TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

KHOA ĐÀO TẠO QUỐC TẾ

BÁO CÁO BÀI TẬP

Môn học: Cơ sở dữ liệu

ĐỀ TÀI:

QUẢN LÝ QUÁN CAFE

Thành viên	: Trịnh Gia Khánh	- 212633937
	: Nguyễn Tiến Hiệu	- 212612454
Lớp	: CNTT V-A 1	
Giảng viên hướng dẫn	: Vũ Huấn	
Khóa:	: 62	

Hà Nội, 31-3-2023

Lời cảm ơn

Để có thể hoàn thành được đồ án Game Java này, chúng em xin bày tỏ lòng biết ơn sâu sắc tới thầy Vũ Xuân Huấn đã tận tình hướng dẫn, chỉ dạy trong suốt quá trình thực hiện đề tài.

Chúng em xin chân thành cảm ơn những người thân, bạn bè đã giúp đỡ trong quá trình thực hiện đề tài để có thể hoàn thành cơ sở dữ liệu đúng thời hạn và thành công tốt đẹp.

Suốt quãng thời gian qua, dù đã rất cố gắng, một phần cũng do kiến thức và kinh nghiệm chưa có đủ nên đồ án này khó tránh khỏi sai sót. Chúng em rất mong được nhận ý kiến đóng góp của Thầy, Cô để đồ án này được hoàn hảo hơn

Trân trọng cảm ơn!

Contents

Chương I: Tổng quan đề tài

I) Giới thiệu về trò chơi:

Đây là tựa game 2D cổ điển nổi tiếng có tên là Pacman. Nhiệm vụ của bạn là điều khiển nhân vật Pacman di chuyển và thu thập những chấm trắng nhất có thể, mỗi chấm trắng tương đương 1 điểm, đồng thời phải né sự săn đuổi của các con ma. Mỗi lần chơi bạn chỉ có thể chạm vào các con ma 3 lần, sau đó trò chơi sẽ kết thúc với điểm số bạn đạt được.

II) Mô tả đề tài

Xây dựng game Pacman có nhân vật chính cùng tên, một mê cung và các con ma thường và 1 ma đặc biệt có khả năng đi xuyên tường. Trong đó, người chơi sẽ điều khiển pac man di chuyển theo bản đồ có sẵn để có thể ăn được hết tất cả các chấm trên bản đồ nhằm qua màn mà không bị chạm phải các con ma. Các con ma có nhiệm vụ di chuyển tự do với nhiệm vụ ngăn chặn người chơi qua màn. Có 3 mạng cho pacman cho đến khi thua cuộc, mỗi lần bị ma chạm phải sẽ mất đi 1 mạng.

III) Cách chơi

- Sử dụng các phím mũi tên điều khiển nhân vật lên, xuống, trái, phải.
- Nhấn P để tạm dừng game.
  Nhấn P lần nữa để tiếp tục chơi.
  Nhấn Esc để chơi lại từ đầu.
- Nếu Pacman chết:
  Mỗi lần chết sẽ trừ đi 1 mạng.
  Hết 3 mạng sẽ GameOver.
  Người chơi cần phải nhập tên để lưu lại điểm.
- Độ khó của game sẽ tăng sau mỗi lần người chơi quan bàn bằng cách tăng tốc độ và số lượng của những con ma.

Chương II: Xây dựng đề tài

I) GamePanel

I.I) Import các thư viện cần thiết:

```
import java.awt.*;
import java.sql.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;

import javax.imageio.ImageIO;
import javax.print.DocFlavor.URL;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.SwingUtilities;
import javax.swing.Timer;

import pacman.SoundTrack;
```

I.II) Lớp GamePanel

a) Khai báo các dữ liệu cần thiết

```java
public class GamePanel extends JPanel implements ActionListener {
    Ghost g = new Ghost(this);
    public String playerName;
    private Dimension d;
    private final Font smallFont = new Font("Arial", Font.BOLD, 14);
    public boolean inGame = false;
    public boolean dying = false;
    public boolean isPaused = false;

    public final int BLOCK_SIZE = 24;
    public final int N_BLOCKS = 21;
    public final int SCREEN_SIZE = N_BLOCKS * BLOCK_SIZE ;

    private final int MAX_GHOSTS = 12;
    private final int PACMAN_SPEED = 6;
    private final int DELAY = 100;

    int menuX,menuY;
    int selectedOption;

    boolean mousePressed;


    private int lives, score;

    public int N_GHOSTS = 6;
    public int[] dx, dy;
    public int[] ghost_x, ghost_y, ghost_dx, ghost_dy, ghostSpeed;
    public int n_x, n_y, n_dx, n_dy, nSpeed;
    public int ndx, ndy;

    public Image heart, ghost,ghost2;
    private Image up, down, left, right;
```

Các thuộc tính của lớp GamePanel bao gồm:

- Ghost g = new Ghost(this) : tạo ra một thực thể mới của đối tượng Ghost và chuyển vào this dưới dạng tham số.
- playerName: tên của người chơi được nhập sau khi trò chơi kết thúc.
- private Dimension d: khai báo biến Dimension có tên d.
- Font(“Arial”, Font.BOLD, 14) : Khai báo font chữ Arial đậm có kích thước là 14.
- inGame: Biến boolean cho biết trò chơi có đang chạy hay không, được gán mặc định là false tương ứng với trò chơi đang dừng.
- dying: Biến boolean kiểm tra việc mất mạng của pacman.
- isPaused: Biến boolean kiểm tra việc tạm dừng của trò chơi.
- BLOCK_SIZE: Kích thước của 1 đơn vị block có giá trị là 24.
- N_BLOCKS: số lượng block theo chiều ngang và dọc, có giá trị là 21 blocks.
- SCREEN_SIZE: kích thước tổng thể của bản đồ tựa game là N_BLOCKS * N_BLOCKS = 441.

- MAX_GHOSTS: Số lượng ma tối đa có thể có trên màn hình.
- PACMAN_SPEED: tốc độ mặc định của pacman.
- DELAY: tốc độ chạy các khung hình của trò chơi.
- menuX:
- menuY:
- selectedOption:
- mousePressed:
- lives: số mạng sống của pacman.
- score: số điểm của người chơi.
- N_GHOSTS: số lượng ma ở thời điểm khởi đầu.
- dx, dy, ghost_x, ghost_y, ghost_dx, ghost_dy: xác định vị trí của các con ma.
- ghostSpeed: tốc độ của ma.
- heart, ghost, ghost2: hình ảnh số mạng của pacman và hình ảnh các con ma.
- up, down, left, right: hình ảnh của pacman khi di chuyển theo các hướng khác nhau.

Vị trí của pacman

```
//pacman position
public int pacman_x, pacman_y, pacmand_x, pacmand_y;
public int req_dx, req_dy;
```

- pacman_x, pacman_y: lưu trữ vị trí x, y hình ảnh(sprites) của pacman.
- pacmand_x, pacmand_y: dữ liệu thay đổi theo chiều dọc và ngang của pacman.
- req_dx, req_dy: hướng di chuyển được yêu cầu bởi người chơi.

