

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**  
**KHOA QUỐC TẾ**

-----o0o-----



**Báo cáo bài tập lớn môn học**  
**Công nghệ Java**

*Đề tài: Làm game pikachu Cổ Điện*

Nhóm thực hiện: 9

Giảng viên hướng dẫn: Vũ Huân

Lớp: Công nghệ thông tin Việt Anh I

Thành viên nhóm:

- Vũ Văn Sử - 212602249
- Nguyễn Trung Hiếu - 882126015

# MỤC LỤC

<b>I. Tổng quan .....</b>	<b>3</b>
1. Giới thiệu .....	3
2. Mô tả Game Pikachu.....	3
<b>II. Xây dựng game.....</b>	<b>3</b>
1. Lớp main.....	3
2. Lớp gameTime .....	5
3. Lớp Line .....	6
4. Lớp GamePanel.....	6
4.1. Khởi tạo ma trận.....	6
4.2. Thập toàn tìm kiếm đường đi giữa 2 icon giống nhau.....	8
4.2.1. Icon cùng nằm trên 1 cạnh.....	8
4.2.2. Được nối bằng tối đa 3 đoạn thẳng trong phạm vi hình chữ nhật hình thành từ tọa độ của 2 icon. ..... .....	9
4.2.3. Được nối bằng tối đa 3 đoạn thẳng vượt ra ngoài phạm vi hình chữ nhật hình thành từ tọa độ của 2 icon. ....	10
4.2.4. Check2icon.....	13
4.3. Hiện thị ma trận game trên màn hình console. ....	15
5. Lớp GameFarme .....	15
5.1. Hàm tạo cửa sổ Game.....	15
5.2. Hàm tạo và sắp xếp components.....	15
5.3. Hàm tạo điểm số, thời gian và New Game xuất hiện ở đầu trang.....	16
5.3.1. Tạo điểm với giá trị ban đầu là 0 .....	16
5.3.2. Tạo điểm số và thời gian vùng chứa bảng điều khiển.....	16
5.3.3. Tạo Panel mới chứa panelScoreAndTime và nút New Game .....	16
5.3.4. Set BorderLayout để panelControl xuất hiện ở đầu trang.....	16
5.4. Hàm tạo 1 button .....	16
5.5. Hàm tạo Icon .....	17
5.6. Hàm reset hết icon,điểm và thời gian về trạng thái ban đầu .....	17

5.7. Hàm báo khi bấm vào NewGame .....	17
5.8. Hàm chạy thời gian .....	17
5.9. Các hàm này được sử dụng để lấy và thiết lập giá trị cho các thuộc tính cho panel.....	18
5.10. Hàm thông báo xem người chơi có muốn chơi lại không.....	19
<b>6. Lớp ButtonEvent.....</b>	<b>19</b>
6.1. Hàm khởi tạo ButtonEvent .....	19
6.2. Hàm khởi tạo JButton.....	20
6.3. Hàm khởi tạo một trò chơi mới.....	20
6.4. Hàm thiết lập icon cho button theo giá trị trong ma trận và sau đó thêm button đó vào .....	20
6.5. Hàm lấy ảnh và trả về định dạng icon.....	21
6.6. Hàm xóa 2 icon giống nhau được chọn.....	21
6.7. Hàm vô hiệu hóa những icon được xóa .....	21
6.8. ActionPerformed .....	21
<b>7. Lớp Sql.....</b>	<b>23</b>
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>27</b>

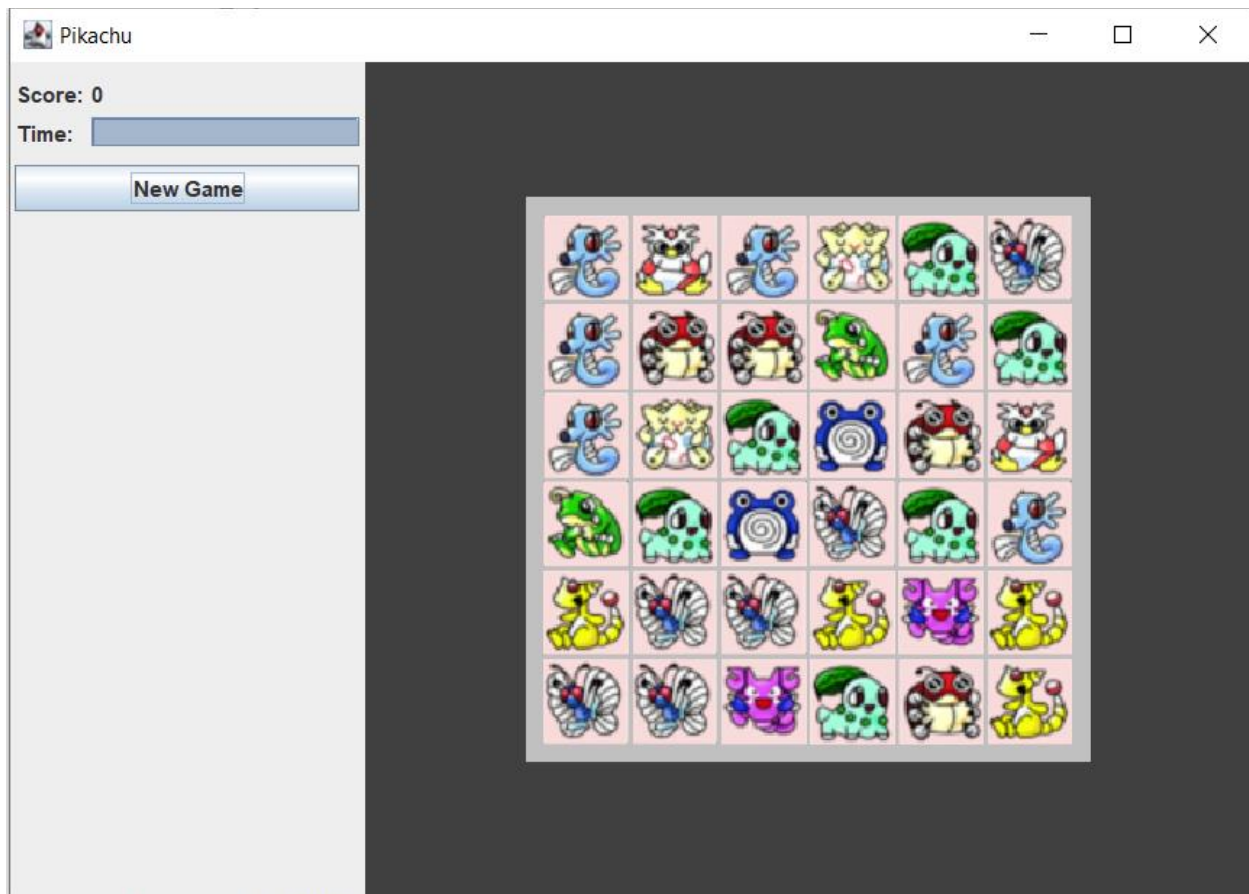
## I. Tổng quan

### 1. Giới thiệu

Game Pikachu cổ điển là một trong những trò chơi giải đố đầu tiên được phát triển trên máy tính. Trò chơi này đã trở thành một biểu tượng của thập niên 90 và vẫn được yêu thích đến ngày nay.

