Todor Tsekov, George Vasileiadis, Jiefan Lin

Group 6 |

Design Document

Trivia

Table of Contents

[1. Architecture diagram 2](#_Toc436907738)

[2. Interface description 2](#_Toc436907739)

[2.1 Methods 2](#_Toc436907740)

[2.2 Callbacks/Events 3](#_Toc436907741)

[3. Class diagram 3](#_Toc436907742)

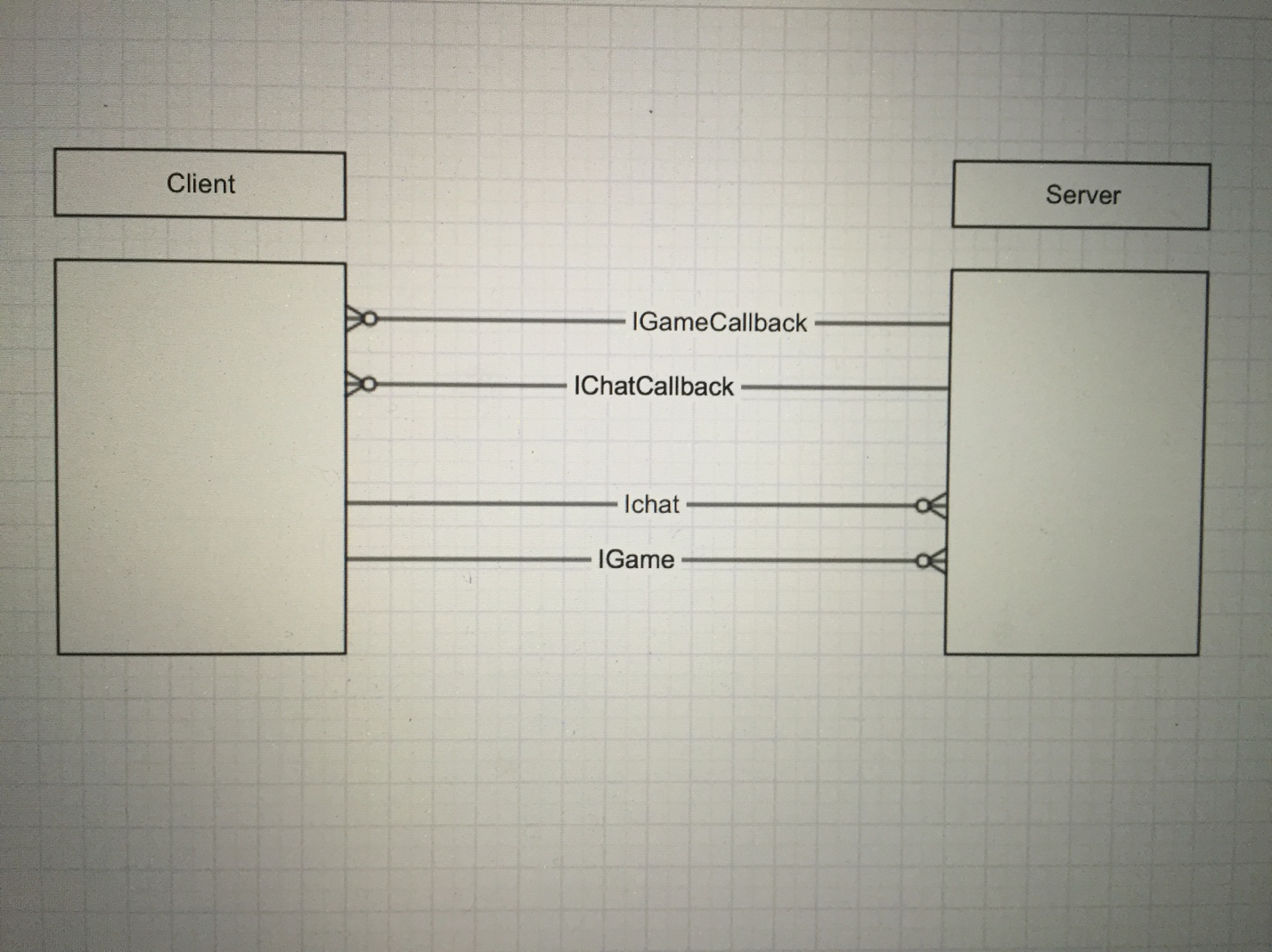
[4. Sequence diagrams 1](#_Toc436907743)

[4.1 Start game 1](#_Toc436907744)

[4.2 Answer Question 2](#_Toc436907745)

[4.3 Win/Lose/Draw 3](#_Toc436907746)

# Architecture diagram



# Interface and methods descriptions

### IGame

This interface implements the various functions for the game.

/// <summary>

/// It starts the game after the check is complete. The check of players is done here.

