**Name: Nampally Sahithi**

**ID: SE22UCSE179**

**Class: CSE3**

**Report**

The game involves matching pairs of numbers hidden under buttons. The player clicks two buttons at a time to try and find matching pairs. The game ends when all pairs are matched or when the player runs out of tries.

The program uses Java’s basic features like classes, objects, arrays, and event handling to make the game interactive. Below is an explanation of the properties of Java and how they are used in the code.

**Properties of Java**

1. **Object-Oriented Programming (OOP)**: Java is an object-oriented language. This means the code is organized into objects, each having its attributes and methods. In this game, classes like MemoryGame and FlipGame are used to define the objects and their behaviors.
2. **Classes and Objects**: A class has attributes and methods. An object is an instance of a class. In the code FlipGame is a class, and the object game created in the Flip class is an instance of FlipGame.
3. **Inheritance**: one class inherits properties and methods from another class. In this game, FlipGame extends MemoryGame, which means FlipGame inherits the basic structure of the game from MemoryGame and adds its specific functionality.
4. **Abstraction:** hiding the complex details of implementation and only showing the necessary features. In the game, the MemoryGame class provides a simple interface without revealing how the game logic is implemented.
5. **Encapsulation:** protecting the data by making it private and providing methods to access or modify it. In the game, variables like tries are protected, and their values are changed only through specific methods, ensuring the game logic is maintained.
6. **Graphics and GUI**: Java provides built-in libraries to create graphical user interfaces (GUIs). In this game, we use JFrame, JPanel, and JButton to create the game window, grid, and buttons.

**Explanation of the Code**

The code is divided into different parts:

1. **MemoryGame Class**: This is the base class that holds common properties and methods for the game.

* It has properties like gridSize (the size of the grid), maxTries (the maximum number of tries), and tries (the current number of tries left).
* It defines two abstract methods: initializeGame() to set up the game, and restartGame() to reset the game and the grid.

2. **FlipGame Class**: This class extends MemoryGame and adds the game-specific logic.

