UNIVERSITY OF
BIRMINGHAM

Software Workshop 2

—

Real Time Multi-User Quiz System

Benjamin Crispin,
Samuel Farmer,
Deedar Fatima,
Rowan Stringer,
Josh Wainwright

Group Osaka

Supervisor: Joe Gardiner

Date: March 2014

Contents

1 Protocol

The protocol for communicating between the different parts of the system is based around objects. There exists an object class that can be created for any of the several message types that could be needed to transfer information from the server to the client, or from the client to the server. Each of these objects implements the `Serializable` interface, allowing them to be converted to bytes and transfered across the socket connection using object streams.

AnswerResponse object
To respond to a question, the Student selects the desired option. This information is passed to the server using an AnswerResponse object which simply holds the response and the timestamp to indicate when they made the selection.

DisplayQuestion object
In order to signify that the allotted time for the current question has ended, and the next question should be displayed, the server sends a DisplayQuestion object to all of the clients and they should move on to the next question in the Quiz object and change the GUI accordingly.

LoginReply object
Once a loginRequest has been received by the server, a LoginReply will be sent back. This gives the client the information about the requested login, most importantly if the login was successful, as well as the type of user that made the login, Student or Admin. This information is used to display the correct user interface.

LoginRequest object
This is the first object that could be created. It is sent, by the client, to the server and contains the username of the Student that is attempting to login and the `java.lang.String` hash code of the inserted password. Though the security concerns of such a trivial system are non-existent, the password is never stored in plaintext.

Question object
There exist several question objects in each Quiz object. They store the information required to present a Student with a question and the possible answers. Again, there is very little functionality as the question only serves as a wrapper for the question text and the possible answers that the user could respond with.

Quiz object
This is the most important object. It has very little functionality, simply acting as a wrapper to hold and easily transfer several Question objects.

QuizRequest object

Score object
When the quiz has been completed, each of the clients will display the position of it's user relative to the other Students. This object contains the score of all of the users so that each of the clients can work out where they are in the ranking.

StartQuiz object
Once the user has successfully logged into the system, the next major event is the start of the quiz. This is signaled by an Admin user who is connected to the server, this information must then be relayed to each of the connected clients. A

StartQuiz object is sent to each of the clients, who, on receiving it, will display the first question to the user.

2 Client

The client exists to accept messages sent by the server, and present them in an order and a format that the user interface can present to the user, as well as accept the information entered by the user into the user interface and pass it to the server.

2.1 System Design

1. **Login**

   When a client is started, a `QuizClient` object is created and starts the main loop. The initial stages set up the connection with the server and waits for the user to login. When the user enters their information, a `LoginRequest` object is created and pass to the server containing the username and password, to be checked against the contents of the database. The client then waits for a reponse from the server to indicate if the login was successfull or not. This comes in the form of a `LoginReply` object. If this says that the login was unsuccessful, the user is asked to re-input their details, otherwise, the display is changed and the options screen is shown.

2. **Student/Admin**

   There exists separate functionality withing the client depending on if the user is a Student user, i.e. going to be answering questions, or an Admin, i.e. the teacher who starts and moderates the quiz. Distinguishing between these two is done by the server by checking the details associated with that user in the database. The `LoginReply` object contains this information and the client can then display the correct interface depending on the type of user that logged in.

3. **Client Listens from Server**

   From here, the user can select the "Start Quiz" option to start the quiz. This causes the display to change to display the waiting screen and the client waits for information from the server.

   From this point on, the client waits for any object to be sent from the server and acts according to what that object was. The possible objects that the client now expects to be able to distinguish between are:

   - `Quiz`
   - `StartQuiz`
   - `DisplayQuestion`
   - `Score`

   `Quiz`
   : This object contains the information about the quiz itself, the number of questions, their contents and the duration that each question should be displayed for. It should only ever be recieved once by the client, at the start of the session, reducing the transfer of information over the connection.

`StartQuiz`
: Once an Admin has logged in, they have control over the start of the quiz. When they decide to start the quiz, this object is sent to each of the listening clients and so the client will procede to show the first question from the `Quiz` object.

