Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A31

Game C/S Model

Team:

[Erneste Iradukunda Senga] - Id: [040973461] / [Akshit Sabharwal] - Id: [041025056]

NumPuz Proposal

***This template is suggested (not mandatory) to answer A31 Specification.***

|  |  |
| --- | --- |
| **Part**  **1** | **C/S Architecture** |

* 1. **Server Model**

*Describe how your server interface should be organized and the main methods to be defined*

* + - ***Example****:*

**Example** (see A31 specification)

INTERFACE:

Class: NumPuzServer

® Components:

JLabel:Port ® label for displaying current port

JTextField: txtPort ® displays port information

JButton:start ® starts the connection with server

JButton: result ® sort the all players who participated in playing the game

JTextField: largeTxtField ® displays information regarding the play of the game

JButton:finalize ® button to stop connection with the client once there is not client connected

JButton: end ® end connection

CONTROLLER:

Class: NumPuzServer – Object: “**server**”

® Method: Start:

try (

NumPuzServer **server** = new NumPuzServer (portNumber);

NumPuzClient client = **server**.accept();

}

Method: Listener :

Results : Display Results of the game (sort the all players who participated in playing the game)

Method : End

try {

server.close() // closes/stop the server

}

Method: Finalize: To check if all users have stopped their connection with server, then we can stop the server

* 1. **Client Model**

INTERFACE:

Class: NumPuzClient

® Components:

JLabel: labUser ® label for username

JTextField: txtUser ® Text field for user to enter name

JTextField: txtServer ® Text field area for displaying information

JLabel: labPort ® Label for the current port for the client

JButton: connect ® Button for connecting with server

JButton: endButton ®  for ending connection with server

JButton: newGame ® Button for initialization of new game

JButton: sendGame ® Button for sending game configuration

JButton: receiveGame ® Button for receiving game configuration

JButton: sendData ® Button for sending the data

JButton: play ® Button for start playing

CONTROLLER:

Class: NumPuzClient – Object: “**client**”

® Method: Connect:

try {

NumPuzClient **client** = new Socket(hostName, portNumber);

}

® Method: NewGame: Starts New Instance of Game

® Method: ReceiveGame:To load previous saved game

® Method: SendGame: To save the Game

® Method: SendData:To send the game characteristics like points, time to server.

® Method: Play:To play the game.

® Method: End

try {

client.close(); //will end the connection of client with Server.

}

* 1. **Protocol Proposal**

CONFIGURATION STRING:

Class: NumPuzModel

® Property: String: connectServer:

® Format: <clientName><dataSeparator><serverInfo>, where:

® <dataSeparator> = comma (,)

® <serverInfo> = <portNumber>, <serverName>.

® Example: server info; 4596, localhost

PROTOCOL P0:

® protocolSeparator: hashtag (#)

® Format: <protocolNumber><protocolSeperator><data>

® Example: 1#Prof.,4596, localhost

® Property: String: gameConfig:

® Format: <dim><dataSeparator><dataConfig>, where:

® <dim> = integer (from 2, 3, etc.)

Cf// ® <dataSeparator> = comma (,)

® <dataConfig> = chars (example: 1-9), obeying the formula (dim2)2.

® Example:

numerical;1,2,3,4,5,6,7,8,0

text;M,y, ,g,a,m,e,!,·.

PROTOCOL P1:

® protocolSeparator: hashtag (#)

® Format: <clientId><protocolSeparator><protocolNumber><data>

® Example: 1#2,3,3,1,2,8,6,5,7,4,0

Class: Controller

® Property: String: serverReply:

® Format: <dataConfig>, where:

® <dataConfig> = “success message” and return the instance of the game with configuration and data enter.

PROTOCOL P2:

® protocolSeparator: hashtag (#)

® Format: <clientId><protocolSeparator><data>

® Example: 1#Success

Class: Model

® Property: String: sendData:

® Format: <username><dataSeparator><points><dataseperator><time> where:

Cf// ® <dataSeparator> = comma (,)

® Example:

Prof.,25,55

PROTOCOL P3:

® protocolSeparator: hashtag (#)

® Format: <clientId><protocolSeparator><protocolNumber><data>

® Example: 1#3, Prof.,25,55

|  |  |
| --- | --- |
| **Part**  **2** | **Game Evolution** |

* 1. **Notes about upgrading the game**
* The client and server will be implemented (integrated with the previous MVC game version) in this version of the game. Client will be able to perform certain tasks like loading new game, or previous saved game.
* Two or more players will be able to play the game simultaneously. Once they finish, we will be seeing their results in server based on their game points.
* Controller class will be modified to give access to many clients and to perform certain actions.

**Example** (About MVC modifications)

MODEL component:

Public methods to change private data that can receive inputs but evaluate if they are valid.

View component:

Some Methods like for displaying the GameGrid (Tiles) will be changed to perform changes based on client selection or inputs.

Controller component:

New Methods/Previous Methods will be added/changed to allow to make changes in the games like changing dimensions or playing a new game directly from the client interface.

* 1. **GitHub (Bonus)**
  + *Link:*

**References**

*[Include eventual references used here]*

Algonquin College

Spring / Summer, 2022