**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP.HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

-----o0o----

****

**ĐỒ ÁN**

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

**NHÓM: 9**

**Thành phố Hồ Chí Minh, tháng 5 năm 2023**

**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP.HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

-----o0o----

****

**ĐỒ ÁN**

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

**NHÓM: 9**

|  |  |
| --- | --- |
| **Thành viên: 4**  1. Trương Thành Trung - 2001203007  2. Nguyễn Thị Minh Thư -2001207013  3. Lê Huỳnh Phúc - 2001202198  4. Mai Nguyễn Phước Yến - 2001200561 | **Giảng viên hướng dẫn:**  **Thầy: Nguyễn Thế Hữu** |

**Thành phố Hồ Chí Minh, tháng 5 năm 2023**

**MỤC LỤC**

[NỘI DUNG 4](#_Toc135209479)

[I .Parallel Programming: 4](#_Toc135209480)

[**1.Lập trình song song trong Java là gì?** 4](#_Toc135209481)

[**2.Fork/Join Framework trong Java SE** 5](#_Toc135209482)

[**3.Chiến lược Fork/Join Framework** 7](#_Toc135209483)

[II.Tổng quan về Concurrency trong Java 9](#_Toc135209484)

[**1.Tìm hiểu về Concurrency** 10](#_Toc135209485)

[**1.1.Khái niệm** 10](#_Toc135209486)

[**1.2.Các phương thức xử lí concurrency thường được sử dụng** 10](#_Toc135209487)

[**2.Các phương thức và hoạt động của Concurrency.** 10](#_Toc135209488)

[**2.1.Các phương thức của concurrency** 10](#_Toc135209489)

[**2.2. Các hoạt động của Concurrency** 11](#_Toc135209490)

[III. Giới thiệu về Minesweeper: 12](#_Toc135209491)

[**1. Giới thiệu game:** 12](#_Toc135209492)

[**2. Luật chơi :** 12](#_Toc135209493)

[**3. Game Minesweeper :** 13](#_Toc135209494)

[**3.1 Giới thiệu ứng dụng :** 13](#_Toc135209495)

[**3.2 Giao diện :** 15](#_Toc135209496)

# **NỘI DUNG**

# **I .Parallel Programming:**

## **1.Lập trình song song trong Java là gì?**

* Lập trình song song là một quá trình chia nhỏ một vấn đề phức tạp thành các nhiệm vụ nhỏ hơn và đơn giản hơn, có thể được thực hiện đồng thời bằng cách sử dụng một số tài nguyên máy tính. Trong quá trình Lập trình song song, các tác vụ độc lập với nhau được thực thi song song bằng cách sử dụng các máy tính khác nhau hoặc nhiều lõi có trong CPU của máy tính. Lập trình song song là một nhu cầu thiết yếu đối với các công ty để xử lý các dự án nặng và quy mô lớn khi họ cần duy trì các tiêu chuẩn kinh tế của mình. Do Lập trình song song, tốc độ của dự án được tăng lên và xác suất xảy ra lỗi của dự án giảm xuống.
* Lập trình song song khá khác với **đa luồng** vì các tác vụ của nó không cần tuân theo thứ tự thực hiện. Mỗi tác vụ của Lập trình song song được thiết kế theo chức năng mà chúng cần thực hiện. Do đó, nó được biết đến rộng rãi là **Tính song song chức năng** hoặc **Tính song song dữ liệu**.
* Trong song song chức năng, mỗi bộ xử lý hoạt động trên phần vấn đề của nó, trong khi ở song song dữ liệu bộ xử lý hoạt động trên phần dữ liệu của nó. Lập trình song song phù hợp với cơ sở vấn đề lớn hơn không phù hợp với kiến ​​trúc CPU đơn lẻ hoặc có thể vấn đề quá lớn đến mức không thể giải quyết trong một khoảng thời gian ước tính hợp lý. Do đó, các tác vụ khi được phân phối giữa các bộ xử lý có thể thu được kết quả tương đối nhanh.
* **Tại sao Lập trình song song được sử dụng?**
* Vì CPU đa lõi đã được cải tiến trong thời gian gần đây, nên có thể đạt được hiệu quả lập trình ở mức tối đa bằng cách triển khai lập trình song song. Về cơ bản, lập trình song song đề cập đến việc thực thi song song các quy trình do có sẵn nhiều tài nguyên như lõi xử lý. Lập trình song song được coi là một phương pháp hiệu quả hơn nhiều so với đa luồng. Các nhiệm vụ song song được giải quyết đồng thời được kết hợp để đưa ra giải pháp cuối cùng cho một vấn đề lớn hơn.
* Môi trường tiêu chuẩn Java ( Java SE ) cung cấp cho lập trình viên " Fork/Join Framework ". Fork/Join Framework giúp lập trình viên triển khai Lập trình song song dễ dàng trong các ứng dụng của họ.

