**BỘ CÔNG THƯƠNG**



**TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

🙜🙢🙠🙞

**BÁO CÁO CÔNG NGHỆ JAVA**

**ĐỀ TÀI: XÂY DỰNG ỨNG DỤNG GAME MINESWEEPER MINI**

**NHÓM**

Giáo viên hướng dẫn:

Sinh viên thực hiện:

1. Nguyễn Tất Ngọc
2. Nguyễn Hồng Huy Bảo
3. Lê Minh Cường
4. Đỗ Hoàng Giang

TP. Hồ Chí Minh - 2025

# **BẢNG PHÂN CÔNG CÔNG VIỆC**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **MSSV** | **HỌ TÊN** | **CÔNG VIỆC ĐƯỢC GIAO** | **ĐÁNH GIÁ** |
| 1 | 2001223103 | Nguyễn Tất Ngọc | Áp dụng Concurrency vào các nút | 100% |
| 2 | 2001220303 | Nguyễn Hồng Huy Bảo | Giao diện, Tương tác, Nhạc nền | 100% |
| 3 | 2001225951 | Lê Minh Cường | Word, Áp dụng Concurrency vào Timer | 100% |
| 4 | 2001221046 | Đỗ Hoàng Giang | Word, Kiểm Thử, Database | 100% |

**MỤC LỤC**

[LỜI MỞ ĐẦU 6](#_Toc199506512)

[CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI 7](#_Toc199506513)

[1.1 Giới thiệu 7](#_Toc199506514)

[1.2 Mục tiêu đề tài 7](#_Toc199506515)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 9](#_Toc199506516)

[2.1 Khái niệm 9](#_Toc199506517)

[2.2 So sánh Concurrency và Parallelism 9](#_Toc199506518)

[2.3 Ưu và nhược điểm 9](#_Toc199506519)

[2.4 Ứng dụng thực tế 10](#_Toc199506520)

[2.5 Ứng dụng Concurrency và Parallelism trong trò chơi Minesweeper 10](#_Toc199506521)

[CHƯƠNG 3: PHÂN TÍCH HỆ THỐNG 13](#_Toc199506522)

[3.1 Các thành phần chính 13](#_Toc199506523)

[3.2 Các chức năng chính 13](#_Toc199506524)

[3.3 Cơ sở dữ liệu 16](#_Toc199506525)

[CHƯƠNG 4: THIẾT KẾ CHƯƠNG TRÌNH 19](#_Toc199506528)

[4.1 DAO 19](#_Toc199506529)

[4.2 MODEL 19](#_Toc199506530)

[4.3 UI 27](#_Toc199506531)

[CHƯƠNG 5: GIAO DIỆN NGƯỜI DÙNG 28](#_Toc199506532)

[5.1 Giao diện chính: 28](#_Toc199506533)

[5.2 Giao diện trước khi bắt đầu trò chơi 29](#_Toc199506535)

[5.3 Giao diện khi trò chơi bắt đầu 30](#_Toc199506537)

[CHƯƠNG 6: KẾT QUẢ THỰC HIỆN 34](#_Toc199506542)

[6.1 Giao diện người dùng 34](#_Toc199506543)

[6.2 Tương tác với người chơi 34](#_Toc199506544)

[6.3 Trạng thái trò chơi 34](#_Toc199506545)

[6.4 Khả năng mở rộng 35](#_Toc199506546)

[TÀI LIỆU THAM KHẢO 36](#_Toc199506547)

# **MỤC LỤC HÌNH ẢNH**

[*Hình 3.1: Sơ đồ Diagrams* 16](#_Toc199506526)

[*Hình 3.2: Hình ảnh BangXepHang* 18](#_Toc199506527)

[*Hình 5.1: Giao diện chính của trò chơi* 28](#_Toc199506534)

[*Hình 5.2: Giao diện của trò chơi* 29](#_Toc199506536)

[*Hình 5.3: Giao diện khi trò chơi bắt đầu* 30](#_Toc199506538)

[*Hình 5.4: Hình ảnh khi người chơi thua* 31](#_Toc199506539)

[*Hình 5.5: Hình khi người dùng chọn thay đổi màu nền* 32](#_Toc199506540)

[*Hình 5.7: Hình khi người dùng nhấn EXIT* 33](#_Toc199506541)

# **LỜI MỞ ĐẦU**

Trong thế giới phát triển phần mềm, việc học thông qua xây dựng các trò chơi kinh điển không chỉ là cách rèn luyện kỹ năng lập trình hiệu quả mà còn là cơ hội để khám phá và áp dụng sâu sắc các nguyên lý cốt lõi của công nghệ. Một trong những trò chơi đã gắn bó với tuổi thơ của biết bao thế hệ – Minesweeper (Dò mìn) – với cơ chế chơi đơn giản nhưng đầy thách thức về tư duy logic, là đề tài lý tưởng cho việc học và thực hành lập trình hướng đối tượng.

