MEMORY GAME

MINOR PROJECT REPORT

By

PARVATHY V NAIR (RA2211003010295)
ANYESHA BISWAS (RA2211003010298)
ACHANTA BHAVYA SREE(RA2211003010316)

Under the guidance of

Dr. Ajanthaa Lakkshmanan

In partial fulfilment for the Course

of

21CSC203P – ADVANCED PROGRAMMING PRACTICE

in **Department of Computing Technologies**

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR

NOVEMBER 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this minor project report for the course **21CSC203P ADVANCED PROGRAMMING PRACTICE** entitled in "**Memory Game**" is the bonafide work of **Parvathy V Nair (RA2211003010295), Anyesha Biswas (RA2211003010298) and Achanta Bhavya Sree (RA2211003010316)** who carried out the work under my supervision.

SIGNATURE	SIGNATURE
DR. AJANTHAA LAKKSHMANAN	**DR. M PUSHPALATHA**
ASSISTANT PROFESSOR	**HEAD OF THE DEPARTMENT**
DEPARTMENT OF COMPUTING TECHNOLOGIES	**DEPARTMENT OF COMPUTING TECHNOLOGIES**

ABSTRACT

In the realm of cognitive enhancement through gamification, the Memory Game project emerges as a dynamic and intellectually stimulating venture, implemented in the versatile Java language. Renowned for its capacity to augment attention, concentration, and critical thinking, memory games have been perennial favorites among individuals seeking both entertainment and mental acuity.

The essence of the Memory Game lies in its unique challenge: players are tasked with recalling and matching pairs of images within a predefined set of attempts. The implementation, realized in Java, not only provides a recreational experience but also serves as a platform for the cultivation of cognitive skills.

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professors Dr. Vadivu. G , Professor, Department of Data Science and Business Systems and Dr. Sasikala. E Professor, Department of Data Science and Business Systems** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to our my Course project Faculty Dr. **Ajanthaa Lakkshmanan , Assistant Professor , Department of Computing Technologies,** for her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **HoD Dr. M Pushpalatha, Department of Computing Technologies** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Motivation

The Memory Game project is motivated by a threefold purpose: cognitive enrichment, educational impact, and Java language proficiency. Our goal is to provide users with an engaging tool that stimulates memory, pattern recognition, and critical thinking, offering a comprehensive cognitive workout. Beyond entertainment, the game serves as an interactive educational tool, promoting adaptability and mental agility through randomized challenges. Choosing Java as the programming language aligns with our commitment to advancing proficiency in this versatile language, reinforcing existing skills and exploring advanced concepts. In crafting an innovative blend of technology with a classic concept, the project prioritizes user engagement, delivering a modernized and visually appealing game experience. Our motivation lies in creating meaningful user interactions, contributing to both software development and the enhancement of cognitive well-being.

1.2 Objective

The main goal of the Memory Game project is to use Java programming to build an engaging and instructive game environment. In order to improve players' cognitive abilities—such as memory, pattern recognition, and critical thinking—the initiative challenges them to match pairs of pictures in a certain amount of time. The game has a 4x4 grid structure with randomly placed images to provide both strategic involvement and diversity. Additionally, the initiative aims to promote Java programming expertise by giving developers a chance to hone their abilities and study more complex ideas in the language. The goal is to provide a visually beautiful and fun game that not only entertains players but also enhances their cognitive growth and learning experiences through creative features and user-centric design.

1.3 Problem Statement

The Memory Game project tackles the problem of developing a Java game that is both entertaining and intellectually demanding while also encouraging cognitive development. The main challenge is creating a system in which participants have to recall picture locations inside a 4x4 grid and strategically match pairings while dealing with a time limit. The work becomes more challenging when elements like picture flipping, a penalty for mismatched pairs, and random placements are implemented. The project also seeks to strike a balance between pleasure and instructional value, posing a challenge in developing a game that improves memory, critical thinking, and pattern recognition abilities while still being aesthetically pleasing and easy to use.

