

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**KHOA CÔNG NGHỆ THÔNG TIN**

----o0o----



**BÁO CÁO BÀI TẬP LỚN**

**MÔN CÔNG NGHỆ JAVA**

**Đề tài: Game Pikachu cổ điển.**

| **Giảng viên hướng dẫn:** |  |
| --- | --- |
|  | **TS. Vũ Huấn** |
| **Nhóm thực hiện:** | **Nhóm 21** |
|  | **Phạm Thu Quỳnh - 223630708** |
|  | **Nguyễn Phương Anh - 223630668** |
|  |
|  |

**Hà Nội, tháng 5 năm 2024**

|  |
| --- |
|  |

# LỜI CẢM ƠN

Lời đầu tiên cho phép nhóm xin gửi lời cảm ơn chân thành nhất đến giảng viên Vũ Huấn. Trong quá trình học tập môn Công nghệ Java, nhóm em đã được thầy quan tâm và giúp đỡ tận tình, hết lòng truyền đạt và giải đáp những kiến thức bổ ích, giúp chúng em tích lũy thêm được nhiều kiến thức để có cái nhìn hoàn thiện hơn về bộ môn này. Thông qua bài tập lớn, nhóm em muốn trình bày lại những gì đã tìm hiểu trong thời gian qua, áp dụng kiến thức vào sản phẩm thực tế.

Trong quá trình hoàn thành bài tập lớn, nhóm em đã tìm hiểu, nghiên cứu các kiến thức để làm một cách tốt nhất trong khả năng của mình. Mặc dù đã cố gắng để hoàn thành nhưng do thời gian có hạn và thiếu kinh nghiệm nên việc phân tích và thiết kế còn nhiều thiếu sót, chúng em rất mong nhận được nhữn góp ý, bổ sung của thầy để nhóm có thể hoàn thiện sản phẩm tốt hơn nữa.

Chúng em kính chúc thầy có thật nhiều sức khỏe và thành công trên con đường giảng dạy của mình.

Chúng em xin trân trọng cảm ơn!

**MỤC LỤC**

**DANH MỤC HÌNH ẢNH**

# LỜI MỞ ĐẦU

Trong thời đại công nghệ thông tin ngày nay, trò chơi điện tử đã trở thành một phần không thể thiếu trong cuộc sống hàng ngày của chúng ta, đó là một phần quan trọng của văn hóa giải trí. Trong khi đó, ngôn ngữ lập trình Java không chỉ đóng một vai trò quan trọng trong ngành công nghiệp phần mềm mà còn trở thành một công cụ mạnh mẽ trong lĩnh vực phát triển game. Lập trình Java không chỉ giúp giải quyết các vấn đề về số liệu và xử lí văn bản, mà còn cho phép phát triển trò chơi điện tử đa dạng và hấp dẫn.

Lựa chọn đề tài về việc phát triển trò chơi Pikachu bằng ngôn ngữ lập trình Java đến từ sự kết hợp của hai yếu tố chính: sự phổ biến và sức hút của trò chơi Pikachu cùng với tính linh hoạt và hiệu suất của Java. Mục đích của đề tài không chỉ giúp nắm vững kĩ năng và kiến thức về lập trình, mà còn là cơ hội tìm hiểu về lập trình hướng đối tượng và bộ công cụ Swing trong Java. Trọng tâm của đề tài là tạo ra một phiên bản trò chơi Pikachu với giao diện thú vị và tính năng hấp dẫn, từ đó mang lại trải nghiệm giải trí độc đáo cho người chơi.

Đề tài bao gồm cơ sở lí thuyết, thiết kế, xây dựng trò chơi và đi đến kết luận chung.

**CHƯƠNG I: CƠ SỞ LÍ THUYẾT**

1. Tổng quan đề tài

Game Pikachu cổ điển mang đến trải nghiệm với đồ họa đơn giản 2D đơn giản nhưng lôi cuốn, người chơi sẽ được trải nghiệm một trận đấu offline đầy hứng khởi. Trò chơi phù hợp với mọi độ tuổi và hoàn toàn miễn phí. Nhạc nền vui tươi và kịch tính kèm theo cũng là điểm nhấn tạo nên cảm giác thư giãn sau những giờ làm việc căng thẳng.

1. Ngôn ngữ và các ứng dụng liên quan
   1. Ngôn ngữ lập trình Java

\* Giới thiệu về ngôn ngữ lập trình Java

Java là ngôn ngữ lập trình hướng đối tượng, ra đời vào năm 1995 bởi Sun Microsystems và hiện nay thuộc sở hữu của Oracle. Java được thiết kết với triết lý “viết một lần, chạy mọi nơi” (write once, run anywhere), cho phép các ứng dụng Java có thể chạy trên nhiều nền tảng khác nhau mà không cần chỉnh sửa mã nguồn, nhờ vào máy ảo (Java Virtual Machine - JVM).

\* Ưu điểm của Java trong lập trình game

- Đa nền tảng: Java có khả năng chạy trên nhiều hệ điều hành khác nhau nhờ JVM. Điều này giúp mở rộng khả năng tiếp cận người dùng trên các nền tảng khác nhau mà không cần thay đổi mã nguồn.

- Hiệu năng ổn định: Mặc dù không nhanh như một sô snoon ngữ khác như C++, Java cẫn cung cấp hiệu năng đủ tốt cho các game đơn giản như Pikachu. Hơn nữa, JVM với khả năng biên dịch Just-In-Time (JIT) giúp cải thiện tốc độ thực thi của chương trình.

- Hệ thống thư viện phong phú: Java có nhiều thư viện hỗ trợ lập trình game như Swing và AWT cho giao diện đồ họa, hoặc các thư viện bên thứ ba như LWJGL và libGDX cho các tính năng nâng cao. Trong dự án này, các thư viện giúp đơn giản hóa việc xây dựng giao diện và xử lí sự kiện.

- Garbage Collection: Java có hệ thống quản lí bộ nhớ tự động (Garbage Collection), gúp giảm bớt các lỗi liên quan đến quản lí bộ nhớ thủ công, như memory leaks và pointer errors. Điều này giúp lập trình viên tập trung vào phát triể tính năng củ game thay vì phải lo lắng về vấn đề quản lí bộ nhớ.

- Cộng đồng và tài nguyên phong phú: Java có một cộng đồng lập trình viên rộng lớn và rất nhiều tài liệu, tutorial có sẵn trên mạng. Điều này rất có ích khi gặp phải các vấn đề trong quá trình phát triển game, vì dễ dàng tìm thấy sự hỗ trợ và giải pháp từ cộng đồng.

* 1. Phần mềm HeidiSQL

HeidiSQL là một công cụ quản lí cơ sở dữ liệu miễn phí và nguồn mở, giúp người dùng kết nối và quản lí cơ dở dữ liệu một cách dễ dàng và hiệu quả. Được thiết kế với giao diện thân thiện và trực quan, HeidiSQL hỗ trợ nhiều loại cơ sở dữ liệu phổ biến như MySQL, MariaDB, Microsoft SQL Server,... Người dùng có thể thực hiện các thao tác như tạo, chỉnh sửa, xóa bảng, xem và thao tác dữ liệu, cũng như thực hiện các truy vấn SQL. HeidiSQL còn hỗ trợ nhập xuất dữ liệu và kết nối an toàn qua SSH và SSL, làm cho việc quản lí cơ sở dữ liệu trở nên đơn giản và an toàn hơn.