Với mong muốn củng cố kiến thức Java, đặc biệt là mảng thiết kế giao diện người dùng (GUI) và xử lý logic trò chơi, nhóm chúng em lựa chọn phát triển lại trò chơi Minesweeper dưới dạng ứng dụng desktop sử dụng Java Swing. Không chỉ dừng lại ở việc tái hiện một trò chơi giải trí quen thuộc, đồ án này còn hướng đến việc thể hiện tư duy hệ thống, thiết kế phần mềm có cấu trúc rõ ràng, dễ mở rộng và mang tính tương tác cao.

Thông qua đề tài, nhóm hy vọng có thể chứng minh rằng: kể cả những trò chơi nhỏ, đơn giản nhất cũng có thể là môi trường học tập lý tưởng để phát triển kỹ năng lập trình, tư duy thuật toán và khả năng làm việc nhóm trong lĩnh vực Công nghệ Thông tin.

# **CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI**

## **1.1 Giới thiệu**

Minesweeper (Dò mìn) là một trò chơi trí tuệ kinh điển xuất hiện từ rất sớm trên các hệ điều hành Windows. Mặc dù có lối chơi đơn giản, nhưng trò chơi yêu cầu người chơi phải sử dụng khả năng suy luận logic để phân tích và tìm ra vị trí các quả mìn được ẩn ngẫu nhiên trên một bảng lưới vuông. Trò chơi nhanh chóng trở nên phổ biến nhờ tính đơn giản, hấp dẫn và khả năng luyện tập tư duy cho người chơi.

Trong đồ án này, nhóm chúng em quyết định tái hiện trò chơi Minesweeper bằng cách sử dụng ngôn ngữ lập trình Java, tận dụng thư viện Swing để xây dựng giao diện người dùng đồ họa (GUI). Với mục tiêu rèn luyện kỹ năng lập trình hướng đối tượng và hiểu rõ cơ chế hoạt động của một trò chơi logic đơn giản, đề tài này giúp chúng em áp dụng đồng thời các kiến thức về:

* Thiết kế giao diện với Java Swing.
* Tư duy thuật toán để xử lý logic game.
* Quản lý trạng thái đối tượng trong trò chơi.
* (Tùy chọn) kết nối với cơ sở dữ liệu để lưu điểm.

Ngoài việc xây dựng được trò chơi hoàn chỉnh, đồ án còn hướng đến việc tối ưu trải nghiệm người dùng thông qua giao diện trực quan, hình ảnh minh họa (flag, bomb, smile...), và xử lý các thao tác chuột nhạy bén, mượt mà.

## **1.2 Mục tiêu đề tài**

Đề tài nhằm cụ thể hóa các mục tiêu kỹ thuật và học thuật sau:

Xây dựng một phiên bản trò chơi Minesweeper hoàn chỉnh, chạy được trên nền tảng Java.

Sử dụng thư viện Swing của Java để tạo GUI tương tác gồm bảng ô vuông, nút chơi lại, đồng hồ đếm thời gian, và bộ đếm mìn.

Thiết kế mã nguồn rõ ràng, dễ bảo trì, áp dụng các nguyên lý lập trình hướng đối tượng như phân tách lớp, đóng gói, kế thừa, và đa hình.

Hiển thị trực quan hình ảnh các trạng thái: cờ, bom, mặt cười, số mìn xung quanh...

Xử lý đầy đủ các tương tác: click trái mở ô, click phải đặt cờ, tự động mở lan các ô trắng.

Kiểm tra và thông báo thắng/thua theo logic trò chơi chuẩn.

Ngoài ra, mục tiêu học thuật gồm:

Củng cố kiến thức về Java, đặc biệt là lập trình hướng đối tượng và xây dựng GUI.

Rèn luyện tư duy logic, kỹ năng thiết kế thuật toán để giải quyết các bài toán lan rộng (recursive flood fill).

Làm quen với kết nối cơ sở dữ liệu SQL từ Java thông qua JDBC, nhằm lưu lại kết quả người chơi.

# **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT**

## **2.1 Khái niệm**

Concurrency (Tính đồng thời) là khả năng tổ chức nhiều tác vụ xảy ra cùng một thời điểm bằng cách chuyển đổi luân phiên giữa chúng. Mặc dù không thực sự chạy đồng thời trên nhiều lõi CPU, nhưng hệ thống cho phép các tác vụ "tiến triển cùng lúc" một cách hiệu quả.

Parallelism (Tính song song) là khả năng thực hiện nhiều tác vụ cùng lúc bằng cách tận dụng nhiều lõi xử lý của CPU. Trong Parallelism, các tác vụ thực sự được xử lý đồng thời về mặt vật lý.

## **2.2 So sánh Concurrency và Parallelism**

|  |  |  |
| --- | --- | --- |
| Tiêu chí | Concurrency (Đồng thời) | Parallelism (Song song) |
| Bản chất | Luân phiên xử lý nhiều tác vụ | Thực hiện nhiều tác vụ cùng lúc |
| CPU sử dụng | Một hoặc nhiều lõi (thường là 1) | Cần nhiều lõi để xử lý đồng thời |
| Mục đích | Tăng hiệu quả phản hồi, giữ cho hệ thống không bị chặn | Tăng hiệu suất thực thi bằng cách tận dụng tài nguyên phần cứng |
| Ứng dụng điển hình | Web server, xử lý sự kiện GUI | Xử lý dữ liệu lớn, tính toán khoa học |
| Khả năng mở rộng | Phụ thuộc vào cách tổ chức và lập lịch | Phụ thuộc vào số lõi CPU |

## **2.3 Ưu và nhược điểm**

Concurrency:

- Ưu điểm: Cho phép xử lý đa nhiệm trên hệ thống đơn lõi, cải thiện khả năng phản hồi.

