Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A21

Game MVC

Team:

[Neeraj Kumar Bansal] - Id: [041000185] / [Jay Patel] - Id: [041028206]

**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: It represents an object which carries data, it also can change data if logic of controller updates.

→ GameModel - Object: “model”

View Class: This represents the visualization of the data that model contains.

→ GameView - Object: “view”

Controller Class: Controller acts on both mode and view, it controls the data flow into model objects and also keeps model and view separate.

→ GameController - Object: “controller”

* 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: “popup”

→ Class: JPanel → Object: “gamePanel” “sidePanel” “menuPanel” “numpuzGamePanel” “sidePanelTop” “sidePanelCenter” “sidePanelBottom” “text”

→ Class: JButtons → Objects: “setButton”, “gameLogoButton”, “showButton” “hideButton” “saveButton” “loadButton” “randButton” “finishButton” “resetButton” “gameButton” “helpButton”

→ Class: JLabel → Objects: “modeLabel”, “dimLabel”, “typeLabel” “moveLabel” “pointLabel” “timeLabel”

→ Class: JComponent → Methods: “paintComponent()”

Class: “View”

→ Event: actionPerformed→ method: showGridColorMenu()

Class: “View”

→ Event: actionPerformed→ method: drawStartMessage()

Class: “View”

→ Event: actionPerformed→ method: drawCentredString()

Class: “View”

→ Event: actionPerformed→ method: drawGrid()

Objects: “popup”

→ Event: actionPerformed→ method: setMenuButtons()

Objects: “popup”

→ Event: actionPerformed→ method: setSidePanelButtons()

Object: “popup”

→ Event: actionPerformed→ method: setLabels()

Class: “View”

→ Event: actionPerformed→ method: setRadioButtons()

* ***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: “gameLogoButton” is a button which shows the logo in maximized size.

→ Event: actionPerformed → method: setGameLogo()

Object: “setButton” is a button which uploads the text from input text field into tiles in randomized character order.

→ Event: actionPerformed → method: setStringInGrid()

Object: “dimSelect” is an action which proceeds further actions when there is a selection in the dimensions bar.

→ Event: actionPerformed → method: setGameDimension()

Class: “addMouseListener”

→ Event: mouseEvent → method: mousePressed()

Class: “Controller”

→ Event: actionPerformed→ method: paintComponent()

Class: “Controller”

→ Event: actionPerformed→ method: drawStartMessage()

Class: “Controller”

→ Event: actionPerformed→ method: drawCentredString()

Class: “Controller”

→ Event: actionPerformed→ method: resetGame()

Class: “Controller”

→ Event: actionPerformed→ method: shuffleNumberInGrid()

Class: “Controller”

→ Event: actionPerformed→ method: startNewGame()

Class: “Controller”

→ Event: actionPerformed→ method: checkSolved()

Class: “Controller”

→ Event: actionPerformed→ method: resetGrid()

Class: “Controller”

→ Event: actionPerformed→ method: checkSolvability()

Class: “Controller”

→ Event: actionPerformed→ method: changeColor()

Class: “Controller”

→ Event: actionPerformed→ method: saveGame()

* 1. **Model Component**

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

**Example**

Data structure used:

Object: “grid”

→method: ”newGame()”, “repaint()”, “reset()”, “updateScore()”, “updateMoves()”, ”setTime()”, ”getTime()”, ”setProgress()”, ”getProgress()”, “getMoves()”.

| **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).*

1.The menu bar in the main JFrame is missing in the recent implementation of Assignment A12.

2.The game layout and color are now vibrant and attractive compared to the proposed layout of A11.

3. Some more buttons like show and hide got added in A12, and also a scroll bar got added to choose between Text input tiles and randomized Number tiles. In Text tiles, users will input any text and try to arrange it in tiles to win the game.

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

1. It plays a pivotal role for providing functionalities to the game and also for the users’ convenience. With the help of the menu bar, users can choose to look at guidelines or description of the game by clicking About button or by settings button in sub-options, color and size of the frame can be changed and adjusted respectvely.

2.The change in the layout color and design resembles the ideal feature for a game and makes it more attractive.

3. By adding a scroll bar for Number and Text, the game becomes more engaging and interesting from users perspective. Text option will allow the user to input text of his/her choice and have to arrange in correct order to win.

* 1. **Others DP**
     + *Define (at least one) additional DP that you could use in your Game application.*
* One of the DP which can be used implicitly is Structural and Command Pattern which is a sub-part of behavioral pattern.
  + *Explain what this DP is and the reason why it could be recommended.*

**Bridge Pattern** - DP lets you split large classes into closely related classes into two separate hierarchies - abstraction and implementation - which can be developed independently of each other.

**Command Pattern** is useful in terms of having multiple buttons with multiple codes which cannot be reverted, but using Command Pattern DP, all the buttons actions can be observed and traced and saved in a file. This makes the GUI tracked and helps him manage it.

**References**

[*https://refactoring.guru/design-patterns*](https://refactoring.guru/design-patterns)

[*https://www.javatpoint.com/design-patterns-in-java*](https://www.javatpoint.com/design-patterns-in-java)

Algonquin College

Spring / Summer, 2022