CS494 - INTERNETWORKING PROTOCOL

LAB 01: SOCKET PROGRAMMING

Team xx:

- Nguyễn Minh Nhật 1751090
- Huỳnh Minh Quốc Nhật 1751089
- Thái Hoàng Tuấn 1751026

GAME 00: THE MAGICAL WHEEL

I. Gameplay

Our group creates a simple game "The Magic Wheel" with a server as a Referee and N clients (players) (N is defined by the server in advance, 2<= N <= 10). The rules of the game are:

For each player, the player needs to register to the server to join the game and choose the nickname(your name must be between 2 and 10 characters, includes all letters and numbers, no special character are accepted other than "_", and can not be the same with others).

You are given 0 points and the order of players at the beginning. We will provide you a keyword and a hint for that keyword.

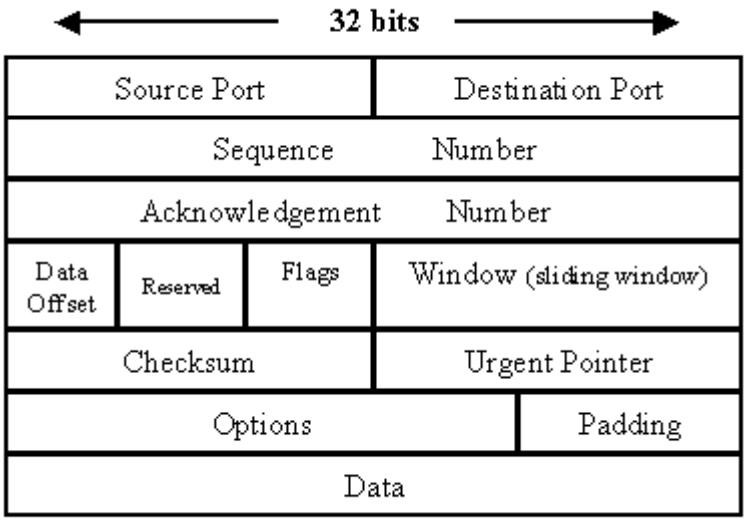
Each turn, you can guess 1 letter for the keywords by fill in the box and press submit. If your answer is correct, you gain 1 points for each letters in the keyword. If not, you pass the turn to other players.

After 3 rounds, you can guess directly for the keyword. If your answer is correct, you get 5 points. If not, you are disqualified.

Finally, we will calculate your scores and rank on the list

II. Structure of packets:

Our game uses Server-Client model. The Server and Client will connect and communication using Sockets through TCP connection.



ServerConnection Class

```
private ServerSocket server;
private int PORT;
private int numberOfPlayers;
private static ArrayList<PlayerHandler> listPlayer;
private ExecutorService pool;
private ArrayList<QuestionHandler> listQuestion;
private int time;
private Timer timer;
private boolean isFinish;
```

```
public ServerConnection() {
    initComponents();
    listPlayer = new ArrayList<PlayerHandler>();
    loadData();
    initTime();
}

/** This method is called from within the constructor to initialize the form ... 5 lines */
@SuppressWarnings("unchecked")
Generated Code

private void btnListenActionPerformed(java.awt.event.ActionEvent evt) { ... 63 lines }

private void txtNumberOfPlayersKeyReleased(java.awt.event.KeyEvent evt) { ... 18 lines }

private void btnStopActionPerformed(java.awt.event.ActionEvent evt) { ... 15 lines }

private void btnNewGameActionPerformed(java.awt.event.ActionEvent evt) { ... 39 lines }

private void cbbTimeActionPerformed(java.awt.event.ActionEvent evt) { ... 4 lines }
```

```
//get random number
private int getRandomNumber(int max) { ... 4 lines }

/** ... 3 lines */
public static void main(String args[]) { ... 31 lines }
```

```
public void setIsFinish(boolean b) { ... 3 lines }

public boolean getIsFinish() { ... 3 lines }

private void initTime() { ... 5 lines }

private void sendQuestionToAll() { ... 14 lines }

public void countDown(int time) { ... 21 lines }

public void alignTurn() { ... 52 lines }
//Kiểm tra trùng tên function

private void readyForNewGame() { ... 5 lines }

private boolean checkDuplicate(String name) { ... 11 lines }

//load data từ database.txt
private void loadData() { ... 19 lines }

//truyền set listPlayer cho từng player
private void updateListPlayerForEachPlayer() { ... 5 lines }

//truyền list of player cho client function
private void sendListPlayer() { ... 15 lines }

//display score board
public void printScoreBoard() { ... 8 lines }

//display notice board
private void printNotice(String msg) { ... 5 lines }
```