* 1. Phần mềm XAMPP

XAMPP là một gói phần mềm mã nguồn mở miễn phí, bao gồm X, Apache, MySQL, PHP và Perl cung cấp một môi trường phát triển web hoàn chỉnh. Khi sử dụng XAMPP, người dùng có thể dễ dàng cài đặt và quản lí MySQL trên máy tính cá nhân. Để kết nối cơ sở dữ liệu với Java, người dùng có thể sử dụng JDBC (Java Database Connectivity), cho phép tương tác với cơ sở dữ liệu từ môi trường lập trình Java. XAMPP đơn giản hóa việc thiết lập và thử nghiệm môi trường phát triển cho các dự án Java liên quan đến cơ sở dữ liệu.

**CHƯƠNG II: THIẾT KẾ TRÒ CHƠI**

1. Cách chơi

- Game Pikachu cổ điển đem lại cảm giác quen thuộc và hoài niệm. Trò chơi bắt đầu với nhiều hình ảnh pokemon khác nhau và người chơi cần tìm cách để nối các cặp hình giống nhau bằng không quá ba đường thẳng.

- Thời gian trong game là giới hạn, đòi hỏi người chơi phải nhanh chóng ghép các hình để chiến thắng và tiến đến các màn chơi tiếp theo.

- Khi không còn cặp nào có thể ăn nhau, trò chơi sẽ tự động xáo lại vị trí các hình ảnh để tạo ra những nước chơi mới, hoặc người chơi có thể chủ động xáo lại vị trí các hình ảnh. Khi không còn lượt xáo và không còn nước đi, người chơi sẽ thua cuộc. Mỗi màn chơi thắng người chơi sẽ được thưởng thêm một lượt xáo mới.

1. Đồ họa

\* Đối tượng: Trong game, các đối tượng chính là những con Pikachu. Mỗi con đại diện bởi một hình ảnh cụ thể, thể hiện các loài Pikachu khác nhau với màu sắc và hình ảnh đa dạng.



Hình 2.1: Các Pikachu

\* Giao diện

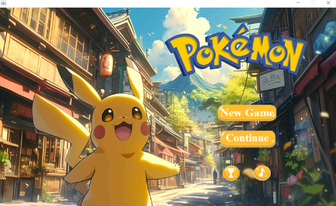
- Màn hình chính: Giao diện mở đầu trò chơi.

+ Bắt đầu trò chơi mới (New Game): Người chơi bắt đầu từ màn chơi đầu tiên và nối các cặp thú cho đến khi hết số cặp.

+ Tiếp tục trò chơi (Continue): Tiếp tục trò chơi được lưu gần đây nhất.

+ Âm nhạc (Music): Người chơi có thể tùy chọn bật hoặc tắt nhạc.

+ Điểm cao (HighScore): Bảng hiển thị danh sách tên người chơi và điểm cao của họ.



Hình 2.2: Màn hình chính

- Giao diện chơi game: Giao diện khi vào trận đấu.

+ Nhãn cấp độ: Hiển thị cấp độ hiện tại ở góc trái trên cùng.

+ Nút lượt đổi: Đặt ở phía trên, cho phép người chơi sắp xếp lại các Pikachu khi cần thiết.

+ Nhãn thời gian: Đặt ở phía trung tâm trên, hiển thị thời gian còn lại của trận game.

+ Nhãn điểm số: Đặt ở phía trên, hiển thị điểm của người chơi.

+ Nút x: Đặt ở phía góc phải trên cùng, cho phép người chơi tạm dừng game.

+ Các nút Pikachu: hình ảnh các con Pikachu sắp xếp trên lưới, người chơi chọn và ghép đôi chúng.



Hình 2.3: Giao diện chơi game

- Tạm dừng (Pause): Giao diện tạm dừng trò chơi

+ Nút Home: Cho phép người chơi quay trở lại màn hình chính. Khi người chơi ấn nút này, game sẽ tự động lưu lại màn chơi hiện tại.

+ Nút Replay: Chơi lại từ đầu.

+ Nút Resume: Tiếp tục chơi màn chơi hiện tại.



Hình 2.4: Giao diện tạm dừng

- Kết thúc trò chơi (Pass): Giao diện khi người chơi hoàn thành tất cả cấp độ nhưng không có điểm cao.

+ Nút Home: Quay về màn hình chính.

+ Nút Replay: Chơi lại ván mới từ đầu.

+ Nhãn điểm, điểm thưởng và tổng điểm: hiển thị thông tin điểm, điểm thưởng và tổng điểm của người chơi.

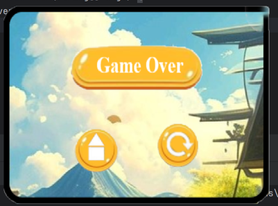


Hình 2.5: Kết thúc trò chơi

- Thua cuộc (GameOver): Giao diện khi người chơi hết thời gian hoặc hết lượt đổi.

+ Nút Home: Quay về màn hình chính.

+ Nút Replay: Chơi lại ván mới từ đầu.



Hình 2.6: Thua cuộc

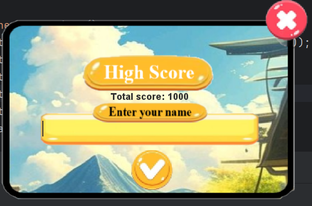
- Ghi danh điểm cao (EnterName): Giao diện khi người chơi đạt điểm cao.

+ Nhãn tổng điểm: Hiển thị tổng điểm của người chơi.

+ Ô nhập tên: Người chơi điền tên.

+ Nút check: Kiểm tra liệu tên của người chơi có hợp lệ không.

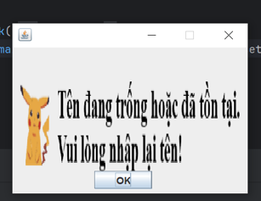
+ Nút x: Khi người chơi ấn vào nút này sẽ quay về màn hình chính và không được lưu điểm cao.



Hình 2.7: Điểm cao

- Báo lỗi (Error): Khi người chơi để trống tên hoặc tên đã tồn tại trong danh sách điểm cao, thông báo này sẽ hiện ra.

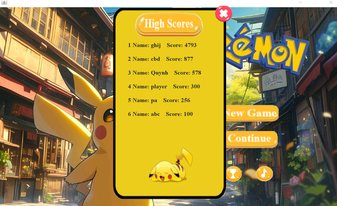
+ Nút OK: Quay lại giao diện điền tên.



Hình 2.8: Báo lỗi

- Danh sách điểm cao (HighScore): Hiển thị danh sách những người chơi đạt điểm cao và điểm số của họ.

+ Nút x: Quay về màn hình chính.



Hình 2.9: Danh sách điểm cao

1. Các cấp độ

- Game bao gồm ba cấp độ chơi. Cấp độ một dễ nhất với các hình đứng im. Ở cấp độ hai, các hình sẽ dịch chuyển xuống dưới sau khi người chơi cho các Pikachu ăn nhau và ở cấp ba là sang bên phải.

- Sau khi kết thúc một màn chơi, người chơi sẽ tự động được chuyển sang màn chơi kế tiếp cho đến khi người chơi thua cuộc hoặc hết màn chơi.