- Nhược điểm: Cần kiểm soát tốt deadlock, race condition.

Parallelism:

- Ưu điểm: Tận dụng được toàn bộ sức mạnh phần cứng, xử lý khối lượng công việc lớn nhanh chóng.

- Nhược điểm: Phức tạp trong việc phân chia và đồng bộ hóa dữ liệu giữa các luồng.

## **2.4 Ứng dụng thực tế**

- Concurrency được ứng dụng trong các hệ thống yêu cầu nhiều tác vụ đồng thời như game, ứng dụng GUI, xử lý mạng, máy chủ web...

- Parallelism phù hợp với các bài toán cần hiệu suất tính toán cao như xử lý ảnh, học máy, mô phỏng vật lý, tổng hợp dữ liệu lớn...

## **2.5 Ứng dụng Concurrency và Parallelism trong trò chơi Minesweeper**

Trong quá trình phát triển trò chơi Minesweeper, nhóm đã áp dụng một số kỹ thuật liên quan đến Concurrency nhằm nâng cao hiệu năng và trải nghiệm người dùng. Cụ thể:

1. **Xử lý sự kiện chuột (Mouse Click) bằng Concurrency:**

Khi người dùng nhấn chuột để mở ô hoặc đánh dấu, các thao tác được xử lý trong luồng sự kiện (Event Dispatch Thread - EDT) của Java Swing.

Điều này giúp giao diện không bị "đóng băng" khi có nhiều sự kiện xảy ra liên tiếp, duy trì khả năng phản hồi của ứng dụng.

addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

if (SwingUtilities.isRightMouseButton(e)) {

toggleFlag();

} else if (SwingUtilities.isLeftMouseButton(e)) {

reveal();

}

}

});

1. **Phát âm thanh nền (nhạc):**

Âm thanh được phát bằng API javax.sound.sampled.Clip, cho phép phát lại âm thanh mà không chặn luồng hiện tại.

Điều này giúp trò chơi chạy mượt mà ngay cả khi nhạc đang phát liên tục, tránh được tình trạng giật/lag nếu âm thanh và giao diện cùng dùng chung một luồng.

public void playMusic(String filepath) {

try {

if (!hasPlayed) {

AudioInputStream audioInput=

AudioSystem.getAudioInputStream(

new File(filepath));

clip = AudioSystem.getClip();

clip.open(audioInput); clip.start();

clip.loop(Clip.LOOP\_CONTINUOUSLY); // lặp vô

hạn

hasPlayed = true;

}

} catch (Exception ex) {

ex.printStackTrace();

}

}

1. **Cập nhật bộ đếm thời gian:**

Bộ đếm thời gian được quản lý bằng javax.swing.Timer, lên lịch các sự kiện cập nhật giao diện người dùng trên Event Dispatch Thread (EDT) sau mỗi khoảng thời gian nhất định.

Điều này cho phép bộ đếm thời gian cập nhật mà không làm chặn các tương tác khác của người dùng với ứng dụng (ví dụ: click vào các ô mìn trong khi thời gian đang chạy).

private void startTimer() {

if (timer != null) {

timer.stop();

}

seconds = 0;

lblTimer.setText("Timer: 0 s");

lblTimer.setVisible(true);

lblTimer.repaint();

timer = new javax.swing.Timer(1000,

new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent e) {

seconds++;

lblTimer.setText("Timer: " + seconds + " s");

lblTimer.repaint();

}

});

timer.start();

}

1. **Ghi dữ liệu vào cơ sở dữ liệu:**

Khi người chơi chiến thắng, kết quả (tên và thời gian) được lưu vào cơ sở dữ liệu SQL Server thông qua JDBC.

# **CHƯƠNG 3: PHÂN TÍCH HỆ THỐNG**

## **3.1 Các thành phần chính**

Giao diện người dùng (GUI): sử dụng thư viện Swing của Java như JFrame, JPanel, JButton để hiển thị bảng chơi, các nút chức năng và đồng hồ.

Xử lý logic trò chơi: xử lý các thao tác người dùng (click chuột), cập nhật trạng thái của các ô, tính toán số mìn xung quanh, lan tỏa ô trắng, kiểm tra điều kiện thắng/thua.

Cơ sở dữ liệu (tùy chọn): lưu thông tin như tên người chơi, thời gian chơi, điểm số. Thực hiện qua kết nối JDBC đến file SQL.