### 2. Mô tả Game Pikachu

Game Pikachu cổ điển là một trò chơi giải đố với nội dung chính là loại bỏ các cặp hình giống nhau để giành được điểm số. Trong game, người chơi phải tìm các cặp hình Pikachu giống nhau và loại bỏ chúng bằng cách kết nối chúng bằng một đường thẳng không quá ba góc vuông. Điểm số của người chơi sẽ tăng lên khi họ tìm được nhiều cặp hình giống nhau. Nếu ăn hết các hình giống nhau bạn sẽ thắng còn nếu hết thời gian thì bạn sẽ thua.



## II. Xây dựng game

### 1. Lớp main

Lớp Main chứa các phương thức Main dùng để bắt đầu game.

```
public class Main {
    GameFrame gframe;
    GamePanel gpanel;
    private sql sql1 = new sql();
    static final String DRIVER_CLASS = "com.mysql.cj.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost/sql1";
    static final String USER = "root";
    static final String PASS = "";
    public Main() {
        gframe = new GameFrame();
        GameTime gtime = new GameTime();
        gtime.start();
        new Thread(gframe).start();
    }
    public static void main(String[] args) {
        new Main();
    }
}
```

## 2. Lớp gameTime

Lớp này được sử dụng để tính và giảm thời gian chơi của một trò chơi và xử lý các trường hợp kết thúc trò chơi.

```
class gameTime extends Thread{
    public void run() {
        while (true) {
            try {
                //1s
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            if (gframe.isPause()) {
                if (gframe.isResume()) {
                    gframe.time--;
                }
            } else {
                gframe.time=gframe.time-100;
            }
            if (gframe.time == 0) {
                String name = JOptionPane.showInputDialog("Enter your name:");
                if(name != null || JOptionPane.CANCEL_OPTION != 0) {
                    try {
                        Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
                        sql.addToDatabase(conn, name, score);

                    } catch (SQLException e1) {
                        e1.printStackTrace();
                    } finally {
                        JOptionPane.showMessageDialog(gframe, "Điểm của bạn là: " + score);
                        gframe.NewGameWindow("Hết thời gian", "You lose!",2);
                    }
                }
            }
        }
    }
}
```

### 3. Lớp Line

Lớp Line được sử dụng để lưu trữ thông tin về đường thẳng nối giữa hai ô vuông trong trò chơi, để xác định xem hai ô vuông đó có thể được ăn và loại bỏ khỏi màn hình chơi hay không.

```
public class Line {  
  
    public Point p1;  
    public Point p2;  
  
    public Line(Point p1, Point p2) {  
        super();  
        this.p1 = p1;  
        this.p2 = p2;  
    }  
  
    public String toString() {  
        String string  
        = "(" + p1.x + "," + p1.y + ") and (" + p2.x + "," + p2.y + ")";  
        return string;  
    }  
}
```

### 4. Lớp GamePanel

#### 4.1. Khởi tạo ma trận

-Một ma trận chứa hình ảnh pokemon có dạng như dưới, mỗi con số trong ảnh được gán ngẫu nhiên từ 1-21 tương đương với số ảnh file res.

```
0 0 0 0 0 0 0 0  
0 6 7 15 7 9 12 0  
0 13 18 13 1 20 7 0  
0 1 14 20 7 8 4 0  
0 9 8 15 10 16 9 0  
0 4 16 12 14 20 10 0  
0 18 9 6 9 9 20 0  
0 0 0 0 0 0 0 0
```

-Hàm createMatrix() tạo một mảng 2 chiều chứa giá trị ngẫu nhiên từ 1-21 tương đương với số hình ảnh pokemon có trong file(imgCount = 21). Kích thước của ma trận được xác định bởi row(hàng) và col(cột).

```
private void createMatrix() {
    Random rnd = new Random();
    int imgCount = 21;
    int max = 5;
    int arr[] = new int[imgCount + 1];

    matrix = new int[row][col];
    for(int i = 0; i < col; i++) {
        matrix[0][i] = matrix[row - 1][i] = 0;
    }
    for(int i = 0; i < row; i++) {
        matrix[i][0] = matrix[i][col - 1] = 0;
    }
}
```

Tiếp đến sử dụng **ArrayList** để lưu trữ các điểm có tọa độ i,j ở trong ma trận (trừ các điểm nằm ở hàng/cột đầu và hàng/cột cuối). Sử dụng hai vòng for để lặp toàn bộ ma trận và thêm các điểm vào **listPoint**.

```
ArrayList<Point> listPoint = new ArrayList<Point>();
for(int i=1; i < row - 1; i++) {
    for(int j = 1 ; j < col - 1; j++) {
        listPoint.add(new Point(i, j));
    }
}
```

-Sử dụng vòng for tạo một cặp ảnh sau mỗi lần lặp (một cặp ảnh có thể xuất hiện nhiều lần theo biến max). Để tạo một cặp hình ảnh, các hình ảnh được chọn ngẫu nhiên từ 1 đến 21 được gán vào biến index. Nếu số lượng hình ảnh chưa đạt max thì số lượng hình ảnh của hình được chọn sẽ tăng lên 2 (arr[index] +=2).

```
for(int i=0 ; i < row*col/2;i++){
    int index=rnd.nextInt(imgCount) + 1;
    if(arr[index]<max) {
        arr[index] +=2;
        for(int j=0; j < 2; j++) {
            try {
                int size = listPoint.size();
                int pointIndex = rnd.nextInt(size);
                matrix[listPoint.get(pointIndex).x]
                    [listPoint.get(pointIndex).y] = index;
                listPoint.remove(pointIndex);
            } catch (Exception e) {
            }
        }
    }
}
```



Tiếp theo, lấy ngẫu nhiên một điểm từ danh sách 'listPoint'(biến 'pointIndex') và đặt giá trị của điểm đó trong ma trận là 'index' . Sau đó, điểm được chọn sẽ được loại bỏ khỏi danh sách listPoint.

