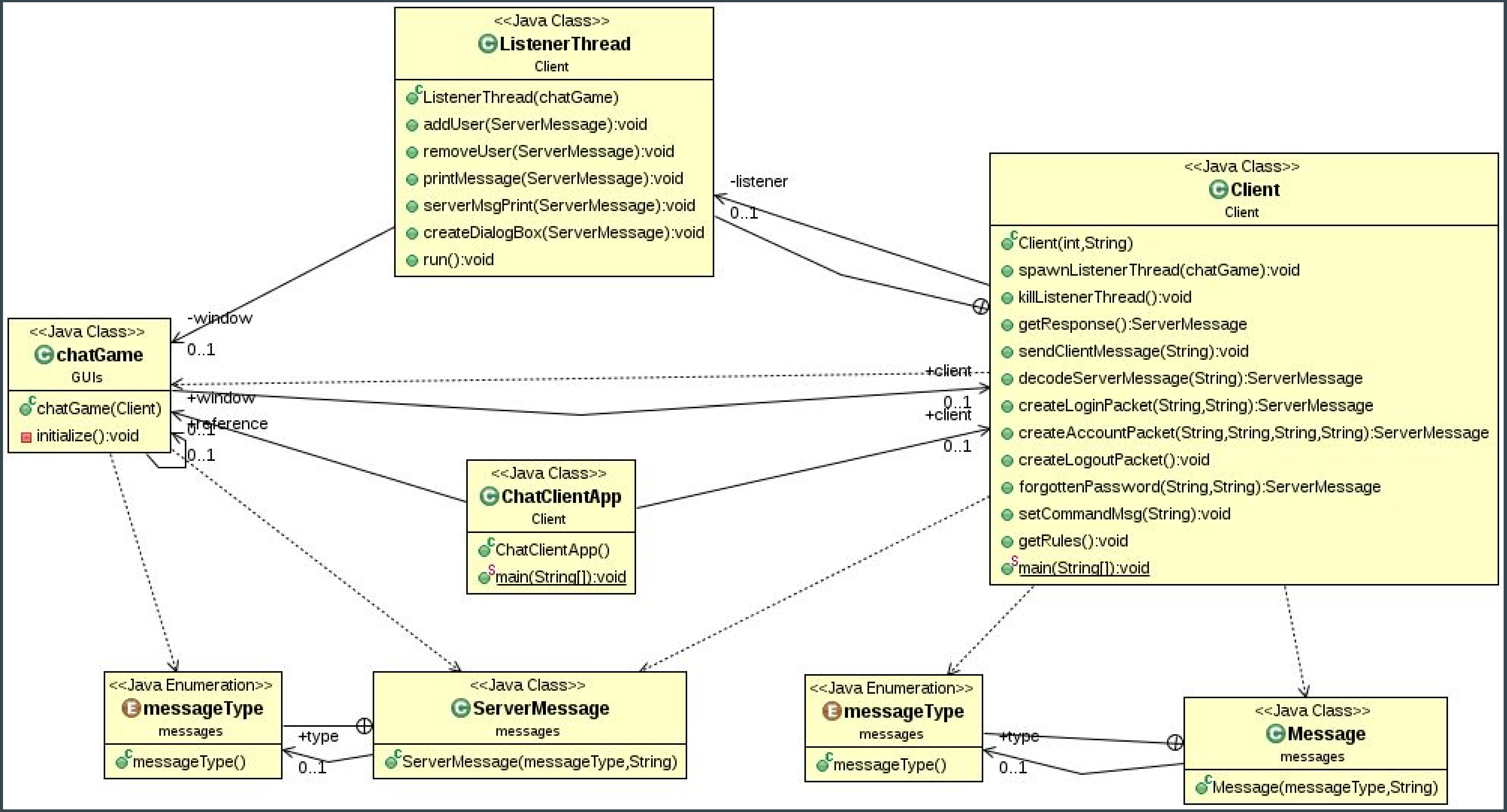
Team Nice - Presentation

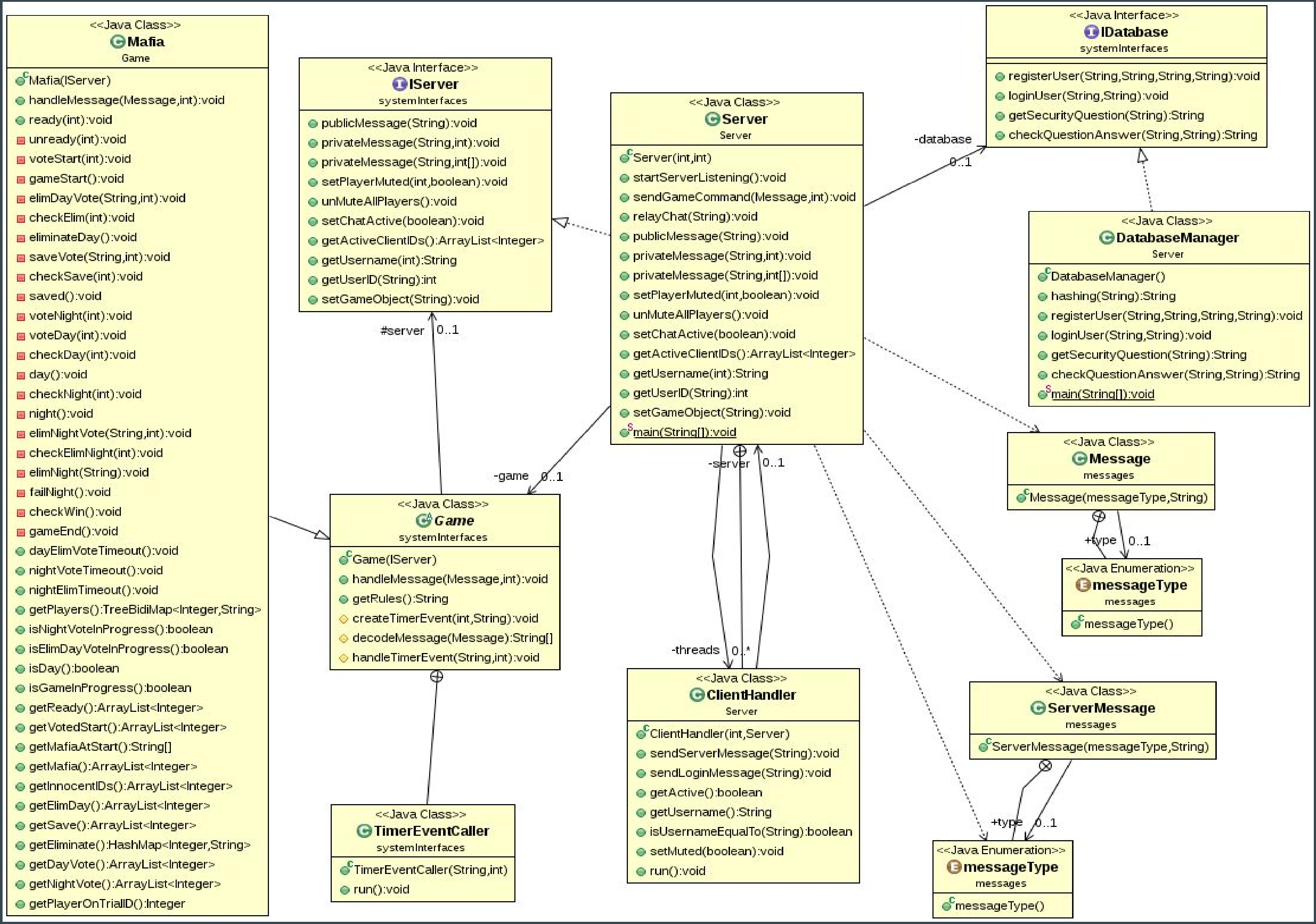
Project Topic

Client Architecture



<<Java Class>>
ListenerThread
Client
ListenerThread(chatGame)
addUser(ServerMessage):void
removeUser(ServerMessage):void
printMessage(ServerMessage):void
serverMsgPrint(ServerMessage):void
createDialogBox(ServerMessage):void
run():void
-listener
0..1
<<Java Class>>
Client
Client
Client(int,String)
spawnListenerThread(chatGame):void
killListenerThread():void
getResponse():ServerMessage
sendClientMessage(String):void
decodeServerMessage(String):ServerMessage
createLoginPacket(String,String):ServerMessage
createAccountPacket(String,String,String,String):ServerMessage
createLogoutPacket():void
forgottenPassword(String,String):ServerMessage
setCommandMsg(String):void
getRules():void
main(String[]):void
-window
0..1
<<Java Class>>
chatGame
GUIs
chatGame(Client)
initialize():void
+client
+window
0..1
+client
+reference
0..1
0..1
0..1
<<Java Class>>
ChatClientApp
Client
ChatClientApp()
main(String[]):void
<<Java Enumeration>>
messageType
messages
messageType()
+type
0..1
<<Java Class>>
ServerMessage
messages
ServerMessage(messageType,String)
<<Java Enumeration>>
messageType
messages
messageType()
+type
0..1
<<Java Class>>
Message
messages
Message(messageType,String)

