Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A21

Game MVC

Team:

[Ahnaf Kamal] - Id: [041034479] / [Skylar Phanenhour] - Id: [041024042]

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: “model”

: Will contain the logic of the game. Things such as scores, timer, gridSize and data related methods.

View Class: GameView -> Object: “view”

: Will contain the GUI elements and data needed to update itself, and provide data/get data through the controller.

Controller Class: GameController -> Object: “controller”

: contains the action listeners, and getter/setters needed to communicate with the model and view.

* 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).*

* + - ***(playWindow)****:*

Play Window

Class: JFrame -> Object: playWindow (Play)

Class: JPanel -> Object: "leftPanel"

Class: JPanel -> Object: "scorePanel"

Class: JLabel -> Object: "scoreLabel"

Class: JTextField -> Object: "scoreCounter"

Class: JPanel -> Object: "timerPanel"

Class: JLabel -> Object: "timerLabel"

Class: JTextField -> Object: "timerCounter"

Class: JPanel -> Object: "buttonPanel"

Class: JButton -> Object: "resetButton"

Class: JButton -> Object: "solveButton"

Class: JButton -> Object: "newBoardButton"

Class: JButton -> Object: "instructionsButton"

Class: JPanel -> Object: "configurationPanel"

Class: JPanel -> Object: "languagePanel"

Class: JPanel -> Object: "languageButtonPanel"

Class: ButtonGroup -> Object: "langButtons"

Class: JRadioButton -> Object: "engRadio"

Class: JRadioButton -> Object: "frRadio"

* + - ***(titlePanel)****:*

Title Panel

Class: JPanel -> Object: "titlePanel"

Class: ImageIcon -> Object: "picrossLogo"

Class: JLabel -> Object: "picrossLabel

* + - ***(controlPanel)****:*

Control Panel

Class: JPanel -> Object: "controlPanel"

Class: JPanel -> Object: "historyPanel"

Class: JTextArea -> Object: "history"

Class: JScrollePane -> Object: "scroll"

* + - ***(boardPanel)****:*

Board Panel

Class: JPanel -> Object: "boardPanel"

Class: JPanel -> Object: "colPanel"

Class: JLabel -> Object: "grid"

Class: JPanel -> Object: "rowPanel"

Class: JLabel -> Object: "grid"

Class: JCheckBox -> Object: "Mark"

Class: JPanel -> Object: "gridButtonPanel"

Class: JButton newGridButton -> Object: "buttons"

* + - ***(menuBar)****:*

Menu Bar

Class: JMenuBar -> Object: "menuBar"

Class: JMenu -> Object: "gameMenu"

Class: JMenuItem -> Object: "newMenu"

Class: ImageIcon -> Object: "newIcon"

Class: JMenuItem -> Object: "solutionMenu"

Class: ImageIcon -> Object: "solutionIcon"

Class: JMenuItem -> Object: "exitMenu"

Class: ImageIcon -> Object: "exitIcon"

Class: JMenuItem -> Object: "saveMenu"

Class: ImageIcon -> Object: "saveIcon"

Class: JMenuItem -> Object: "openMenu"

Class: ImageIcon -> Object: "openIcon"

Class: JMenu -> Object: "helpMenu"

Class: JMenuItem -> Object: "colourMenu"

Class: ImageIcon -> Object: "colourIcon"

Class: JMenuItem -> Object: "aboutMenu"

Class: ImageIcon -> Object: "aboutIcon"

* + - ***(instructionsWindow)****:*

Instructions Window

Class: JFrame -> Object: "instructionsWindow" (Instructions)

Class: JPanel -> Object: "instructionsPanel"

Class: JLabel -> Object: "instructionsLabel"

Class: JButton -> Object: "instructionsBack"

* + - ***(designWindow)****:*

Design Window

Class: JFrame -> Object: "designWindow" (Design)

Class: JPanel -> Object: "boardPanel"

Class: JPanel -> Object: "configGrid"

Class: JPanel -> Object: "languagePanel"

Class: JPanel -> Object: "gridSizeCmboPanel"

Class: JComboBox -> Object: options[] = {"5x5", "6x6", "7x7"}

Class: JLabel -> Object: "gridSize"

* + - ***(startWindow)****:*

Start Window

Class: JFrame -> Object: "startWindow" (Splash)

Class: JPanel -> Object: "splashTitlePanel"

Class: ImageIcon -> Object: "titleLogo"

Class: JLabel -> Object: "titleLabel"

* + - ***(launcherWindow)****:*

Launcher Window

Class: JFrame -> Object: "launcherWindow" (Splash)

Class: JPanel -> Object: "splashTitlePanel"