/// </summary>

[OperationContract]

void startGame();

/// <summary>

/// It sends a question to a player.

/// </summary>

/// <returns>The next question to be asked.</returns>

[OperationContract]

Question getQuestion();

/// <summary>

/// It sets that a player is ready to start a game.

/// </summary>

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

[OperationContract]

void setReady(int playerId);

/// <summary>

/// It sends the answer of a player to the server. The check of the answer is done here.

/// </summary>

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

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

[OperationContract]

void setAnswer(int playerId, 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>

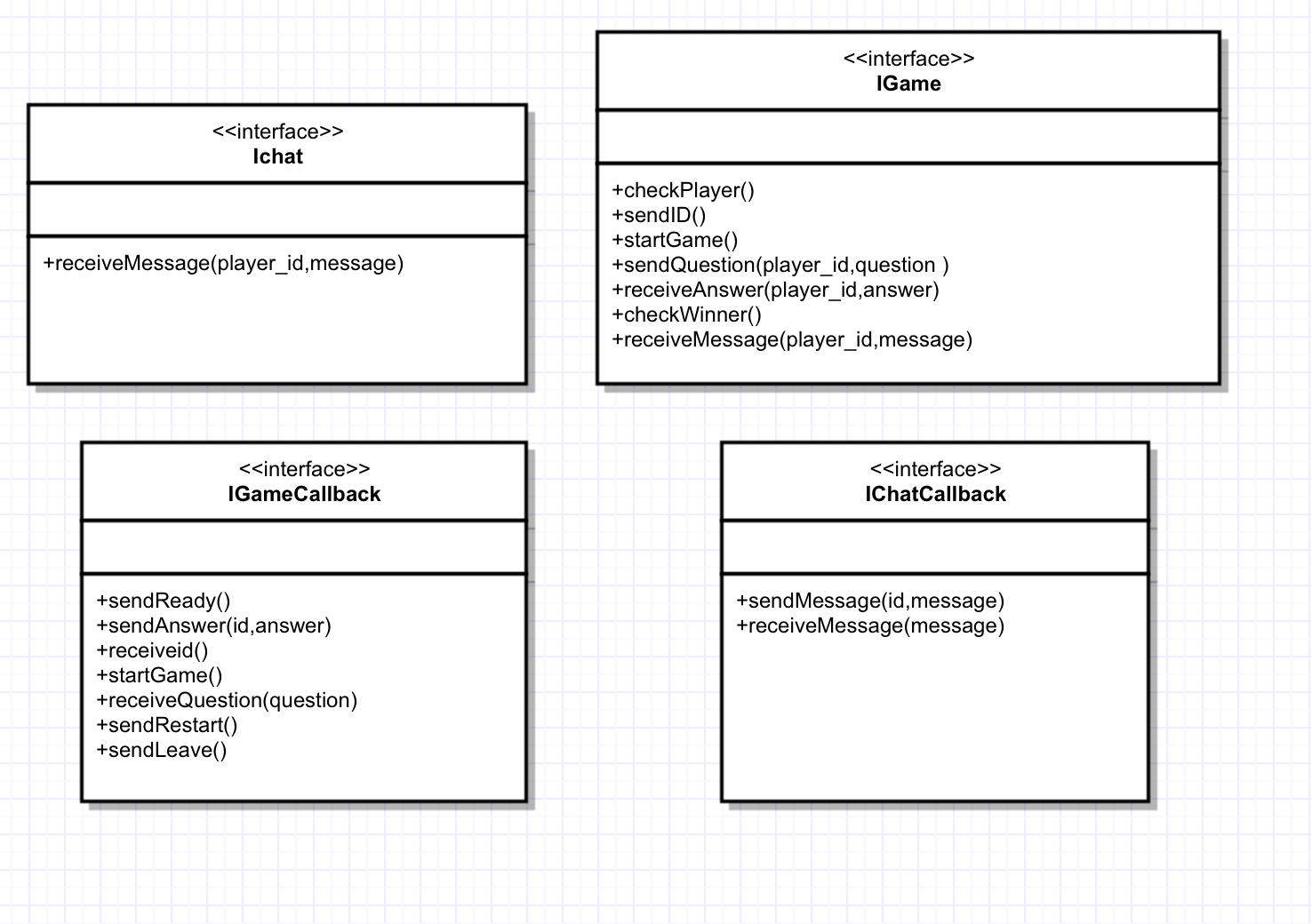
/// <param name="id">The id of the player that sent the message.</param>

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

void getMessage(int id, string message);