I.III) Map game:

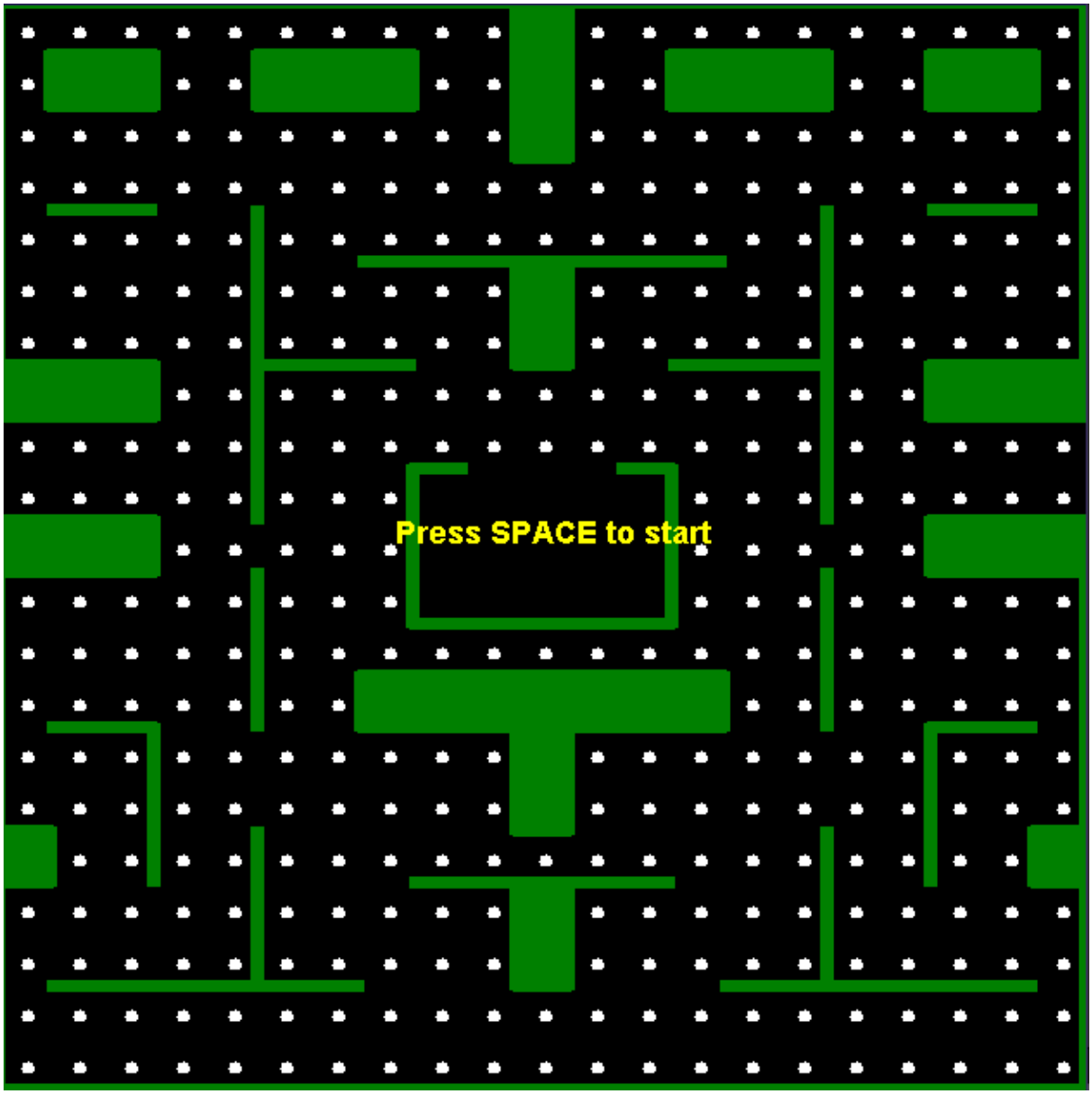
```
//map
private final short levelData[] = {
        19, 26, 26, 18, 18, 26, 26, 26, 18, 22, 0,  19, 18, 26, 26, 26, 18, 18, 26, 26, 22,
        21, 0,  0,  17, 20, 0,  0,  0,  17, 20, 0,  17, 20, 0,  0,  0,  17, 20, 0,  0,  21,
        17, 18, 18, 16, 16, 18, 18, 18, 16, 20, 0,  17, 16, 18, 18, 18, 16, 16, 18, 18, 20,
        17, 24, 24, 16, 16, 16, 16, 16, 16, 16, 18, 16, 16, 16, 16, 16, 16, 16, 24, 24, 20,
        17, 18, 18, 16, 20, 17, 16, 24, 24, 24, 24, 24, 24, 24, 16, 20, 17, 16, 18, 18, 20,
        17, 16, 16, 16, 20, 17, 16, 18, 18, 22, 0,  19, 18, 18, 16, 20, 17, 16, 16, 16, 20,
        25, 24, 24, 16, 20, 25, 24, 24, 16, 20, 0,  17, 16, 24, 24, 28, 17, 16, 24, 24, 28,
        0,  0,  0,  17, 20, 19, 18, 18, 16, 16, 18, 16 , 16, 18, 18, 22, 17, 20, 0, 0,  0,
        18, 18, 18, 16, 20, 17, 16, 16, 24, 16, 16, 16, 24, 16, 16, 20, 17, 16, 18, 18, 18,
        24, 24, 24, 16, 20, 17, 16, 20, 3,  32, 32, 32, 6,  17, 16, 20, 17, 16, 24, 24, 24,
        0,  0,  0,  17, 16, 16, 16, 20, 1,  32, 32, 32, 4,  17, 16, 16, 16, 20, 0,  0,  0,
        19, 18, 18, 16, 20, 17, 16, 20, 9,  8,  8,  8,  12, 17, 16, 20, 17, 16, 18, 18, 22,
        17, 16, 16, 16, 20, 17, 16, 24, 26, 26, 26, 26, 26, 24, 16, 20, 17, 16, 16, 16, 20,
        17, 24, 24, 16, 20, 17, 20, 0,  0,  0,  0,  0,  0,  0,  17, 20, 17, 16, 24, 24, 20,
        17, 18, 22, 17, 16, 16, 16, 18, 18, 22, 0,  19, 18, 18, 16, 16, 16, 20, 19, 18, 20,
        25, 16, 20, 17, 16, 16, 16, 16, 16, 20, 0,  17, 16, 16, 16, 16, 16, 20, 17, 16, 28,
        0,  17, 20, 17, 20, 17, 16, 16, 24, 24, 26, 24, 24, 16, 16, 20, 17, 20, 17, 20, 0,
        19, 16, 16, 16, 20, 17, 16, 16, 18, 22, 0,  19, 18, 16, 16, 20, 17, 16, 16, 16, 22,
        17, 24, 24, 24, 28, 25, 24, 16, 16, 20, 0,  17, 16, 16, 24, 28, 25, 24, 24, 24, 20,
        17, 18, 18, 18, 18, 18, 18, 16, 16, 16, 18, 16, 16, 16, 18, 18, 18, 18, 18, 18, 20,
        25, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 28

};
```

Giải thích map:

- 0: 1 block.
- 1: viền trái.
- 2: viền trên.
- 4: viền phải.
- 8: viền dưới.
- 16: 1 dot.

Map sau khi được chạy:



Khai báo liên quan đến tốc độ của trò chơi:

```
//speed
private final int validSpeeds[] = {1, 2, 3, 4, 6, 7, 8};
private final int maxSpeed = 8;
private int currentSpeed = 4;
public short[] screenData;
private Timer timer;
```

- validSpeeds: các giá trị tốc độ cho phép.
- maxSpeed: tốc độ tối đa.
- currentSpeed: tốc độ hiện tại của pacman.
- screenData: Dùng để lấy toàn bộ dữ liệu từ levelData để vẽ lại trò chơi.

timer: Load animation.

