

# Team Nice - Presentation

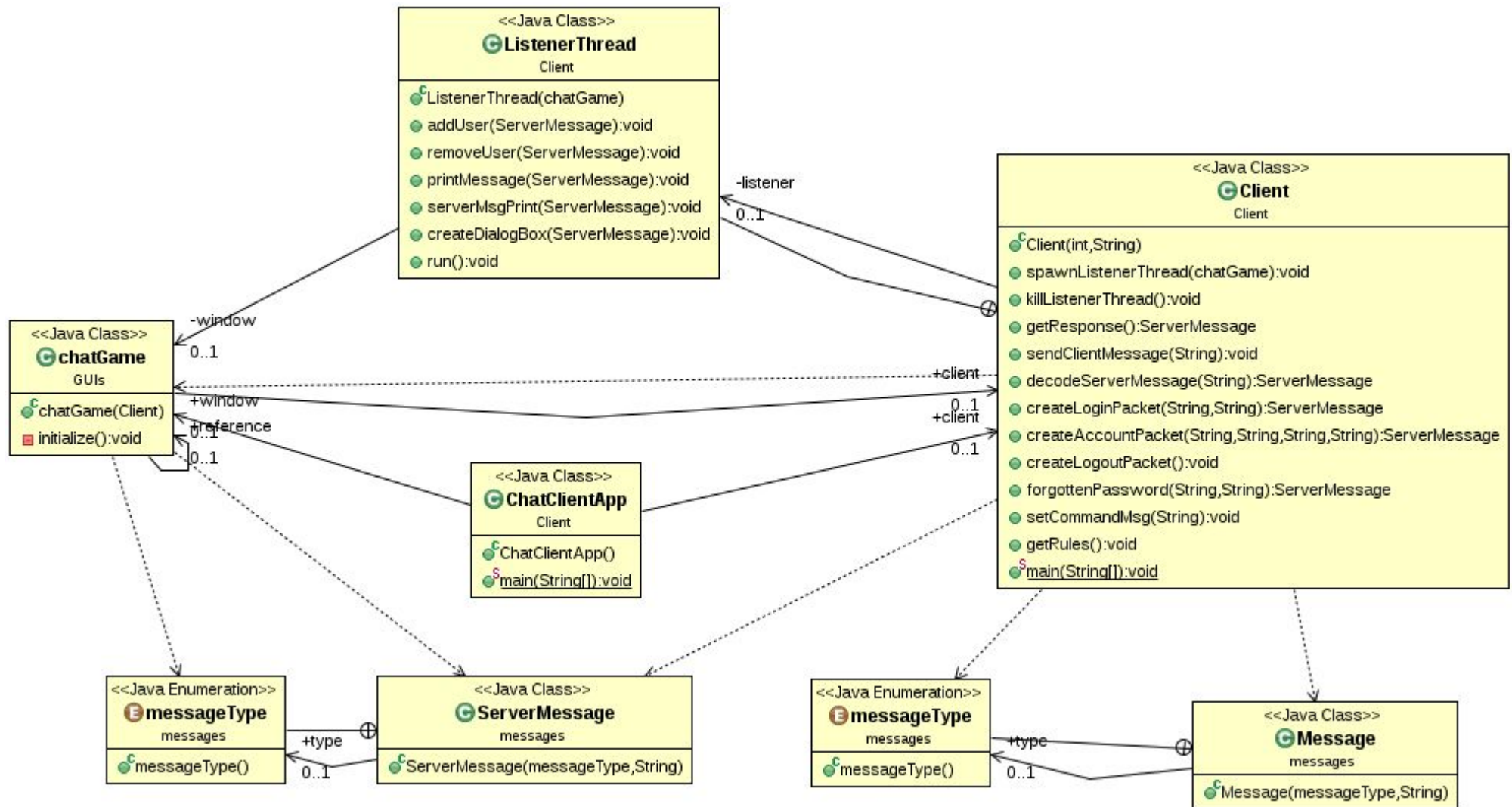


# Project Topic

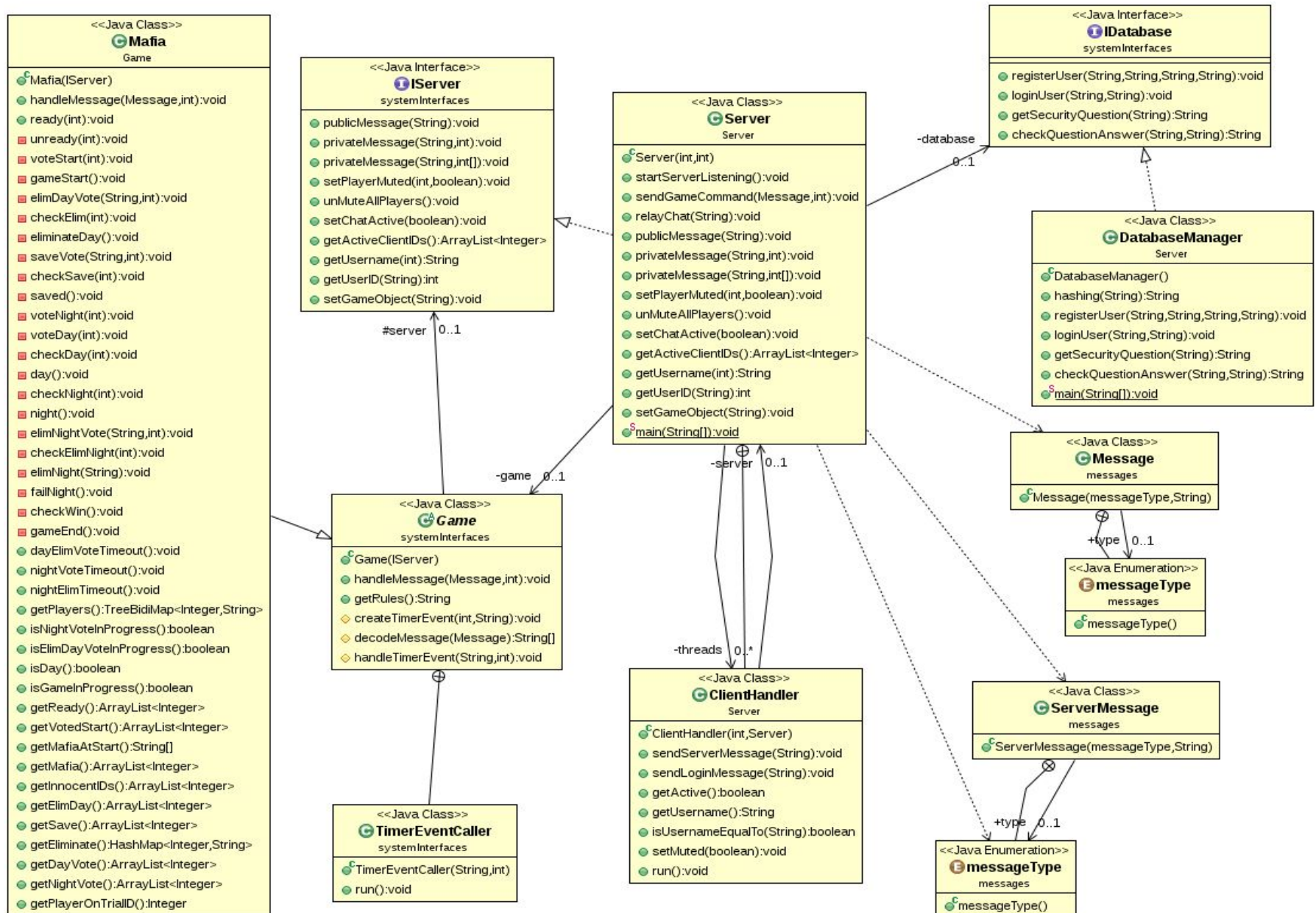


[inthemovie.tumblr.com](http://inthemovie.tumblr.com)

# Client Architecture



# Server Architecture



# 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
- Observation of functions working during user input in the GUI
- 