1.4 Challenges

Creating the Memory Game project involves a number of complex difficulties. The creation of a captivating Java-based game that skillfully blends enjoyment and cognitive enrichment is the main priority. The difficulty is in putting the theory into practice where participants strategically match pairings inside a 4x4 grid while having a restricted number of attempts. Complexities like picture flipping, mismatched pair penalty dynamics, and random placement requirements necessitate a careful balancing act between user accessibility and gameplay complexity. Another challenge is striking a balance between instructional value and visual appeal; this calls for careful design to improve abilities in pattern recognition, memory, and critical thinking. An additional level of complexity is introduced by the game's adaptation to different user demographics. The project is complex as developers have to overcome these obstacles while improving their knowledge of Java programming.

2. LITERATURE SURVEY

Algorithms for Memory Game Logic:

The algorithms category focuses on the development of efficient and challenging logic for the memory game. This includes methods for shuffling and presenting cards, as well as assessing player performance to maintain an optimal level of difficulty.

Integration Techniques:

Integration techniques involve incorporating engaging game elements to enhance the user experience. This category explores methods to seamlessly blend game features into the memory game, motivating players and maintaining their interest.

GUI Design for Memory Games:

The GUI design category concentrates on creating an intuitive and visually appealing user interface. Usability principles and guidelines are applied to design a user-friendly interface that facilitates easy navigation and enhances the overall gaming experience.

Offline Functionality in Java:

This category addresses the implementation of offline functionality, allowing users to play the memory game without an internet connection. Considerations include data storage, synchronization when online, and ensuring a seamless transition between online and offline modes within the Java environment.

Game Variations:

Introduce various game modes or difficulty levels, such as timed challenges, limited attempts, or progressive difficulty, to cater to different player preferences and skill levels.

Score Tracking:

Implement a scoring system to track and display players' performance over time. This can include metrics like accuracy, completion time, and streaks, providing motivation for players to improve.

3. REQUIREMENT ANALYSIS

1. Functional Requirements:

- Allow players to flip and view images on a 4x4 grid.
- Enable users to attempt matching pairs of images.

-Track and display the number of attempts.

-Identify and keep correctly matched pairs open on the grid.

- Enforce a maximum limit of 10 attempts per game.

-End the game when the player exhausts all attempts.

2. Non-Functional Requirements:

-The graphical user interface (GUI) should be intuitive and visually appealing.

-The game should be accessible and enjoyable for users of various age groups

- The game should be compatible with different operating systems (Windows, macOS, Linux).
- The system should be scalable to accommodate potential future updates or additional features.

3. Technical Requirements:

- Implementation in the Java programming language.

-The system must efficiently handle user input for flipping images and interacting with the game.

-Implement a reliable algorithm for shuffling and random placement of images.

- Utilize appropriate Java libraries for graphics rendering to create an engaging user interface.

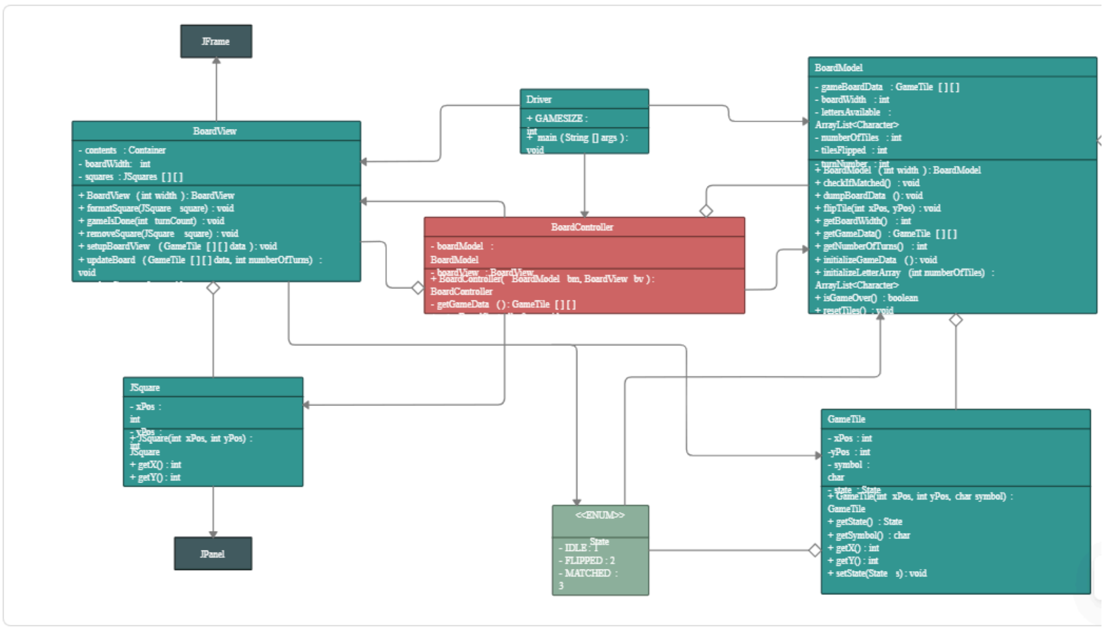
4. Open-Source Collaboration:

- Comprehensive documentation to facilitate open-source contributions.
- Implementation of versioning practices for effective code management.

5. Testing Requirements:

- Rigorous testing for functional correctness, performance, and security.
- Continuous integration and automated testing for code quality assurance.

4. ARCHITECTURE AND DESIGN

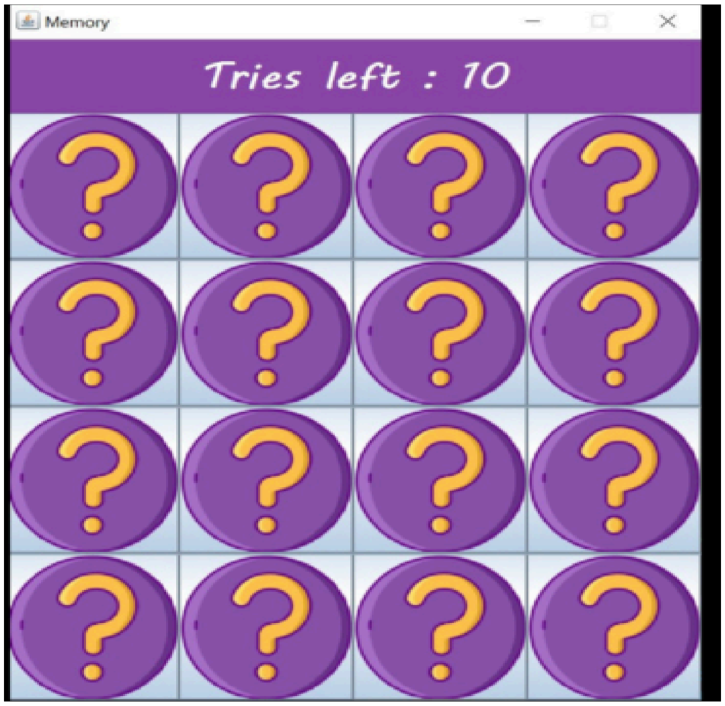

JFrame
BoardView
- contents : Container
- boardWidth: int
- squares : JSquares [ ] [ ]
+ BoardView (int width): BoardView
+ formatSquare(JSquare square) : void
+ gameIsDone(int turnCount) : void
+ removeSquare(JSquare square) : void
+ setupBoardView (GameTile [ ] [ ] data): void
+ updateBoard (GameTile [ ] [ ] data, int numberOfTurns) : void
Driver
+ GAMESIZE : int
+ main (String [] args): void
BoardController
- boardModel : BoardModel
- boardView : BoardView
+ BoardController(BoardModel bm, BoardView bv): BoardController
- getGameData ( ): GameTile [ ] [ ]
BoardModel
- gameBoardData : GameTile [ ] [ ]
- boardWidth : int
- lettersAvailable : ArrayList<Character>
- numberOfTiles : int
- tilesFlipped : int
- turnNumber : int
+ BoardModel (int width): BoardModel
+ checkIfMatched() : void
+ dumpBoardData (): void
+ flipTile(int xPos, yPos) : void
+ getBoardWidth() : int
+ getGameData() : GameTile [ ] [ ]
+ getNumberOfTurns() : int
+ initializeGameData ( ): void
+ initializeLetterArray (int numberOfTiles) : ArrayList<Character>
+ isGameOver() : boolean
+ resetTiles() : void
JSquare
- xPos : int
- yPos : int
+ JSquare(int xPos, int yPos) : JSquare
+ getX() : int
+ getY() : int
JPanel
<<ENUM>>
State
- IDLE : 1
- FLIPPED : 2
- MATCHED : 3
GameTile
- xPos : int
-yPos : int
- symbol : char
- state : State
+ GameTile(int xPos, int yPos, char symbol) : GameTile
+ getState() : State
+ getSymbol() : char
+ getX() : int
+ getY() : int
+ setState(State s) : void