## **2.Fork/Join Framework trong Java SE**

* Fork/Join Framework được định nghĩa trong gói  **java.util.concurrent**. Nó bao gồm một số lớp và giao diện hỗ trợ lập trình song song. Những gì nó làm chủ yếu là nó đơn giản hóa quá trình tạo nhiều luồng, cách sử dụng chúng và tự động hóa cơ chế phân bổ quy trình giữa nhiều bộ xử lý. Ở đây, phần xử lý được tối ưu hóa để sử dụng nhiều bộ xử lý không giống như đa luồng, trong đó thời gian nhàn rỗi của một CPU được tối ưu hóa trên cơ sở chia sẻ thời gian. Ưu điểm bổ sung với khung này là sử dụng đa luồng trong môi trường thực thi song song.
* Các lớp của Fork/Join Framework:
* ForkJoinTask<V>: Đây là một lớp trừu tượng định nghĩa một tác vụ. Thông thường, một tác vụ được tạo với sự trợ giúp của phương thức fork() được định nghĩa trong lớp này. Nhiệm vụ này gần giống với một luồng bình thường được tạo bằng lớp Thread , nhưng nhẹ hơn nó. Cơ chế mà nó áp dụng là nó cho phép quản lý một số lượng lớn tác vụ với sự trợ giúp của một số lượng nhỏ các luồng thực tế tham gia ForkJoinPool . Phương thức fork () cho phép thực thi không đồng bộ tác vụ đang gọi. Phương thức join() cho phép đợi cho đến khi tác vụ mà nó được gọi cuối cùng kết thúc. Có một phương thức khác, được gọi là invoke() , kết hợp fork vàtham gia hoạt động vào một cuộc gọi duy nhất.
* ForkJoinPool: Lớp này cung cấp một nhóm chung để quản lý việc thực thi các tác vụ ForkJoinTask . Về cơ bản, nó cung cấp điểm vào để gửi từ các ứng dụng khách không phải ForkJoinTask , cũng như các hoạt động quản lý và giám sát.
* RecursiveAction: Đây cũng là một phần mở rộng trừu tượng của lớp ForkJoinTask . Thông thường, chúng tôi mở rộng lớp này để tạo một tác vụ không trả về kết quả hoặc có kiểu trả về void. Phương thức compute() được xác định trong lớp này được ghi đè để bao gồm mã tính toán của tác vụ.
* RecursiveTask<V>: Đây là một phần mở rộng trừu tượng khác của lớp ForkJoinTask . Chúng tôi mở rộng lớp này để tạo một tác vụ trả về kết quả. Và, tương tự như ResursiveAction, nó cũng bao gồm một phương thức compute() trừu tượng được bảo vệ . Phương pháp này được ghi đè để bao gồm phần tính toán của tác vụ.

