

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI



BÁO CÁO BÀI TẬP LỚN

Nhóm: 1

Môn: Lập trình Java

Đề tài: Game di chuyển Pacman

Giảng viên hướng dẫn: Vũ Huấn

Lớp: CNTT VA1

Thành viên nhóm:

- Đinh Công Linh - 212602748
- Trần Minh Hiếu - 212643346

Hà Nội, 28-04-2023.

Mục lục nội dung

I.	Tổng quan	6
II.	Bố cục, thiết kế game	6
1.	Thiết kế giao diện	6
1.1.	Giao diện đăng nhập	6
1.2.	Giao diện chơi game	6
1.3.	Giao diện kết thúc game	7
1.4.	Giao diện database: Lưu trữ thông tin người chơi	7
2.	Chức năng, phím bấm	7
III.	Xây dựng cấu trúc game	8
1.	Thư mục game	8
1.1.	Package Authen	8
1.2.	Package PacMan	8
1.3.	Package Resource	8
2.	Giới thiệu về giao diện hệ thống và tương tác người dùng	9
2.1.	Kết nối Database	9
2.2.	Xây dựng giao diện đăng nhập, đăng ký	9
2.3.	Xây dựng tương tác của người dùng	10
3.	Xây dựng các phương thức và khởi tạo game	15
1.	Xây dựng lớp Board:	15
2.	Xây dựng các hàm	21
3.	Lớp Pacman:	39
IV.	Tổng kết	40
	Kết luận	40
	Tài liệu tham khảo:	40

Mục lục ảnh:

Ảnh 1:	Giao diện đăng nhập	6
Ảnh 2:	Giao diện chơi game	6
Ảnh 3:	Giao diện kết thúc game	7
Ảnh 4:	Giao diện database	7

Ảnh 5: Thư mục game	8
Ảnh 6: Kết nối Database	9
Ảnh 7: Màn hình đăng nhập	9
Ảnh 8: Màn hình đăng ký	10
Ảnh 9: File thiết kế giao diện	10
Ảnh 10: File xây dựng Jpanel đăng nhập và đăng ký.....	10
Ảnh 11: File Account	11
Ảnh 12: File AccountDAO #1	11
Ảnh 13: File AccountDAO #2	12
Ảnh 14: File AccountDAO – Update điểm số	12
Ảnh 15: File FormLoginController.....	13
Ảnh 16: File FormRegisterController	13
Ảnh 17: Xử lý sự kiện click: Submit.....	14
Ảnh 18: Xử lý sự kiện click: Reset	14
Ảnh 19: Xử lý sự kiện click: Login.....	15
Ảnh 20: Xử lý sự kiện click: SignUp.....	15
Ảnh 21: Khai báo thư viện	16
Ảnh 22: Khởi tạo các biến #1.....	17
Ảnh 23: Khởi tạo các biến #2.....	18
Ảnh 24: Hàm khởi tạo	21
Ảnh 25: Hàm loadSound.....	22
Ảnh 26: Hàm loadImages()	22
Ảnh 27: Hàm initVariables()	23
Ảnh 28: Hàm playGame.....	24
Ảnh 29: Hàm showIntroScreen.....	24
Ảnh 30: Hàm drawScore.....	25
Ảnh 31: Hàm checkMaze.....	26
Ảnh 32: Hàm death	27
Ảnh 33: Hàm gameOver	28
Ảnh 34: Hàm moveGhost	29
Ảnh 35: Hàm drawGhost	30
Ảnh 36: Hàm movePacman	31
Ảnh 37: Phương thức kiểm tra việc va chạm với tường.....	32
Ảnh 38: Hàm drawPacman	33
Ảnh 39: Hàm drawMaze.....	34
Ảnh 40: Điểm mỗi cuối cùng.....	35
Ảnh 41: Hàm initGame	35

Ảnh 42: Hàm continueLevel	37
Ảnh 43: Hàm paintComponent.....	38
Ảnh 44: Hàm Tadapter, xử lý sự kiện bấm phím.....	39
Ảnh 45: Lớp Pacman	39

Lời mở đầu

- Các tựa game di chuyển bằng nút, phím bấm đã phổ biến từ lâu. Như game đi cảnh Megaman, Mario,Trong số đó thì tựa game di chuyển phổ biến, được nhiều người biết đến và từng chơi đó là Pacman.
- Giới thiệu qua về tựa game này thì Pac-Man (パックマン Pakkuman) là một trò chơi arcade được phát triển bởi Namco và phát hành đầu tiên tại Nhật Bản vào 22 tháng 5 năm 1980.[1][2] Trở nên nổi tiếng và được ưa thích ngay từ khi được phát hành cho đến ngày nay, Pac-Man được xem là một trò chơi kinh điển và trở thành một biểu tượng của văn hóa đại chúng những năm 80.
- Tựa game này đã được nhóm em lập trình bằng Java, một ngôn ngữ lập trình hướng đối tượng, kết hợp với kiến thức về cơ sở dữ liệu để bổ sung thêm tính năng cho game như việc lưu trữ điểm, tài khoản của người chơi
- Trong bài báo cáo này, nhóm em sẽ trình bày cơ bản về các kỹ thuật được sử dụng trong quá trình làm game, mô tả cách hoạt động của game cũng như cơ sở dữ liệu của game

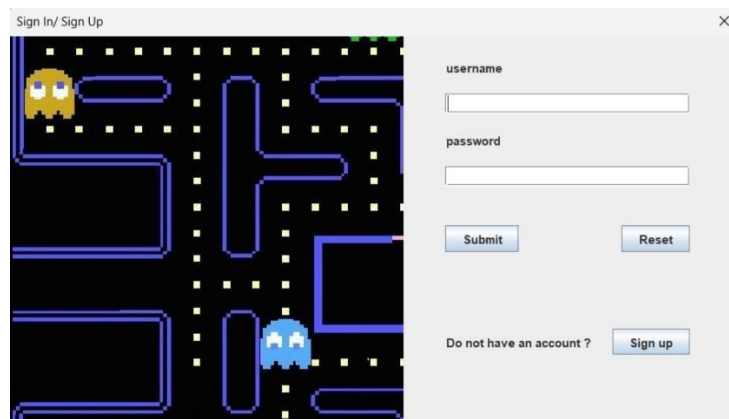
I. Tổng quan

- Tên trò chơi: Game Pacman 2D Java
- Ngôn ngữ lập trình: Java
- Phần mềm sử dụng: IDE IntelliJ IDEA, IDE Netbeans, bộ công cụ JDK (Java SE Development Kit 19.0.2)

II. Bố cục, thiết kế game

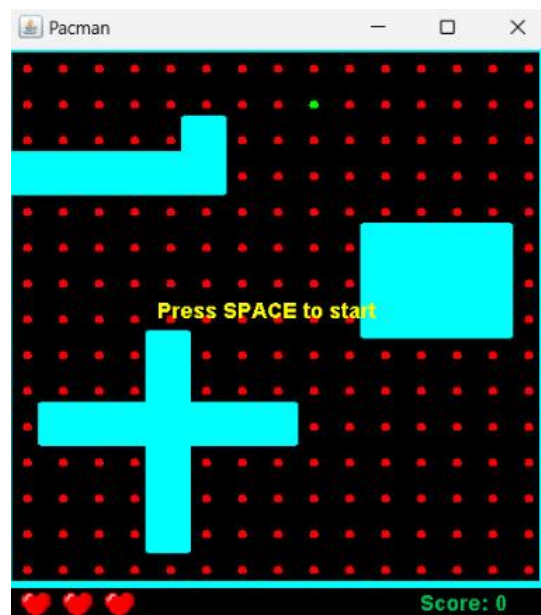
1. Thiết kế giao diện

1.1. Giao diện đăng nhập



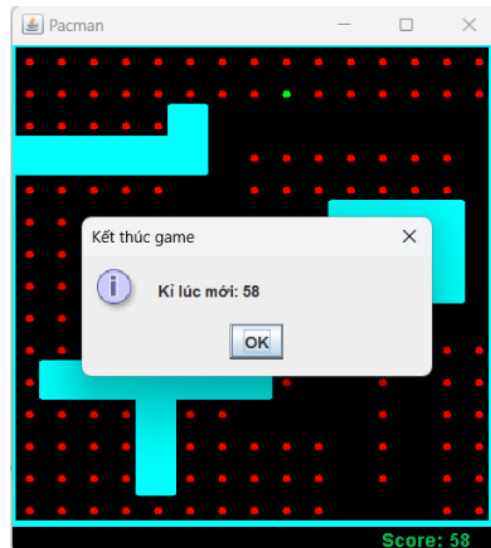
Ảnh 1: Giao diện đăng nhập

1.2. Giao diện chơi game



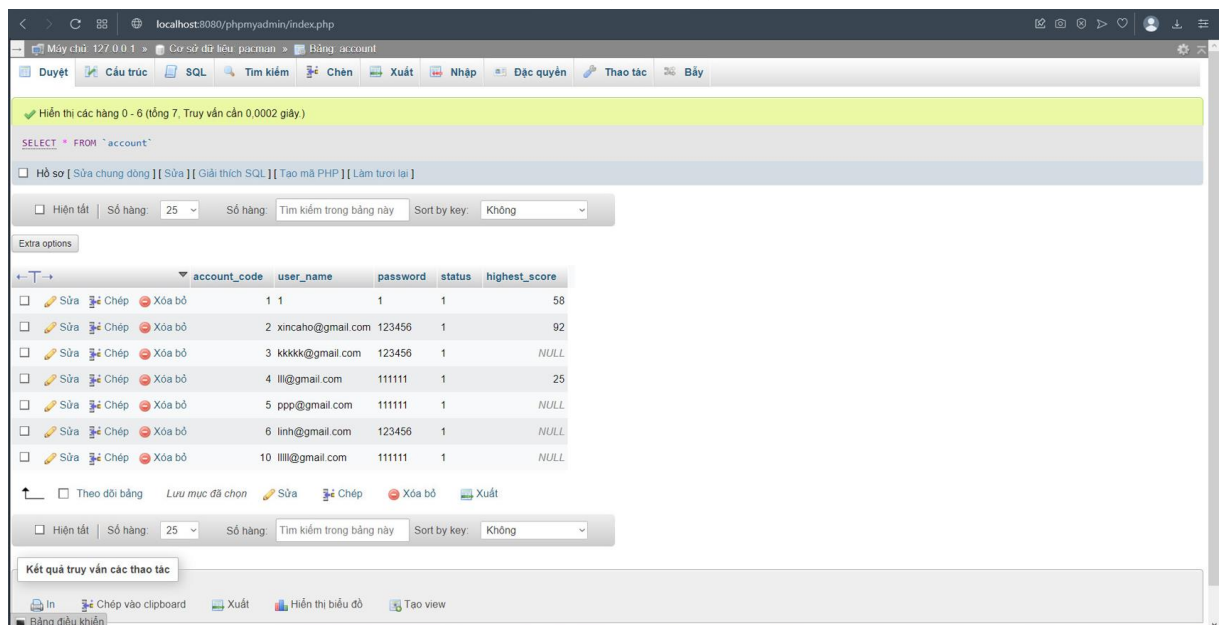
Ảnh 2: Giao diện chơi game

1.3. Giao diện kết thúc game



Ảnh 3: Giao diện kết thúc game

1.4. Giao diện database: Lưu trữ thông tin người chơi



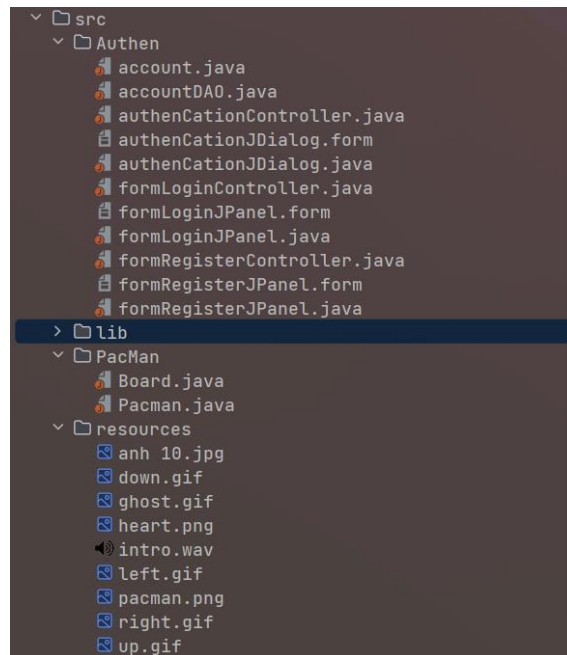
Ảnh 4: Giao diện database

2. Chức năng, phím bấm

- Tạo tài khoản mới
- Hiển thị điểm số cao nhất của tài khoản
- Các nút di chuyển nhân vật, nút chơi lại

III. Xây dựng cấu trúc game

1. Thư mục game



Ảnh 5: Thư mục game

1.1. Package Authen

Chứa các lớp mô tả và xây dựng cơ sở dữ liệu trong database.

