Multithreaded Rock-Paper-Scissors, Ömer Halit Cinar

Generated by Doxygen 1.10.0

1 Class Index 1

1.1 Class List . . . 1

2 File Index 3

2.1 File List . . . 3

3 Class Documentation 5

3.1 Client Class Reference . . . 5

3.1.1 Detailed Description . . . 5

3.1.2 Constructor & Destructor Documentation . . . 6

3.1.2.1 Client() . . . 6

3.1.3 Member Function Documentation . . . 6

3.1.3.1 connectToServer() . . . 6

3.1.3.2 playGame() . . . 6

3.1.3.3 start() . . . 6

3.1.4 Member Data Documentation . . . 6

3.1.4.1 clientSocket . . . 6

3.1.4.2 port . . . 7

3.1.4.3 serverIP . . . 7

3.2 HighScoreBoard Class Reference . . . 7

3.2.1 Detailed Description . . . 7

3.2.2 Member Function Documentation . . . 8

3.2.2.1 getHighScores() . . . 8

3.2.2.2 printHighScores() . . . 8

3.2.2.3 updateHighScore() . . . 8

3.2.3 Member Data Documentation . . . 8

3.2.3.1 highScoreMutex . . . 8

3.2.3.2 highScores . . . 8

3.3 Server Class Reference . . . 9

3.3.1 Detailed Description . . . 9

3.3.2 Constructor & Destructor Documentation . . . 9

3.3.2.1 Server() . . . 9

3.3.3 Member Function Documentation . . . 10

3.3.3.1 handleClient() . . . 10

3.3.3.2 playGame() . . . 10

3.3.3.3 start() . . . 10

3.3.4 Member Data Documentation . . . 10

3.3.4.1 highScoreBoard . . . 10

3.3.4.2 serverSocket . . . 10

4 File Documentation 11

4.1 client/client.cpp File Reference . . . 11

4.1.1 Macro Definition Documentation . . . 11

4.1.1.1 BUFFER_SIZE 11
4.2 client/client.h File Reference 11
4.2.1 Detailed Description 12
4.3 client.h 12
4.4 client/main.cpp File Reference 12
4.4.1 Function Documentation 12
4.4.1.1 main() 12
4.5 server/main.cpp File Reference 12
4.5.1 Function Documentation 13
4.5.1.1 main() 13
4.6 server/game_logic.cpp File Reference 13
4.6.1 Function Documentation 13
4.6.1.1 determineWinner() 13
4.7 server/game_logic.h File Reference 13
4.7.1 Detailed Description 14
4.7.2 Function Documentation 14
4.7.2.1 determineWinner() 14
4.8 game_logic.h 14
4.9 server/high_score_board.cpp File Reference 15
4.10 server/high_score_board.h File Reference 15
4.10.1 Detailed Description 15
4.11 high_score_board.h 15
4.12 server/server.cpp File Reference 16
4.12.1 Macro Definition Documentation 16
4.12.1.1 BUFFER_SIZE 16
4.13 server/server.h File Reference 16
4.13.1 Detailed Description 16
4.14 server.h 17
4.15 unit_test/test_game_logic.cpp File Reference 17
4.15.1 Detailed Description 17
4.15.2 Function Documentation 18
4.15.2.1 TEST() [1/4] 18
4.15.2.2 TEST() [2/4] 18
4.15.2.3 TEST() [3/4] 18
4.15.2.4 TEST() [4/4] 18
4.16 unit_test/test_high_score_board.cpp File Reference 18
4.16.1 Detailed Description 19
4.16.2 Function Documentation 19
4.16.2.1 TEST() [1/3] 19
4.16.2.2 TEST() [2/3] 19
4.16.2.3 TEST() [3/3] 19

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Client
Manages the client-side operations of the game 5
HighScoreBoard
Manages the high scores for the game 7
Server
Manages the server-side operations of the game 9

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/client/client.cpp 11

C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/client/client.h
Client class header file 11

C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/client/main.cpp 12

C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/server/game_logic.cpp . . . 13

C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/server/game_logic.h
Game logic header file 13

C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/server/high_score_board.cpp
15

C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/server/high_score_board.h
High Score Board class header file 15

C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/server/main.cpp 12

C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/server/server.cpp 16

C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/server/server.h
Server class header file 16

C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/unit_test/test_game_logic.cpp
Tests for game logic functions 17

C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/unit_test/test_high_score_board.cpp
Tests for the HighScoreBoard functionality 18

Chapter 3

Class Documentation

3.1 Client Class Reference

Manages the client-side operations of the game.

```
#include <client.h>
```

Public Member Functions

- Client (const std::string &serverIP, int port)
  Constructs a new Client object.
- void start ()
  Starts the client.