## **3.Chiến lược Fork/Join Framework**

* Framework này sử dụng chiến lược *phân chia và chinh phục* đệ quy để thực hiện xử lý song song. Về cơ bản, nó chia một nhiệm vụ thành các nhiệm vụ nhỏ hơn sau đó, mỗi nhiệm vụ được chia thành các nhiệm vụ con. Quá trình này được áp dụng đệ quy trên từng tác vụ cho đến khi nó đủ nhỏ để xử lý tuần tự. Giả sử chúng ta tăng các giá trị của một mảng gồm *N* số. Đây là nhiệm vụ. Bây giờ, chúng ta có thể chia mảng cho hai để tạo hai nhiệm vụ con. Lại chia mỗi nhiệm vụ đó thành hai nhiệm vụ con nữa, v.v. Bằng cách này, chúng ta có thể áp dụng *chiến thuật chia để trị* chiến lược một cách đệ quy cho đến khi các nhiệm vụ được tách ra thành một bài toán đơn vị. Sự cố đơn vị này sau đó có thể được thực thi song song bởi nhiều bộ xử lý lõi có sẵn. Trong một môi trường không song song, những gì chúng ta phải làm là duyệt qua toàn bộ mảng và thực hiện xử lý theo trình tự. Đây rõ ràng là một cách tiếp cận không hiệu quả theo quan điểm xử lý song song. Nhưng, câu hỏi thực sự là mọi vấn đề có thể được *phân chia và chinh phục*? Tất nhiên là không! Tuy nhiên, có những vấn đề thường liên quan đến một số loại mảng, bộ sưu tập, nhóm dữ liệu đặc biệt phù hợp với phương pháp này. Tuy nhiên, có những vấn đề không thể sử dụng việc thu thập dữ liệu nhưng vẫn có thể được tối ưu hóa để sử dụng chiến lược lập trình song song.
* **Ví dụ chương trình sử dụng Fork/ Join framework trong Java**
* Chương tính tổng của một mảng số nguyên

import java.util.concurrent.\*;

public class DemoParallel extends RecursiveTask<Integer> {

private static final int THRESHOLD = 5;

private int[] array;

private int start;

private int end;

public DemoParallel(int[] array, int start, int end) {

this.array = array;

this.start = start;

this.end = end;

}

@Override

protected Integer compute() {

if (end - start <= THRESHOLD) {

int sum = 0;

for (int i = start; i < end; i++) {

sum += array[i];

}

return sum;

} else {

int mid = (start + end) / 2;

DemoParallel leftTask = new DemoParallel(array, start, mid);

DemoParallel rightTask = new DemoParallel(array, mid, end);

leftTask.fork();

int rightSum = rightTask.compute();

int leftSum = leftTask.join();

return leftSum + rightSum;

}

}

public static void main(String[] args) {

// TODO code application logic here

int[] array = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

ForkJoinPool pool = new ForkJoinPool();

DemoParallel task = new DemoParallel(array, 0, array.length);

int sum = pool.invoke(task);

System.out.println("Sum: " + sum);

}

}

* Trong ví dụ này chúng ta tạo một lớp ‘DemoParallel’ kế thừa từ ‘RecursiveTask<Integer>’ để tính tổng của một mảng số nguyên. Nếu kích thước của mảng nhỏ hơn hoặc bằng một ngưỡng `THRESHOLD chúng ta tính tổng trực tiếp. Nếu không chúng ta chia mảng thành hai phần và tạo hai ‘DemoParallel’ con để tính tổng của từng phần. Sau đó chúng ta sử dụng phương thức fork() để bắt đầu tính tổng của phần bên phải và sử dụng phương thức `join()` để đợi kết quả của phần bên trái. Cuối cùng chúng ta sử dụng ForkJoinPool để thực thi DemoParallel’ và tính tổng của mång.

