Object Oriented Programming Lab
Mini Project

Title: 2048 in Java

Submitted by:

Adriteyo Das (RegNo: 230953244)

Neema Kini (RegNo: 230953070)

Shivam Lahoty (RegNo: 230953026)

Department of Information and Communication Technology

MIT Manipal (CCE-B)

2024

About 2048 GAME

2048 is a popular sliding puzzle game where the player combines numbered tiles on a 4x4 grid to create a tile with the number 2048. The player moves tiles in four directions: up, down, left, or right. When two tiles of the same number collide, they merge into one tile with the sum of their values. The goal is to reach the tile with the value 2048, although the game can continue beyond this.

The challenge comes from the fact that every move adds a new tile (usually with the value of 2 or 4), and the grid becomes progressively fuller. Players must strategically combine tiles to prevent the grid from filling up and losing the game.

The Game is implemented as a WebApp using JavaScript and the aim of our project is to create an offline version for it using JavaFX.

Java Concepts Used in the Project

1. Object-Oriented Programming (OOP)

- **Classes and Objects:** The 2048 game consists of multiple classes such as Tile, GameBoard, and GameDirection, each representing different components of the game.
- **Encapsulation:** Attributes like the current state of the grid, score, and available moves are encapsulated in the relevant classes, preventing unauthorized access and modification.
- **Inheritance and Polymorphism:** These are used to handle different types of actions or game events (like different movements or tile merging behaviors).

2. JavaFX Framework Scene Graph

- **Event Handling:**
  - User Input Handling: The player's moves (up, down, left, right) are handled through event listeners.
  - Key Handling: JavaFX uses SetOnKeyPress and SetOnKeyReleased to capture user input.
- **Layout Components:**
  - GridPane and StackPane for the window and grid
  - JavaFX Layout class to draw the tiles and grid

- **UI Elements:**
  - Property Binding
  - UI Controls

3. Exception Handling

- **Try-Catch Blocks:** Exception handling is used for recognizing special cases where the Game State has reached where there are no more possible moves or the goal has been reached.
- **Random Generation:** The Random class is used to generate the random placement of new tiles (with values 2 or 4) on the grid after every move.

4. Game Logic and State Management

- **Tile Merging:** A key part of the game logic is determining when two tiles can merge. This involves checking adjacent tiles, adding their values, and replacing the original tiles.
- **Game State:** The game keeps track of whether it's in progress or over, if moves are still possible, or if the player has won. This involves managing the state transitions carefully.

Contribution by Team Members

Member	Contributions
Shivam Lahoty	• Inheritance (extending Application, GridPane) • Encapsulation (private fields, public methods) • Polymorphism (method overriding)
Neema Kini	• Exception Handling • Input validation • Game state exceptions • Game Logic
Adriteyo Das	• JavaFX Framework Scene Graph • Event Handling • Property Binding • UI Controls

