

JAVA PROJECT REPORT

(Project Term January-May 2023)

MEMORY MATCH MANIA: A FUN AND CHALLENGING BRAIN EXERCISE!

Submitted by

Name: Shah Arsalan

Registration Number: 12102195

Course Code: CSE310

Under the Guidance of

Dr.A.Ranjith Kumar

School of Computer Science and Engineering



LOVELY
PROFESSIONAL
UNIVERSITY

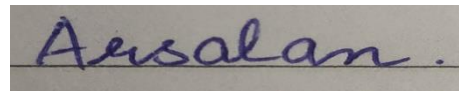
DECLARATION

I hereby declare that the project work entitled (“Memory Match Mania: A Fun and Challenging Brain Exercise! : - A Pair Matching Game”) is an authentic record of my own work carried out as requirements of Capstone Project for the award of B.Tech degree in Computer Science and Engineering from Lovely Professional University, Phagwara, under the guidance of Dr.A.Ranjith Kumar, during January to May 2023. All the information furnished in this capstone project report is based on my own intensive work and is genuine.

Name of Student: Shah Arsalan

Registration Number: 12102195

Signature:

A rectangular box containing a handwritten signature in blue ink that reads "Arsalan.".

Date:19-04-2023

TABLE OF CONTENTS

- I. Introduction
- II. Scope of Project
- III. Modules:
 - Login
 - Level (where user can select the level of difficulty)
 - Game
 - Pair
- IV. Sample code
- V. Outcome of Project
- VI. Conclusion

INTRODUCTION

Pair matching game is a popular and entertaining puzzle game that challenges the player's memory skills. The game involves flipping over pairs of cards to reveal hidden images, with the objective of finding all matching pairs. In this game, the player needs to remember the location of different cards with hidden images and match them in pairs by flipping them over. The number of cards and the complexity of the game can vary, depending on the level of difficulty. In this project, I will be developing a pair matching game using Java programming language. The game will be designed to run on desktop computers and will involve creating a graphical user interface (GUI) using Java Swing. To start, we will create a login page which will then redirect us to the GUI of the game. Then we will design the graphical interface. We will then develop the logic for generating the card pairs, randomizing their positions, and checking for matches when flipped. If two cards match, then score will increase. The game will also consist of a timer. The game will automatically terminate after the allocated time ends. Java is a widely used programming language for creating desktop applications and games. It provides a rich set of libraries and tools for developing graphical user interfaces, which makes it an ideal choice for creating a pair matching game. With Java, we can create a visually appealing and interactive game that can be run on multiple platforms. Overall, this project will provide me a great opportunity to enhance my Java programming skills while creating an engaging and interactive game. Pair matching game, also known as memory game or concentration game, is a classic puzzle game that has been popular among people of all ages for decades. The game is not only entertaining but also a great way to improve memory, concentration, and cognitive abilities.

SCOPE OF THE PROJECT

The scope of the project is to create a pair matching game using Java programming language. The game will be designed to run on desktop computers and will involve developing a graphical user interface (GUI) using Java Swing.

The key features of the game will include:

- A login page which will only redirect us to the GUI if the entered username and password is correct.
- Different levels of difficulty.
- A game board consisting of a grid of cards with hidden images.
- The ability to flip over cards by clicking on them to reveal the hidden image.
- Randomized placement of the card pairs on the game board.
- The ability to detect matching pairs and remove them from the game board.
- The ability to terminate the game after the time finishes.
- A scoring system that tracks the player's score.
- The ability to restart the game and reshuffle the cards.
- The game will have multiple levels of difficulty, with different grid sizes and numbers of card pairs.

The focus will be on developing a well-designed and functional pair matching game that demonstrates proficiency in Java programming and GUI development.

