Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A31

Game C/S Model

Team:

[Rashin Babaee] - Id: [041043836] / [Sewmini Jayasinghe] - Id: [041057012]

Picross 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: ViewServer

→ Components:

JFrame: splashWindow.

JLabel: logo.

JLabel: labPort, JTextField: txtPort.

JBotton: executeButt.

JBotton: resultsButt.

JBotton: endButt.

JCheckBox: finalize.

JPanel: centerPanel.

JPanel: southPanel.

JTextArea: textArea.

GameServer:

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

→ Method: Start:

try (

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

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

}

→ Method: void execute():

Start the server in true range of port.

→ Method: void results():

display the scores of players.

→ Method: void endServer():

shut down the server.

* ***Note****: The professor interface continues being a proposal. Focus on your ideas using the best user experience.*
  1. **Client Model**

*Describe aspects of your client (interface and methods) considering the proposed idea.*

**Example** (see A31 specification)

INTERFACE:

Class: ViewClient

→ Components:

JFrame: splashWindow.

JLabel: logo.

JLabel: labUser, JTextField: txtUser.

JLabel: labServer, JTextField: txtServer.

JLabel: labPort, JTextField: txtPort.

JBotton: connectBott.

JBotton: endButt.

JBotton: newGameButt.

JBotton: sendGameButt.

JBotton: receiveGameButt.

JBotton: sendDataButt.

JBotton: playButt.

JPanel: centerPanel.

JPanel: southPanel.

JTextArea: textArea.

GameClient:

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

→ Method: Start:

try {

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

} …

→ Method: connect:

It connects to server.

→ Method: endClient:

Disconnect from the server.

→ Method: newGame:

Get the new game random board from server.

→ Method: sendGame:

Sending a board game to server

→ Method: receiveGame:

Receiving the board game from server

→ Method: sendData:

Sending scores of users to server

→ Method: Play:

Playing the game

* 1. **Protocol Proposal**

*Finally, what is your idea to define the protocol to be used.*

**Example** (using the string definition mentioned in the A21 specification)

CONFIGURATION STRING:

Class: GameModel

→ Property: String: gameConfig:

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

→ <dim> = integer (from 2, 3, etc.)

→ <dataSeparator> = comma (,)

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

→ Example: 00001,10111,00110,11111,00011

PROTOCOL P1:

→ protocolSeparator: hashtag (#)

→ Format: <clientId><protocolSeparator><data>

→ Example: 1#3; 00001,10111,00110,11111,00011

P1: sendingBoardToClient<separator> width<separator> height<separator> boardInformation

<separator>: dash (-)

boardInformation=row;row;

There is a boardInformation for each game that it represents as row \* row. There are width and height for each board, and each board information has 0 and 1 for each row. 0 and 1 and spaces in each row represent as row and semicolon is like a next line in the code.

P2: sendingBoardToServer<separator> width<separator> height<separator> boardInformation

<separator>: dash (-)

boardInformation=row;row;

there is a boardInformation for each game that it represents as row \* row. there are width and height for each board, and each board information has 0 and 1 for each row. 0 and 1 and spaces in each row represent as row and semicolon is like a next line in the code.

P3: sendingGameDataToClient<separator> <username><separator><points><separator><time>

<separator>: dash (-)

For sending data to client there is a separator to separate information and it is a dash. Then there is a separator, then points then separator and then time that they are sending to client.

P4: sendingGameDataToServer<username><separator><points><separator><time>

<separator>: dash (-)

For sending data to server there is a separator to separate information and it is a dash. Then there is a separator, then points then separator and then time that they are sending to server.

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

* 1. **Notes about upgrading the game**
  + *Describe the main modifications to be proposed in the C/S version of the game.*
    - *What are the differences between the original proposal (A11 / A21) and the current project to be developed (A31).*
    - *If so, explain why you need to do some adjustments.*

**Example** (About MVC modifications)

MODEL component:

Public methods to change private data (ex: dataConfig), that can receive inputs, but evaluate if they are valid.

For a ssignment A11, it was just an interface of board game, and we explained about that interface. It had JFrames, JButton, BorderLayout, GridLayout, JLabel, JTextField, JCheckBox, JMenu and.JMenuItem. For assignment A21, it was game view, controller, and model that it was functional. The user could play, reset the game, save the game, open the game, change the color of board game, exit from game, see the solution of game, change the language to another language and choose a random game. Also, there were time, points and checked box in that assignment. In assignment A31, we explained about server and client that how they connect to each other and functions and interface. The component of interface is the same of A11 like JButton, JLabel, and …. we explained for each function how it works and what it does.

* 1. **GitHub / Database Integration (Bonus)**
  + *The use of GitHub is also a bonus to be considered:*
    - *Be sure that you can inform the updated repository and branch.*
    - *TIP: To avoid problems, also include the document (template answer) in the BrightSpace.*
  + *Considering this proposal for 3-tier architecture using Databases, define:*
    - *What to persist.*
    - *What is the DB datatype to be used.*
    - *How frequently to update.*

**References**

*[Include eventual references used here]*

Algonquin College

Winter, 2022