Hàm tạo GamePanel: gọi các function khác nhau để chạy trò chơi

```
public GamePanel() {

    loadImages();
    initVariables();
    addKeyListener(new TAdapter());
    setFocusable(true);
    initGame();
}
```

- loadImages(): Tải các hình ảnh(sprites) vào trò chơi.
- initVariables: khởi tạo các biến của trò chơi.
- addKeyListener: Chức năng điều khiển của trò chơi.
- setFocusable: cho phép tập trung vào 1 khung hình sau khi được hiển thị.
- initGame(): dùng để bắt đầu trò chơi

I.IV) Các function để chạy nhạc

```
public void playMusic(int i) {
    sound.setFile(i);
    sound.play();
    sound.loop();
}
public void stopMusic() {
    sound.stop();
}
public void playSE(int i) {
    sound.setFile(i);
    sound.play();
}
```

I.V) loadImage

```
public void loadImages() {
    down = new ImageIcon(getClass().getResource("/images/down.gif")).getImage();
    up = new ImageIcon(getClass().getResource("/images/up.gif")).getImage();
    left = new ImageIcon(getClass().getResource("/images/left.gif")).getImage();
    right = new ImageIcon(getClass().getResource("/images/right.gif")).getImage();
    ghost = new ImageIcon(getClass().getResource("/images/ghost2.gif")).getImage();
    ghost2 = new ImageIcon(getClass().getResource("/images/ghost.gif")).getImage();
    heart = new ImageIcon(getClass().getResource("/images/heart.png")).getImage();
}
```

Tải lên các hình ảnh của trò chơi

I.VI) initVariables

```
    private void initVariables() {
     screenData = new short[N_BLOCKS * N_BLOCKS];
     d = new Dimension(505, 600);
     ghost_x = new int[MAX_GHOSTS];
     ghost_dx = new int[MAX_GHOSTS];
     ghost_y = new int[MAX_GHOSTS];
     ghost_dy = new int[MAX_GHOSTS];
     ghostSpeed = new int[MAX_GHOSTS];
     dx = new int[4];
     dy = new int[4];
     timer = new Timer(DELAY, this);
     timer.restart();
 }
```

- Phương thức initVariables() khởi tạo các biến và đối tượng cần thiết cho trò chơi.

- screenData: một mảng chứa các dữ liệu liên quan đến cấu trúc của mê cung được sử dụng để hiển thị mê cung lên màn hình.

- d: một đối tượng kiểu Dimension để lưu kích thước của cửa sổ trò chơi.

- ghost_x, ghost_dx, ghost_y, ghost_dy, ghostSpeed: một số mảng và biến để lưu trạng thái của các ma trong trò chơi, bao gồm tọa độ, tốc độ và hướng di chuyển.

- dx và dy: mảng chứa 4 giá trị để xác định hướng di chuyển của Pacman.

- timer: một đối tượng Timer để định kỳ cập nhật trạng thái của trò chơi.

I.VII) pauseMenu

```
private void drawPauseMenu(Graphics2D g2d) {
        Font menuFont = new Font("Arial", Font.BOLD, 24);
        g2d.setFont(menuFont);

        String pauseMsg = "Game Paused";
        int msgWidth = g2d.getFontMetrics().stringWidth(pauseMsg);
        int centerX = SCREEN_SIZE / 2;
        int centerY = SCREEN_SIZE / 2 - 50;
        g2d.setColor(Color.white);
        g2d.fillRect(0, 0, SCREEN_SIZE, SCREEN_SIZE);
        g2d.setColor(Color.black);
        g2d.fillRect(10, 10, SCREEN_SIZE - 20, SCREEN_SIZE-20);
        g2d.setColor(Color.yellow);
        g2d.drawString(pauseMsg, centerX - msgWidth / 2, centerY);

        Font optionFont = new Font("Arial", Font.PLAIN, 20);
        g2d.setFont(optionFont);

        String continueMsg = "Press P to continue";
        String quitMsg = "press enter to give up";
        int continueWidth = g2d.getFontMetrics().stringWidth(continueMsg);
        int quitWidth = g2d.getFontMetrics().stringWidth(quitMsg);
        int optionY = centerY + 50;
        g2d.setColor(selectedOption == 0 ? Color.yellow : Color.white);
        g2d.drawString(continueMsg, centerX - continueWidth / 2, optionY);
        g2d.setColor(selectedOption == 1 ? Color.yellow : Color.white);
        g2d.drawString(quitMsg, centerX - quitWidth / 2, optionY + 30);
    }
```

drawPauseMenu được sử dụng để vẽ menu tạm dừng game. Bên trong phương thức, ta thấy các bước sau:

- Tạo font cho tiêu đề menu **Game Paused** với kiểu **Arial**, độ đậm **BOLD** và kích cỡ **24**.
- Tính toán độ rộng của tiêu đề bằng cách sử dụng **g2d.getFontMetrics().stringWidth(pauseMsg).**
- Tính toán tọa độ centerX và centerY của tiêu đề ở giữa màn hình.
- Sử dụng **g2d.setColor** để đặt màu nền là màu trắng.
- Sử dụng **g2d.fillRect** để vẽ một hình chữ nhật đen bao quanh nội dung menu, bên trong hình chữ nhật trắng ở bước 4.
- Sử dụng **g2d.setColor** để đặt màu cho tiêu đề là màu vàng.
- Sử dụng **g2d.drawString** để vẽ tiêu đề tạm dừng game ở tọa độ tính toán ở bước 3.
- Tạo font cho các lựa chọn menu với kiểu **Arial**, độ đậm **PLAIN** và kích cỡ **20**.
- Tính toán độ rộng của các lựa chọn với **g2d.getFontMetrics().stringWidth(continueMsg) và g2d.getFontMetrics().stringWidth(quitMsg).**
- Tính toán tọa độ **optionY** của lựa chọn đầu tiên.
- Sử dụng **g2d.setColor** để đặt màu của lựa chọn đang được chọn là màu vàng và màu của các lựa chọn khác là màu trắng.
- Sử dụng **g2d.drawString** để vẽ các lựa chọn trên màn hình. Lựa chọn "Press P to continue" được vẽ ở **centerX - continueWidth / 2** và **optionY**, lựa chọn "press enter to give up" được vẽ ở **centerX - quitWidth / 2** và **optionY + 30**.

- Kết thúc phương thức.

I.VII) playGame

```
private void playGame(Graphics2D g2d) {

    if (isPaused) {
        drawPauseMenu(g2d);
    } else {
        if (dying) {
            playSE(2);
            death(g2d);

        } else {
            movePacman();
            drawPacman(g2d);
            g.moveGhosts(g2d);
            g.moveGhost2(g2d);
            checkMaze();
        }
    }
}
```

- Phương thức playGame được sử dụng để chơi trò chơi. Nó được gọi liên tục trong vòng lặp chính của trò chơi để cập nhật các trạng thái mới nhất của trò chơi và vẽ chúng lên màn hình.

- Đầu tiên, phương thức kiểm tra xem trò chơi có đang bị tạm dừng không. Nếu có, nó sẽ vẽ menu tạm dừng lên màn hình. Nếu không, phương thức sẽ kiểm tra xem Pacman có đang chết hay không. Nếu đang chết, nó sẽ phát âm thanh và gọi phương thức death để xử lý việc hiển thị hình ảnh của Pacman khi chết. Nếu không, nó sẽ di chuyển Pacman và các con ma, và kiểm tra xem Pacman có ăn được chấm điểm hay không. Sau đó, nó sẽ vẽ Pacman và các con ma lên màn hình.

