**\*Thành viên nhóm:**

**1. Phạm Thị Mỹ Hạnh**

**2. Phan Trần Phong  
3. Phương Hữu Gia Lộc**

**4. Lê Hữu Phước**

**5. Lê Tấn Quân**

**6. Đặng Thị Ngọc Tiên  
7. Võ Duy Khang**

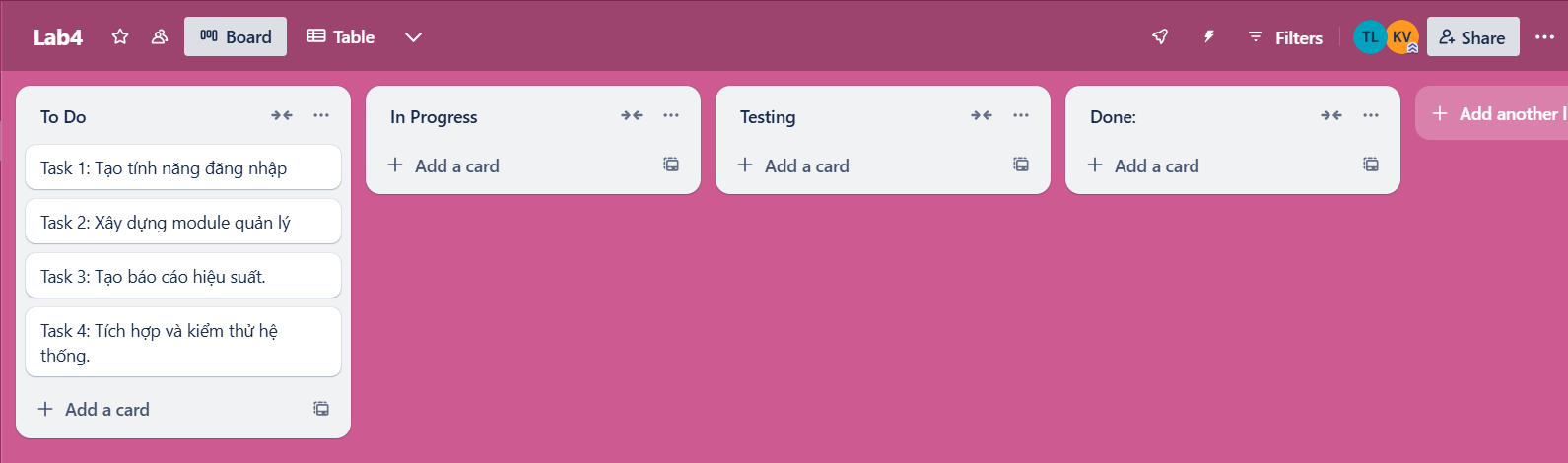
**BÀI LÀM  
Tình huống mô tả**

Công ty XYZ, chuyên về dịch vụ nhân sự, yêu cầu nâng cấp hệ thống quản lý thông tin nội bộ (MIS) với các tính năng sau:

1. **Chức năng Đăng nhập Nâng Cao**
   * Phân quyền Admin và User.
   * Khóa tài khoản sau 3 lần nhập sai.
   * Ghi log hoạt động đăng nhập.
2. **Chức năng Quản lý Nhân viên** 
   * Admin: Thêm, sửa, xóa thông tin nhân viên.
   * User: Chỉ xem thông tin cá nhân.
3. **Chức năng Báo Cáo Hiệu Suất**
   * Admin: Xem báo cáo hiệu suất toàn bộ công ty hoặc theo phòng ban.
   * User: Chỉ xem được báo cáo cá nhân.
4. **Ghi Log Hệ Thống**
   * Lưu trữ tất cả các hoạt động như: đăng nhập, thêm/sửa/xóa nhân viên, xem báo cáo.
   * Phân tích log để phát hiện lỗi hoặc hành vi bất thường.