Lặp lại vòng for cho đến khi nào đủ hình ảnh (row\*col/2).

#### 4.2. Thập toàn tìm kiếm đường đi giữa 2 icon giống nhau

##### 4.2.1. Icon cùng nằm trên 1 cạnh

Ta sẽ chia trường hợp này ra làm 2 trường hợp con: Nằm trên cùng 1 hàng ngang ( $x1 = x2$ ) hoặc nằm trên cùng 1 hàng dọc ( $y1 = y2$ )

TH1: 2 icon nằm trên cùng 1 hàng ngang ( $x1 = x2$ )

Ta xét 2 tọa độ còn lại là  $y1$  và  $y2$  để tìm ra điểm có hoành độ nhỏ hơn (giả sử trả về  $y1 < y2$ ), rồi xét liên tục các ô theo hàng ngang  $x1$  từ vị trí  $y1$  đến  $y2$ , nếu các ô được xét đều là ô trống (có giá trị là 0) thì sẽ trả về true và biến mất , còn nếu như các ô được xét có vật cản sẽ trả về false.

```
private boolean checkLineX(int y1, int y2, int x) {
    System.out.println("check line x");
    // find point have column max and min
    int min = Math.min(y1, y2);
    int max = Math.max(y1, y2);
    // run column
    for (int y = min + 1; y < max; y++) {
        if (matrix[x][y] > notBarrier) { // if see barrier then die
            System.out.println("die: " + x + " " + y);
            return false;
        }
        System.out.println("ok: " + x + " " + y);
    }

    // not die -> success
    return true;
}
```

TH2: 2 icon cùng nằm trên 1 hàng ngang ( $y1=y2$ )

Trường hợp này ta cũng xét tương tự trường hợp trên

```

private boolean checkLineY(int x1, int x2, int y) {
    System.out.println("check line 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] > notBarrier) {
            System.out.println("die: " + x + " " + y);
            return false;
        }
        System.out.println("ok: " + x + " " + y);
    }
    return true;
}

```

4.2.2. Được nối bằng tối đa 3 đoạn thẳng trong phạm vi hình chữ nhật hình thành từ tọa độ của 2 icon.

- Xét 2 điểm chỉ trong phạm vi được bọc bởi giới hạn từ 2 tọa độ của 2 icon ứng với 2 đầu của hình chữ nhật. Với trường hợp này, mình cũng chia làm 2 trường hợp nhỏ: Đó là xét theo chiều ngang và chiều dọc.

- 2 trường hợp này hoàn toàn tương tự nhau về cách xét, chỉ là khác về hướng đi. Mục đích của việc chia làm 2 trường hợp là để mình có thể bao quát toàn bộ các con đường với các hướng đi khác nhau.

- Ta xét 2 điểm p1, p2 với trường hợp xét theo chiều ngang trước nhé. Với việc bắt đầu tìm hướng đi theo chiều ngang, mình lấy ra 2 hoành độ của 2 điểm là y1 và y2. Sau đó, mình so sánh 2 hoành độ này, tìm ra hoành độ nhỏ hơn, và bắt đầu tịnh tiến từ ô có hoành độ nhỏ hơn sang phía ô có hoành độ lớn hơn với vòng for và giá trị tăng là 1. Với mỗi lần tịnh tiến y thêm 1 đơn vị ấy, mình liên tục kiểm tra xem ô nằm tại vị trí kiểm tra có phải là ô trống không. Nếu đó là ô trống, mình sẽ kiểm tra thêm 2 đoạn thẳng còn lại để nối 2 điểm ấy có thỏa mãn không bằng 2 hàm checkLineX và checkLineY với các tham số truyền vào tương ứng là điểm cuối cùng trong mỗi đoạn thẳng được vẽ. Nếu kiểm tra được 1 đường đi thỏa mãn, hàm kiểm tra sẽ trả về x. Ngược lại nếu gặp 1 icon khác trước khi tìm được đường đi thỏa mãn, hàm sẽ trả về -1. Nếu kiểm tra hết đường đi đến giới hạn của hình chữ nhật, nếu không có trường hợp nào đúng sẽ mặc định trả về -1.

- Trường hợp xét 2 icon theo chiều dọc cũng tương tự như vậy

```

private int checkRectY(Point p1, Point p2) {
    System.out.println("check rect y");
    // find point have y min
    Point pMinX = p1, pMaxX = p2;
    if (p1.x > p2.x) {
        pMinX = p2;
        pMaxX = p1;
    }
    // find line and y begin
    for (int x = pMinX.x; x <= pMaxX.x; x++) {
        if (x > pMinX.x && matrix[x][pMinX.y] > notBarrier) {
            return -1;
        }
        if ((matrix[x][pMaxX.y] == 0)
            && checkLineX(pMinX.y, pMaxX.y, x)
            && checkLineY(x, pMaxX.x, pMaxX.y)) {

            System.out.println("Rect y");
            System.out.println("(" + pMinX.x + "," + pMinX.y + ") -> (" + x
                + "," + pMinX.y + ") -> (" + x + "," + pMaxX.y
                + ") -> (" + pMaxX.x + "," + pMaxX.y + ")");
            return x;
        }
    }
    return -1;
}

```

4.2.3. Được nối bằng tối đa 3 đoạn thẳng vượt ra ngoài phạm vi hình chữ nhật hình thành từ tọa độ của 2 icon.

- Về cơ bản, phần này khá giống với phần trên, chỉ là giới hạn của việc tìm kiếm không phải chỉ ở giữa tung độ và hoành độ của 2 icon, mà vượt ra ngoài khoảng ấy. Mình chia việc kiểm tra thành 2 trường hợp: Theo chiều dọc và theo chiều ngang. Vì 2 trường hợp này cũng đều tương tự nhau, chỉ khác nhau về hướng tìm kiếm.