## **3.2 Các chức năng chính**

Hệ thống trò chơi Minesweeper được phát triển nhằm mô phỏng trò chơi dò mìn kinh điển với các chức năng cơ bản nhưng không kém phần thử thách. Mỗi chức năng trong hệ thống đều đóng vai trò quan trọng trong việc mang đến trải nghiệm người dùng trực quan, mượt mà và hợp lý. Dưới đây là phân tích chi tiết các chức năng chính của hệ thống:

1. Hiển thị bảng chơi Minesweeper theo kích thước cố định.

Ngay khi khởi động trò chơi, hệ thống sẽ tiến hành khởi tạo một bảng chơi gồm các ô vuông được bố trí theo hàng và cột. Trong phiên bản hiện tại, kích thước mặc định là 10x10 tương ứng với 100 ô. Mỗi ô được biểu diễn bằng một thành phần đồ họa (control) trong giao diện người dùng, đảm bảo người chơi có thể dễ dàng tương tác bằng chuột.

Bảng chơi sẽ được lưu trữ dưới dạng một mảng hai chiều trong bộ nhớ, cho phép hệ thống dễ dàng truy cập và thao tác với từng ô riêng lẻ. Thông tin mỗi ô bao gồm: vị trí hàng/cột, trạng thái (đã mở/chưa mở), có mìn hay không, số lượng mìn lân cận và cờ đánh dấu.

1. Đặt mìn ngẫu nhiên lên các ô.

Sau khi bảng được tạo ra, hệ thống sẽ tiến hành rải ngẫu nhiên một số lượng mìn cố định trên các ô. Số lượng mìn mặc định là 10 quả trong bảng 10x10, tương đương với mật độ 10%.

Để đảm bảo tính ngẫu nhiên và công bằng, hệ thống sử dụng thuật toán chọn ngẫu nhiên các chỉ số hàng và cột sao cho không trùng lặp vị trí. Điều này đảm bảo rằng mỗi trò chơi mới đều là một thử thách hoàn toàn khác biệt với lần chơi trước.

1. Tính toán số lượng mìn xung quanh cho mỗi ô.

Khi đã xác định xong các vị trí chứa mìn, hệ thống sẽ thực hiện việc tính toán số lượng mìn có trong 8 ô lân cận của mỗi ô không phải là mìn. Nếu một ô không có mìn nhưng có các mìn xung quanh, hệ thống sẽ lưu trữ và hiển thị con số tương ứng (từ 1 đến 8).

Thuật toán này được thực hiện bằng cách duyệt qua tất cả các ô và kiểm tra vùng lân cận bằng các chỉ số hàng/cột liền kề (trên, dưới, trái, phải và các đường chéo). Việc xử lý vùng biên (ô ở cạnh bảng) cũng được quan tâm để tránh lỗi vượt chỉ số mảng.

1. Xử lý click chuột trái để mở ô, chuột phải để đánh cờ.

Giao diện người dùng hỗ trợ hai kiểu thao tác chuột, tương ứng với hai hành động cơ bản của trò chơi:

* **Chuột trái (Left click)**: Dùng để mở ô. Nếu ô chứa mìn, trò chơi kết thúc ngay lập tức với kết quả thua. Nếu ô không có mìn, số lượng mìn xung quanh (nếu có) sẽ được hiển thị. Trong trường hợp ô không có mìn và không có mìn xung quanh, hệ thống sẽ tự động mở tiếp các ô lân cận (xem chi tiết tại mục 5).
* **Chuột phải (Right click)**: Dùng để đánh dấu hoặc gỡ đánh dấu một ô bằng cờ. Việc này giúp người chơi lưu ý những vị trí nghi ngờ có mìn. Một ô chưa mở có thể chuyển đổi giữa ba trạng thái: không đánh dấu, đánh dấu bằng cờ, và đánh dấu bằng dấu hỏi (nếu áp dụng).

1. Tự động mở lan các ô trống không có mìn xung quanh.

Một trong những đặc điểm quan trọng của Minesweeper là mở lan (flood fill). Khi người chơi mở một ô không có mìn và số mìn xung quanh bằng 0, hệ thống sẽ tự động mở tất cả các ô lân cận không chứa mìn. Nếu trong số đó tiếp tục có các ô trống (0 mìn xung quanh), việc mở lan sẽ tiếp tục theo dạng đệ quy hoặc sử dụng hàng đợi để duyệt theo chiều rộng.

Việc mở lan giúp người chơi tiết kiệm thời gian và tăng khả năng quan sát các khu vực an toàn.

1. Kiểm tra điều kiện thắng hoặc thua.

Hệ thống định nghĩa hai trạng thái kết thúc trò chơi:

* **Thua cuộc**: Khi người chơi mở một ô có chứa mìn, trò chơi kết thúc ngay lập tức. Tất cả các mìn trên bảng sẽ được hiển thị, và các dấu cờ sai sẽ được làm nổi bật.
* **Thắng cuộc:** Khi người chơi đã mở toàn bộ các ô không chứa mìn, hệ thống sẽ thông báo chiến thắng. Khi đó, toàn bộ bảng sẽ được mở, các mìn sẽ được đánh dấu bằng cờ (nếu chưa đánh), và người chơi có thể khởi động trò chơi mới.