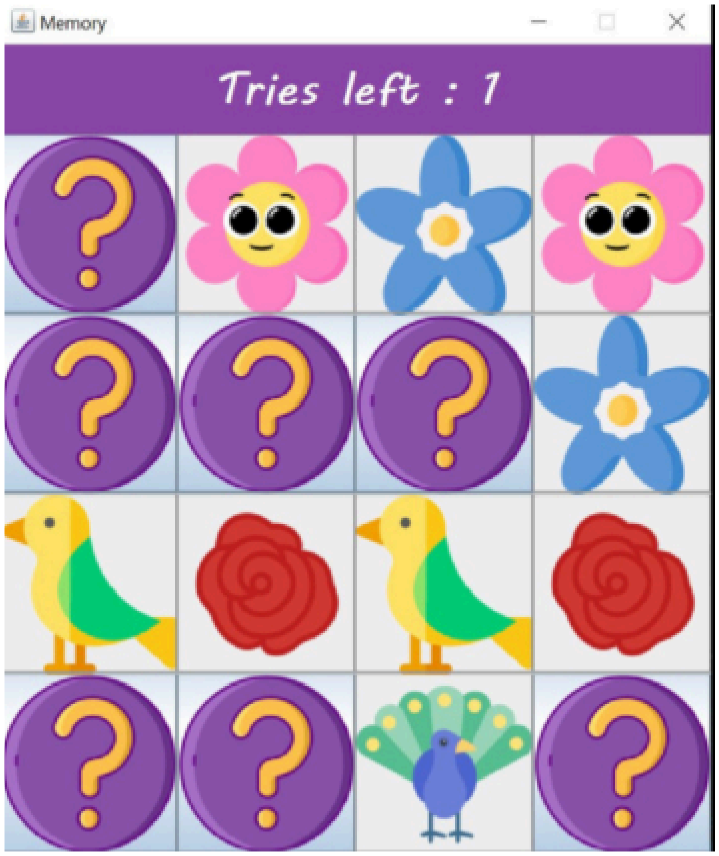
5. IMPLEMENTATION

```
import javax.imageio.ImageIO;
import javax.imageio.stream.ImageInputStream;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;
import java.util.stream.Collectors;
import javax.swing.Timer;
class Game {
   public static class Controller {
      final JFrame window;
      Model model;
      View view;
       // Constructor
      public Controller(Model model) {
         this.window = new JFrame("Memory"); // Create the window

this.window.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);  //
Close the window when the user clicks on the close button
         this.window.setResizable(false); // Disable the resizing of the window
         this.reset(model); // Reset the game
      }
      // Reset the game
      public void reset(Model model) {
         this.model = model;
         this.view = new View(model);
         this.window.setVisible(false);
         this.window.setContentPane(view);
         this.window.pack();
         this.window.setLocationRelativeTo(null);
         for (JButton button : this.model.getButtons()) {
            button.addActionListener(new ButtonActionListener(this));
         }
         Utilities.timer(200, (ignored) -> this.window.setVisible(true)); // Show the
window after 200ms
      }
      public JFrame getWindow() {
         return this.window;
      }
      public Model getModel() {
         return this.model;
      }
      public View getView() {
         return this.view;
      }
   }
   public static class Model {
      // Constants for the game
      static final String[] AVAILABLE_IMAGES = new String[]{"0.png", "1.png",
"2.png", "3.png", "4.png", "5.png", "6.png", "7.png", "8.png"};
      static final Integer MAX_REGISTERED_SCORES = 10;
      final ArrayList<Float> scores;
      final ArrayList<JButton> buttons;
      final int columns; // Number of columns
      int tries; // Number of tries left
      boolean gameStarted; // Is the game started
      public Model(int columns) {
         this.columns = columns; // Number of columns
         this.buttons = new ArrayList<>(); // List of buttons in the game
         this.scores = new ArrayList<>(); // List of scores in the game
         this.tries =  10; // Number of tries initially
         this.gameStarted = false; // Game is not started initially
         int numberOfImage = columns * columns; // Number of images
         Vector<Integer> v = new Vector<>(); // Vector to store the images
         for (int i = 0; i < numberOfImage - numberOfImage % 2; i++) { // Add the
images twice
            v.add(i % (numberOfImage / 2));
         }
         if (numberOfImage % 2 != 0) v.add(AVAILABLE_IMAGES.length - 1); //
Add the last image if the number of images is odd
         // Add the images as a button to the game
         for (int i = 0; i < numberOfImage; i++) {
            int rand = (int) (Math.random() * v.size()); // Randomly select an image
            String reference = AVAILABLE_IMAGES[v.elementAt(rand)]; // Get the
image
            this.buttons.add(new MemoryButton(reference)); // Add the image as a
button
            v.removeElementAt(rand); // Remove the image from the vector
         }
      }
      public int getColumns() {
         return columns;
      }
      public ArrayList<JButton> getButtons() {
         return buttons;
      }
      // Get the number of tries left
      public int getTries() {
         return tries;
      }
```

```
    // Decrement the tries count by calling this method
    public void decrementTries() {
        this.tries--;
    }
    // return if the game has started
    public boolean isGameStarted() {
        return this.gameStarted;
    }
    // start the game
    public void startGame() {
        this.gameStarted = true;
    }
}
// class to handle the UI of the game
public static class View extends JPanel {
    final JLabel tries;
    public View(Model model) {
        this.setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
