Name: Trần Thị Anh Thư (1751036)
Lê Phạm Ngọc Yến (1751028)

Class: 17CTT2

Subject: CS494 - Internetworking Protocol

Lab01: SOCKET PROGRAMMING

REPORT

Team information:

No.	Name	Student ID	Contribution (%)
1	Trần Thị Anh Thư	1751036	50
2	Lê Phạm Ngọc Yến	1751028	50

Game description:

Game title: Racing Arena

Game story: Racing Arena is all about using your brain power to excel in a race of intelligence.

The **server** (here I will prefer as **Game Master**) will first pick 2 configurations as desire, the first is **number of racers** and second is **race length**. Once these configurations are set and the connection has opened, Game Master can no longer change these configurations.

Game Master can only send out question once there are enough number of **clients** (here I will prefer as **racers**) join the game. Once that requirement is met then the Game Master can send out question.

The racers will have only 10 seconds to answer the question.

- If the answer is **correct** and is **fastest** then the racer will receive base points of 2 and more points to come if other races send incorrect answer.
- If the answer is **correct** but is not **fastest** then the racer will only receive 1 point.
- If the answer is **incorrect** then the racer will receive -1 point. If the racer makes 3 incorrect answer then he/she will be eliminated.
- If the racer **doesn't send in the answer in time** than he/she will receive -1 but won't be counted in the incorrect answer.

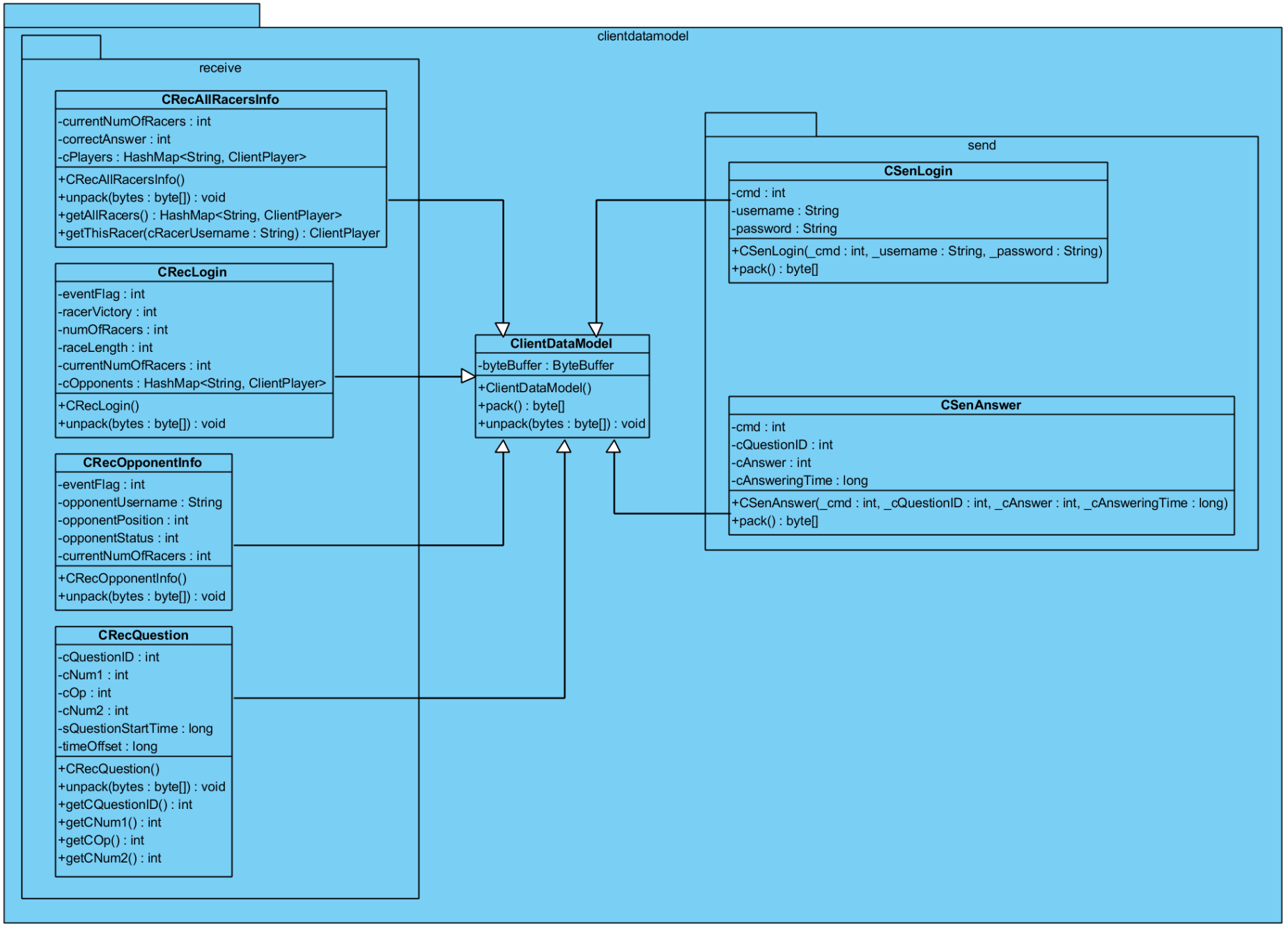
The game will be stop once there is a winner or there are winners or all racers have been eliminated.