1. Âm thanh: Playing BMG: Are you lost || Park Bird

**CHƯƠNG III: XÂY DỰNG TRÒ CHƠI**

1. Kĩ thuật lập trình

Game được phát triển sử dụng kĩ thuật lập trình hướng đối tượng (OOP) nhằm tạo ra một cấu trúc code linh hoạt, dễ bảo trì và mở rộng. Các điểm chính của việc sử dụng OOP trong dự án:

- Đối tượng và lớp:

+ Mỗi thành phần trong trò chơi được biểu diễn bằng một đối tượng: các nút bấm, hình ảnh, điểm số,...

+ Mỗi đối tượng nằm trong các lớp riêng biệt giúp quản lí và tương tác dễ dàng.

- Kế thừa: Được dùng để tái sử dụng mã nguồn và tạo ra các đối tượng mới từ các lớp cũ, chẳng hạn như việc tạo ra giao diện cho các cửa sổ mới.

- Đa hình: Cho phép một phương thức có thể có hoạt động khác tùy thuộc vào loại đối tượng mà nó được gọi. Ví dụ thông qua triển khai phương thức actionPerformed mà ở nhiều lớp khác nhau sẽ xử lí sự kiện riêng tùy vào mục đích.

- Đóng gói và ứng dụng: Giúp che giấu thông tin và cung cấp quyền kiểm soát truy cập. Sử dụng public, private và protected để quản lí quyền truy cập đến thành phần của các lớp.

- Trừu tượng: Sử dụng các lớp và phương thức để mô hình hóa các đối tượng và hành vi một cách trừu tượng. Ví dụ như sử dụng các lớp và giao diện để định nghĩa hành vi chung mà các đối tượng cần thực hiện, mà không cần biết về cách cụ thể mà các hành vi đó được triển khai.

1. Cách tổ chức

Game được triển khai theo mô hình MVC (Model - View - Controller ) với ba package chính là model, view và controller. Mô hình MVC là một mô hình thiết phổ biến, được chia thành ba phần chính: Model, View, Controller. Điều này giúp cho việc phát triển game trở nên dễ dàng hơn.

- Model: Chứa các lớp đại diện cho dữ liệu và logic. Gồm các lớp GameTimer, IOFile, Matrix, Player, Update và một file để đọc và lưu trò chơi.

- View: Chứa các lớp là giao diện của trò chơi, chịu trách nhiệm hiển thị thông tin và tương tác với người chơi. Các lớp trong package gồm MenuView, GamePlayView, PauseView, PassView, GameOverView, EnterName, ErrorView, HighScoreView, BackGroundMusic, JpanelBackGround và Pikachu.

- Controller: Chứa các lớp xử lí logic và sự kiện, kết nối giữa Model và View. Controller trong Pikachu xử lí các sự kiện từ người chơi, kiểm tra và cập nhật trạng thái game, cũng như điều khiển luồng trò chơi. Package chứa lớp GamePlayController.

Ngoài ba package chính còn các package khác như:

- DAO: Package để tương tác với cơ sở dữ liệu, chứa interface DataAccessObject và lớp PlayerDAO.

- Database: Chứa lớp JDBCUtil để kết nối với cơ sở dữ liệu.

- Library: Chứa các thư viện để kết nối với cơ sở dữ liệu.

- Resources: Chứa các hình ảnh và âm thanh dùng cho giao diện.

- Util: Chứa lớp tiện ích, dễ dàng sử dụng và tái sử dụng trong dự án.

- Test: Chạy chương trình.

1. Thuật toán cốt lõi

3.1. Random và định hình ma trận

\* randomMatrix(): Phương thức này tạo ra ma trận Pikachu ngẫu nhiên và sau đó điều chỉnh để mỗi loại Pikachu xuất hiện với số lượng chẵn.

- Tạo ma trận và khởi tạo giá trị:

| Public void randomMatrix(){  this.matrix = new int[row]col];  this.countMap = new HashMap<>();  for(int i = 0; i < row; i++){  for(int j = 0; j < col; j++){  matrix[i][j] = CONST\_VALUE;  }  } |
| --- |

- Gán giá trị ngẫu nhiên cho ma trận:

| Random random = new Random();  for(int i = 1; i <= row-2; i++) {  for(int j = 1; j <= col - 2; j++) {  matrix[i][j] = random.nextInt(PIKACHU\_NUMBER) + 1;  countMap.put(matrix[i][j], countMap.getOrDefault(matrix[i][j], 0) + 1 );  }  } |
| --- |

\* Định hình lại ma trận: Đảm bảo số lượng Pikachu xuất hiện là chẵn

| for(int pikachu : countMap.keySet()) {  int count = countMap.get(pikachu);  if (count % 2 != 0) {  int newpikachu = 0;  boolean check = false;  while(!check) {  newPikachu = random.nextInt(PIKACHU\_NUMBER) + 1;  cnt++;  if(newPikachu != pikachu) {  if (countMap.containsKey(newPikachu)) {  if(countMap.get(newPikachu) %2 !=0) {  System.out.println(pikachu + “ “ + count + “ “ + newPikachu + “ “ + countMap.get(newPikachu));  countMap.put(newPikachu, countmap.get(newPikachu) + 1);  check = true;  }  }  }  }  outerloop;  for(int i = 1; i <= row-1; i++) {  for(int j = 1;j <= col-1; j++) {  if(matrix[i][j] == pikachu){  matrix[i][j] = newPikachu;  countMap.put(pikachu, countMap.get(pikachu) - 1);  break outerloop;  }  }  }  }  } |
| --- |

3.2. Xác định hai Pikachu ăn nhau

Xác định các đường mà Pikachu ăn nhau, tổng quan là 6 trường hợp.

\* checkLineX(int y1, int y2, int x): Kiểm tra xem có đường thẳng nằm ngang (theo trục x) giữa 2 điểm không.

| private boolean checkLineX(int y1, int y2, int x) {  int min = Math.min(y1, y2);  int max = Math.max(y1, y2);  for (int y = min + 1; y < max; y++) {  if (matrix[x][y] != CONST\_VALUE)  return false;  }  return true;  } |
| --- |

\* checkLineY( int x1, int x2, int y): Kiểm tra xem có đường thẳng nằm dọc (theo trục y) giữa 2 điểm không.

| private boolean checkLineY(int x1, int x2, int y) {  int min = Math.min(x1, x2);  int max = Math.max(x1, x2);  for (int x = min+1; x < max; x++) {  if (matrix[x][y] != CONST\_VALUE)  return false;  }  return true; } |
| --- |

\* checkRectX( Pikachu one, Pikachu two): Kiểm tra đường đi hình chữ nhật với cạnh nằm ngang giữa hai Pikachu.

| private boolean checkRectX(Pikachu one, Pikachu two) {  Pikachu pMinY = one;  Pikachu pMaxY = two;  if (one.getyPoint() > two.getyPoint()) {  pMaxY = one;  pMinY = two;  }  for (int y = pMinY.getyPoint() ; y <= pMaxY.getyPoint(); y++) {  if(y > pMinY.getyPoint() && matrix[pMinY.getxPoint()][y] != CONST\_VALUE) {  return false;  }  if((matrix[pMaxY.getxPoint()][y] == CONST\_VALUE || y == pMaxY.getyPoint())  && checkLineY(pMinY.getxPoint(), pMaxY.getxPoint(), y)  && checkLineX(y, pMaxY.getyPoint(), pMaxY.getxPoint())){  return true;  }  }  return false;  } |
| --- |