Class: ImageIcon -> Object: "titleLogo"

Class: JLabel -> Object: "titleLabel"

Class: JPanel -> Object: "startPanel"

Class: JButton -> Object: "designButton"

Class: JButton -> Object: "playButton"

* 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:* 4

Instructions Button

Object: "instructionsButton"

-> Event: actionPerformed -> method: showInstructions()

Instructions Back Button

Object: "instructionsBack"

-> Event: actionPerformed -> method: Game()

Reset Button

Object: "resetButton"

-> Event: actionPerformed -> method: resetBoard()

Solve Button

Object: "solveButton"

-> Event: actionPerformed -> method: solveBoard()

New Board Button

Object: "newBoardButton"

-> Event: actionPerformed -> method: makeNewBoard()

Design Button

Object: "designButton"

-> Event: actionPerformed -> method: makeDesignWindow()

Design Back Button

Object: "designBackButton"

-> Event: actionPerformed -> method: Game()

Play Button

Object: "playButton"

-> Event: actionPerformed -> method: makePlayWindow()

English Radio Button

Object: "engRadio"

-> Event: actionPerformed -> method: changeLang()

French Radio Button

Object: "frRadio"

-> Event: actionPerformed -> method: changeLang()

Mark Check Box

Object: "markCheckBox"

-> Event: actionPerformed -> method: updateGameMode()

Grid Size Combo Box

Object: "gridSizeCmbo"

-> Event: actionPerformed -> method: updateBoardSize()

2-D Button Array

Object: "buttons"

-> Event: actionPerformed -> method: gridValidation()

**Example**

Object: “playButton” & Object: “designButton”

private void launcherActions() {

view.playButton.addActionListener((actionEvent) -> {

System.***out***.println("clicked play");

view.startWindow.dispose();

view.Play();

playActions();

});

view.designButton.addActionListener((actionEvent) -> {

System.***out***.println("clicked design button");

view.startWindow.dispose();

view.Design();

designActions();

});

}

* 1. **Model Component**

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

**Model Components**

Data structure used:

Values: play -> method: makePlayWindow()

Properties: gridSize(int) -> methods related: updateBoardSize(), getBoardSize(), setBoardSize()

Properties: markMode(boolean) -> methods related: updateGameMode()

Properties: language -> methods related: changeLang()

Properties: points(int) -> methods related: updateData(), user(), getMaxScore(), setMaxScore()

Properties: time(int) -> methods related: timer(), user(), getBestTime(), setBestTime()

Properties: Player(class)

Properties: name(string) -> methods related: user()

Properties: maxScore(int) -> methods related: user(), updateData(), getMaxScore(), setMaxScore()

Properties: bestTime(int) -> methods related: user(), getBestTime(), setBestTime()

Values: design -> makeDesignWindow()

Properties: gridSize(int) -> methods related: updateBoardSize(), getBoardSize(), setBoardSize()

Properties: language -> methods related: changeLang()

Properties: Player(class)

Properties: name(string) -> methods related: user()

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

In our original proposal in A11, we were lacking the menu system, and all it's functionalities/components, such as the save/load. We have a launcher type system already that allows the user to choose between, but we still need to add a default splash screen that has a timer to set the load time.

Design and Play options(added during A12, not proposed in A11), but they need to be modified to allow for the load, save and other menu bar functions.

In addition, color configuration will need to be re-added, since we weren't able to properly implement it for A12 submission. In A11, we proposed a JTextField for chat input, but since that isn't implemented anytime soon, we did not include it (yet).

Other misc. differences include using the ResourceBundles, Locales, new JFrames (Design, launcher/splash), JCheckBox.

* 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.*

Singleton

-- Singleton ensures that a class has one global identifier, and only one

instance of that class is allowed at a time. This DP is recommended because,

considering new game windows opened as objects (Design window, play window, etc),

the game SHOULDN'T have multiple of these open at a time. For example, if the "Play"

window is open, another one should not be opened under any circumstances. This will

also work great with MVC, since the MVC will make view, model, controller, and only

one instance of these 3 classes should be used. There shouldn't be multiple controllers,

view and model.

Composite

-- The composite design pattern treats similar objects as a single object, with an

overall tree like hierarchy. The game application can be seen as a tree hierarchy, with

the three classes (view, model, controller) acting as the main three branch from the main class,

holding their own components. By grouping similar objects, imposing changes such as updating

language is easier, by treating all JLabels as one object.

**References**

*[Include eventual references used here]*

<https://www.tutorialspoint.com/design_pattern/singleton_pattern.htm>

<https://www.tutorialspoint.com/design_pattern/composite_pattern.htm>

Algonquin College

Winter, 2022