1. Giao diện hiển thị hình ảnh bom, cờ, mặt cười...

Giao diện đồ họa (GUI) là phần không thể thiếu để nâng cao trải nghiệm người dùng. Trong hệ thống, các biểu tượng được sử dụng bao gồm:

* **Hình bom (mìn)**: xuất hiện khi người chơi thua hoặc khi trò chơi kết thúc.
* **Hình cờ**: hiển thị khi người chơi đánh dấu một ô nghi ngờ có mìn.
* **Biểu tượng khuôn mặt (mặt cười / mặt buồn)**: thường được đặt ở đầu giao diện, dùng để thể hiện trạng thái của trò chơi (bắt đầu, đang chơi, thua, thắng).
* **Số lượng mìn còn lại**: thường được hiển thị dưới dạng bộ đếm, cho phép người chơi biết còn bao nhiêu mìn cần được đánh dấu.

## **3.3 Cơ sở dữ liệu**



## *Hình 3.1: Sơ đồ Diagrams*

Cơ sở dữ liệu gồm 2 bảng:

Bảng Players: Dùng để lưu tên của người chơi nếu người chơi chiến thắng. PlayerID(INT) được tạo tự động để đánh số thứ tự của người chơi, PlayerName(VARCHAR(50)) được dùng để lưu tên của người chơi.

Code tạo bảng Players:

CREATE TABLE Players (

PlayerID INT IDENTITY PRIMARY KEY,

PlayerName NVARCHAR(50) NOT NULL UNIQUE,

);

Bảng Scores: Dùng để lưu thời gian người chơi tham gia trò chơi. Thuộc tính PlayerID(INT) dùng liên với thuộc tính PlayerID của bảng Players mục đích để xác định xem là người chơi nào. Thuộc tính TimeTakenlnSeconds(INT) dùng để lưu thời gian mà người chơi tham gia trò chơi.

Code tạo bảng Scores:

CREATE TABLE Scores (

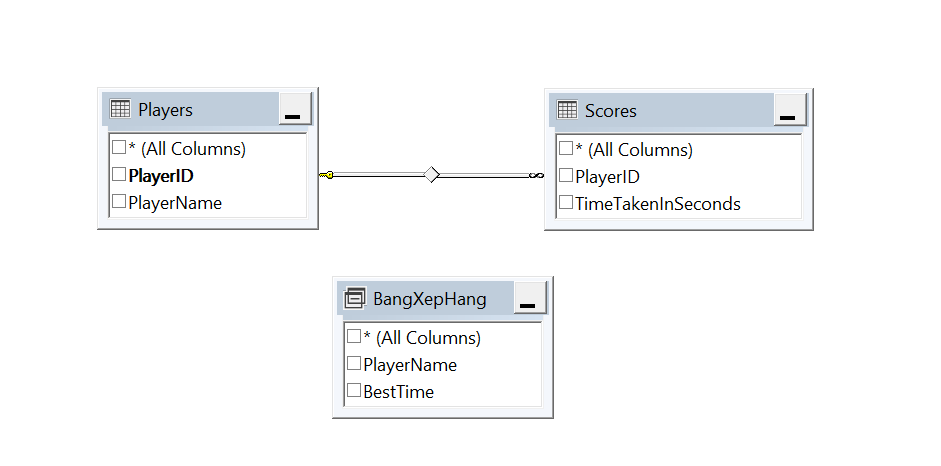
PlayerID INT FOREIGN KEY REFERENCES Players(PlayerID),

TimeTakenInSeconds INT NOT NULL,

);

GO

Ngoài ra, cơ sở dữ liệu còn tạo 1 view BANGXEPHANG để chứa thông tin PlayerID và thời gian người chơi chiến thắng trò chơi.



## *Hình 3.2: Hình ảnh BangXepHang*

Code tạo BangXepHang:

CREATE VIEW BangXepHang AS

SELECT

P.PlayerName,

MIN(S.TimeTakenInSeconds) AS BestTime

FROM Scores S

JOIN Players P ON S.PlayerID = P.PlayerID

GROUP BY P.PlayerName

GO

View tạo ra lưu 2 giá trị:

* PlayerName: là tên của người chơi, được lấy ra từ bảng Scores
* BestTime: Là thời gian thấp nhất mà người chơi chiến thắng, được lấy ra bằng cách sử dụng hàm MIN để lấy ra thời gian thấp của người chơi.

# **CHƯƠNG 4: THIẾT KẾ CHƯƠNG TRÌNH**

Chương trình được thiết kế theo mô hình 3 lớp(DAO-MODEL-UI):

## **4.1 DAO**

Chứa kết nối và các truy vấn đến Cơ sở dữ liệu.

Class KetNoiCSDL: Chứa phương thức kết nối đến cơ sở dữ liệu

* openConnection(): Tải Driver và kết nối đến cơ sở dữ liệu

Code:

public static Connection openConnection() throws Exception {

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

System.out.println("Loading...");

String url = "jdbc:sqlserver://localhost:1433;databaseName=MINESWEEPER;encrypt=true;trustServerCertificate=true";

String user="sa";

String password="123";

Connection con = DriverManager.getConnection(url, user, password);

System.out.println("Connecting...");

return con;

}

## **4.2 MODEL**

Chứa các lớp định nghĩa để lưu dữ giá trị của các bảng từ cơ sở dữ liệu.

