**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**🙟●🙝**

**A blue and white logo with a book and a square cap

Description automatically generated**

**ĐỒ ÁN MÔN CÔNG NGHỆ JAVA**

**XÂY DỰNG ỨNG DỤNG GAME MINESWEEPER MINI**

**Giáo viên giảng dạy: Huỳnh Hữu Nghĩa**

**Thành viên nhóm:**

Nguyễn Tấn Hữu Danh – 2001215653

Trương Đặng Minh Tân – 2001210185

Nguyễn Thanh Tùng – 2001216287

Nguyễn Duy Luân - 2001215937

**TP. Hồ Chí Minh, Ngày 04 Tháng 06 năm 2023**

**LỜI CÁM ƠN**

Tôi xin dành những dòng lời cảm ơn chân thành nhất đến những người đã đồng hành và hỗ trợ tôi trong hành trình nghiên cứu và thực hiện dự án này.

Đặc biệt, tôi muốn gửi lời tri ân đến giảng viên Huỳnh Hữu Nghĩa, người đã tận tâm dẫn dắt và truyền đạt kiến thức quý báu cho tôi. Nhờ có sự chỉ bảo và sự hỗ trợ không ngừng, dự án này đã có được sự hoàn thiện và phát triển không ngừng.

**BẢNG PHÂN CÔNG**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Họ và tên - MSSV** | **Công việc đảm nhận** | **Nhóm đánh giá kết quả** | **GV đánh giá** |
| 1 | Nguyễn Tấn Hữu Danh –2001215653 | Code : initBom(), init(int i, int j, float ratio), open(int i, int j), openEmpty(int i, int j), addFlag(int i, int j), loss(). | 100% |  |
| 2 | Trương Đặng Minh Tân – 2001210185 | actionPerformed(ActionEvent e), keyPressed(KeyEvent e), keyTyped(KeyEvent e), keyReleased(KeyEvent e). | 100% |  |
| 3 | Nguyễn Thanh Tùng – 2001216287 | Code hàm checkPoint(String s, int k), hightPoint(int k), saveGameHistory(), readGameHistory(), showRank(). | 100% |  |
| 4 | Nguyễn Duy Luân - 2001215937 | Code hàm actionPerformed(ActionEvent e), keyPressed(KeyEvent e), keyTyped(KeyEvent e), keyReleased(KeyEvent e). | 100% |  |

Chương 1. TỔNG QUAN ĐỀ TÀI

* 1. Giới thiệu đề tài.

Minesweeper là một trò chơi kinh điển, được phát triển lần đầu tiên vào những năm 1960 và phổ biến rộng rãi từ khi xuất hiện trên hệ điều hành Windows. Trò chơi đòi hỏi sự tư duy logic và khả năng phán đoán của người chơi nhằm phát hiện và tránh các ô có chứa mìn trên một bảng vuông. Đề tài "Xây dựng ứng dụng Game Minesweeper mini" nhằm mục tiêu tái hiện trò chơi này trên nền tảng hiện đại, với một giao diện đơn giản, dễ sử dụng và có khả năng mở rộng. Ứng dụng này sẽ giúp người dùng có thể giải trí, rèn luyện kỹ năng tư duy logic và phản xạ nhanh.

* 1. Nội dung đề tài.

Đề tài bao gồm các nội dung chính sau:

* Khảo sát và nghiên cứu trò chơi Minesweeper: Tìm hiểu về luật chơi, các phiên bản khác nhau của trò chơi, và các yếu tố làm nên thành công của Minesweeper.
* Thiết kế ứng dụng: Xác định các yêu cầu kỹ thuật, giao diện người dùng, và chức năng của ứng dụng Minesweeper mini.
* Phát triển ứng dụng: Lập trình và tích hợp các chức năng của trò chơi bao gồm:

+ Khởi tạo bảng chơi với các mức độ khó khác nhau.

+ Chức năng cắm cờ, mở ô và xử lý tình huống khi mở phải ô chứa mìn.

+ Tính toán và hiển thị thời gian chơi, điểm số.

* Kiểm thử và hoàn thiện ứng dụng: Kiểm tra tính ổn định, khả năng sử dụng và sửa lỗi phát sinh.
* Tài liệu hóa: Soạn thảo tài liệu hướng dẫn sử dụng, hướng dẫn cài đặt và các tài liệu kỹ thuật khác liên quan.
  1. Giới hạn đề tài.

Trong phạm vi đề tài này, chúng tôi sẽ tập trung vào các giới hạn sau:

* Phạm vi chức năng: Ứng dụng Minesweeper mini chỉ bao gồm các chức năng cơ bản của trò chơi Minesweeper cổ điển. Các tính năng mở rộng như bảng xếp hạng trực tuyến, chế độ nhiều người chơi, hoặc tích hợp các thuật toán AI nâng cao sẽ không được triển khai trong đề tài này.
* Nền tảng triển khai: Ứng dụng sẽ được phát triển và thử nghiệm trên nền tảng Windows. Các phiên bản cho các hệ điều hành khác (như macOS, Linux, Android, iOS) có thể được xem xét trong các nghiên cứu và phát triển tiếp theo.
* Giao diện người dùng: Giao diện sẽ được thiết kế đơn giản, tập trung vào tính trực quan và dễ sử dụng, không bao gồm các yếu tố thiết kế phức tạp hoặc yêu cầu đồ họa cao cấp.
* Hiệu năng: Ứng dụng sẽ được tối ưu hóa để chạy mượt mà trên các máy tính cá nhân với cấu hình trung bình. Các yêu cầu về hiệu năng cao cho các hệ thống có cấu hình đặc biệt sẽ không được đề cập chi tiết.

