MEMORY CARD GAME

Our team 3 members **(Sorted by ID Number)**		
section	**Student ID**	**Names**
20	20812020100185	يوسف محمد إبراهيم محمد احمد
19	20812020100546	وحيد ناصر حسن حسيني
15	20812020100783	محمد هشام محروس جمال
19	20812020100902	وليد عبد الوهاب هلال إبراهيم جمعة
14	20812020100922	محمد عبد الحليم محمد بكر
19	20812020100952	وليد الشوادفي عبدالله السيد
15	20812020101150	محمد وسيم محمد احمد
16	20812020101157	محمود مصطفي فرج الدمرداش
19	20812020101212	ياسر حسن حسن عبد الرازق
19	20812020102027	نشأت محمد عبدالعزيز عبدالرحيم

Main idea

A ***memory Card Game*** is a simple game starts with showing 6 images to the player each image is repeated two times in different places, the player should memorize the place of the similar images and after 5 seconds the images disappear and the player should guess their places right, the player has only 3 shots if he used them he loses and if he guesses all 6 images right he wins.

Implementation

Design

Our game contains only one default package. We didn't use any custom package and only one stage that we use to show all the four scenes .The four scenes are (Entry scene – Play scene – winning scene – losing scene).

1. Entry scene.
   Contains Two buttons one called "Play" to start the game and one called "Exit" to close the game, a Background for the game (we downloaded it from the internet), and a music playing.

2. Play scene.
   Here we have our game. There are the 12 image Viewers (representing the cards of the game) that contains the 6 repeated images and two buttons in the bottom one to restart the game and one to go to the entry scene and 6 labels to display the number of guesses, misses and a timer.

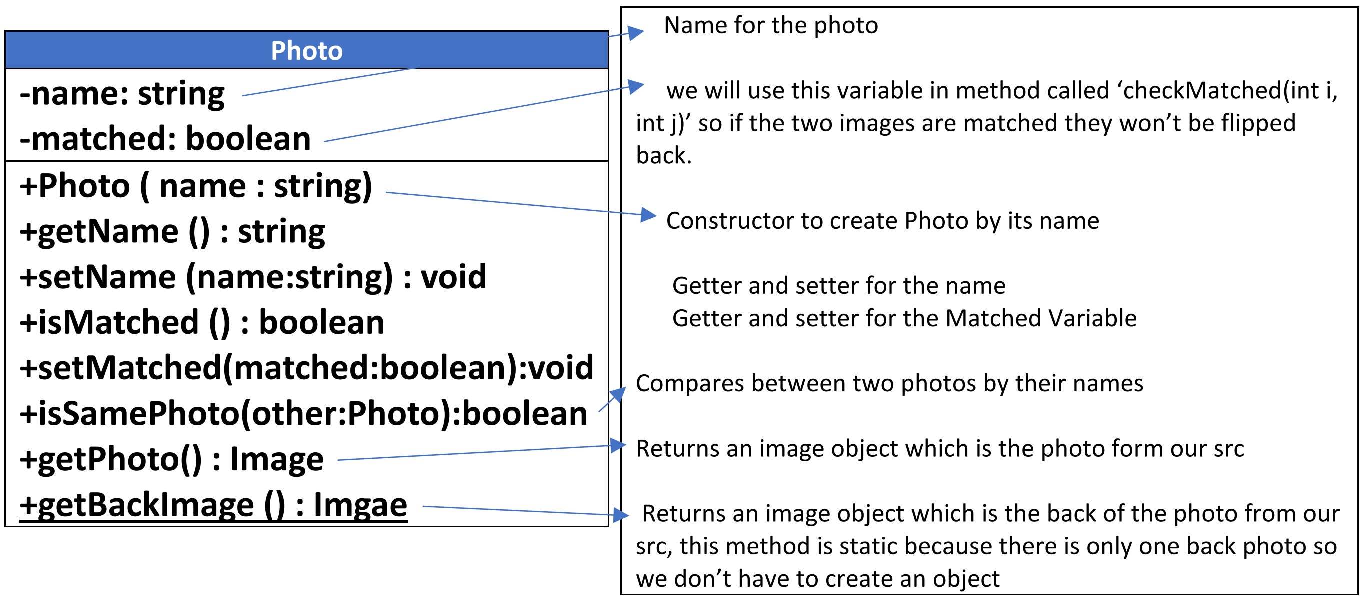
3. Winning scene
   When the player wins this scene appears it contains a text and a background that describe that the player's winning, and two buttons one to play again and another to go to the entry and a little sound effect.

4. Losing scene.
   When the player loses this scene appears, it contains a background that describes that the player's losing, and two buttons one to play again and another to go to the entry and a little sound effect.

Classes

We created two classes, (***‘Photo’*** – ***‘PhotoViewer’***) class. That will represent the photos that we use, and we put our game in a class called ***‘MemoryCard’*** that contains our start method, main methods, the GUI and All the methods used to creates the game

Photo Class



Photo

-name: string
-matched: boolean

+Photo (name : string)
+getName () : string
+setName (name:string) : void
+isMatched () : boolean
+setMatched(matched:boolean):void
+isSamePhoto(other:Photo):boolean
+getPhoto() : Image
+getBackImage () : Imgae

Name for the photo

we will use this variable in method called ‘checkMatched(int i, int j)’ so if the two images are matched they won’t be flipped back.

