ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

———————— * ————————

BÁO CÁO ĐỊNH KỲ

PROJECT II

POKEMON GAME

Sinh viên: Nguyễn Đức Duy
MSSV: 20210275
Giảng viên hướng dẫn: TS. Đinh Thị Hà Ly

Hà Nội, tháng 4 năm 2024

Tạo ra thanh thời gian gian, điểm và nút new game

Khai báo các biến cần thiết trong class FrameController

```
private int MAX_TIME = 500;
public int time = MAX_TIME;
public JLabel lbScore;
private JProgressBar progressTime;
private JButton btnNewGame;
```

Khởi tạo nút new game

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

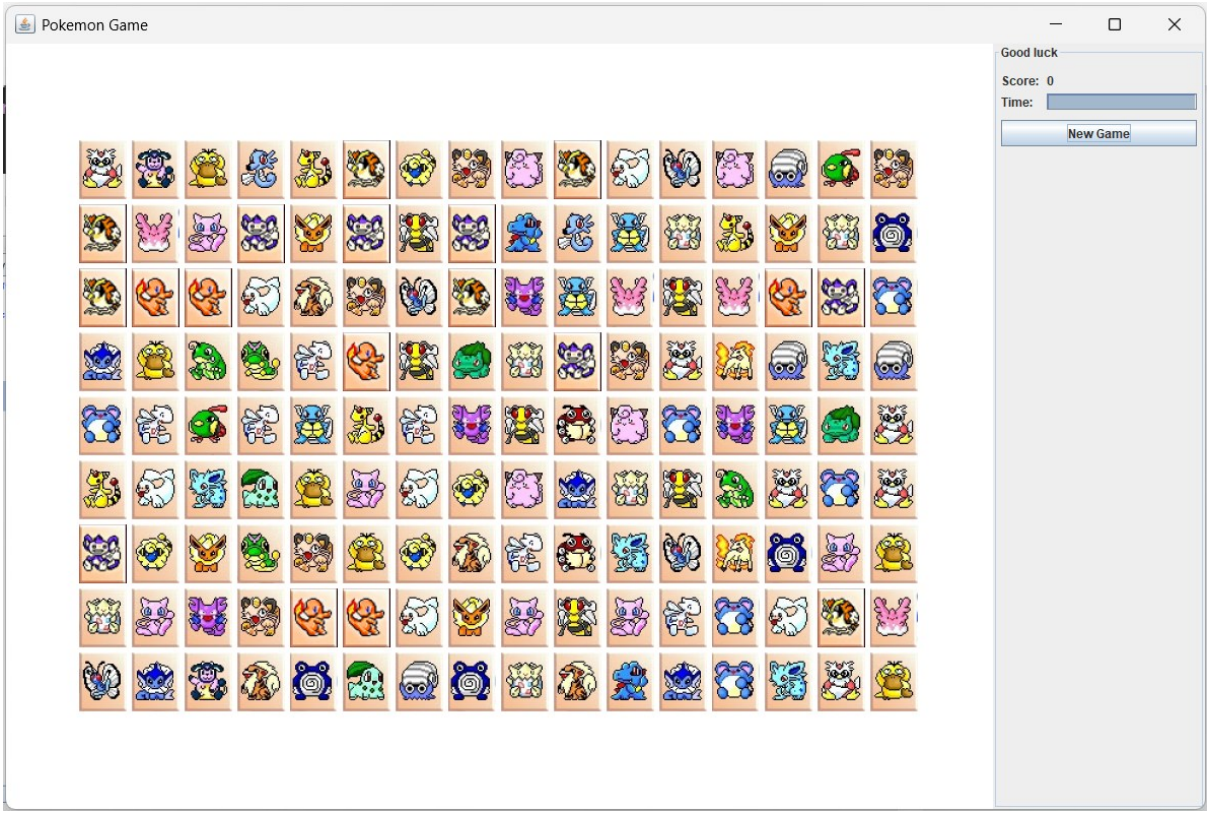
Khởi tạo nốt các thành phần như thanh thời gian, thanh điểm, nút new game,… bằng cách hoàn thiện hàm createControlPanel()

```
private JPanel createControlPanel() {
    lbScore = new JLabel("0");
    progressTime = new JProgressBar(0, 100);
    progressTime.setValue(100);

    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(lbScore);
    panelCenter.add(progressTime);

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

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

    JPanel panel = new JPanel(new BorderLayout());
    panel.setBorder(new TitledBorder("Good luck"));
    panel.add(panelControl, BorderLayout.PAGE_START);

    return panel;
}
```