Server Architecture



<<Java Class>>
Mafia
Game
Mafia(IServer)
handleMessage(Message,int):void
ready(int):void
unready(int):void
voteStart(int):void
gameStart():void
elimDayVote(String,int):void
checkElim(int):void
eliminateDay():void
saveVote(String,int):void
checkSave(int):void
saved():void
voteNight(int):void
voteDay(int):void
checkDay(int):void
day():void
checkNight(int):void
night():void
elimNightVote(String,int):void
checkElimNight(int):void
elimNight(String):void
failNight():void
checkWin():void
gameEnd():void
dayElimVoteTimeout():void
nightVoteTimeout():void
nightElimTimeout():void
getPlayers():TreeBidiMap<Integer,String>
isNightVoteInProgress():boolean
isElimDayVoteInProgress():boolean
isDay():boolean
isGameInProgress():boolean
getReady():ArrayList<Integer>
getVotedStart():ArrayList<Integer>
getMafiaAtStart():String[]
getMafia():ArrayList<Integer>
getInnocentIDs():ArrayList<Integer>
getElimDay():ArrayList<Integer>
getSave():ArrayList<Integer>
getEliminate():HashMap<Integer,String>
getDayVote():ArrayList<Integer>
getNightVote():ArrayList<Integer>
getPlayerOnTrialID():Integer
<<Java Interface>>
IServer
systemInterfaces
publicMessage(String):void
privateMessage(String,int):void
privateMessage(String,int[]):void
setPlayerMuted(int,boolean):void
unMuteAllPlayers():void
setChatActive(boolean):void
getActiveClientIDs():ArrayList<Integer>
getUsername(int):String
getUserID(String):int
setGameObject(String):void
#server
0..1
<<Java Class>>
Server
Server
Server(int,int)
startServerListening():void
sendGameCommand(Message,int):void
relayChat(String):void
publicMessage(String):void
privateMessage(String,int):void
privateMessage(String,int[]):void
setPlayerMuted(int,boolean):void
unMuteAllPlayers():void
setChatActive(boolean):void
getActiveClientIDs():ArrayList<Integer>
getUsername(int):String
getUserID(String):int
setGameObject(String):void
main(String[]):void
-database
0..1
<<Java Interface>>
IDatabase
systemInterfaces
registerUser(String,String,String,String):void
loginUser(String,String):void
getSecurityQuestion(String):String
checkQuestionAnswer(String,String):String
<<Java Class>>
DatabaseManager
Server
DatabaseManager()
hashing(String):String
registerUser(String,String,String,String):void
loginUser(String,String):void
getSecurityQuestion(String):String
checkQuestionAnswer(String,String):String
main(String[]):void
-game
0..1
<<Java Class>>
Game
systemInterfaces
Game(IServer)
handleMessage(Message,int):void
getRules():String
createTimerEvent(int,String):void
decodeMessage(Message):String[]
handleTimerEvent(String,int):void
<<Java Class>>
TimerEventCaller
systemInterfaces
TimerEventCaller(String,int)
run():void
-server
0..1
-threads
0..*
<<Java Class>>
ClientHandler
Server
ClientHandler(int,Server)
sendServerMessage(String):void
sendLoginMessage(String):void
getActive():boolean
getUsername():String
isUsernameEqualTo(String):boolean
setMuted(boolean):void
run():void
<<Java Class>>
Message
messages
Message(messageType,String)
+type
0..1
<<Java Enumeration>>
messageType
messages
messageType()
<<Java Class>>
ServerMessage
messages
ServerMessage(messageType,String)
+type
0..1
<<Java Enumeration>>
messageType
messages
messageType()

Team Organization

- Fozia Mehboob – GUI
- Jonathan Keslake – Server
- Jacob Smith – Database
- Vishnu Raman – Client
- Ian Kirkman - Game

Database Design

- Stores user registration information
- Queries existing and duplicate usernames to avoid errors and confusion in the client - server
- Forgotten Password button retrieves stored passwords when a security question is queried by the input of an existing usernames. The password is retrieved when the matching answer stored in the database entered satisfies the security question
- Unique primary key assigned to each user so that further tables may be created to link additional information being stored

