VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY
INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

ALGORITHMS AND DATA STRUCTURES

REPORT
TOPIC: MINESWEEPER

MEMBER:

Mai Đức Thiện - ITITIU21319

Contents

I. INTRODUCTION

1. What is Minesweeper?

Minesweeper is a logic puzzle video game genre generally played on personal computers. The game features a grid of clickable tiles, with hidden "mines" (depicted as naval mines in the original game) scattered throughout the board. The objective is to clear the board without detonating any mines, with help from clues about the number of neighboring mines in each field.

2. Project Overview

Minesweeper was created by Microsoft in the 1990s. To win this game the player must open all the unmined squares on a grid, while not detonating any mines. The game is ranked by completion time, so completing the game in the shortest time is also an important goal for skilled players. Beginner is usually on an 8x8 or 9x9 board containing 10 mines, intermediate is usually on a 16x16 board with 40 mines and expert is usually on a 30x16 board with 99 mines; however, there is usually an option to customize board size and mine count.

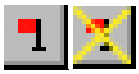
II. PROJECT ANALYSIS

1. Object:

1.1. Object number:

1.2. Mine:

1.3. Marks:



1.4. Tiles:

2. Rules

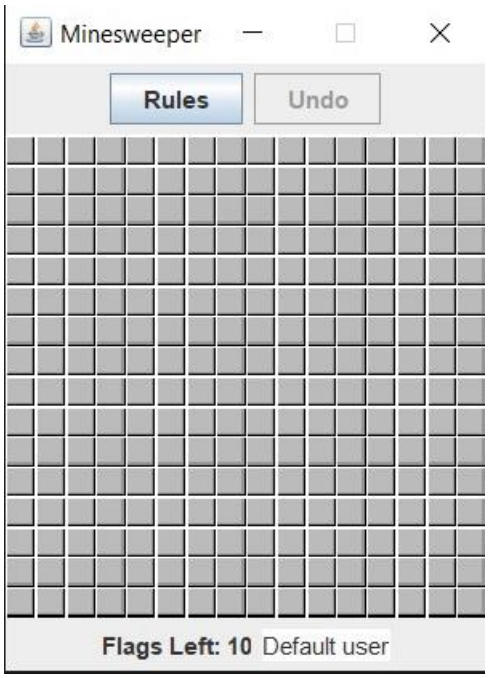
- Players start with a blank grid of squares representing a "minefield".
- Click on a square on the grid. If a square contains a mine (which is common for beginners), the game ends. Conversely, if there are no mines in that square, a field of squares will open with numbers. The number on a square represents the number of mines in the 8 surrounding squares.
- If a square is definitely mined, the player can mark it with a flag by right-clicking.
- If the neighboring squares of a square have enough mines and there are still other empty squares, then those squares are mine-free.
- The game ends with the player winning if all the squares without mines are opened

3. Data structure:

- Arrays (2D Arrays): Used to represent the game board and store information about mines, numbers, and revealed cells.

```
//2D Array of cells in gameboard
gameBoard = new Cell[N_ROWS][N_COLS];

for (int x = 0; x < N_ROWS; x++) {//initially have everything be empty
    for(int y=0; y < N_COLS; y++) {
        gameBoard[x][y] = new EmptyCell();
    }
}

statusbar.setText("Flags Left: " + Integer.toString(minesLeft));

int i = 0;
```

- Stack: Used for undo finding mines, uncovering cells function.



- HashMap: Implemented for save/load the game.

```
//Load Game if user saves game status

File statusFile = new File(STATUS_FILE);
if (statusFile.exists()) {

    String[] options = { "yes, please!", "no, thanks!" };
    int result = JOptionPane.showOptionDialog(parentComponent:null, message:"Do you want to restore the previous status?", title:"",
        JOptionPane.DEFAULT_OPTION, JOptionPane.INFORMATION_MESSAGE, icon:null, options, options[0]);

    if (result == 1) {
        newGame();
    } else {
        loadStatusFromFile();
        repaint();
    }

} else {
    newGame();
}
}
```

4. Algorithms:

- **Create the board:**

```
//Initializes the game board
private void initBoard() throws IOException {

    setPreferredSize(new Dimension(BOARD_WIDTH, BOARD_HEIGHT));
    images = new java.util.HashMap<>();


  //Put all relevant images in the map, some images named with integers, others named with descriptors
    for (int i = 1; i < 9; i++) {
        String path = "src/resources/" + i + ".png";
        images.put(Integer.toString(i), (new ImageIcon(path)).getImage());
    }

    images.put(key:"Bomb", (new ImageIcon(filename:"src/resources/Bomb.png")).getImage());
    images.put(key:"Covered", (new ImageIcon(filename:"src/resources/Covered.png")).getImage());
    images.put(key:"Empty", (new ImageIcon(filename:"src/resources/Empty.png")).getImage());
    images.put(key:"Marked", (new ImageIcon(filename:"src/resources/Marked.png")).getImage());
    images.put(key:"Wrongmarked", (new ImageIcon(filename:"src/resources/Wrongmarked.png")).getImage());

    addMouseListener(new MinesAdapter());

    showRules();
```

