quizzes through the terminal, and students will be able to access and take quizzes.

Fields

Field Name	Type	Access Modifier	Description
name	String	private	The type of question (MC, TF, or FIB)
username	String	private	The correct answer
isTeacher	boolean	private	The question or statement
password	String	private	If multiple choice, option A. Else, null

Constructors

Parameters	Access Modifier(s)
String name, String username, String password, boolean isTeacher	public

Methods

Method Name	Return Type	Access Modifier	Input Parameters	Description
getName()	String	public	None	Returns the name of the user
setName()	void	public	String name	Set the name of the user
getUsername()	String	public	None	Returns the username of the user
setUsername()	void	public	String	Set the username of the user

			username	
getPassword()	String	public	None	Returns the password of the user
setPassword()	void	public	String password	Set the password of the user
getIsTeacher()	boolean	public	None	Returns the boolean for if the user is teacher
checkUserExists()	boolean	public	None	Returns true if the user already exists and catches any IOException.
createNewUser()	void	public	None	Creates a new user and catches any IOException while reading from the UserList.txt file.
writeUserData()	void	public	None	Writes user data using a print writer.
getUserID()	String	public	None	Makes user ID based on the variables in the class
deleteUserData()	void	public	None	Modify the user information based on the details and new input

LogInGUI.java

Description:

The LogInGUI class is the first class that a user sees. It welcomes the user, allows the user to choose to create an account or login, and receives the input for creating an account or logging in. After it does that, it sends the information to the server to either login or create an account. If it is a success, it calls the main menu GUI with the proper user information. If it is not a success, it allows the user to retry either logging in or creating an account. This class is very important, as it is the first thing the user sees and interacts with.

Relationship to other classes:

The LogInGUI class is where this whole program starts. When you open a client, this class is called. After running and either creating a user or logging in, it calls the main menu class with the proper user to continue running the program. The LogInGUI is the entry point of our quiz!

Testing:

Testing of the LogInGUI are the first few test cases in the test case file. It also is included in almost every test case, since you must be signed in to do anything! We also did a lot of testing to guarantee this worked manually. Every time we test anything in our program, you must sign in or create an account first.

EditAccountGUI.java

Description:

This class is the GUI that allows a user to edit their account. It receives the user information and allows the user to change their account information. Once this is done, it sends the new information to the server. If this all is a success, it calls the main menu again with the new user information.

Relationship to other classes:

This class is called from the main menu and is passed the original user information. It also calls the main menu again after it completes, with the new user information.

Testing:

Testing of the EditAccountGUI is included in the test cases. We also did a lot of testing to make sure this worked and edited the files correctly manually.

Submission.java

Description:

This class stores a single submission, including information about the student and quiz, the point value and points obtained, and an array list of answer objects. It also has methods to construct a submission, read the Submission_List.txt file and construct submissions from it, and save a new submission to the file.

Relationship to other classes:

Constructors in this class call the answer constructor repeatedly. The server thread interacts with the submission class several times, since the server thread must interact with submission saving and reading using Submission.java methods.

Testing:

A main method was temporarily implemented in this class that tested the constructor, the method to read the submission list file, and the method to save a submission to a file.

Answer.java

Description:

This class stores a single answer in a quiz, its point value, and its points obtained. A constructor in this class also automatically grades the answer given

Relationship to other classes:

The submission class contains an array list of answer objects. The constructor to make a submission object also calls the answer constructor several times using the given answers. The code to manually grade a quiz also involves editing the grades of answer objects.

Testing:

This class was tested as during the testing for the submission class. We made sure the automatic grading function worked properly and ensured the other constructor worked as well.

MainMenuGUI.java

Description:

This class runs a Main Menu which allows the user to call several other GUIs after logging in. It also locally implements functionality for logging out and deleting an account as well.

Relationship to other classes:

This class is capable of calling the Create Course GUI, Modify Course GUI, Take Quiz GUI, View Grades GUI, and Edit Account GUI. It also interacts with the Server Thread several times and is itself called by the Log In GUI.

Testing:

This class was tested extensively by running it several times testing all the implemented buttons ensuring they functioned as intended.

ViewGradesGUI.java

Description:

This class runs a GUI that allows a student to view their submitted quizzes and grades. The viewable submissions can be of any length due since the window is scrollable.

Relationship to other classes:

This class is called by the Main Menu. It also constructs submission objects using the Submission class.

Testing:

This class was tested by running it extensively and making sure the close button and scroll feature worked properly. We checked to make sure the quiz was formatted properly with all questions, answers, and points as well.

ClientSession.java

Description: This class is responsible for connecting the client to the server. It prints a confirmation message upon connecting

Relationship to other classes: This class contains our main method for calling the program. After the client connects, this class calls LogInGUI.java and passes the socket and the object input and output streams.

Testing: The confirmation message is used to ensure that a connection is secured.

CreateQuizGUI.java

Description: This class is responsible for letting teachers create new quizzes by either importing them, starting with an existing quiz, or starting from scratch. The gui has two constructors, one for editing an existing quiz, and one for creating a new one from scratch.

Relationship to other classes: This class is called by ModifyCourseGUI.java, and is called by pressing either the edit quiz button or the create quiz button. When creating a new quiz, a blank quiz is given to the user with nothing but a blank multiple choice question. When editing a quiz, the user is able to select from a list of pre-existing quizzes in the current course. After saving, the quiz is either added to the existing list of quizzes for the course, or replaces the edited quiz.

Testing: This GUI was tested by creating many different types of quizzes using as many combinations of questions and answers as possible. After creating the quiz, we ensured that the quiz was visible right away in the course. We also tested to make sure that creating quizzes was concurrent by opening multiple clients and seeing that no errors occurred.

ModifyCourseGUI.java

Description: This class allows teachers to modify a course by creating, editing, deleting, and viewing grades.

Relationship to other classes: To create or edit a quiz, this class will call CreateQuizGUI.java and use the corresponding constructor depending on which option is selected. If a teacher wants to view and grade student submissions, ViewSubmissionsGUI.java is called

Testing: We tested this GUI to ensure that it calls all of its other GUI's properly. When it comes to deleting quizzes, we ensured that the quiz was deleted on both the client and server side concurrently.

Server.java

Description: This is a class that starts a server on port 4243. The server listens for clients to connect, and when a client successfully connects, a new thread is created for the client to interact with the server

Relationship to other classes: Every time a client connects, Server.java calls ServerThread.java

Testing: We tested this class by ensuring that multiple clients could connect to the server and access data simultaneously.

ServerThread.java

Description: This class listens for instructions from the client, and upon receiving the instructions will send and store data from the client.

Relationship to other classes: ServerThread.java is called by Server.java every time a new client connects.

Testing: This class was tested by ensuring that it stores and sends data correctly. It was relatively simple, since the run() method is essentially one big selection structure. Every time a new if statement was added to the run() method, it was tested from the client side by sending the corresponding instructions and checking to ensure that data was stored correctly in the right files.

TakeQuizGUI.java

Description: TakeQuizGUI.java allows students to take a quiz. Students are able to click through each question and enter their choices. When the GUI is loaded in, a timer is displayed and begins counting up. Students may import answers to questions if they would like. Lastly,

when the student is done with the quiz, they press the Submit Quiz button, the timer is stopped, and a new submission is created and stored.

Relationship to other classes: This class is called by MainMenuGUI.java when a student presses the Take Quiz button. After the quiz is taken, it uses Submission.java and ServerThread.java to create and store a submission

Testing: To test this class, we took many quizzes through the GUI. We ensured that every type of question was properly displayed, and that answers were saved between questions. After submitting, we ensured that the submission data was accurate.