**Nội dung Thực hành**

**Phần 1: Quản lý Yêu Cầu Thay Đổi**

**Bài tập 1: Lập kế hoạch trong Trello/Jira**

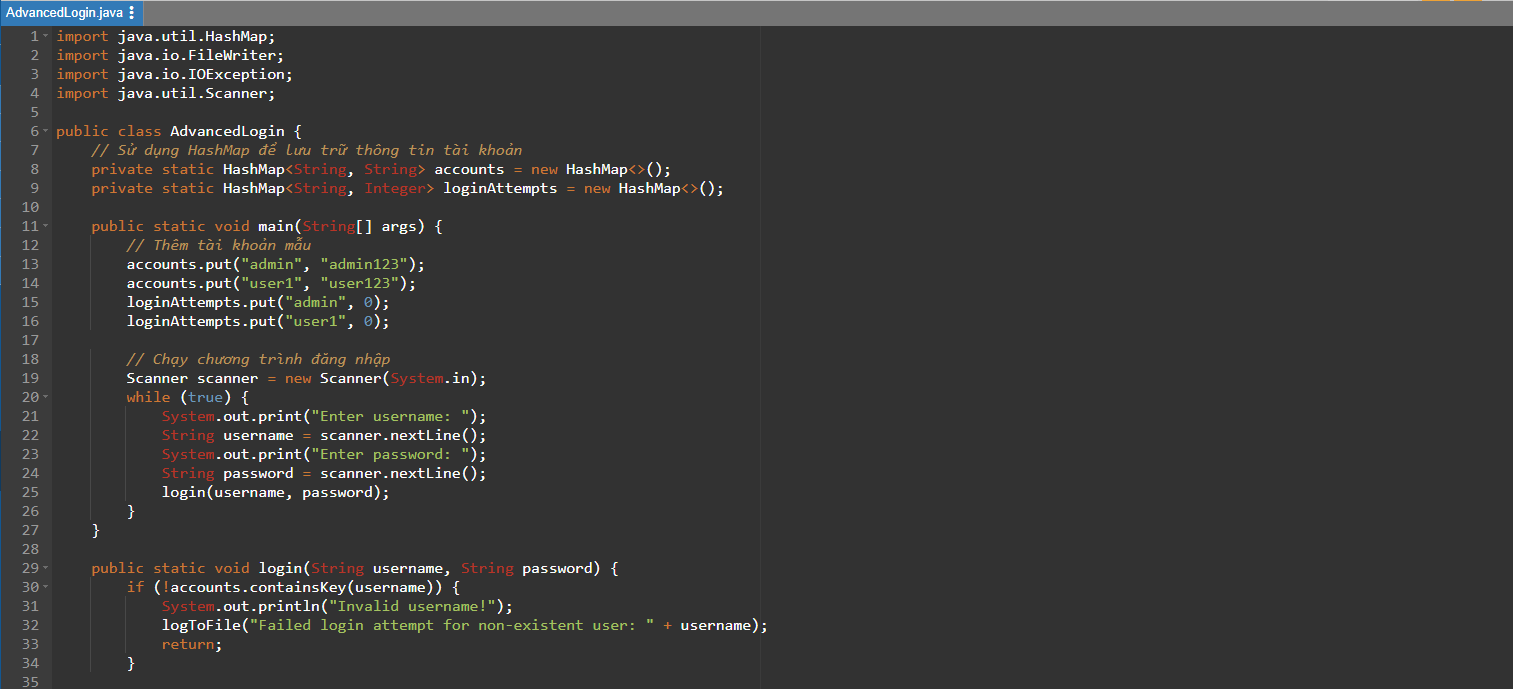
**Câu hỏi:**

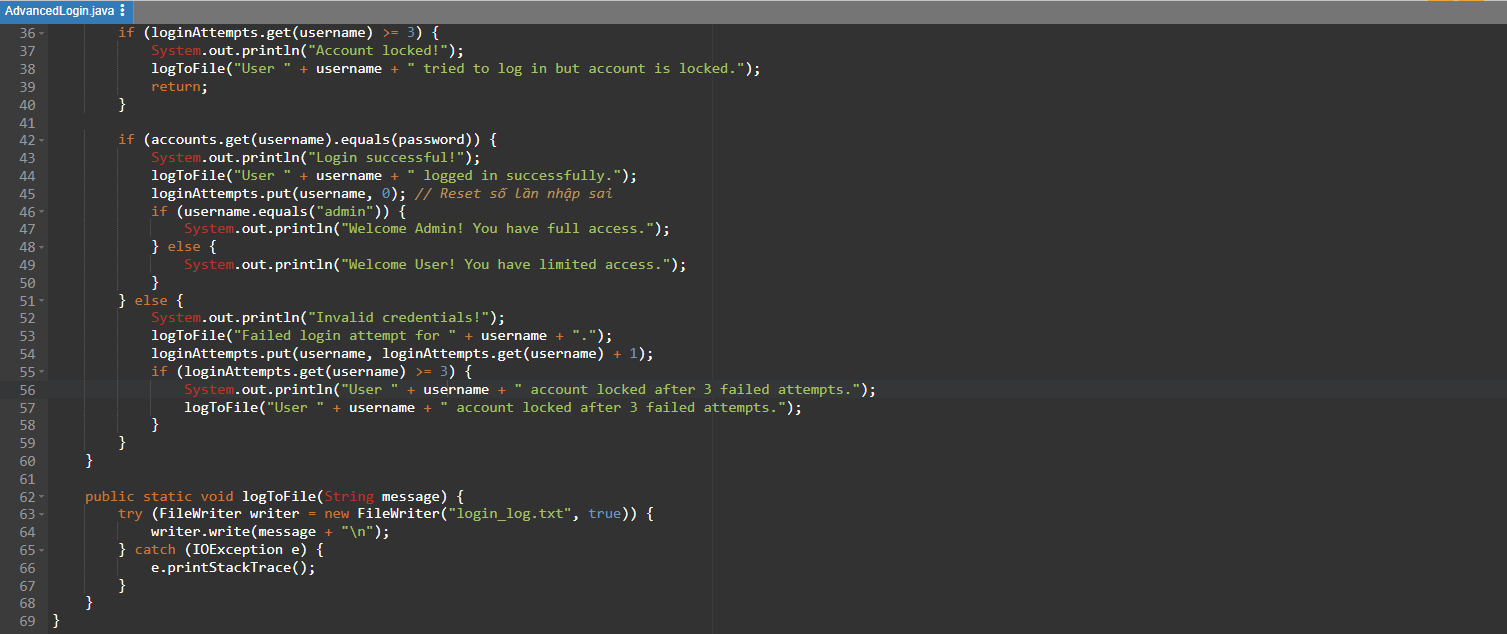
1. **Làm thế nào để phân chia công việc hiệu quả trong nhóm?**
   * Xác định các nhiệm vụ rõ ràng, giao cho thành viên dựa trên kỹ năng.
   * Sử dụng các công cụ như Trello/Jira để theo dõi tiến độ.
   * Lên lịch họp định kỳ để cập nhật tình hình.