Qua việc xây dựng ứng dụng Game Minesweeper mini, đề tài này không chỉ giúp tái hiện một trò chơi cổ điển mà còn góp phần nâng cao khả năng lập trình, thiết kế và quản lý dự án của người thực hiện.

* 1. Cấu trúc báo cáo.

Phần I: Tổng Quan Đề Tài

1.1 Giới thiệu đề tài

1.2 Nội dung đề tài

1.3 Giới hạn đề tài

1.4 Cấu trúc báo cáo

Phần II: Cơ sở Lý Thuyết

2.1 Lịch sử và Phát triển của Minesweeper.

2.2 Luật chơi Minesweeper.

2.3 Các Thuật Toán Liên Quan.

2.4 Các Công Nghệ và Công Cụ Sử Dụng.

2.5 Các Kỹ Thuật Tối Ưu và Bảo Trì.

2.6 Ứng Dụng Thực Tiễn và Lợi Ích.

Phần III: Phương Pháp Nghiên Cứu

3.1 Hiện thực thuật toán leo đồi

3.2 Tạo môi trường thử nghiệm

3.3 Thu thập và xử lý dữ liệu

Phần IV: Kết Quả và Phân Tích

4.1 Kết quả thử nghiệm thuật toán leo đồi

4.2 So sánh hiệu suất với các phương pháp khác

Phần V: Kết Luận và Hướng Phát Triển

5.1 Tóm tắt kết quả

5.2 Đánh giá dự án và đề xuất hướng phát triển tương lai

Chương 2. Cơ sở lý thuyết

### 2.1 Lịch sử và Phát triển của Minesweeper

* **Lịch sử ra đời:** Trò chơi Minesweeper xuất hiện lần đầu tiên vào những năm 1960 và trở nên phổ biến khi được tích hợp trên hệ điều hành Windows từ phiên bản Windows 3.1.
* **Các phiên bản và biến thể:** Minesweeper có nhiều phiên bản và biến thể khác nhau qua các thời kỳ, từ các phiên bản đồ họa đơn giản đến những phiên bản có đồ họa phức tạp và chức năng mở rộng.

### 2.2 Luật Chơi Minesweeper

* **Mục tiêu của trò chơi:** Mục tiêu của người chơi là mở hết các ô không chứa mìn và cắm cờ vào các ô chứa mìn.
* **Cách chơi cơ bản:**
  + **Khởi tạo bảng chơi:** Bảng chơi là một lưới vuông chứa các ô, một số ô chứa mìn và số còn lại không chứa mìn.
  + **Cắm cờ:** Người chơi có thể cắm cờ để đánh dấu các ô mà họ cho là chứa mìn.
  + **Mở ô:** Khi người chơi mở một ô không chứa mìn, số lượng mìn ở các ô lân cận sẽ được hiển thị. Nếu mở phải ô chứa mìn, trò chơi sẽ kết thúc.
  + **Chiến thắng và thua cuộc:** Người chơi thắng khi mở hết các ô không chứa mìn và thua khi mở phải ô chứa mìn.

### 2.3 Các Thuật Toán Liên Quan

* **Thuật toán Khởi Tạo Bảng Chơi:**
  + **Ngẫu nhiên phân bố mìn:** Sử dụng thuật toán ngẫu nhiên để phân bố các mìn trên bảng chơi sao cho mỗi lần chơi đều khác nhau.
  + **Đảm bảo tính công bằng:** Đảm bảo người chơi luôn có ít nhất một bước đi an toàn từ lần mở ô đầu tiên.
* **Thuật toán Mở Ô:**
  + **Duyệt ô lân cận:** Sử dụng các thuật toán duyệt như BFS (Breadth-First Search) hoặc DFS (Depth-First Search) để mở các ô lân cận khi người chơi mở một ô không chứa mìn.
  + **Xử lý các trường hợp đặc biệt:** Xử lý các trường hợp như ô không có mìn xung quanh (mở liên tiếp các ô lân cận) và các ô ở biên, góc của bảng chơi.

### 2.4 Các Công Nghệ và Công Cụ Sử Dụng

* **Ngôn ngữ lập trình:** Lựa chọn ngôn ngữ lập trình phù hợp như Python, Java, C#, hoặc JavaScript tùy thuộc vào nền tảng và yêu cầu của ứng dụng.
* **Thư viện và Framework:**
  + **Giao diện người dùng:** Sử dụng các thư viện như Tkinter (Python), Swing (Java), WPF (C#) hoặc HTML/CSS/JavaScript để xây dựng giao diện người dùng.
  + **Quản lý sự kiện:** Sử dụng các cơ chế quản lý sự kiện để xử lý các hành động của người chơi như nhấp chuột, cắm cờ.
* **Phương pháp kiểm thử:** Sử dụng các công cụ và phương pháp kiểm thử để đảm bảo ứng dụng hoạt động ổn định và không có lỗi.