\* checkRectY( Pikachu one, Pikachu two): Kiểm tra đường đi hình chữ nhật với cạnh dọc giữa hai Pikachu.

| private boolean checkRectY(Pikachu one, Pikachu two) {  Pikachu pMinX = one;  Pikachu pMaxX = two;  if (one.getxPoint() > two.getxPoint()) {  pMaxX = one;  pMinX = two;  }  for (int x = pMinX.getxPoint() ; x <= pMaxX.getxPoint(); x++) {  if(x > pMinX.getxPoint() && matrix[x][pMinX.getyPoint()] != CONST\_VALUE)  return false;  if((matrix[x][pMaxX.getyPoint()] == CONST\_VALUE || x == pMaxX.getxPoint())  && checkLineX(pMinX.getyPoint(), pMaxX.getyPoint(), x)  && checkLineY(x, pMaxX.getxPoint(), pMaxX.getyPoint())) {  return true;  }  }  return false;  } |
| --- |

\* checkMoreLineX (Pikachu one, Pikachu two, int type): Kiểm tra đường đi hình chữ Z hoặc U với cạnh ngang giữa hai Pikachu.

| private boolean checkMoreLineX(Pikachu one, Pikachu two, int type) {  Pikachu pMinY = one;  Pikachu pMaxY = two;  if (one.getyPoint() > two.getyPoint()) {  pMaxY = one;  pMinY = two;  }  int y = pMaxY.getyPoint() + type;  int row = pMinY.getxPoint();  int colFinish = pMaxY.getyPoint();  if (type == -1) {  y = pMinY.getyPoint() + type;  row = pMaxY.getxPoint();  colFinish = pMinY.getyPoint();  }  if ((matrix[row][colFinish] == CONST\_VALUE || pMinY.getyPoint() == pMaxY.getyPoint())  && checkLineX(pMinY.getyPoint(), pMaxY.getyPoint(), row)) {  while (this.matrix[pMinY.getxPoint()][y] == CONST\_VALUE  && this.matrix[pMaxY.getxPoint()][y] == CONST\_VALUE) {  if (checkLineY(pMinY.getxPoint(), pMaxY.getxPoint(), y)) {  return true;  }  y += type;  }  }  return false;  } |
| --- |

\* checkMoreLineY( Pikachu one, Pikachu two, int type): Kiểm tra đường đi hình chữ Z hoặc U với cạnh dọc giữa hai Pikachu.

| private boolean checkMoreLineY(Pikachu one, Pikachu two, int type) {  Pikachu pMinX = one;  Pikachu pMaxX = two;  if (one.getxPoint() > two.getxPoint()) {  pMaxX = one;  pMinX = two;  }  int x = pMaxX.getxPoint() + type;  int col = pMinX.getyPoint();  int rowFinish = pMaxX.getxPoint();  if (type == -1) {  x = pMinX.getxPoint() + type;  col = pMaxX.getyPoint();  rowFinish = pMinX.getxPoint();  }  if ((this.matrix[rowFinish][col] == CONST\_VALUE || pMinX.getxPoint() == pMaxX.getxPoint())  && checkLineY(pMinX.getxPoint(), pMaxX.getxPoint(), col)) {  while (this.matrix[x][pMinX.getyPoint()] == CONST\_VALUE  && this.matrix[x][pMaxX.getyPoint()] == CONST\_VALUE) {  if (checkLineX(pMinX.getyPoint(), pMaxX.getyPoint(), x)){  return true;  }  x += type;  }  }  return false;  } |
| --- |

3.3. Điều kiện để tiếp tục chơi

Phương thức canPlay(): Dùng để kiểm tra xem trong trò chơi còn nước đi nào hợp lệ không. Cụ thể, nó kiểm tra xem còn cặp Pikachu nào có thể kết nối với nhau hay không. Nếu tìm thấy, hàm trả về true, ngược lại trả về false.

| public boolean canPlay() {  for (int i = 1; i <= row - 2; i++) {  for (int j = 1; j <= col - 2; j++) {  if (matrix[i][j] != CONST\_VALUE) {  for (int m = 1; m <= row - 2; m++) {  for (int n = 1; n <= col - 2; n++) {  if ((m != i || n != j) && matrix[m][n] != CONST\_VALUE && matrix[m][n] == matrix[i][j]) {  if (algorithm(new Pikachu(i, j), new Pikachu(m, n))) {  Util.Utils.debug(getClass(), "i:" + i + " j:" + j + " ->" + "m:" + m + " n:" + n);  System.out.println("---");  System.out.println();  return true;  }  }  }  }  }  }  }  return false;  } |
| --- |

1. Liên kết cơ sở dữ liệu

\* Cơ sở dữ liệu: Gồm một bảng duy nhất chứa hai thuộc tính là PlayerName (tên người chơi) và Score (điểm số).

\* Kết nối: Khi người chơi đạt điểm cao sẽ được lưu thông tin tên và điểm trong database, đồng thời được xuất hiện trong bảng điểm cao ở giao diện HighScoreView. Sử dụng phần mềm HeidiSQL để kết nối và quản lí cơ sở dữ liệu một cách hiệu quả. Môi trường lập trình được kết nối tới cơ sở dữ liệu thông qua lớp JDBCUtil.

\* Lớp JDBCUtil

- Mô tả: Dùng để kết nối tới cơ sở dữ liệu, đóng kết nối và in ra một số thông tin về cơ sở dữ liệu. Lớp giúp quản lí kết nối và thực hiện thao tác cơ bản với cơ sở dữ liệu trong môi trường lập trình Java.

- Các phương thức:

+ getConnection(): Thiết lập kết nối đến cơ sở dữ liệu.

+ closeConnection(): Đóng kết nối với cơ sở dữ liệu, việc này giúp giải phóng tài nguyên và làm sạch các kết nối không cần thiết.

+ printInfor(): In ra thông tin về cơ sở dữ mà kết nối đang kết nối tới.

| public class JDBCUtil {  public static Connection getConnection() {  Connection c = null;  try{  DriverManager.registerDriver(new com.mysql.cj.jdbc.Driver());  String url = "jdbc:mysql://localhost:3306/Player";  String user = "root";  String password = "";  c = DriverManager.getConnection(url,user,password);  } catch (SQLException e){ e.printStackTrace(); }  return c;  }  public static void closeConnection(Connection c){  try{  if(c != null){ c.close(); }  } catch (Exception e){ e.printStackTrace(); }  }  public static void printInfor(Connection c){  try{  if (c != null){  DatabaseMetaData data = c.getMetaData();  System.out.println(data.getDatabaseProductName());  }  } catch (SQLException e){ e.printStackTrace(); }  }  } |
| --- |

1. Xây dựng các class
   1. Class trong package model

5.1.1. Lớp GameTimer

- Mô tả: Lớp được sử dụng để quản lí thời gian đếm ngược cho trò chơi, cập nhật mỗi giây và hiển thị thời gian còn lại dưới dạng phút và giây trên giao diện thông qua một JLabel. Khi thời gian đếm ngược kết thúc, lớp này sẽ thông báo qua một TimerListener.