# **II.Tổng quan về Concurrency trong Java**

Java là một ngôn ngữ lập trình đa luồng có nghĩa là chúng ta có thể phát triển chương trình đa luồng bằng Java. Một chương trình đa luồng chứa hai hoặc nhiều phần có thể chạy đồng thời và mỗi phần có thể xử lý một tác vụ khác nhau cùng lúc để sử dụng tối ưu các tài nguyên có sẵn, đặc biệt khi máy tính của bạn có nhiều CPU.

Theo định nghĩa, đa nhiệm là khi nhiều quy trình chia sẻ tài nguyên xử lý chung như CPU. Đa luồng mở rộng ý tưởng về đa nhiệm vào các ứng dụng nơi bạn có thể chia nhỏ các hoạt động cụ thể trong một ứng dụng thành các luồng riêng lẻ. Mỗi luồng có thể chạy song song. Hệ điều hành phân chia thời gian xử lý không chỉ giữa các ứng dụng khác nhau mà còn giữa từng luồng trong một ứng dụng.

Đa luồng cho phép bạn viết theo cách mà nhiều hoạt động có thể tiến hành đồng thời trong cùng một chương trình.

## **1.Tìm hiểu về Concurrency**

### **1.1.Khái niệm**

Concurrency là khả năng của một hệ thống xử lí nhiều tác vụ đồng thời. Trong lập trình, concurrency thường được sử dụng để tăng tốc việc xử lí dữ liệu, tối ưu hóa tài nguyên.

### **1.2.Các phương thức xử lí concurrency thường được sử dụng**

- Multi-threading: là một phương thức cho phép các tác vụ được thực thi đồng thời trên các thread riêng biệt trong cùng một tiến trình. Mỗi thread sẽ có thể thực hiện một tác vụ riêng mà không ảnh hưởng đến các thread khác.

-Multiprocessing: là một phương thức cho phưps các tác vụ được thực thi đồng thời trên các process riêng biệt, mỗi process có thể thực hiện một tác vụ riêng biệt mà không ảnh hương đến các process khác

-Coroutine: là một phương thức cho phép tác vụ chia sẻ cùng một tiến trình hoặc process và chuyển đổi giữa các tác vụ một cách linh hoạt, giúp tối ưu tài nguyên và tăng tốc độ xử lí.

## **2.Các phương thức và hoạt động của Concurrency.**

### **2.1.Các phương thức của concurrency**

Java cung cấp khả năng kiểm soát hoàn toàn chương trình đa luồng. Chúng ta có thể phát triển một chương trình đa luồng có thể bị tạm dừng, tiếp tục hoặc dừng hoàn toàn dựa trên yêu cầu.Có nhiều phương thức tĩnh khác nhau mà chúng ta có thể sử dụng trên các đối tượng luồng để kiểm soát hành vi của chúng. Bảng sau liệt kê các phương pháp đó:

|  |  |
| --- | --- |
| Phương thức | Mô tả |
| public void suspend | Phương thức này đặt một luồng ở trạng thái bị treo và có thể được tiếp tục lại bằng cách sử dụng phương thức resume (). |
| public void stop() | Phương thức dừng một chủ đề hoàn toàn |
| public void resume() | Phương thức này tiếp tục một luồng đã bị tạm ngưng bằng cách sử dụng phương pháp Susan() |
| public void wait() | Làm cho luồng hiện tại đợi cho đến khi mộ luồng khác gọi thông báo() |
| public void notify() | Đánh thức một luồng duy nhất đang chờ trên màn hình của đối tượng này |

### **2.2. Các hoạt động của Concurrency**

- Tạo và quản lí các thread : Java cung cấp cho chúng ta lớp Thread để tạo và quản lí các thread, chúng ta có thể sử dụng phương thức start() để bắt đầu thực thi một thread.

-Đồng bộ hóa thread: điều này đảm bảo rằng các thread không gây xung đột với nhau khi truy cập các tài nguyên chung. Java cung cấp các cơ chế đồng bộ hóa như synchronized methods và synchronized blocks để đảm bảo an toàn khi truy cập các tài nguyên chia sẻ.