1.2. Package PacMan

Chứa 2 lớp chính là:

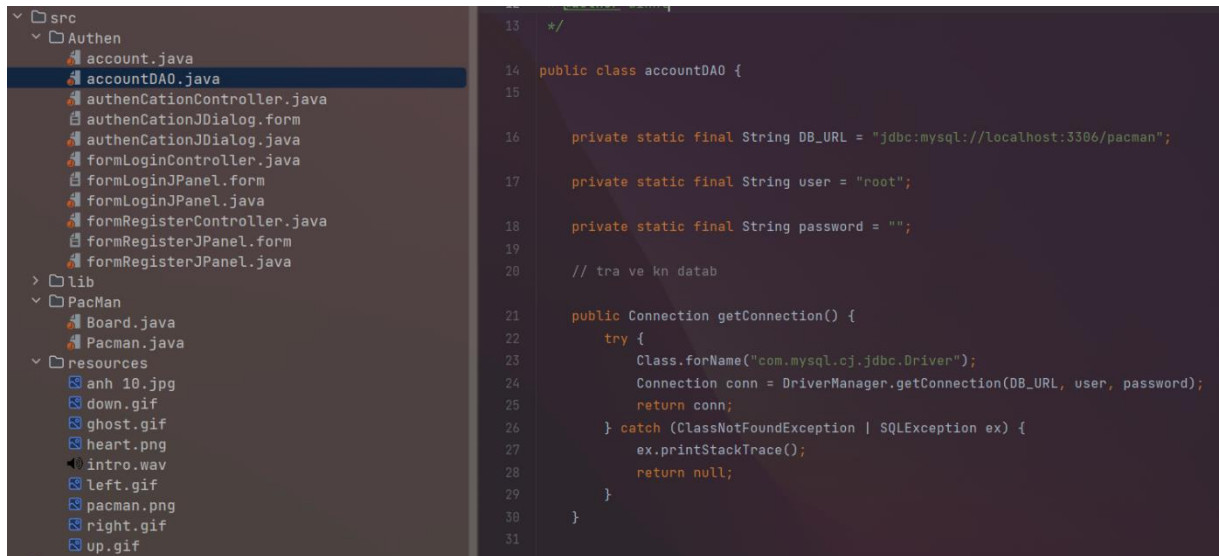
- Class Board: Lớp xây dựng Object, các thuật toán di chuyển, vẽ bản đồ, xử lý sự kiện trong game.
- Class Main: Là file main của ứng dụng, khởi động và cấu hình game.

1.3. Package Resource

Lưu trữ các tài nguyên trong game: Ảnh nhân vật, nhạc nền, ảnh minh họa

2. Giới thiệu về giao diện hệ thống và tương tác người dùng

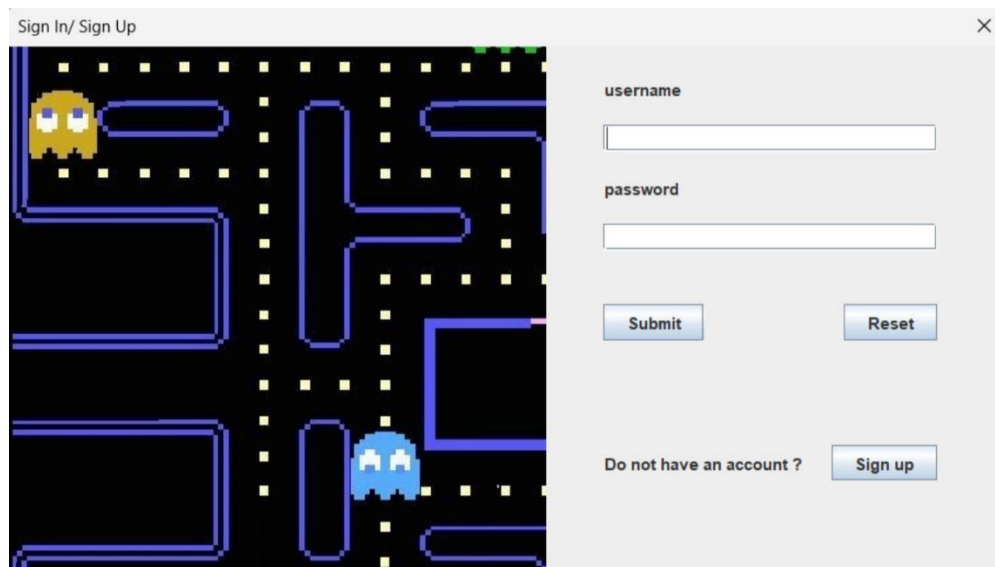
2.1. Kết nối Database



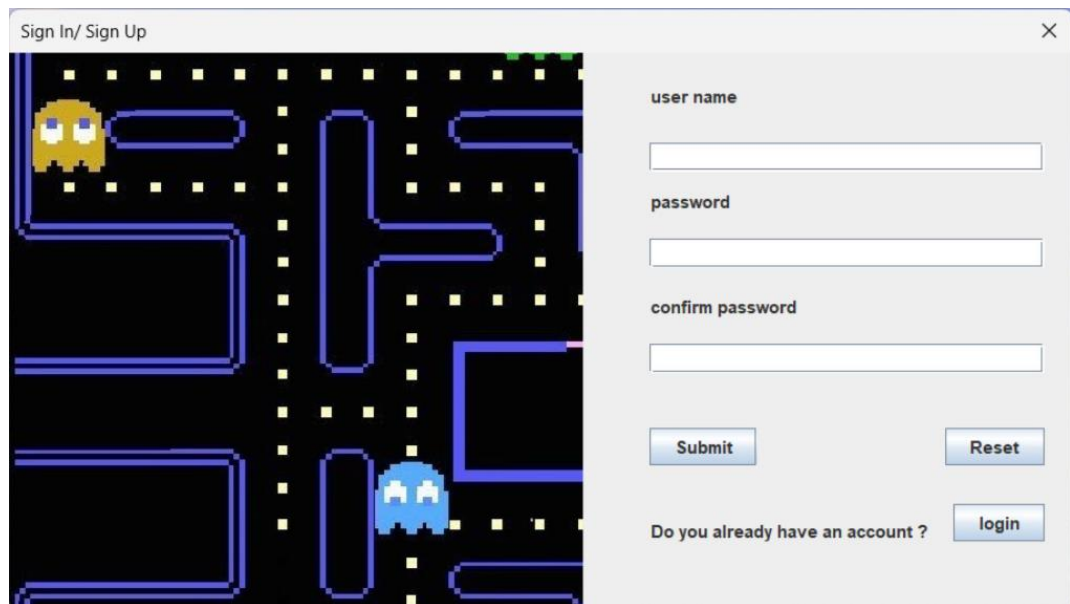
Ảnh 6: Kết nối Database

2.2. Xây dựng giao diện đăng nhập, đăng ký

2.2.1. Giao diện:



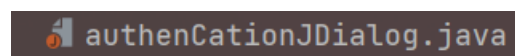
Ảnh 7: Màn hình đăng nhập



Ảnh 8: Màn hình đăng ký

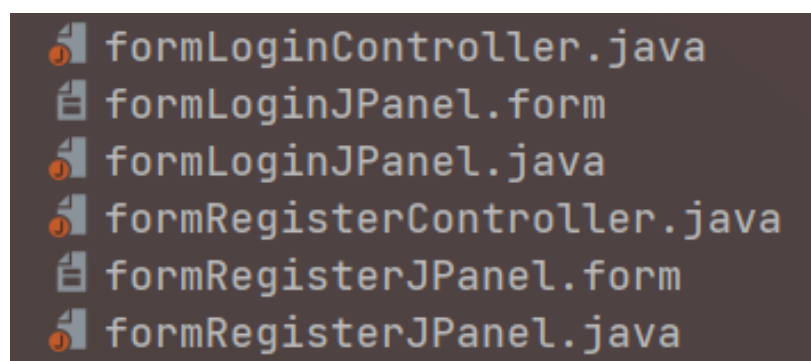
2.2.2. Xây dựng giao diện

- File `authenCationJDialog.java` dùng để thiết kế ra 1 JFrame trong đó có 2 JPanel: 1 JPanel dùng để hiện phần ảnh nền trang trí và 1 JPanel dùng để hiện `formLoginController.java` và `formRegisterController.java`



Ảnh 9: File thiết kế giao diện

- File `formLoginController.java` và `formRegisterController.java` dùng để thiết kế ra JFrame trong đó có thiết kế ra 2 JPanel tương ứng là

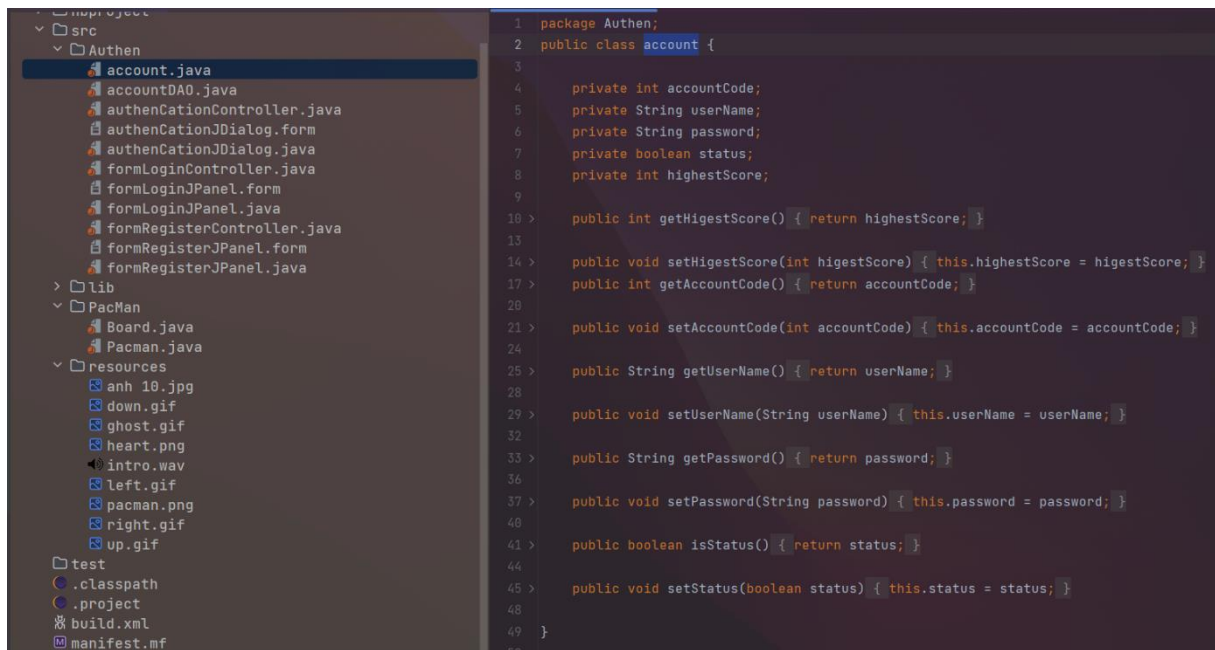


Ảnh 10: File xây dựng JPanel đăng nhập và đăng ký

2.3. Xây dựng tương tác của người dùng

2.3.1. File Account

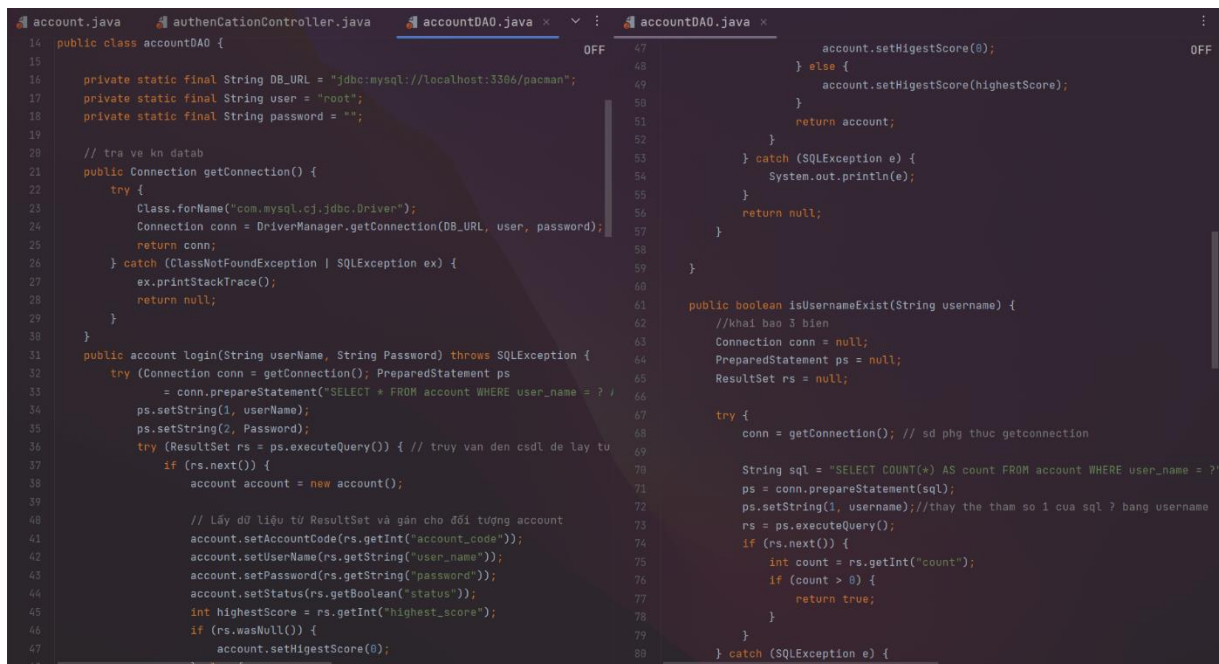
Xây dựng các thuộc tính, phương thức quản lý thông tin tài khoản