Mafia Design

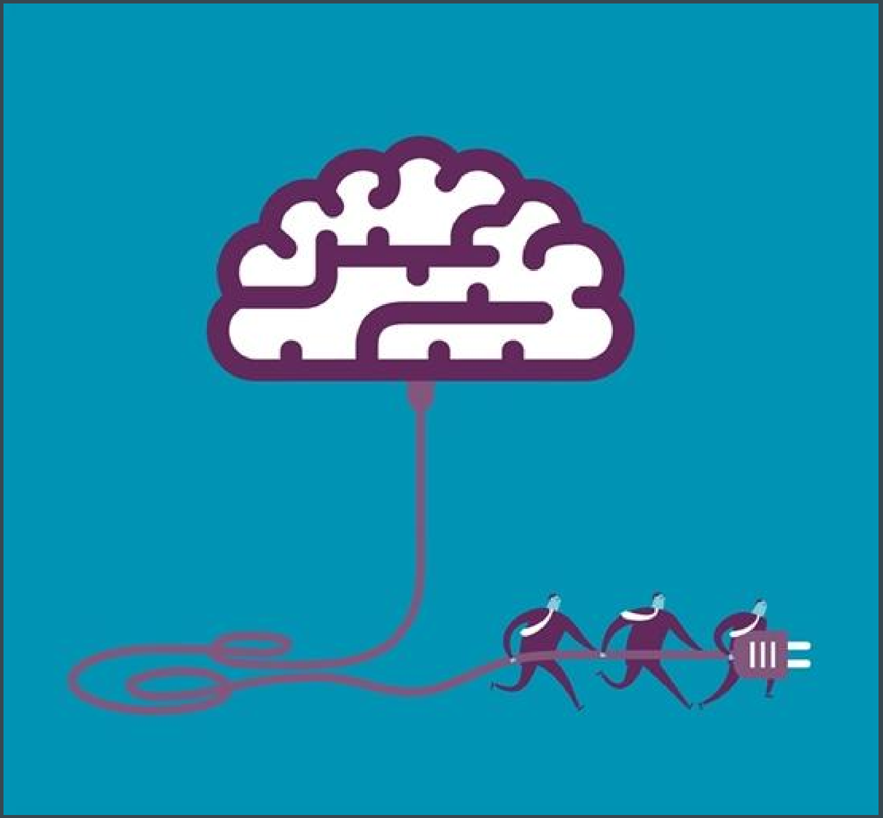
- Handles all user commands relating to the game
- Ensures that the rules of the game are followed
- Is in progress until the victory condition for either the innocent/mafia team is achieved.
- Mafia team is assigned randomly.
- Each player is informed of their role privately where the mafia players are informed who the other mafia players are
- Game scales between 6 and 16 players
- Users cannot join a game once it is in session

Server Design

- Contains a Game and IDatabase reference
- Contains ClientHandler listener threads
- Event-Driven
- Relaying Messages to Game and Database classes or handled within the Server
- Produces ServerMessages which are sent to all clients or to specified clients

Client Design

- Connects to GUI.
- Passes messages and commands to server.
- Receives messages from server.



GUI Design

- Single Frame - 5 Panels
- Welcome Screen Panel
- Login Screen Panel
- Sign-in Panel
- Game Panel
- ForgottenPassword Panel
- Use of action listeners
- DefaultListModel JLists attached to scroll panes for chat windows