Game Screenshots

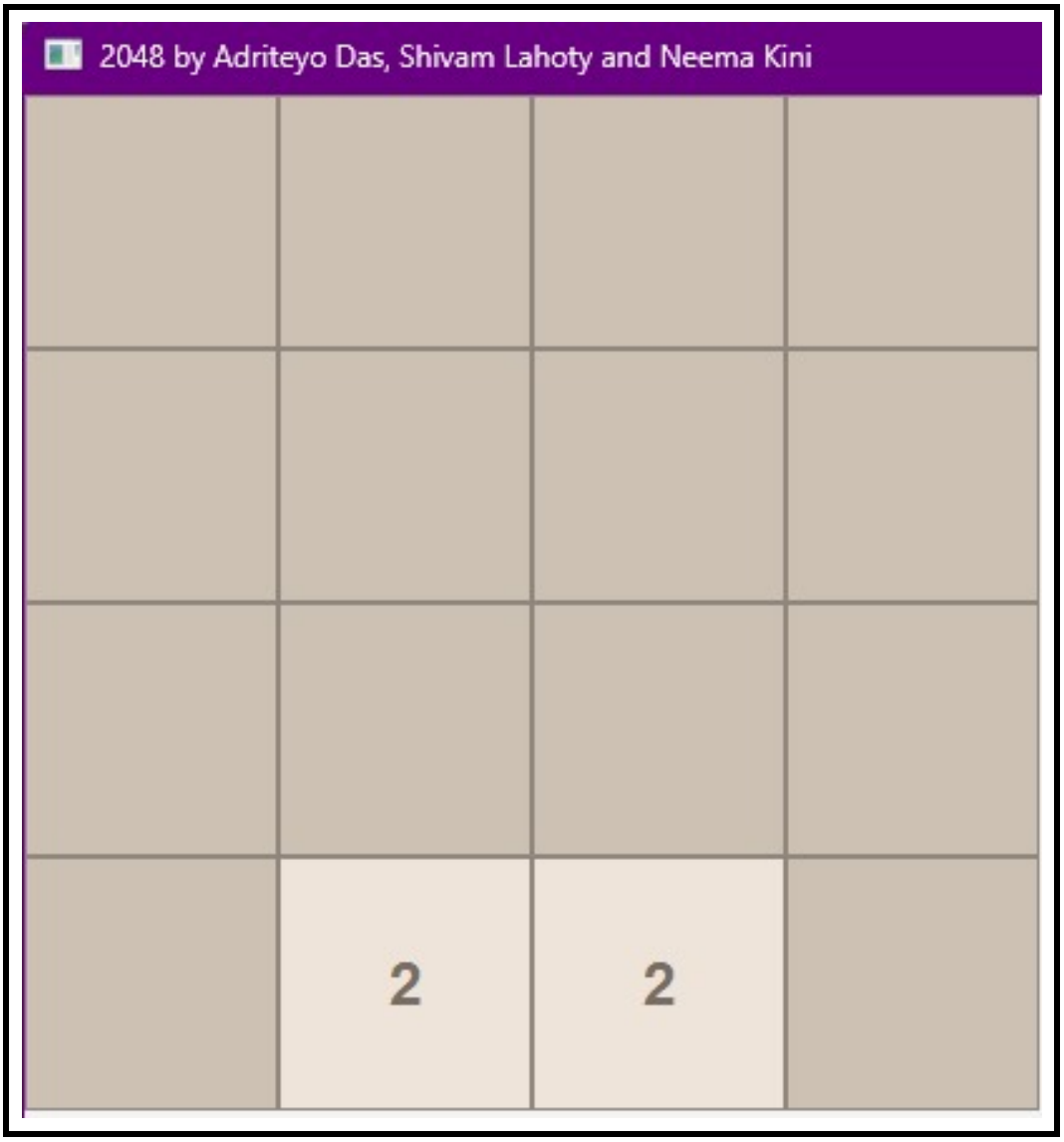

Figure 1: **Initial Game Screen:** The game starts with two tiles randomly placed on the grid