Class Cell: Mở rộng từ Jbutton, đại diện cho 1 ô trong trò chơi, xử lý việc hiển thị, tương tác và logic của từng ô.

Thuộc tính:

* Row và col: Vị trí hàng và cột
* isMine: Xem ô có phải là mìn hay không
* isRevealed: Trạng thái ô đã mở hay chưa
* isFlagged: Ô đã được cắm cờ hay chưa
* neighborMines: Số lượng mìn xung quanh

Phương thức:

* Cell(int row, int col): Khởi tạo vị trí, kích thước, font chữ, cắm hoặc gỡ cờ, mở ô.
* toggleFlag(): Đánh dấu hoặc gỡ cờ ô chưa mở, hiển thị hình ảnh khi đặt cờ, xóa hình ảnh khi gỡ cờ
* reveal(): Được gọi khi người chơi nhấp chuột trái, hiển thị ‘M’ nếu ô chứa mìn, hiển thị số mìn xung quanh.
* Getter/Setter: Truy xuất và gán giá trị cho các thuộc tính.
* updateTheme(boolean darkMode): Đổi màu nền và màu chữ

Code:

public class Cell extends JButton {

private int row, col;

private boolean isMine;

private boolean isRevealed;

private boolean isFlagged;

private int neighborMines;

public Cell(int row, int col) {

this.row = row;

this.col = col;

this.isMine = false;

this.isRevealed = false;

this.isFlagged = false;

setPreferredSize(new Dimension(50, 50));

setFont(new Font("Arial", Font.BOLD, 20));

addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

if (SwingUtilities.isRightMouseButton(e)) {

toggleFlag();

} else if (SwingUtilities.isLeftMouseButton(e)) {

reveal();

}

}

});

}

private void toggleFlag() {

if (!isRevealed) {

isFlagged = !isFlagged;

if (isFlagged) {

URL url = getClass().getResource("/minesweeper/img/flag.jpg");

if (url != null) {

ImageIcon icon = new ImageIcon(url);

Image scaledImage = icon.getImage().getScaledInstance(24, 24, Image.SCALE\_SMOOTH);

setIcon(new ImageIcon(scaledImage));

} else {

System.err.println("Không tìm thấy ảnh flag.png!");

}

setText("");

} else {

setIcon(null);

setText("");

}

}

}

public void reveal() {

if (isFlagged || isRevealed) return;

isRevealed = true;

if (isMine) {

setText("M");

setBackground(Color.RED);

((Minesweeper) SwingUtilities.getWindowAncestor(this)).gameOver(false);

} else {

if (neighborMines > 0) {

setText(String.valueOf(neighborMines));

switch (neighborMines) {

case 1: setForeground(Color.BLUE); break;

case 2: setForeground(new Color(0, 128, 0)); break;

case 3: setForeground(Color.RED); break;

case 4: setForeground(new Color(128, 0, 128)); break;

default: setForeground(Color.BLACK); break;

}

}

setEnabled(false);

if (neighborMines == 0) {

((Minesweeper) SwingUtilities.getWindowAncestor(this)).revealNeighbors(row, col);

}

((Minesweeper) SwingUtilities.getWindowAncestor(this)).checkWin();

}

}

public boolean isMine() { return isMine; }

public void setMine(boolean isMine) { this.isMine = isMine; }

public void setNeighborMines(int count) { this.neighborMines = count; }

public boolean isRevealed() { return isRevealed; }

public boolean isFlagged() { return isFlagged; }

public void updateTheme(boolean darkMode) {

if (darkMode) {

this.setBackground(java.awt.Color.DARK\_GRAY);

this.setForeground(java.awt.Color.WHITE);

} else {

this.setBackground(null); // hoặc màu mặc định

this.setForeground(java.awt.Color.BLACK);

}

}

}

* Class Scores: Dùng để lưu tên và thời gian của người chơi

Thuộc tính:

* Name: Lưu tên người chơi
* Time: Lưu thời gian người chơi
* Phương thức: Khởi tạo, getter/setter

Code:

public class Score implements Comparable<Score> {

private String name;

private int time;

public Score(String name, int time) {

this.name = name;

this.time = time;

}

public String getName() { return name; }

public int getTime() { return time; }

@Override

public int compareTo(Score other) {

return Integer.compare(this.time, other.time);

}

@Override

public String toString() {

return name + " - " + time + " giây";

}

}

* Class SoundPlayer: Dùng để cung cấp trình phát âm thanh cho trò chơi

Thuộc tính:

* Chip: Đối tượng phát âm thanh
* hasPlayer: biến tĩnh, đảm bảo nhạc chỉ phát 1 lần duy nhất

Phương thức:

* playMusic(String filepath): phát nhạc từ đường dẫn filepath
* stopMusic(): Ngừng việc phát nhạc nếu đang phát

Code:

public class SoundPlayer {

private Clip clip;

private static boolean hasPlayed = false;

public void playMusic(String filepath) {

try {

if (!hasPlayed) {

AudioInputStream audioInput = AudioSystem.getAudioInputStream(new File(filepath));

clip = AudioSystem.getClip();

clip.open(audioInput);

clip.start();

clip.loop(Clip.LOOP\_CONTINUOUSLY); // lặp vô hạn

hasPlayed = true;

