Todor Tsekov, George Vasileiadis, Jiefan Lin

Group 6 |

Design Document(version2)

Trivia

Table of Contents

[1. Architecture diagram 2](#_Toc436911016)

[2. Interface and methods descriptions 2](#_Toc436911017)

[IGame 2](#_Toc436911018)

[IChat 3](#_Toc436911019)

[IGameCallback and IChatcallback 3](#_Toc436911020)

[2.1 Callbacks/Events 4](#_Toc436911021)

[3. Class diagram 4](#_Toc436911022)

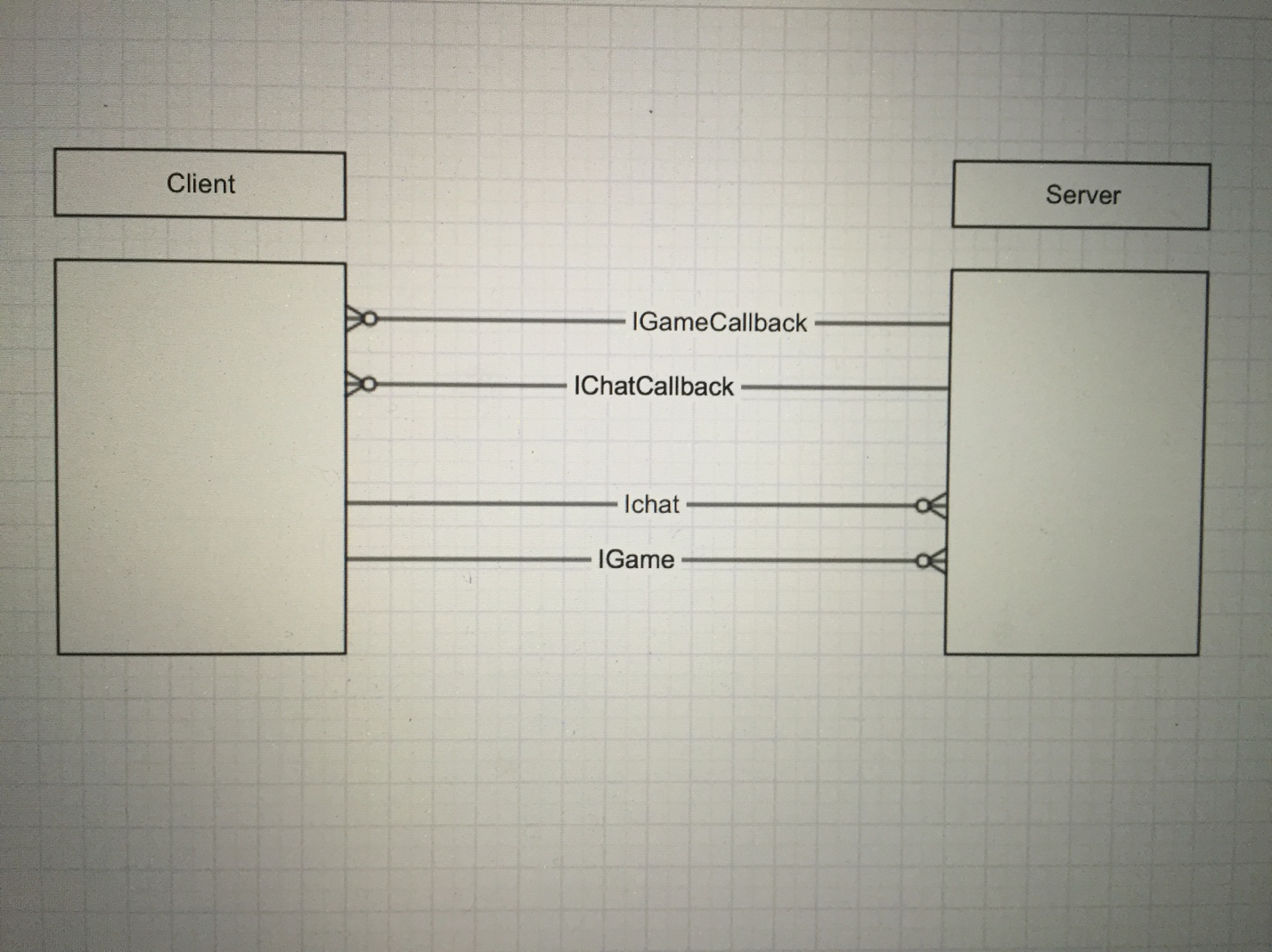
[4. Sequence diagrams 1](#_Toc436911023)