ClientConnection Class

```
private final String HOST = "127.0.0.1";
private int PORT;
private ArrayList<Player> listPlayer;
private BufferedReader in;
private PrintWriter out;
private Socket client;
private String name;

private String keyword;
private String hint;
private int length;
private int intTurn;
private Timer timerM;
private Timer timerO;

private boolean isMyTurn;
private int time;

/**
 * Creates new form ClientConnection
 */
public ClientConnection() {
    initComponents();
    listPlayer = new ArrayList<Player>();
    txtGuessKey.setText("");
    setTurn(false);

}
```

```
private void txtNameKeyReleased(java.awt.event.KeyEvent evt) { ... 17 lines }
private void btnConnectActionPerformed(java.awt.event.ActionEvent evt) { ... 34 lines }
private void btnExitActionPerformed(java.awt.event.ActionEvent evt) { ... 11 lines }
private void btnSubmitActionPerformed(java.awt.event.ActionEvent evt) { ... 4 lines }
```

```
private void stopTimer() { ... 11 lines }

private void submitAction() { ... 8 lines }

private void txtGuessCharKeyReleased(java.awt.event.KeyEvent evt) { ... 13 lines }

private void txtGuessKeyKeyReleased(java.awt.event.KeyEvent evt) { ... 5 lines }

private void recieveListPlayer() { ... 23 lines }

private void recieveQuestion() { ... 24 lines }

private void recieveCurrentTurn() { ... 19 lines }

private void setTurn(boolean status) { ... 10 lines }

private void recieveDialog() { ... 16 lines }

private void recieveNotice() { ... 23 lines }

private void recieveBlurKeyword() { ... 16 lines }

private void recieveScore() { ... 21 lines }

private void printQuestion() { ... 5 lines }

private void updateKeyword() { ... 3 lines }

private void printScoreBoard() { ... 8 lines }

private void mainCountDown() { ... 25 lines }

private void ortherCountDown() { ... 22 lines }
```

```
    //display notice board
    private void printNotice(String msg) { ... 5 lines }

    //lắng nghe response từ server, luôn chạy, router để thực hiệ
    private void listenResponse() { ... 27 lines }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        Look and feel setting code (optional)

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ClientConnection().setVisible(true);
            }
        });
    }
```

PlayerHandlerClass

```
private String name;
private int score;
private Socket player;
private BufferedReader in;
private PrintWriter out;
private ArrayList<PlayerHandler> listPlayer;
private QuestionHandler question;
private String gChar;
private String gKey;
private ServerConnection serverConn;
private boolean isDisqualified;
private boolean isTurn;
private boolean isSubmit;

public PlayerHandler(String name, Socket playerSocket, BufferedReader in, PrintWriter out, ServerConnection server)
```

```
public boolean getIsTurn() { ... 3 lines }

public boolean getIsSubmit() { ... 3 lines }

public void setIsSubmit(boolean isSubmit) { ... 3 lines }

public void setIsTurn(boolean turn) { ... 3 lines }

public void setQuestion(QuestionHandler question) { ... 3 lines }

public void setDisqualified(boolean dis) { ... 3 lines }

public boolean getDisqualified() { ... 3 lines }

public QuestionHandler getQuestion() { ... 3 lines }

public String getName() { ... 3 lines }

public void setScore(int score) { ... 3 lines }

public int getScore() { ... 3 lines }

public PrintWriter getOut() { ... 3 lines }

public void setListPlayer(ArrayList<PlayerHandler> listPlayer) { ... 3 lines }

public void sendInfoToAll() { ... 6 lines }

public void close() { ... 9 lines }

public void sendKeyword() { ... 3 lines }
```

```
public void sendDescription() { ... 3 lines }

public void sendLengthOfKeyword() { ... 3 lines }

private boolean guessChar() { ... 11 lines }

private boolean guessKey() { ... 8 lines }

public void sendNotice(String msg) { ... 5 lines }

public void sendScoreToAll() { ... 10 lines }

public void sendTurnToAll(String name) { ... 7 lines }

public void sendNoticeToAll(String msg) { ... 7 lines }

public void sendBlurKeyToAll(boolean isFinish) { ... 12 lines }

public void sendDialogToAll(String msg) { ... 7 lines }

private void addGuessCharSeq(String ch) { ... 8 lines }

private void listenRequest() { ... 91 lines }

@Override
public void run() {
    //listen to client
    listenRequest();
}
```