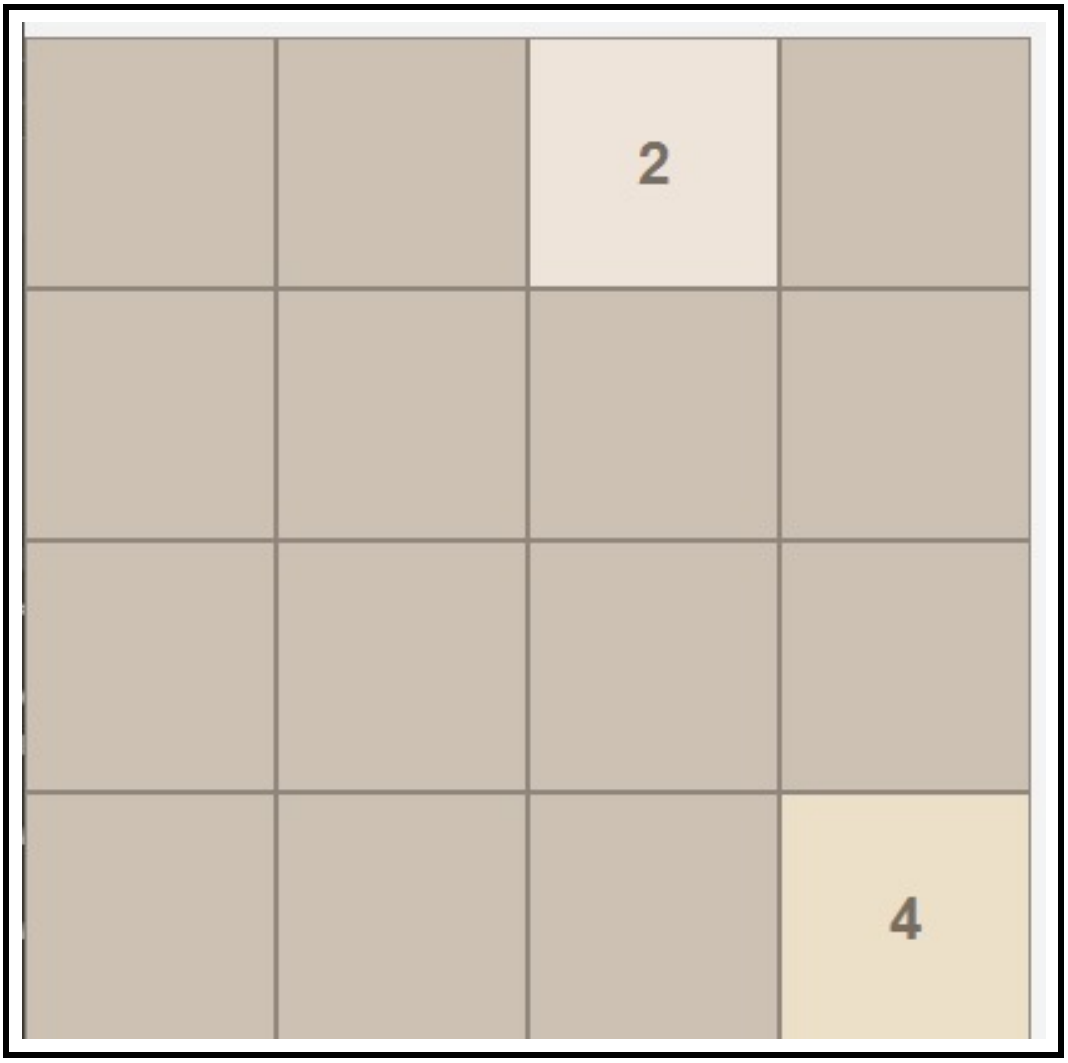

Figure 2: **Right Movement:** The two tiles with value 2 in positions (4,2) and (4,3) are merged to form a single tile with value 4 in position (4,4). A new tile with value 2 is randomly generated in position (1,3)

Figure 3: **Mid-Game State:** The game board shows various merged tiles and strategic positioning

