Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A21

Game MVC

Team:

[Rashin Babaee] - Id: [041043836] / [Sewmini Jayasinghe] - Id: [041057012]

Picross Proposal

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

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

*This template is very similar to your A11, but going deeper with the components and methods descriptin and dividing them into the MVC components.*

* 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) …

**Picross class**: main class

* 1. **View Component**

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

*\** ***TIP****: Review the components defined in the A11 and put them in the proper hierarchy (as you done in the A12).*

* + - ***Example (Main Frame)****:*

**Example** (from vision “top-down”)

Class: JFrame – Object: “ f ”

→ Class: JPanel → Object: “gamepanel”

→ Class: JPanel → Object: “centerpanel”

→ Class: JPanel → Object: “eastpanel”

→ Class: JButtons → Objects: “button” = buttons[j][i]

→ Class: JButtons → Objects: “savebutton”

→ Class: JButtons → Objects: “resetbutton”

→ Class: JLabel → Objects: “language”

→ Class: JLabel → Objects: “points”

→ Class: JLabel → Objects: “time”

→ Class: JCheckBox → Objects: “checkbox”

→ Class: JTextArea → Objects: “area”

→ Class: JTextField → Objects: “pointsfield”

→ Class: JComboBox → Objects: “Languagefield”

→ Class: ImageIcon → Objects: “imageicon”

→ Class: JMenuBar → Objects: “menuBar”

→ Class: JMenu → Objects: “menu”

→ Class: JMenuItem → Objects: “openItem”

→ Class: JMenuItem → Objects: “newItem”

→ Class: JMenuItem → Objects: “saveItem”

→ Class: JMenu → Objects: “settings”

→ Class: JMenuItem → Objects: “backgroundItem”

…

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

**Example** (from vision “top-down”)

Class: JFrame – Object: “splashWindow”

→ Class: JPanel → Object: “splashpanelcenter”

→ Class: JLabel → Object: “logo”

→ Class: JLabel→ Object: “loading”

…

* 1. **Controller Component**

*Describe aspects of your controller using, for example, one unique action command. Create the “map” to define functions with actions. This activity is to plan what will happen in your action. Ex:*

**Example**

→ Event: actionPerformed → method: showAbout()

Etc.

// Future implementations - events (in controller)

**class** Controller **implements** ActionListener {

**public** **void** actionPerformed(ActionEvent e) {

String action = e.getActionCommand();

out.println(action);

if(action == "Language") {

JComboBox<String> language = (JComboBox<String>) e.getSource();

String selectlanguage = (String)language.getSelectedItem();

out.println(selectlanguage);

}

else if(action == "Open") {

game.openFile();

}

else if(action == "Save") {

game.saveFile();

}

else if(action == "BackgroundColor") {

game.chooseColor();

}

else if(action == "New") {

game.chooseNew();

}

}

}

* 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: game → method: changeSize()

→ Properties: dimension (int) → methods related: changeDimension(), getDimension(), …

→ Properties: Player (class)

→ Property: name (String) → methods related: changeDimension(), getDimension(),

|  |  |
| --- | --- |
| **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).*
    - In the A11 proposal, we decided not to use the checked mark option because it will not be useful for implementing our game. Since it is a requirement in A21 and we understood that it will be useful for players, so we decided to add the checked mark back to our implementation.
    - *Moreover, in A11 we explained that we have frame, label, button and label but in A21 we explained lots of details like for each class we have object and we want to have GameController, GameView and GameModel classes. Also, when the user run the game, it pops up with splash window.*
    - *If so, explain why you need to do some adjustments.*
    - *Because when we go deeper with the components and methods description, it is like a plan to write a code and it will be easier to write a code.*
  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.*
  + We can use the Observer Design Pattern which allows multiple objects to be notified and updated automatically when the state of a subject-object change. In our game application, the Observer pattern could be used to edit different parts of the game interface and components based on the player's actions. For example, when the player clicks a particular square, the Observer pattern could be used to update the player's score and update the inventory.

**References**

*[Include eventual references used here]*

Algonquin College

Winter, 2022