I.IX) ConstructorcheckMaze

```
private void checkMaze() {
    int i, dotCount;
    dotCount = 0;

    // check dot
    boolean allDotsEaten = true;
    for (i = 0; i < N_BLOCKS * N_BLOCKS; i++) {
        if ((levelData[i] & 16) != 0) {
            if ((screenData[i] & 16) != 0) {
                allDotsEaten = false;
                break;
            }
        }
    }

    // reset dots
    if (allDotsEaten) {
        score+=100;

        initLevel();
        if(N_GHOSTS <= MAX_GHOSTS) {
            N_GHOSTS++;
        }
        for (i = 0; i < MAX_GHOSTS; i++) {
            if(ghostSpeed[i]<=8) {
                ghostSpeed[i] += 1.5; //tăng tốc ghost
            }else if(ghostSpeed[i]>8) {
                ghostSpeed[i] =ghostSpeed[i];
            }
        }
    }

}
```

Giải thích:

- Trước hết ta khai báo biến "i" và "dotCount" kiểu số nguyên và khởi tạo "dotCount" bằng 0.
- Tiếp theo ta kiểm tra việc ăn hết "dot" (chấm) trên màn hình. Trong vòng lặp "for" từ 0 đến "N_BLOCKS * N_BLOCKS" (tổng số ô vuông trong mê cung), nếu ô đó chứa dot và dot chưa bị ăn (được đánh dấu bằng bit thứ 5 của "levelData" và "screenData"), thì sẽ gán biến "allDotsEaten" bằng false và thoát khỏi vòng lặp. Nếu tất cả các dot đã bị ăn, thì biến "allDotsEaten" sẽ được giữ nguyên giá trị true.
- Nếu tất cả dot đã bị ăn, thì sẽ tăng điểm "score" thêm 100 điểm. Sau đó, sẽ khởi tạo màn chơi mới bằng phương thức "initLevel()". Nếu số lượng ghost "N_GHOSTS" còn nhỏ hơn hoặc bằng "MAX_GHOSTS", thì sẽ tăng số lượng ghost lên 1. Cuối cùng, trong vòng lặp "for", tăng tốc độ của tất cả ghost bằng 1.5 đơn vị, nhưng nếu tốc độ ghost đã lớn hơn 8 thì sẽ giữ nguyên tốc độ hiện tại của ghost đó.

I.X) Constructor death:

```
private void death(Graphics g2d)  {
    lives--;
    int freezeTimeInSeconds = 2;
    try {
       Thread.sleep(freezeTimeInSeconds * 1000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    if (lives == 0) {
        inGame = false;
        JOptionPane.showMessageDialog(null, "Game Over");
        playerName = JOptionPane.showInputDialog(this, "Nhập tên của bạn:");
        // Kiểm tra nếu người dùng không nhập tên hoặc nhập tên rỗng
        while (playerName == null || playerName.trim().equals("")) {
            JOptionPane.showMessageDialog(this, "Bạn phải nhập tên để lưu điểm!", "Lỗi", JOptionPane.ERROR_MESSAGE);
            playerName = JOptionPane.showInputDialog(this, "Nhập tên của bạn:");
        }
        try {
            Class.forName(MySqlDemo.DRIVER_CLASS);
            Connection conn = DriverManager.getConnection(MySqlDemo.DB_URL, MySqlDemo.USER, MySqlDemo.PASS);
            String sql = "INSERT INTO highscore (name, score) VALUES (?, ?)";
            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, playerName);
            pstmt.setInt(2, score);

            int rowsInserted = pstmt.executeUpdate();
            if (rowsInserted > 0) {
                System.out.println("Hey, it is work");
            }
            pstmt.close();
            conn.close();
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
```

```
        System.out.println(playerName+","+score);

        JFrame gameFrame = (JFrame) SwingUtilities.getWindowAncestor(this);
        gameFrame.setVisible(false);
        new GameMenu().setVisible(true);
    }
    g.continueLevel();

}
```

- Đầu tiên mỗi khi lives giảm 1 đơn vị.
- Khai báo và khởi tạo biến "**freezeTimeInSeconds**" bằng **2** giây.
- Sử dụng phương thức "**Thread.sleep()**" để đợi một thời gian bằng "**freezeTimeInSeconds**" giây. Điều này giúp người chơi dễ dàng định hướng khi bị "**death**" (mất một lượt chơi).
- Nếu số lượt chơi "**lives**" bằng **0**, thì đánh dấu trò chơi kết thúc ("**inGame = false**"). Hiển thị hộp thoại "**Game Over**" thông báo cho người chơi.
- Yêu cầu người chơi nhập tên của mình và lưu điểm số của người chơi vào cơ sở dữ liệu. Nếu người chơi không nhập tên hoặc nhập tên rỗng, sẽ hiển thị thông báo lỗi yêu cầu nhập lại.
- Sau khi lưu điểm số, ẩn cửa sổ chơi game và mở cửa sổ trang chủ của trò chơi.
- Nếu số lượt chơi vẫn còn, tiếp tục chơi từ màn chơi hiện tại bằng cách gọi phương thức **"continueLevel()"** của đối tượng "**g**".

I.XI) Constructor movePacman

```
private void movePacman() {
    int pos;
    short ch;
    if (pacman_x % BLOCK_SIZE == 0 && pacman_y % BLOCK_SIZE == 0) {
        pos = pacman_x / BLOCK_SIZE + N_BLOCKS * (int) (pacman_y / BLOCK_SIZE);
        ch = screenData[pos];

        // Check for food
        if ((ch & 16) != 0) {
            screenData[pos] = (short) (ch & 15);
            score++;
            playSE(1);
        }

        // Check for walls and update direction if needed
        if (req_dx != 0 || req_dy != 0) {
            if (!((req_dx == -1 && req_dy == 0 && (ch & 1) != 0)
                    || (req_dx == 1 && req_dy == 0 && (ch & 4) != 0)
                    || (req_dx == 0 && req_dy == -1 && (ch & 2) != 0)
                    || (req_dx == 0 && req_dy == 1 && (ch & 8) != 0))) {
                pacmand_x = req_dx;
                pacmand_y = req_dy;
            }
        }

        // Check for standstill
        if ((pacmand_x == -1 && pacmand_y == 0 && (ch & 1) != 0)
                || (pacmand_x == 1 && pacmand_y == 0 && (ch & 4) != 0)
                || (pacmand_x == 0 && pacmand_y == -1 && (ch & 2) != 0)
                || (pacmand_x == 0 && pacmand_y == 1 && (ch & 8) != 0)) {
            pacmand_x = 0;
            pacmand_y = 0;
        }
    }

    // Update Pacman's position
    pacman_x = pacman_x + PACMAN_SPEED * pacmand_x;
    pacman_y = pacman_y + PACMAN_SPEED * pacmand_y;

    // Check if Pacman is out of bounds and teleport to the other side if needed
    if (pacman_x < 0) {
        pacman_x = (N_BLOCKS - 1) * BLOCK_SIZE;
    } else if (pacman_x >= N_BLOCKS * BLOCK_SIZE) {
        pacman_x = 0;
    }
    if (pacman_y < 0) {
        pacman_y = (N_BLOCKS - 1) * BLOCK_SIZE;
    } else if (pacman_y >= N_BLOCKS * BLOCK_SIZE) {
        pacman_y = 0;
    }
}
```

- Đầu tiên hàm sẽ kiểm tra vị trí hiện tại của Pacman và xác định hướng di chuyển dựa trên các yêu cầu đầu vào từ người dùng (**req_dx** và **req_dy**). Nếu Pacman chạm vào thức ăn (chứa trong biến **ch**), hàm sẽ cập nhật điểm số và phát âm thanh đồng thời xóa đi bit chứa dot trên màn hình.