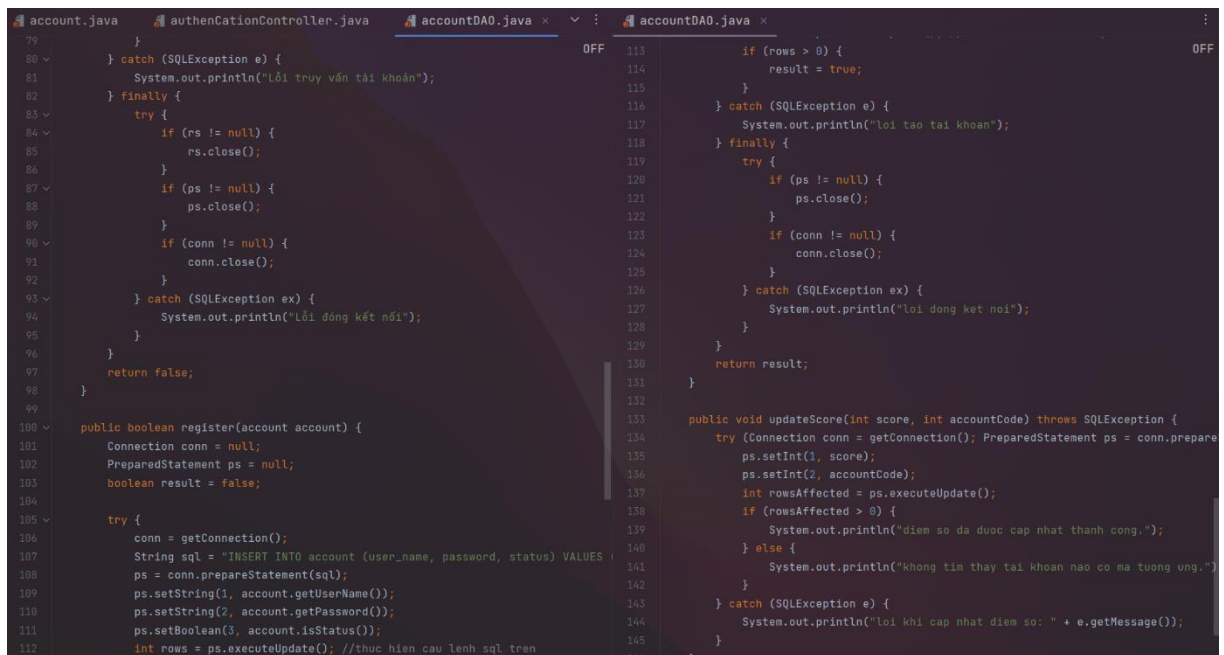
Ảnh 11: File Account

2.3.2. File AccountDAO

File này dùng để định nghĩa và xây dựng các phương thức tương tác với bảng Account trên Database bao gồm phương thức login để kiểm tra tên đăng nhập và mật khẩu của tài khoản, phương thức register để đăng ký tài khoản và phương thức isUsernameExist để kiểm tra xem tên đăng nhập đã tồn tại trong cơ sở dữ liệu chưa

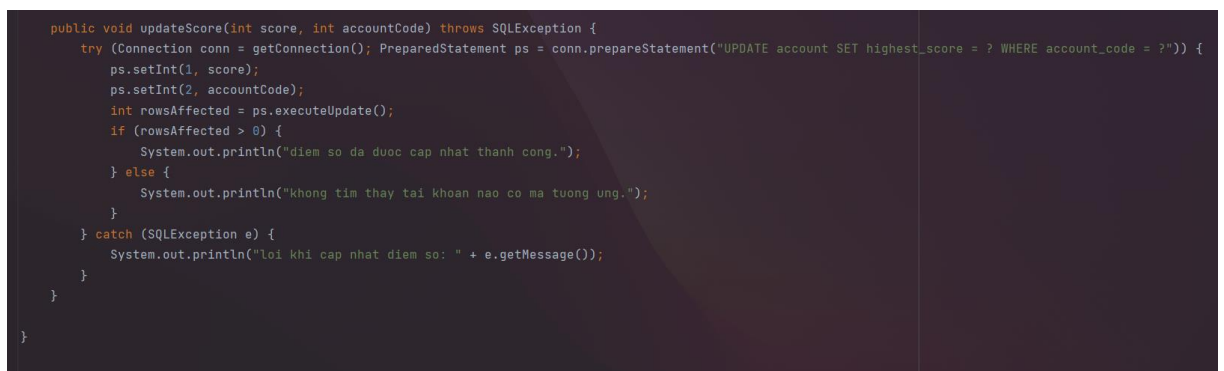


Ảnh 12: File AccountDAO #1



```
79
80 } catch (SQLException e) {
81     System.out.println("Lỗi truy vấn tài khoản");
82 } finally {
83     try {
84         if (rs != null) {
85             rs.close();
86         }
87         if (ps != null) {
88             ps.close();
89         }
90         if (conn != null) {
91             conn.close();
92         }
93     } catch (SQLException ex) {
94         System.out.println("Lỗi đóng kết nối");
95     }
96 }
97 return false;
98 }
99
100 public boolean register(account account) {
101     Connection conn = null;
102     PreparedStatement ps = null;
103     boolean result = false;
104
105     try {
106         conn = getConnection();
107         String sql = "INSERT INTO account (user_name, password, status) VALUES ("
108             + account.getUserName() + ", "
109             + account.getPassword() + ", "
110             + account.isStatus() + ")";
111         ps = conn.prepareStatement(sql);
112         int rows = ps.executeUpdate(); //thực hiện câu lệnh sql trên
113
114         if (rows > 0) {
115             result = true;
116         }
117     } catch (SQLException e) {
118         System.out.println("lỗi tạo tài khoản");
119     } finally {
120         try {
121             if (ps != null) {
122                 ps.close();
123             }
124             if (conn != null) {
125                 conn.close();
126             }
127         } catch (SQLException ex) {
128             System.out.println("lỗi đóng kết nối");
129         }
130     }
131     return result;
132 }
133
134 public void updateScore(int score, int accountCode) throws SQLException {
135     try (Connection conn = getConnection(); PreparedStatement ps = conn.prepareStatement(
136         "UPDATE account SET highest_score = ? WHERE account_code = ?")) {
137         ps.setInt(1, score);
138         ps.setInt(2, accountCode);
139         int rowsAffected = ps.executeUpdate();
140         if (rowsAffected > 0) {
141             System.out.println("điểm số đã được cập nhật thành công.");
142         } else {
143             System.out.println("không tìm thấy tài khoản nào có mã tương ứng.");
144         }
145     } catch (SQLException e) {
146         System.out.println("lỗi khi cập nhật điểm số: " + e.getMessage());
147     }
148 }
```

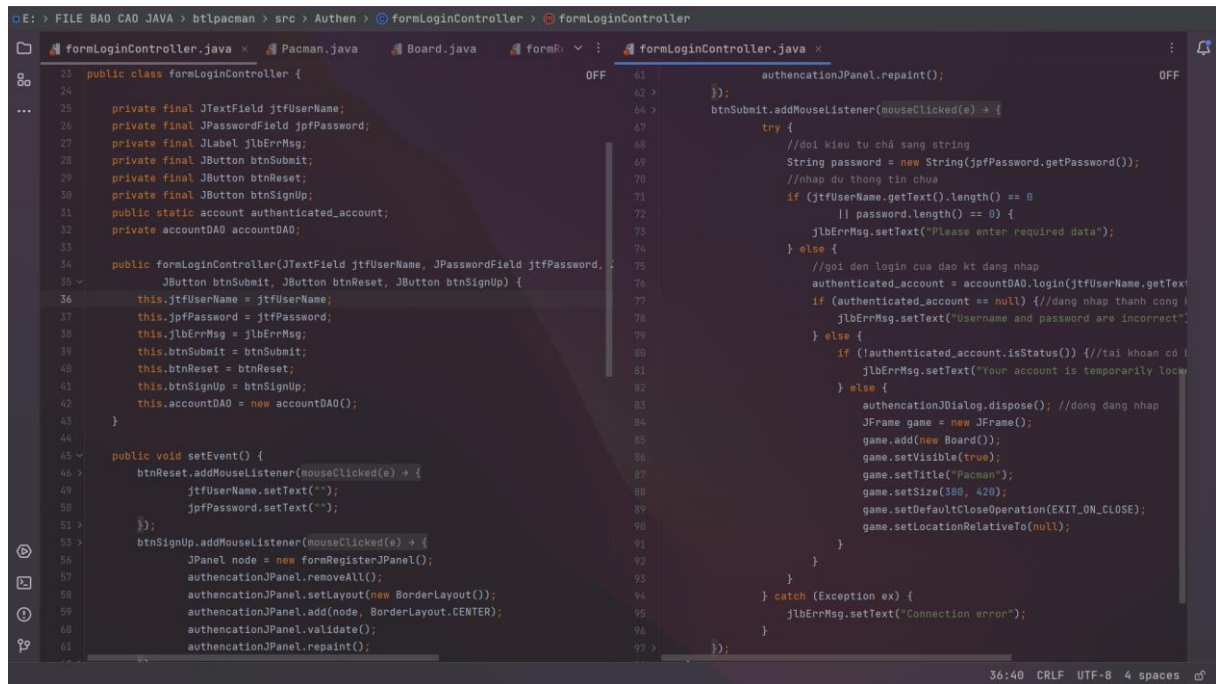
Ảnh 13: File AccountDAO #2



```
public void updateScore(int score, int accountCode) throws SQLException {
    try (Connection conn = getConnection(); PreparedStatement ps = conn.prepareStatement("UPDATE account SET highest_score = ? WHERE account_code = ?")) {
        ps.setInt(1, score);
        ps.setInt(2, accountCode);
        int rowsAffected = ps.executeUpdate();
        if (rowsAffected > 0) {
            System.out.println("điểm số đã được cập nhật thành công.");
        } else {
            System.out.println("không tìm thấy tài khoản nào có mã tương ứng.");
        }
    } catch (SQLException e) {
        System.out.println("lỗi khi cập nhật điểm số: " + e.getMessage());
    }
}
```

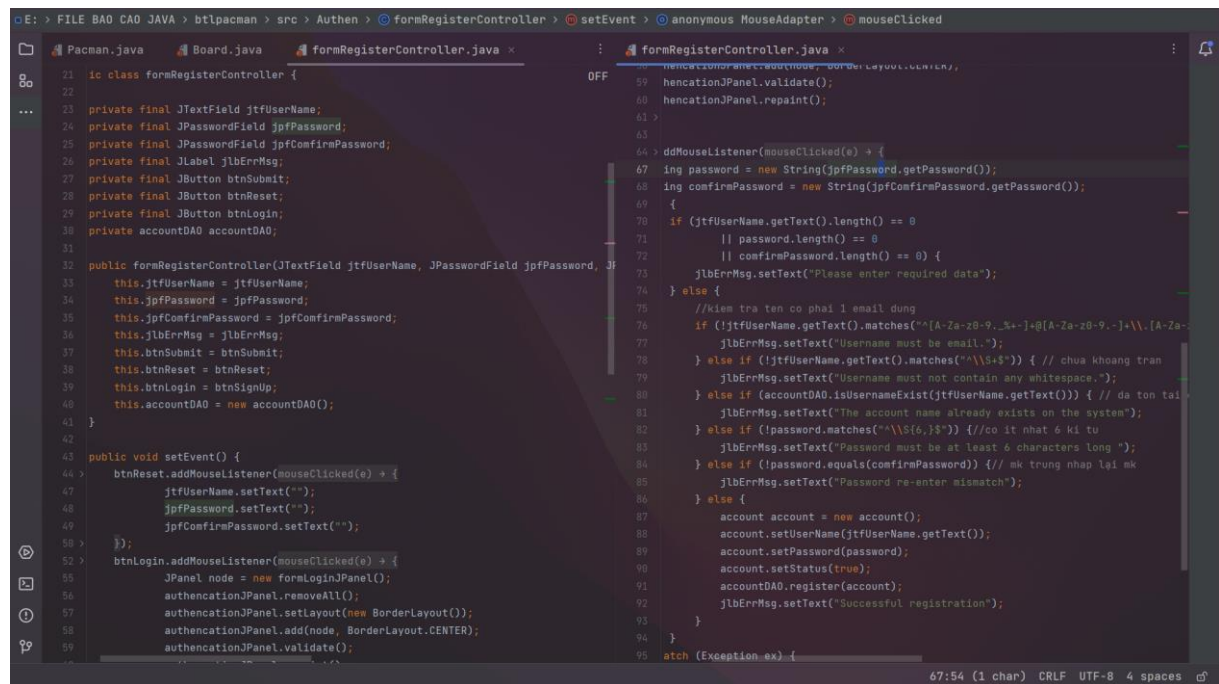
Ảnh 14: File AccountDAO – Update điểm số