QuestionHandleClass

```
private String keyword;
private String description;
private String blurKeyword;
private String guessCharSeq;

public QuestionHandler(String keyword, String description) {
    this.keyword = keyword.trim();
    this.description = description.trim();
    this.blurKeyword = "";
    makeBlurKeyword('@');
    guessCharSeq = "";
}
```

```
public String getKeyword() { ... 3 lines }

public String getDescription() { ... 3 lines }

public int getLengthOfKeyword() { ... 3 lines }

public String getBlurKeyword() { ... 3 lines }

public void makeBlurKeyword(char guessChar) { ... 30 lines }

private boolean checkGuessCharSeq(char ch) { ... 10 lines }

public void addGuessCharSeq(char ch) { ... 3 lines }

public int guessChar(String guessChar) { ... 24 lines }

public boolean guessKey(String key) { ... 7 lines }
```

Connection between Server and Client

*SERVER

```
new Thread(() -> {

    try {
        server = new ServerSocket(PORT);
        JOptionPane.showMessageDialog(this, "Listening on port " + PORT);
        btnListen.setEnabled(false);

        while (true) {

            Socket player = server.accept();
            BufferedReader in = new BufferedReader(new InputStreamReader(player.getInputStream()));
            PrintWriter out = new PrintWriter(player.getOutputStream(), true);

            String name = in.readLine();

            if (numberOfPlayers == listPlayer.size()) {
                out.println("The room is out of slot");
                player.close();
                in.close();
                out.close();
            } else if (checkDuplicate(name)) { // Kiểm tra trùng tên

                PlayerHandler playerThread = new PlayerHandler(name, player, in, out, this);
                listPlayer.add(playerThread);
                printNotice(name + " was involved");
```

```
            } else if (checkDuplicate(name)) { // Kiểm tra trùng tên

                PlayerHandler playerThread = new PlayerHandler(name, player, in, out, this);
                listPlayer.add(playerThread);
                printNotice(name + " was involved");

                //kiểm tra nếu đủ người thì enabled NewGame Button
                if (numberOfPlayers == listPlayer.size()) {
                    btnNewGame.setEnabled(true);
                    printNotice("Already have enough players, can get started now!!");
                }

                printScoreBoard();
                //gửi thoong báo đăng ký thành công và gửi cho client danh sách Players
                out.println("Registration Completed Successfully");
                sendListPlayer();

                pool.execute(playerThread);
            } else {
                out.println("This name is already in use");
                player.close();
                in.close();
                out.close();
            }
        }

    } catch (IOException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "I/O ERROR: " + ex.getMessage());

    }
}).start();
```

***CLIENT**

```
try {

    client = new Socket(HOST, PORT);
    in = new BufferedReader(new InputStreamReader(client.getInputStream()));
    out = new PrintWriter(client.getOutputStream(), true);

    this.name = txtName.getText();
    out.println(this.name);

    String response = in.readLine();
    if (response.contains("This name is already in use") || response.contains("The room is out of slot")) {
        txtName.setText("");
        client.close();
        in.close();
        out.close();
    } else {
        btnConnect.setEnabled(false);
        new Thread(() -> {
            listenResponse();
        }).start();
    }
    JOptionPane.showMessageDialog(this, response);

} catch (IOException ex) {
    ex.printStackTrace();
    JOptionPane.showMessageDialog(this, "I/O ERROR: " + ex.getMessage());
}
```

III. **Evaluation**:

1. Team contribution:

No.	Name	Student ID	Contribution (%)
1	Nguyễn Minh Nhật	1751090	30
2	Huỳnh Minh Quốc Nhật	1751089	40
3	Thái Hoàng Tuấn	1751026	30

2. Score Sheet:

No.	Requirements	Score	Evaluate
1	Use C/C++, Java, C#	2	2
2	Implement whole gameplay properly	3	3

3	Socket Non-blocking	2	2
4	Have a good GUI (MFC, WPF, Swing, etc.)	3	2.5
	Total	**10**	9.5