- Các thuộc tính:

+ private Timer timer;

+ private JLabel counterLabel;

+ private int second = SECOND;

+ private int minute = MINUTE;

+ private TimerListener timerListener;

- Các phương thức:

+ GameTimer(): Khởi tạo Timer với chu kì 1 giây. Mỗi lần kích hoạt, cập nhật nhãn thời gian và giảm thời gian. Nếu hết giờ, dừng bộ đếm và thông báo qua TimerListener.

+ setTimerListener(TimerListener listener): Thiết lập lắng nghe sự kiện khi thời gian kết thúc.

+ start(): Bắt đầu đếm ngược thời gian với giá trị khởi tạo từ MINUTE và SECOND.

+ continueTimer(int minute, int second): Tiếp tục đếm ngược thời gian từ giá trị giây và phút đã cho.

+ stop(): Dừng bộ đếm thời gian.

+ notifyTimeUp(): Thông báo thời gian đã hết thông qua timerListener.

+ setCounterLabel(JLabel counterLabel): Nhãn hiển thị thời gian.

+ getSecond(): Lấy số giây còn lại.

+getMinute(): Lấy số phút còn lại.

+ Interface TimerListener{onTimeUp()}: Định nghĩa phương

thức để xử lý sự kiện khi thời gian kết thúc. Lớp bên ngoài có thể triển khai interface này để thực hiện hành động khi thời gian kết thúc.

5.1.2. Lớp IOFile

- Mô tả: Lớp này được sử dụng để lưu trạng thái hiện tại của trò chơi vào một tệp và đọc trạng thái từ tệp này khi người chơi sử dụng nút Continue. Cho phép trò chơi có thể tiếp tục từ màn đã dừng lại.

- Các thuộc tính:

+ private int level;

+ private int turns;

+ private int score;

+ private int minute;

+ private int second;

+ private int[][] table;

+ boolean state = true;

- Các phương thức:

+ IOFile(): Hàm khởi tạo mặc định.

+ printFile(GameTimer gameTimer, Matrix matrix): Lưu trạng thái hiện tại của trò chơi vào tệp, bao gồm các thông tin như cấp độ, số lượt xáo, điểm số, thời gian còn lại và trạng thái ma trận trò chơi.

+ readFile(Matrix matrix): Đọc trạng thái trò chơi từ tệp đã lưu và khôi phục các thông số trò chơi như màn chơi, số lượt xáo, điểm số thời gian còn lại và trạng thái ma trận trò chơi.

+ getState(): trả về trạng thái hiện tại của trò chơi (có thể tiếp tục hay không).

+ getLevel(): Trả về cấp độ hiện tại của trò chơi.

+ getTurns(): Trả về số lượt đổi còn lại.

+ getScore(): Trả về điểm số hiện tại của người chơi.

+ getMinute(): Trả về số phút còn lại trên đồng hồ.

+ getSecond(): Trả về số giây còn lại trên đồng hồ.

+ getTable(): Trả về ma trận lưu trữ trạng thái trò chơi.

5.1.3. Lớp Matrix

- Mô tả: Lớp quản lí một ma trận 2D các số nguyên, mô phỏng

một bảng trò chơi Pikachu. Lớp này gồm các phương thức để khởi tạo, thao tác và xác minh ma trận, tạo một ma trận ngẫu nhiên, điều chỉnh nó để đảm bảo sự phân phối đúng đắn của các phần từ và kiểm tra hai phần tử có ăn nhau theo quy tắc trò chơi không.

- Các thuộc tính:

+ private int[][] matrix;

+ private int row;

+ private int col;

+ private int value; private static final int CONST\_VALUE = 0;

+ private HashMap<Integer, Integer> countMap;

- Các phương thức:

+ Matrix(): Hàm khởi tạo mặc định.

+ Các phương thức Getters và Setters: Truy cập và thay đổi cho matrix, row, col và countMap, đặt gíá trị cho các ô trong ma trận và trả về giá trị của ô ma trận tại vị trí được chỉ định.

+ randomMatrix(): Tạo ra một ma trận ngẫu nhiên đảm bảo mỗi giá trị xuất hiện một số lần chẵn.

+ Các phương thức checkLineX, checkLineY, checkRectX, checkRectY, checkMoreLineX và checkMoreLineY: Được sử dụng để kiểm tra xem có thể nối hai ô Pikachu theo các quy tắc của trò chơi không.

+ algorithm(): Kiểm tra hai ô Pikachu được chỉ định có thể kết nối với nhau được hay không.

+ canPlay(): Kiểm tra xem còn cặp nào ăn nhau để chơi tiếp trong game.

+ shuffleMatrix(): Trộn các ô có giá trị trong ma trận để tạo ra một trạng thái ma trận mới.

+ level2() và level3(): Tạo ra hiệu ứng di chuyển các ô Pikachu sau khi ăn nhau.

5.1.4. Lớp Player

- Mô tả: Là lớp đại diện cho người chơi trong trò chơi.

- Các thuộc tính:

+ private String userName;

+ private int score;

- Các phương thức:

+ Player(): Hàm khởi tạo không tham số.

+ Player(String userName, int score): Khởi tạo đối tượng Player với tên người chơi và điểm số được chỉ định.

+ toString(): Hàm ghi đè, cung cấp một biểu diễn chuỗi của đối tượng Player, bao gồm tên người chơi và điểm số của họ.

+ getUserName(), getScore(): Các phương thức trả về tên và điểm số của người chơi.

+setUsername(String userName), setScore(int score): Các phương thức thiết lập tên và điểm số của người chơi.

5.1.5. Lớp Update

- Mô tả: Lớp này quản lí và cập nhật thông tin về điểm số, cấp độ và lượt đổi.

- Các thuộc tính:

+ private static PlayerDAO playerDAO;

+ private static int score = 0;

+ private static int level = 1;

+ private static int turns = 10;

- Các phương thức:

+ getScore(), getLevel(), getTurns(): Các phương thức trả về điểm số, cấp độ hiện tại và số lượt đổi còn lại của người chơi.

+ setScore(int score), setLevel(int level), setTurns(int turns): Các phương thức thiết lập số điểm, cấp độ, lượt đổi của người chơi.

+ increaseScore(), increaseLevel(), increaseTurns(): Tăng đ iểm, cấp độ, số lượt đổi của người chơi.

+ decreaseTurns(): Giảm số lượt đổi còn lại đi 1.

5.1.6. Lớp BackGroundMusic

- Mô tả: Lớp này là một luồng trong Java, chịu trách nhiệm cho việc phát nhạc nền trong trò chơi.

-Hàm khởi tạo: BackGroundMusic(String path): Nhận đường dẫn tới file âm thanh và tạo ra đối tượng Clip để phát nhạc.

- Các phương thức:

+ run(): Phương thức chạy trong luồng, được gọi khi luồng bắt đầu. Nó kiểm tra xem clip có tồn tại và không đang phát, sau đó phát nhạc nền trong chế độ lặp vô hạn.

+ stopMusic(): Dừng việc phát nhạc, đóng clip và đặt isPlaying về false.

+pause(): Tạm dừng việc phát nhạc bằng cách dừng clip. +isPlaying(): Trả về trạng thái của việc phát nhạc ( true nếu đang phát, false nếu đang không phát).

5.1.7. Lớp Pikachu

- Mô tả: Lớp Pikachu kế thừa lớp JButton để hiển thị ô Pikachu trên ma trận của trò chơi.