2.3.3. File FormRegisterController và FormLoginController



```
23 public class FormLoginController {
24
25     private final JTextField jtfUserName;
26     private final JPasswordField jpfPassword;
27     private final JLabel jlbErrMsg;
28     private final JButton btnSubmit;
29     private final JButton btnReset;
30     private final JButton btnSignUp;
31     public static account authenticated_account;
32     private accountDAO accountDAO;
33
34     public FormLoginController(JTextField jtfUserName, JPasswordField jpfPassword,
35                               JButton btnSubmit, JButton btnReset, JButton btnSignUp) {
36         this.jtfUserName = jtfUserName;
37         this.jpfPassword = jpfPassword;
38         this.jlbErrMsg = jlbErrMsg;
39         this.btnSubmit = btnSubmit;
40         this.btnReset = btnReset;
41         this.btnSignUp = btnSignUp;
42         this.accountDAO = new accountDAO();
43     }
44
45     public void setEvent() {
46         btnReset.addMouseListener(new MouseAdapter() {
47             public void mouseClicked(e) {
48                 jtfUserName.setText("");
49                 jpfPassword.setText("");
50             }
51         });
52         btnSignUp.addMouseListener(new MouseAdapter() {
53             public void mouseClicked(e) {
54                 JPanel node = new formRegisterJPanel();
55                 authenticationJPanel.removeAll();
56                 authenticationJPanel.setLayout(new BorderLayout());
57                 authenticationJPanel.add(node, BorderLayout.CENTER);
58                 authenticationJPanel.validate();
59                 authenticationJPanel.repaint();
60             }
61         });
62         btnSubmit.addMouseListener(new MouseAdapter() {
63             public void mouseClicked(e) {
64                 try {
65                     //doi kieu tu char sang string
66                     String password = new String(jpfPassword.getPassword());
67                     //nhap du thong tin chua
68                     if (jtfUserName.getText().length() == 0
69                         || password.length() == 0) {
70                         jlbErrMsg.setText("Please enter required data");
71                     } else {
72                         //goi den login cua dao kt dang nhap
73                         authenticated_account = accountDAO.login(jtfUserName.getText(),
74                             password);
75                         if (authenticated_account == null) {
76                             //dang nhap thanh cong
77                             jlbErrMsg.setText("Username and password are incorrect");
78                         } else {
79                             if (!authenticated_account.isStatus()) {
80                                 //tai khoan co
81                                 jlbErrMsg.setText("Your account is temporarily locked");
82                             } else {
83                                 authenticationJDialog.dispose(); //dong dang nhap
84                                 JFrame game = new JFrame();
85                                 game.add(new Board());
86                                 game.setVisible(true);
87                                 game.setTitle("Pacman");
88                                 game.setSize(380, 420);
89                                 game.setDefaultCloseOperation(EXIT_ON_CLOSE);
90                                 game.setLocationRelativeTo(null);
91                                 game.setVisible(true);
92                             }
93                         }
94                     } catch (Exception ex) {
95                         jlbErrMsg.setText("Connection error");
96                     }
97                 }
98             }
99         });
100     }
101 }
```

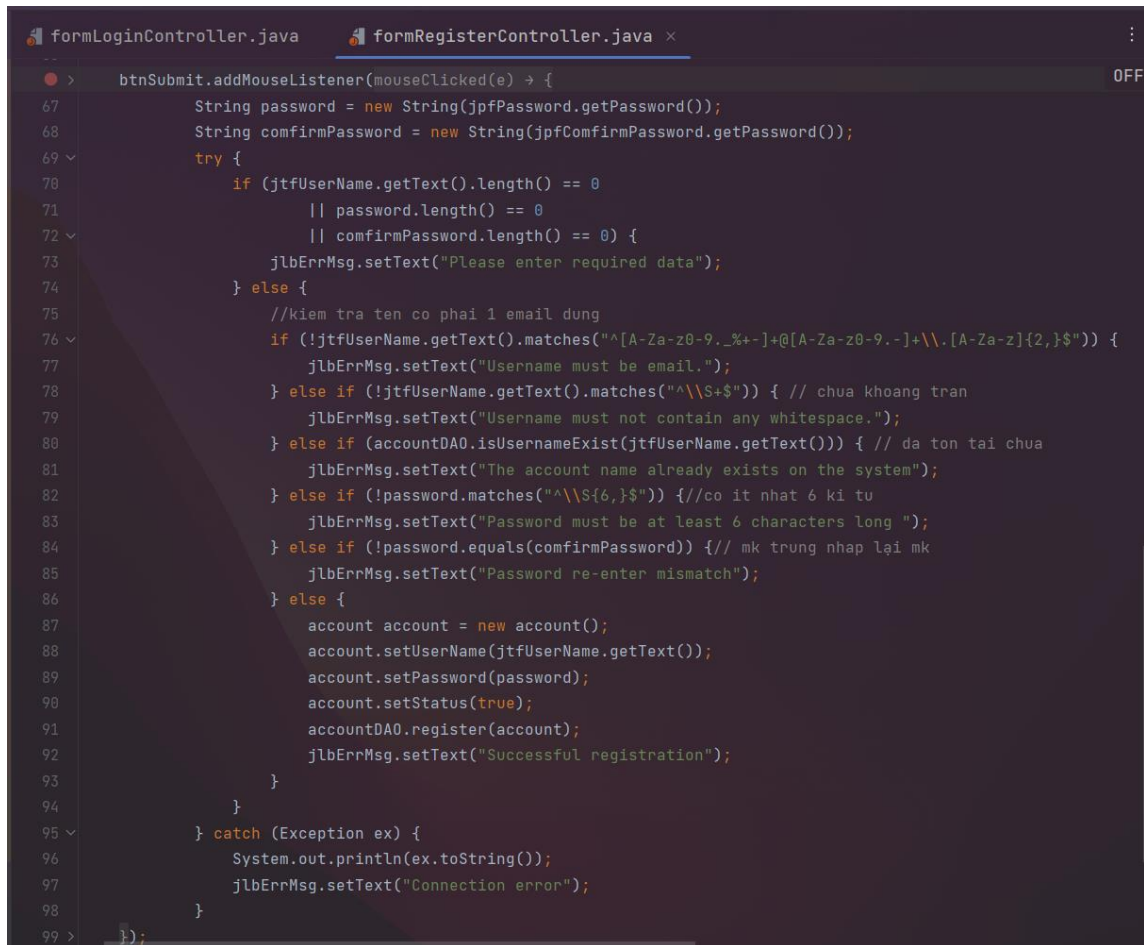
Ảnh 15: File FormLoginController



```
21 public class FormRegisterController {
22
23     private final JTextField jtfUserName;
24     private final JPasswordField jpfPassword;
25     private final JPasswordField jpfConfirmPassword;
26     private final JLabel jlbErrMsg;
27     private final JButton btnSubmit;
28     private final JButton btnReset;
29     private final JButton btnLogin;
30     private final JButton btnSignUp;
31     private accountDAO accountDAO;
32
33     public FormRegisterController(JTextField jtfUserName, JPasswordField jpfPassword,
34                                   JPasswordField jpfConfirmPassword, JLabel jlbErrMsg,
35                                   JButton btnSubmit, JButton btnReset, JButton btnLogin,
36                                   JButton btnSignUp) {
37         this.jtfUserName = jtfUserName;
38         this.jpfPassword = jpfPassword;
39         this.jpfConfirmPassword = jpfConfirmPassword;
40         this.jlbErrMsg = jlbErrMsg;
41         this.btnSubmit = btnSubmit;
42         this.btnReset = btnReset;
43         this.btnLogin = btnLogin;
44         this.btnSignUp = btnSignUp;
45         this.accountDAO = new accountDAO();
46     }
47
48     public void setEvent() {
49         btnReset.addMouseListener(new MouseAdapter() {
50             public void mouseClicked(e) {
51                 jtfUserName.setText("");
52                 jpfPassword.setText("");
53                 jpfConfirmPassword.setText("");
54             }
55         });
56         btnLogin.addMouseListener(new MouseAdapter() {
57             public void mouseClicked(e) {
58                 JPanel node = new formLoginJPanel();
59                 authenticationJPanel.removeAll();
60                 authenticationJPanel.setLayout(new BorderLayout());
61                 authenticationJPanel.add(node, BorderLayout.CENTER);
62                 authenticationJPanel.validate();
63                 authenticationJPanel.repaint();
64             }
65         });
66         btnSignUp.addMouseListener(new MouseAdapter() {
67             public void mouseClicked(e) {
68                 try {
69                     //kiem tra ten co phai 1 email dung
70                     if (!jtfUserName.getText().matches("[A-Za-z0-9-_.%+]+@[A-Za-z0-9-_.]+\\.[A-Za-z]{2,6}")) {
71                         jlbErrMsg.setText("Username must be email.");
72                     } else if (!jtfUserName.getText().matches("[\\S+]+$")) {
73                         //chua khoang tran
74                         jlbErrMsg.setText("Username must not contain any whitespace.");
75                     } else if (accountDAO.isUsernameExist(jtfUserName.getText())) {
76                         //da ton tai
77                         jlbErrMsg.setText("The account name already exists on the system");
78                     } else if (!jpfPassword.getPassword().length() < 6) {
79                         //co it nhut 6 ki tu
80                         jlbErrMsg.setText("Password must be at least 6 characters long");
81                     } else if (!jpfPassword.getPassword().equals(jpfConfirmPassword.getPassword())) {
82                         //mk trung nhap lai mk
83                         jlbErrMsg.setText("Password re-enter mismatch");
84                     } else {
85                         account account = new account();
86                         account.setUserName(jtfUserName.getText());
87                         account.setPassword(jpfPassword.getPassword());
88                         account.setStatus(true);
89                         accountDAO.register(account);
90                         jlbErrMsg.setText("Successful registration");
91                     }
92                 } catch (Exception ex) {
93                     jlbErrMsg.setText("Connection error");
94                 }
95             }
96         });
97     }
98 }
```

Ảnh 16: File FormRegisterController

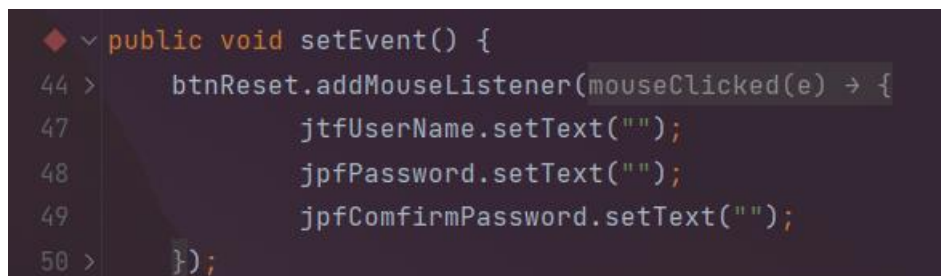
- Xét sự kiện nút Submit. Đăng nhập đúng thì vào game(login), hay tạo thành công 1 tài khoản mới và lưu trên database (register) , nếu sai sẽ in ra thông báo lỗi, hoặc thiếu thì cũng xuất hiện thông báo.



```
formLoginController.java  formRegisterController.java x
btnSubmit.addMouseListener(mouseClicked(e) -> {
67     String password = new String(jpfPassword.getPassword());
68     String confirmPassword = new String(jpfConfirmPassword.getPassword());
69     try {
70         if (jtfUserName.getText().length() == 0
71             || password.length() == 0
72             || confirmPassword.length() == 0) {
73             jlbErrMsg.setText("Please enter required data");
74         } else {
75             //kiem tra ten co phai 1 email dung
76             if (!jtfUserName.getText().matches("^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,}$")) {
77                 jlbErrMsg.setText("Username must be email.");
78             } else if (!jtfUserName.getText().matches("^\\S+$")) { // chua khoang tran
79                 jlbErrMsg.setText("Username must not contain any whitespace.");
80             } else if (accountDAO.isUsernameExist(jtfUserName.getText())) { // da ton tai chua
81                 jlbErrMsg.setText("The account name already exists on the system");
82             } else if (!password.matches("^\\S{6,}$")) { //co it nhat 6 ki tu
83                 jlbErrMsg.setText("Password must be at least 6 characters long ");
84             } else if (!password.equals(confirmPassword)) { // mk trung nhap lai mk
85                 jlbErrMsg.setText("Password re-enter mismatch");
86             } else {
87                 account account = new account();
88                 account.setUserName(jtfUserName.getText());
89                 account.setPassword(password);
90                 account.setStatus(true);
91                 accountDAO.register(account);
92                 jlbErrMsg.setText("Successful registration");
93             }
94         }
95     } catch (Exception ex) {
96         System.out.println(ex.toString());
97         jlbErrMsg.setText("Connection error");
98     }
99 }
});
```