Constructor to create Photo by its name

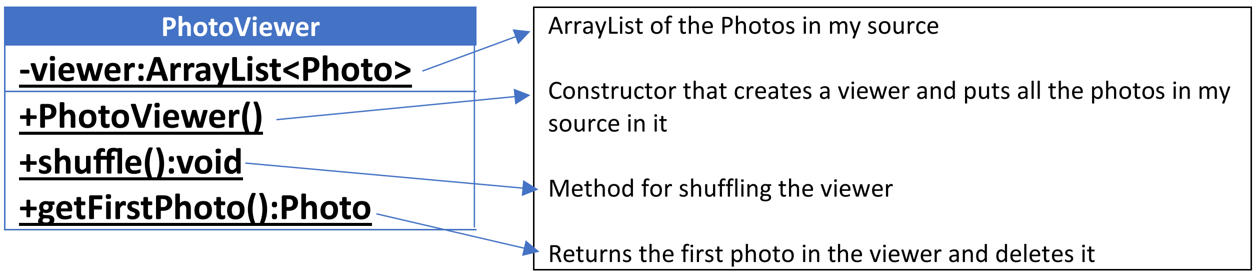
Getter and setter for the name
Getter and setter for the Matched Variable

Compares between two photos by their names

Returns an image object which is the photo form our src

Returns an image object which is the back of the photo from our src, this method is static because there is only one back photo so we don’t have to create an object

PhotoViewer Class



PhotoViewer

-viewer:ArrayList<Photo>

+PhotoViewer()
+shuffle():void
+getFirstPhoto():Photo

ArrayList of the Photos in my source

Constructor that creates a viewer and puts all the photos in my source in it

Method for shuffling the viewer

Returns the first photo in the viewer and deletes it

Steps

1. We made our GUI in the start method.

2. We ceated the ‘LetsPlay()’ method that starts the game,so and button will start the game should has this method in its setOnAction handling.

LetsPlay() methode :

we reset almost everything in the game (timer, counter, mediaPlayer, the listener of the imagViews….etc.) and
we set two Photo object ‘Photo1’, ‘Photo2’ to null then we create our viewer and shuffle it and create an ArrayList<Photo> called ***‘photosInGame’*** to take the first 6 photos from the viewer each photo two times and then shuffle them, so we have 12 shuffled photos to use. Then we use a timer to count 5 seconds and then hide the photos using ‘***hidePhotos()***’ method and give the player the ability to click on the imageViews using setOnMouseClick() method

3. Then we created the logic of the game using three different methods (showPhoto(int i) - hidePhotos() – chechMatch(int i, int j)).

And simply the logic is:

when we click on an imageView we call the ***showPhoto(int i)*** and i is the index of the imageView on the photoPane,
In the first click we hide all the unmatched photos (here all photos) and then take the photo that has the index of the imageView from the ***‘photosInGame’*** arrayList and put it in ‘photo1’ and display it and we save the index in variable ‘photo1Index’ to prevent the player from click on it again.
In the second click we take the photo that has the index of the imageView we clicked on from the ***‘photosInGame’*** arrayList and put it in ‘photo2’ and display it, and we save the index in variable ‘photo2Index’
and immediately we check if 'photo1' and 'photo2' are the same or not using 'checkMatch(int i,int j)' method if they are the same we change their ‘matched’ datafield to true and disable the imageViews that contain them and set ‘photo1’ and ‘photo2’ to null again in any case
In the third click we hide all unmatched photos (here if the last to photos were matched, they won’t hide) and then repeat click one.

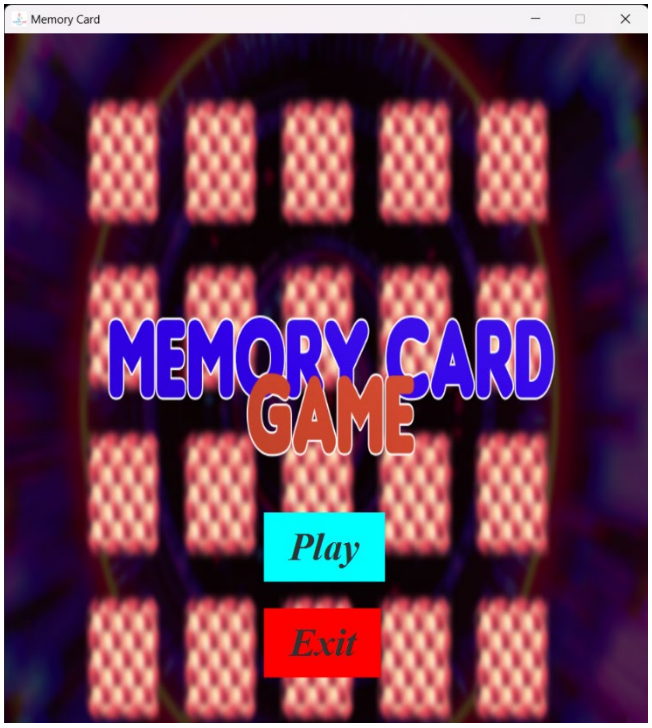
Running the program (Screenshots and Comments)