Gọi nó trong hàm createMainPanel()

```
private JPanel createMainPanel() {
    JPanel panel = new JPanel(new BorderLayout());
    panel.add(createGraphicsPanel(), BorderLayout.CENTER);
    panel.add(createControlPanel(), BorderLayout.EAST); // khởi tạo các thanh điểm, score, new game
    return panel;
}
```

Được giao diện như sau



Thêm các chức năng cho nút new game, khi gọi hàm này mọi dữ liệu về time, score, icon đều được cài lại mặc định

```
public void newGame() {
    time = MAX_TIME;
    graphicsPanel.removeAll();
    mainPanel.add(createGraphicsPanel(), BorderLayout.CENTER);
    mainPanel.validate();
    mainPanel.setVisible(true);
    lbScore.setText("0");
}
```

Hàm giảm dần thanh time

```
@Override
public void run() {
    while(true) {
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        progressTime.setValue((int) ((double) time / MAX_TIME * 100));
    }
}
```

Tạo ra hàm showDialogNewGame() để hiển thị thông báo khi thắng, thua hoặc ấn nút new game, đồng thời khi gọi hàm này sẽ hiện lên pop up, khi đó thanh thời gian sẽ dừng lại, và khi tiếp tục chơi thì thanh thời gian chạy tiếp.

```
private boolean pause = false;
private boolean resume = false;
```

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

Hoàn thiện việc xử lý thời gian của game, thay đổi giá trị time, trong class Main, tạo class Time extend từ lớp Thread

```
class Time extends Thread {
    public void run() {
        while(true) {
            try {
                Thread.sleep(1000);
            } catch(InterruptedException e) {
                e.printStackTrace();
            }

            if(frame.isPause()) {
                if(frame.isResume()) {
                    frame.time--;
                }
            } else {
                frame.time--;
            }
            if(frame.time == 0) {
                frame.showDialogNewGame("Full time\nDo you want play again?", "Lose", 1);
            }
        }
    }
}
```

Xử lý thao tác với các icon

Tạo class PointLine trong package controller định nghĩa việc kết nối 2 icon với nhau

Main.java 2, M | PointLine.java U | InitMatrixCont

src > com > pokemon > controller > PointLine.java > ...

```
1   package com.pokemon.controller;
2
3   import java.awt.Point;
4
5   public class PointLine {
6       public Point p1;
7       public Point p2;
8
9       public PointLine(Point p1, Point p2) {
10          super();
11          this.p1 = p1;
12          this.p2 = p2;
13      }
14  }
15
```

Trong class IconEventController tạo ra các biến

```
private Point p1 = null;
private Point p2 = null;
private PointLine line;
private int score = 0;
private int item;
```

Tạo 2 hàm là execute() và setDisable() để thực hiện việc xóa các icon, trong hàm setDisable() khi các icon tương ứng trên các button bị xóa đi, các button ấy sẽ bị vô hiệu hóa và background trả về background của các khối panel, còn hàm execute() sẽ nhận 2 tham số truyền vào là hai Point p1, p2 để setDisable() tương ứng với 2 point này

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

private void setDisable(JButton btn) {
    btn.setIcon(null);
    btn.setBackground(backgroundColor);
    btn.setEnabled(false);
}
```

Override hàm actionPerformed() trong class IconEventController để xử lý event của các button tương ứng với các icon được tạo ra.