[4.1 Start game 1](#_Toc436911024)

[4.2 Answer Question 2](#_Toc436911025)

[4.3 Win/Lose/Draw 3](#_Toc436911026)

# Architecture diagram



# Interface and methods descriptions

### IGame

This interface implements the various functions for the game.

/// <summary>

/// Sets the id of each client.

/// </summary>

/// <returns>The id of the player.</returns>

[OperationContract]

int setId();

/// <summary>

/// It starts the game after the check is complete.

/// </summary>

/// <param name="player1">Id of the 1st player.</param>

/// <param name="player2">Id of the 2nd player.</param>

[OperationContract]

void startGame(int player1, int player2);

/// <summary>

/// Gets the result to the player.

/// </summary>

/// <param name="player\_id">The player's id requesting the result.</param>

[OperationContract]

void getResult(int player\_id);

/// <summary>

/// It sends a question to a player.

/// </summary>

/// <param name="counter">The id of the question.</param>

/// <param name="player\_id">The id of the player asking.</param>

/// <returns>The question with the possible answers.</returns>

[OperationContract]

Question getQuestion(int counter, int player\_id);

/// <summary>

/// It sets if a player is ready. He is added to player list.

/// </summary>

/// <param name="playerId">The id of the player.</param>

[OperationContract]

void setReady(int playerId);

/// <summary>

/// It receives the answer the player has supplied.

/// </summary>

/// <param name="playerId">The id of the player that gave the answer.</param>

/// <param name="questionId">The id of the question.</param>

/// <param name="answer">The id of question's answer.</param>

[OperationContract]

void setAnswer(int playerId, int questionId, int answer);

/// <summary>

/// It indicates that a player wants to leave the game.

/// </summary>

/// <param name="playerId">The id of the player.</param>

[OperationContract]

void leave(int playerId);

/// <summary>

/// It indicates that a player wants to leave the game.

/// </summary>

/// <param name="playerId">The id of the player.</param>

[OperationContract]

void restart(int playerId);

### IChat

This interface implements the chat.

/// <summary>

/// It sends a player's message to the server.

/// </summary>

/// <param name="player\_id">The player that send the message.</param>

/// <param name="message">The message that was sent.</param>

void sendMessage(int player\_id, string message);

### IGameCallback and IChatcallback

These interfaces are implemented in the client and have basic functionality.

interface IGameCallback

{

/// <summary>

/// It sets the unique id on the client.

/// </summary>

[OperationContract(IsOneWay = true)]

void setId();

/// <summary>

/// It notifies the client that the game starts.

/// </summary>

[OperationContract(IsOneWay = true)]

void startGameInClient();

}

interface IChatCallback

{

/// <summary>

/// It receives a message from the chat.