}

## Callbacks/Events

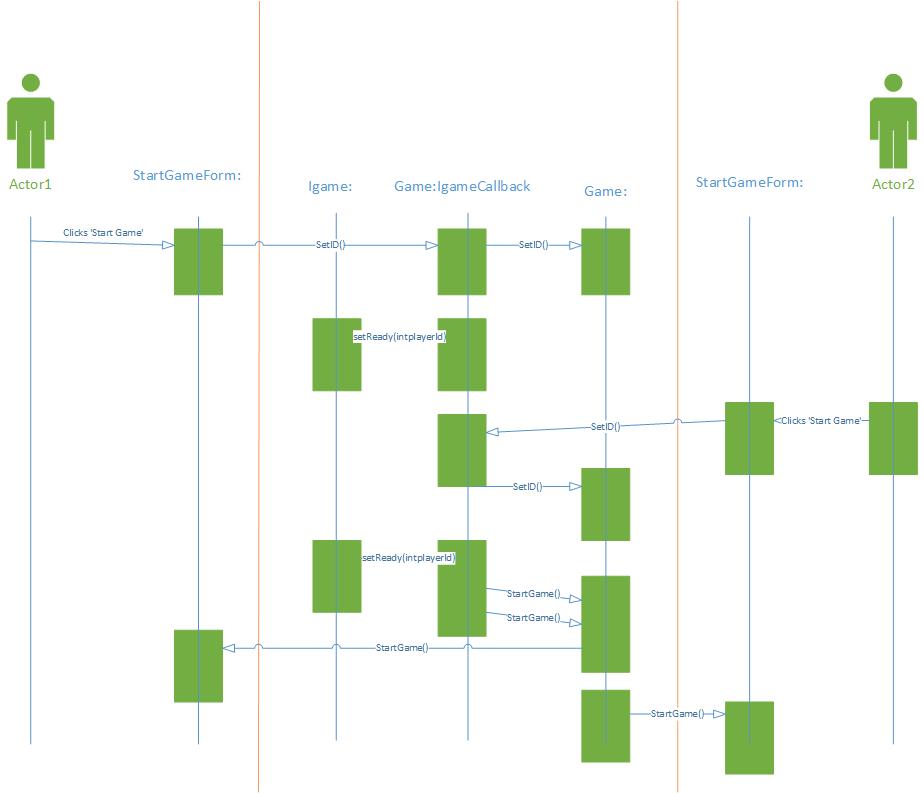


# Class diagram

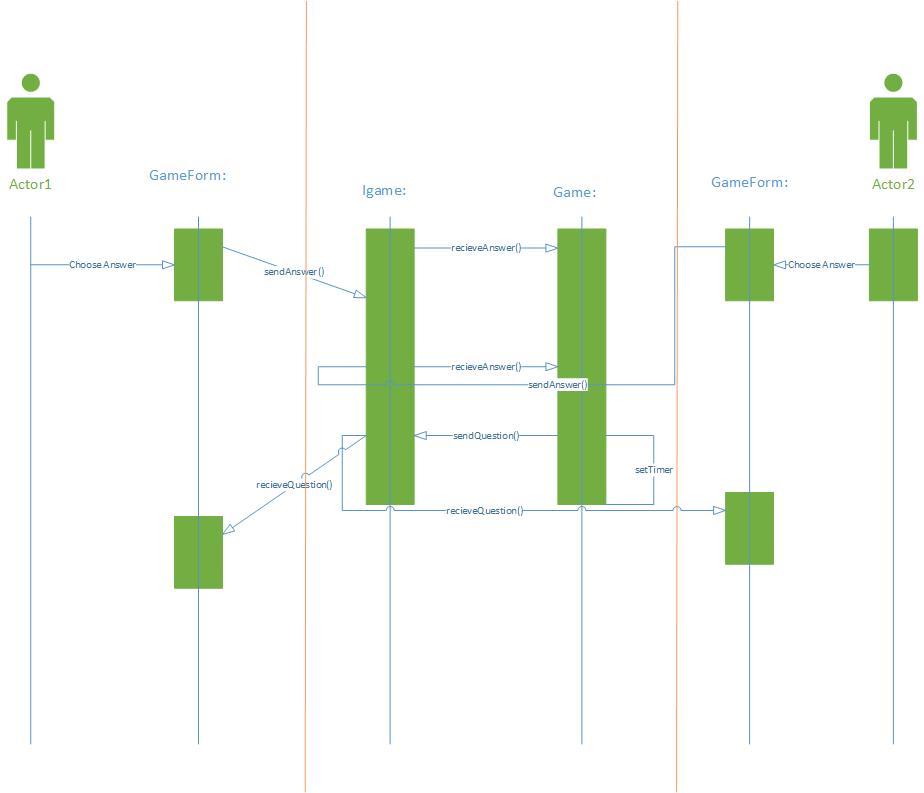


# Sequence diagrams

## 4.1 Start game



## 4.2 Answer Question



## 4.3 Win/Lose/Draw

