**Introduction**

The project supposed to serve users to communicate and send messages through the server. The client connects and communicates with a server and performs activities such as login, registration, send, receives and display messages.

**Presentation Layer**

**Menu Class**

This is the main screen of the program.

**Functionality**

public PrintMenu(); prints the GUI.

**Parser Class**

This class is responsible for insuring that the client entered valid comment and to send the input to the Chatroom class in Business Layer, else prints an error message.

**Attributes**

private MESSAGES\_TO\_DISPAY: int the number of messages to display when user asks to display.

loggedIn: bool the user's state- online or offline

loggedInNickname: string the nickname of the logged in user.

lastRegistered: string[] the group ID and nickname of the last registered user.

enum cmdCheckType: int holds the types of checks the user must pass to finish a request.

enum alertType: int holds all types of alerts can be shown during the program is running.

**Functionality**

public parse(string str); takes relevant information for forwarding process.

private receiveCMD (int type, Logger logger); checks if input is legal and lets the user three opportunities to insert a valid input.

private checkValidity (string str, int type, Logger logger); returns true of false whether the user's input has only valid characters.

private alerts(int type, Logger logger); displays an alert.

**Business Layer**

**Chatroom Class**

This class constitutes as a “User Manager” and as a “Message Manager”.

**Attributes**

private loggedInUser: User holds the link to the current logged in user. null by default.

private url: string server's address.

private messages: List<Message> list of last messages, ordered by timestamp.

private users: List<User> list of users.

private logger: Logger a logger which keeps and traces all errors and messages onto a log file.

private msgHandler: MessageHandler retrieves messages from file-system and keeps new ones for backup.

private usersHandler: UserHandler retrieves users from file-system and keeps new ones for backup.

private MAX\_MESSAGE\_LENGTH: const int the max input length for message.

**Functionality**

public Chatroom(); default constructor.

public registration(string nickname, int gID); generates a new username and adds to file-system.

public login(String nickname, int gID); asks username and password(optional) entered by user, and activates if matches.

public logout(); disconnects from chatroom.

public retrieveMessages(int numOfMessages); retrieves last amount of messages requested by user. Messages are retrieved from server.

public displayLastMsg(int numOfMessages); displays last amount of messages requested by user which are already retrieved.

public displayAllByUser (string nickname, int numOfMessages); displays specified amount of messages sent by a certain user.

public send(string message); collects a message from the user and calls *send* function in user's functionality. Saves the returned Message object into messages list and file-system.

**User Class**

User is an object which interacts with the server.

**Attributes**

private nickname: string describes user's name.

private charLimitPerMessage: int the limit of characters for a message.

private G\_id: int describes user's group ID number.

private numOfMessages: int counts the number of messages sent by user.

**Functionality**

public User(string nickname, int gID); constructs a new user and then saves user details in the file-system.

public send(string message); wraps a received string into an IMessage object and then creates a Message object.

**Message Class**

Message is an object that wraps an entered string by user with details such as user's nickname, time, etc.

**Attributes**

private body: string the message delivered by user.

private nickname: string nickname of the user who sent the message.

private GUID: Guid a unique message ID generated by the server.

private date: DateTime the date the message was published.

**Functionality**

public Message(IMassage message); gets an IMassage object with GUID, timestamp and the message content and finally constructs a new message and stores in the file-system.

public static factory(string message, string url, string g\_id, string nickname, int charLimitPerMsg); the first step in the message's creation factory happens here, this method gets all details about a requested message, if it passes all validity checks, factory asks the constructor to create a new message.

public static checkValidity(string message); validates if the message has maximum of 150 characters.

public toString() prints message with its details.

**Data Access Layer**

**IHandler Interface**

IHandler represents the basic methods which handlers must implement.

**MessageHandler Class**

MessageHandler class's responsibly is to write Messages to the file-system and read from it. Implements IHandler.

**Attributes**

private static List<Object>: Message; a list which includes all messages.

**Functionality**

public save(Object); save the current Message object into the file-system.

public read(); reads from the file-system and returns a list of messages.

**UserHandler Class**

UserHandler class's responsibly is to write Users to the file-system and read from it. Implements IHandler.

**Attributes**

private static List<Object>: User; a list which includes all users.

**Functionality**

public save(Object); save the current User object into the file-system.

public read(); reads from the file-system and returns a list of users.

**ILogger Layer**

**Logger Class**

The Logger class writes to a text file and documents errors, messages about processes, bad logins/registrations.

**Attributes**

private static readonly log: log4net.ILog the origin logger

private static instance : Logger the log object we use to save messages, which uses the original logger.

public static Instance; returns the *instance logger* object.

**Functionality**

public logInfoMessage(string message); events that happen following the system process.

public logWarnMessage(string message); warnings that occur due to user misuse.

public logErrorMessage(string message); errors that appear when system is not working properly.

public logFatalMessage(string message); errors that crash the system.

**Files**

* Messages.bin ; This file includes the history of messages which sent over the CLI to the server.
* Users.bin ; This file holds information about users which registered into the CLI.