/// </summary>

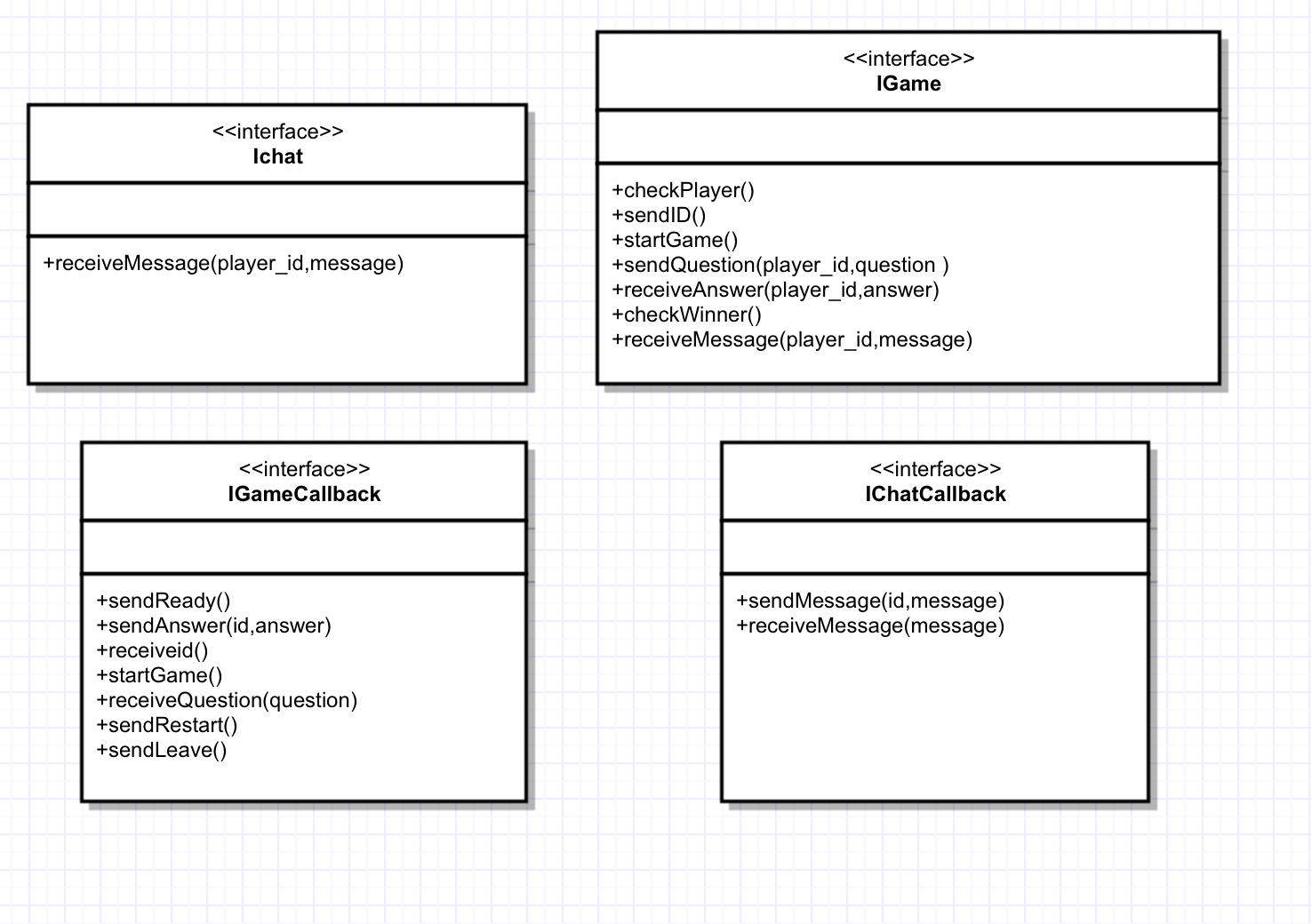
/// <param name="id">The id of the player that sent the message.</param>

/// <param name="message">The body of the message.</param>

void getMessage(int id, string message);