- Ta xét trường hợp theo chiều ngang hàm checkMoreLineX
- Để giải quyết vấn đề đoạn thẳng vượt ra ngoài hình chữ nhật hình thành từ tọa độ của 2 icon, tạo ra biến type trong tham số truyền vào của hàm checkMoreLineX. Biến type chỉ nhận 2 giá trị: 1 và -1, tương ứng là 2 hướng đi về phía trước, hoặc ngược lại. So sánh 2 icon rồi tìm ra icon có hoành độ nhỏ hơn tương ứng là pMinY và pMaxY. Tương ứng với giá trị của type, nếu type = 1, tạo ra 1 biến y nhận giá trị bằng pMaxY.y + type, ngược lại, nếu type = -1, biến y của mình sẽ nhận giá trị bằng pMaxY.y + type. Đối với việc xét theo chiều ngang này, với 2 biến pMaxY.y và pMinY.y chính là giới hạn hoành độ của hình chữ nhật tạo thành bởi 2 icon đang xét. Việc cộng

thêm type vào 2 biến chính là việc mở rộng đường tìm kiếm ra ngoài phạm vi của hình chữ nhật đó. Ta khởi tạo thêm 2 biến đó là row và colFinish. Quá trình tìm kiếm luôn có 2 điểm: Điểm bắt đầu tìm kiếm và điểm kết thúc tìm kiếm

- Tại đây ta có 3 điểm nổi bởi vì trường hợp đó là xét 2 icon và các đoạn nổi sẽ vượt ra ngoài phạm vi của hình chữ nhật tạo từ 2 icon ấy. Ta kiểm tra giá trị của ô tại điểm nổi này, chính là ô `matrix[row][colFinish]`, ô này bắt buộc phải có giá trị bằng 0 hoặc ô này chính là 1 trong 2 icon mà chúng ta đang xét, điều đó cũng có nghĩa là 2 icon nằm ở vị trí cùng 1 cột. Khi đó chúng ta mới tiếp tục xét đến các ô nằm ở vị trí trong khoảng từ ô bắt đầu kiểm tra đến ô tại điểm nổi thứ nhất. Đường đi này không được bị chặn bởi 1 icon nào, nghĩa là tất cả các giá trị của các ô trong khoảng đó đều sẽ bằng 0, kiểm tra nó bằng hàm `checkLineX(pMinY.y, pMaxY.y, row)`.

- Sau khi đã kiểm tra và vẽ được đoạn nổi đến điểm nổi thứ nhất thỏa mãn, chúng ta sẽ tiếp tục với điểm nổi thứ 2 và thứ 3. Về phần này, vì không còn giới hạn phạm vi, nên không thể dùng vòng for được. Thay vào đó, với biến y đã được khởi tạo ở trên, mình dùng vòng while để kiểm tra xem chừng nào 2 ô `matrix[pMinY.x][y]` và `matrix[pMaxY.x][y]` sẽ còn cùng có giá trị bằng 0. Trong vòng while, mình liên tục kiểm tra đoạn nổi giữa 2 điểm ứng với 2 ô ma trận đang xét bằng hàm `checkLineY(pMinY.x, pMaxY.x, y)`, nếu đoạn nổi ấy thỏa mãn, nghĩa là đã tìm được đường đi thỏa mãn. Khi này, hàm `checkMoreLineX` sẽ trả về x, nếu không, giá trị của y sẽ tiếp tục được cộng vào 1 giá trị bằng type (Tiếp tục tăng lên hoặc tiếp tục giảm đi). Nếu vòng While kết thúc mà vẫn chưa tìm được đường đi thỏa mãn, hàm sẽ trả về -1.

```

private int checkMoreLineX(Point p1, Point p2, int type) {
    System.out.println("check chec more x");
    Point pMinY = p1, pMaxY = p2;
    if (p1.y > p2.y) {
        pMinY = p2;
        pMaxY = p1;
    }
    int y = pMaxY.y + type;
    int row = pMinY.x;
    int colFinish = pMaxY.y;
    if (type == -1) {
        colFinish = pMinY.y;
        y = pMinY.y + type;
        row = pMaxY.x;
        System.out.println("colFinish = " + colFinish);
    }
    if ((matrix[row][colFinish] == notBarrier || pMinY.y == pMaxY.y)
        && checkLineX(pMinY.y, pMaxY.y, row)) {
        while (matrix[pMinY.x][y] == notBarrier
            && matrix[pMaxY.x][y] == notBarrier) {
            if (checkLineY(pMinY.x, pMaxY.x, y)) {

                System.out.println("TH X " + type);
                System.out.println("(" + pMinY.x + "," + pMinY.y + ") -> ("
                    + pMinY.x + "," + y + ") -> (" + pMaxY.x + "," + y
                    + ") -> (" + pMaxY.x + "," + pMaxY.y + ")");
                return y;
            }
            y += type;
        }
    }
    return -1;
}

```

```

private int checkMoreLineY(Point p1, Point p2, int type) {
    System.out.println("check more y");
    Point pMinX = p1, pMaxX = p2;
    if (p1.x > p2.x) {
        pMinX = p2;
        pMaxX = p1;
    }
    int x = pMaxX.x + type;
    int col = pMinX.y;
    int rowFinish = pMaxX.x;
    if (type == -1) {
        rowFinish = pMinX.x;
        x = pMinX.x + type;
        col = pMaxX.y;
    }
    if ((matrix[rowFinish][col] == notBarrier || pMinX.x == pMaxX.x)
        && checkLineY(pMinX.x, pMaxX.x, col)) {
        while (matrix[x][pMinX.y] == notBarrier
            && matrix[x][pMaxX.y] == notBarrier) {
            if (checkLineX(pMinX.y, pMaxX.y, x)) {
                System.out.println("TH Y " + type);
                System.out.println("(" + pMinX.x + "," + pMinX.y + ") -> ("
                    + x + "," + pMinX.y + ") -> (" + x + "," + pMaxX.y
                    + ") -> (" + pMaxX.x + "," + pMaxX.y + ")");
                return x;
            }
            x += type;
        }
    }
    return -1;
}