- Sau đó kiểm tra xem Pacman có đang đứng yên không, nếu đang đứng yên thì sẽ kiểm tra xem hướng đi mới của Pacman có gặp phải tường không. Nếu không gặp tường thì cập nhật lại hướng đi mới của Pacman.

- Trong đó, biến **req_dx** và **req_dy** lưu hướng đi mới được yêu cầu của Pacman. Nếu **req_dx** hoặc **req_dy** khác **0,** tức là hướng đi mới được yêu cầu, thì sẽ kiểm tra xem có gặp tường ở hướng đi mới đó không. Nếu không gặp tường, tức là Pacman có thể đi hướng mới được yêu cầu, thì sẽ cập nhật lại hướng đi của Pacman (**pacmand_x** và **pacmand_y**) với hướng mới đó.
- Ta sử dụng sử dụng toán tử logic || để kiểm tra xem hướng đi mới có khác 0 hay không. Nếu khác 0, sẽ sử dụng toán tử **&&** và các phép so sánh để kiểm tra xem hướng đi mới có gặp tường không. Nếu không gặp tường, sẽ cập nhật lại hướng đi của Pacman.

//update pacman's position

- Cập nhật vị trí mới của Pacman bằng cách thêm vào vị trí hiện tại của Pacman một giá trị bằng tốc độ di chuyển của Pacman theo hướng x và hướng y. Cụ thể hơn, nếu **pacmand_x** là **1** (đi về phía bên phải) thì **pacman_x** sẽ được cộng thêm **PACMAN_SPEE**D để di chuyển sang phải. Nếu **pacmand_x** là **-1** (đi về phía bên trái) thì **pacman_x** sẽ được trừ đi **PACMAN_SPEED** để di chuyển sang trái. Tương tự với hướng **y,** nếu **pacmand_y** là **1** (đi lên) thì **pacman_y** sẽ được cộng thêm **PACMAN_SPEED** để di chuyển lên, và nếu **pacmand_y** là **-1** (đi xuống) thì **pacman_y** sẽ được trừ đi **PACMAN_SPEED** để di chuyển xuống.

//check if pacman out of bound

- Đoạn code này kiểm tra xem Pacman có đang ở ngoài màn hình hoặc không. Nếu Pacman đã di chuyển hết màn hình thì sẽ được "**teleport**" đến phía bên kia màn hình để tiếp tục chơi.
- Cụ thể, nếu giá trị của **pacman_x** (vị trí theo trục hoành) nhỏ hơn **0** (tức là Pacman đi quá biên màn hình bên trái), thì vị trí của Pacman sẽ được đặt lại ở phía bên phải của màn hình (cách biên phải 1 ô). Tương tự, nếu **pacman_x** vượt quá biên phải của màn hình (tức là lớn hơn hoặc bằng **N_BLOCKS * BLOCK_SIZE**), thì Pacman sẽ được đặt lại ở phía bên trái của màn hình (cách biên trái 1 ô).
- Tương tự, nếu giá trị của **pacman_y** (vị trí theo trục tung) nhỏ hơn **0** (tức là Pacman đi quá biên màn hình phía trên), thì vị trí của Pacman sẽ được đặt lại ở phía dưới của màn hình (cách biên dưới 1 ô). Ngược lại, nếu **pacman_y** vượt quá biên dưới của màn hình (tức là lớn hơn hoặc bằng **N_BLOCKS * BLOCK_SIZE**), thì Pacman sẽ được đặt lại ở phía trên của màn hình (cách biên trên 1 ô).

I.XII) ConstructordrawPacman

```
private void drawPacman(Graphics2D g2d) {

    if (req_dx == -1) {
        g2d.drawImage(left, pacman_x + 1, pacman_y + 1, this);
    } else if (req_dx == 1) {
        g2d.drawImage(right, pacman_x + 1, pacman_y + 1, this);
    } else if (req_dy == -1) {
        g2d.drawImage(up, pacman_x + 1, pacman_y + 1, this);
    } else {
        g2d.drawImage(down, pacman_x + 1, pacman_y + 1, this);
    }
}
```

- Phương thức drawPacman được sử dụng để vẽ Pacman trên màn hình. Hình ảnh Pacman được chọn dựa trên hướng đi của nó được xác định bởi giá trị của req_dx và req_dy.

- Nếu **req_dx** bằng **-1**, Pacman đang đi sang trái, phương thức drawImage sẽ vẽ hình ảnh của Pacman hướng sang trái (left) tại vị trí (**pacman_x + 1, pacman_y + 1**). Tương tự, nếu **req_dx** bằng **1**, Pacman đang đi sang phải, hình ảnh của Pacman hướng sang phải (right) sẽ được vẽ. Nếu **req_dy** bằng **-1**, Pacman đang đi lên, hình ảnh của Pacman hướng lên (up) sẽ được vẽ. Nếu không, Pacman đang đi xuống, hình ảnh của Pacman hướng xuống (down) sẽ được vẽ.

I.XIII) Constructor drawMaze

```
private void drawMaze(Graphics2D g2d) {

    short i = 0;
    int x, y;

    for (y = 0; y < SCREEN_SIZE; y += BLOCK_SIZE) {
        for (x = 0; x < SCREEN_SIZE; x += BLOCK_SIZE) {

            g2d.setColor(new Color(0, 128, 0));
            g2d.setStroke(new BasicStroke(5));

            if ((levelData[i] == 0)) {
                g2d.fillRect(x, y, BLOCK_SIZE, BLOCK_SIZE);
             }

            if ((screenData[i] & 1) != 0) {
                g2d.drawLine(x, y, x, y + BLOCK_SIZE - 1);
            }

            if ((screenData[i] & 2) != 0) {
                g2d.drawLine(x, y, x + BLOCK_SIZE - 1, y);
            }

            if ((screenData[i] & 4) != 0) {
                g2d.drawLine(x + BLOCK_SIZE - 1, y, x + BLOCK_SIZE - 1,
                        y + BLOCK_SIZE - 1);
            }
```

```
            if ((screenData[i] & 8) != 0) {
                g2d.drawLine(x, y + BLOCK_SIZE - 1, x + BLOCK_SIZE - 1,
                        y + BLOCK_SIZE - 1);
            }

            if ((screenData[i] & 16) != 0) {
                g2d.setColor(new Color(255,255,255));
                g2d.fillOval(x + 10, y + 10, 6, 6);
            }

            if ((screenData[i] & 32) != 0) {
                  g2d.setColor(new Color(0,0,0));
            }
             i++;
        }
    }
}
```

- Biến **i** đại diện cho vị trí của một phần tử trong mảng **levelData** và **screenData**.
- Biến x và y đại diện cho tọa độ của các khối trong mê cung.
- Vòng lặp đầu tiên (y) chạy qua các hàng trong mê cung và vòng lặp thứ hai (x) chạy qua các cột trong mỗi hàng.
- Đoạn **g2d.setColor(new Color(0, 128, 0)); và g2d.setStroke(new BasicStroke(5))**; được sử dụng để đặt màu và độ dày cho các nét vẽ sau đó.