**2.5 Các Kỹ Thuật Tối Ưu và Bảo Trì**

* **Tối ưu hóa hiệu năng:**
  + **Giảm độ trễ khi mở ô:** Tối ưu thuật toán mở ô để giảm độ trễ khi mở các ô trên bảng chơi lớn.
  + **Quản lý bộ nhớ:** Đảm bảo việc sử dụng bộ nhớ hiệu quả, tránh các rò rỉ bộ nhớ.
* **Bảo trì và nâng cấp:**
  + **Thiết kế mã nguồn dễ bảo trì:** Sử dụng các nguyên tắc thiết kế phần mềm như SOLID để mã nguồn dễ bảo trì và mở rộng.
  + **Lập kế hoạch nâng cấp:** Xác định các tính năng có thể thêm vào trong tương lai và chuẩn bị mã nguồn sẵn sàng cho việc nâng cấp.

Chương 3. PHƯƠNG PHÁP THỰC HIỆN

3.1 Yêu cầu bài toán.

* + 1. Mô tả bài toán.

**Bài toán xây dựng ứng dụng Game Minesweeper mini** yêu cầu phát triển một trò chơi điện tử tương tự như trò chơi Minesweeper cổ điển. Mục tiêu của bài toán là tạo ra một phiên bản Minesweeper đơn giản nhưng đầy đủ chức năng cơ bản, bao gồm giao diện người dùng trực quan, tính năng chơi đơn giản và khả năng lưu trữ điểm số. Các chi tiết cụ thể của bài toán bao gồm:

* **Khởi tạo bảng chơi:**
* Bảng chơi có thể có nhiều kích thước khác nhau tùy thuộc vào mức độ khó: dễ, trung bình và khó.
* Số lượng mìn sẽ thay đổi theo kích thước của bảng và mức độ khó.
* **Chức năng cắm cờ và mở ô:**
* Người chơi có thể nhấp chuột trái để mở ô và nhấp chuột phải để cắm cờ lên ô mà họ nghi ngờ chứa mìn.
* Khi mở một ô không chứa mìn, số lượng mìn xung quanh ô đó sẽ được hiển thị.
* Nếu người chơi mở phải ô chứa mìn, trò chơi sẽ kết thúc.
* **Chức năng cắm cờ và mở ô:**
* Người chơi có thể nhấp chuột trái để mở ô và nhấp chuột phải để cắm cờ lên ô mà họ nghi ngờ chứa mìn.
* Khi mở một ô không chứa mìn, số lượng mìn xung quanh ô đó sẽ được hiển thị.
* Nếu người chơi mở phải ô chứa mìn, trò chơi sẽ kết thúc.
* **Hiển thị thông tin trò chơi:**
* Thời gian chơi: Hiển thị thời gian mà người chơi đã sử dụng từ khi bắt đầu trò chơi.
* Số lượng mìn còn lại: Hiển thị số lượng mìn chưa được cắm cờ.
* **Giao diện người dùng:**
* Giao diện thân thiện, dễ sử dụng, có các nút chức năng rõ ràng.
* Màu sắc và bố cục hợp lý để người chơi dễ dàng nhận biết các thông tin quan trọng.
* **Tính năng bổ sung:**
* Lưu trữ và hiển thị điểm số cao nhất.
* Khả năng bắt đầu lại trò chơi hoặc chọn mức độ khó khác ngay từ giao diện chính.
* **Yêu cầu về kỹ thuật:**
* Ứng dụng phải hoạt động mượt mà trên các máy tính cá nhân có cấu hình trung bình.
* Mã nguồn phải dễ bảo trì và mở rộng, tuân theo các nguyên tắc thiết kế phần mềm tốt.

Mô tả bài toán trên giúp định hình rõ ràng các yêu cầu và chức năng mà ứng dụng Minesweeper mini cần phải có, từ đó làm cơ sở cho việc thiết kế và triển khai phần mềm.

3.1.2 Mục tiêu.

Mục tiêu của đề tài "Xây dựng ứng dụng Game Minesweeper mini" là phát triển một phiên bản đơn giản và đầy đủ chức năng của trò chơi Minesweeper, với giao diện người dùng thân thiện, khả năng chơi mượt mà, và tích hợp các tính năng cơ bản như cắm cờ, mở ô, hiển thị thời gian chơi và lưu trữ điểm số cao nhất.