        this.tries = new JLabel("", SwingConstants.CENTER);
        this.tries.setFont(new Font("MV Boli", Font.BOLD, 30));
        this.tries.setForeground(Color.WHITE);

        JPanel imagePanel = new JPanel();
        int columns = model.getColumns();
        imagePanel.setLayout(new GridLayout(columns, columns));
        for (JButton button : model.getButtons()) {
            imagePanel.add(button);
        }
        this.setTries(model.getTries());
        JPanel triesPanel = new JPanel();
        triesPanel.add(this.tries);
        triesPanel.setAlignmentX(CENTER_ALIGNMENT);
        triesPanel.setBackground(new Color(0X8946A6));
        this.add(triesPanel);
        this.add(imagePanel);
    }
    public void setTries(int triesLeft) {
        this.tries.setText("Tries left : " + triesLeft);
    }
}
// class to handle the button clicks
public static class ReferencedIcon extends ImageIcon {
    final String reference;
    public ReferencedIcon(Image image, String reference) {
        super(image);
        this.reference = reference;
    }
    public String getReference() {
            return reference;
        }
    }
    // class to handle the button on the images
    public static class MemoryButton extends JButton {
        static final String IMAGE_PATH = "";
        static final Image NO_IMAGE = Utilities.loadImage("no_image.png");
        public MemoryButton(String reference) {
            Image image = Utilities.loadImage(IMAGE_PATH + reference);
            Dimension dimension = new Dimension(120, 120);
            this.setPreferredSize(dimension);
            this.setIcon(new ImageIcon(NO_IMAGE));
            this.setDisabledIcon(new ReferencedIcon(image, reference));
        }
    }
    public static class Dialogs {
        public static void showLoseDialog(JFrame window) {
            UIManager.put("OptionPane.background", new Color(0XEA99D5));
            UIManager.put("Panel.background", new Color(0XEA99D5));
            JOptionPane.showMessageDialog(window, "You lost, try again !", "You lost
!", JOptionPane.INFORMATION_MESSAGE);
        }
        public static void showWinDialog(JFrame window, Model model) {
            String message = String.format("Congrats you won!!");
            UIManager.put("OptionPane.background", new Color(0XEA99D5));
            UIManager.put("Panel.background", new Color(0XEA99D5));
            JOptionPane.showMessageDialog(window.getContentPane(), message, "",
JOptionPane.INFORMATION_MESSAGE);
        }
    }
    // class to handle the button clicks
    public static class ButtonActionListener implements ActionListener {
        final Controller controller;
        final Model model;
        final View view;
        final JFrame window;
        static int disabledButtonCount = 0;
        static JButton lastDisabledButton = null;
        static final Image TRAP_IMAGE = Utilities.loadImage("no_image.png");
        final ReferencedIcon trap;
        public ButtonActionListener(Controller controller) {
            this.controller = controller;
            this.model = controller.getModel();
            this.view = controller.getView();
            this.window = controller.getWindow();
            this.trap = new ReferencedIcon(TRAP_IMAGE, "no_image.png");
        }
        // Method to handle the button clicks and check if two images are same
        @Override
        public void actionPerformed(ActionEvent e) {
            JButton button = (JButton) e.getSource();
            button.setEnabled(false);
            ReferencedIcon thisIcon = (ReferencedIcon) button.getDisabledIcon();
            disabledButtonCount++;
            if (!model.isGameStarted()) { // If the game has not started
                model.startGame(); // Start the game
            }
            if (disabledButtonCount == 2) { // If two buttons are disabled
                ReferencedIcon thatIcon = (ReferencedIcon)
lastDisabledButton.getDisabledIcon();