2. **Nếu nhiệm vụ gặp vấn đề trong kiểm thử, bạn sẽ xử lý thế nào?**
   * Ghi nhận lỗi cụ thể và nguyên nhân xảy ra lỗi.
   * Ưu tiên sửa lỗi, thông báo cho các thành viên liên quan.
   * Nếu lỗi nghiêm trọng sẽ điều chỉnh lại kế hoạch.

**Phần 2: Thực hiện Tính Năng**

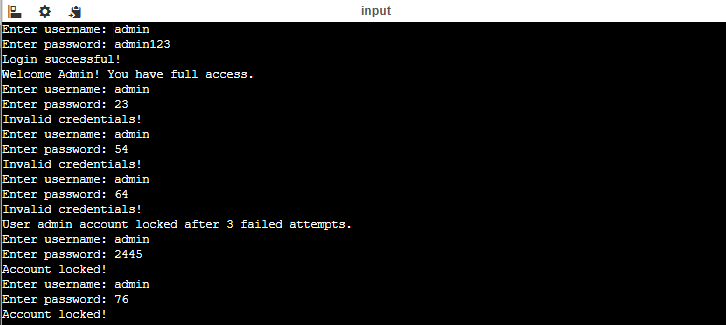
**Bài tập 2: Đăng Nhập Nâng Cao**

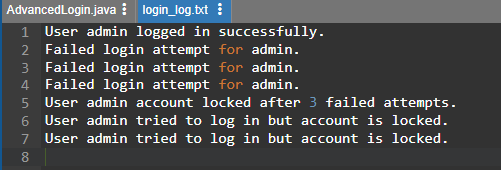
Viết chương trình hoàn chỉnh cho chức năng đăng nhập nâng cao theo yêu cầu trên.