3.1.3 Ràng buộc.

Các ràng buộc liên quan đến đề tài "Xây dựng ứng dụng Game Minesweeper mini" bao gồm:

1. **Ràng buộc về kỹ thuật:**
   * Ứng dụng phải được phát triển bằng một ngôn ngữ lập trình phù hợp với nền tảng mục tiêu (như Python, Java, C#, hoặc JavaScript).
   * Ứng dụng phải chạy mượt mà trên các máy tính cá nhân có cấu hình trung bình.
   * Giao diện người dùng phải đơn giản, dễ sử dụng và thân thiện.
2. **Ràng buộc về chức năng:**
   * Ứng dụng phải có các chức năng cơ bản của Minesweeper như cắm cờ, mở ô, hiển thị số lượng mìn xung quanh ô đã mở.
   * Ứng dụng phải có các mức độ khó khác nhau và khởi tạo bảng chơi tương ứng với mỗi mức độ khó.
   * Ứng dụng phải có khả năng lưu trữ và hiển thị điểm số cao nhất.
3. **Ràng buộc về giao diện:**
   * Giao diện phải trực quan, dễ sử dụng, và có thể hiển thị rõ ràng các thông tin cần thiết như thời gian chơi, số lượng mìn còn lại.
   * Giao diện phải phản hồi nhanh chóng với các thao tác của người chơi (như nhấp chuột để mở ô hoặc cắm cờ).
4. **Ràng buộc về hiệu năng:**
   * Ứng dụng phải đảm bảo thời gian phản hồi nhanh khi người chơi mở ô hoặc cắm cờ.
   * Ứng dụng phải quản lý bộ nhớ hiệu quả để tránh rò rỉ bộ nhớ và đảm bảo hoạt động ổn định trong thời gian dài.
5. **Ràng buộc về bảo trì và mở rộng:**
   * Mã nguồn phải được viết theo các nguyên tắc thiết kế phần mềm tốt để dễ bảo trì và mở rộng.
   * Cấu trúc mã nguồn phải cho phép dễ dàng thêm các tính năng mới hoặc điều chỉnh các tính năng hiện có.
6. **Ràng buộc về kiểm thử:**
   * Ứng dụng phải được kiểm thử kỹ lưỡng để đảm bảo không có lỗi và hoạt động ổn định trên các cấu hình hệ thống khác nhau.
   * Phải có kế hoạch kiểm thử rõ ràng, bao gồm kiểm thử chức năng, kiểm thử giao diện và kiểm thử hiệu năng.

3.1.4 Dữ liệu đầu vào và đầu ra.

**Dữ liệu đầu vào:**

1. **Cấu hình trò chơi:**
   * **Kích thước bảng chơi:** Số hàng và cột của bảng chơi (ví dụ: 9x9, 16x16).
   * **Số lượng mìn:** Số mìn sẽ được đặt ngẫu nhiên trên bảng chơi, thay đổi theo mức độ khó (dễ, trung bình, khó).
2. **Tương tác của người chơi:**
   * **Nhấp chuột trái:** Để mở một ô trên bảng chơi.
   * **Nhấp chuột phải:** Để cắm hoặc gỡ cờ trên một ô.
   * **Nhấp chuột vào nút khởi động lại:** Để bắt đầu lại trò chơi.
   * **Chọn mức độ khó:** Lựa chọn từ các mức độ khó (dễ, trung bình, khó).

**Dữ liệu đầu ra:**

1. **Trạng thái của bảng chơi:**
   * **Ô mở:** Hiển thị số lượng mìn xung quanh (từ 0 đến 8) nếu ô không chứa mìn.
   * **Ô cắm cờ:** Hiển thị biểu tượng cờ khi người chơi cắm cờ trên ô nghi ngờ có mìn.
   * **Ô chứa mìn:** Hiển thị mìn khi người chơi mở phải ô chứa mìn (kết thúc trò chơi).
2. **Thông tin trò chơi:**
   * **Thời gian chơi:** Hiển thị thời gian đã trôi qua từ khi bắt đầu trò chơi.
   * **Số lượng mìn còn lại:** Hiển thị số lượng mìn chưa được cắm cờ (giảm dần khi người chơi cắm cờ).
   * **Thông báo kết thúc trò chơi:** Hiển thị thông báo chiến thắng hoặc thua cuộc khi trò chơi kết thúc (mở phải mìn hoặc hoàn thành nhiệm vụ).
3. **Điểm số:**
   * **Điểm số cao nhất:** Lưu trữ và hiển thị điểm số cao nhất mà người chơi đạt được trong các lần chơi trước đó (có thể bao gồm thời gian chơi tốt nhất).
   1. Hiện thực.

package com.mycompany.gamedomin;

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Map;

import java.util.TreeMap;

import javax.swing.\*;

public class GameDoMin extends JFrame implements ActionListener, KeyListener {

//so lan nhap

ArrayList<GameRecord> gameRecords = new ArrayList<>();

int countClicks = 0;

int BOM, dem = 0;

int score = 0;

int maxXY = 100;

boolean die = false;