MODULES

LOGIN

```
class LoginFrame extends JFrame implements ActionListener {  
    private JTextField userField;  
    private JPasswordField passwordField;  
    private JButton loginButton;
```

This code above is for a simple login page using Java. It creates a JFrame with a JPanel inside that contains a JLabel for the username, a JTextField for the user to enter their username, a JLabel for the password, a JPasswordField for the user to enter their password, and a JButton for the user to click to login.

```
FocusListener highlighter = new FocusListener() {  
  
    @Override  
    public void focusGained(FocusEvent e) {  
        e.getComponent().setBackground(Color.cyan);  
    }  
  
    @Override  
    public void focusLost(FocusEvent e) {  
        e.getComponent().setBackground(UIManager.getColor("TextField.back  
ground"));  
    }  
};  
  
userField.addFocusListener(highlighter);  
passwordField.addFocusListener(highlighter);  
loginButton.addFocusListener(highlighter);
```

The code includes a FocusListener that changes the background color of the text fields and button when they are in focus.

```
public void actionPerformed(ActionEvent ae) {  
    String username = userField.getText();  
    String password = new String(passwordField.getPassword());  
    boolean matched=false;
```

```

try{

    FileReader f=new FileReader("account.txt");
    int c;
    String line="";
    while ((c = f.read()) != -1)
    {
        if (c == '\n')
        {
            String line2=username+password;
            line = line.replaceAll("\\s+", "");
            if(line.equals(line2))
            {
                matched=true;
                break;
            }
            line = "";
        }
        else {
            line += (char)c;
        }
    }
    f.close();
    if(matched){
        new Window(username);
        dispose();
    }
    else{
        JOptionPane.showMessageDialog(this, "INVALID USERNAME OR
PASSWORD");
    }
}catch(Exception e){}
}

```

When the user clicks the login button, the actionPerformed method is called, which reads from a text file named "account.txt" to check if the entered username and password match any account in the file. If there is a match, a new window is created with the username passed as an argument to a Window constructor and the current LoginFrame is disposed of. If there is no match, an error message is displayed using JOptionPane.

DIFFICULTY WINDOW

(Where user can select the difficulty of the game)

```
public Window(String name) throws HeadlessException {

    setDefaultCloseOperation(EXIT_ON_CLOSE);
    setTitle(name + " : Choose Difficulty Level");
    setBackground(Color.lightGray);
    setLocation(500, 300);
    setSize(500, 280);
    //setResizable(false);
    JPanel levelsPanel = new JPanel(new FlowLayout());
    JButton btnEasyLevel = new JButton("LEVEL-EASY 4x4");
    btnEasyLevel.addActionListener(a -> {
        this.dispose();
        new Game(4 , name);
    });
    JButton btnHardLevel = new JButton("LEVEL-HARD 6x6");
    btnHardLevel.addActionListener(a -> {
        this.dispose();
        new Game(6,name);
    });
}
```

This code defines a Window class that extends the JFrame class from the javax.swing package. The class constructor takes a String argument name which is used to set the title of the window. The Window class is used to display a window to choose the difficulty level of a game.

The JPanel is used to create a panel that contains two JButton components that represent the easy and hard levels of the game, respectively. The panel is set to use FlowLayout and is added to the window using the add method.

The JButton components have an action listener that disposes of the current window and creates a new Game object, passing the appropriate difficulty level and player name as parameters.


```

ImageIcon img=new ImageIcon(ClassLoader.getResource("diff.jpg"));
    Image scaledImg = img.getImage().getScaledInstance(400, 150,
Image.SCALE_SMOOTH);
    ImageIcon scaledIcon = new ImageIcon(scaledImg);
    JLabel image = new JLabel(scaledIcon);

```

ImageIcon is used to load an image that is displayed at the top of the panel. The image is scaled to fit the panel dimensions using Image.SCALE_SMOOTH.

```

levelsPanel.setBorder(BorderFactory.createLineBorder(Color.BLACK));
    levelsPanel.setBorder(BorderFactory.createTitledBorder(BorderFactory.crea
teEtchedBorder(),"Choose Level"));

```

The JPanel is given several border styles using the setBorder method to create a visually appealing panel with a title.