```

#### 4.2.4. Check2icon

kiểm tra 2 icon giống nhau không nếu có thì tiếp tục kiểm tra xem chúng có nằm trên cùng 1 hàng hoặc 1 cột không, rồi đến kiểm tra xem chúng có được nối với nhau bằng đường ziczac nằm trong phạm vi hình chữ nhật tạo bởi 2 icon đó không, và cuối cùng là kiểm tra 2 icon có được nối bằng đường ziczac nằm ngoài phạm vi hình chữ nhật ấy không. Việc kiểm tra theo thứ tự này sẽ giúp chúng ta kiểm tra tuần tự, tiết kiệm thời gian và không bị bỏ sót trường hợp nào cả

```

public Line check2Icon(Point p1, Point p2) {
    if (!p1.equals(p2) && matrix[p1.x][p1.y] == matrix[p2.x][p2.y]) {
        // check line with x
        if (p1.x == p2.x) {
            System.out.println("line x");
            if (checkLineX(p1.y, p2.y, p1.x)) {
                return new Line(p1, p2);
            }
        }
        // check line with y
        if (p1.y == p2.y) {
            System.out.println("line y");
            if (checkLineY(p1.x, p2.x, p1.y)) {
                System.out.println("ok line y");
                return new Line(p1, p2);
            }
        }
        int t = -1; // t is column find
        // check in rectangle with x
        // check in rectangle with y
        if ((t = checkRectX(p1, p2)) != -1) {
            System.out.println("rect x");
            return new Line(new Point(p1.x, t), new Point(p2.x, t));
        }
        // check in rectangle with y
        if ((t = checkRectY(p1, p2)) != -1) {
            System.out.println("rect y");
            return new Line(new Point(t, p1.y), new Point(t, p2.y));
        }

        // check more right
        if ((t = checkMoreLineX(p1, p2, 1)) != -1) {
            System.out.println("more right");
            return new Line(new Point(p1.x, t), new Point(p2.x, t));
        }
        // check more left
        if ((t = checkMoreLineX(p1, p2, -1)) != -1) {
            System.out.println("more left");
            return new Line(new Point(p1.x, t), new Point(p2.x, t));
        }
        // check more down
        if ((t = checkMoreLineY(p1, p2, 1)) != -1) {
            System.out.println("more down");
            return new Line(new Point(t, p1.y), new Point(t, p2.y));
        }
        // check more up
        if ((t = checkMoreLineY(p1, p2, -1)) != -1) {
            System.out.println("more up");
            return new Line(p1, p2);
        }
    }
    return null;
}

```

#### 4.3. Hiển thị ma trận game trên màn hình console.

```
// hàm hiển thị ma trận trên console
public void showMatrix() {
    for(int i=1; i < row - 1; i++) {
        for(int j=1; j < col - 1; j++) {
            System.out.printf("%3d", matrix[i][j]);
        }
        System.out.println();
    }
}
```

### 5. Lớp GameFrame

#### 5.1. Hàm tạo cửa sổ Game

```
public GameFrame() {
    jframe = new JFrame();

    jframe.add(jpanel = createMainPanel());
    jframe.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    jframe.setResizable(false);
    //(width, height)
    jframe.setSize(width, height);

    jframe.setResizable(true);
    jframe.setTitle("Pikachu");
    jframe.setLocationRelativeTo(null);
    jframe.setVisible(true);
}
```

#### 5.2. Hàm tạo và sắp xếp components



```
private JPanel createMainPanel() {
    JPanel jpanel = new JPanel(new BorderLayout());
    jpanel.add(createIconButton(), BorderLayout.CENTER);
    jpanel.add(createControlPanel(), BorderLayout.WEST);
    return jpanel;
}
```

### 5.3. Hàm tạo điểm số, thời gian và New Game xuất hiện ở đầu trang

#### 5.3.1. Tạo điểm với giá trị ban đầu là 0

```
//tạo JLabel lblScore với giá trị ban đầu là
lblScore = new JLabel("0");
progressTime = new JProgressBar(0, 100);
progressTime.setValue(100);
```

#### 5.3.2. Tạo điểm số và thời gian vùng chứa bảng điều khiển

```
JPanel panelLeft = new JPanel(new GridLayout(2, 1, 5, 5));
panelLeft.add(new JLabel("Score:"));
panelLeft.add(new JLabel("Time:"));

JPanel panelCenter = new JPanel(new GridLayout(2, 1, 5, 5));
panelCenter.add(lblScore);
panelCenter.add(progressTime);

JPanel panelScoreAndTime = new JPanel(new BorderLayout(5, 0));
panelScoreAndTime.add(panelLeft, BorderLayout.WEST);
panelScoreAndTime.add(panelCenter, BorderLayout.CENTER);
```

#### 5.3.3. Tạo Panel mới chứa panelScoreAndTime và nút New Game

```
JPanel panelControl = new JPanel(new BorderLayout(10, 10));
panelControl.setBorder(new EmptyBorder(10, 3, 5, 3));
panelControl.add(panelScoreAndTime, BorderLayout.PAGE_START);
panelControl.add(btnNewGame = createButton("New Game"), BorderLayout.PAGE_END);
```

#### 5.3.4. Set BorderLayout để panelControl xuất hiện ở đầu trang

```
JPanel jpanel = new JPanel(new BorderLayout());
jpanel.add(panelControl, BorderLayout.PAGE_START);

return jpanel;
```

### 5.4. Hàm tạo 1 button

```
private JButton createButton(String ButtonName) {
    JButton btn = new JButton(ButtonName);
    btn.addActionListener(this);
    return btn;
}
```

### 5.5. Hàm tạo Icon

```
// tạo icon
private JPanel createIconButton() {
    IconButton = new ButtonEvent(this, row, col);
    //resize in Button event so no need for gridbagconstraints
    JPanel panel = new JPanel(new GridBagLayout());
    panel.setBackground(Color.DARK_GRAY);
    panel.add(IconButton);
    return panel;
}
```

### 5.6. Hàm reset hết icon, điểm và thời gian về trạng thái ban đầu

```
public void newGame() {
    time = maxTime;
    IconButton.removeAll();
    jpanel.add(createIconButton(), BorderLayout.CENTER);
    jpanel.validate();
    jpanel.setVisible(true);
    lbScore.setText("0");
}
```

### 5.7. Hàm báo khi bấm vào NewGame

```
override
public void actionPerformed(ActionEvent e) {
    if(e.getSource() == btnNewGame) {
        NewGameWindow("Game chưa xong bạn ơi!. Muốn chơi lại hử?" , "Warning!", 0);
    }
}
```

### 5.8. Hàm chạy thời gian

```

@Override
public void run() {
    while(true) {
        try {
            //1s
            Thread.sleep(1000);

        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        progressTime.setValue((int) ((double) time / maxTime * 100));
    }
}
}

```

5.9. Các hàm này được sử dụng để lấy và thiết lập giá trị cho các thuộc tính cho panel

- `getProgressTime` và `setProgressTime`: lấy và thiết lập giá trị cho thanh tiến trình của panel.

```

public JProgressBar getProgressTime() {
    return progressTime;
}

public void setProgressTime(JProgressBar progressTime) {
    this.progressTime = progressTime;
}

