Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A31

Game C/S Model

Team:

[Preston Barre] - Id: [041032867] / [Jawad Awada] - Id: [041032251]

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*

For our Client/Server Architecture we will be to create randomized configurations of the puzzle using a method accessed by only one server and sent to the client for the user to use said randomly generated puzzle. The Client will communicate with our MVC solution that we had previously implemented whilst grabbing from the server’s methods in order to run the game through the server, this will be implemented in order to allow the client to communicate with the Game and the Server. Taking from the Server to run certain aspects of the game, and invoking the already established MVC.

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

**Example** (see A31 specification)

INTERFACE:

Class: GameServer

→ Components: JLabel: labPort, JTextField: txtPort, etc.

Constructor: Initializes the server socket

waitForClient: waits for the client to connect to the server, it will return a object representing the client to confirm connection.

sendPuzzle: will send the puzzle across to the connected client

receiveSolution: Receives a solution to a Picross puzzle from the connected client.

sendScore: Sends the user's score to the server after a completed puzzle.

close: Closes the server socket and any associated resources.

CONTROLLER:

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

→ Method: Start:

try (

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

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

}

// CONTINUE…

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

**INTERFACE:**

Class: GameClient

Components: JLabel: labServer, JTextField: txtServer, etc.

// Define the methods that will be used in the Picross game client

Constructor: Initializes the client socket and any other necessary resources.

connectToServer: Connects to the server using the given IP address and port number.

receivePuzzle: Receives a Picross puzzle from the server.

sendSolution: Sends a solution to a Picross puzzle to the server.

receiveScore: Receives the user's score from the server after a completed puzzle.

close: Closes the client socket and any associated resources.

CONTROLLER:

Class: GameClient

Object: "client"

// Define the methods that will be used to connect to the server and play the game

connectToServer: Prompts the user for the server IP address and port number and connects to the server using the connectToServer method.

playGame: The main method that runs the Picross game, calling the necessary methods for each puzzle.

receiveAndDisplayPuzzle: Receives a puzzle from the server using the receivePuzzle method and displays it to the user.

getAndSendSolution: Prompts the user for their solution to the puzzle, sends it to the server using the sendSolution method, and receives the user's score from the server using the receiveScore method.

close: Closes the client socket and any associated resources.

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

**Example** (see A31 specification)

* 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

CONFIGURATION STRING OUR EXAMPLE:

Class: GameModel

Method: welcomePlayer()

Property: int: dimension

Methods: changeDimension(), getDimension()

Class: Player

Property: String: name

Methods: changeName(), getName()

Class: GameView

Property: JPanel: centerPanel

Layout: GridLayout(x, y)

Class: JButton

Property: Dimension: preferredSize

PROTOCOL P1:

protocolSeparator: hashtag (#)

Format: <clientId><protocolSeparator><data>

Example: 1#action=welcomePlayer&dimension=3&playerName=John

|  |  |
| --- | --- |
| **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).*

In this version of our project, we must implement the C/S architecture so a big part of the modifications was making sure that we created a layout the would-be user friendly, to do this we need to be sure that all of the new components/methods that we are adding are able to run cohesively with the UI and perform the tasks on a visual scale for the user to see whilst things are running behind the scenes. Modifications that will be made would be exemplified by us adding the ability for the client to invoke the previously made MVC model of A21 so that was a modification that was needed in order to successfully create A31’s proposal.

**Example** (About MVC modifications)

MODEL component:

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

// CONTINUE…

* 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