GAME WINDOW

```

public Game(int len , String name2) throws HeadlessException {
    this.name3=name2;
    ArrayList<ImageIcon> listOfImages = new ArrayList<ImageIcon>();
    for (int i = 0; i <= 24; i++) {
        listOfImages.add(new ImageIcon(i + ".jpg"));
    }
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    if (len == 4) {
        setTitle("EASY LEVEL");
    } else if (len == 6) {
        setTitle("HARD LEVEL");
    }
}

```

The game has two levels, "EASY" and "HARD", which correspond to grid sizes of 4x4 and 6x6, respectively.

```

ImageIcon timerIcon = null;
    java.net.URL imgURL1 = Game.class.getResource("timer.jpg");
    if (imgURL1 != null) {
        timerIcon = new ImageIcon(imgURL1);
    }

    timerLabel = new JLabel(timerIcon);
    timerLabel.setText("01:59");
btn.addActionListener(a -> {
    if (!isTimerStart) {
        int interval = 1000;
        int duration = 120;
        Timer timer = new Timer(interval, new
TimerListener(duration, this));
        timer.start();
        isTimerStart = true;
    }
}

```

The game also has a timer that starts when the player clicks on the first button and counts down from 2 minutes.

```

JButton exitButton = new JButton(arrowIcon);
exitButton.addActionListener(e -> {
    dispose(); // Close the JFrame
    JOptionPane.showMessageDialog(this, "You chose to Quit, SCORE="
"+score);
});

```

The player can choose to exit the game at any time, and a score is displayed when the game is closed.

```

public int[][] assignImages(int len) {
    int n = len * len / 2;
    int[][] imgs = new int[len][len];
    Random rnd = new Random();
    int[] arr = new int[25];
    for (int i = 1; i <= 24; i++) {
        arr[i] = i;
    }
    int ind = 0;
    for (int i = 0; i < n; i++) {
        while (true) {

```

```

        arr[ind] = 0;
        ind = rnd.nextInt(arr.length - 1);
        if (arr[ind] != 0)
            break;
    }
    Pair pair1 = new Pair().getRandomPair(imgs);
    imgs[pair1.getX()][pair1.getY()] = ind;
    Pair pair2 = new Pair().getRandomPair(imgs);
    imgs[pair2.getX()][pair2.getY()] = ind;
}
return imgs;
}

```

The main game logic is implemented in the assignImages which assigns images to the buttons. and handle the button clicks, respectively.

```

JButton[][] buttons = new JButton[len][len];
ArrayList<Pair> pairs = new ArrayList<>();
for (int i = 0; i < len; i++) {
    for (int j = 0; j < len; j++) {
        buttons[i][j] = new JButton();
        buttons[i][j].setIcon(listOfImages.get(0));
        buttons[0][0].setBorder(BorderFactory.createTitledBorder("Click
to reveal"));

        JButton btn = buttons[i][j];
        btn.setBorder(BorderFactory.createEtchedBorder(0));
        btn.setBackground(Color.cyan);
        int finalI = i;
        int finalJ = j;
        Pair pair = new Pair();
        btn.addActionListener(a -> {
            if (!isTimerStart) {
                int interval = 1000;
                int duration = 120;
                Timer timer = new Timer(interval, new
TimerListener(duration, this));
                timer.start();
                isTimerStart = true;
            }
            btn.setIcon(listOfImages.get(imgs[finalI][finalJ]));
            pair.setX(finalI);
            pair.setY(finalJ);
            pairs.add(pair);
        });
    }
}

```

```

        cnt++;
        if ((pairs.size() == 2) &&
(pairs.get(0).equals(pairs.get(1)))) {
            cnt--;

            pairs.remove(1);
        }
        if (pairs.size() == 2) {
            if ((imgs[pairs.get(0).getX()][pairs.get(0).getY()]) ==
(imgs[pairs.get(1).getX()][pairs.get(1)
            .getY()]))) {
                buttons[pairs.get(0).getX()][pairs.get(0).getY()].set
Enabled(false);
                buttons[pairs.get(1).getX()][pairs.get(1).getY()].set
Enabled(false);

                pairs.remove(1);
                pairs.remove(0);

                score++;
                scoreLabel.setText(" SCORE = " + String.valueOf(score));
            } else {
                try {
                    Thread.sleep(500);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                buttons[pairs.get(0).getX()][pairs.get(0).getY()].set
Icon(listOfImages.get(0));
                buttons[pairs.get(1).getX()][pairs.get(1).getY()].set
Icon(listOfImages.get(0));

                pairs.remove(1);
                pairs.remove(0);
                cnt -= 2;
            }
        }
    }

```

This code consists of a grid of buttons that are initially all hidden, and the player must click on pairs of buttons to reveal their images. If the images match, the buttons are disabled and the score is incremented, and if they do not match, the images are hidden again. The ActionListener handles the button clicks, respectively.