Therefore, the overall time complexity of the initBoard() method is constant time, O(1). This means that the method will execute its operations in a fixed amount of time, regardless of the size of any input or any external factors not accounted for in the method itself.

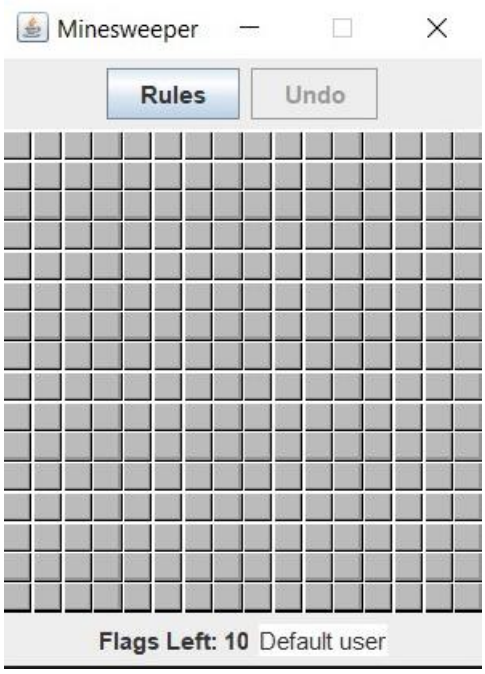
- **Neighbor:**

```
  //sets up neighbor cells
  for(int dx = -1; dx <= 1; dx++) {
      for(int dy = -1; dy <= 1; dy++) {
          if((dx != 0 || dy != 0) && positionX + dx < N_COLS && positionY + dy < N_ROWS
                  && positionX + dx >= 0 && positionY + dy >=0) {
              CellType typeOfCell = gameBoard[positionX + dx][positionY + dy].getCellType();
              if(typeOfCell != CellType.Bomb) {//not already a neighbor cell
                          if (typeOfCell != CellType.BombNeighbor) {
                              NeighborOfBombCell neighbor = new NeighborOfBombCell();
                                 neighbor.cellCount();
                                 gameBoard[positionX + dx][positionY + dy] = neighbor;
                          }
                          else {//already a neighbor cell, just need to update the neighbor count

                          gameBoard[positionX + dx][positionY + dy].cellCount();
              }
              }

          }
      }
  }
```

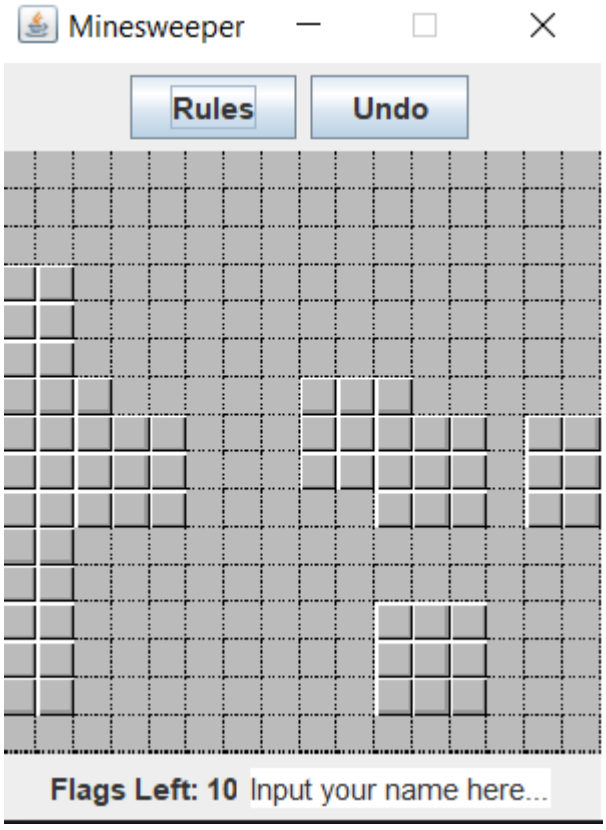
Hence, the overall time complexity of the provided code is O(1)×9=O(9), which simplifies to O(1).