```
@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) {
        p1 = new Point(x, y);
        btn[p1.x][p1.y].setBorder(new LineBorder(Color.red));
    } else {
        p2 = new Point(x, y);
        System.out.println("(" + p1.x + "," + p1.y + ")" + "--> " + "(" + p2.x + "," + p2.y + ")");
        line = initMatrixController.checkTwoPoint(p1, p2);
        if(line != null) {
            System.out.println("line != null");
            initMatrixController.getMatrix()[p1.x][p1.y] = 0;
            initMatrixController.getMatrix()[p2.x][p2.y] = 0;
            initMatrixController.showMatrix();
            execute(p1, p2);
            line = null;
            score += 10;
            item--;
            frame.time++;
            frame.lbScore.setText(score + "");
        }
        btn[p1.x][p1.y].setBorder(null);
        p1 = null;
        p2 = null;
        System.out.println("done");
        if(item == 0) {
            frame.showDialogNewGame("Win!\nDo you want play again?", "Win", 1);
        }
    }
}
```

Khi click vào icon sẽ lưu lại tọa độ và lần lượt lưu vào 2 biến x, y

```
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()));
```

Kiểm tra p1, nếu p1 = null tức là chưa icon nào được chọn, gán cho p1 bằng point mới với tọa độ x, y trên, thêm border màu đỏ để nổi bật

```
if(p1 == null) {
    p1 = new Point(x, y);
    btn[p1.x][p1.y].setBorder(new LineBorder(Color.red));
```

Nếu không nghĩa là có 1 icon đã được chọn rồi, khởi tạo Point mới, check các điều kiện, thỏa mãn sẽ tăng điểm và thời gian

```
p2 = new Point(x, y);
System.out.println("(" + p1.x + "," + p1.y + ")" + "--> " + "(" + p2.x + "," + p2.y + ")");
line = initMatrixController.checkTwoPoint(p1, p2);
if(line != null) {
    System.out.println("line != null");
    initMatrixController.getMatrix()[p1.x][p1.y] = 0;
    initMatrixController.getMatrix()[p2.x][p2.y] = 0;
    initMatrixController.showMatrix();
    execute(p1, p2);
    line = null;
    score += 10;
    item--;
    frame.time++;
    frame.lbScore.setText(score + "");
```

Thuật toán tìm kiếm giữa 2 icon giống nhau

Khởi tạo lại ma trận để tạo ra ma trận được bao quanh bởi các số 0, gọi là các vị trí trống để sau này có thể tìm đường đi men theo các đường viền ma trận

```
public IconEventController(FrameController frame, int row, int col) {
    this.frame = frame;
    this.row = row + 2;
    this.col = col + 2;
    item = row * col / 2;
```

```
private void createMatrix() {
    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;
    }

    Random rand = new Random();
    int imgCount = 36;
    int max = 6;
    int[] arr = new int[imgCount + 1];
    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));
        }
    }

    int i = 0;
```

```java
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();
    }
}
```

```java
private void addArrayButton() {
    btn = new JButton[row][col];
    for(int i = 1; i < row - 1; ++i) {
        for(int j = 1; j < col - 1; ++j) {
            btn[i][j] = createButton(i + "," + j);
            Icon icon = getIcon(initMatrixController.getMatrix()[i][j]);
            btn[i][j].setIcon(icon);
            add(btn[i][j]);
        }
    }
}
```

Trường hợp 1: 2 icon nằm trên cùng một cạnh

Nếu nằm trên cùng 1 hàng ngang x1=x2, xét 2 tọa độ còn lại để tìm ra điểm có hoành độ nhỏ hơn, rồi xét liên tục các ô theo hàng ngang từ x1 từ vị trí y1 đến y2, nếu các ô đều trống nghĩa là thỏa mãn

Tương tự với cùng hàng dọc

```java
private boolean checkLineX(int y1, int y2, int x) {
    System.out.println("check line 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] != 0) {
            System.out.println("die: " + x + "" + y);
            return false;
        }
        System.out.println("ok: " + x + "" + y);
    }

    return true;
}

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] != 0) {
            System.out.println("die: " + x + "" + y);
            return false;
        }
        System.out.println("ok: " + x + "" + y);
    }

    return true;
}
```

Trường hợp 2: được nối bằng tối đa 3 đoạn trong phạm vi hình chữ nhật hình thành từ tọa độ của 2 icon

Xét 2 điểm p1, p2 với trường hợp xét theo chiều ngang. Lấy ra tung độ y1, y2, so sánh tìm hoành độ nhỏ hơn, bắt đầu tịnh tiến từ ô có tung độ nhỏ sang lớn, mỗi lần tịnh tiến kiểm tra xem có phải ô trống không, nếu trống kiểm tra thêm 2

đoạn thẳng còn lại để nối 2 điểm có thỏa mãn không bằng 2 hàm checkLineX() và checkLineY()

Tương tự với trường hợp chiều dọc