`DisplayQuestion`
: The first question is displayed as soon as the `StartQuiz` object is recieved. After this point in the quiz, the questions are changed when this object is received. The value contained verifies which question is to be displayed.

`Score`
: After each question has been answered, the client can display a leader board showing the score of all the clients that have so far answered the current question along with the current client's position in this list. This object tells the client the relevant information for displaying the scores of the other clients.

4. **Sent by Client**

   There are also a number of objects that the client can create and send to the server at different stages of the quiz:

   - `LoginRequest`
   - `QuizRequest`
   - `AnswerResponse`

`QuizRequest`
: This is sent by the client when an Admin is logged in in order to request a particular quiz from the server. Since the server can hold many quizzes, each with their own set of questions and answers, the Admin has the option to choose which of these to play when they log in.

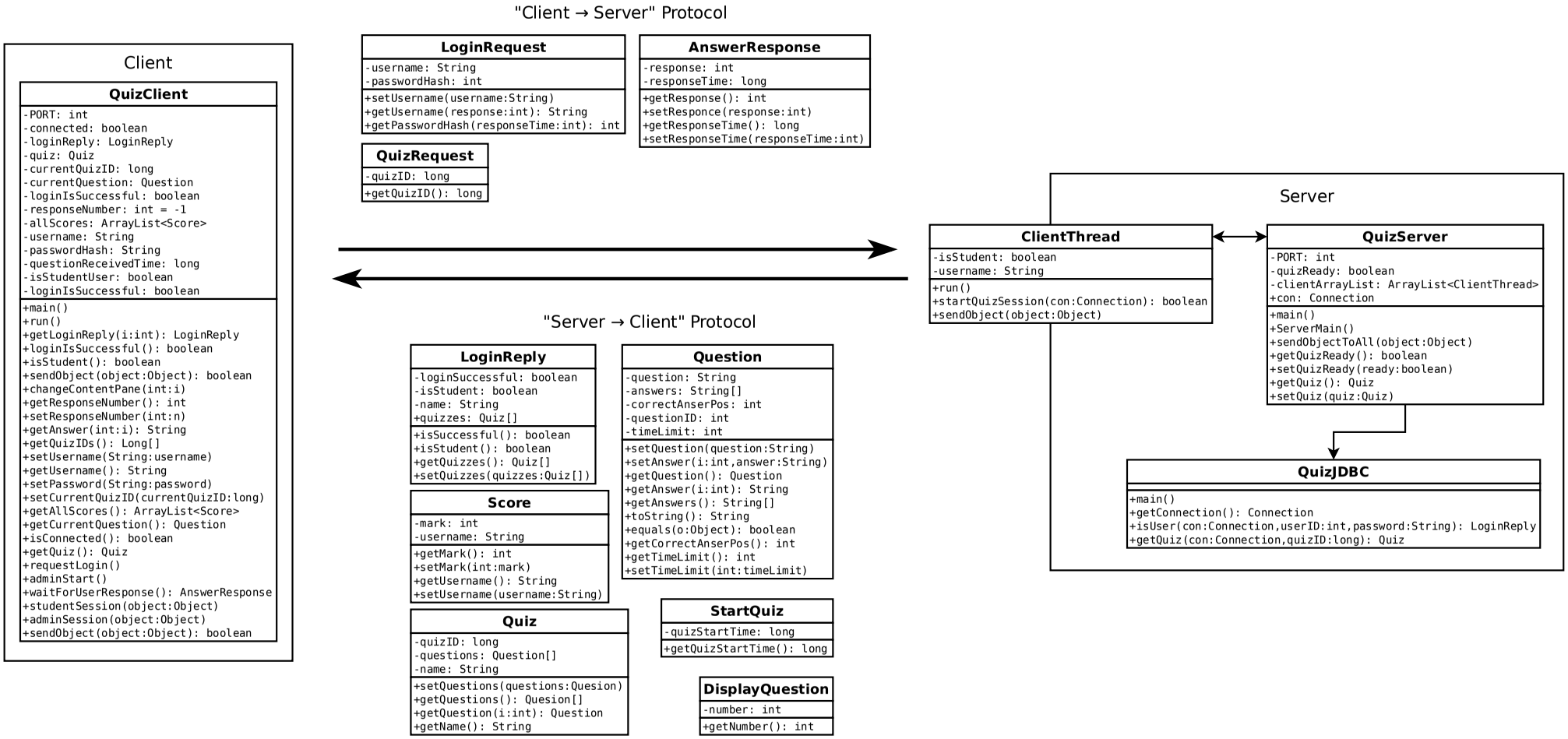
`AnswerResponse`
: This is the object that tells the server what answer the Student gave. It contains their response, so that it can be logged in the database, and the time it took for the Student to make their selection.