- Các thuộc tính:

+ private int xPoint;

+ private int yPoint;

- Các phương thức

+ Pikachu(int xPoint, int yPoint): khởi tạo một ô Pikachu với các tọa độ x và y được chỉ định.

+ Pikachu(): Khởi tạo một ô Pkachu mặc định.

+getters và setters: Truy cập và cập nhật tọa độ x,y của ô Pikachu.

5.1.8. Lớp JpanelBackGround

- Mô tả: Là lớp kế thừa JPanel, dùng để hiển thị một hình nền tùy chỉnh. Lớp này có thể sử dụng để đặt hình nề cho bất kì giao diện nào cần có hình nền.

- Thuộc tính: protected Image backGroundImage = null;

- Các phương thức:

+ JpanelBackGround(): Gọi phương thức khởi tạo của lớp cha và không làm gì cả.

+ paintComponent(Graphics g): Phương thức ghi đè. Gọi super.paintComponent(g) để đảm bảo tất cả các thao tác vẽ mặc định của JPanel được thực hiện. Sau đó nó vẽ hình nền *backGroundImage* trên toàn bộ kích thước của panel bằng cách sử dụng phương thức getHeight() và getWidth().

+setBackgroundImage(String imagePath): Phương thức này cho phép thay đổi hình nền của một panel theo ý muốn. Sử dụng getClass().getResource(imagePath) để tìm đường dẫn của ảnh trong dự án, đảm bảo hình ảnh có thể được tải từ thư mục Resources. Lớp này rất dễ sử dụng, linh hoạt và có thể tái sử dụng nhiều lần cho nhiều lớp khác nhau.

* 1. Các class trong view

5.2.1. Lớp MenuView

- Mô tả: Lớp này là một lớp mở rộng từ JFrame. Lớp này đại diện cho giao diện mở đầu, chứa các nút của trò chơi như: New Game, Continue, Highscore và Music. Người chơi có thể tùy chọn chơi ván mới hoặc tiếp tục ván cũ, bật/tắt nhạc, xem danh sách điểm cao của trò chơi.

- Các thuộc tính:

+ public static final int WIDTH = 1050;

+ public static final int HEIGHT = 650;

+ private JButton jbtNewGame;

+ private JButton jbtContinue;

+ private JButton jbtHighScore;

+ private JButton jbtMusic;

+ private JButton jbtMute;

- Các phương thức:

+ MenuView(): Gọi phương thức init() và đặt cửa sổ không hiển thị.

+ init(): Phương thức tạo và cấu hình cửa sổ giao diện, khởi tạo nút và gán hình ảnh cho các nút, tạo một Panel và thêm hình nền, thêm các nút cho Panel.

+ getJbtNewGame(), getJbtContinue(), getJbtHighScore(), getJbtMusic(), getMute(): Phương thức trả về các nút New Game, Continue, HighScore, Music, Mute.

5.2.2. Lớp HighScoreView

- Mô tả: Lớp này là một lớp mở rộng từ JFrame. Đại diện cho giao diện hiển thị danh sách điểm cao trong trò chơi. Người chơi được xem điểm cao và có thể nhấn nút x để quay lại màn hình chính.

- Các thuộc tính:

+ private JButton jbtX;

+ private JTextArea jtfHighScore;

+ private final int width = 450;

+ private final int height = 600;

- Các phương thức:

+ HighScoreView(): Gọi phương thức init() và đặt cửa sổ không hiển thị.

+ init(): Khởi tạo và cấu hình các thành phần của giao diện danh sách điểm cao. Thiết lập kích thước, vị trí cho nút X, tạo vùng hiển thị danh sách điểm cao và cấu hình font chữ.

+ getJbtX(): Trả về đối tượng JButton của nút X (đóng cửa sổ).

+ setTextHighScore(String s): Đặt văn bản hiển thị danh sách điểm cao thành chuỗi được cung cấp.

5.2.3. Lớp GamePlayView

- Mô tả: Lớp này là một lớp mở rộng từ JFrame. Là giao diện hiển thị trò chơi Pikachu khi người chơi vào màn chơi. Người chơi tương tác với các nút Pikachu, biết được màn chơi hiện tại, số lượt đổi còn lại, thời gian còn lại và điểm số của họ. Ngoài ra họ có thể lựa chọn thoát ra với nút x.

-Các thuộc tính:

+ private Update update;

+ private final Matrix logic;

+ private JpanelBackGround panelMain;

+ private JLabel jlbLevel;

+ private JLabel jlbNumLevel;

+ private JLabel jlbTurns;

+ private JLabel jlbNumTurns;

+ private JLabel jlbTime;

+ private JLabel jlbTimeRemaining;

+ private JLabel jlbScore;

+ private JLabel jlbNumScore;

+ private Pikachu[][] pikachus;

+ private JButton jbtX;

+ private JPanel pikachuPanel;

+ private int row = MAP\_ROW + 2;

+ private int col = MAP\_COL + 2;

+ private Pikachu one;

+ private Pikachu two;

- Các phương thức:

+ GamePlayView(): Sử dụng Constructor của Matrix để tạo một đối tượng logic cho trò chơi. Gọi phương thức init() và đặt cửa sổ không hiển thị.

+ init(): Phương thức tạo và cấu hình cửa sổ giao diện, thiết lập các nhãn và nút để hiển thị thông tin cấp độ, số lượt xáo, thời gian còn lại, điểm số. Tạo một Panel để chứa các ô Pikachu và gán hình ảnh cho các nút.

+ makeMap(): Tạo ra các ô Pikachu bằng cách sử dụng một mảng hai chiều. Thêm các ô Pikachu vào panel chứa ma trận.

+ updateMap(int [][] matrix): Cập nhật trạng thái của ma trận dựa trên ma trận đầu vào.

+ Các phương thức getters và setters: Được sử dụng để truy cập và cập nhật các thuộc tính từ bên ngoài.

+ getIcon(index): Trả về biểu tượng Pikachu tương ứng với chỉ số.

+ setBackGround(String imagePath): Tạo hình nền với đường dẫn được chỉ định.

+ creatPikachu(int x, int y): Tạo một ô Pikachu tại vị trí x, y trên ma trận.

+ createSelectedIcon(ImageIcon icon): Tạo một hình ảnh Pikachu đã chọn để hiển thị.

5.2.4. Lớp PauseView

- Mô tả: Lớp này là một lớp mở rộng từ JFrame. Đại diện cho giao diện tạm dừng trò chơi. Ở giao diện này, các thông số được giữ nguyên và thời gian tạm dừng. Người chơi có thể quay lại màn hình chính, chơi lại ván mới hoặc tiếp tục ván chơi hiện tại.

- Các thuộc tính:

+ private JButton jbtHome;

+ private JButton jbtReplay;

+ private JButton jbtResume;

- Các phương thức:

+ PauseView(): Khởi tạo giao diện ở trạng thái tạm dừng, gọi phương thức init() và đặt cửa sổ ở trạng thái không hiển thị.

+ init(): Khởi tạo và cấu hình các thành phần của giao diện tạm dừng. Thiết lập kích cỡ của cửa sổ, thoát ứng dụng khi đóng cửa sổ, căn cửa sổ ra giữa màn hình, ô hiệu hóa khả năng thay đổi kích thước của cửa sổ, bỏ khung cửa sổ mặc định và cho màu nền cửa sổ trong suốt Tạo các nút Home, Replay và Resume và thiết lập hình ảnh, vị trí cho từng nút. Tạo một Panel, đạt hình nền và cho chứa các thành phần của giao diện.