```
private boolean checkRectX(Point p1, Point p2) {
    System.out.println("check rect x");

    Point pMiny = p1, pMaxy = p2;
    if(p1.y > p2.y) {
        pMiny = p2;
        pMaxy = p1;
    }

    for(int y = pMiny.y; y <= pMaxy.y; y++) {
        if(y > pMiny.y && matrix[pMiny.x][y] != 0) {
            return false;
        }

        if((matrix[pMaxy.x][y] == 0) && checkLineY(pMiny.x, pMaxy.x, y) && checkLineX(y, pMaxy.y, pMaxy.y)) {
            System.out.println("Rect x");
            System.out.println("(" + pMiny.x + "," + pMiny.y + ") -> ("
                    + pMiny.x + "," + y + ") -> (" + pMaxy.x + "," + y
                    + ") -> (" + pMaxy.x + "," + pMaxy.y + ")");

            return true;
        }
    }

    return false;
}
```

```
private boolean checkRectY(Point p1, Point p2) {
    System.out.println("check rect y");
    Point pMinx = p1, pMaxx = p2;
    if(p1.x > p2.x) {
        pMinx = p2;
        pMaxx = p1;
    }

    for(int x = pMinx.x; x <= pMaxx.x; x++) {
        if(x > pMinx.x && matrix[x][pMinx.y] != 0) {
            return false;
        }

        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 true;
        }
    }
    return false;
}
```

Trường hợp 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.

Đối với trường hợp này, tạo ra hàm checkMoreLineX(). Để giải quyết vấn đề đoạn thẳng vượt ra ngoài hình chữ nhật hình thành từ đọa 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** sẽ nhận giá trị bằng pMinY.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.

