Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A21

Game MVC

Team:

[Quoc Thang Tran] - Id: [041009239]

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: “myModel” (Plain Java Old Object)

View Element: GameView – Object: “myView” (extends Application implements GameController)

Controller Class: GameController – Object: “myController” (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: Stage → Object: “primaryStage”

→ Class: Scene → Object: “scene”

→ Class: BorderPane → Object: “root”

→ Class: MenuBar → Object: “menuBar”

→ Class: Menu → Objects: ”game, help”

→ Class: MenuItem - > Objects: “new, solution, exit, colors, about

→ Class: GridPane → Objects: "getPlayPane, getFunctionPane”

→ Class: Button → Objects: “button, design, play, save, load, rand, finish”

→ Class: Label → Objects: “mode, dim, type, moves, points, time”

→ Class: ComboBox → Objects: “dimComboBox, typeComboBox”

→ Class: TextField → Objects: “movesTextField, pointsTextField, movementsTextField, timeTextField”

→ Class: Dialog → Object: “aboutDialog, save, load, color”

→ Class: FlowPane → Object: “getBottomTextField”

→ Class: TextField → Object: “userInput”

→ Class: Button → Object: “submitButton”

…

* ***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: “design”

→ Event: ActionEvent → method: setMode()

Object: “play”

→ Event: ActionEvent → method: setMode()

Object: “dimComboBox”

→ Event: ActionEvent → method: setDim()

Object: “typeComboBox”

→ Event: ActionEvent → method: setType()

Object: “save”

→ Event: ActionEvent → method: showSaveDialog()

Object: “load”

→ Event: ActionEvent → method: showLoadDialog()

Object: “rand”

→ Event: ActionEvent → method: getRandConfiguration()

Object: “finish”

→ Event: ActionEvent → method: finishGame()

Object: “button”

→ Event: ActionEvent → method: buttonClick()

Object: “submitButton”

→ Event: ActionEvent → method: submitText()

Object: “new”

→ Event: ActionEvent → method: newGame()

Object: “solution”

→ Event: ActionEvent → method: showSolution()

Object: “exit”

→ Event: ActionEvent → method: exitGame()

Object: “colors”

→ Event: ActionEvent → method: showColorsDialog()

Object: “about”

→ Event: ActionEvent → method: showAboutDialog()

* 1. **Model Component**

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

**Example**

Data structure used: ArrayList

→ Values: dim → methods: setDim(), getDim()

→ Values: input → method: setInput(), getInput()

→ Values: type → method: setType(), getType()

→ Values: gridValue[][] → method: updateData(), compareSolution()

→ Values: gridSolution[][] → method: showSolution(), updateSolution()

→ Values: points → methods: setPoints(), getPoints()

→ Values: Time → method: setTime(), getTime ()

→ 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).*
      * The A11 only included general ideas of how the GUI looks like and needed components.
      * The A21 separates the game into 3 parts (controller, view and model) and shows all the components’ information in details about how.
    - *If so, explain why you need to do some adjustments.*
      * These updates are very useful for later application maintenance. Because each part is separated individually, changing or updating the code of one part will not make other part crash easily.
  1. **Others DP**
     + *Define (at least one) additional DP that you could use in your Game application.*
  + *Explain what is this DP and the reason why it could be recommended.*

1. ***Getters, Setters (Accessors, Mutators)***

Description: Access or update a data field.

Reason: Guarantee the data encapsulation

1. ***Iterator***

Description: Access elements of a collection sequentially without exposing its implementation

Reason: Main data (data in the grid) is kept in ArrayList (which also use the Iterator pattern)

**References**

*[Include eventual references used here]*

*https://brightspace.algonquincollege.com/d2l/le/content/461737/viewContent/7434830/View*

Algonquin College

Spring / Summer, 2022