```

- `isResume` và `setResume`: lấy và thiết lập giá trị cho trạng thái "resume" của panel. Giá trị "resume" chỉ ra xem trò chơi có được khôi phục hay không.

```

public boolean isResume() {
    return resume;
}

public void setResume(boolean resume) {
    this.resume = resume;
}

```

- `isPause` và `setPause`: lấy và thiết lập giá trị cho trạng thái "pause" của panel. Giá trị "pause" chỉ ra xem trò chơi có đang ở trạng thái tạm dừng hay không.

```

public boolean isPause() {
    return pause;
}

public void setPause(boolean pause) {
    this.pause = pause;
}

```

5.10. Hàm thông báo xem người chơi có muốn chơi lại không.

- Hàm sẽ hiện thị thông báo xem người chơi có muốn tiếp tục chơi không nếu có hàm sẽ gọi về phương thức newgame trả về true và nếu người chơi muốn dừng lại sẽ về system.exit(0) trả về false

```
public boolean NewGameWindow(String text, String title, int t) {  
  
    pause = true;  
    resume = false;  
    int select = JOptionPane.showOptionDialog(null, text, title,  
        JOptionPane.YES_NO_OPTION,  
        JOptionPane.QUESTION_MESSAGE, null, null, null);  
    if(select == 0) {  
        pause = false;  
        newGame();  
        return true;  
    } else {  
        if(t==1) {  
            System.exit(0);  
            return false;  
        } else {  
            resume = true;  
            return true;  
        }  
    }  
}  
}
```

## 6. Lớp ButtonEvent

### 6.1. Hàm khởi tạo ButtonEvent

Hàm này được sử dụng để khởi tạo đối tượng ButtonEvent, lấy các thông số là số hàng, số cột của trò chơi và khởi tạo một số thuộc tính khác của đối tượng như kích thước, màu nền, đường viền, số cặp thẻ, sau đó gọi hàm newGame() để bắt đầu trò chơi.

```

public ButtonEvent(GameFrame gframe, int row, int col) {
    this.gframe = gframe;
    this.row = row + 2;
    this.col = col + 2;
    //10*10/2=50
    pair = row * col / 2;

    //row=10, col=10, gian cach = bound
    setLayout(new GridLayout(row, col, bound, bound));
    setBackground(bgColor);
    setPreferredSize(new Dimension((size + bound) * col, (size + bound) * row));
    setBorder(new EmptyBorder(10, 10, 10, 10));
    setAlignmentY(JPanel.CENTER_ALIGNMENT);
    |
    newGame();
}

```

## 6.2. Hàm khởi tạo JButton

```

private JButton createButton(String action) {
    JButton btn = new JButton();
    btn.setActionCommand(action);
    btn.setBorder(null);
    btn.addActionListener(this);
    return btn;
}

```

## 6.3. Hàm khởi tạo một trò chơi mới.

Hàm này dùng để khởi tạo một game mới với kích thước đã được chỉ định bởi row và col

```

public void newGame() {
    gpanel = new GamePanel(this.gframe, this.row, this.col);
    addArrayButton();
}

```

## 6.4. Hàm thiết lập icon cho button theo giá trị trong ma trận và sau đó thêm button đó vào

```

private void addArrayButton() {
    btn = new JButton[row][col];
    for(int i = 1; i < row - 1; i++) {
        for(int j = 1; j < col - 1; j++) {
            //tao nut (i,j) cho icon o vi tri (i,j)
            btn[i][j] = createButton(i + ", " + j);
            Icon icon = getIcon(gpanel.getMatrix()[i][j]);
            btn[i][j].setIcon(icon);
            add(btn[i][j]);
        }
    }
}

```

### 6.5. Hàm lấy ảnh và trả về định dạng icon

```

private Icon getIcon(int index){
    int width = 48, height = 48;
    Image image = new ImageIcon("src/res/" + index + ".png").getImage();
    Icon icon = new ImageIcon(image.getScaledInstance(width, height,
        image.SCALE_SMOOTH));
    return icon;
}

```

### 6.6. Hàm xóa 2 icon giống nhau được chọn

```

public void execute(Point p1, Point p2) {
    System.out.println("delete");
    setDisable(btn[p1.x][p1.y]);
    setDisable(btn[p2.x][p2.y]);
}

```

### 6.7. Hàm vô hiệu hóa những icon được xóa

```

//xóa icon q1
private void setDisable(JButton btn) {
    btn.setIcon(null);
    btn.setBackground(bgColor);
    btn.setEnabled(false);
}

```

### 6.8. ActionPerformed

Hàm này được sử dụng để xử lý sự kiện khi người dùng nhấn vào một nút trên game board. Nó lấy vị trí của nút được nhấn, kiểm tra xem đó có phải là vị trí của cặp icon có thể loại bỏ được không, nếu có thì thực hiện việc loại bỏ cặp icon đó và cập nhật điểm số và thời gian chơi. Nếu không, nó sẽ xóa hiệu ứng chọn icon trên cặp icon trước đó và đợi cho người dùng chọn lại một cặp icon khác. Nếu không còn cặp icon nào nữa, hàm này sẽ yêu cầu người dùng

nhập tên của họ và lưu điểm số của họ vào cơ sở dữ liệu, sau đó hiển thị một hộp thoại hỏi người dùng có muốn chơi lại không.