The code uses several Swing components, including JLabel, JButton, JPanel, and ImageIcon, to create the game interface.

PAIR

```
public class Pair {  
    private int x;  
    private int y;  
  
    public Pair() {  
    }  
  
    public Pair(int x, int y) {  
        this.x = x;  
        this.y = y;  
    }  
}
```

The code defines a class `Pair` that represents a pair of integer coordinates (x, y) in a 2-dimensional grid. The class has two instance variables `x` and `y` that hold the coordinates and getter and setter methods to access and modify them. The class also has a constructor with two parameters to initialize the coordinates, and a no-argument constructor.

```
public Pair getRandomPair(int[][] tab) {  
    Random rnd = new Random();  
    int i, j;  
    while (true) {  
        i = rnd.nextInt(tab.length);  
        j = rnd.nextInt(tab.length);  
        if (tab[i][j] == 0) break;  
    }  
    return new Pair(i, j);  
}
```

The class has a method named `getRandomPair` that takes a 2D integer array as input and returns a random `Pair` object whose coordinates correspond to an element in the input array that has a value of 0. The method achieves this by using a while loop that generates random row and column indices using the `Random` class from the `java.util` package until an element in the input array with a value of 0 is found. Once such an element is found, a new `Pair` object is created with the corresponding row and column indices and returned.

