Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A31

Game C/S Model

Team:

[Ngoc Phuong Khanh Le] - Id: [041004318] / [Dan McCue] - Id: [040727646]

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: portNum,

JTextField: txtPort,

JButton: endServer, startServer,

JTextArea: executionDetail,

CONTROLLER:

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

→ Method: Start:

try (

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

NumPuzClient 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: NumPuzClient

→ Components:

JLabel: labUser, server, portNum

JTextField: txtUser, txtServer, txtPort

JButton: connect, end, newGame, sendGameConfig, receiveGameConfig, sendData (points and timer)

JTextArea: executionDetail

CONTROLLER:

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

→ Method: Connect: If user clicks connect button, client connects to server

try {

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

}

→ Method: End: If user clicks end button, server is disconnected

- Send to server using protocol P1 (e.g: 1#3#end)

- socket.close();

→ Method: sendGameConfig:

- send formatted game config from the game using protocol P1 (e.g: 2#2#4 Number 5,3,2,6,8,6,11,16,1,4,10,7,12,13,0)

→ Method: sendData:

- send game data (points, timer) using protocol P1 (e.g: 2#3#eri 10 150)

→ Method: newGame:

- invoke MVC for GUI to 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: NumPuzModel

→ Property: String: gameConfig:

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

→ <dim> = integer (from 2, 3, etc.)

→ <type> = Number (N) or Text (T) type

→ <separator> = space (“ “)

→ <dataSeparator> = comma (,)

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

→ Example:

3 N 3,5,1,3,6,8,5,8

4 T M,y, ,g,a,m,e,!

PROTOCOL P0: When client is connecting with the server

→ RequestID: <clientId>

PROTOCOL P1: When client is sending a game configuration or game data (username, points, timer) to server

→ protocolSeparator: hashtag (#)

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

→ Example: 1#1#3 N 3,5,1,3,6,8,5,8 (game config)

→ Example: 1#3#eri 50 350 (game data)

PROTOCOL P2: When server is replying P1

→ protocolSeparator: hashtag (#)

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

→ Example: 1#3 N 3,5,1,3,6,8,5,8

|  |  |
| --- | --- |
| **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).*
      * *Adding C/S architecture to the game with the ability to start a server and control multiple users at the same time – Communication between server and clients*
      * *Creating GUI for C/S architecture with separate JFrame for Server and Clients*
      * *Implementing different protocols (P0: when clients first connect to server, P1: When clients send data to server to process, P2: When server replies to clients’ request)*
      * *Sending different data to server (username, points, timer, game config) to the server*
      * *Creating 2 more classes for Server and Clients*
    - *If so, explain why you need to do some adjustments.*
* *We need to make some adjustments because it will allow the applications to expand for multiple users.*
* *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 in order to:*
  + *Server side: allow server to reply to clients’ requests appropriately*
  + *Client side: make sure that different requests from clients make sense and understood by server.*
* *We need to add 2 more classes to enable connections of users to an online server.*

**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.*
      * *VARCHAR for String, text in shuffleText array*
      * *INTEGER for points, timer, numbers in shuffleNum array*
    - *How frequently to update.*
      * *Data should be UPDATED when:* 
        + *Highest points among all user’s games*
        + *Final result after users wins or lose*
        + *User changes username or modify an existing configuration with same file name*
      * *Data should be INSERTED when:*
        + *A new user connects to server*
        + *A new game is played with new configurations (points, timer, movements, etc.)*

**References**

*[Include eventual references used here]*

Algonquin College

Spring / Summer, 2022