Ảnh 17: Xử lý sự kiện click: Submit

- Xét sự kiện nút Reset: xóa đi hết những thông tin mà người chơi vừa đăng nhập trên tài khoản mật khẩu đòi với login, register thêm cả confirmpasswork.



```
public void setEvent() {
44 >     btnReset.addMouseListener(mouseClicked(e) -> {
47         jtfUserName.setText("");
48         jpfPassword.setText("");
49         jpfConfirmPassword.setText("");
50 >     });
}
```

Ảnh 18: Xử lý sự kiện click: Reset

- Xét sự kiện nút Login: Thì giao diện Đăng nhập sẽ xuất hiện để người dùng Đăng nhập tài khoản đã tạo thay vì Đăng kí tài khoản mới.

```

btnLogin.addMouseListener(mouseClicked(e) → {
    JPanel node = new formLoginJPanel();
    authencationJPanel.removeAll();
    authencationJPanel.setLayout(new BorderLayout());
    authencationJPanel.add(node, BorderLayout.CENTER);
    authencationJPanel.validate();
    authencationJPanel.repaint();
});

```

Ảnh 19: Xử lý sự kiện click: Login

- Xét sự kiện click vào SignUp: Thì giao diện Đăng kí sẽ xuất hiện để người tạo tài khoản mới.

```

btnSignUp.addMouseListener(mouseClicked(e) → {
    JPanel node = new formRegisterJPanel();
    authencationJPanel.removeAll();
    authencationJPanel.setLayout(new BorderLayout());
    authencationJPanel.add(node, BorderLayout.CENTER);
    authencationJPanel.validate();
    authencationJPanel.repaint();
});

```

Ảnh 20: Xử lý sự kiện click: SignUp

3. Xây dựng các phương thức và khởi tạo game

1. Xây dựng lớp Board:

Đây là lớp xây dựng các đối tượng chính trong game như nhân vật, bản đồ và các phương thức khác như điểm số, di chuyển, nhạc nền và hình ảnh,..

1.1. Khai báo thư viện và kết nối với database:

Sử dụng một vài thư viện phổ biến như: awt để cung cấp lớp tạo ra giao diện người dùng đồ họa GUI hay java.sql để cho phép tương tác với cơ sở dữ liệu trên nền tảng Java,..

Ngoài ra còn sử dụng các thư viện cung cấp việc chơi và ghi các tệp âm thanh và xử lý các sự kiện khi click,...

```
3 ~ import Authen.accountDAO;  
4 import static Authen.formLoginController.authenticated_account;  
5 import java.awt.*;  
6 import java.awt.event.ActionEvent;  
7 import java.awt.event.ActionListener;  
8 import java.awt.event.KeyAdapter;  
9 import java.awt.event.KeyEvent;  
10 import java.io.File;  
11 import java.io.IOException;  
12 import java.io.UnsupportedEncodingException;  
13 import javax.sound.sampled.*;  
14 import javax.swing.ImageIcon;  
15 import javax.swing.JPanel;  
16 import javax.swing.Timer;  
17 import javax.sound.sampled.AudioInputStream;  
18 import javax.sound.sampled.AudioSystem;  
19 import javax.sound.sampled.Clip;  
20 import java.io.File;  
21 import javax.sound.sampled.LineUnavailableException;  
22 import javax.sound.sampled.UnsupportedAudioFileException;  
23 import java.io.IOException;  
24 import java.net.URL;  
25 import java.sql.SQLException;  
26 import java.util.logging.Level;  
27 import java.util.logging.Logger;  
28 import javax.swing.JOptionPane;
```

Ảnh 21: Khai báo thư viện

1.2. Khởi tạo các biến cần thiết

```
31
32     private Dimension d; // Biến để lưu trữ kích thước của khung hình hiển thị trò chơi.
33     private final Font smallFont = new Font("Arial", Font.BOLD, 14); // đối tượng Font được sử dụng để hiển thị văn bản trong trò chơi
34     private boolean inGame = false; // Biến boolean để xác định liệu trò chơi đang chạy hay không.
35     private boolean dying = false; // Biến boolean để xác định liệu Pac-Man đang chết hay không.
36     private Clip clipBack; // nhạc nền
37     private final int BLOCK_SIZE = 24; // Kích thước 1 ô trong ma trận
38     private final int N_BLOCKS = 15; // Số ô trong mỗi hàng và cột của ma trận
39     private final int SCREEN_SIZE = N_BLOCKS * BLOCK_SIZE; // Kích thước của khung hình hiển thị trò chơi
40     private final int MAX_GHOSTS = 12; // Số ma tối đa xuất hiện trong game
41     private final int PACMAN_SPEED = 6; // Tốc độ di chuyển của Pacman
42     private int N_GHOSTS = 6; // Số lượng ma trong trò chơi
43     private int lives, score; // Số mạng và điểm của Pacman
44     private static int highestScore; // Điểm số cao nhất đạt được trong trò chơi
45     private int[] dx, dy; // mảng các giá trị để xác định hướng di chuyển của Pac-Man.
46     private int[] ghost_x, ghost_y, ghost_dx, ghost_dy, ghostSpeed;
47 //các biến liên quan đến vị trí và tốc độ của ma.
48     private Image heart, ghost;
49     private Image up, down, left, right;
50 //các đối tượng Image để lưu trữ hình ảnh của Pac-Man, ma và các hướng di chuyển của Pac-Man.
51     private int pacman_x, pacman_y, pacmand_x, pacmand_y;
52     // các biến để lưu trữ vị trí và hướng di chuyển hiện tại của Pac-Man.
53     private int req_dx, req_dy;
54 //các biến để lưu trữ hướng di chuyển được yêu cầu bởi người chơi.
55     private boolean gameOver = false; // biến boolean để xác định liệu trò chơi đã kết thúc hay chưa.
56     private boolean isPaused = false; //biến boolean để xác định liệu trò chơi đang được tạm dừng hay không.
57     private boolean isStarted = false; //biến boolean để xác định liệu trò chơi đã bắt đầu hay chưa.
58     private boolean isReverse = false; //biến boolean để xác định liệu Pac-Man đang trong trạng thái đảo hướng di chuyển hay không.
59     private boolean reverseDirection = false;
60     private int reverseCountdown = 0;
61     private final int REVERSE_DURATION = 5;
```

Ảnh 22: Khởi tạo các biến #1

```

63     private final short levelData[] = {
64         19, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 22,
65         17, 16, 16, 16, 16, 24, 16, 16, 16, 16, 16, 16, 16, 16, 20,
66         25, 24, 24, 24, 28, 0, 17, 16, 16, 16, 16, 16, 16, 16, 20,
67         0, 0, 0, 0, 0, 0, 17, 16, 16, 16, 16, 16, 16, 16, 20,
68         19, 18, 18, 18, 18, 18, 16, 16, 16, 16, 24, 24, 24, 24, 20,
69         17, 16, 16, 16, 16, 16, 16, 16, 16, 20, 0, 0, 0, 0, 21,
70         17, 16, 16, 16, 16, 16, 16, 16, 16, 20, 0, 0, 0, 0, 21,
71         17, 16, 16, 16, 24, 16, 16, 16, 16, 20, 0, 0, 0, 0, 21,
72         17, 16, 16, 20, 0, 17, 16, 16, 16, 16, 18, 18, 18, 18, 20,
73         17, 24, 24, 28, 0, 25, 24, 24, 16, 16, 16, 16, 16, 16, 20,
74         21, 0, 0, 0, 0, 0, 0, 0, 17, 16, 16, 16, 16, 16, 20,
75         17, 18, 18, 22, 0, 19, 18, 18, 16, 16, 16, 16, 16, 16, 20,
76         17, 16, 16, 20, 0, 17, 16, 16, 16, 16, 16, 16, 16, 16, 20,
77         17, 16, 16, 20, 0, 17, 16, 16, 16, 16, 16, 16, 16, 16, 20,
78         25, 24, 24, 24, 26, 24, 24, 24, 24, 24, 24, 24, 24, 24, 28
79     };
80 // khai báo ma trận lưu trữ thông tin cần thiết - giải thích ở phần sau
81     private final int validSpeeds[] = {1, 2, 3, 4, 6, 8};
82     // là một mảng chứa các giá trị tốc độ hợp lệ cho trò chơi. Trong trường hợp này,
83     private final int maxSpeed = 6;
84 // Biến "maxSpeed" lưu trữ giá trị tốc độ tối đa của trò chơi, ở đây là 6.
85     private int currentSpeed = 3;
86     //Biến "currentSpeed" lưu trữ tốc độ hiện tại của pacman trong trò chơi.
87     private short[] screenData;
88     // là một mảng lưu trữ dữ liệu màn hình, cụ thể là các phần tử có kiểu dữ liệu "short".
89     private Timer timer;
90     // là một đối tượng "Timer" được sử dụng để quản lý thời gian trong trò chơi.
91     private int maxScore = 194;
92     // là giới hạn điểm số cao nhất có thể đạt được trong trò chơi.
93     private int DELAY;
94     // lưu trữ thời gian trễ giữa các khung hình trong trò chơi
95     private int[][] maze;
96     // Biến "maze" là một mảng hai chiều chứa dữ liệu về bản đồ mê cung của trò chơi.

```

Ảnh 23: Khởi tạo các biến #2

Giải thích mảng levelData[]:

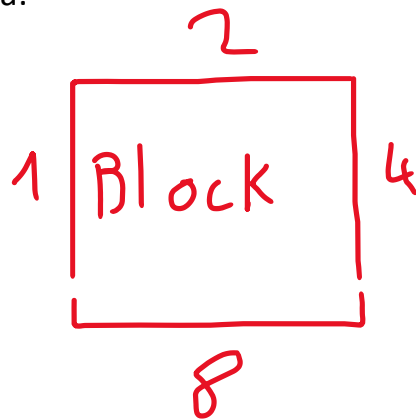
Ở đây các bit chính được sử dụng là:

- | | |
|---------------------------|----------------------|
| 0: Các ô được tô màu xanh | 1: Cạnh trái của 1 ô |
| 2: Cạnh trên của 1 ô | 4: Cạnh phải của 1 ô |
| 8: Cạnh dưới của 1 ô | 16: Các chấm điểm |

Cơ chế:

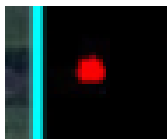
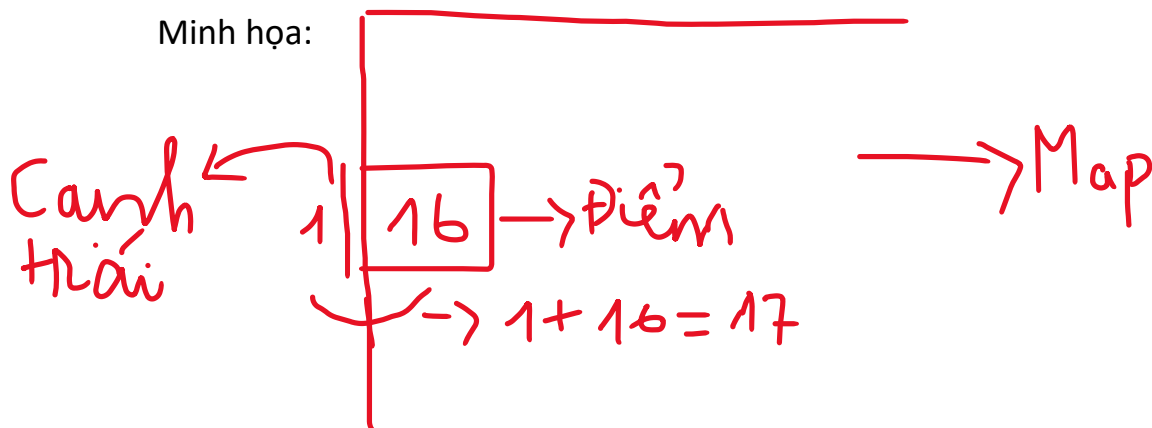
- Xây dựng bản đồ theo các bit trên:

Minh họa:

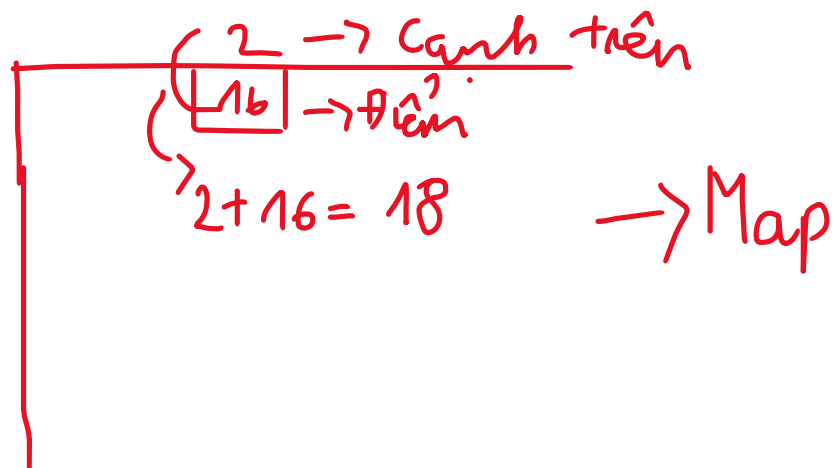


+ Bản đồ được bao bọc xung quanh bởi các **tường**, ví dụ **tường** bên trái sẽ được gán bit 1, nếu bên cạnh đó là 1 chấm điểm thì lấy $1 + 16 = 17$. Để tránh việc Pacman ăn điểm và đi lộn qua map

Minh họa:

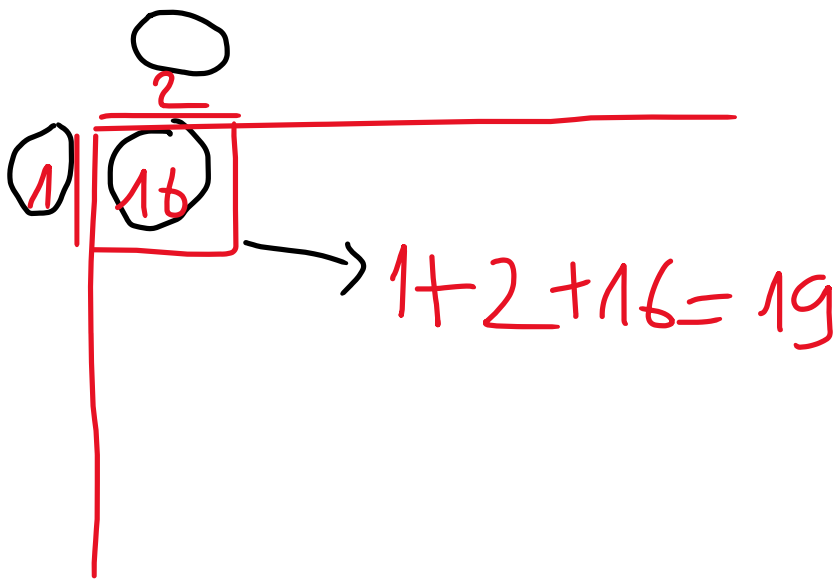


+ Tương tự, **tường** bên trên được gán bit 2, nếu là chấm điểm thì lấy $2 + 16$ là 18.





+ Nếu chấm điểm tiếp xúc với **tường** bên trái và bên trên, lấy bit của 2 **tường (trái + trên)** cộng với bit điểm: $1 + 2 + 16 = 19$

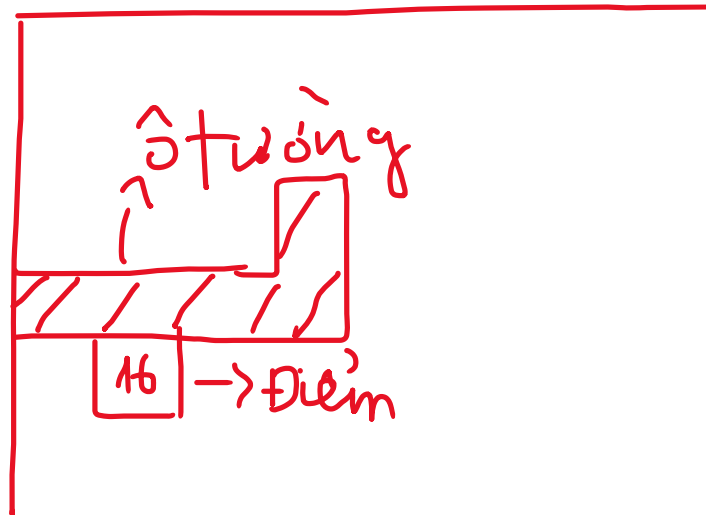
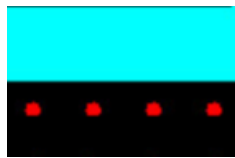


+ Nếu không có điểm mà muốn tạo ra các tường, các ô tường sẽ được tô kín màu thì các ô bên trong sẽ được gán bit 0, tuy nhiên nếu điểm nằm dưới ô tường này thì chúng ta vẫn sẽ cộng theo quy tắc bên trên, coi các tường là các khung và cộng theo vị trí của điểm so với tường



Ví dụ: Nếu điểm nằm dưới 1 ô tường, tương ứng với việc cạnh trên của ô điểm là tường

-> bit của điểm lúc này là: $16 + 2 = 18$



Tương tự, nếu bên trong map, các điểm tiếp xúc với các border thì sử dụng quy tắc trên để cộng bit, điều này giúp việc pacman khi ăn điểm và di chuyển hướng vào ô tường thì sẽ bị chặn lại, không di chuyển xuyên qua tường được

2. Xây dựng các hàm

2.1. Hàm khởi tạo Constructor Board:

Được sử dụng để thiết lập các giá trị cho một số biến (như screenData, timer, maxScore, và maze) và gán một số giá trị khởi tạo cho các thuộc tính khác của lớp. Nó cũng gọi các phương thức loadSound(), loadImages(), initVariables(), và initGame() để khởi tạo các giá trị và tải các tài nguyên cần thiết cho game. Cuối cùng, nó cũng thêm một KeyListener để theo dõi các sự kiện nhập từ bàn phím.

```
public Board() {  
    loadSound();  
    loadImages();  
    initVariables();  
    addKeyListener(new TAdapter());  
    setFocusable(true);  
    initGame();  
}
```

Ảnh 24: Hàm khởi tạo

2.2. Hàm loadSound():

Tải tệp âm thanh từ mục resources

- Tạo một AudioInputStream từ tệp âm thanh đầu vào.

- Tạo một Clip để phát tệp âm thanh.
- Mở Clip bằng cách sử dụng clipBack.open(audioIn) và chơi tệp âm thanh liên tục bằng cách sử dụng clipBack.loop(Clip.LOOP_CONTINUOUSLY).

Nếu có bất kỳ lỗi nào xảy ra trong quá trình tải và phát tệp âm thanh, nó sẽ in ra stack trace của lỗi.

```
private void loadSound() {
    try {
        URL soundUrl = getClass().getResource("/resources/intro.wav");
        AudioInputStream audioIn = AudioSystem.getAudioInputStream(soundUrl);
        clipBack = AudioSystem.getClip();
        clipBack.open(audioIn);
        clipBack.loop(Clip.LOOP_CONTINUOUSLY);

    } catch (IOException | LineUnavailableException | UnsupportedAudioFileException e) {
        e.printStackTrace();
    }
}
```

Ảnh 25: Hàm loadSound

2.3. Hàm loadImages():

Tải tệp ảnh từ resource, tạo các biến tương ứng với các hình ảnh phù hợp

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

Ảnh 26: Hàm loadImages()

2.4. Hàm initVariables():

Khởi tạo các biến cần thiết của game

- screenData: mảng lưu trữ dữ liệu về các ô vuông của mê cung.
- d: kích thước của cửa sổ trò chơi.
- ghost_x, ghost_dx, ghost_y, ghost_dy, ghostSpeed: các biến lưu trữ thông tin về ma trong trò chơi.

- dx và dy: các biến lưu trữ thông tin về hướng đi của nhân vật.
- timer: đối tượng Timer dùng để cập nhật trạng thái của trò chơi sau một khoảng thời gian nhất định.

Sau khi khởi tạo các biến, hàm sẽ kích hoạt Timer để bắt đầu trò chơi

```
private void initVariables() {

    screenData = new short[N_BLOCKS * N_BLOCKS];
    d = new Dimension(380, 420);
    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(60, this);
    timer.start();
}
```

Ảnh 27: Hàm initVariables()

2.5. Hàm playGame:

- Xử lý các trạng thái của Pacman:
- Nếu Pacman chết thì sự kiện death() xảy ra
- Nếu không thì các sự kiện di chuyển (movePacman), vẽ pacman (drawPacman), di chuyển Ghost (moveGhosts), và kiểm tra mê cung xảy ra

```

private void playGame(Graphics2D g2d) {

    if (dying) {

        death();

    } else {

        movePacman();
        drawPacman(g2d);
        moveGhosts(g2d);
        checkMaze();

    }

}

```

Ảnh 28: Hàm playGame

2.6. Hàm showIntroScreen:

Tạo ra 1 chuỗi văn bản khi bắt đầu vào trò chơi, với màu sắc là vàng, vị trí hiển thị lúc bắt đầu là: $(SCREEN_SIZE) / 2 - 80$, $(SCREEN_SIZE) / 2$

```

private void showIntroScreen(Graphics2D g2d) {

    String start = "Press SPACE to start";
    g2d.setColor(Color.yellow);
    g2d.drawString(start, (SCREEN_SIZE) / 2 - 80, (SCREEN_SIZE) / 2);

}

```

Ảnh 29: Hàm showIntroScreen

2.7. Hàm drawScore:

Vẽ trạng thái điểm số và hình ảnh trái tim tương ứng với mạng

- Font chữ sử dụng là smallFont, được khởi tạo ban đầu
- Màu sắc là RGB (5, 181, 79)
- Điểm số được hiển thị cạnh chuỗi ký tự " Score "
- Vị trí hiển thị là: $SCREEN_SIZE / 2 + 96$, $SCREEN_SIZE + 16$
- Vòng lặp for: `for (int i = 0; i < lives; i++) {`
`g.drawImage(heart, i * 28 + 8, SCREEN_SIZE + 1, this);`
`}`

-> Đây là một vòng lặp for, được sử dụng để vẽ các hình ảnh trái tim trên màn hình. Vòng lặp này lặp lại số lượng lần tương đương với số mạng

sống của người chơi. Mỗi lần lặp, một hình ảnh trái tim sẽ được vẽ trên màn hình tại một vị trí cố định được tính toán bằng cách sử dụng biến `i`. Vị trí `x` của hình ảnh sẽ được tính bằng công thức " $i * 28 + 8$ ", trong đó 28 là khoảng cách giữa các hình ảnh trái tim và 8 là khoảng cách từ mép trái của màn hình. Vị trí `y` của hình ảnh là "`SCREEN_SIZE + 1`", nơi `SCREEN_SIZE` là kích thước của màn hình được định nghĩa là 400 pixel.

```
private void drawScore(Graphics2D g) {
    g.setFont(smallFont);
    g.setColor(new Color(5, 181, 79));
    String s = "Score: " + score;
    g.drawString(s, SCREEN_SIZE / 2 + 96, SCREEN_SIZE + 16);

    for (int i = 0; i < lives; i++) {
        g.drawImage(heart, i * 28 + 8, SCREEN_SIZE + 1, this);
    }
}
```

Ảnh 30: Hàm `drawScore`

2.8. Hàm `checkMaze`:

Đoạn mã được cung cấp là một phương thức có tên là `checkMaze()`, phục vụ cho trò chơi Pac-Man. Phương thức này sẽ kiểm tra xem lưới chơi đã được hoàn thành hay chưa.