- Giao tiếp giữa các thread: Java cung cấp các cơ chế giao tiếp giữa các thread, chẳng hạn như phương thức wait(), notify() và notifyAll() để các thread có thể chờ đợi và thông báo cho nhau.

-Xử lí các vấn đề liên quan đến deadlock và starvation: Java cung cấp các kĩ thuật để giải quyết các vấn đề liên quan đến deadlock và starvation, bao gồm khóa và các công cụ giám sát để phát hiện và giải quyết các vấn đề này.

- Tối ưu hóa hiệu suất: Java cung cấp các công cụ để tối ưu hóa hiệu suất, chẳng hạn như ThreadPoolExecutor để quản lí một nhóm thread và ExecutorService để quản lí và thực thi các Runnable hoặc Callable task . Java cũng cung cấp các cơ chế scheduling để phân bổ tài nguyên thực thi của CPU một cách hiệu quả và giảm thiểu thời gian chờ đợi của các thread.

# **III**. **Giới thiệu về Minesweeper:**

## **1. Giới thiệu game:**

**Minesweeper** hay còn được gọi bằng một cái tên khác là **Dò mìn hoặc Gỡ bom**, đây là một trò chơi được xây dựng như một chương trình giải trí cho người dùng sử dụng Microsoft Windows được chơi trên máy tính. Đây cũng là trò chơi được bắt nguồn từ trò chơi Cube được phát hành năm 1973.

Đến với Minesweeper, nhiệm vụ của người chơi là **mở hết các ô vuông mà không được bấm vào các ô có chứa mìn** và nếu như bạn bấm vào ô vuông có chứa mìn thì bạn sẽ thua ngay lập tức.

Có ba cấp độ chơi trong Minesweeper:

* Dễ (beginner): bảng ô vuông 8 × 8 (9×9 từ bản Microsoft Windows XP trở đi) trên đó rải 10 quả mìn.
* Trung bình (intermediate): bảng ô vuông 16 × 16 trên đó rải 40 quả mìn.
* Khó (expert): bảng ô vuông 30 × 16 trên đó rải 99 quả mìn.

**2. Luật chơi :**

 Mục tiêu cuối cùng của người chơi là mở hết tất cả các ô không chứa mìn, đó là luật chơi chung của Minesweeper :

* Người chơi bắt đầu với một bảng ô vuông trống, đây được xem là “bãi mìn”. Tùy vào cấp độ khó bạn chọn mà trò chơi sẽ hiện ra số ô khác nhau.
* Bạn chọn vào một ô vuông bất kỳ bằng cách click chuột trái. Có hai trường hợp xảy ra: Nếu không may trúng phải ô có mìn thì trò chơi ngay lập tức kết thúc và bạn sẽ phải bắt đầu lại. Còn nếu ô đó không chứa mìn thì một vùng các ô khác sẽ được mở ra cùng với những con số. Mỗi số trên một ô tương ứng cho chỉ dẫn số lượng mìn trong 8 ô nằm quanh liền kề với ô đó.
* Nếu bạn suy đoán và chắc chắn một ô có mìn thì hãy đánh dấu vào ô đó bằng hình lá cờ bằng cách click chuột phải (cắm cờ).
* Nếu như 8 ô lân cận của một ô đã có đủ số mìn như con số ở ô đó hiển thị mà vẫn còn các ô trống khác thì theo quy luật những ô đó không có mìn.
* Bạn sẽ giành chiến thắng nếu như mở được tất cả các ô không có mìn.