Require install: H2 Database

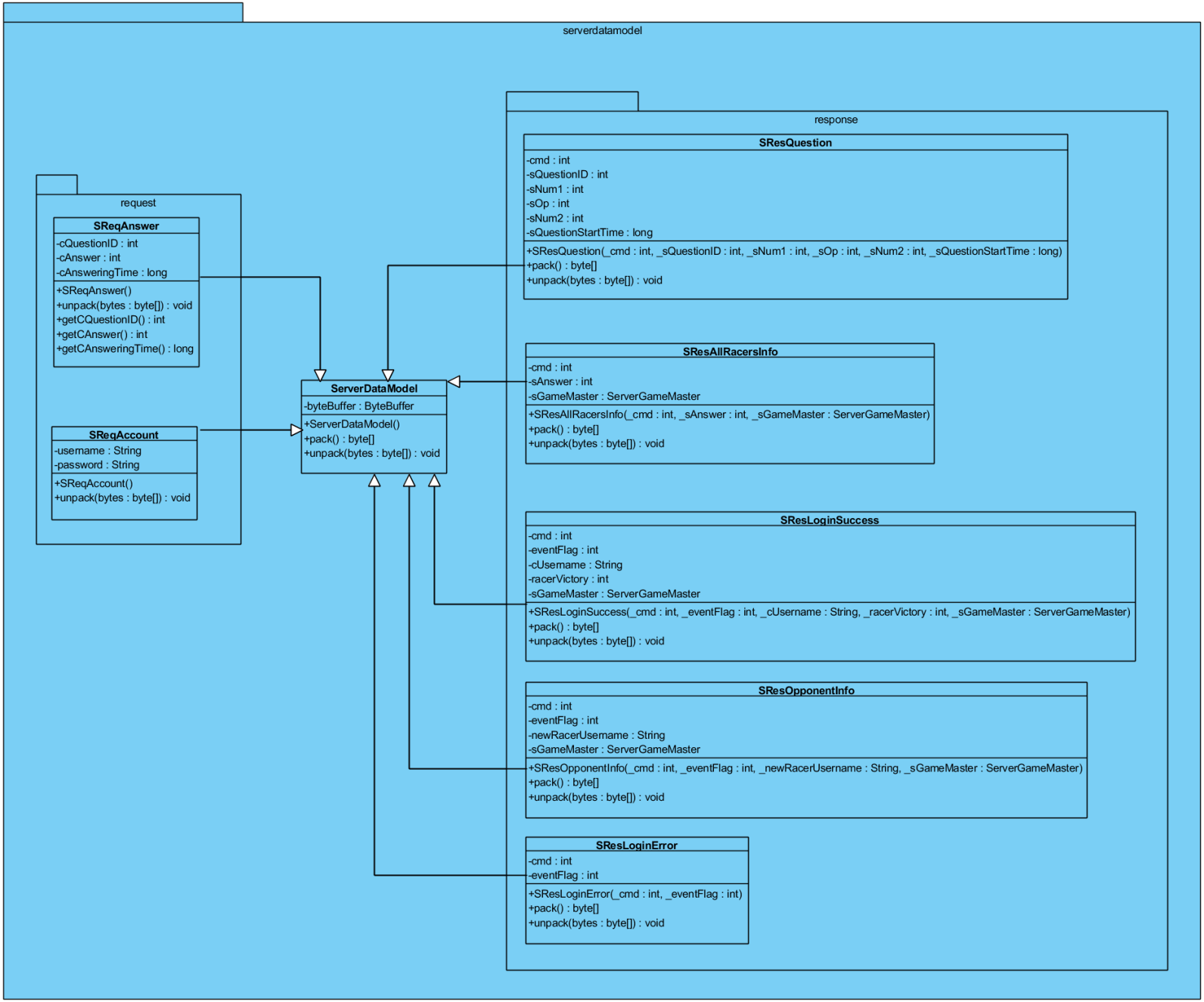
Server-client communication and packet structure