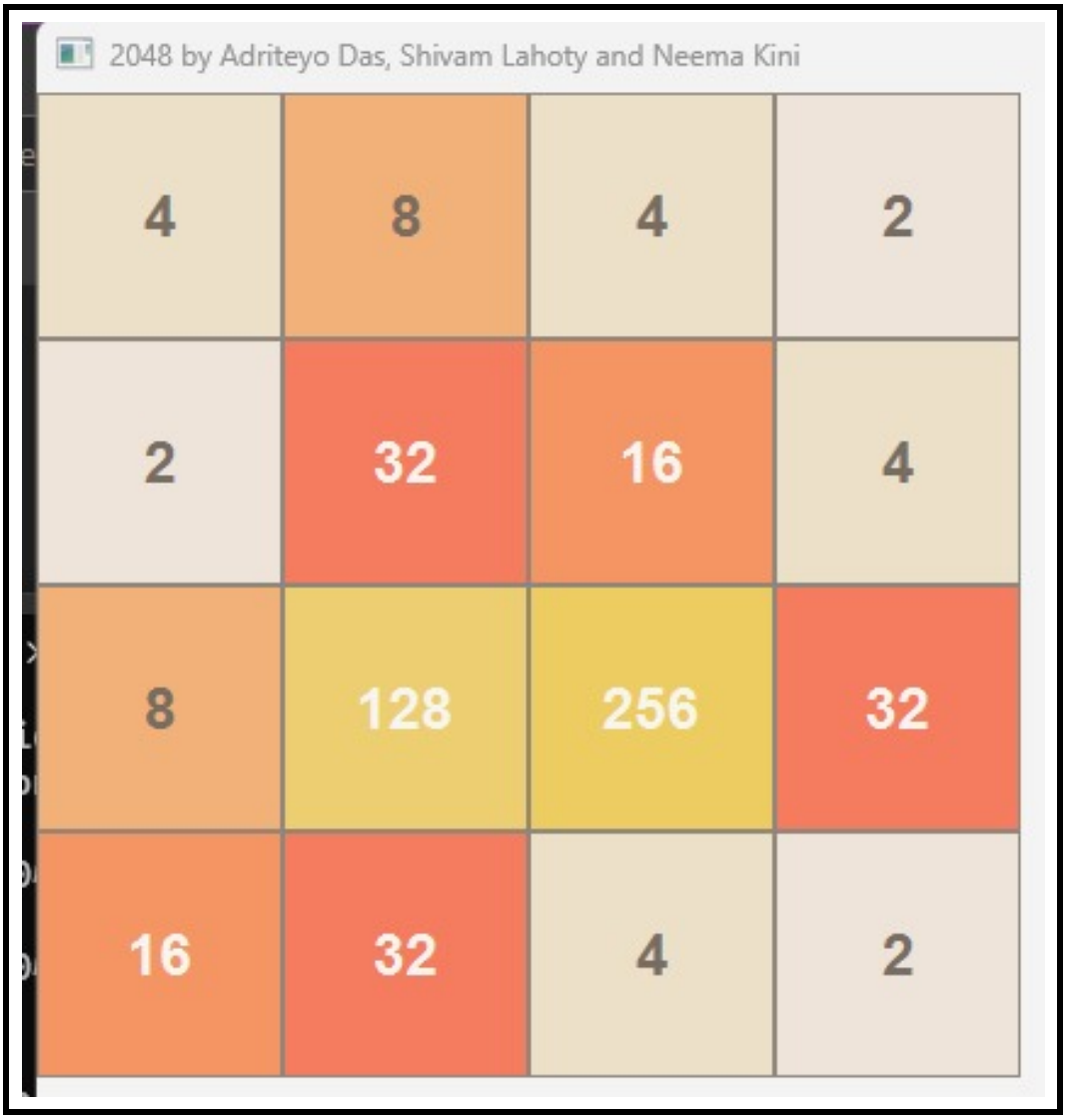

Figure 4: **End-Game Scenario:** The grid is nearly full, requiring careful planning for subsequent moves

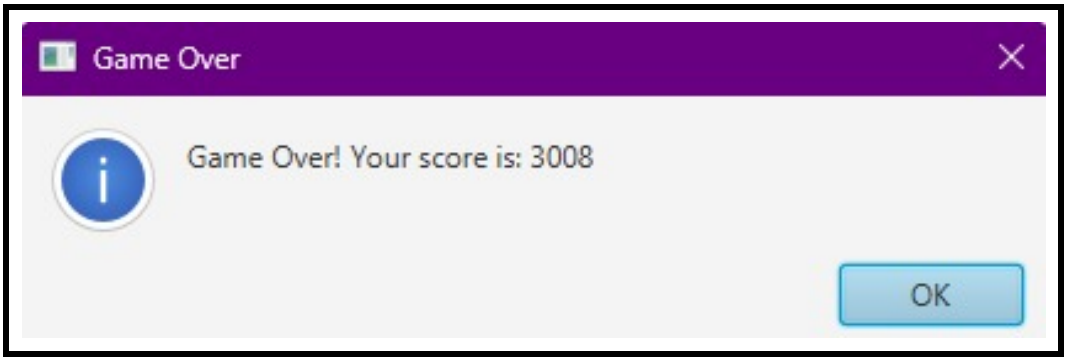

Figure 5: **Game Over Alert:** Displayed when no more valid moves are possible