CMPE 114/115

Fundamentals of Programming II

Section 05

**Memory Card Game – Flipping Tiles**

**Design Draft**

Authors:

Ahmad Fardous Azimi – (99825537334)

Ceren Kızılırmak – (14125057252)

Zeynep Çetin - (10162279930)

Efe Mert Güler – (44851959858)

Table of Contents

[Main Class 3](#_Toc134036925)

[Controller Class 4](#_Toc134036926)

[Board and Card Classes 5](#_Toc134036927)

[Shuffle Class 6](#_Toc134036928)

[Message Class 6](#_Toc134036929)

[Tracker Class 7](#_Toc134036930)

[Sound Class 7](#_Toc134036931)

[Program’s UML Diagram 8](#_Toc134036932)

[Workload Division 9](#_Toc134036933)

# **Main Class**

Diagram

Description automatically generated

The Main class inherits the Application class from the JavaFx library. Application class has two important methods:

* *start* Method: The main entry point for all JavaFX applications. The *start* method is called after the *init* method has returned and the system is ready for the application to run.

NOTE: This method is called on the JavaFX Application Thread.

* + Params: *primaryStage* – the primary stage for this application, onto which the application scene can be set. Applications may create other stages if needed, but they will not be primary stages.
  + Throws: Exception – if something goes wrong.
* *launch* Method: This is a static method that launches a standalone application. This method is typically called from the {Main *main*} method. It must not be called more than once, or an exception will be thrown.

The Main class overrides the *start* method and instantiates the *FXMLLOADER* class to create an instance of it to run the *FXML* file for the UI of the program, uses the Scene class to create an instance of the scene in order to display the content of the program visually by loading the *FXML* file, and uses the Stage class to create an environment for displaying the scene by using the *setScene* method and passing the scene object that has been created and by invoking the *show* method of the Stage class when running the program the content will be displayed. Also, there’s another method “*setTitle*” that we can set the title of the application by passing a string to it. Finally, in the main method of the Main class, we invoke the launch method to begin running the program.

# **Controller Class**

A picture containing text, screenshot, font, line

Description automatically generated

The primary class for controlling the whole UI (User Interface) of the application, which links the UI elements with their logic to make them interactive, is called the Controller class. The controller class implements the Initializable interface. This interface has been superseded by the automatic injection of location and resource properties into the controller. *FXMLLoader* will automatically call any suitably annotated no-arg to initialize the method defined by the controller.

* *initialize* Method:

It’s Called to initialize a controller after its root element has been completely processed.

* Params:

*location* – The *location* method is used to resolve relative paths for the root object, or null if the location is not known.

*resources* – The *resources* method is used to localize the root object, or null if the root object was not localized.

* *initialCardsView* Method: This method is a private method that is being invoked inside the *initial* method and sets the initial cards to view on the board which is a question mark image. Additionally, it uses two for-loops to iterate through the grid pane (two-dimensional array) of the cards to initialize the cards’ view.
* *checkCardsView* Method: This method is also called inside the *initialize* method each time we run the game, so, it attaches a mouse click event on each card and it calls the *changeCardView(event)* method that is receiving the event as a parameter and by clicking each card it flips the question mark image to a random image and finally it invokes the *setCardsView(selectedRow, selectedCol)* method inside the two nested for-loops which iterates over the array and by receiving the specific row and column of the card as params it set the view of that card. Also, this method checks, when two cards are open, whether they are matching or not. If yes, then it disables the cards from being used anymore and takes them out of the array by changing their color on the UI. If they don’t match, then, they are getting closed/flipped back to their initial state.

# **Board and Card Classes**

A screenshot of a computer

Description automatically generated with low confidence

The board parent class is responsible for representing the game board containing all cards. Managing game logic and controlling the game's flow are among its duties. The board class establishes a structure of cards and positions them on the board. Moreover, the Board class initializes a two-dimensional array with 4 rows and 4 columns and places the cards into the array that is the *gradePane* in JavaFx. Board class has the two following methods:

* *populateCards* Method: This method is being called inside the initialize method of the Controller class that populates the cards on the board randomly which uses the *shuffleCards* method, the static method of the Shuffle class that assigns a random image each time the game is being run.
* *isBoardFull* Method: This method is a private method that only lives inside the Board class and is only being called till the board is full of populating cards on the board.

The Card child class is the main blueprint of the cards that are being used to instantiate the cards on the board and has the following methods:

* *Card* Method: The constructor method is used each time when creating a card and it takes a name (value of the image) as a string, a row number, and a column number to assign the fields primarily.
* *getName* Method: This method returns the name or the value of the image of the card.
* *getRow* Method: This method returns the row of the card as an integer.
* *getColumn* Method: This method returns the column of the card as an integer.

# A picture containing text, screenshot, font, line Description automatically generated**Shuffle Class**

The Shuffle class has only one static method and uses Random Class from the Java Library.

* *shuffleCards* Method: This method takes *cards* as Card 2-D array type, *randomImage* as a string type as parameters, and uses the Random class to shuffle the cards on the board randomly each time it is called.

# A picture containing text, screenshot, font, line Description automatically generated**Message Class**

The main objective of this class is a message that will appear throughout the game that if you finish earlier or in the time interval (winning), you have won. If it is not cultivated during the time limit and is lost in the game, a message will be given to the player that they have lost on the screen. This class has one static method:

* *showMessage* Method: This method displays the sentences that will be shown on the screen when the game ends.

# A picture containing text, screenshot, font, line Description automatically generated**Tracker Class**

In the “Tracker” class, there will be the following methods:

* *trackMoves* Method: This method counts the moves that the player did. When the player chooses two cards, the *moves* variable counts it as one move. The player can see their moves during the game and at the end of the game on the screen.
* *startTimer* Method: This method will count one hundred twenty seconds since the player clicks the “start the game” button. Players can see it below the screen during the game. If the player can match all the cards in under one hundred twenty seconds (two minutes), the timer will stop; if the player can’t, they lose the game and when the player clicks to start the game button again, the timer will start to click again.

# A screenshot of a computer Description automatically generated with low confidence**Sound Class**

The “Sound” class is to keep the audio clips of our game. There is a *soundValue* string array that contains all the sounds, and a *soundValue* string variable that will be containing each audio clip according to the specific situation. There will be sounds of:

• When two cards are opened and they are not matched, it gives a wrong pair sound.

• Congratulation sound when the game ends and the player wins.

• Losing sound when the player loses the game for not finding the pairs under the given time limit.

This class will have the following methods:

* *Sound* Method: This is a constructor method used to create and instantiate a sound class and assign the *soundValue* and *haveWon* variables’ initial values.
* *play* Method: This method is invoked to play the specific sound in certain situations such as, while winning the game, losing the game, or not being able to find a matching pair.

# A screenshot of a computer Description automatically generated with medium confidence**Program’s UML Diagram**

# **Workload Division**

Since we are using JavaFx to build our project, obviously we will have Main and Controller classes as base classes for our project.

* Ahmad is in charge of building the Main, and Controller classes in full detail, including all their contained methods and fields, creating the class's UML diagram, and outlining the connections between these OOP-based classes.
* Zeynep precisely explains the Sound and Tracker classes including all related methods and fields.
* For the project, Ceren is in the task of creating the Message class and Shuffle class, which extends the Random class.
* Efe is responsible for writing the Board and Cards classes which are both interrelated to each other as parent and child classes.