- Phương thức này sử dụng một biến `i` để duyệt qua từng ô trong lưới chơi. Biến `N_BLOCKS` đại diện cho số lượng ô trên mỗi hàng và cột, và do đó, tổng số ô trong lưới chơi sẽ là `N_BLOCKS * N_BLOCKS`.
- Biến `finished` là một biến kiểu boolean được khởi tạo là `true`. Nếu tìm thấy bất kỳ ô nào chưa được hoàn thành (giá trị khác 0), biến `finished` sẽ được gán giá trị `false`.
- Phương thức này sử dụng một vòng lặp `while` để kiểm tra các ô trong lưới chơi. Trong vòng lặp, nếu tìm thấy ô chưa hoàn thành,
- Biến `finished` sẽ được gán giá trị `false`, và vòng lặp sẽ dừng lại. Nếu tất cả các ô đều đã hoàn thành, biến `finished` vẫn giữ nguyên giá trị `true`, và vòng lặp sẽ dừng lại khi `i` đạt đến giá trị `N_BLOCKS * N_BLOCKS`.
- Sau khi kiểm tra xong lưới chơi, nếu tất cả các ô đều đã hoàn thành, thì các hành động sau sẽ được thực hiện:
- Biến `score` sẽ được tăng thêm 50 điểm.
- Nếu số lượng ma (`N_GHOSTS`) chưa đạt tối đa (`MAX_GHOSTS`), thì số lượng ma sẽ được tăng thêm một.

- Nếu tốc độ hiện tại (currentSpeed) chưa đạt tối đa (maxSpeed), thì tốc độ hiện tại sẽ được tăng thêm một.
- Phương thức initLevel() sẽ được gọi để khởi tạo lại lượt chơi.

```
private void checkMaze() {

    int i = 0;
    boolean finished = true;

    while (i < N_BLOCKS * N_BLOCKS && finished) {

        if ((screenData[i]) != 0) {
            finished = false;
        }
        i++;
    }
    if (finished) {
        score += 50;
        if (N_GHOSTS < MAX_GHOSTS) {
            N_GHOSTS++;
        }
        if (currentSpeed < maxSpeed) {
            currentSpeed++;
        }
        initLevel();
    }
}
```

Ảnh 31: Hàm checkMaze

2.9. Hàm death:

Hàm này khiến số mạng bị giảm đi 1, nếu số mạng bằng 0 thì người chơi sẽ hết mạng và phương thức inGame sẽ sai, sau đó phương thức gameOver sẽ được gọi. Nếu lives khác 0 thì trò chơi sẽ tiếp tục bằng cách gọi phương thức continueLevel()

```
private void death() {  
  
    lives--;  
  
    if (lives == 0) {  
        inGame = false;  
        gameOver();  
    }  
  
    continueLevel();  
}
```

Ảnh 32: Hàm death

2.10. Hàm gameOver:

Được gọi khi số mạng sống còn lại bằng 0 hoặc khi người chơi đạt được mục tiêu điểm số tối đa.

- Nếu người chơi thua, một hộp thoại thông báo sẽ xuất hiện với điểm số của người chơi.
- Nếu người chơi thắng, một hộp thoại thông báo sẽ xuất hiện với thông báo "Bạn đã thắng" và điểm số của người chơi.
- Nếu người chơi đạt điểm số cao hơn điểm số cao nhất và điểm số cao nhất chưa đạt được, điểm số cao nhất sẽ được cập nhật và lưu vào cơ sở dữ liệu.
- Cuối cùng, số mạng sống còn lại sẽ giảm đi 1 và chương trình sẽ tiếp tục với màn chơi mới hoặc kết thúc nếu không còn mạng sống nào.

```

private void gameOver() {
    // Hiển thị hộp thoại thông báo với điểm số của người chơi
    if (highestScore == maxScore) {
        JOptionPane.showMessageDialog(
            this, "Bạn đã thắng: " + score,
            "Kết thúc game", JOptionPane.INFORMATION_MESSAGE);
    } else if (score > highestScore && highestScore != maxScore) {
        highestScore = score;
        accountDAO accountDAO = new accountDAO();
        try {
            accountDAO.updateScore(score, authenticated_account.getAccountCode());
        } catch (SQLException ex) {
            Logger.getLogger(Board.class.getName()).log(Level.SEVERE, null, ex);
        }
        JOptionPane.showMessageDialog(
            this, "Ki lúc mới: " + score,
            "Kết thúc game", JOptionPane.INFORMATION_MESSAGE);
    } else {
        JOptionPane.showMessageDialog(this, "Bạn đã thua! Điểm số của bạn là: " + score,
            "Kết thúc game", JOptionPane.INFORMATION_MESSAGE);
    }

    lives--;
    if (lives == 0) {
        gameOver = true;
    } else {
        continueLevel();
    }
}
}

```

Ảnh 33: Hàm gameOver

2.11. Hàm moveGhost:

Điều khiển việc di chuyển và hành động của các con ma trong game.

- Trong phương thức này, một vòng lặp for được sử dụng để lặp lại các con ma (số lượng con ma được quy định bởi biến N_GHOSTS).
- Đối với mỗi con ma, ta kiểm tra xem con ma có đang nằm trên một ô vuông (block) của mê cung hay không bằng cách kiểm tra xem giá trị tọa độ x và y của con ma có chia hết cho kích thước của ô vuông (BLOCK_SIZE) hay không.
- Nếu có, ta tính vị trí hiện tại của con ma trong mê cung bằng cách chia tọa độ x của con ma cho kích thước của ô vuông, cộng với tích của kích thước của mê cung và tọa độ y của con ma chia cho kích thước của ô vuông.
- Sau đó, ta kiểm tra xem con ma có thể di chuyển tới các ô vuông xung quanh nó hay không. Ta kiểm tra điều này bằng cách sử dụng các bit trong screenData,

- biểu thị trạng thái của mỗi ô vuông trong mê cung. Nếu bit 0 (tương ứng với giá trị 1) trong screenData[pos] bằng 0 và con ma không đang di chuyển sang phải (ghost_dx[i] != 1),
- ta cho con ma di chuyển sang trái bằng cách thiết lập dx[count] = -1 và dy[count] = 0. Tương tự, nếu bit 1 trong screenData[pos] bằng 0 và con ma không đang di chuyển xuống (ghost_dy[i] != 1), ta cho con ma di chuyển lên bằng cách thiết lập dx[count] = 0 và dy[count] = -1.
- Nếu bit 2 hoặc bit 3 trong screenData[pos] bằng 0 và con ma không đang di chuyển sang trái hoặc xuống (ghost_dx[i] != -1 hoặc ghost_dy[i] != -1),
- ta cho con ma di chuyển sang phải hoặc xuống tương ứng. Cuối cùng, nếu không có ô vuông nào xung quanh con ma mà con ma có thể di chuyển đến (count == 0),
- ta thay đổi hướng di chuyển của con ma bằng cách đổi dấu của ghost_dx[i] hoặc ghost_dy[i].
- Nếu có ô vuông nào xung quanh con ma mà con ma có thể di chuyển đến (count > 0), ta chọn một ô vuông ngẫu nhiên
- Sau đó, trong vòng lặp for, vị trí của mỗi con ma được cập nhật dựa trên vận tốc hiện tại của chúng và hướng di chuyển của chúng. Hàm drawGhost() được gọi để vẽ mỗi con ma lên màn hình.
- Cuối cùng, nếu Pacman chạm vào một con ma trong khi đang ở chế độ "inGame", biến dying được đặt thành true. Điều này sẽ làm cho Pacman rơi vào trạng thái dying và trò chơi sẽ kết thúc.

```

276
277 private void moveGhosts(Graphics2D g2d) {
278
279     int pos;
280     int count;
281
282     for (int i = 0; i < N_GHOSTS; i++) {
283         if (ghost_x[i] % BLOCK_SIZE == 0 && ghost_y[i] % BLOCK_SIZE == 0) {
284             pos = ghost_x[i] / BLOCK_SIZE + N_BLOCKS + (int) (ghost_y[i] / BLOCK_SIZE);
285
286             count = 0;
287
288             if ((screenData[pos] & 1) == 0 && ghost_dx[i] != 1) {
289                 dx[count] = -1;
290                 dy[count] = 0;
291                 count++;
292             }
293
294             if ((screenData[pos] & 2) == 0 && ghost_dy[i] != 1) {
295                 dx[count] = 0;
296                 dy[count] = -1;
297                 count++;
298             }
299
300             if ((screenData[pos] & 4) == 0 && ghost_dx[i] != -1) {
301                 dx[count] = 1;
302                 dy[count] = 0;
303                 count++;
304             }
305
306             if ((screenData[pos] & 8) == 0 && ghost_dy[i] != -1) {
307                 dx[count] = 0;
308                 dy[count] = 1;
309                 count++;
310             }
311
312             if (count == 0) {
313                 if ((screenData[pos] & 15) == 15) {
314                     ghost_dx[i] = 0;
315                     ghost_dy[i] = 0;
316                 } else {
317                     ghost_dx[i] = -ghost_dx[i];
318                     ghost_dy[i] = -ghost_dy[i];
319                 }
320             } else {
321                 count = (int) (Math.random() * count);
322                 if (count > 3) {
323                     count = 3;
324                 }
325                 ghost_dx[i] = dx[count];
326                 ghost_dy[i] = dy[count];
327             }
328
329             ghost_x[i] = ghost_x[i] + (ghost_dx[i] * ghostSpeed[i]);
330             ghost_y[i] = ghost_y[i] + (ghost_dy[i] * ghostSpeed[i]);
331             drawGhost(g2d, ghost_x[i] + 1, ghost_y[i] + 1);
332             if (pacman_x > (ghost_x[i] - 12) && pacman_x < (ghost_x[i] + 12)
333                 && pacman_y > (ghost_y[i] - 12) && pacman_y < (ghost_y[i] + 12)
334                 && inGame) {
335                 dying = true;
336             }
337         }
338     }
339 }

```

Ảnh 34: Hàm moveGhost

2.12. Hàm drawGhost:

Vẽ hình ảnh của ghost

```
private void drawGhost(Graphics2D g2d, int x, int y) { g2d.drawImage(ghost, x, y, this); }
```

Ảnh 35: Hàm drawGhost

2.13. Hàm movePacman:

Phương thức movePacman() sẽ thực hiện di chuyển Pac-Man. Các bước thực hiện của phương thức được mô tả chi tiết dưới đây:

- Khai báo biến pos và ch. Biến pos sẽ lưu trữ vị trí của Pac-Man trên mê cung, và biến ch sẽ lưu trữ dữ liệu của ô mà Pac-Man đang đứng trên đó.
- Kiểm tra xem Pac-Man có đang đứng tại một vị trí nguyên khối hay không. Nếu không phải, Pac-Man sẽ không di chuyển.
- Tính toán vị trí của Pac-Man trên mê cung và lưu vào biến pos. Lấy dữ liệu của ô mà Pac-Man đang đứng trên đó và lưu vào biến ch.
 - Nếu Pac-Man ăn được một điểm, thì giá trị tương ứng của ô đó sẽ được xoá đi (bằng cách thay đổi bit 5 của ch thành 0) và biến score sẽ được tăng lên 1.
 - Nếu bàn phím được nhấn (điều khiển di chuyển của Pac-Man), kiểm tra xem Pac-Man có thể di chuyển theo hướng đó không (có tường ở phía trước hay không). Nếu không có tường, Pac-Man sẽ di chuyển theo hướng đó.
 - Nếu Pac-Man đang đứng yên, kiểm tra xem có tường ở phía trước của Pac-Man không. Nếu có, Pac-Man sẽ đứng yên. Thực hiện di chuyển của Pac-Man dựa trên hướng di chuyển hiện tại và tốc độ di chuyển của Pac-Man (PACMAN_SPEED), sau đó đó cập nhật vị trí mới của Pac-Man trên mê cung.

```

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];

        if ((ch & 16) != 0) {
            screenData[pos] = (short) (ch & 15);
            score++;
        }
        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;
        }
    }
    pacman_x = pacman_x + PACMAN_SPEED * pacmand_x;
    pacman_y = pacman_y + PACMAN_SPEED * pacmand_y;
}

```

Ảnh 36: Hàm movePacman

Kiểm tra vị trí của pacman để tránh di chuyển xuyên tường: Đây là đoạn mã kiểm tra vị trí của Pacman và di chuyển Pacman trên màn hình.

Đầu tiên, nó sử dụng phương thức getMap() để lấy mã ASCII của khối ở vị trí của Pacman.

Nếu mã này là một trong số các giá trị được xác định trong các điều kiện if, thì nghĩa là Pacman đã chạm vào một tường và không thể di chuyển theo hướng đó.

Trong trường hợp này, pacmand_x và pacmand_y (biến quyết định hướng di chuyển của Pacman) được đặt thành 0 để Pacman dừng lại.

Nếu Pacman không đối mặt với tường, `pacman_x` và `pacman_y` được cập nhật để di chuyển Pacman theo hướng hiện tại của `pacmand_x` và `pacmand_y`, với tốc độ di chuyển là `PACMAN_SPEED`.

```
// 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;
}
}
pacman_x = pacman_x + PACMAN_SPEED * pacmand_x;
pacman_y = pacman_y + PACMAN_SPEED * pacmand_y;
}
```

Ảnh 37: Phương thức kiểm tra việc va chạm với tường

2.14. Hàm `drawPacman`:

Phương thức `drawPacman` được sử dụng để vẽ hình ảnh của nhân vật chính Pacman trên màn hình.

