Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science

Graphical user interface, website

Description automatically generated

A31

Game C/S Model

[Quoc Thang Tran] - Id: [041009239]

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:

Image, ImageView: serverImage,

Label: portLabel

Button: startServerButton, endServerButton, resultsButton

CheckBox: finalizeCheckBox

TextField: portTextField

TextArea: serverActivityTextArea

CONTROLLER:

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

→ Method: Start:

try (

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

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

}

→ Method: End:

socket.close();

→ Method: Result:

- Show list of players that is sorted by decrescent number of points, movements and time.

- Is called every time a player connects to the server or submit their result.

→ Method: Finalize: Close the server

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:

Image, ImageView: clientImage

Label: userLabel, serverLabel, portLabel

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

TextField: userTextField, serverTextField, portTextField

TextArea: serverResponseTextArea

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: Disconnect to the server

- Send configuration to server

- socket.close();

→ Method: sendGameConfig:

- Send clientId, dimension, type and solution to the Server (Ex: 2#3 Number 1,2,3,4,5,6,7,8,0)

→ Method: sendData:

- Send points, movements 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: NumPuzModel

→ Property: String: gameConfig:

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

→ <dim> = integer (from 3, 4 , etc.)

→ <type> = “Number” or “Text”

→ <separator> = space (“ “)

→ <dataSeparator> = comma (,)

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

→ Example:

3 Number 3,5,1,3,6,8,5,8

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

→ protocolSeparator: hashtag (#)

PROTOCOL P0: When client is connecting with the server

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

→ Example: 1#0

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

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

→ Example: 1#1#3 Number 3,5,1,3,6,8,5,8

PROTOCOL P2: When server is replying P1

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

→ Example: 1#3 Number 3,5,1,3,6,8,5,8

PROTOCOL P3: When client is sending game data

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

→ Example: 1#3#Brian 16 80 120

|  |  |
| --- | --- |
| **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 with each other more than interface (A11) or the game’s implementation (A21)*
    - *If so, explain why you need to do some adjustments.*
* *Because the C/S structure require more classes for client(s) and server.*
* *Because now I may have more than one player at a time and need to make sure all the versions of the game run properly for each client.*

**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.*
      * *Nothing*
    - *What is the DB datatype to be used.*
      * *VARCHAR for user’s name, type*
      * *INTEGER for all the rest data*
    - *How frequently to update.*
      * *Data should be UPDATED when:* 
        + *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

Spring / Summer, 2022