Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A21

Game MVC

Team:

[Haoyun Deng] - Id: [041012253]

NumPuz Proposal

***This template is suggested (not mandatory) to answer A21 Specification.***

|  |  |
| --- | --- |
| **Part**  **1** | **GUI Definition** |

* 1. **MVC Details**

*Describe the way you can define the MVC components in your game.*

**Example** (from vision “top-down”)

Model Class: GameModel – Object: “gameModel” (POJO / Plain Java Old Object)

View Element: GameView – Object: “gameView” (extends JFrame implements GameController)

Controller Class: GameController – Object: “gameController” (responsible for all Actions)

* 1. **View Component**

*Describe how your interface should be organized using new components. Show the idea about your “top-down” organization.*

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

**Example** (from vision “top-down”)

Class: JFrame – Object: “Frame”

→ Class: JMenuBar → Object: “bar”

→ Class: JMenu → Object: “language”

→ Class: JMenuItem → Objects: “english”, “chinese”

→ Class: JMenu → Object: “game”

→ Class: JMenuItem → Objects: “new”, “solution”,”exit”,”colors”,”help”

→ Class: JPanel → Object: “playPanel”

→ Class: JPanel → Object: “logoPanel”

→ Class: JPanel → Object: “topPanel”

→ Class: JLabel[size]→ Object: “number Lable”

→ Class: JPanel → Object: “leftPanel”

→ Class: JLabel[size]→ Object: “number Lable”

→ Class: JPanel → Object: “boardPanel”

→ Class: JPanel[size][size] → : “p[size][size]”

→ Class: JPanel → Object: “controlPanel”

→ Class: JButtons → Objects: “resetButton”, “solutionButton”, ” saveButton”,” loadButton”,” randomButton”,” creatButton”

→ Class: JTextField→ Objects: “sizeField”, “SizeField”,

→ Class: JPanel→ Objects: “emptyPanel”, “savePanel”, “creatpanel”, “randompanel”, “solutionpanel”, “resetpanel”

→ Class: JPanel → Object: “ informationPanel”

→ Class: JPanel → Object: “ infoPanel”,“ timePanel”,“ pointPanel”,“ dispalyPanel”

→ Class: JTextField → Objects: “timeField”,”pointField”

→ Class: JTextArea → Objects: “infoField”

→ Class: JLabel→ Objects: “timeLabel”,”pointLabel”

→ Class: JDialog → Object: “help”, “creat”, “solution”,” color”

…

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

*Describe aspects of your controller using, for example, one unique action command. Create the “map” to define functions with actions.*

**Example**

Object: “language”

Event: actionPerformed → method: updateLanguage()

Object: “random”

→ Event: actionPerformed → method: setMatrixSize(),creat(),randomly()

Object: “creat”

→ Event: actionPerformed → method: setMatrixSize(),creat()

Object: “reset”

→ Event: actionPerformed → method: reset()

Object: “save”

→ Event: actionPerformed → method: saveArray()

Object: “load”

→ Event: actionPerformed → method: loadArray()

Object: “click”

→ Event: mouseClicked→ method: update(),mark()

Object: “solution”

→ Event: actionPerformed → method: solution()

Object: “exit”

→ Event: actionPerformed → method: exit()

Object: “colors”

→ Event: actionPerformed → method: colors()

Object: “help”

→ Event: actionPerformed → method: help()

* 1. **Model Component**

*Finally, what is your idea to define the model to be used in a “default” (randomized) game.*

**Example**

Data structure used:

→ Values: solution[][] → method: setArray(choose),getArray(choose),randomly(),printTop(),printLeft(),

mark(row,column,state)

→ Values: playArea[][] → method: setArray(choose),getArray(choose),reset(),saveArray(),loadArray(),

update(row,column,state)

→ Values: saveArea[][] → method: setArray(choose),getArray(choose),saveArray(),loadArray()

→ Values: matrixSize → method: getMatrixSize(),setMatrixSize(matrixSize),randomly(),reset(),

setArray(choose)

→ Values: mode → method: initializeControl(panel),initializeBoard(panel),setMode(), getMode()

→ Values: points → method: point(),setPoint(), getPoint();

→ Values: name → method: setName(), getName()

→ Values: maxScore → method: setMaxScore(), getMaxScore()

|  |  |
| --- | --- |
| **Part**  **2** | **Implementation Design** |

* 1. **Game Evolution**
  + *Considering this new model, explain:*
    - *What are the differences between the original proposal (A11) and the current project to be developed (A21).*

*A11 only contains information about GUI appearance and required components to build.*

*A21 will focus more on the interaction with users and the control of the underlying data. And realize the functions of various games*

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

*These updates facilitate maintenance. Because each part is modular, changing or updating the code in one part won't easily break other parts. And realize the function and data record of user's play.*

* 1. **Others DP**

* Define (at least one) additional DP that you could use in your Game application.*

*o Explain what is this DP and the reason why it could be recommended.*

***Accessors and Mutators***

Description: Access or update a data field.

Reason: Guarantee the data encapsulation,enhanced security

**References**

*[Include eventual references used here]*

Algonquin College

Spring / Summer, 2022