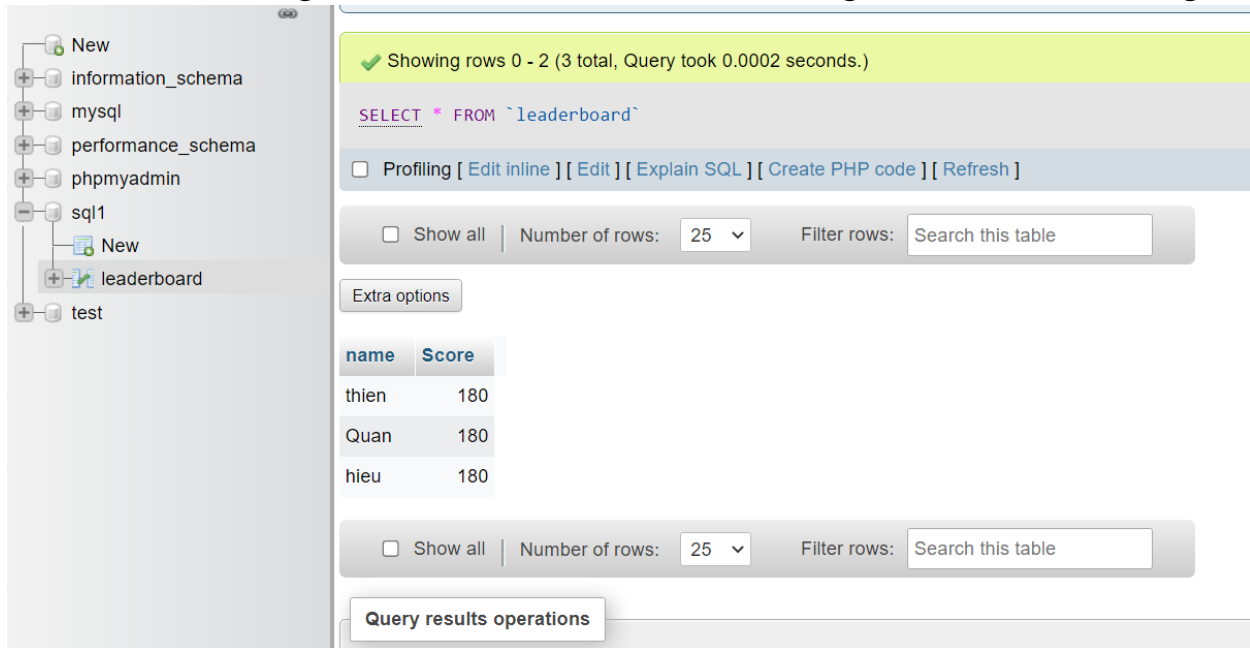
```

@Override
public void actionPerformed(ActionEvent e) {
    String btnIndex = e.getActionCommand();
    int indexDot = btnIndex.lastIndexOf(".");
    int x = Integer.parseInt(btnIndex.substring(0, indexDot));
    int y = Integer.parseInt(btnIndex.substring(indexDot + 1,
        btnIndex.length()));
    if (p1 == null) {
        //hieu ung chon icon
        p1 = new Point(x, y);
        btn[p1.x][p1.y].setBorder(new LineBorder(Color.GREEN));
    } else {
        p2 = new Point(x, y);
        line = gpanel.check2Icon(p1, p2);
        if (line != null) {
            System.out.println("line != null");
            gpanel.getMatrix()[p1.x][p1.y] = 0;
            gpanel.getMatrix()[p2.x][p2.y] = 0;
            gpanel.showMatrix();
            execute(p1, p2);
            line = null;
            score += 10;
            pair--;
            gframe.time++;
            gframe.lbScore.setText(score + "");
        }
        //xoa hieu ung chon icon
        btn[p1.x][p1.y].setBorder(null);
        p1 = null;
        p2 = null;
        System.out.println("done");
        if (pair == 0) {
            String name = JOptionPane.showInputDialog("Enter your name:");
            try {
                Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
                sql.addToDatabase(conn, name, score);
            } catch (SQLException e1) {
                e1.printStackTrace();
            }
            gframe.NewGameWindow("Thắng rồi! Muốn chơi lại khum?", "Win", 1);
        }
    }
}
}
}

```

## 7. Lớp Sql

Sử dụng cơ sở dữ liệu đã được tạo ở trên PHPMYAdmin để lưu trữ dữ liệu như tên và điểm của người chơi mỗi khi chơi xong hoặc hết thời gian.



The screenshot shows the PHPMYAdmin interface. On the left, the database structure is visible, showing a database named 'sql1' containing a table named 'leaderboard'. The main area displays the query results for the SQL query 'SELECT \* FROM `leaderboard`'. The results show three rows of data:

name	Score
thien	180
Quan	180
hieu	180

The interface also includes a query box with the SQL query, a status bar indicating 'Showing rows 0 - 2 (3 total, Query took 0.0002 seconds.)', and various controls for displaying the results, such as 'Show all', 'Number of rows', and 'Filter rows'.

Các bước để kết nối đến cơ sở dữ liệu tên sql1 có bảng leaderboard chứa tên và điểm người chơi trên PHPMYAdmin.



Đây là một đoạn code Java kết nối và truy vấn cơ sở dữ liệu MySQL:

```
import java.sql.*;

public class sql {

    static final String DRIVER_CLASS = "com.mysql.cj.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost/sql1";
    static final String USER = "root";
    static final String PASS = "";
    private static Statement stmt;

    public static void main(String[] args) throws SQLException, ClassNotFoundException {
        Connection conn = null;
        Statement stmt = null;
        try {
            System.out.println("STEP 1: Register JDBC driver");
            Class.forName(DRIVER_CLASS);

            System.out.println("STEP 2: Open a connection");
            conn = DriverManager.getConnection(DB_URL,USER,PASS);

            System.out.println("STEP 3: Execute a query");
            String query = "Select * FROM leaderboard";
            stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(query);

            System.out.println("STEP 4: Extract data from result set");
            while (rs.next()) {
                int score = rs.getInt("score");
                String name = rs.getString("name");

                System.out.println("name: " + name);
                System.out.println("Score: " + score);
            }
            rs.close();
        } finally {
            System.out.println("STEP 5: Close connection");
            if (stmt != null)
                stmt.close();
            if (conn != null)
                conn.close();
        }
        System.out.println("Done!");
    }
}
```

```
public class sql {

    static final String DRIVER_CLASS = "com.mysql.cj.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost/sql1";
    static final String USER = "root";
    static final String PASS = "";
    private static Statement stmt;
```

Bước 1: Đăng ký driver JDBC.(DRIVER\_CLASS đã được xác định ở trên)

```
System.out.println("STEP 1: Register JDBC driver");
Class.forName(DRIVER_CLASS);
```

Bước 2: Mở một kết nối đến cơ sở dữ liệu bằng phương thức getConnection() của DriverManager. (DB\_URL,USER,PASS đã được xác định ở trên)

```
System.out.println("STEP 2: Open a connection");
conn = DriverManager.getConnection(DB_URL,USER,PASS);
```

Bước 3: Thực hiện truy vấn SQL bằng câu lệnh SQL SELECT. (SELECT \* FROM leaderboard)

```
System.out.println("STEP 3: Execute a query");
String query = "Select * FROM leaderboard";
stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery(query);
```