}

} catch (Exception ex) {

ex.printStackTrace();

}

}

public void stopMusic() {

if (clip != null && clip.isRunning()) {

clip.stop();

}

hasPlayed = false;

}

public boolean isPlaying() {

return clip != null && clip.isRunning();

}

public static boolean hasPlayed() {

return hasPlayed;

}

}

## **4.3 UI**

* Chứa các Jframe tương tác với người dùng.
* Jframe HomeFrame: Frame chính khi chạy bắt đầu chương trình, có các chức năng: Chơi, xem bảng xếp hạng, thoát, ngừng phát nhạc.
* Jframe BangXepHang: Hiển thị bảng xếp hạng
* Jframe Minesweeper: Hiển thị giao diện trò chơi, có các chức năng bắt đầu chơi, thay đổi màu hình nền, trở lại Frame trước.

# **CHƯƠNG 5: GIAO DIỆN NGƯỜI DÙNG**

## **5.1 Giao diện chính:**



## *Hình 5.1: Giao diện chính của trò chơi*

Giao diện có ảnh nền và các nút PLAY, BXH, EXIT, PAUSE/PLAY.

Khi người chơi chọn PLAY sẽ chuyển đến giao diện của trò chơi.

Khi người chơi chọn BXH sẽ hiển thị danh sách các người chơi và thời gian chiến thắng nhanh nhất của họ.

Khi người dùng chọn EXIT sẽ hiển thị thông báo và nếu người dùng xác nhận thoát thì trò chơi sẽ đóng.

Khi người dùng chọn II thì nhạc nền sẽ tắt nếu đang bật và ngược lại.

## **5.2 Giao diện trước khi bắt đầu trò chơi**



## *Hình 5.2: Giao diện của trò chơi*

Tại đây người chơi có thể nhập tên và bắt đầu trò chơi, thay đổi màu nền và quay trở lại trang trước.

Khi người dùng nhập tên và ấn START trò chơi sẽ bắt đầu.

Khi người dùng chọn Dark/Light Mode thì màu nền của trò chơi sẽ thay đổi thành tối nếu đang sáng và ngược lại.

Khi người dùng chọn BACK thì sẽ quay trở lại giao diện chính.

## **5.3 Giao diện khi trò chơi bắt đầu**



## *Hình 5.3: Giao diện khi trò chơi bắt đầu*

Tại đây người dùng sẽ bắt đầu trò chơi.

Khi người dùng chọn vào một ô sẽ hiển thị các ô có mìn xung quanh và số lượng.

Khi người dùng chọn phải một ô chứa mìn(thua) thì trò chơi sẽ hiển thị thông báo rằng người chơi đã thua và có muốn chơi lại hay không.

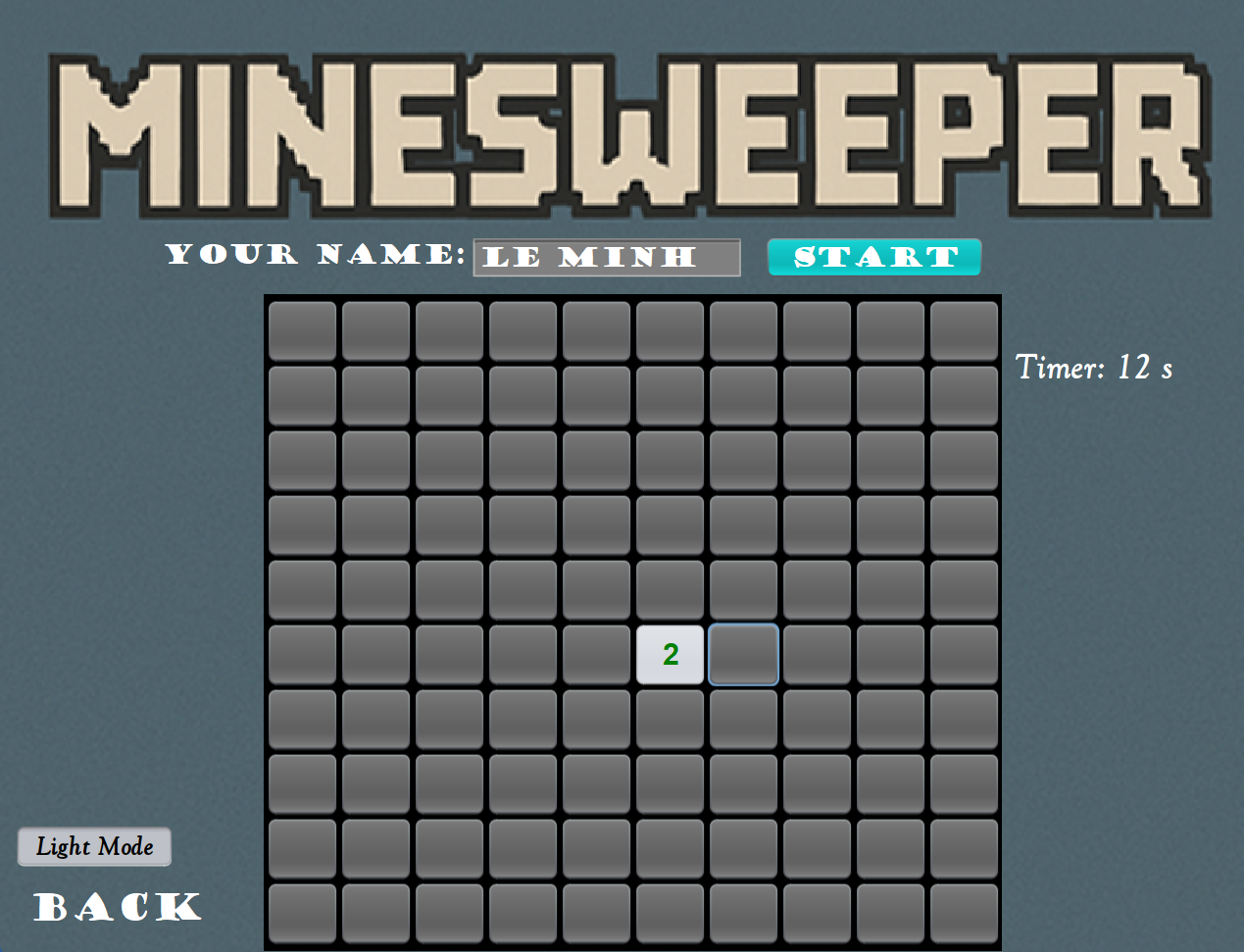
Khi người dùng đã chọn hết các ô không chứa mìn(thắng) thì người dùng sẽ được lưu vào cơ sở dữ liệu với tên và thời gian hoàn thành, trò chơi sẽ thông báo Bạn đã chiến thắng! và có muốn chơi lại hay không.



