|  |
| --- |
|  |
| Valepaska |
| An online multiplayer card game |

|  |
| --- |
| Juho Rantala  1-27-2022 |

Version history

|  |  |  |
| --- | --- | --- |
| Changes | Date | Author |
| Created documentation | 25.1.2022 | Juho Rantala |
| Added Group, Lobby and Game | 27.1.2022 | Juho Rantala |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Contents

[1. Server side 3](#_Toc94138704)

[1.1. Server 4](#_Toc94138705)

[1.1.1. Public Methods 4](#_Toc94138706)

[1.1.2. Protected Methods 4](#_Toc94138707)

[1.1.3. Private Methods 4](#_Toc94138708)

[1.2. EventHandler 5](#_Toc94138709)

[1.2.1. Public Methods 5](#_Toc94138710)

[1.2.2. Protected Methods 5](#_Toc94138711)

[1.2.3. Private Methods 6](#_Toc94138712)

[1.3. Group 7](#_Toc94138713)

[1.3.1. Public Methods 7](#_Toc94138714)

[1.3.2. Protected Methods 7](#_Toc94138715)

[1.3.3. Private Methods 7](#_Toc94138716)

[1.4. Lobby 8](#_Toc94138717)

[1.4.1. Public Methods 8](#_Toc94138718)

[1.4.2. Protected Methods 8](#_Toc94138719)

[1.4.3. Private Methods 8](#_Toc94138720)

[1.5. Game 9](#_Toc94138721)

[1.5.1. Public Methods 9](#_Toc94138722)

[1.5.2. Protected Methods 9](#_Toc94138723)

[1.5.3. Private Methods 9](#_Toc94138724)

[2. Client side 10](#_Toc94138725)

[3. Template 11](#_Toc94138726)

[3.1. <class> 11](#_Toc94138727)

[3.1.1. Public Methods 11](#_Toc94138728)

[3.1.2. Protected Methods 11](#_Toc94138729)

[3.1.3. Private Methods 11](#_Toc94138730)

# Server side

This section covers the documentation for the server side of Valepaska. The code is in **./server/** and the project is defined in **server.pro**.

Server side is responsible to allocate new clients to their lobbies, start and handle their games and removing them after they are finished.

## Server

Class: Server

Files: **server.h** and **server.cpp**

A TCP server which communicates with clients.

This is an abstract class, which is meant to be derived from.

### Public Methods

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| Server(*const* std::string& port = "12345") | Initialize the server. By default, uses port 12345. Creates a listening socket |
| *virtual* ~*Server*() | Closes all sockets |

### Protected Methods

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| bool sendToClient(SOCKET& client, *const* std::string& msg) | Send a string ***msg*** to socket ***client***. Return false if an error occurred |
| bool broadcast(*const* std::string& msg) | Send a string ***msg***to all connected clients. Return false if any failed, but try to send to each client |
| void acceptClients() | Start main loop accepting clients to connect. This loop will run as long as the server is on |
| *virtual* bool *handleEvent*(Event& event)=0 | Pure virtual method to handle event ***event.***  Must be implemented by deriving class |
| *virtual* bool *addClient*(SOCKET client) | Virtual method to add a new client ***client.*** This method will call Server::handle(client) on a detached thread.  Return true if client was added. If the client already existed, return false |
| *virtual* bool *removeClient*(SOCKET client) | Virtual method to remove client ***client***.  If client doesn’t exist return false |

### Private Methods

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| **void closeConnection(SOCKET client)** | Closes connection with client ***client*** |
| **bool hasClient(SOCKET& client)** | Return true if Server has client ***client*** |
| **void stopListen()** | Closes listening socket |
| **void handle(SOCKET client)** | Receive messages from client ***client*** and call Server::handleEvent for each message. Removes client if receive fails |

## EventHandler

Class: EventHandler : protected Server

Files: **eventhandler.h** and **eventhandler.cpp**

A class to handle server events and manage lobbies. Derives from Server.

### Public Methods

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| EventHandler() | Initialize EventHandler. Create latest lobby |
| ~*EventHandler*() | Delete all lobbies |
| void **start**(Lobby\* lobby) | Call EventHandler::toGame if there are enough members in ***lobby*** |

### Protected Methods

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| *virtual* bool *handleEvent*(Event& event) *override* | Passes event to its handler. Return false if event is unknown |
| *virtual* bool *addClient*(SOCKET client) *override* | Overrides Server::addClient. Return false if Server::addClient returns false. Otherwise add ***client*** to latest lobby and create new if needed |
| *virtual* bool *removeClient*(SOCKET client) *override* | Overrides Server::removeClient. Return false if Server::removeClient returns false or if EventHandler already has ***client***. Removes ***client*** from lobby and destroys the lobby if it’s empty. |

### Private Methods

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| bool sendEvent(Event& event) | Return false if event command is unknown.  Send ***event*** to the client specified in ***event***. |
| void broadcast(Event& event) | Send ***event*** to all clients. |
| bool isHandler(command command) | Return true if command is valid to handle |
| bool isGenerator(command command) | Return true if command is valid to send |
| void createNewLobby() | Creates new empty lobby |
| Group \*getGroupByClient(SOCKET client) | Return pointer to the Group where ***client*** is. If not found, return nullptr |
| std::vector<SOCKET> getClientsByGroup(Group\* group) | Return ***group***’s clients |
| bool hasClient(SOCKET client) | Return true if EventHandler has ***client*** |
| id changeForClient(SOCKET client, id player) | Return player reference to ***player*** according to ***client*** |

## Group

Class: Group

Files: **group.h** and **group.cpp**

A class to handle a group of members

### Public Methods

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| *virtual* void *transferTo*(Group\* other) | Add all members to other group |
| int size() | Return group size |

### Protected Methods

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| *virtual* bool *add*(id member) | Adds ***member*** if not already in group. Return false if already in group |
| *virtual* bool *remove*(id member) | Removes ***member*** if in group. Return false if not in group |
| *virtual* bool *hasMember*(id member) | Return true if ***member*** is in group |

### Private Methods

|  |  |
| --- | --- |
| **Method** | **Explanation** |

## Lobby

Class: Lobby : public Group

Files: **lobby.h** and **lobby.cpp**

A group that keeps track on ready-state of members

### Public Methods

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| Lobby(EventHandler\* eventHandler) | Link ***eventhandler*** |
| void setState(id member, bool isReady) | Change ***member*** state to ***state***. Do nothing if ***member*** not in Lobby |
| bool *add*(id member) *override* | Return false if Group::add returns false. Insert new ***member*** with ready-state false. |

### Protected Methods

|  |  |
| --- | --- |
| **Method** | **Explanation** |

### Private Methods

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| bool **isReady**() | Return true if all members are ready |
| void **signalReady**() | Calls eventHandler::start |

## Game

Class: Game

Files: **game.h** and **game.cpp**

<description>

### Public Methods

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| Game(EventHandler\* eventHandler) | Link **eventHandler** |

### Protected Methods

|  |  |
| --- | --- |
| **Method** | **Explanation** |

### Private Methods

|  |  |
| --- | --- |
| **Method** | **Explanation** |

# Client side

# Template

## <class>

Class: <class>

Files: **<class>.h** and **<class>.cpp**

<description>

### Public Methods

|  |  |
| --- | --- |
| **Method** | **Explanation** |

### Protected Methods

|  |  |
| --- | --- |
| **Method** | **Explanation** |

### Private Methods

|  |  |
| --- | --- |
| **Method** | **Explanation** |