International University - VNU HCMC

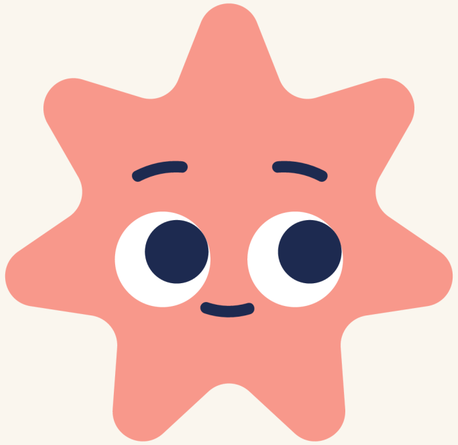
Algorithms and Data Structures

(IT013IU)

Final Project

MINES SWEEPER

Presentation date: 6th June 2025



Group Members

Le Vo Hong Na - ITITSB23007
Le Nguyen Thanh Truc - ITITWE22168
Nguyen Hoang Bao Tran - ITITSB22027

Contents

Let’s Go!

Introduction

Motivation

- Minesweeper is a classic puzzle game where the player must clear a grid without triggering any hidden mines.
- Clicking on a safe cell reveals a number showing how many mines are nearby. Using logic, players must avoid the mines and mark their locations with flags.
- The game ends when all safe cells are uncovered or a mine is clicked.

System Design

Working Tools and Platforms

Software	Purpose
JDK 23	for running Java Program
IntelliJ	IDE to run game code
Canva	Design items
Github	Upload project's code and manage contribution from different individuals

Core Features

- Interactive Grid
- Dynamic Difficulty
- Custom Configuration
- Safe First Click
- Recursive Reveal
- Flagging System
- Win/Loss Detection
- UI Controls
- Sound Effects
- Hint System
- Undo / Redo
- AutoPlay (AI Solver)

Data Structures and Algorithms

Data Structures

- Time
  - Instant: Stores timestamps of two signals (signalA, signalB).
  - Duration: Calculates time difference between the two.

```
import java.time.Duration;
import java.time.Instant;
```

```
private Instant signalA;
private Instant signalB;
```

```
public Duration calculateTimeDifference() {
```

```
Duration timeDifference = calculateTimeDifference();
```

Data Structures

- GameState
  - tileState: 0 = unopened, 1 = opened, 2 = flagged.
  - tileNumber: Number on tile; -1 if unopened.

```
public int[][] tileState;
public int[][] tileNumber;
```

Data Structures

- Minesweeper
  - MineTile[][] mainBoard: 2D grid of game tiles.
  - ArrayList<MineTile> mineList: Dynamic list of all mine tiles.
  - Stack<GameState> undoStack, redoStack: Supports undo/redo with LIFO behavior.

```
protected MineTile[][] mainBoard = new MineTile[Level.getNumRows()][Level.getNumCols()];
```

```
private ArrayList<MineTile> mineList;
```

```
Stack<GameState> undoStack = new Stack<>();
Stack<GameState> redoStack = new Stack<>();
```

Data Structures

- ScoreFileHandler
  - ArrayList<Long>: Stores and manages multiple scores for rankings.

```
public static List<Long> readScoreFromFile(String fileName) {
    List<Long> scoreValues = new ArrayList<>();
```

- Sound
  - Clip[]: Fixed array of sound effects for game actions.

```
import javax.sound.sampled.*;
import java.io.File;
import java.io.IOException;
```

```
private static Clip[] clips;
```

Algorithms

- Flood Fill
  - Purpose: When an empty tile (no adjacent mines) is clicked, automatically open its neighboring tiles until numbered tiles are found.
  - Algorithm: Recursive Depth-First Search (DFS) in 8 directions.
  - Implementation:
    - checkMine(r, c) checks bounds, disables the tile, and continues recursively if no mines are around.
    - Stops when reaching tiles that have adjacent mines.

Algorithms

- Chained Reasoning
  - Purpose: Use logic between neighboring number tiles to deduce mine positions.
  - Idea: If the unopened tiles around one number tile are a subset of another, and the difference in numbers matches the difference in unknown tiles, mines can be identified.
  - Implementation:
    - autoSolveChainedReasoning() compares neighboring number tiles and infers which tiles must be mines using subset logic.

Algorithms

- Probabilistic Move Suggestion
  - Purpose: Suggest the safest tile to click when no certain logic-based move is available.
  - Algorithm:
    - Estimate mine probability based on nearby numbered tiles and surrounding unopened tiles.
    - Choose tile with lowest probability of being a mine.
  - Implementation:
    - suggestBestProbabilisticMove() analyzes the board and picks the best guess.

Algorithms

- Edge Guessing Strategy
  - Purpose: Suggest a move along the edge or corner of the board when no logical or probabilistic move is found.
  - Implementation:
    - suggestBestEdgeMove() selects a tile on the border as a fallback move.

Algorithms

- Undo and Redo Feature
  - Purpose: Allow players to go back to a previous game state or redo an undone move.
  - Data Structure: Two Stacks:
    - undoStack: stores past game states.
    - redoStack: stores states after undo.
  - Implementation:
    - saveStateForUndo(), undo(), and redo() manage game state transitions.

Algorithms

- Player Ranking and Sorting
  - Purpose: Display the top 10 players based on score and time.
  - Sorting Criteria:
    - Higher score first.
    - If scores are equal → shorter time wins.
  - Implementation:
    - toStringScore() reads from file and sorts using a custom Comparator.