Timer timer;

boolean flag = false;

private Color bom\_cl = Color.red;

int key\_flag = KeyEvent.VK\_H;

private Color background\_number\_cl = Color.yellow;

private Color background\_null\_cl = Color.gray;

private Color flag\_cl = Color.green;

private int m, n;

int M[] = {8, 15, 21};

int N[] = {10, 19, 27};

int Mines[] = {10, 40, 100};

private int values[][] = new int[maxXY][maxXY];

private JButton bt[][] = new JButton[maxXY][maxXY];

private JLabel point\_lb, hightPoint\_lb, temp\_lb, flag\_lb, lv\_lb;

private JComboBox lv = new JComboBox();

private boolean tick[][] = new boolean[maxXY][maxXY];//true = chưa flag, false = đã flag

private JButton repeat\_bt, mines\_bt, rank\_bt;

private JPanel pn0, pn, pn2;

private JLabel score\_lb = new JLabel("Điểm số: 0");

Container cn;

public GameDoMin(String s, int k) {

super(s);

BOM = Mines[k];

m = M[k];

n = N[k];

initBom();

cn = this.getContentPane();

pn = new JPanel();

pn.setLayout(new GridLayout(m,n));

for (int i = 0; i <= m + 1; i++)

for (int j = 0; j <= n + 1; j++){

bt[i][j] = new JButton(" ");

tick[i][j] = true;

}

for (int i = 1; i <= m; i++)

for (int j = 1; j <= n; j++){

bt[i][j] = new JButton(" ");

pn.add(bt[i][j]);

bt[i][j].setActionCommand(i + " " + j);

bt[i][j].addActionListener(this);

bt[i][j].addKeyListener(this);

}

point\_lb = new JLabel("00:00:00:00");

point\_lb.setFont(new Font("Arial", 1, 20));

hightPoint\_lb = new JLabel(hightPoint(k));

hightPoint\_lb.setFont(new Font("Arial", 1, 20));

temp\_lb = new JLabel(" ");

mines\_bt = new JButton(String.valueOf(BOM));

mines\_bt.setBackground(flag\_cl);

repeat\_bt = new JButton("Ván mới");

repeat\_bt.addActionListener(this);

rank\_bt = new JButton("Rank");

rank\_bt.addActionListener(this);

flag\_lb = new JLabel("Đang gỡ bom (h)");

flag\_lb.setForeground(bom\_cl);

flag\_lb.setFont(new Font("Arial", 1, 15));

lv\_lb = new JLabel("Mức độ: ");

lv\_lb.setFont(new Font("Arial", 1, 15));

lv.addItem("Dễ");

lv.addItem("Trung bình");

lv.addItem("Khó");

lv.setSelectedIndex(k);

pn2 = new JPanel();

pn2.setLayout(new FlowLayout());

pn2.add(lv\_lb);

pn2.add(lv);

pn2.add(mines\_bt);

pn2.add(repeat\_bt);

pn2.add(flag\_lb);

pn2.add(rank\_bt);

pn0= new JPanel();

pn0.setLayout(new FlowLayout());

pn0.add(point\_lb);

pn0.add(temp\_lb);

pn0.add(hightPoint\_lb);

cn.add(pn0, "North");

cn.add(pn);

cn.add(pn2, "South");

score\_lb.setFont(new Font("Arial", 1, 15));

pn0.add(score\_lb);

this.setVisible(true);

this.pack();

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

this.setLocationRelativeTo(null);

timer = new Timer(10, new ActionListener() {

public void actionPerformed(ActionEvent e) {

point\_lb.setText(next(point\_lb));

}

});

// for (int i = 1; i <= m; i++) {

// for (int j = 1; j <= n; j++)

// System.out.printf("%2s ", values[i][j]);

// System.out.println("\n");

// }

}

public void initBom() {

float ratio = (float) 0.05; // Tỉ lệ tạo mìn những ô xung quanh

int i, j;

for (i = 0; i <= m + 1; i++)

for (j = 0; j <= n + 1; j++)

values[i][j] = 0;

while (dem < BOM) {

// Chọn ra tạo độ chưa phải là mìn.

do {

i = (int) ((m - 1) \* Math.random()) + 1;

j = (int) ((n - 2) \* Math.random()) + 1;

} while (values[i][j] != 0);

if (values[i][j] == 0)

init(i, j, ratio);

}

}

public void init(int i, int j, float ratio) {

if (Math.random() < ratio) {

values[i][j] = -1;

for (int k = i - 1; k <= i + 1; k++)

for (int h = j - 1; h <= j + 1; h++)

if (values[k][h] != -1) {

values[k][h]++;

}

dem++;

//Gọi hàm tạo bom cho những ô xung quanh.

for (int k = i - 1; k <= i + 1; k++)

for (int h = j - 1; h <= j + 1; h++)

if (k > 0 && h > 0 && values[k][h] != -1 && dem < BOM)

init(k, h, ratio);

}

}

public String next(JLabel lb) {

String str[] = lb.getText().split(":");

int tt = Integer.parseInt(str[3]);

int s = Integer.parseInt(str[2]);

int m = Integer.parseInt(str[1]);

int h = Integer.parseInt(str[0]);