1. Entry Scene

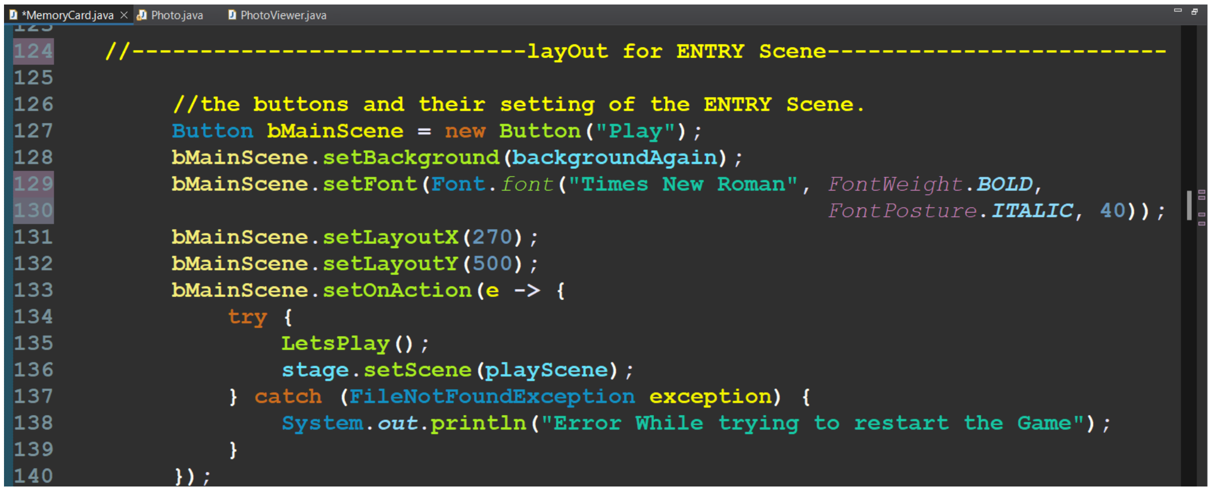
when we run the Program the entry scene pops up with music in the Background image and 2 buttons on it

- Play Button
  Which takes us to the Play Scene.

- Exit Button
  Which end the Game.



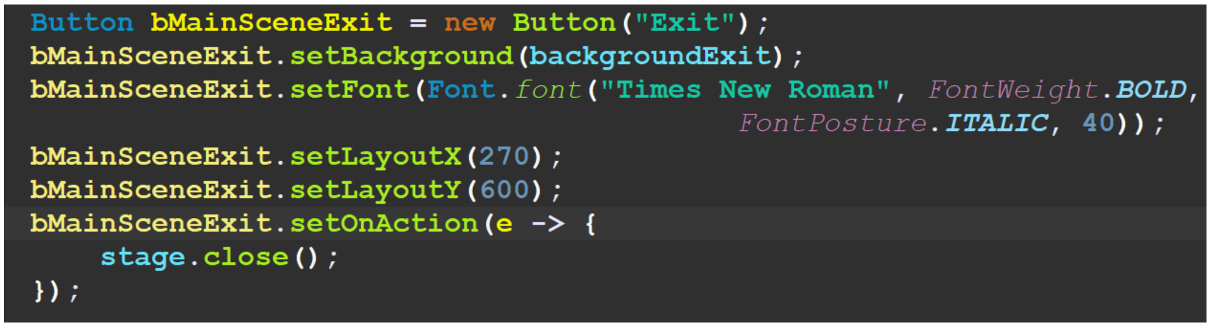
// The Code of the Entry Scene



```java
    //----------------------------layOut for ENTRY Scene--------------------------

        //the buttons and their setting of the ENTRY Scene.
        Button bMainScene = new Button("Play");
        bMainScene.setBackground(backgroundAgain);
        bMainScene.setFont(Font.font("Times New Roman", FontWeight.BOLD,
                                                        FontPosture.ITALIC, 40));
        bMainScene.setLayoutX(270);
        bMainScene.setLayoutY(500);
        bMainScene.setOnAction(e -> {
            try {
                LetsPlay();
                stage.setScene(playScene);
            } catch (FileNotFoundException exception) {
                System.out.println("Error While trying to restart the Game");
            }
        });
```

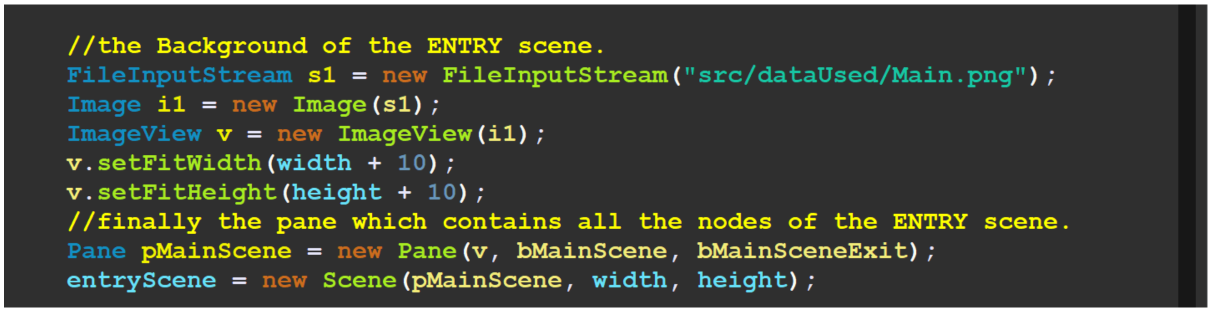
1) *The play Button code*
   - We gave it a name bMainScene and we adjusted the font, the color and the position
   - We programmed it so when we press it , it takes us to the PlayScene and calls the function let's play() that from which starts the game.



```
Button bMainSceneExit = new Button("Exit");
bMainSceneExit.setBackground(backgroundExit);
bMainSceneExit.setFont(Font.font("Times New Roman", FontWeight.BOLD,
                                        FontPosture.ITALIC, 40));
bMainSceneExit.setLayoutX(270);
bMainSceneExit.setLayoutY(600);
bMainSceneExit.setOnAction(e -> {
    stage.close();
});
```

2) *The Exit Button code*

- We gave it a name bMainSceneExit and we adjusted the font, the color, and the position.
- We programmed it so when we press it, it closes the stage and ends the program.