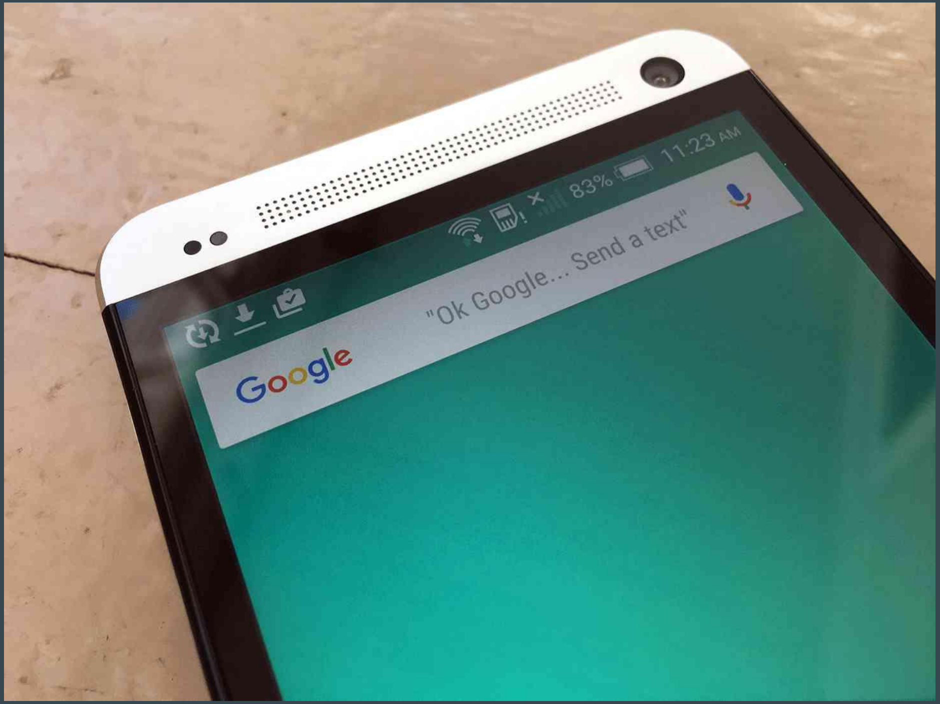
Testing Mafia Game Class

JUnit testing was used to test methods of class
These tests checked:

- That the rules of the game were effectively enforced
- The game could start and end correctly
- Only players could participate in the game once it started
- A player not voting wouldn't prevent the vote from ending

Testing GUI

- Functional Tests
- Layout and Design
- Resolution Issues
- 'Normal' Running Conditions
- Extreme Condition Testing



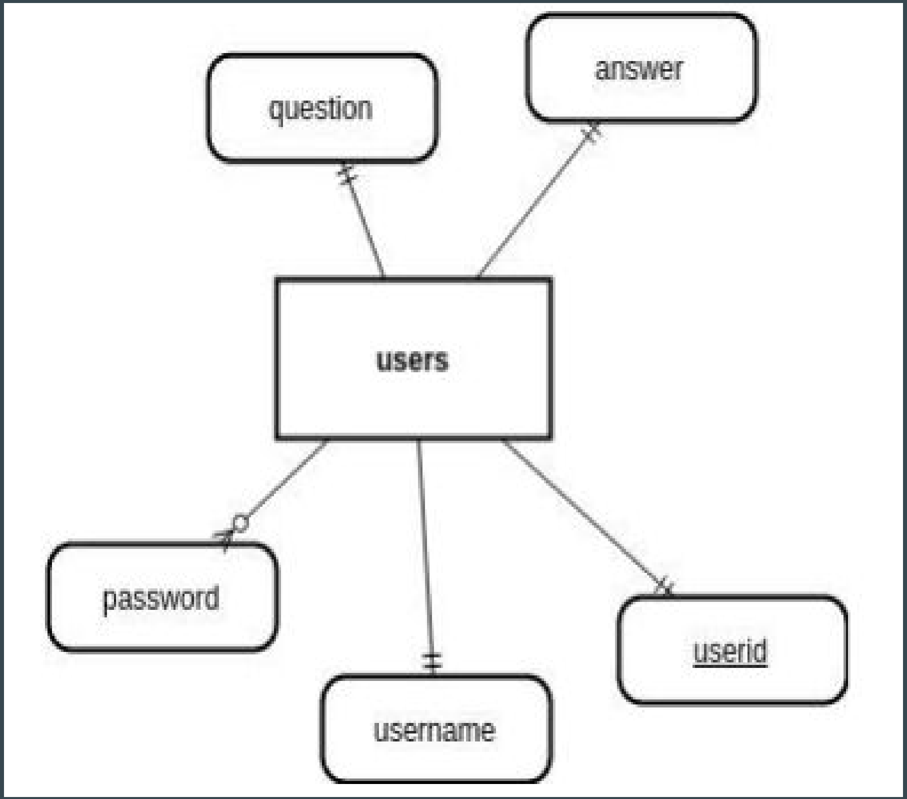
Testing Client-Server

- Basic testing on connection, ability to relay messages when developing the client and the server.
- Simple Functional Tests
    - Use of Stubs
- Simultaneous Client tests
- Beta testing with multiple users.

Testing Database

JUnit Testing for the methods where:

- All functions produced the expected return values
- If the functions produced the correct updates to the database
- The errors that were produced were at the expected time
- Observed the functions working during user input in the GUI
- SQL queries were ran to test reading the table


question
answer
users
password
username
userid