## **3. Game Minesweeper :**

### **3.1 Giới thiệu ứng dụng :**

**a/ Chức năng :**

* Admin :
* Quản lý tài khoản : Tạo tài khoản mới, xóa tài khoản cũ , cập nhật lại thông tin tài khoản
* Quản lý bảng xếp hạng : Xóa dữ liệu người chơi
* Quản lý đăng nhập
* Chơi game gồm các chế độ dễ, trung bình, khó
* Người chơi **:**
* Chơi game gồm các chế độ dễ, trung bình, khó
* Xem bảng xếp hạng

**b/ Phi Chức năng:**

* Thiết kế đơn giản
* Dễ sử dụngss
* Dễ bảo trì và nâng cấp
* Phù hợp với mọi thiết bị

**c/ Cơ sở dữ liệu :**

Gồm 3 bảng : TAIKHOAN, CHEDO , DIEM

CREATE TABLE [dbo].[TAIKHOAN](

[User] [varchar](50) PRIMARY KEY ,

[Password] [char](20) not null,

[Quyen] int not null,

[TrangThai] varchar(10),

)

CREATE TABLE [dbo].[CHEDO](

[MaCD] [int] primary key,

[TenCD] [nvarchar](50) not null,

[TrangThai] varchar(10),

[ThoiGian] int

)

CREATE TABLE [dbo].[DIEM](

[MaTK] int IDENTITY(1,1) primary key ,

[Player] [varchar](50),

[MaCD] [int] NOT NULL,

[KETQUA] int ,

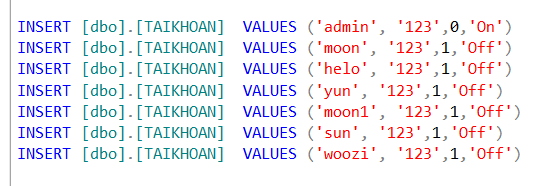
[THOIGIAN] int ,

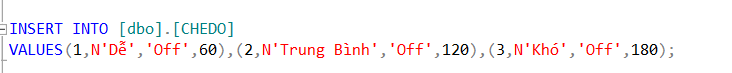
FOREIGN KEY (MaCD) REFERENCES [CHEDO] ([MaCD]),

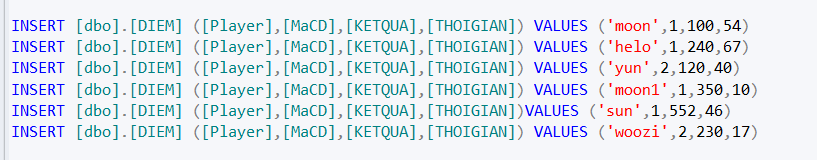
FOREIGN KEY ([Player]) REFERENCES [TAIKHOAN] ([User])

)

Nhập dữ liệu cho các bảng :

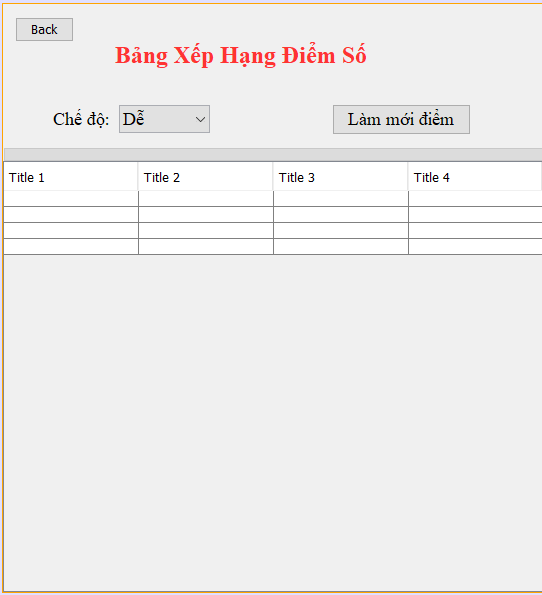






### **3.2 Giao diện :**

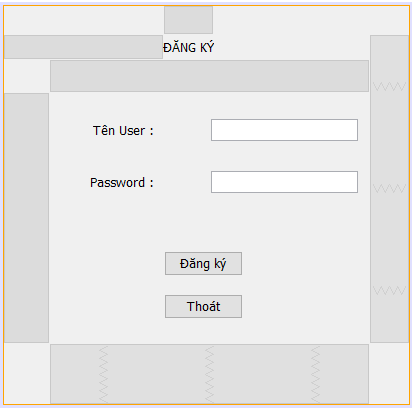
a/ Giao diện bảng xếp hàng dành cho người chơi :