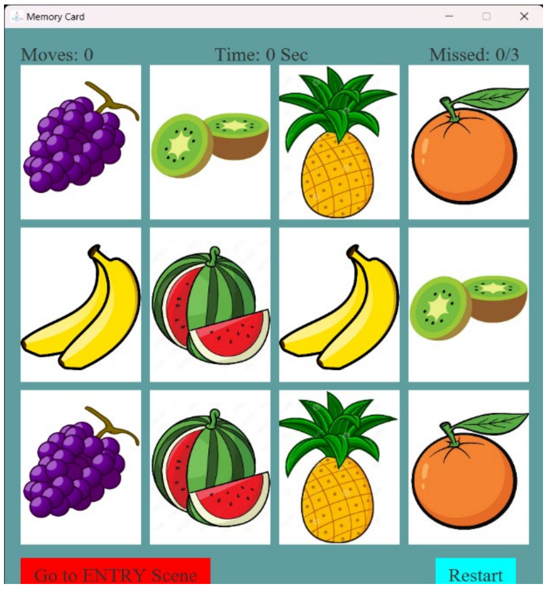
```
//the Background of the ENTRY scene.
FileInputStream s1 = new FileInputStream("src/dataUsed/Main.png");
Image i1 = new Image(s1);
ImageView v = new ImageView(i1);
v.setFitWidth(width + 10);
v.setFitHeight(height + 10);
//finally the pane which contains all the nodes of the ENTRY scene.
Pane pMainScene = new Pane(v, bMainScene, bMainSceneExit);
entryScene = new Scene(pMainScene, width, height);
```

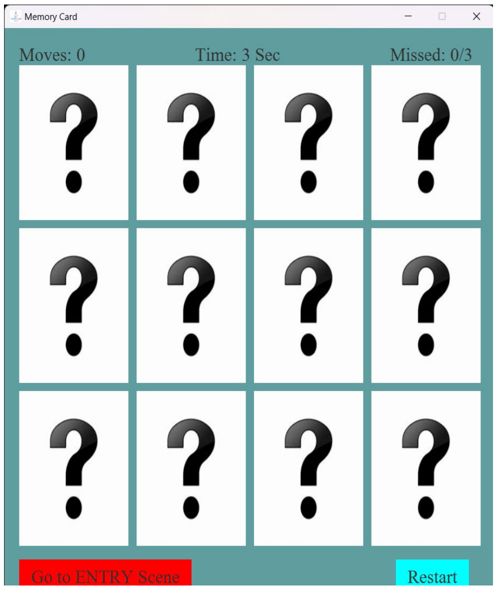
3) *The Background of the Entry Scene code*

- We took an image method Path and we set it as image and we put that image on an Image view as Shown.
- After we finished with the entry Scene, we set it on a regular pane that put on a scene and we called it entryScene.

2. Play Scene

- When we press the play button it takes us to the Play scene which consists of:-
    - ➢ 12 images that are displayed for few seconds and then they get flipped or if I should say back cards are added on them.
    - ➢ 2 Buttons Go to ENTRY Scene and Restart.
    - ➢ 3 labels move, timer and missed





// The Code of the Play Scene

```
                        //LayOut for Play Scene
/**
 * first we add 12 ImageViews in the "PhotoPane" FlowPane using loop and
 * the method "addImageView()" in line
 * ............................................................. so now
 * there are 12 imageViewers exists in the flow pane i.e. the 12 cards
 * represent the game deck
 */
PhotoPane.setHgap(10);
PhotoPane.setVgap(10);
for (int i = 0; i < 12; i++) {
    PhotoPane.getChildren().add(addImageView());
}
```

```
//Adding ImageView to "PhotoPane" FlowPane
public static ImageView addImageView() {
    ImageView imageView = new ImageView();
    imageView.setFitWidth(150);
    imageView.setFitHeight(200);
    return imageView;
}
```

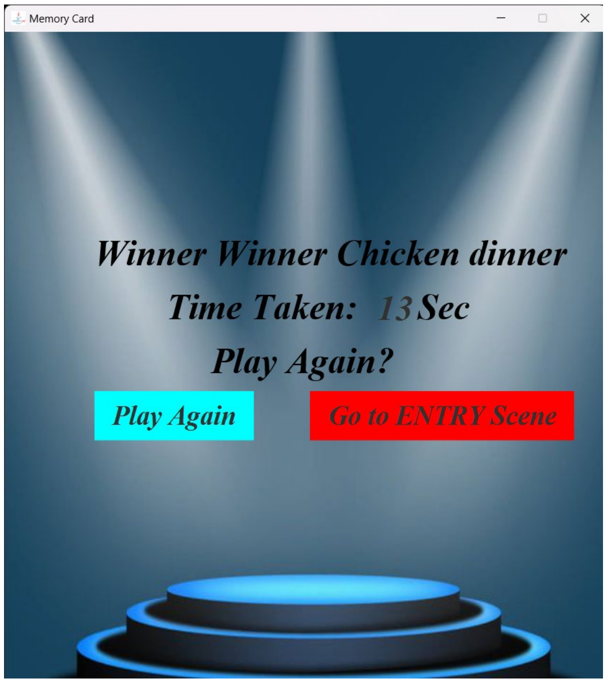
1) *Creating Image Views to have the Images in them: -*

So basically what this part is doing is adding 12 image view using the method in the second figure add Image View() with the given width and height on the photopane which is a FlowPane with vertical Orientation