## *Hình 5.4: Hình ảnh khi người chơi thua*

Khi người dùng nhấn Yes thì trò chơi sẽ bắt đầu lại.

Khi người dùng nhấn No thì sẽ quay trở lại giao diện chính.



## *Hình 5.5: Hình khi người dùng chọn thay đổi màu nền*

Khi người dùng nhấn vào một lần nữa thì màu nền sẽ sáng lại.

**

*Hình 5.6: Hình khi người dùng nhấn vào BXH*



## *Hình 5.7: Hình khi người dùng nhấn EXIT*

Khi người dùng nhấn Yes thì sẽ thoát khỏi trò chơi.

# **CHƯƠNG 6: KẾT QUẢ THỰC HIỆN**

Sau khi hoàn thành quá trình phát triển và kiểm thử, ứng dụng game **Dò mìn (Minesweeper)** hoạt động ổn định và đáp ứng đầy đủ các chức năng đã đề ra.

## **6.1 Giao diện người dùng**

Giao diện trực quan, được thiết kế bằng thư viện Java Swing, hiển thị bảng chơi với các ô vuông kích thước đồng đều.

Các biểu tượng như bom, cờ, mặt cười được hiển thị rõ ràng giúp người chơi dễ dàng quan sát.

Hỗ trợ thay đổi giao diện sáng/tối giúp nâng cao trải nghiệm người dùng.

## **6.2 Tương tác với người chơi**

Click chuột trái: mở ô tương ứng. Nếu ô chứa mìn thì thua, nếu không thì hiện số lượng mìn lân cận.

Click chuột phải: đánh hoặc huỷ cờ để đánh dấu nghi ngờ ô có mìn.

Khi mở ô trống (không có mìn xung quanh), game sẽ tự động mở lan ra các ô lân cận không chứa mìn.

Hệ thống tự động kiểm tra điều kiện chiến thắng khi người chơi mở hết các ô không chứa mìn.

## **6.3 Trạng thái trò chơi**

Hiển thị thông báo rõ ràng khi người chơi thắng hoặc thua.

Sau khi kết thúc trò chơi, người chơi có thể bắt đầu lại ván mới một cách dễ dàng.

## **6.4 Khả năng mở rộng**

Hệ thống có thể dễ dàng tích hợp với cơ sở dữ liệu (SQL Server, MySQL...) để:

* Lưu điểm cao
* Ghi lại thời gian chơi, số lượt click, hoặc tên người chơi.

Có thể bổ sung thêm các mức độ khó khác nhau (dễ, trung bình, khó) với số lượng mìn và kích thước bảng khác nhau.

# **TÀI LIỆU THAM KHẢO**

[1] N. V. Hiếu, Giáo trình Java căn bản, Hà Nội: NXB Giáo dục Việt Nam, 2020.

[2] T. M. Đức, Lập trình hướng đối tượng với Java, Hà Nội: NXB Khoa học và Kỹ thuật, 2021.

[3] W3Schools, “Java JDBC Tutorial,” [Trực tuyến]. Có sẵn: https://www.w3schools.com/java/java\_jdbc.asp. [Truy cập: 28-May-2025].

[5] Khoa CNTT – Trường Đại học Công Thương TP.HCM, Tài liệu hướng dẫn môn học Công nghệ Java, 2025.