String kq = "";

int sum = tt + s \* 100 + m \* 60 \* 100 + h \* 60 \* 60 \* 100 + 1;

if (sum % 100 > 9)

kq = ":" + sum % 100 + kq;

else

kq = ":0" + sum % 100 + kq;

sum /= 100;

if (sum % 60 > 9)

kq = ":" + sum % 60 + kq;

else

kq = ":0" + sum % 60 + kq;

sum /= 60;

if (sum % 60 > 9)

kq = ":" + sum % 60 + kq;

else

kq = ":0" + sum % 60 + kq;

sum /= 60;

if (sum > 9)

kq = sum + kq;

else

kq = "0" + sum +kq;

return kq;

}

public void open(int i, int j) {//mở ô có số

if (tick[i][j] && values[i][j] != -1) {

bt[i][j].setText(String.valueOf(values[i][j]));

bt[i][j].setBackground(background\_number\_cl);

tick[i][j] = false;

checkWin();

}

}

public void openEmpty(int i, int j) {//mở ô rỗng

if (tick[i][j]) {

tick[i][j] = false;

bt[i][j].setBackground(background\_null\_cl);

checkWin();

for (int h = i - 1; h <= i + 1; h++)//loop chạy 9 ô xung quanh bt[i][j] và bản thân bt[i][j]

for (int k = j - 1; k <= j + 1; k++)

if (h >= 0 && h <= m && k >= 0 && k <= n) {

if (values[h][k] == 0 && tick[h][k])//nếu mở ra ô rỗng lại gọi hàm tiếp

openEmpty(h, k);

else

open(h, k);

}

}

}

public void addFlag(int i, int j) {

if (bt[i][j].getBackground() == flag\_cl) {//nếu đã flag thì unflag

tick[i][j] = true;//tắt flag cho ô

bt[i][j].setBackground(null);

mines\_bt.setText(String.valueOf(Integer.parseInt(mines\_bt.getText()) + 1));

}

else if (tick[i][j] && Integer.parseInt(mines\_bt.getText()) > 0) {//chưa flag thì flag

tick[i][j] = false;//bật flag cho ô

bt[i][j].setBackground(flag\_cl);

mines\_bt.setText(String.valueOf(Integer.parseInt(mines\_bt.getText()) - 1));//chỉnh số mìn hiển thị

}

checkWin();

}

public void checkWin() {

for (int i = 1; i <= m; i++)

for (int j = 1; j <= n; j++)

if (tick[i][j]) {

return;

}

int k = 0;

for (int i = 1; i <= m; i++)

for (int j = 1; j <= n; j++)

if (bt[i][j].getBackground() == flag\_cl && values[i][j] != -1)

return;

else if (bt[i][j].getBackground() == flag\_cl)

k++;

if (k <= BOM) {

checkAndShowScore(); // Gọi hàm kiểm tra điểm và hiển thị thông báo

}

}

public void checkPoint(String s, int k) throws IOException {

String file = "point.txt";

FileReader fr = new FileReader(file);

BufferedReader br = new BufferedReader(fr);

String str[] = {"", "", ""};

for (int i = 0; i <= 2; i++)

str[i] = br.readLine();

fr.close();

if (s.compareTo(str[k]) < 0) {

str[k] = s;

FileWriter f = new FileWriter(file);

f.write(str[0] + "\n");

f.write(str[1] + "\n");

f.write(str[2] + "\n");

f.flush();

f.close();

}

}

String hightPoint(int k) {

String file = "point.txt";

String str = "";

FileReader fr;

try {

fr = new FileReader(file);

BufferedReader br = new BufferedReader(fr);

try {

for (int i = 0; i <= k; i++)

str = br.readLine();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

} catch (FileNotFoundException e) {

String maxTime = "99:99:99:99";

// TODO Auto-generated catch block

FileWriter f;

try {

f = new FileWriter(file);

f.write(maxTime + "\n");

f.write(maxTime + "\n");

f.write(maxTime + "\n");

f.flush();

f.close();

return "99:99:99:99";

} catch (IOException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

}

return str;

}

public void loss() {//chạy vòng lặp để hiển thị các ô chứa bom còn lại

for (int i = 1; i <= m; i++)

for (int j = 1; j <= n; j++)

if (values[i][j] == -1) {

bt[i][j].setBackground(bom\_cl);

bt[i][j].setText("X");

}

}

public void actionPerformed(ActionEvent e) {

if (e.getActionCommand().equals(repeat\_bt.getText())) {

checkAndShowScore(); // Kiểm tra điểm trước khi tạo mới trò chơi

new GameDoMin("Game do min", lv.getSelectedIndex()/\*lấy độ khó người chơi chọn từ combobox\*/);

this.dispose();

}

else if (e.getActionCommand().equals(rank\_bt.getText())) {

showRank();

}

else if (!die) {

timer.start();

int i = 0,j = 0;

String s = e.getActionCommand();

System.out.println("s = "+s);//lấy tọa độ i, j

int k = s.indexOf(32);

System.out.println("k = " +k);//lấy index spacebar

i = Integer.parseInt(s.substring(0, k));

j = Integer.parseInt(s.substring (k + 1, s.length()));

if (tick[i][j]/\*chưa flag\*/ && !flag/\*không trong chế độ flag\*/) {

this.countClicks++;

if (values[i][j] == -1 /\*trúng bom\*/) {

if (bt[i][j].getBackground() != flag\_cl) {

bt[i][j].setBackground(bom\_cl);

bt[i][j].setText("X");

timer.stop();

loss();