2)*invoking the let's play method:*

The method does basically 3

3. Win Scene



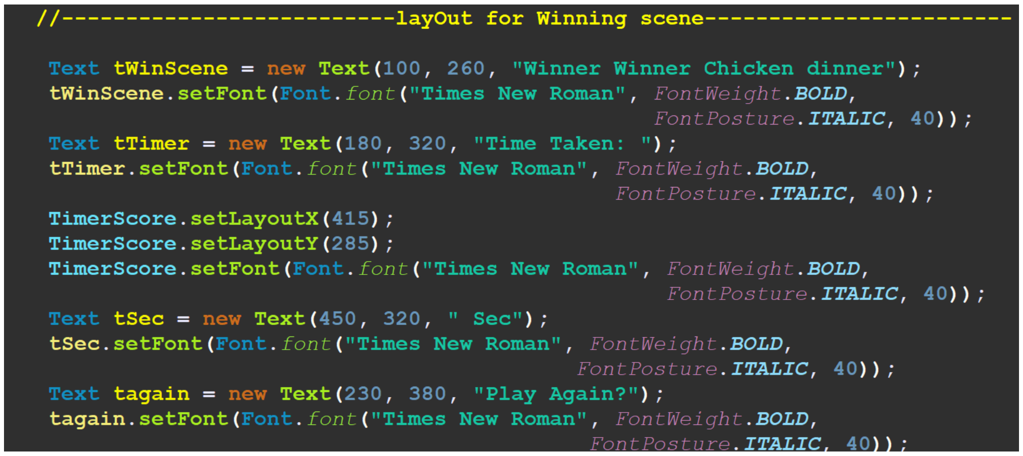
The win Scene consists of

- 4 Texts and 1 label..
- 2 buttons
- 1 Image

// The code of the Win scene

1) *The labels code :*

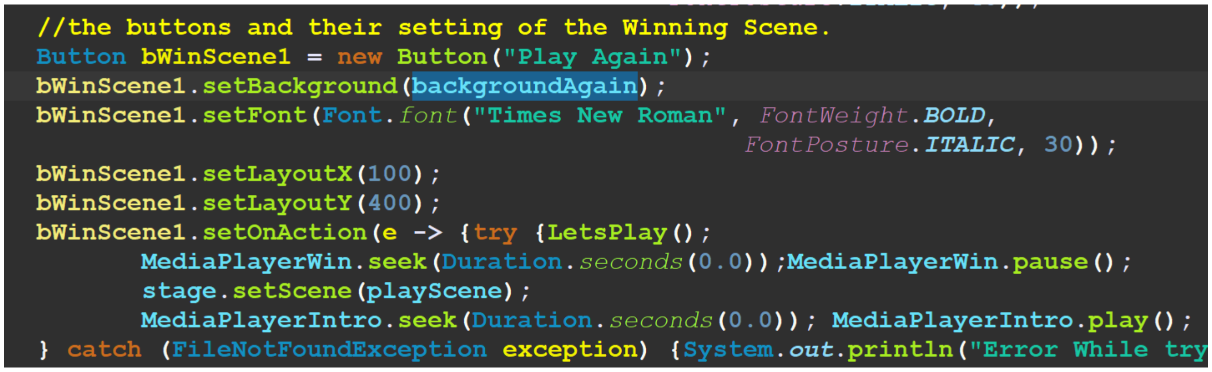
- we declared the 3 texts, the tWinScene with the shown properties , tTimer ,the TimerScore label that inform the user about the time taken for finishing the game by taking the value of the timer as a text and show it on the screen , the text tsec that the word “Sec” next to the TimerScore and the text tagain .



```
//-------------------------layOut for Winning scene------------------------

 Text tWinScene = new Text(100, 260, "Winner Winner Chicken dinner");
 tWinScene.setFont(Font.font("Times New Roman", FontWeight.BOLD,
                                                 FontPosture.ITALIC, 40));
 Text tTimer = new Text(180, 320, "Time Taken: ");
 tTimer.setFont(Font.font("Times New Roman", FontWeight.BOLD,
                                              FontPosture.ITALIC, 40));
 TimerScore.setLayoutX(415);
 TimerScore.setLayoutY(285);
 TimerScore.setFont(Font.font("Times New Roman", FontWeight.BOLD,
                                                  FontPosture.ITALIC, 40));
 Text tSec = new Text(450, 320, " Sec");
 tSec.setFont(Font.font("Times New Roman", FontWeight.BOLD,
                                            FontPosture.ITALIC, 40));
 Text tagain = new Text(230, 380, "Play Again?");
 tagain.setFont(Font.font("Times New Roman", FontWeight.BOLD,
                                              FontPosture.ITALIC, 40));
```

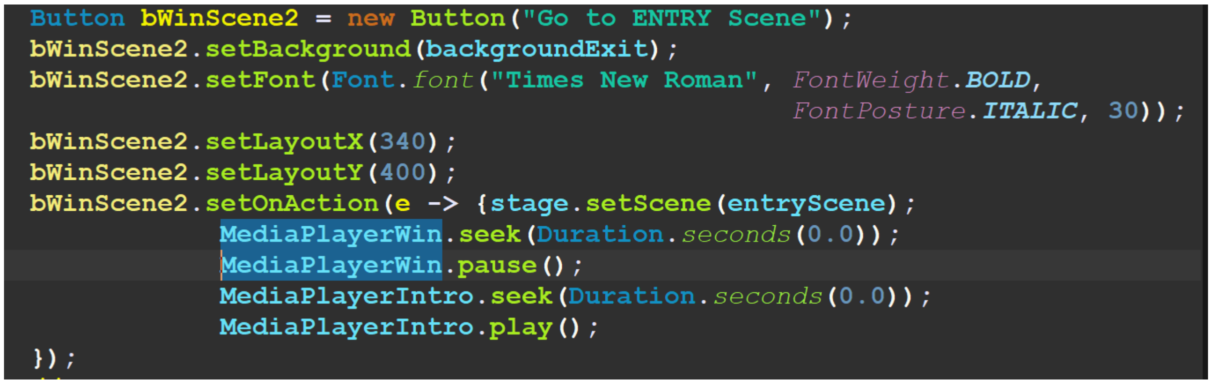
2) *The “Play again” Button code :*



```
 //the buttons and their setting of the Winning Scene.
 Button bWinScene1 = new Button("Play Again");
 bWinScene1.setBackground(backgroundAgain);
 bWinScene1.setFont(Font.font("Times New Roman", FontWeight.BOLD,
                                                  FontPosture.ITALIC, 30));
 bWinScene1.setLayoutX(100);
 bWinScene1.setLayoutY(400);
 bWinScene1.setOnAction(e -> {try {LetsPlay();
         MediaPlayerWin.seek(Duration.seconds(0.0));MediaPlayerWin.pause();
         stage.setScene(playScene);
         MediaPlayerIntro.seek(Duration.seconds(0.0)); MediaPlayerIntro.play();
 } catch (FileNotFoundException exception) {System.out.println("Error While try
```