Private Member Functions

- void connectToServer ()
  Connects to the server.
- std::string playGame ()
  Plays the game by choosing rock, paper, or scissors.

Private Attributes

- std::string serverIP
  The IP address of the server.
- int port
  The port number of the server.
- SOCKET clientSocket
  The client socket.

3.1.1 Detailed Description

Manages the client-side operations of the game.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Client()

```
Client::Client (
              const std::string & serverIP,
              int port )
```

Constructs a new Client object.

Parameters

server↵ IP	The IP address of the server.
port	The port number of the server.

3.1.3 Member Function Documentation

3.1.3.1 connectToServer()

```
void Client::connectToServer ( )  [private]
```

Connects to the server.

3.1.3.2 playGame()

```
std::string Client::playGame ( )  [private]
```

Plays the game by choosing rock, paper, or scissors.

Returns

The chosen option as a string.

3.1.3.3 start()

```
void Client::start ( )
```

Starts the client.

3.1.4 Member Data Documentation

3.1.4.1 clientSocket

```
SOCKET Client::clientSocket  [private]
```

The client socket.

3.1.4.2 port

```
int Client::port  [private]
```

The port number of the server.

3.1.4.3 serverIP

```
std::string Client::serverIP  [private]
```

The IP address of the server.

The documentation for this class was generated from the following files:

- C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/client/client.h
- C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/client/client.cpp

3.2 HighScoreBoard Class Reference

Manages the high scores for the game.

```
#include <high_score_board.h>
```

Public Member Functions

- void updateHighScore (const std::string &name, int score)

  Updates the high score for a given player.
- void printHighScores () const

  Prints the high scores to the console.
- std::vector< std::pair< std::string, int > > getHighScores () const

  Retrieves the high scores.

Private Attributes

- std::vector< std::pair< std::string, int > > highScores

  Vector to store high scores.
- std::mutex highScoreMutex

  Mutex to protect access to high scores.

3.2.1 Detailed Description

Manages the high scores for the game.

3.2.2 Member Function Documentation

3.2.2.1 getHighScores()

```
std::vector< std::pair< std::string, int > > HighScoreBoard::getHighScores ( ) const
```

Retrieves the high scores.

Returns

A vector of pairs containing player names and scores.

3.2.2.2 printHighScores()

```
void HighScoreBoard::printHighScores ( ) const
```

Prints the high scores to the console.

3.2.2.3 updateHighScore()

```
void HighScoreBoard::updateHighScore (
            const std::string & name,
            int score )
```

Updates the high score for a given player.

Parameters

name	The name of the player.
score	The score of the player.

3.2.3 Member Data Documentation

3.2.3.1 highScoreMutex

```
std::mutex HighScoreBoard::highScoreMutex  [mutable], [private]
```

Mutex to protect access to high scores.

3.2.3.2 highScores

```
std::vector<std::pair<std::string, int> > HighScoreBoard::highScores  [private]
```

Vector to store high scores.

The documentation for this class was generated from the following files:

- C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/server/high_score_board.h
- C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/server/high_score_board.cpp

3.3 Server Class Reference

Manages the server-side operations of the game.

```
#include <server.h>
```

Public Member Functions

- Server (int port)

  Constructs a new Server object.
- void start ()

  Starts the server.

Private Member Functions

- void handleClient (SOCKET clientSocket)

  Handles a client connection.
- std::string playGame ()

  Plays the game by choosing rock, paper, or scissors.

Private Attributes

- SOCKET serverSocket

  The server socket.
- HighScoreBoard highScoreBoard

  High score board.

3.3.1 Detailed Description

Manages the server-side operations of the game.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Server()

```
Server::Server (
                int port )
```

Constructs a new Server object.

Parameters

port	The port number for the server.

3.3.3 Member Function Documentation

3.3.3.1 handleClient()

```
void Server::handleClient (
            SOCKET clientSocket )  [private]
```

Handles a client connection.

Parameters

clientSocket	The socket for the connected client.

3.3.3.2 playGame()

```
std::string Server::playGame ( )  [private]
```

Plays the game by choosing rock, paper, or scissors.

Returns

The chosen option as a string.

3.3.3.3 start()

```
void Server::start ( )
```

Starts the server.