+ getJbtHome(), getJbtReplay(), getJbtResume(): Trả về đối tượng JButton của nút quay về màn hình chính, chơi lại và tiếp tục chơi.

5.2.5. Lớp PassView

- Mô tả: Lớp này là một lớp mở rộng từ JFrame. Đại diện cho giao diện khi người chơi hoàn thành tất cả các màn chơi ( Không đạt được điểm cao trong trò chơi), hiển thị điểm số và cung cấp tùy chọn chơi lại hoặc quay về màn hình chính.

- Các thuộc tính:

+ private JButton jbtHome;

+ private JButton jbtReplay;

+ private JLabel jlbScore;

+ private JLabel jlbBonusScore;

+ private JLabel jlbTotalScore;

- Các phương thức:

+ Khởi tạo một đối tượng PassView, gọi phương thức init() để thiết lập các thành phần giao diện, đặt hiển thị là false.

+ init(): Khởi tạo và cấu hình các thành phần của giao diện Pass. Thiết lập kích cỡ của cửa sổ, thoát ứng dụng khi đóng cửa sổ, căn cửa sổ ra giữa màn hình, ô hiệu hóa khả năng thay đổi kích thước của cửa sổ, bỏ khung cửa sổ mặc định và cho màu nền cửa sổ trong suốt Tạo các nút Home, Replay và thiết lập hình ảnh, vị trí cho từng nút. Tạo một Panel, đặt hình nền và chứa các thành phần của giao diện.

+ getJbtHome(), getJbtReplay(): Trả về đối tượng nút quay về màn hình chính và nút chơi lại.

+ setJlbScore(int score): Cập nhật và hiển thị điểm số của màn chơi.

+ setJlbBonusScore(int bonusScore): Cập nhật và hiển thị điểm thưởng của màn chơi.

+ setJlbtotalScore(int totalScore): Cập nhật và hiển thị tổng điểm của màn chơi.

5.2.6. Lớp GameOverView

- Mô tả: Lớp này là một lớp mở rộng từ JFrame. Dùng để hiển thị giao diện kết thúc trò chơi khi người chơi thua cuộc. Người chơi có thể lựa chọn chơi lại màn mới hoặc về màn hình chính.

- Các thuộc tính:

+ private JButton jbtReplay;

+ private JButton jbtHome;

- Các phương thức:

+ GameOverView(): Gọi phương thức init() để khởi tạo giao diện. Không hiển thị cửa sổ ngay lập tức khi đối tượng được tạo ra.

+ init(): Thiết lập kích cỡ của cửa sổ, đảm bảo ứng dụng sẽ không thoát ra khi đóng cửa sổ, căn cửa sổ ra giữa màn hình, ô hiệu hóa khả năng thay đổi kích thước của cửa sổ, bỏ khung cửa sổ mặc định và cho màu nền cửa sổ trong suốt. Tạo các nút và các hình ảnh cho nút. Tạo một panel để chứa ảnh nền và các nút. Thiết lập các thuộc tính và vị trí của các thành phần.

+ getJbtReplay(), getJbtHome(): Trả về các nút quay lại màn hình chính và chơi lại.

5.2.7. EnterName

- Mô tả: Lớp này là một lớp mở rộng từ JFrame. Dùng để hiển thị giao diện cho phép người chơi nhập tên của mình sau khi đạt được điểm cao trong trò chơi. Lớp này gồm các thành phần nút xác nhận, nút đóng, ô nhập tên và hiển thị tổng điểm.

- Các thuộc tính:

+ private JButton jbtCheck;

+ private JButton jbtX;

+ private JTextField jtfName;

+ private JLabel jlbTotalScore;

- Các phương thức:

+ EnterName(): Gọi phương thức init() để khởi tạo giao diện. Đặt thuộc tính hiển thị cửa sổ thành false để không hiển thị ngay lập tức khi được tạo ra.

+ init(): Thiết lập kích cỡ của cửa sổ, ứng dụng sẽ thoát ra khi đóng cửa sổ, căn cửa sổ ra giữa màn hình, vô hiệu hóa khả năng thay đổi kích thước của cửa sổ, bỏ khung cửa sổ mặc định và cho màu nền cửa sổ trong suốt.Tạo ra các nút, các nhãn và gán hình ảnh cho các nút, nhãn. Tạo ra một panel để dặt hình nền và chứa các nút.

+ getJbtCheck(), getJbtX(), getJtfName(): Trả về nút check, nút X và ô nhập tên.

+ sẹtlbTotalScore(int totalScore): Thiết lập văn bản cho nhãn tổng điểm.

5.2.8. Lớp ErrorView

- Mô tả: Là một lớp kế thừa từ JFrame, được thiết kế để hiển thị một thông báo lỗi đơn giản với một nút OK cho phép người dùng đóng cửa sổ thông báo.

-Thuộc tính: private JButton ok;

- Các phương thức:

+ ErrorView(): Gọi phương thức init() để khởi tạo giao diện. Đặt thuộc tính hiển thị của sổ thành false để không hiển thị ngay lập tức khi được tạo ra.

+ init(): Thiết lập kích cỡ của cửa sổ, ứng dụng sẽ thoát ra khi đóng cửa sổ, căn cửa sổ ra giữa màn hình, vô hiệu hóa khả năng thay đổi kích thước của cửa sổ, bỏ khung cửa sổ mặc định và cho màu nền cửa sổ trong suốt.Tạo ra nút OK. Tạo ra một panel để đặt hình nền và chứa nút.

+ getOK(): Trả về nút OK.

* 1. Class trong package controller: GamePlayController

\* Mô tả: Là lớp điều khiển chính của trò chơi, đảm nhận việc quản lí giao diện người dùng và các logic cảu trò chơi. Lớp này kết nối giữa các mô hình, giao diện người dùng và các sự kiện để đảm bảo trò chơi hoạt động mượt mà và logic game được thực thi đúng đắn.

\* Các thuộc tính:

- menuView, gamePlayView, gameOverView, highScoreView, pauseView, passView, enterName, errorView: Các đối tượng giao diện của trò chơi.

- private PlayDAO playerDAO: Đối tượng truy xuất dữ liệu của trò chơi.

- private Pikachu one, Pikachu two: Hai đối tượng Pikachu đại diện hai ô Pikachu mà người chơi chọn.

- private static int countClick = 0, coupleDone = 0: Biến đếm số lần Click và số cặp Pikachu đã hoàn thành.

- private ImageIcon defaultIcon: Icon của Pikachu.

- private GameTimer gameTimer: Đối tượng đồng hồ đếm giờ của trò chơi.

- private IOFile file: Đối tượng quản lí đọc và ghi dữ liệu trò chơi.

- private BackGroundMusic music: Đối tượng quản lí nhạc nền của trò chơi.

\* Các phương thức:

- Xử lí sự kiện cho các nút

+ addPikachuListener(): Thêm sự kiện ActionListener cho các ô Pikachu.

+ actionPerformed(ActionEvent e): Xử lí sự kiện khi người chơi click vào một ô Pikachu. Kiểm tra các điều kiện của trò chơi và cập nhật trạng thái trò chơi.

+ addPropertyTursListener(): Thêm sự kiện thay đổi thuộc tính cho số lượt đổi còn lại.