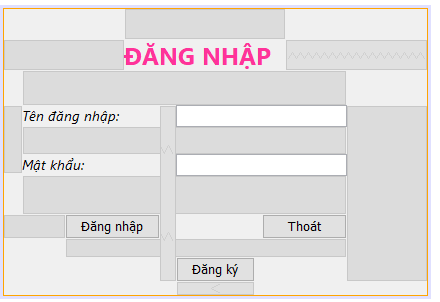
b/ Giao diện bảng xếp hạng dành cho Admin:



c/ Giao diện đăng ký tài khoản mới dành cho người chơi :



d/ Giao diện đăng nhập



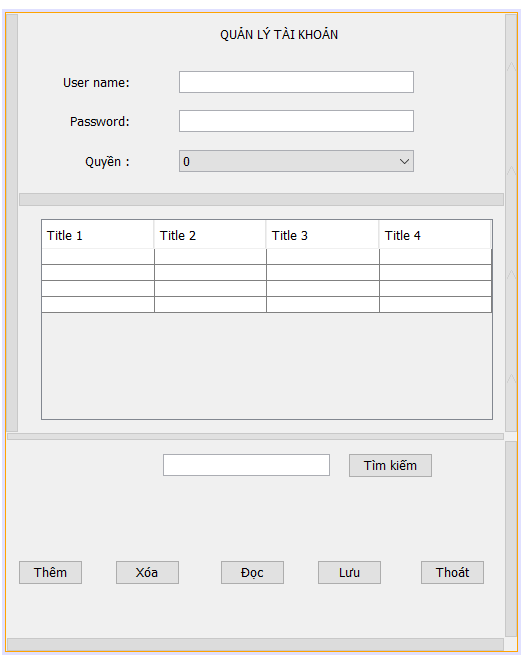
e/ Giao diện hướng dẫn luật chơi



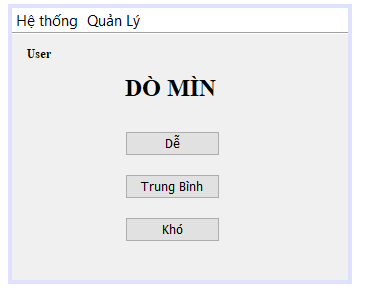
f/ Giao diện trang chủ dành cho người chơi:



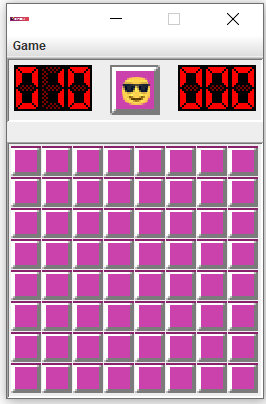
j/ Giao diện quản lý tài khoản dành chơ Admin:



h/ Giao diện trang chủ dành cho người chơi :



k/ Giao diện chơi game:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | MSSV | Họ Tên | Phân Công | Tiến độ |
| 1 | 2001203007 | Trương Thành Trung | Word , Hoàn thiện code , đồng hồ đếm giờ | 100% |
| 2 | 2001207013 | Nguyễn Thị Minh Thư | Đăng nhập, phân quyền , trang chủ (admin, user), tìm hiểu Concurrency ( đồng thời ) | 100% |
| 3 | 2001202198 | Lê Huỳnh Phúc | Bảng xếp hạng (Admin, user), tìm hiểu Parallel (song song) | 100% |
| 4 | 2001200561 | Mai Nguyễn Phước Yến | Quản lý tài khoản,Đăng ký tài khoản, Word , tạo CSDL , code | 100% |