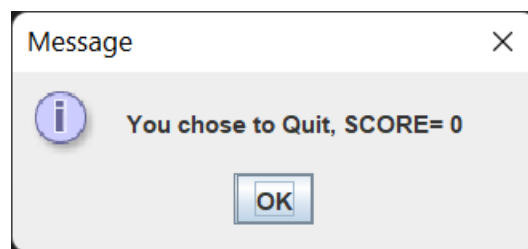
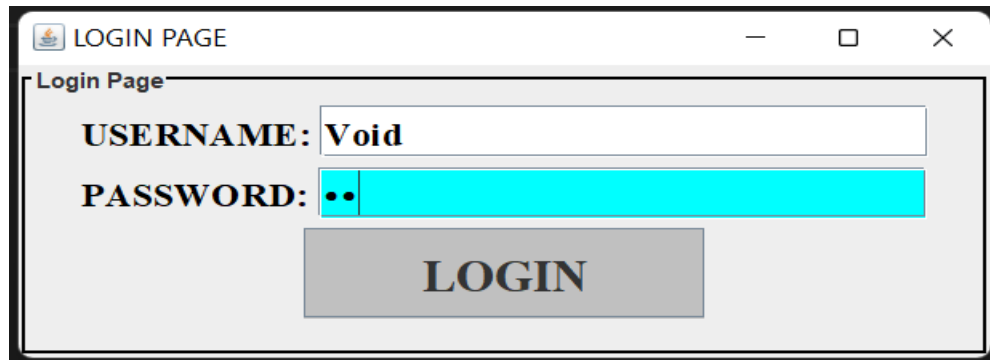
```
public boolean equals(Pair pair) {  
    if ((x == pair.getX()) && (y == pair.getY())) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

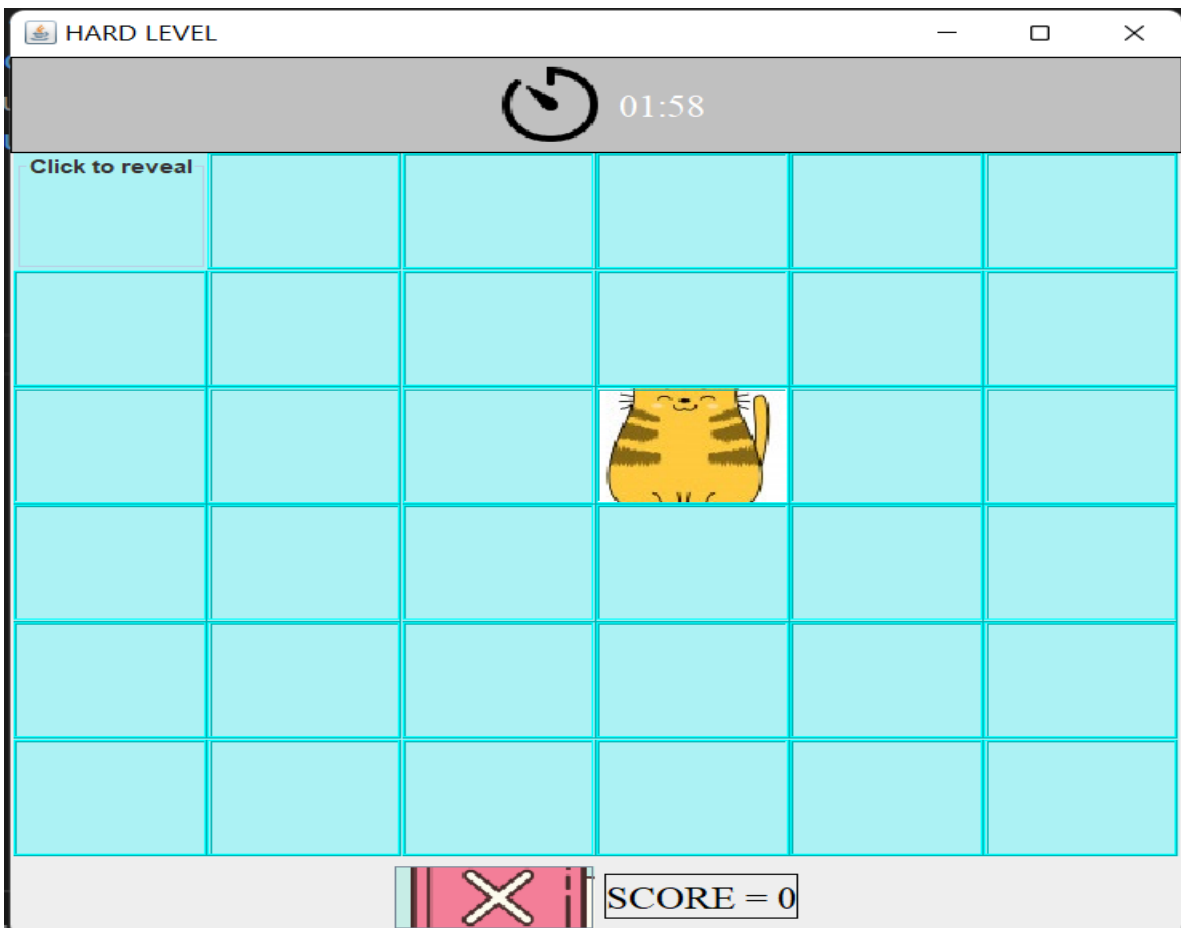
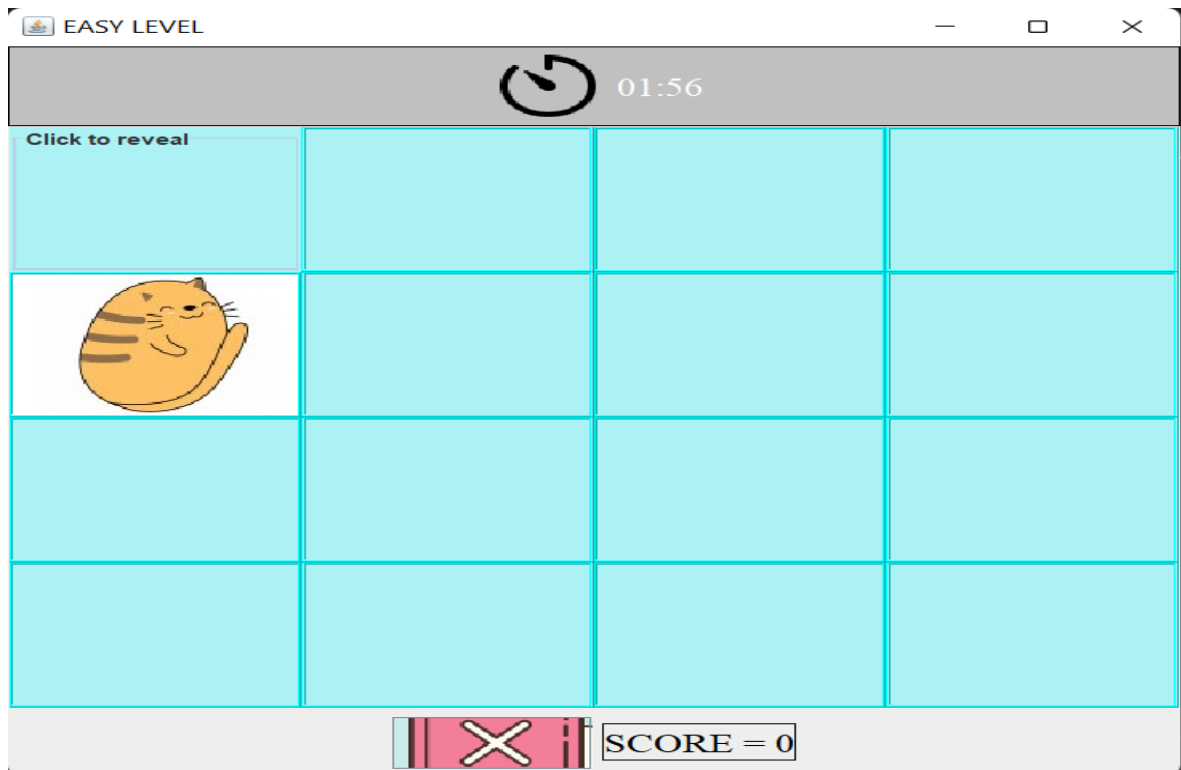
The equals method is used to compare two Pair objects for equality based on their x and y coordinates. If two Pair objects have the same x and y values, the method returns true; otherwise, it returns false.

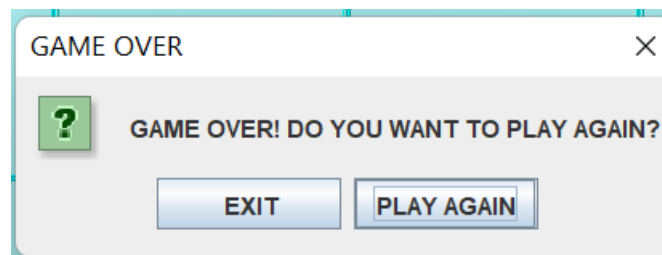
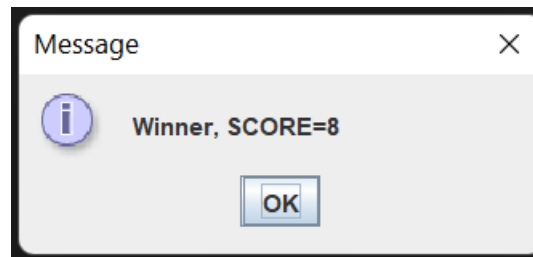
SAMPLE CODE

(Added in module wise description)

OUTCOME OF THE PROJECT







CONCLUSION

In conclusion, the matching game is a simple yet interesting game that was implemented in Java using basic GUI components. The objective of the game was to match pairs of images hidden behind buttons by flipping them over in pairs until all pairs are found. The game's implementation uses a 2D array to store the images' locations and a list of pairs to keep track of the matched images.

The game implementation also includes a scoring mechanism that tracks the player's score as they progress through the game, and it also has a timer that keeps track of the time taken by the player to complete the game.

Overall, this game was a good exercise for me to practice my programming skills and improve my problem-solving abilities. It can be enhanced by adding more features, such as Database Connectivity and animations, to make it more engaging and entertaining.