JOptionPane.showMessageDialog(null, "Trúng Bom, bạn đã thua!");

die = true;

}

}

else if (values[i][j] == 0) {//ô rỗng thì mở những ô rỗng kế

System.out.println(i+" "+j);

openEmpty(i, j);

}

else {//ô có số thì mở mỗi ô đó th

System.out.println(i+" "+j);

open(i, j);

}

}

else if(flag){

addFlag(i, j);

}

}

}

public void keyPressed(KeyEvent e) {

if(e.getKeyCode() == key\_flag) {

flag = !flag;

}

if (flag) {

flag\_lb.setText("Đang cắm cờ (h)");

flag\_lb.setForeground(flag\_cl);

}

else {

flag\_lb.setText("Đang gỡ bom (h)");

flag\_lb.setForeground(bom\_cl);

}

}

public void checkAndShowScore() {

if (!die) {

int clicks = countClicks;

double efficiencyIndex = calculateEfficiencyIndex(clicks);

score = calculateScore(clicks, efficiencyIndex);

JOptionPane.showMessageDialog(this, "Chúc mừng! Bạn đã hoàn thành trò chơi với số điểm: " + score);

score\_lb.setText("Điểm số: "+score);

saveGameHistory();

die = true;

}

}

// Phuong thuc tinh hieu xuat

public double calculateEfficiencyIndex(int clicks) {

return (BOM \* 100.0) / (clicks + BOM);

}

public int calculateScore(int clicks, double efficiencyIndex) {

// Điểm = Số lần nhấp chuột \* (100 - Chỉ số hiệu suất)

return (int)(clicks \* (100.0 - efficiencyIndex));

}

private void showRank() {

gameRecords = readGameHistory();

//check du lieu

if (!gameRecords.isEmpty()) {

// Hiển thị dữ liệu xếp hạng

StringBuilder rankInfo = new StringBuilder();

int rank = 1;

for (GameRecord record : gameRecords) {

rankInfo.append("Rank ").append(rank).append(": Score: ").append(record.getScore()).append(", Time: ").append(record.getDate()).append(" ").append(record.getTime()).append("\n");

rank++;

}

JOptionPane.showMessageDialog(this, "Xếp hạng:\n" + rankInfo.toString(), "Xếp hạng", JOptionPane.INFORMATION\_MESSAGE);

} else {

JOptionPane.showMessageDialog(this, "Không có dữ liệu xếp hạng!", "Xếp hạng", JOptionPane.INFORMATION\_MESSAGE);

}

}

private void saveGameHistory() {

String fileName = "game\_history.txt";

try {

FileWriter writer = new FileWriter(fileName, true); // Append mode

BufferedWriter bufferedWriter = new BufferedWriter(writer);

bufferedWriter.write("Game History:\n");

bufferedWriter.write("Date: " + java.time.LocalDate.now() + "\n");

bufferedWriter.write("Time: " + java.time.LocalTime.now() + "\n");

bufferedWriter.write("Number of clicks:" + this.countClicks + "\n");

bufferedWriter.write("Score: " + score + "\n");

bufferedWriter.write("------------------------------------\n");

// Write additional game history information as needed

bufferedWriter.close();

System.out.println("Game history saved successfully!");

} catch (IOException e) {

e.printStackTrace();

System.out.println("Game history saved faile! \n");

}

}

private ArrayList<GameRecord> readGameHistory() {

String fileName = "game\_history.txt";

ArrayList<GameRecord> gameRecords = new ArrayList<>();

try {

FileReader fr = new FileReader(fileName);

BufferedReader br = new BufferedReader(fr);

String line;

while ((line = br.readLine()) != null) {

if (line.startsWith("Date: ")) {

String[] parts = line.split(":");

String date = parts[1].trim(); //lay phan tu dau

String time = br.readLine().split(":")[1].trim();

long clicks = Long.parseLong(br.readLine().split(":")[1].trim());

int score = Integer.parseInt(br.readLine().split(":")[1].trim());

br.readLine();

gameRecords.add(new GameRecord(date, time, clicks, score));

}

}

br.close();

fr.close();

} catch (IOException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Failed to load rank!", "Error", JOptionPane.ERROR\_MESSAGE);

}

return gameRecords;

}

public void keyTyped(KeyEvent e) {

}

public void keyReleased(KeyEvent e) {

}

public static void main(String[] args) {

new GameDoMin("CodeLearn.io - Game Dò Mìn", 0);

}

}

Chương 4. KẾT QUẢ THỰC HIỆN

4.1 Tối ưu hóa hiệu năng và phản hồi người dùng

#### Concurrency (Tính đồng thời)

Concurrency trong ứng dụng Minesweeper mini có thể được áp dụng để đảm bảo rằng giao diện người dùng luôn phản hồi mượt mà, ngay cả khi các tác vụ phức tạp đang được xử lý. Một số ví dụ cụ thể bao gồm:

* **Xử lý sự kiện người dùng:** Đảm bảo rằng các sự kiện nhấp chuột, cắm cờ, và mở ô được xử lý một cách đồng thời mà không làm chậm giao diện người dùng. Điều này có thể đạt được bằng cách sử dụng các luồng (threads) hoặc tác vụ không đồng bộ (asynchronous tasks) để xử lý các sự kiện này.
* **Cập nhật giao diện:** Khi người chơi mở một ô, giao diện cần được cập nhật ngay lập tức mà không bị đơ hoặc giật.