- Nếu **((levelData[i] == 0))** kiểm tra xem giá trị của phần tử trong levelData có bằng 0 hay không, nếu có thì vẽ một hình chữ nhật bằng cách sử dụng **g2d.fillRect(x, y, BLOCK_SIZE, BLOCK_SIZE);.**
- Nếu **((screenData[i] & 1) != 0),** và **((screenData[i] & 2) != 0), ((screenData[i] & 4) != 0)** và **((screenData[i] & 8) != 0)** kiểm tra các bit đầu tiên của giá trị trong screenData. Nếu bit thứ i bằng 1 thì vẽ các đường thẳng để tạo thành các tường của mê cung. Cụ thể, nếu bit thứ 1 bằng 1 thì vẽ đường thẳng từ điểm (x, y) đến **(x, y + BLOCK_SIZE - 1),** nếu bit thứ 2 bằng 1 thì vẽ đường thẳng từ điểm (x, y) đến **(x + BLOCK_SIZE - 1, y)** và tương tự cho các bit khác.
- Nếu **((screenData[i] & 16) != 0)** kiểm tra bit thứ 5 của giá trị trong **screenData**. Nếu bit này bằng 1, thì vẽ một hình tròn trắng nhỏ ở giữa khối để đại diện cho người chơi.
- Nếu **((screenData[i] & 32) != 0)** kiểm tra bit thứ 6 của giá trị trong **screenData**. Nếu bit này bằng 1, thì không vẽ gì cả.

I.XIV) initGame

```
private void initGame() {
    lives = 3;
    score = 0;
    initLevel();
    N_GHOSTS = 4;
    currentSpeed = 3;
    for(int i=0;i<MAX_GHOSTS;i++) {
        g.spawnGhost(i, 11, 11);
    }
    g.spawnGhost2(11, 11);

}
```

- Khai báo số mạng khởi đầu là 3, và số điểm là 0;
- Đồng thời khởi chạy initLevel().
- Khởi tạo số ma là 4.
- Và tốc độ hiện tại là 3.
- Vòng lặp dùng để spaw các con ma tại vị trí cố định có x = 11, y = 11 kể cả khi pacman chết và vị trí ma được reset.

I.XV) initLevel:

```
private void initLevel() {

    int i;
    for (i = 0; i < N_BLOCKS * N_BLOCKS; i++) {
        screenData[i] = levelData[i];
    }

    g.continueLevel();
}
```

Sử dụng vòng lặp for để copy toàn bộ dãy levelData vào screenData để có thể khởi tạo level

o) paintComponent

```
public void paintComponent(Graphics g) {
    super.paintComponent(g);

    Graphics2D g2d = (Graphics2D) g;

    g2d.setColor(Color.black);
    g2d.fillRect(0, 0, d.width, d.height);

    drawMaze(g2d);
    drawScore(g2d);

    if (inGame) {
        playGame(g2d);
    } else {

        showIntroScreen(g2d);
    }

    Toolkit.getDefaultToolkit().sync();
    g2d.dispose();
}
```

paintComponent() được gọi mỗi khi cần vẽ lại giao diện. Trong phương thức này, đầu tiên chúng ta lấy đối tượng Graphics2D từ đối tượng Graphics được truyền vào, và thiết lập màu nền đen bằng phương thức **setColor()** và vẽ hình chữ nhật với kích thước bằng kích thước của JPanel bằng phương thức **fillRect().**

Sau đó, chúng ta gọi phương thức **drawMaze()** để vẽ mê cung, phương thức **drawScore()** để hiện điểm số và cuối cùng kiểm tra xem trò chơi đang chạy hay không. Nếu đang chạy, thì chúng ta gọi phương thức **playGame()** để xử lý các thao tác trong trò chơi, ngược lại chúng ta gọi phương thức **showIntroScreen()** để hiển thị màn hình giới thiệu.

Cuối cùng, chúng ta gọi phương thức **sync()** của đối tượng **Toolkit** để đảm bảo rằng bộ đệm đồ họa của thiết bị được đồng bộ hóa, và giải phóng đối tượng Graphics2D bằng phương thức **dispose().**

I.XVI) control

```
//controls
class TAdapter extends KeyAdapter {

    @Override
    public void keyPressed(KeyEvent e) {

        int key = e.getKeyCode();
        if (key == KeyEvent.VK_P) { // Kiểm tra phím P
            isPaused = !isPaused; // Đảo ngược trạng thái tạm dừng
            if (isPaused) {
                stopMusic(); // Tạm dừng nhạc nền khi tạm dừng game
                timer.stop();
            } else {
                 // Chơi lại nhạc nền khi tiếp tục game
            }
        }
        if (inGame) {
            if (key == KeyEvent.VK_LEFT) {
                req_dx = -1;
                req_dy = 0;
            } else if (key == KeyEvent.VK_RIGHT) {
                req_dx = 1;
                req_dy = 0;
            } else if (key == KeyEvent.VK_UP) {
                req_dx = 0;
                req_dy = -1;
            } else if (key == KeyEvent.VK_DOWN) {
                req_dx = 0;
                req_dy = 1;
            } else if (key == KeyEvent.VK_ESCAPE && timer.isRunning()) {
                inGame = false;
            }
        } else {
```

Phương thức TAdapter được sử dụng để xử lý sự kiện nhấn phím. Khi một phím được nhấn, phương thức keyPressed được gọi và sự kiện nhấn phím được truyền vào phương thức.

Nếu phím P được nhấn, biến isPaused sẽ được đảo ngược trạng thái tạm dừng. Nếu trạng thái là tạm dừng, nhạc nền sẽ được tạm dừng và đồng hồ đếm thời gian timer sẽ dừng lại. Nếu trạng thái là chơi game, phím mũi tên sẽ được sử dụng để điều khiển pacman. Nếu phím ESCAPE được nhấn, trò chơi sẽ dừng lại. Nếu trò chơi đã kết thúc, phím SPACE sẽ được sử dụng để bắt đầu một trò chơi mới.

Nếu biến isPaused là true, các phím mũi tên sẽ được sử dụng để di chuyển giữa các tùy chọn trên menu tạm dừng. Nếu phím ENTER được nhấn và tùy chọn được

chọn là 0, trạng thái tạm dừng sẽ được đổi lại thành false và trò chơi sẽ tiếp tục. Nếu tùy chọn là 1, trò chơi sẽ kết thúc.

II) Class gameFrame

```
package pacman;

import javax.swing.*;

public class GameFrame extends JFrame{

    public GameFrame() {

        add(new GamePanel());
    }
    public static void main(String[] args) {
        GameFrame pac = new GameFrame();
        pac.setVisible(true);
        pac.setTitle("Pacman");
        pac.setSize(518,580);
        pac.setDefaultCloseOperation(EXIT_ON_CLOSE);
        pac.setLocationRelativeTo(null);

    }

}
```

Class này tạo ra một cửa sổ JFrame cho game Pacman.

Nó có một constructor không tham số, nó thêm một đối tượng GamePanel vào JFrame bằng phương thức add().

Phương thức main() tạo một đối tượng GameFrame mới, thiết lập các thuộc tính của cửa sổ, đặt tiêu đề, kích thước, vị trí và đóng ứng dụng khi cửa sổ được đóng.

III) GameMenu.java:

```
// Thiết lập cấu hình của JFrame
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setTitle("Pacman");
setSize(400, 600);
setLocationRelativeTo(null);
setResizable(false);
```

Đây là một constructor của lớp GameMenu trong một trò chơi PacMan. Constructor này được gọi khi đối tượng của lớp GameMenu được khởi tạo. Constructor này được sử dụng để thiết lập cấu hình cho JFrame và tạo các thành phần giao diện cho trò chơi.