3 Server

4 Database

References

A Class Diagram


Client
QuizClient
-PORT: int
-connected: boolean
-loginReply: LoginReply
-quiz: Quiz
-currentQuizID: long
-currentQuestion: Question
-loginIsSuccessful: boolean
-responseNumber: int = -1
-allScores: ArrayList<Score>
-username: String
-passwordHash: String
-questionReceivedTime: long
-isStudentUser: boolean
-loginIsSuccessful: boolean
+main()
+run()
+getLoginReply(i:int): LoginReply
+loginIsSuccessful(): boolean
+isStudent(): boolean
+sendObject(object:Object): boolean
+changeContentPane(int:i)
+getResponseNumber(): int
+setResponseNumber(int:n)
+getAnswer(int:i): String
+getQuizIDs(): Long[]
+setUsername(String:username)
+getUsername(): String
+setPassword(String:password)
+setCurrentQuizID(currentQuizID:long)
+getAllScores(): ArrayList<Score>
+getCurrentQuestion(): Question
+isConnected(): boolean
+getQuiz(): Quiz
+requestLogin()
+adminStart()
+waitForUserResponse(): AnswerResponse
+studentSession(object:Object)
+adminSession(object:Object)
+sendObject(object:Object): boolean
"Client → Server" Protocol
LoginRequest
-username: String
-passwordHash: int
+setUsername(username:String)
+getUsername(response:int): String
+getPasswordHash(responseTime:int): int
AnswerResponse
-response: int
-responseTime: long
+getResponse(): int
+setResponce(response:int)
+getResponseTime(): long
+setResponseTime(responseTime:int)
QuizRequest
-quizID: long
+getQuizID(): long
Server
ClientThread
-isStudent: boolean
-username: String
+run()
+startQuizSession(con:Connection): boolean
+sendObject(object:Object)
QuizServer
-PORT: int
-quizReady: boolean
-clientArrayList: ArrayList<ClientThread>
+con: Connection
+main()
+ServerMain()
+sendObjectToAll(object:Object)
+getQuizReady(): boolean
+setQuizReady(ready:boolean)
+getQuiz(): Quiz
+setQuiz(quiz:Quiz)
QuizJDBC
+main()
+getConnection(): Connection
+isUser(con:Connection,userID:int,password:String): LoginReply
+getQuiz(con:Connection,quizID:long): Quiz
"Server → Client" Protocol
LoginReply
-loginSuccessful: boolean
-isStudent: boolean
-name: String
+quizzes: Quiz[]
+isSuccessful(): boolean
+isStudent(): boolean
+getQuizzes(): Quiz[]
+setQuizzes(quizzes:Quiz[])
Question
-question: String
-answers: String[]
-correctAnserPos: int
-questionID: int
-timeLimit: int
+setQuestion(question:String)
+setAnswer(i:int,answer:String)
+getQuestion(): Question
+getAnswer(i:int): String
+getAnswers(): String[]
+toString(): String
+equals(o:Object): boolean
+getCorrectAnserPos(): int
+getTimeLimit(): int
+setTimeLimit(int:timeLimit)
Score
-mark: int
-username: String
+getMark(): int
+setMark(int:mark)
+getUsername(): String
+setUsername(username:String)
StartQuiz
-quizStartTime: long
+getQuizStartTime(): long
Quiz
-quizID: long
-questions: Question[]
-name: String
+setQuestions(questions:Quesion)
+getQuestions(): Quesion[]
+getQuestion(i:int): Question
+getName(): String
DisplayQuestion
-number: int
+getNumber(): int