3.3.4 Member Data Documentation

3.3.4.1 highScoreBoard

```
HighScoreBoard Server::highScoreBoard  [private]
```

High score board.

3.3.4.2 serverSocket

```
SOCKET Server::serverSocket  [private]
```

The server socket.

The documentation for this class was generated from the following files:

- C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/server/server.h
- C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/server/server.cpp

Chapter 4

File Documentation

4.1 C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/client/client.cpp File Reference

```
#include "client.h"
#include <iostream>
#include <winsock2.h>
#include <ctime>
#include <WS2tcpip.h>
#include <vector>
```

Macros

- #define BUFFER_SIZE 256

4.1.1 Macro Definition Documentation

4.1.1.1 BUFFER_SIZE

```
#define BUFFER_SIZE 256
```

4.2 C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/client/client.h File Reference

Client class header file.

```
#include <string>
#include <winsock2.h>
```

Classes

- class Client

  Manages the client-side operations of the game.

4.2.1 Detailed Description

Client class header file.

4.3 client.h

Go to the documentation of this file.

```
#ifndef CLIENT_H
#define CLIENT_H

#include <string>
#include <winsock2.h>

class Client {
public:
    Client(const std::string& serverIP, int port);

    void start();

private:
    std::string serverIP;
    int port;
    SOCKET clientSocket;

    void connectToServer();

    std::string playGame();
};

#endif
```

4.4 C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/client/main.cpp File Reference

```
#include "client.h"
```

Functions

- int main ()

4.4.1 Function Documentation

4.4.1.1 main()

```
int main ( )
```

4.5 C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/server/main.cpp File Reference

```
#include "server.h"
```

Functions

- int main ()

4.5.1 Function Documentation

4.5.1.1 main()

```
int main ( )
```

4.6 C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/server/game_logic.cpp File Reference

```
#include "game_logic.h"
```

Functions

- std::string determineWinner (const std::string &serverChoice, const std::string &clientChoice)

  Determines the winner of a Rock-Paper-Scissors game.

4.6.1 Function Documentation

4.6.1.1 determineWinner()

```
std::string determineWinner (
             const std::string & serverChoice,
             const std::string & clientChoice )
```

Determines the winner of a Rock-Paper-Scissors game.

Determines the winner between the server's and client's choices.

Parameters

serverChoice	The server's choice.
clientChoice	The client's choice.

Returns

"client" if the client wins, "server" if the server wins, or "draw" if it's a draw.

4.7 C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/server/game_logic.h File Reference

Game logic header file.

```
#include <string>
```

Functions

- std::string determineWinner (const std::string &serverChoice, const std::string &clientChoice)
  Determines the winner between the server's and client's choices.

4.7.1 Detailed Description

Game logic header file.

4.7.2 Function Documentation

4.7.2.1 determineWinner()

```
std::string determineWinner (
            const std::string & serverChoice,
            const std::string & clientChoice )
```

Determines the winner between the server's and client's choices.

Parameters

serverChoice	The server's choice ("rock", "paper", or "scissors").
clientChoice	The client's choice ("rock", "paper", or "scissors").

Returns

"client" if the client wins, "server" if the server wins, or "draw" if it's a draw.

Determines the winner between the server's and client's choices.

Parameters

serverChoice	The server's choice.
clientChoice	The client's choice.

Returns

"client" if the client wins, "server" if the server wins, or "draw" if it's a draw.

4.8 game_logic.h

Go to the documentation of this file.

```
#ifndef GAME_LOGIC_H
#define GAME_LOGIC_H

#include <string>

std::string determineWinner(const std::string& serverChoice, const std::string& clientChoice);

#endif
```

4.9 C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/server/high_score_board.cpp File Reference

```
#include "high_score_board.h"
#include <algorithm>
#include <iostream>
```

4.10 C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/server/high_score_board.h File Reference

High Score Board class header file.

```
#include <vector>
#include <string>
#include <mutex>
```

Classes

- class HighScoreBoard

  Manages the high scores for the game.

4.10.1 Detailed Description

High Score Board class header file.