So basically, what it does is when you click it , it takes you to the play scene and stop the winning music and reset and play the intro music. And the properties of it are shown.

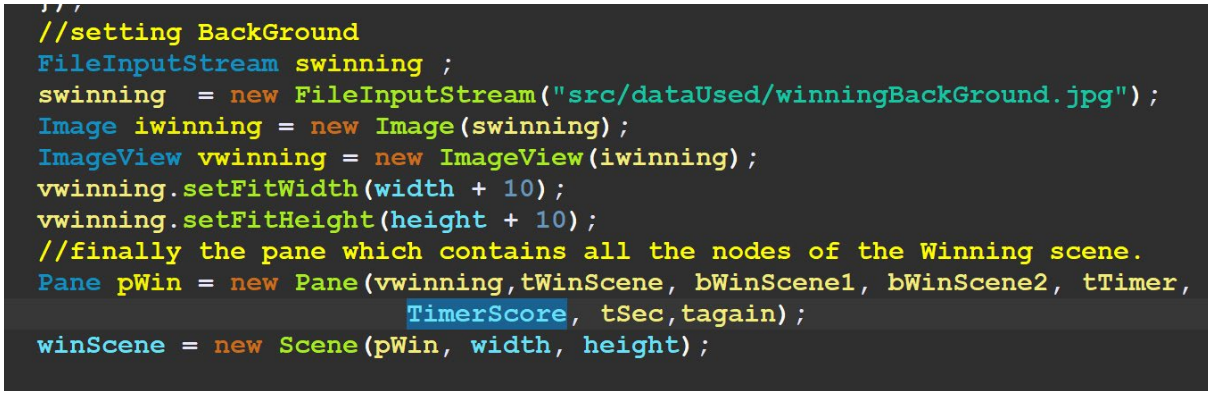
3) *The “Go To Entry Scene” Button code :*



```
 Button bWinScene2 = new Button("Go to ENTRY Scene");
 bWinScene2.setBackground(backgroundExit);
 bWinScene2.setFont(Font.font("Times New Roman", FontWeight.BOLD,
                                                  FontPosture.ITALIC, 30));
 bWinScene2.setLayoutX(340);
 bWinScene2.setLayoutY(400);
 bWinScene2.setOnAction(e -> {stage.setScene(entryScene);
             MediaPlayerWin.seek(Duration.seconds(0.0));
             MediaPlayerWin.pause();
             MediaPlayerIntro.seek(Duration.seconds(0.0));
             MediaPlayerIntro.play();
 });
```

So basically, what it does is when you click it , it takes you to the entry scene and stop the winning music and reset and play the intro music. And the properties of it are shown.

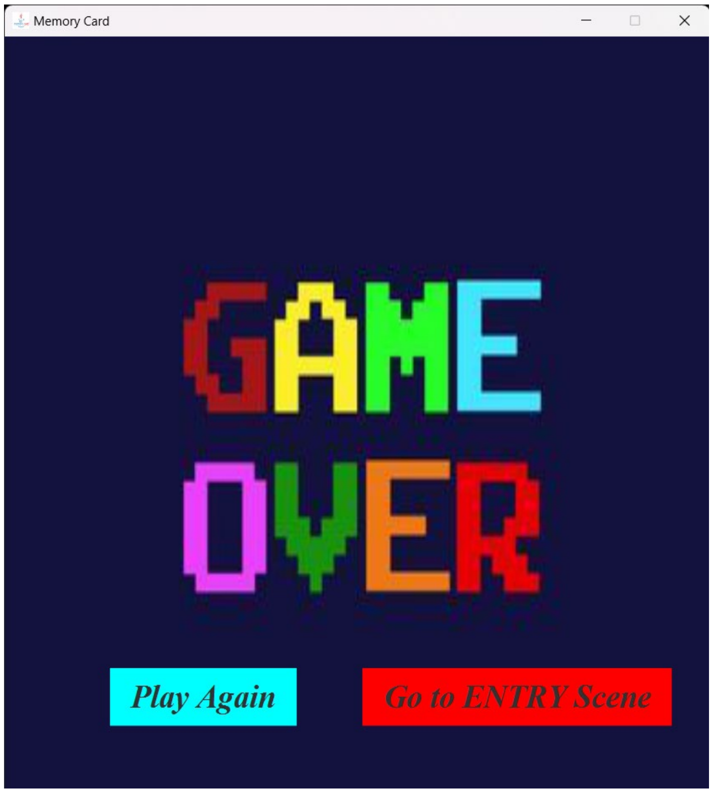
4) *__the Background code :__*



```
//setting BackGround
FileInputStream swinning ;
swinning  = new FileInputStream("src/dataUsed/winningBackGround.jpg");
Image iwinning = new Image(swinning);
ImageView vwinning = new ImageView(iwinning);
vwinning.setFitWidth(width + 10);
vwinning.setFitHeight(height + 10);
//finally the pane which contains all the nodes of the Winning scene.
Pane pWin = new Pane(vwinning,tWinScene, bWinScene1, bWinScene2, tTimer,
                        TimerScore, tSec,tagain);
winScene = new Scene(pWin, width, height);
```

We imported an Image and set it on an image view called vwinning with the given properties , then we added all the components on a regular pane and then we put the pane on our winscene.