+addHomePauseView(),addReplayPauseView(), addResumePaus eView(): Thêm sự kiện cho các nút Home, Replay và Resume trong giao diện Pause.

+ addNewGame(), addContinue(), addRank(), addMusic(), addMute(), endGame(): Thêm sự kiện cho các nút NewGame, Continue, HighScore, Music, Mute trong giao diện MenuView và xử lí kết thúc trò chơi.

+ x\_EnterName(), check\_EnterName(): Thêm sự kiện cho các nút X và Check trong giao diện EnterName.

+addHomeGameOverView(),addReplayGameOverView(): Thêm sự kiện cho các nút trong GameOverView.

+ ok\_ErrorView(): Thêm sự kiện cho nút OK trong ErrorView.

+ x\_HighScore(): Thêm sự kiện cho nút X trong HighScoreView.

+ updateButtonColor(JButton button): Cập nhật màu của nút Pikachu khi được nhấn vào.

- Các phương thức khác

+ addChangeMap(): Thêm sự kiện thay đổi lại ma trận.

+ bonusScore(): Tính toán điểm thưởng dựa trên thời gian còn lại.

+ totalScore(): Tính toán tổng điểm của người chơi.

+ checkScore(): Kiểm tra điểm số người chơi xem có đủ điều kiển điểm cao không để hiển thị GameOver hoặc EnterName.

+ updateTimer(): Khởi động lại đồng hồ đếm giờ.

+ newGame(): Bắt đầu một trò chơi mới, thiết lập lại các thuộc tính và giao diện.

+ getCoupleDone(): Tính số cặp Pikachu đã hoàn thành trong trò chơi.

* 1. Các class khác

\* Interface DataAccessObject

- Mô tả: Interface định nghĩa một tập hợp các phương thức để thao tác với cơ sở dữ liệu cho các đối tượng.

- Các phương thức:

+ add(T t): Thêm một đối tượng vào cơ sở dữ liệu. Nhận một đối tượng kiểu T làm đối số.

+ delete(T t): Xóa một đối tượng từ cơ sở dữ liệu. Nhận một đối tượng kiểu T làm đối số.

+ selectAll(): Lấy tất cả các đối tượng từ cơ sở dữ liệu.

+ selectById(T t): Chọn một đối tượng từ cơ sở dữ liệu dựa trên một thông tin xác định. Nhận một đối tượng kiểu T làm đối số.

\* Lớp PlayerDAO

- Mô tả: Là lớp được thiết kế để quản lí và tương tác với dữ liệu người chơi trong cơ sở dữ liệu.

- Các phương thức:

+ getInstance(): Trả về một thể hiện duy nhất của lớp PlayerDAO, đảm bảo chỉ có một đối tượng PlayerDAO được sử dụng trong toàn bộ dự án.

+ add(Player player): Thực hiện thêm người mới vào cơ sở dữ liệu với tên và điểm số tương ứng.

+ delete(Player player): Xóa người chơi ra khỏi cơ sở dữ liệu.

+ selectAll(): Truy vấn toàn bộ dữ liệu từ bảng trong cơ sở dữ liệu, dùng để lấy toàn bộ danh sách trong cơ sở dữ liệu.

+ selectById(Player player): Truy vấn một người chơi trong cơ sở dữ liệu thông qua tên người chơi.

+ minScorePlayer(): Trả về người chơi có điểm thấp nhất trong cơ sở dữ liệu.

+ isExists(String playerName): Kiểm tra xem tên của một người chơi đã tồn tại trong cơ sở dữ liệu chưa.

+sortPlayer(): Lấy danh sách người chơi từ cơ sở dữ liệu được sắp xếp theo số điểm giảm dần.

\* Lớp Utils: Là lớp tiện ích, chứa một tập hợp các hằng số, phương thức dùng trong dự án. Ngoài ra còn lưu sẵn đường dẫn của các file ảnh, âm thanh.

**CHƯƠNG 4: KẾT LUẬN**

1. Kết luận tổng quan

Dự án trò chơi Pikachu bằng ngôn ngữ Java đã đạt được một số kết quả và mục đích đề ra, tạo ra một phiên bản Pikachu với giao diện thú vị và tính năng hấp dẫn. Dự án lần này đã đem lại cho chúng em một trải nghiệm học tập và phát triển quý báu. Đây là cơ hội giúp chúng em nắm vững kiến thức và kĩ năng lập trình, đồng thời hiểu rõ hơn tầm quan trọng của sự hợp tác, giao tiếp và phối hợp công việc. Chúng em hy vọng trò chơi sẽ mang lại niềm vui và giải trí cho người chơi, đồng thời đánh dấu một bước mới trong việc phát triển kĩ năng và kinh nghiệm của chúng em.

1. Đánh giá

2.1. Ưu điểm

- Trò chơi đơn giản, dễ chơi: Người chơi chỉ cần nối hai cặp giống nhau bằng không quá ba đường.

- Tính thách thức cao: Mặc dù đơn giản nhưng trò chơi lại mang tính thách thức cao, người chơi cần nối hết các cặp trong thời gian quy định và tránh để hết lượt xáo.

- Linh hoạt: Người chơi có thể tùy chỉnh âm thanh, trò chơi có nhiều màn và độ khó tăng dần mang lại cảm giác chinh phục và mới mẻ cho người chơi.

- Bảng xếp hạng: Người chơi có thể so sánh điểm số của mình và người chơi khác thông qua bảng xếp hạng, tạo thêm động lực để người chơi có điểm số cao hơn.

2.2. Nhược điểm

- Đồ họa đơn giản: Trò chơi có đồ họa 2D đơn giản, không phong phú và nhiều chi tiết như các trò chơi hiện đại khác, có thể không hấp dẫn những người thích hình ảnh sống động.

- Chức năng giới hạn: Trò chơi chỉ tập chung vào việc nối các cặp thú giống nhau. Chưa có nút tìm kiếm để trợ giúp người chơi.

- Chưa có chế độ 2 người chơi: Trò chơi chỉ có duy nhất chế độ chơi đơn offline , người chơi chưa thể giao lưu thi đấu trực tiếp cùng những người chơi khác.

**TÀI LIỆU THAM KHẢO**

[1.] Nguyễn Văn Quân. (2014). Thuật toán game pokemon (pikachu). Truy cập từ <https://cachhoc.net/2014/03/25/thuat-toan-game-pokemon-pikachu>

[2.] Oracle. (N.d.). The Java™ Tutorial. Truy cập từ <https://docs.oracle.com/javase/tutorial/>

[3.] “*JDBC 03. Cách kết nối với cơ sở dữ liệu bằng JDBC”.* Youtube, được tải lên bởi TITV, ngày 19 tháng 4 năm 2022, [https://www.youtube.com/watch?v=8OlMDhqZUk0](https://www.youtube.com/watch?v=8OlMDhqZUk0/)

[4.] Paul Fischer. *An Introduction to Graphical User Interfaces with Java Swing.* Hallow: Addition - Wesler, 2005.

[5.] Matt Weisfeld. *The Object-Oriented Thought Process, 5th Edition.* Addison-Wesley Professional, 2019.

[6.] NEOS Thành. *Lập trình hướng đối tượng JAVA core dành cho người mới bắt đầu học lập trình.* Nhà Xuất Bản Lao Động,2017.