Packet Structure class diagram

Client packets



Server packets



Server-Client Main Communication Steps:

Case	SERVER		CLIENT	
	Actions	Packet Structure	Actions	Packet Structure
Server-Client Handshake	1. Open connection			
			2. Send connection request to server at known host and port	
	3. Accepting the request by creating a socket for this client			
Case Login			1. Send login request to server	Packet structured as class **CSenLogin**
	2. Server receives packet, unpacks and finds command type is Login	Packet structured as class **SReqAccount**		
	3. Run login logic			
	4. Send login result to client	Packet structured as class **SResLoginError** and **SResLoginSuccess**		
			5. Receive login result	Packet structured as class **CRecLogin**
			6. Unpack packet and act accordingly to the result	
Case Question	1. When receive enough clients, server allows the admin to give question	Packet structured as class **SResQuestion**		
			2. Receive the question from server	Packet structured as class **SRecQuestion**
			3. Unpack packet and act accordingly to the result	
Case Answer			1. Send answer to server	Packet structured as class **SSenAnswer**
	2. Server receive the individual answer from client	Packet structured as class **SReqAnswer**		
	3. Server immediately check for this client's answer's correctness and give based points			
	4. When timer is up, server check through all answers of clients and do extra calculation: rewards fastest, eliminate 3-wrong users, find the victors			
	5. Send the final validation after each question to all clients	Packet structured as class **SResAllRacersInfo** with command type is **RESULT**		

			6. Receive the evaluation packet	Packet structured as class **SRecAllRacersInfo**
			7. Unpack and update the client and its opponents' status	
Case Replay Game	1. When there is a (are) victor(s) or all clients are eliminated, server allow the admin to replay the game. Announce this to all clients	Packet structured as class **SResAllRacersInfo** with command type is **REPLAY**		
			2. Receive the packet	Packet structured as class **SRecAllRacersInfo** with command REPLAY
			3. Unpack and update the client and its opponents' status (to initial state)	

Implementation logic

- The network module is running on a separate thread so that it does not block UI thread. For the server, when it open a socket for a client, it will communicate with such client on a new thread. When such client is disconnected, it closes the connection and stop the according thread.
- All the packing, sending and receiving tasks are the responsibility of Network module on both Server and Client
- Due to complex structures, please refer to source code (which has been commented in detail) to see the evaluation logic of the servers in file **ServerCSocketThread.java** (for login evaluation) and **ServerGameMaster.java** (for game play evaluation).

