Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A21

Game MVC

Team:

Jack Johnston - Id: 041047407

NumPuz Proposal

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

* 1. **MVC Details**

**Example** (from vision “top-down”)

Model Class: NumPuzModel – Object: “model” (POJO / Plain Java Old Object)

View Element: NumPuzView – Object: “view” (extends JFrame implements GameController)

Controller Class: NumPuzController – Object: “control” (implements action listener responsible for all Actions)

Class: JFrame – Object: “Frame”

→ Class: JPanel → Object: gameBoard, topControls, bottomControls, sideGrid

→ Class: ActionListener → Object: ActionEvent e

→ Class: JMenu – Object: “Mhelp”,’Mfile”,”Mexit”

→ Class: JButtons → Objects: “Bmode”,” BsetNames”,”Btemplate”,”Bshow”,”Bhide”,”Brandom”,”Bfinish”,

”Bapply”,”Bsave”,”Bload”,”Blogo”

→ Class: JTextField → Objects: “Points”,” time”,”setNameTextBox”

→ Class: JFileChooser – Object: “fileDirectory”

→ Class: JComboBox – Object: “comboBoxSize”,”comboBoxGrid”

→ Class: JTextArea – Object: “textField”

…

* 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: JPanel → Object: gameBoard, topControls, bottomControls, sideGrid

→ Class: JMenu – Object: “Mhelp”,’Mfile”,”Mexit”, “Msolution”

→ Class: JButtons → Objects: “Bmode”,” BsetNames”,”Btemplate”,”Bshow”,”Bhide”,”Brandom”,”Bfinish”,

”Bapply”,”Bsave”,”Bload”,”Blogo”,”Gbuttons”

→ Class: JTextField → Objects: “Points”,” time”,”setNameTextBox”

→ Class: JFileChooser – Object: “fileDirectory”

→ Class: JComboBox – Object: “comboBoxSize”,”comboBoxGrid”

→ Class: JTextArea – Object: “textField”

…

* ***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**

→ Class: ActionListener → Object: ActionEvent e

Object:”Gbuttons”

→ Event: actionPerformed → method: updateData()

Object: “BSave”

→ Event: actionPerformed → method: saveGame()

Object: “Bload”

→ Event: actionPerformed → method: loadGame()

Object “BsetNames”

→ Event: actionPerformed → method: newGridText()

Object “Bfinish”

→ Event: actionPerformed → method: finishGame()

Object “Bhide”

→ Event: actionPerformed → method: hideGame()

Object “Brandom”

→ Event: actionPerformed → method: randomGame()

Object “Bshow”

→ Event: actionPerformed → method: showGame()

Object “Btemplate”

→ Event: actionPerformed → method: changeGrid()

Object “Bplay”

→ Event: actionPerformed → method: setEnabled()

Object “Bdesign”

→ Event: actionPerformed → method: setEnabled()

Object “Mexit”

→ Event: actionPerformed → method: dispose()

Object “Mfile”

→ Event: actionPerformed → method: saveGame() / loadGame()

Object “Mhelp”

→ Event: actionPerformed → method: showHelp()

Object “Msolution”

→ Event: actionPerformed → method: showSolution()

Object “Button” with name “Bgrid” attached using function “StartsWith(“Bgrid”)

Can help identify buttons located in the game board grid

→ Values: findSolution → method: getSolutionData()

→ Values: availableCoordinates → method: getavailableCoordinates ()

→ Values: moveTilesIndex → method: moveTilesIndex()

* 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: gridValue → method: updateData()

→ Values: findSolution → method: getSolutionData()

→ Values: availableCoordinates → method: getavailableCoordinates ()

→ Values: finishedSolution → method: getFinalGrid()

→ Values: moveTilesIndex → method: moveTilesIndex()

|  |  |
| --- | --- |
|  | **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).*

*There are differences between the original proposal, the main difference is the design pattern to be used. I didn’t delve that much to design pattern in my original design but for the foreseeable future I will be implementing the MVC design pattern to make the program much more efficient and easily modifiable. The addition of a splash screen and menu bars to select colors is a new addition to be added in my program.*

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

*Implementing the MVC pattern will make an easier adjustment to future features to be added so that we can modify it without changing or affecting much of the program, I should also consider the type of layout manager to be used as I’m currently using a combination of them.*

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

*I would be using the observer design pattern for an alternative.*

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

*Observer design pattern, at its core, is very usefll as dependent objects to the class are to be notified automatically, for some programs this may be abit redundant but focusing the fact that we are making a game this could prove it self to be very usefull*

**References**

[*https://www.tutorialspoint.com/design\_pattern/observer\_pattern.htm*](https://www.tutorialspoint.com/design_pattern/observer_pattern.htm)

*https://www.geeksforgeeks.org/mvc-design-pattern/*

Algonquin College

Spring / Summer, 2022