#### Parallelism (Song song)

Parallelism có thể không cần thiết trong mọi phần của ứng dụng Minesweeper mini, nhưng nó có thể hữu ích trong một số trường hợp cụ thể để tăng tốc độ xử lý. Một số ví dụ bao gồm:

* **Khởi tạo bảng chơi:** Phân bố mìn ngẫu nhiên và tính toán số lượng mìn xung quanh mỗi ô có thể được thực hiện song song để giảm thời gian khởi tạo bảng chơi, đặc biệt khi bảng chơi có kích thước lớn.
* **Kiểm thử hiệu năng:** Nếu ứng dụng cần kiểm tra nhiều cấu hình khác nhau hoặc chạy các bài kiểm tra hiệu năng, các tác vụ này có thể được thực hiện song song để tiết kiệm thời gian.

4.2 Kết quả.

Dem thoi gian

Diem so

- Mức dễ 10x8 và 10 quả mìn:

A screenshot of a computer

Description automatically generated

Lịch sử điểm

Số lượng mìn

Mức độ

* Mức trung bình và khó tăng kích thước bảng và số lượng mìn

A screenshot of a graph

Description automatically generated

A screenshot of a computer

Description automatically generated

* Kết quả khi thua

A screenshot of a computer game

Description automatically generated

A screenshot of a computer

Description automatically generated

Trạng thái cắm cờ

A screenshot of a computer

Description automatically generated

Ô có mìn

* Trang thái chiến thắng

A screenshot of a computer

Description automatically generated

* Lịch sử trò chơi.

A screenshot of a computer

Description automatically generated

Chương 5. KẾT LUẬN

5.1 Kết quả đạt được.

Trong quá trình xây dựng ứng dụng Game Minesweeper mini, chúng tôi đã đạt được những kết quả sau:

* Hoàn thành việc triển khai một phiên bản đầy đủ và hoạt động của trò chơi Minesweeper.
* Thiết kế và triển khai giao diện người dùng thân thiện và dễ sử dụng.
* Hiện thực các tính năng chính bao gồm mức độ khó, đếm thời gian, điểm số và xếp hạng người chơi.

5.2 Ưu và nhược điểm phương pháp đề xuất.

Ưu điểm:

* Đã xây dựng một ứng dụng chơi game hoàn chỉnh và thú vị với nhiều tính năng hấp dẫn.
* Giao diện người dùng thân thiện và dễ sử dụng, tạo ra trải nghiệm tốt cho người dùng.
* Hiệu suất của ứng dụng được cải thiện thông qua việc tối ưu hóa mã nguồn và sử dụng các kỹ thuật lập trình hiệu quả.

Nhược điểm:

* Một số tính năng có thể được mở rộng để cải thiện trải nghiệm người chơi, ví dụ: thêm âm thanh, hiệu ứng hoạt hình, hoặc chế độ đa người chơi.
* Cần phát triển thêm các công cụ quản lý lỗi và báo cáo lỗi để cải thiện tính ổn định của ứng dụng.
* Một số phần của mã nguồn có thể cần được tối ưu hóa thêm để đảm bảo hiệu suất tốt hơn đối với các mức độ khó lớn và bảng chơi lớn.

5.3 Hướng mở rộng

- Có một số hướng mở rộng có thể được thực hiện để nâng cao trải nghiệm người chơi và tính đa dạng của ứng dụng, bao gồm:

* Thêm tính năng âm thanh và hiệu ứng hoạt hình để làm cho trò chơi trở nên sống động hơn.
* Phát triển chế độ đa người chơi để cho phép người chơi cùng thi đấu hoặc cạnh tranh với nhau.
* Tăng cường tính tương tác với người dùng bằng cách thêm tính năng gợi ý và hướng dẫn.
* Phát triển phiên bản di động của ứng dụng để mở rộng phạm vi sử dụng đến các thiết bị di động.

TÀI LIỆU THAM KHẢO

[Minesweeper Strategy - Efficiency (minesweepergame.com)](https://minesweepergame.com/strategy/efficiency.php#:~:text=The%20statistic%20for%20efficiency%20is%20the%20Index%20of,50%20left%20clicks%20for%20an%20IOE%20of%201.00.)(10:46 pm 05-06-2024)

[Minesweeper Rules (minesweeper-online.org)](https://www.minesweeper-online.org/about/minesweeper-rules/#google_vignette)( 10:46 pm 05-06-2024)

[Authoritative Minesweeper (minesweepergame.com)](https://minesweepergame.com/)( 10:46 pm 05-06-2024)