Đầu tiên, constructor sử dụng các phương thức của JFrame để thiết lập các thuộc tính cơ bản của cửa sổ trò chơi như setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE) để đóng ứng dụng khi người dùng thoát khỏi cửa sổ, setTitle("PacMan") để đặt tiêu đề cửa sổ là "PacMan", setSize(400, 600) để đặt kích thước của cửa sổ và setLocationRelativeTo(null) để đặt vị trí của cửa sổ ở giữa màn hình. Sau đó, constructor sử dụng setResizable(false) để ngăn chặn người dùng thay đổi kích thước cửa sổ.

```
// Tạo các thành phần giao diện
gamePanel = new GamePanel();
gamePanel.playSE(0);
startButton = new JButton("Bắt đầu chơi");
startButton.setMaximumSize(new Dimension(300, 60)); // cao 50, rộng 300
startButton.setAlignmentX(Component.CENTER_ALIGNMENT); // Căn giữa theo trục X
startButton.addActionListener(this);

quitButton = new JButton("Thoát");
quitButton.setMaximumSize(new Dimension(300, 60)); // cao 50, rộng 300
quitButton.setAlignmentX(Component.CENTER_ALIGNMENT); // Căn giữa theo trục X
quitButton.addActionListener(this);

hscoreButton = new JButton("Điểm cao nhất");
hscoreButton.setMaximumSize(new Dimension(300, 60)); // cao 50, rộng 300
hscoreButton.setAlignmentX(Component.CENTER_ALIGNMENT); // Căn giữa theo trục X
hscoreButton.addActionListener(this);
```

Constructor tạo các thành phần giao diện cho trò chơi. Đầu tiên, nó tạo một đối tượng GamePanel làm nền cho trò chơi và chơi âm thanh bằng phương thức playSE(0) của GamePanel. Sau đó, nó tạo các nút bằng cách sử dụng JButton và đặt các thuộc tính như setMaximumSize(new Dimension(300, 60)) để đặt kích thước của nút là 300 x 60, setAlignmentX(Component.CENTER_ALIGNMENT) để căn giữa theo trục X và addActionListener(this) để đăng ký bộ xử lý sự kiện cho nút.

```
        // JPanel chứa giao diện
        JPanel panel = new JPanel();
        panel = new JPanel() {
            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                Image img = null;
                img = setIMG("/images/images.jpg");
                g.drawImage(img, 0, 0, getWidth(), getHeight(), this);
            }
        };

        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
        panel.add(Box.createRigidArea(new Dimension(100, 300)));
        panel.add(startButton);
        panel.add(Box.createRigidArea(new Dimension(100, 10)));
        panel.add(quitButton);
        panel.add(Box.createRigidArea(new Dimension(100, 10)));
        panel.add(hscoreButton);
        panel.add(Box.createVerticalGlue());

        // Thêm JPanel vào JFrame
        add(panel);

        // Hiển thị JFrame
        setVisible(true);
    }
```

Constructor tạo một đối tượng JPanel để chứa các thành phần giao diện. Đối tượng này được thiết lập với layout là BoxLayout theo chiều dọc và các thành phần giao diện được thêm vào đối tượng này sử dụng các phương thức add(). Ngoài ra, constructor còn sử dụng Box.createRigidArea(new Dimension(100, 300)) để thêm một khoảng cách cứng giữa nút "Bắt đầu chơi" và đầu cửa sổ, và sử dụng Box.createVerticalGlue() để đặt nút "Điểm cao nhất" ở đáy cửa sổ.

Cuối cùng, constructor sử dụng add(panel) để thêm đối tượng JPanel chứa các thành phần giao diện vào JFrame và sử dụng setVisible(true) để hiển thị cửa sổ.

IV) SoundTrack.java:

```java
import java.net.URL;

public class SoundTrack {
    Clip clip;
    URL soundURL[] = new URL[40];

    public SoundTrack() {
        soundURL[0] = getClass().getResource("/SoundTrack/PacManTheme.wav");
        soundURL[1] = getClass().getResource("/SoundTrack/waka.wav");
        soundURL[2] = getClass().getResource("/SoundTrack/gameOver.wav");
    }

    public void setFile(int i) {
        try {
            AudioInputStream aud = AudioSystem.getAudioInputStream(soundURL[i]);
            clip = AudioSystem.getClip();
            clip.open(aud);
        }catch(Exception e) {

        }
    }
    public void play() {
        clip.start();

    }
    public void loop() {
        clip.loop(clip.LOOP_CONTINUOUSLY);
    }
    public void stop() {
        if (clip != null) {
            clip.stop();
        }
    }

}
```

lớp SoundTrack để quản lý âm thanh trong trò chơi Pacman. Lớp này sử dụng thư viện javax.sound.sampled để tải và chơi các tệp âm thanh từ các tài nguyên trong project.

Lớp SoundTrack có các thuộc tính và phương thức sau:

- clip: Đối tượng Clip được sử dụng để chơi âm thanh.
- soundURL: Một mảng chứa các URL tới các tệp âm thanh được sử dụng trong trò chơi.
- SoundTrack(): Phương thức khởi tạo lớp SoundTrack. Trong phương thức này, các URL tới các tệp âm thanh được lưu trữ trong mảng soundURL.
- setFile(int i): Phương thức để chọn tệp âm thanh để phát. Truyền vào một chỉ số i tương ứng với URL trong mảng soundURL. Từ URL này, phương thức sử dụng AudioSystem.getAudioInputStream để tải tệp âm thanh và sử dụng AudioSystem.getClip để tạo đối tượng Clip để phát âm thanh.
- play(): Phương thức để bắt đầu phát âm thanh.
- loop(): Phương thức để phát âm thanh lặp lại vô hạn số lần.
- stop(): Phương thức để dừng phát âm thanh nếu đang phát.

V) MySQLDemo.java:

Nơi chứa phương thức để kết nối với một cơ sở dữ liệu SQL Server và truy xuất dữ liệu từ bảng highscore.

```java
static final String DRIVER_CLASS = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
static final String DB_URL = "jdbc:sqlserver://localhost:1433;DatabaseName=gamescore;encrypt=true;trustServerCertificate=true;";
static final String USER = "sa";
static final String PASS = "123123";
```

Các biến định nghĩa:

- DRIVER_CLASS: định nghĩa tên lớp trình điều khiển JDBC cho SQL Server.
- DB_URL: định nghĩa đường dẫn đến cơ sở dữ liệu và thông tin kết nối.
- USER: định nghĩa tên người dùng để kết nối đến cơ sở dữ liệu.
- PASS: định nghĩa mật khẩu người dùng để kết nối đến cơ sở dữ liệu.

Các bước để truy xuất dữ liệu:

B1: Tạo một đối tượng Connection để kết nối đến cơ sở dữ liệu.

B2: Tạo một đối tượng Statement để thực thi câu truy vấn SQL.

B3: Thực thi câu truy vấn và lưu kết quả vào đối tượng ResultSet.

B4 Sử dụng phương thức next() của đối tượng ResultSet để lấy từng dòng dữ liệu một cho đến khi hết dữ liệu.

B5: Sử dụng các phương thức getXXX() của đối tượng ResultSet để lấy giá trị của từng cột trong dòng dữ liệu.