4. Lose Scene

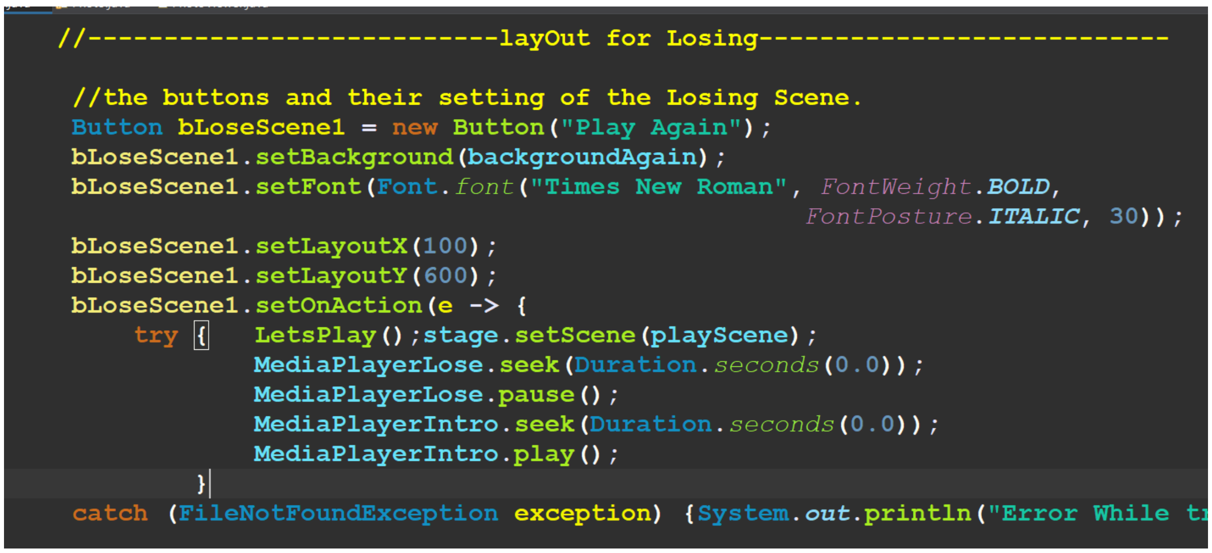


The lose Scene consists of :-

- 2 buttons
- 1 Image

// The code of the Lose scene

1) *__The "play again" button code :__*

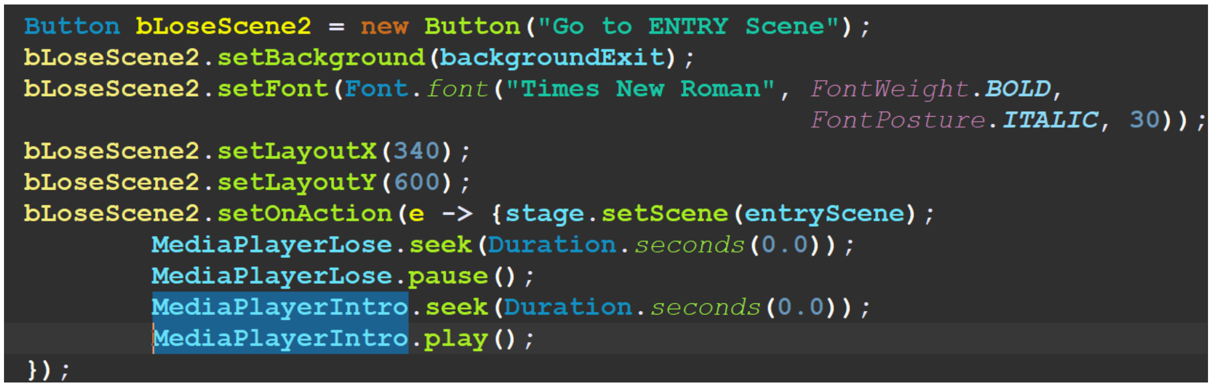


```
//--------------------------layOut for Losing--------------------------

 //the buttons and their setting of the Losing Scene.
 Button bLoseScene1 = new Button("Play Again");
 bLoseScene1.setBackground(backgroundAgain);
 bLoseScene1.setFont(Font.font("Times New Roman", FontWeight.BOLD,
                                                  FontPosture.ITALIC, 30));
 bLoseScene1.setLayoutX(100);
 bLoseScene1.setLayoutY(600);
 bLoseScene1.setOnAction(e -> {
     try {    LetsPlay();stage.setScene(playScene);
              MediaPlayerLose.seek(Duration.seconds(0.0));
              MediaPlayerLose.pause();
              MediaPlayerIntro.seek(Duration.seconds(0.0));
              MediaPlayerIntro.play();
         }
 catch (FileNotFoundException exception) {System.out.println("Error While t
```

So basically, what it does is when you click it , it takes you to the play scene and stop the losing music and reset and play the intro music. And the properties of it are shown.