}

## Callbacks/Events

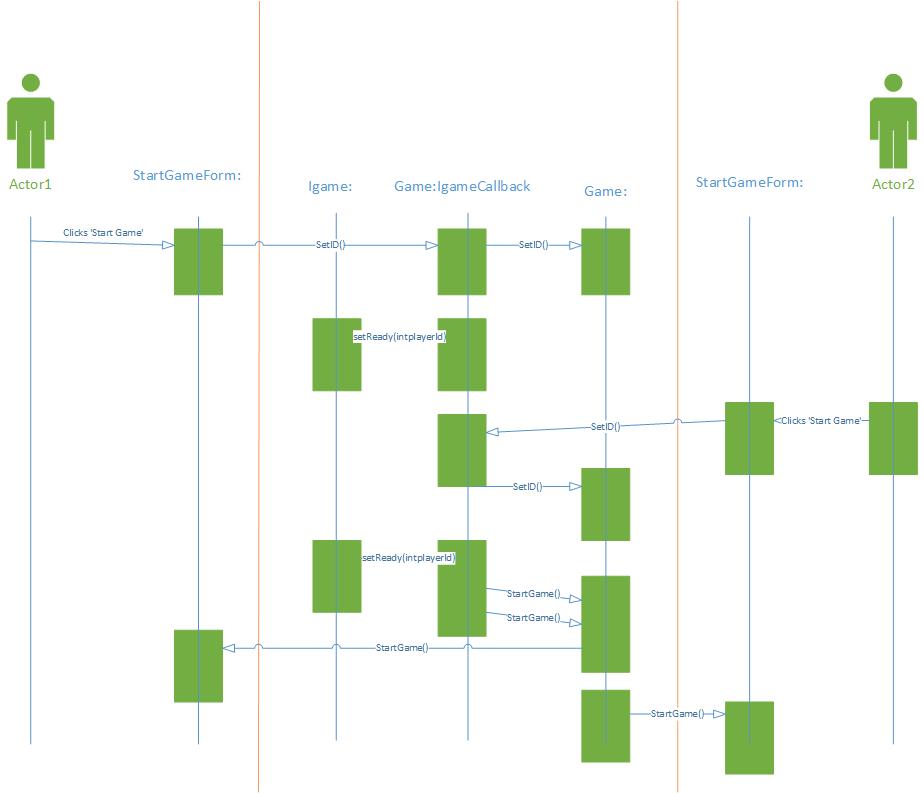


# Class diagram

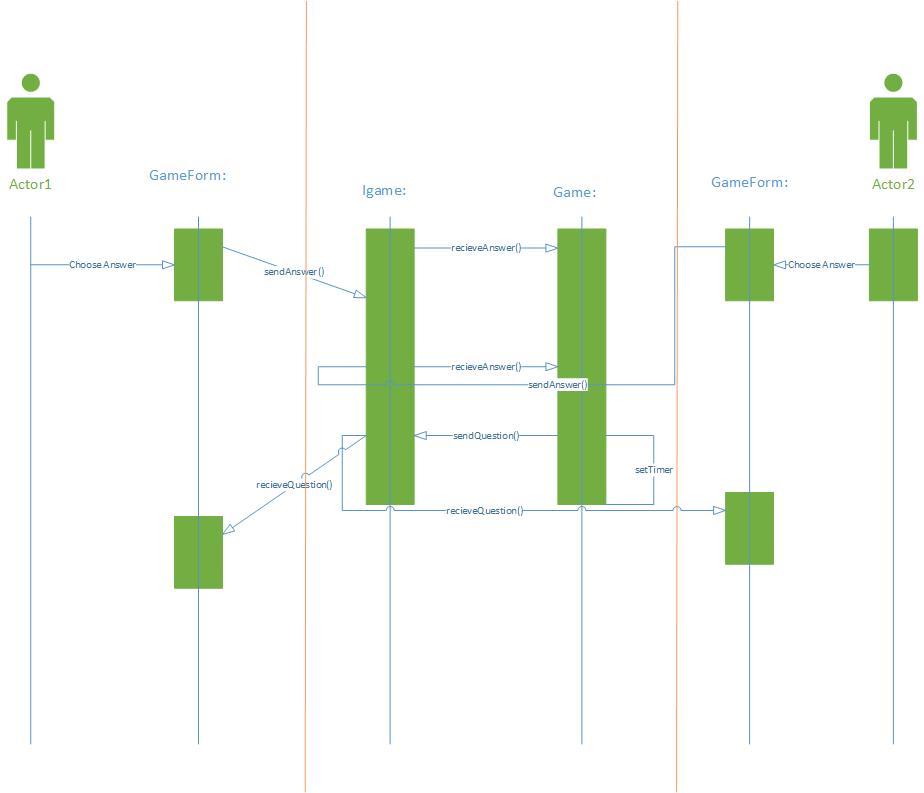


# Sequence diagrams

## 4.1 Start game



## 4.2 Answer Question



## 4.3 Win/Lose/Draw