Bước 4: Xử lý kết quả truy vấn bằng cách lấy dữ liệu name và score từ ResultSet và in ra màn hình.

```
System.out.println("STEP 4: Extract data from result set");
while (rs.next()) {
    int score = rs.getInt("score");
    String name = rs.getString("name");

    System.out.println("name: " + name);
    System.out.println("Score: " + score);
}
rs.close();
```

Bước 5: Đóng kết nối đến cơ sở dữ liệu.

```
    } finally {
        System.out.println("STEP 5: Close connection");
        if (stmt != null)
            stmt.close();
        if (conn != null)
            conn.close();
        System.out.println("Done!");
    }
}
```

Phương thức addToDatabase có mục đích để ghi thông tin name và score nhập từ game vào trong bảng leaderboard của cơ sở dữ liệu MySQL.

```

public static void addToDatabase(Connection conn, String name, int score) throws SQLException {
    PreparedStatement ps = null;
    try {
        String query = "INSERT INTO leaderboard (name, score) VALUES (?, ?)";
        ps = conn.prepareStatement(query);
        ps.setString(1, name);
        ps.setInt(2, score);
        ps.executeUpdate();
        System.out.println("Record inserted successfully");
    } finally {
        if (ps != null)
            ps.close();
    }
}

```

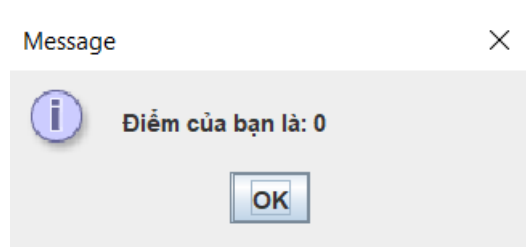
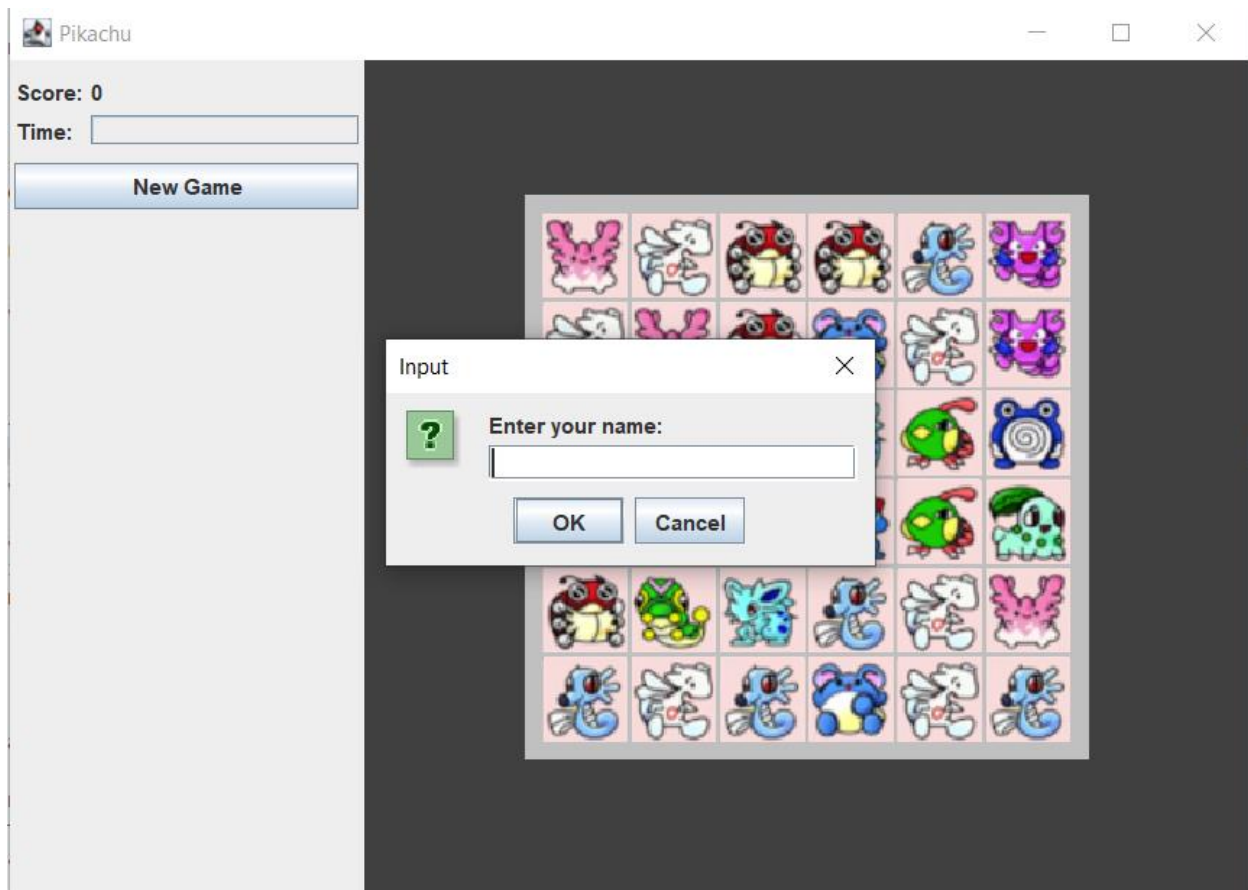
Áp dụng vào game pikachu:

```

)        if (pair == 0) {
.            String name = JOptionPane.showInputDialog("Enter your name:");
!            try {
:                Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
:                sql.addToDatabase(conn, name, score);
;
:            } catch (SQLException e1) {
:                e1.printStackTrace();
:            }
)            JOptionPane.showMessageDialog(gframe, "Điểm của bạn là: " + score);
)            gframe.NewGameWindow("Thắng rồi! Muốn chơi lại khum?", "Win",1);
:        }
:    }

```

Khi chơi xong thì sẽ có một popup hiện lên để người chơi điền tên vào



Sau khi nhập tên thì game sẽ nhận giá trị name và tiếp đến là nhận giá trị Score tại thời điểm nhập tên và cập nhật bảng leaderboard của cơ sở dữ liệu sql1.

## TÀI LIỆU THAM KHẢO

<https://nguyenvanquan7826.wordpress.com/2014/03/25/thuat-toan-game-pokemon-pikachu/>

<https://codelearn.io/sharing/lam-game-sieu-xin-bang-java-phan-1>

<https://codelearn.io/sharing/lam-game-sieu-xin-bang-java-phan-2>

