ALGONQUIN
COLLEGE

## SCHOOL OF ADVANCED TECHNOLOGY

## ICT - Applications & Programming
## Computer Engineering Technology – Computing Science

# A21

# Game MVC

**Team:**
**[Savas Erturk]** - Id: **[040919022]**

## NumPuz Proposal

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

# Part 1 GUI Definition

## 1.1. MVC Details

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

**Example** (from vision "top-down")

Model Class: GameModel – Object "myModel"

View Element: GameView – Object "myView"

Controller Class: GameController – Object: "myController"

…

## 1.2. 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: "GameFrame"

→ Class: JMenuBar → Object: "MenuBar"

→ Class: JButtons → Objects: "Main", "Help", "About"

→ Class: JPanel → Object: "GameBoard"

→ Class: JButtons[row][col] → : "grid[row][col]" (depends on dimensional size)

→ Class: JPanel → Object: "RightPanel"

→ Class: JLabel → Objects: "TimeLabel", "Timer", "Dim",

→ Class: JButtons → Objects: "gameStart", "gameReset", " setGameType"

→ Class: JComboBox → Objects: "DimSize", "gameType",

→ Class: JRadioButton → Objects: "normalOption", "hardOption",

→ Class: JPanel → Object: " SouthPanel"

→ Class: JTextField → Objects: "gameText"

→ Class: JButton → Objects: "setGameText"

…

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

## 1.3. Controller Component

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

**Example**

Object: "gameStart"

→ Event: actionPerformed → method: shuffleBoard()

Object: "gameReset"

→ Event: actionPerformed → method: gameReset()

Object: "setGameType"

→ Event: actionPerformed → method: changeGameState()

Object: "setGameDifficulty"

→ Event: actionPerformed → method: changeGameDifficulty()

Object: "setGameText"

→ Event: actionPerformed → method: setGameText(gameMode, gameText)

Object: "selectedButton"

→ Event: actionPerformed → method: setSelected()

Object: "movement"

→ Event: actionPerformed → method: movement(selectedButton, targetButton)

## 1.4. Model Component

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

**Example**

Data structure used:

→ Values: gridValue[row][col] → method: updateData()

→ Values: shuffleboard(gameMode, gameText) → method: shuffleBoard() return board[][]

→ Values: board[][] → method: getBoard()

→ Values: board[][] → method: solution(board)

→ Values: board,solutionBoard → method: solveBoard()

→ Values: gameResult,gridValue → method: refresh()

→ Values: gridValue[row][col] → method: updateData()

Points, time, name, score, board[][], solutionBoard[][] will create setter and getters.

When it’s created the board result of the board will be created automatically then every refresh step it will compare with result and movement.

# Part 2 Implementation Design

## 2.1. Game Evolution

- *Considering this new model, explain:*
    - *What are the differences between the original proposal (A11) and the current project to be developed (A21).*
    - *If so, explain why you need to do some adjustments.*

*A11 I did not think about refresh method is necessary on project. But I realized It must be there according to the my plan. Because …. Also with this I start to get understand the idea of better solution. I will create in Model class, shuffle the data and solution of the data. When User will do movement on the game it will refresh game data and will compare it with final data at the same time. If user reach the final data game will and with success.*

## 2.2. Others DP

- *Singleton Pattern*
    - *It's really common pattern to find classes for which object instance should exist.*
    - *In MVC architecture controller if its not have an static data it should be singleton.*
- *State Pattern*

*• User when click the "setGameType" object it will run changeGameState. Game state will be Boolean which needs true and false. If the user will select the numbers it will be false otherwise if a user wants to play with letters it will be true which is the board will provide text which is given by the user.*

*Also state pattern will use on restrict movement. For example, user have to choose first button then user only allow to chose left, right, up or down.*

- *Null Object Pattern*

  *User needs to select 2 button in order to do movement.*

## References

*[Include eventual references used here]*

Algonquin College
Spring / Summer, 2022