Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A31

Game C/S Model

Team:

[Haoyun Deng] - Id: [041012253]

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: GameServer

→ → Components:

Image: serverImage,

JLabel: portLabel,

JTextField: txtPort,

JButton: startServerButton, endServerButton, resultsButton

JTextArea: executionDetail,

Jradiobuttom: finalize

CONTROLLER:

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

→ Method: Start:

try (

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

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

}

→ Method: End: Server ends when there are no more clients connecting.

socket.close();

disable EXIT\_ON\_CLOSE in the JFrame

→ Method: Result: Show ranking list

- Every time new user connects to server, refresh result by alphabetic users’ names

- Each user might send to server multiple times, if that’s the case, I will sort by the time the request is sent for each user.

→ Method: Finalize: Close the server

If (nclients == 0) {

System.exit(0)

}

* ***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: GameClient

→ Components:

Image: clientImage

Label: userLabel, serverLabel, portLabel

Button: connectButton, endClientButton, newGameButton, sendGameButton, receiveGameButton, sendDataButton, playButton

TextField: userTextField, serverTextField, portTextField

TextArea: serverResponseTextArea

CONTROLLER:

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

→ Method: Start:

try {

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

} …

→ Method: End: Disconnect to the server

- Send configuration to server

- socket.close();

→ Method: sendGameConfig:

- Send clientId, dataConfig to the Server

→ Method: sendData:

- Send points, name and time to the server

* 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 P0: When client is connecting with the server

→ protocolSeparator: hashtag (#)

→ Format: <clientId><protocolSeparator><protocol\_Id>

→ Example: 1#0;

PROTOCOL P1: When client is sending a game configuration to server

→ protocolSeparator: hashtag (#)

→ Format: <clientId><protocolSeparator><protocol\_Id><data>

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

PROTOCOL P2: When server is replying P1

→ protocolSeparator: hashtag (#)

→ Format: <clientId><protocolSeparator><protocol\_Id><data>

→ Example: 1#2; 00001,10111,00110,11111,00011

PROTOCOL P3: When client is sending game data to the server

→ protocolSeparator: hashtag (#)

→ Format: <clientId><protocolSeparator><protocol\_Id><protocolSeparator><data>

→ Example: 1#3#; AA 16 60

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

*I need to focus on proposing how the server and client(s) – or players communicate.I need to use network programming knowledge.*

* + - *If so, explain why you need to do some adjustments.*

*We need to add 2 more classes to enable connections of users to an online server.*

*We will add GUI to the game for server and client connections. This will bring clarification for better user experience and to make the server class implementations create server/client communication effectively.*

*We add different protocols*

**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.*

*String for username and configuration*

*Integer for points, timer*

* + - *How frequently to update*

*Data should be update:*

*Player sends configuration.*

*Result from an existing player is submitted*

*Data should be inserted when:*

*A new user connects to server*

*Result from a new user is submitted*

**References**

*[Include eventual references used here]*

Algonquin College

Winter, 2022