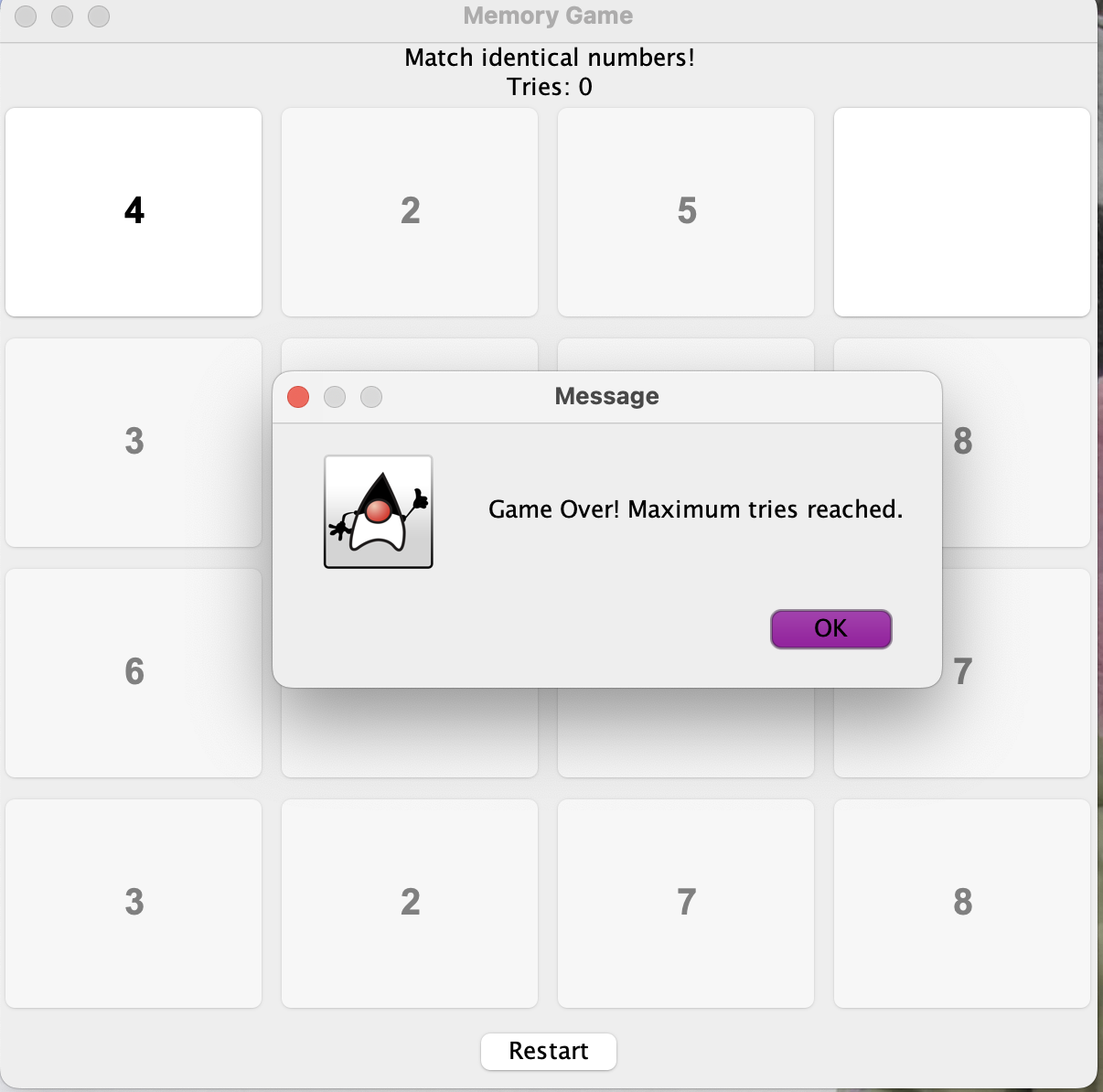
* It has an ArrayList called numbers, which stores pairs of numbers to be placed on the grid.
* The populateGrid() method creates a grid of buttons using the JButton class and places the numbers under the buttons.
* The handleClick() method manages what happens when a player clicks a button. It checks if the clicked buttons match and updates the game accordingly.
* If the player finds a match, the buttons are disabled; if not, the buttons are hidden after a short delay using a Timer.
* The game ends when all pairs are matched or when the player runs out of tries.

3. **Flip Class (Main Class)**: This is the main class that runs the game.

* **Creates Game Window**: The main game window is created using JFrame with a size of 600x600 pixels.
* **Displays Instructions**: Before the game starts, a pop-up window appears showing instructions for the player, explaining how to play the game and the objective.
* **Sets Up Panels**:
  + **Top Panel**: Displays the instructions and the number of remaining tries.
  + **Grid Panel**: Displays the grid of buttons, each representing a tile in the game.
  + **Bottom Panel**: Contains a Restart button, which allows the player to start a new game.
* **Handles Restart**: The Restart button calls the restartGame() method from the FlipGame class to reset the game and shuffle the tiles.
* **Displays Game Window**: The game window is displayed using frame.setVisible(true), allowing the player to interact with the game.

**How the Game Works**

1. **Starting the Game**: When the game starts, the grid is populated with random numbers (pairs). The player has a limited number of tries to match the pairs.
2. **Making a Move**: When the player clicks a button, the value under the button is shown. If it’s the first button clicked, it’s saved. When the second button is clicked, the game checks if the two values match. If they do, the buttons are disabled. If they don’t, the numbers are hidden after a brief delay.
3. **Game Over**: The game ends when all pairs are found, or the player runs out of tries. A message is shown to tell the player whether they won or lost.
4. **Restarting the Game**: The game can be restarted by clicking the "Restart" button, which resets the grid and the number of tries.

This memory game demonstrates how Java can be used to build interactive games with graphical user interfaces. It uses object-oriented principles like inheritance, event handling, and arrays to make the game functional. The game allows players to practice memory skills and is a fun way to learn about Java programming.