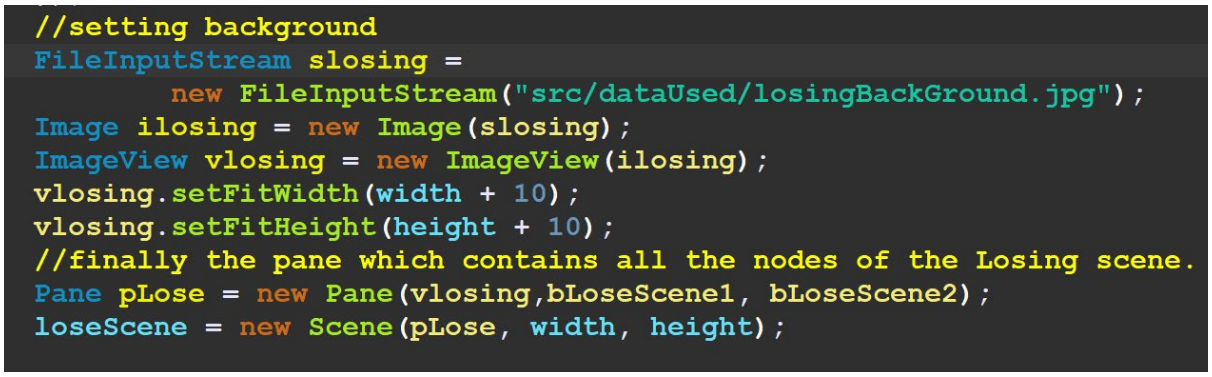
2) *__The "Go to ENTRY Scene" Button code :__*



```
Button bLoseScene2 = new Button("Go to ENTRY Scene");
bLoseScene2.setBackground(backgroundExit);
bLoseScene2.setFont(Font.font("Times New Roman", FontWeight.BOLD,
                                                 FontPosture.ITALIC, 30));
bLoseScene2.setLayoutX(340);
bLoseScene2.setLayoutY(600);
bLoseScene2.setOnAction(e -> {stage.setScene(entryScene);
        MediaPlayerLose.seek(Duration.seconds(0.0));
        MediaPlayerLose.pause();
        MediaPlayerIntro.seek(Duration.seconds(0.0));
        MediaPlayerIntro.play();
});
```

So basically, what it does is when you click it , it takes you to the entry scene and stop the Losing music and reset and play the intro music. And the properties of it are shown.

3) *__The "Go to ENTRY Scene" Button code :__*



```
//setting background
FileInputStream slosing =
        new FileInputStream("src/dataUsed/losingBackGround.jpg");
Image ilosing = new Image(slosing);
ImageView vlosing = new ImageView(ilosing);
vlosing.setFitWidth(width + 10);
vlosing.setFitHeight(height + 10);
//finally the pane which contains all the nodes of the Losing scene.
Pane pLose = new Pane(vlosing,bLoseScene1, bLoseScene2);
loseScene = new Scene(pLose, width, height);
```

4) *the Background code :*
We imported an Image and set it on an image view called vlosing with the given properties , then we added all the components on a regular pane and then we put the pane on our losescene.

Difficulties and how we solved them.

1. Repeated clicking.

The difficulty

If the player clicked on the same imageView many times this imageView and the other imageView that contains the same photo as the first imageView will show the images and check them as matched (the player can win only by clicking at 6 different imageViews without guessing)

How we solved it

We created an integer variable called 'photo1Index' and put the index of the first imageView in it then we put this variable in if statement of the second click so that the index of the second imageView should be different than the first imageView.

2. Disabling the imageViews that contains matched photos.

The difficulty

When you make a match the two photos would remain if you clicked in other imageView, the problem is that if you clicked on these photos again the program will count it as a move and make a missed, in other words, we need to disable the imageViews that contain matched photos

How we solved it

We used the ‘photo1Index’ and another variable ‘photo2Index’ send them to the ‘chechMatch(int i, int j)’ method and if the two photos are matched we make the setOnMouseclicked() of these two imageView that have the two indexes empty, so they will not do anything if the player clicked on them

3. Disabling the imageViews at the first 5 seconds of the game

The difficulty

During the show of the photos for the player to remember them (first 5 seconds of the game) the player can click on any imageView and so the rest will hide the photos and cause error

How we solved it

This difficulty was quite easy, we make the setOnMouseclicked() of these two imageView that have the two indexes empty in the beginning of the ‘LetsPlay()’ method before the counter ends , so they will not do anything if the player clicked on them but when the counter finishes they will be available to be clicked on.

Things we tried to achieve but were quite difficult.

- Set animation to the “Entry scene” were the label holding the game’s name (i.e., Memory Card Game) then the Entry scene itself shows up.
- During the match, as the player picks two cards which are not matched, he must flip the third card so that the previous two get back to their “Unflapped condition”.so we tried to set a time limit for those two cards so they wouldn’t have to wait till a third card is flipped.
- We tried to set an additional scene for the game setting which gives the user the ability to change some features of the game like the volume of the sound running in each scene & the match difficulty.
- We tried to add the ability for the user to have a continuous match so when he finishes the first match, the next match automatically begins with a higher level of difficulty till he beats the highest level (more like a tournament).

The Improvements to be added in the future.

Regarding The GUI

1. We shall allow the stage to be resizable and have the ability to get full screen mode.
2. We shall add an animation for flipping the selected card.
3. We shall improve the style of the buttons.

Regarding Playing the Game

1. We shall provide an “End Game Timer” that will end the match after a specific period.
2. We shall provide multi levels of difficulty and have the user choose among them before the game.
    a. The levels of difficulty will differ with respect to
        i. the time the user will have to see “The Match Deck” before the cards are flipped to their back.
        ii. The number of the cards of “The Match Deck”.
        iii. The End Time we will set for the “End Game Timer”.
3. We shall provide effect for getting two cards matched.

References

All the Ideas used in this project have been taken from this playlist however, we didn’t copy anything we just got the ideas.

https://youtube.com/playlist?list=PLoodc-fmtJNbeL8P1DizFcgjp62UjvJ3t