                boolean isPair = thisIcon.getReference().equals(thatIcon.getReference()); //
Check if the two images are the same
                if (!isPair) { // If the two images are not the same
                    model.decrementTries(); // Decrement the number of tries
                    view.setTries(model.getTries()); // Update the number of tries
                    JButton lastButton = lastDisabledButton; // Store the last button
                    Utilities.timer(500, ((ignored) -> { // Wait 500ms before re-enabling the
buttons
                        button.setEnabled(true); // Re-enable the button
                        lastButton.setEnabled(true); // Re-enable the last button
                    }));
                }
                disabledButtonCount = 0; // Reset the counter
            }
            ArrayList<JButton> enabledButtons = (ArrayList<JButton>)
model.getButtons().stream().filter(Component::isEnabled).collect(Collectors.toList())
;
            if (enabledButtons.size() == 0) { // If all the buttons are disabled
                controller.reset(new Model(controller.getModel().getColumns())); // Reset
the game
                Dialogs.showWinDialog(window, model); // Show the win dialog
            }
            lastDisabledButton = button;    // Store the last button
            if (model.getTries() == 0) { // If the number of tries is 0
                controller.reset(new Model(controller.getModel().getColumns())); // Reset
the game
                Dialogs.showLoseDialog(window); // Show the lose dialog
                Utilities.timer(1000, (ignored) -> model.getButtons().forEach(btn ->
btn.setEnabled(false))); // Wait 1s before disabling all the buttons
            }
        }
    }
    public static class Utilities {
        static final ClassLoader cl = Utilities.class.getClassLoader();
        // Method to create a timer
        public static void timer(int delay, ActionListener listener) {
            Timer t = new Timer(delay, listener);
            t.setRepeats(false);
            t.start();
        }
        // Method to load an image
        public static Image loadImage(String s) {
            Image image = null;
            try {
                InputStream resourceStream = cl.getResourceAsStream(s);
                if (resourceStream != null) {
                    ImageInputStream imageStream =
ImageIO.createImageInputStream(resourceStream);
                    image = ImageIO.read(imageStream);
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
            return image;
        }
    }
}
//  Main class to run the game
class Main {
    static final int DEFAULT_SIZE = 4;
    public static void main(String[] args) {
        Locale.setDefault(Locale.ENGLISH);
        SwingUtilities.invokeLater(() -> new Game.Controller(new
Game.Model(DEFAULT_SIZE)));
    }
}
```

6. RESULTS





7. CONCLUSION

To sum up, the Memory Game project has skillfully combined Java programming knowledge, educational enrichment, and cognitive engagement. The game provides players with an engaging experience, strategic gameplay, and an easy-to-use interface—all while paying close attention to functional needs. Not only have open-source collaborative tools expedited development, but they have also laid the groundwork for further improvements. Because of its combination of instructional and entertaining components, the game is positioned to make a significant contribution to the field of software development and cognitive enrichment. The Memory Game demonstrates the team's dedication to developing a significant and memorable user experience at the intersection of gaming and cognitive growth by effectively implementing randomization algorithms, managing user input efficiently, and adhering to a restricted number of attempts.

8. REFERENCES

1. https://www.geeksforgeeks.org
2. https://www.w3schools.in
3. https://wiingy.com
4. https://www.learningmilestone.com
5. https://medium.com