Algorithms

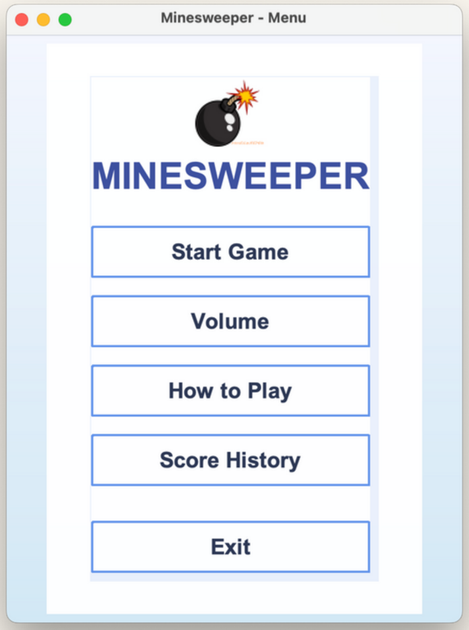
- Counting Adjacent Mines
  - Purpose: Determine how many mines are adjacent to a given tile.
  - Algorithm: Scan all 8 surrounding positions and count mines.
  - Implementation:
    - numOfMinesAround(r, c) calls countTheMinePosition() for each direction.

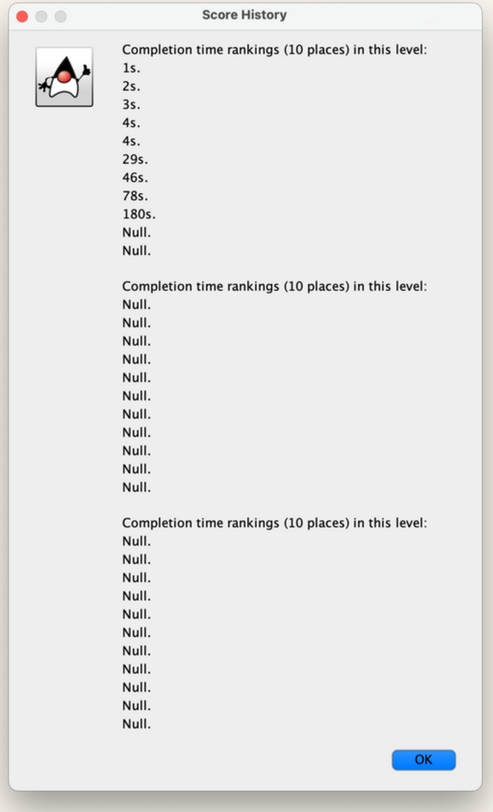
User Interface and User Experience

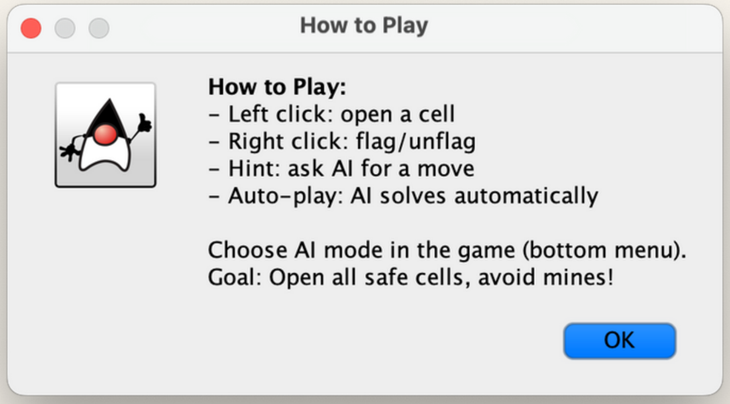
User Interface

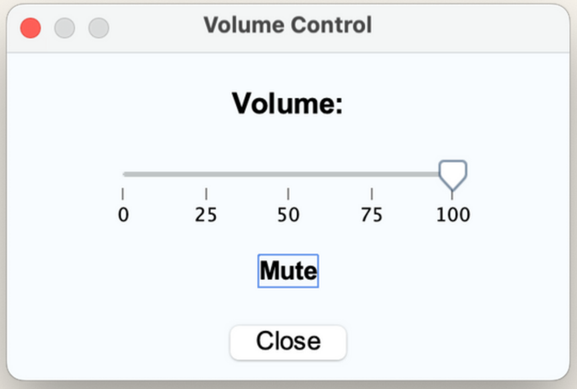
- Game Board
  - The main grid is made of clickable buttons representing cells. Left-click reveals a cell; right-click toggles a flag. Cells show numbers (adjacent mines), a mine icon, or remain blank.
- Control Panel
  - Located at the bottom, it includes “Play Again,” difficulty options (Easy, Medium, Hard), a “Ranking” button, a customizable “Custom” mode, and a sound toggle for user convenience.
- Pop-up Dialogs
  - The game shows dialogs for game over, victory, or invalid custom settings to provide clear user feedback and interaction.

User Interface









User Experience

- First-click safety: The game ensures that the first cell clicked by the player is never a mine, providing a fair starting point.
- Visual consistency: Icons and numbers follow traditional Minesweeper conventions, making the game instantly recognizable.
- Responsiveness: The game responds instantly to user actions without noticeable delay.
- Keyboard-free: All gameplay is controlled via mouse, simplifying interaction.

Final Game

References

GeeksforGeeks. (2025, Mar 27). Stack Data Structure. Retrieved from https://www.geeksforgeeks.org/stack-data-structure/

GeeksforGeeks. (2025). Artificial Intelligence (AI) Algorithms. Retrieved from https://www.geeksforgeeks.org/ai-algorithms/

GeeksforGeeks. (2023, Aug 24). ArrayList vs LinkedList. Retrieved from https://www.geeksforgeeks.org/arraylist-vs-linkedlist-java/

Reference Video: https://www.youtube.com/watch?v=5VrMVSDjeso

Thank You

Q & A Section