- Đầu tiên, phương thức sử dụng các biến `req_dx` và `req_dy` để xác định hướng di chuyển của Pacman. Nếu `req_dx` có giá trị bằng -1, nghĩa là Pacman đang di chuyển sang trái, phương thức sẽ vẽ hình ảnh của Pacman hướng sang trái (bằng cách sử dụng hình ảnh `left`).
- Tương tự, nếu `req_dx` có giá trị bằng 1, nghĩa là Pacman đang di chuyển sang phải, phương thức sẽ vẽ hình ảnh của Pacman hướng sang phải (bằng cách sử dụng hình ảnh `right`).
- Nếu `req_dy` có giá trị bằng -1, nghĩa là Pacman đang di chuyển lên, phương thức sẽ vẽ hình ảnh của Pacman hướng lên (bằng cách sử dụng hình ảnh `up`).
- Nếu không, Pacman đang di chuyển xuống, phương thức sẽ vẽ hình ảnh của Pacman hướng xuống (bằng cách sử dụng hình ảnh `down`).


```
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);
    }
}
```

Ảnh 38: Hàm drawPacman

2.15. Hàm drawMaze:

Được sử dụng để vẽ bản đồ của trò chơi PacMan

Hàm này nhận đầu vào là một đối tượng Graphics2D và sử dụng nó để vẽ các phần tử của bản đồ bao gồm các viên gạch, các đường vẽ giữa các viên gạch và các chấm ăn được. Hàm lặp lại qua từng phần tử của bản đồ và kiểm tra loại phần tử đó.

- Nếu phần tử đó là một viên gạch, hàm sẽ vẽ một hình vuông có kích thước là BLOCK_SIZE (kích thước của mỗi viên gạch) tại vị trí của viên gạch đó.

- Nếu phần tử đó là đường vẽ thì hàm sẽ vẽ một đường thẳng từ vị trí hiện tại của x và y đến vị trí x và y + BLOCK_SIZE - 1 (hoặc x + BLOCK_SIZE - 1 và y tùy thuộc vào loại đường vẽ).

- Nếu phần tử là một chấm ăn, hàm sẽ vẽ một hình tròn tại vị trí của chấm ăn đó.

Hàm này sử dụng các giá trị được lưu trữ trong mảng levelData để xác định loại của mỗi viên gạch và sử dụng các giá trị được lưu trữ trong mảng screenData để xác định loại đường vẽ và chấm ăn.

Cuối cùng, hàm sẽ tăng biến i lên một đơn vị để lặp lại qua từng phần tử của mảng.

```

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, 255, 255));
            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) {
                // Tìm ra vị trí của điểm mồi ở vị trí (2, 3)
                if (i == 23) {
                    g2d.setColor(new Color(0, 255, 0)); // Đổi màu điểm mồi thành màu xanh
                } else {
                    g2d.setColor(new Color(255, 0, 0));
                }
                g2d.fillOval(x + 10, y + 10, 6, 6);
            }
        }
        i++;
    }
}

```

Ảnh 39: Hàm drawMaze

- Chúng ta sử dụng một mảng screenData để lưu trữ thông tin về nội dung của từng ô trên màn hình. Trong vòng lặp, chúng ta kiểm tra bit thứ 5 của giá trị tại vị trí hiện tại của screenData[i] để xác định xem ô hiện tại có chứa điểm mồi hay không. Nếu có, chúng ta đặt màu và vẽ một hình tròn ở giữa của ô hiện tại để đại diện cho điểm mồi.
- Trong trường hợp ô hiện tại là ô điểm mồi cuối cùng (tức là ô vị trí 23), chúng ta sẽ đặt màu xanh cho điểm mồi để đánh dấu là đây là điểm mồi cuối cùng. Các điểm mồi khác sẽ có màu đỏ.

```

if ((screenData[i] & 16) != 0) {
    // Tìm ra vị trí của điểm mỗi ở vị trí (2, 3)
    if (i == 23) {
        g2d.setColor(new Color(0, 255, 0)); // Đổi màu điểm mỗi thành màu xanh
    } else {
        g2d.setColor(new Color(255, 0, 0));
    }
    g2d.fillOval(x + 10, y + 10, 6, 6);
}
i++;
}

```

Ảnh 40: Điểm mỗi cuối cùng

2.16. Hàm initGame:

Được sử dụng để khởi tạo trạng thái ban đầu của trò chơi khi bắt đầu một vòng mới.

- Cụ thể, phương thức này thiết lập số lượt chơi (lives) bằng 3, điểm số (score) bằng 0, và sử dụng phương thức initLevel() để khởi tạo màn chơi mới.
- Ngoài ra, phương thức còn thiết lập số lượng ma (N_GHOSTS) bằng 6, tốc độ chơi hiện tại (currentSpeed) bằng 3, và trạng thái tạm dừng (isPaused) bằng false.
- Các giá trị này được sử dụng để điều khiển hoạt động của trò chơi trong quá trình chơi.

```

private void initGame() {

    lives = 3;
    score = 0;
    highestScore = authenticated_account.getHigestScore();
    initLevel();
    N_GHOSTS = 6;
    currentSpeed = 3;
    isPaused = false;
}

```

Ảnh 41: Hàm initGame

2.17. Hàm initLevel:

Được sử dụng để khởi tạo màn chơi mới. Trong đó, hàm này sẽ sao chép dữ liệu màn chơi từ mảng levelData sang mảng screenData, với $N_BLOCKS * N_BLOCKS$ là tổng số lượng ô trong màn chơi. Sau đó, hàm continueLevel() được gọi để bắt đầu màn chơi.

```
private void initLevel() {  
  
    int i;  
    for (i = 0; i < N_BLOCKS * N_BLOCKS; i++) {  
        screenData[i] = levelData[i];  
    }  
  
    continueLevel();  
}
```

2.18. Hàm continueLevel:

Trong hàm, các giá trị ban đầu cho pacman và ghosts được đặt lại để chuẩn bị cho màn chơi mới.

- Cụ thể, vị trí ban đầu cho các con ma được đặt tại hàng thứ 4, vị trí của pacman được đặt tại hàng thứ 11 và cả hai đều được đặt tại giữa màn hình. Hướng di chuyển của pacman cũng được đặt lại thành không di chuyển và hướng điều khiển cũng được đặt lại.
- Cuối cùng, biến dying cũng được đặt lại thành giá trị false.

```

private void continueLevel() {

    int dx = 1;
    int random;

    for (int i = 0; i < N_GHOSTS; i++) {

        ghost_y[i] = 4 * BLOCK_SIZE; //start position
        ghost_x[i] = 4 * BLOCK_SIZE;
        ghost_dy[i] = 0;
        ghost_dx[i] = dx;
        dx = -dx;
        random = (int) (Math.random() * (currentSpeed + 1));

        if (random > currentSpeed) {
            random = currentSpeed;
        }

        ghostSpeed[i] = validSpeeds[random];
    }

    pacman_x = 7 * BLOCK_SIZE; //start position
    pacman_y = 11 * BLOCK_SIZE;
    pacmand_x = 0; //reset direction move
    pacmand_y = 0;
    req_dx = 0; // reset direction controls
    req_dy = 0;
    dying = false;
}

```

Ảnh 42: Hàm continueLevel

2.19. Hàm paintComponent:

Phương thức paintComponent(Graphics g) được gọi để vẽ trò chơi lên màn hình. Nó được kế thừa từ lớp JPanel. Các bước để vẽ trò chơi như sau:

- Đầu tiên, phương thức super.paintComponent(g) được gọi để xóa các vẽ cũ trên màn hình.
- Tiếp theo, một đối tượng Graphics2D được khởi tạo từ đối tượng g.
- Màu nền đen được thiết lập và một hình chữ nhật được vẽ lên màn hình để xóa mọi nội dung cũ.
- Hình maze được vẽ lên màn hình bằng phương thức drawMaze().

- Điểm số của người chơi được vẽ lên màn hình bằng phương thức `drawScore()`.
- Nếu trò chơi đang diễn ra, phương thức `playGame()` được gọi để vẽ các thành phần trò chơi.
- Nếu trò chơi chưa bắt đầu hoặc đã kết thúc, màn hình giới thiệu được vẽ bằng phương thức `showIntroScreen()`.
- Cuối cùng, `Toolkit.getDefaultToolkit().sync()` được gọi để đồng bộ hóa với hệ thống đồ họa và `g2d.dispose()` được gọi để giải phóng bộ nhớ và tài nguyên đồ họa.

```
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();
}
```

Ảnh 43: Hàm `paintComponent`

2.20. Xử lý sự kiện `KeyListener`:

Đây là một lớp được định nghĩa bên trong lớp `PacMan` để xử lý các sự kiện bấm phím từ người dùng khi đang chơi hoặc không chơi trò chơi. Lớp này kế thừa từ lớp `KeyListener` và ghi đè phương thức `keyPressed()` để xử lý các sự kiện bấm phím. Nếu trong trò chơi, các phím mũi tên được sử dụng để điều khiển Pac-Man, phím cách được sử dụng để bắt đầu trò chơi hoặc chơi lại khi Pac-Man chết và phím Enter được sử dụng để tạm dừng hoặc tiếp tục trò chơi. Nếu không trong trò chơi, phím cách

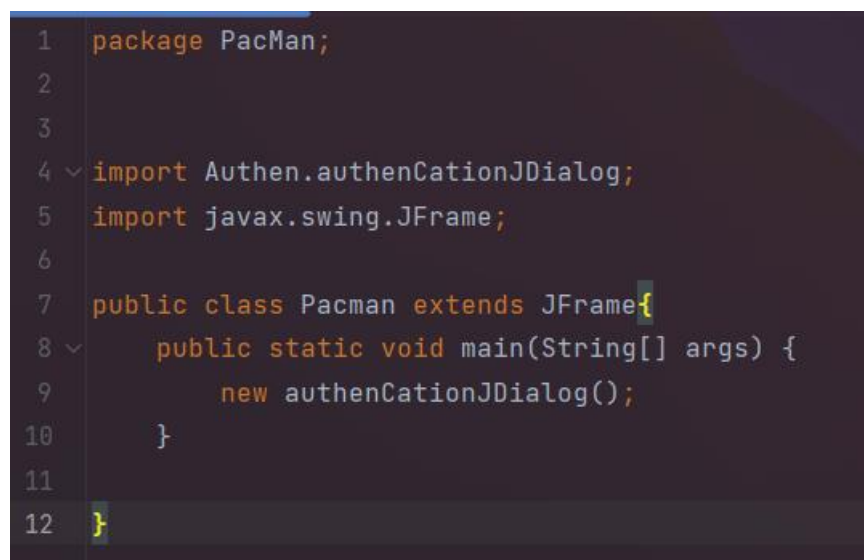
được sử dụng để bắt đầu trò chơi và phím Enter được sử dụng để tạm dừng hoặc tiếp tục trò chơi.



Ảnh 44: Hàm Tadapter, xử lý sự kiện bấm phím

3. Lớp Pacman:

Đây là một chương trình Java Swing để chơi trò chơi Pac-Man. Trong main() phương thức của lớp Pacman, một đối tượng authenCationJDialog được khởi tạo. Đây là một hộp thoại xác thực đăng nhập, có thể được sử dụng để yêu cầu người chơi nhập tên đăng nhập và mật khẩu trước khi trò chơi bắt đầu. Sau khi người chơi xác thực thành công, trò chơi Pac-Man sẽ được khởi động và hiển thị trên một cửa sổ mới.



Ảnh 45: Lớp Pacman

IV. Tổng kết

Kết luận

- Như vậy Pacman là 1 tựa game tương đối dễ chơi, không quá phức tạp. Bạn chỉ cần có sự khéo léo và nhanh tay thì có thể hoàn thành trò chơi 1 cách dễ dàng
- Về ưu điểm của code: Game hoạt động tương đối mượt, không giật, đã vận dụng tốt hầu hết các thuật toán cơ bản của lập trình hướng đối tượng, hệ thống cơ sở dữ liệu chạy tốt
- Về nhược điểm: Hệ thống cơ sở dữ liệu chưa mở rộng được cho nhiều máy khác nhau mà chỉ hoạt động trên 1 máy, phân chia các phương thức chưa được tốt do gói hết vào 1 class. Nếu phát triển game thêm thì sẽ gây rối code
- Hướng phát triển: Tham khảo học tập và rèn luyện thêm các kiến thức lập trình để hoàn thiện game hơn,..

Tài liệu tham khảo:

1. <https://stackoverflow.com/questions/13573281/how-to-play-a-background-music-when-the-program-run-in-java>
2. <https://stackoverflow.com/questions/42895933/how-can-i-make-an-object-move-randomly>
3. <https://www.javatpoint.com/steps-to-connect-to-the-database-in-java>
4. <https://www.javatpoint.com/java-awt>