B6: Đóng đối tượng ResultSet, Statement và Connection để giải phóng tài nguyên.

```
public static void main(String[] args) throws SQLException, ClassNotFoundException {
        Connection conn = null;
        Statement stmt = null;
        try {
            Class.forName(DRIVER_CLASS);
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT name, score FROM highscore");

            while (rs.next()) {
                String name = rs.getString("name");
                int score = rs.getInt("score");
                System.out.print("name: " + name);
                System.out.println(", score: " + score);
            }
            rs.close();
        } finally {
            if (stmt != null)
            stmt.close();
            if (conn != null)
            conn.close();
        }
        System.out.println("Done!");
    }
}
```

Trong đoạn mã trên:

- Class.forName(DRIVER_CLASS) được sử dụng để tải trình điều khiển JDBC cho SQL Server vào bộ nhớ.
- DriverManager.getConnection(DB_URL, USER, PASS) được sử dụng để tạo đối tượng Connection để kết nối đến cơ sở dữ liệu.
- conn.createStatement() được sử dụng để tạo đối tượng Statement để thực thi câu truy vấn SQL.
- stmt.executeQuery("SELECT name, score FROM highscore") được sử dụng để thực thi câu truy vấn SQL và lưu kết quả vào đối tượng ResultSet.

- rs.getString("name") và rs.getInt("score") được sử dụng để lấy giá trị của các cột trong dòng dữ liệu.
- rs.close(), stmt.close() và conn.close() được sử dụng để đóng đối tượng ResultSet, Statement và Connection.

VI) HighScore.java:

```
public class HighScore extends JDialog
```

Lớp HighScore kế thừa từ lớp JDialog để tạo một hộp thoại hiển thị danh sách 5 điểm cao nhất từ cơ sở dữ liệu.

```
private JPanel scorePanel;
private JButton exitButton;

public class Score {
    private String name;
    private int score;

    public Score(String name, int score) {
        this.name = name;
        this.score = score;
    }

    public String getName() {
        return name;
    }

    public int getScore() {
        return score;
    }
}
```

Lớp Score là một lớp nội bộ được định nghĩa bên trong lớp HighScore để đại diện cho một cặp tên và điểm số của người chơi. Lớp Score có hai thuộc tính name và score để lưu trữ tên và điểm số của người chơi, và có một phương thức khởi tạo để thiết lập giá trị cho hai thuộc tính này. Ngoài ra, lớp Score cũng có hai phương thức getName() và getScore() để trả về tên và điểm số của người chơi.

```
public HighScore(JFrame parentFrame) {
    super(parentFrame, "Điểm cao nhất", true);
    setDefaultCloseOperation(DISPOSE_ON_CLOSE);
    setSize(300, 300);
    setLocationRelativeTo(parentFrame);

    // Tạo JPanel để chứa bảng điểm cao nhất
    scorePanel = new JPanel();
    scorePanel.setLayout(new BoxLayout(scorePanel, BoxLayout.Y_AXIS));
    add(scorePanel, BorderLayout.CENTER);

    // Lấy danh sách 5 điểm cao nhất từ cơ sở dữ liệu
    ArrayList<Score> highScores = getHighScores();
    JLabel scoreLabel = new JLabel("Highscore Babeee");
    scoreLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
    scoreLabel.setFont(new Font("Arial", Font.BOLD, 24));
    scorePanel.add(scoreLabel);
    // Thêm điểm cao nhất vào bảng
    for (int i = 0; i < highScores.size(); i++) {
        JLabel Title = new JLabel((i + 1) + ". " + highScores.get(i).name + "|" + highScores.get(i).score);
        Title.setAlignmentX(Component.CENTER_ALIGNMENT);
        Title.setFont(new Font("Arial", Font.BOLD, 20));
        scorePanel.add(Title);
    }

    // Thêm nút "Exit" để quay lại GameMenu
    exitButton = new JButton("Exit");
    exitButton.setAlignmentX(Component.CENTER_ALIGNMENT);
    exitButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            dispose();
        }
    });
    scorePanel.add(Box.createRigidArea(new Dimension(0, 10)));
    scorePanel.add(exitButton);
}
```

class HighScore được viết bằng Java, đây là một cửa sổ hiển thị danh sách 5 điểm số cao nhất từ cơ sở dữ liệu. Các phương thức được sử dụng trong class này bao gồm:

public HighScore(JFrame parentFrame): đây là constructor của class HighScore, nhận một đối tượng JFrame làm tham số. Constructor này thiết lập cấu hình cho cửa sổ hiển thị, bao gồm tiêu đề, kích thước và vị trí của cửa sổ. Nó cũng tạo ra một JPanel để chứa danh sách điểm số và nút "Exit" để quay lại menu chính.

```
public ArrayList<Score> getHighScores() {
    ArrayList<Score> highScores = new ArrayList<>();
    Connection conn = null;
    java.sql.Statement stmt = null;
    ResultSet rs = null;
    // Tạo kết nối đến cơ sở dữ liệu
    try {
        Class.forName(MySqlDemo.DRIVER_CLASS);
        conn = DriverManager.getConnection(MySqlDemo.DB_URL, MySqlDemo.USER, MySqlDemo.PASS);

        // Thực hiện truy vấn để lấy toàn bộ bảng điểm và sắp xếp lại theo thứ tự giảm dần của điểm số
        String sql = "SELECT top 5 name, score FROM highscore ORDER BY score DESC ";
        stmt = conn.createStatement();
        rs = stmt.executeQuery(sql);

        // Lấy tên và điểm của người có điểm cao nhất và trả về chúng
        while (rs.next()) {
            String name = rs.getString("name");
            int score = rs.getInt("score");
            Score highestScore = new Score(name, score);
            highScores.add(highestScore);
        }
    }
    // Xử lý ngoại lệ và đóng kết nối
    catch (ClassNotFoundException | SQLException ex) {
        ex.printStackTrace();
    }
```

public ArrayList<Score> getHighScores(): đây là phương thức dùng để lấy danh sách 5 điểm cao nhất từ cơ sở dữ liệu. Phương thức này sử dụng JDBC để kết nối đến cơ sở dữ liệu MySQL và thực hiện truy vấn để lấy danh sách điểm số. Sau đó, phương thức trả về một ArrayList chứa danh sách các đối tượng Score đại diện cho các điểm số cao nhất.

Đối tượng Score được sử dụng trong class này để lưu trữ tên và điểm số của người chơi. Class Score có 2 thuộc tính là name và score. Trong constructor của Score, 2 thuộc tính này được khởi tạo.

Class HighScore sử dụng cấu trúc dữ liệu ArrayList để lưu trữ danh sách các đối tượng Score. Nó cũng sử dụng JLabel để hiển thị các điểm số trong JPanel. Cuối cùng, nó thêm một ActionListener vào nút "Exit" để đóng cửa sổ và quay lại menu chính.

Chương III: Kết luận

1. Về kiến thức học tập

Nắm vững kiến thức lý thuyết cơ bản phục vụ tốt cho việc thiết kế chương trình

Thiết kế được cơ sở dữ liệu tương đối hoàn chỉnh, đáp ứng tốt cho việc viết chương trình.

Củng cố lại các kiến thức đã học, đặc biệt là kĩ năng phân tích, giải quyết vấn đề.

Biết cách ứng dụng lí thuyết vào thực tế.

2. Hạn chế của đề tài

Do thời gian nghiên cứu hạn chế, nên tính chuyên nghiệp chưa cao

Tài liệu tham khảo

1) https://www.w3schools.com/java/
2) https://viettuts.vn/java-jdbc/ket-noi-java-voi-sqlserver