4.11 high_score_board.h

Go to the documentation of this file.

```

#ifndef HIGH_SCORE_BOARD_H
#define HIGH_SCORE_BOARD_H

#include <vector>
#include <string>
#include <mutex>

class HighScoreBoard {
public:
    void updateHighScore(const std::string& name, int score);

    void printHighScores() const;

    std::vector<std::pair<std::string, int» getHighScores() const;

private:
    std::vector<std::pair<std::string, int» highScores;
    mutable std::mutex highScoreMutex;
};

#endif
```

4.12 C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/server/server.cpp File Reference

```
#include "server.h"
#include "game_logic.h"
#include <iostream>
#include <vector>
#include <cstring>
#include <winsock2.h>
#include <ctime>
```

Macros

- #define BUFFER_SIZE 256

4.12.1 Macro Definition Documentation

4.12.1.1 BUFFER_SIZE

```
#define BUFFER_SIZE 256
```

4.13 C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/server/server.h File Reference

Server class header file.

```
#include "high_score_board.h"
#include <winsock2.h>
#include <thread>
```

Classes

- class Server

  Manages the server-side operations of the game.

4.13.1 Detailed Description

Server class header file.

4.14 server.h

Go to the documentation of this file.

```
#ifndef SERVER_H
#define SERVER_H

#include "high_score_board.h"
#include <winsock2.h>
#include <thread>

class Server {
public:
    Server(int port);

    void start();

private:
    SOCKET serverSocket;
    HighScoreBoard highScoreBoard;

    void handleClient(SOCKET clientSocket);

    std::string playGame();
};

#endif
```

4.15 C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/unit_test/test_game_logic.cpp File Reference

Tests for game logic functions.

```
#include <gtest/gtest.h>
#include "../server/game_logic.h"
```

Functions

- TEST (GameLogicTest, RockBeatsScissors)

  Test case for rock beating scissors.
- TEST (GameLogicTest, ScissorsBeatsPaper)

  Test case for scissors beating paper.
- TEST (GameLogicTest, PaperBeatsRock)

  Test case for paper beating rock.
- TEST (GameLogicTest, SameChoiceDraw)

  Test case for identical choices resulting in a draw.

4.15.1 Detailed Description

Tests for game logic functions.

This file contains tests for the game logic, specifically testing the outcomes of rock-paper-scissors games between a server and a client.

4.15.2 Function Documentation

4.15.2.1 TEST() [1/4]

```
TEST (
            GameLogicTest ,
            PaperBeatsRock  )
```

Test case for paper beating rock.

This test checks that the game logic correctly assigns paper as the winner over rock.

4.15.2.2 TEST() [2/4]

```
TEST (
            GameLogicTest ,
            RockBeatsScissors  )
```

Test case for rock beating scissors.

This test ensures that the game logic correctly identifies rock as the winner over scissors.

4.15.2.3 TEST() [3/4]

```
TEST (
            GameLogicTest ,
            SameChoiceDraw  )
```

Test case for identical choices resulting in a draw.

This test ensures that the game logic identifies games where both players make the same choice as a draw.

4.15.2.4 TEST() [4/4]

```
TEST (
            GameLogicTest ,
            ScissorsBeatsPaper  )
```

Test case for scissors beating paper.

This test verifies that the game logic accurately determines scissors as the winner over paper.

4.16 C:/Users/katka/source/nokia_homework_omer/multithreaded_rps - doxygen/unit_test/test_high_score_board.cpp File Reference

Tests for the HighScoreBoard functionality.

```
#include "high_score_board.h"
#include <gtest/gtest.h>
```

Functions

- TEST (HighScoreBoardTest, AddNewScore)

  Test case for adding new scores to the high score board.
- TEST (HighScoreBoardTest, UpdateExistingScore)

  Test case for updating an existing score on the high score board.
- TEST (HighScoreBoardTest, KeepTop10Scores)

  Test case for maintaining only the top 10 scores on the high score board.

4.16.1 Detailed Description

Tests for the HighScoreBoard functionality.

This file contains tests for the HighScoreBoard class, ensuring that scores are added, updated, and maintained correctly within the top 10 scores.

4.16.2 Function Documentation

4.16.2.1 TEST() [1/3]

```
TEST (
          HighScoreBoardTest ,
          AddNewScore )
```

Test case for adding new scores to the high score board.

This test verifies that new scores are added correctly to the high score board and that the scores are sorted in descending order.

4.16.2.2 TEST() [2/3]

```
TEST (
          HighScoreBoardTest ,
          KeepTop10Scores )
```

Test case for maintaining only the top 10 scores on the high score board.

This test checks that the high score board correctly limits itself to storing only the top 10 scores, even when more than 10 scores are added.

4.16.2.3 TEST() [3/3]

```
TEST (
          HighScoreBoardTest ,
          UpdateExistingScore )
```

Test case for updating an existing score on the high score board.

This test ensures that if a player's new score is higher than their existing score, the high score board is updated accordingly.