```
private boolean 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] == 0 || pMiny.y == pMaxy.y) && checkLineX(pMiny.y, pMaxy.y, row)) {
        while(matrix[pMiny.x][y] == 0 && matrix[pMaxy.x][y] == 0) {
            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 true;
            }
            y += type;
        }
    }
    return false;
}
```

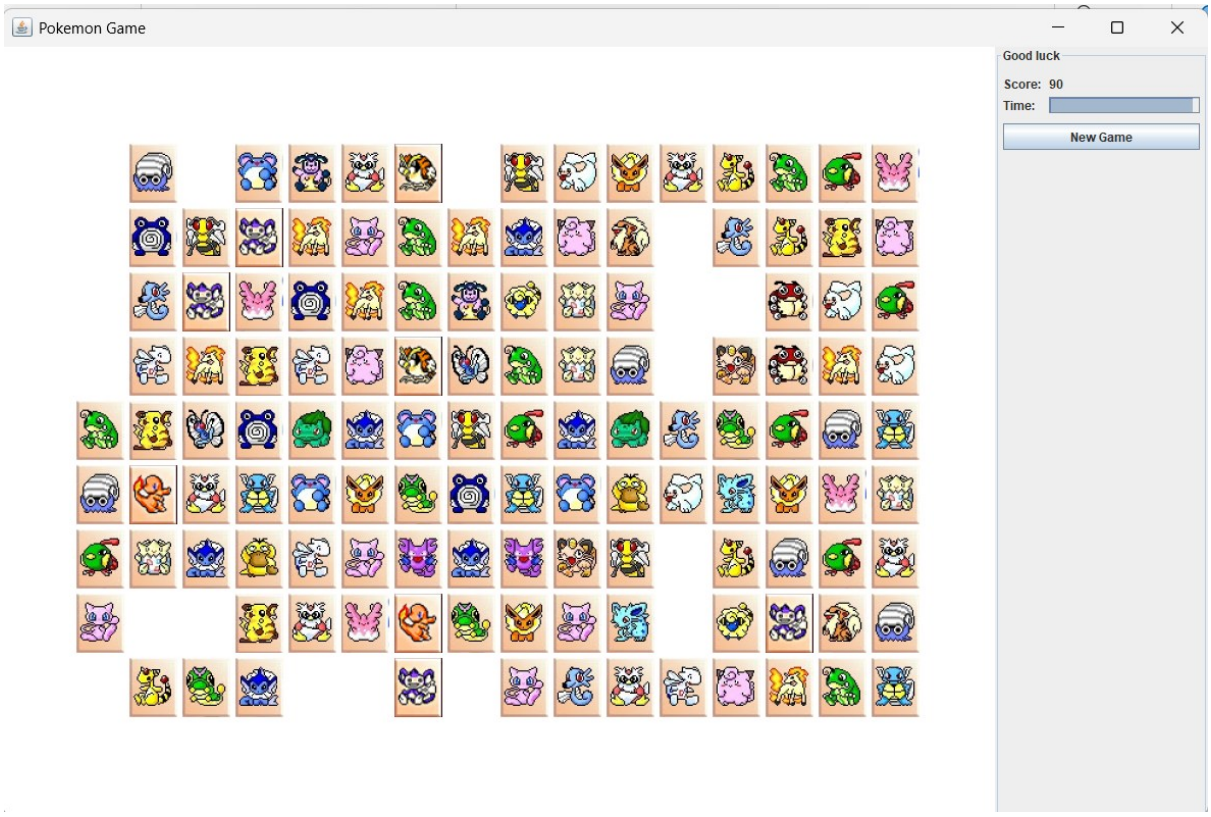
```
private boolean checkMoreLineY(Point p1, Point p2, int type) {
    System.out.println("check more y");
    Point pMinx = p1, pMaxx = p2;
    if (p1.x > p2.x) {
        pMinx = p2;
        pMaxx = p1;
    }
    int x = pMaxx.x + type;
    int col = pMinx.y;
    int rowFinish = pMaxx.x;
    if (type == -1) {
        rowFinish = pMinx.x;
        x = pMinx.x + type;
        col = pMaxx.y;
    }
    if ((matrix[rowFinish][col] == 0|| pMinx.x == pMaxx.x)
            && checkLineY(pMinx.x, pMaxx.x, col)) {
        while (matrix[x][pMinx.y] == 0
                && matrix[x][pMaxx.y] == 0) {
            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 true;
            }
            x += type;
        }
    }
    return false;
}
```

Hoàn thiện hàm checkTwoPoint() với 3 trường hợp ở trên

```
public PointLine checkTwoPoint(Point p1, Point p2) {
    if(!p1.equals(p2) && matrix[p1.x][p1.y] == matrix[p2.x][p2.y]) {
        if(p1.x == p2.x) {
            System.out.println("line x");
            if(checkLineX(p1.y, p2.y, p1.x)) {

                return new PointLine(p1, p2);
            }
        }

        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 PointLine(p1, p2);
            }
        }

        if(checkRectX(p1, p2)) {
            System.out.println("rect x");
            return new PointLine(p1, p2);
        }

        if(checkRectY(p1, p2)) {
            System.out.println("rect y");
            return new PointLine(p1, p2);
        }

        if(checkMoreLineX(p1, p2, 1)) {
            System.out.println("more right");
            return new PointLine(p1, p2);
        }
```

```
        if(checkMoreLineX(p1, p2, -1)) {
            System.out.println("more left");
            return new PointLine(p1, p2);
        }

        if(checkMoreLineY(p1, p2, 1)) {
            System.out.println("more down");
            return new PointLine(p1, p2);
        }

        if(checkMoreLineY(p1, p2, -1)) {
            System.out.println("more up");
            return new PointLine(p1, p2);
        }
    }
    return null;
}
```

Hiện tại:



- Giao diện em làm chưa được đẹp, với việc ăn 2 icon giống nhau chưa hiện line giữa chúng

- Mới chỉ hoàn thành cơ bản thuật toán sinh và ăn icon giống nhau, chưa có các màn chơi tiếp như thụt xuống, sang phải, sang trái

- Em dự định 2 tuần tới sẽ làm cho giao diện đẹp nhất, kẻ line ghi ăn icon và có thể tạo ra thêm các màn chơi mới