GUI (MFC, WPF, Swing, etc.)

- We create GUI with Swing.

Server GUI:

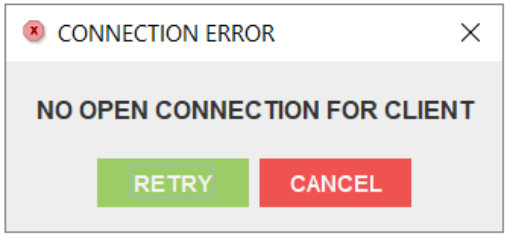
- Allow game master to set **configuration options**.
  - **Number of racers** can join the race.
  - **Race length** that each racer has to reach in order to win.
- Click **"Open Connection" button** to open new server after finish setting configurations.
  - These configurations have to be set before open a server, and cannot be change once server is open.
- Has **indicator the number of people have joined the server**.
- Click **"Start Game" button** to start the race.
  - This button is clickable when number of people currently in the server has yet to reach the previously set configuration.
- Has a **"Racer Statistic" table** to:
  - Before start game: update who is in the server.

- After start game: racers' gain, status, and position in the race after every question.

- Has a **"Question Section"** to show the question, its correct answer, and a timer that is reset after every question.

Client GUI:

- If no open connection is available for client then this **error message** will popup.



  The **"Retry" button** will try to find an open connection.

  - If nothing found then the error message will still be there.
  - If found then it will connect to the server, open the Client GUI and turn off the message.

  The **"Cancel" button** will turn off the message and exit the program.

- Has **color themes** to change, 15 colors to be exact.

- Has **nickname and password input places** to either create a new racer in the database or retrieve old racer.

  - If new user then the initial the **number of victory** will show "0".
  - If racer is already in the database, then the number of victory will be set as according to racer achievement.

  Nickname and password text boxes also have **hover tooltips** to notify racers what they should input according to some of their constraints.

- Click **"Join Server" button** to connect to server.

  This button cannot be click if no nickname and password entered or the entered nickname is invalid.

  There can be 2 possible **notification** after clicking this button.

  - If the nickname of a racer has yet to connect to server or yet to exist in the database, then the racer will be noticed that he/she has join the server, "Login Successfully" .
  - If that nickname of a racer has been in the server, then the racer will be noticed that the nickname is already in, "Duplicated Login".

- Has a **timer** that counts down for every question sent from the server.

- Has a place to show **1st number**, **operator**, then **2nd number**. After 10s for answering, an **answer** will be updated.

- Has an **answer text box** for racer to input their answer.

  - This text box is only available when the question is sent and there is still time to answer.
  - Answer text box also has **hover tooltips** to notify racers that they should only input integer as answer.

- Click **"Submit Answer" button** to send the answer.

  - This button only clickable when the question is sent and there is still time to answer.
  - Racers can also press [Enter] to submit answer.

- Has a **status** next to the answer text box to update racer that they are correct, incorrect, or out-of-time for each question.
  - If correct, then notice the gaining point: +1 for normal, and +n (2 at minimum) for fastest.
  - If incorrect, then notice how many wrong answer racer has made, along with gain -1.
  - If out-of-time, then notice the gaining point: -1.
- Has a **racer status panel** to update the racer and his/her opponent position in the racer after every question.
  - Before joining the server, there is only one status bar of label "Me" to indicate that is racer (racer of a client always has his/her bar at the top).
  - After joining the server, the GUI will shows more bars indicating how many opponent the racer has.
    - If the racer is the first to join then the name of the bar will be updated when a new racer join.
    - If there have been some other racers that have already been joined the server then when the racer joins the server some of the bars have already been changed to the opponent racers' nickname.

Self-Evaluation:

No.	Requirements	Score	Evaluate
1	Use C/C++, Java, C#	2	2
2	Implement whole gameplay properly	3	2.5
3	Socket Non-blocking	2	2
4	Have a good GUI (MFC, WPF, Swing, etc.)	3	3
	Total	10	9.5