5. Gameplay

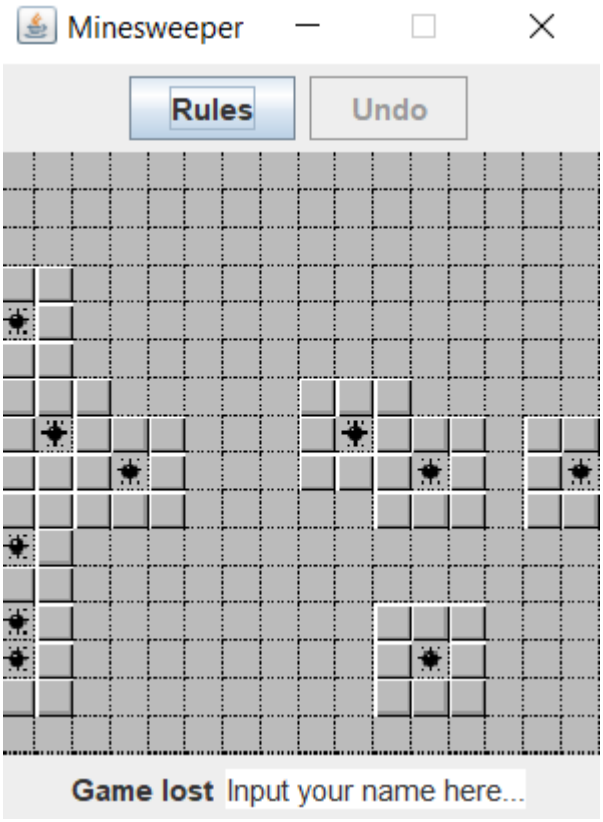
- The initialization of this game:



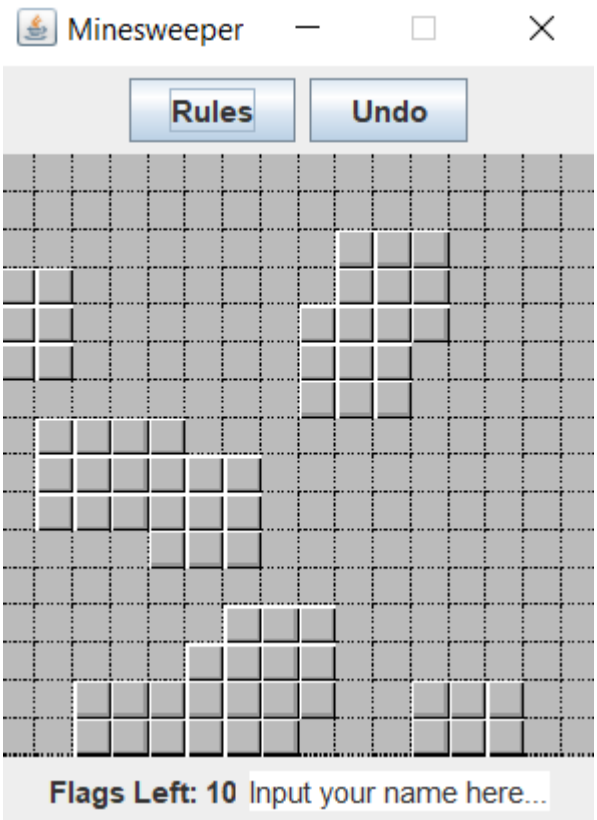
- Start the game: Click anywhere in the grid



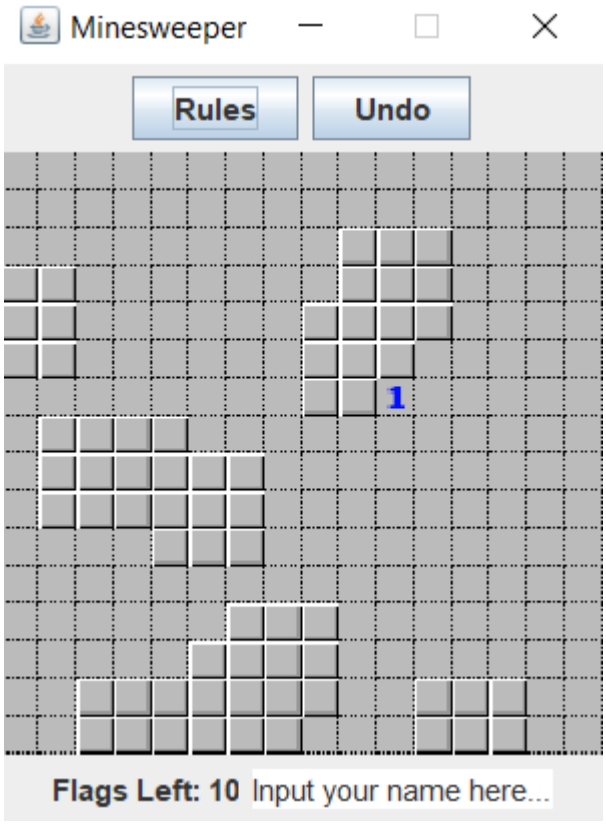
- Game over when you click the mine:

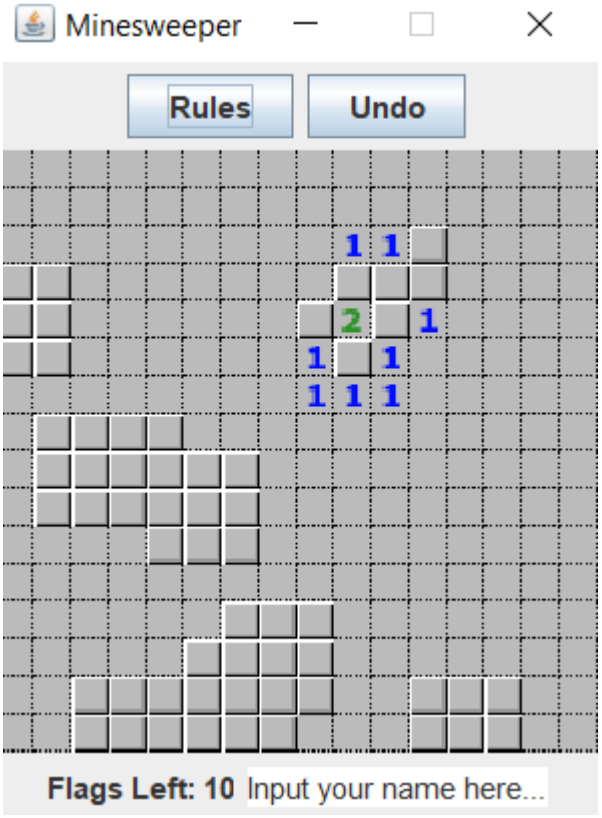


- Click anywhere to restart the game:

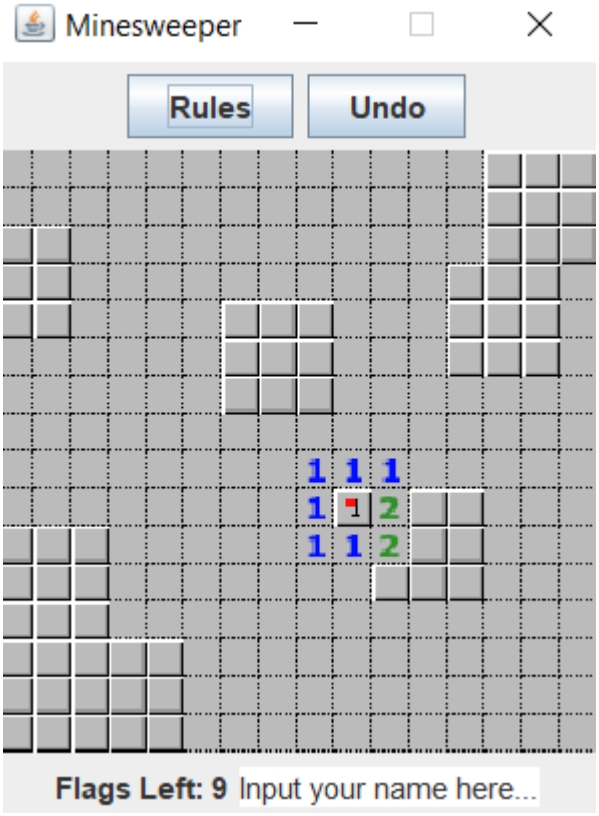


- It will show the number of mine around to help you know:





- Player can use right mouse to mark the square has a mine:



III. CONCLUSION & FUTURE WORK

1. Conclusion

The integration of Algorithm and Data Structure (DSA) concepts has greatly improved the structured and effective advancement of this game's development. Apart from the immediate advantages of enhanced code performance and upkeep, the utilization of DSA principles positions the project for easier incorporation of new features in the upcoming phases. The structured approach encouraged by DSA not only simplifies the current development process but establishes a groundwork for the continuous growth and enlargement of the game, ensuring its flexibility and durability.

2. Future work

In the future, the project will be update the time countdown, more mode to play, add hint to help player can know the mine where or can play with friends in this game. Optimizing the code for better performance and creating the leaderboards can create a sense of community and competition among players, enhancing them to improve their ability.