**Đoạn mã:**



Kiểm tra:

* Đăng nhập thành công.
* Đăng nhập thất bại.
* Tài khoản bị khóa sau 3 lần nhập sai.

Ghi log vào file:

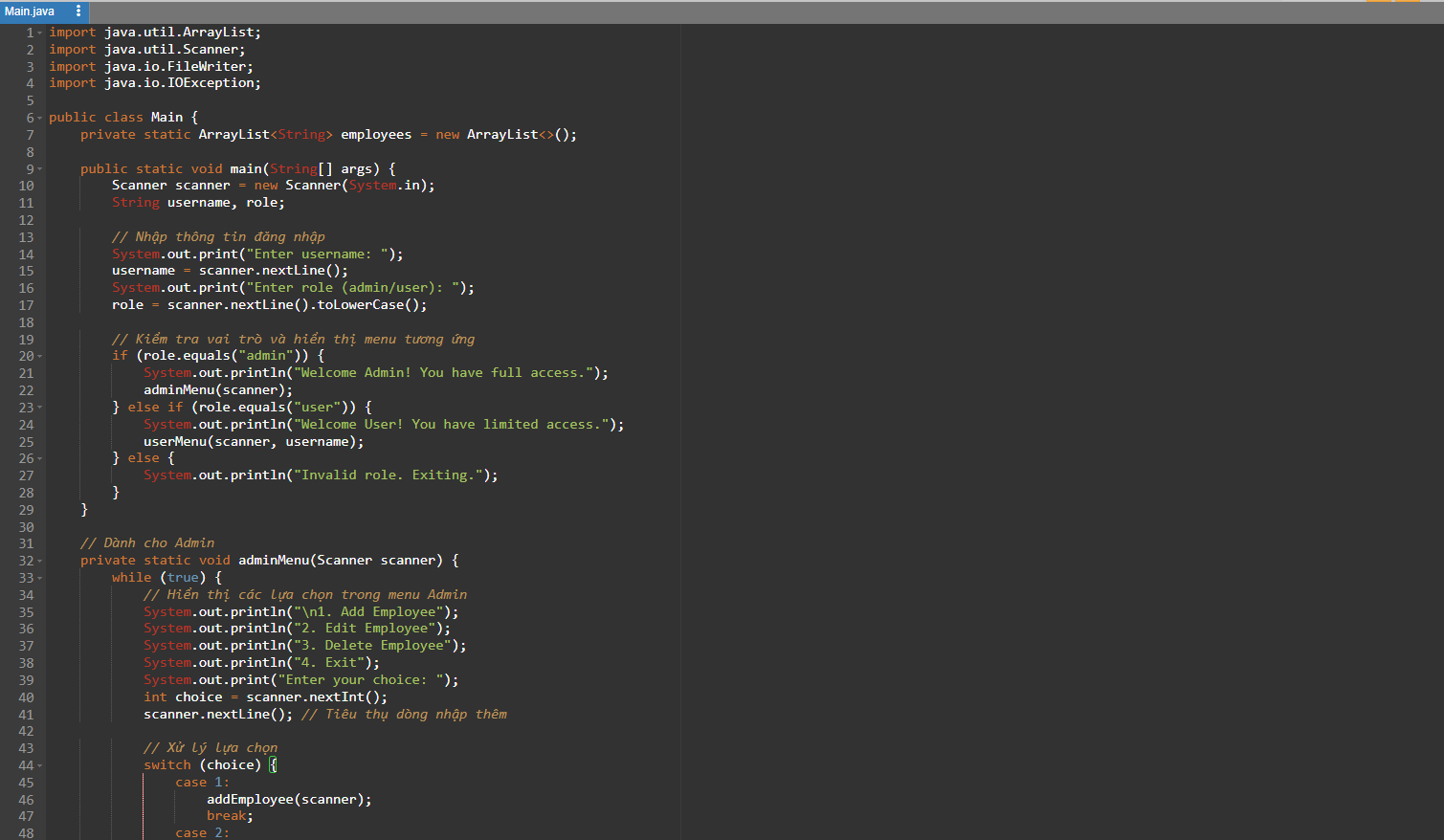
**Bài tập 3: Quản lý Nhân Viên**

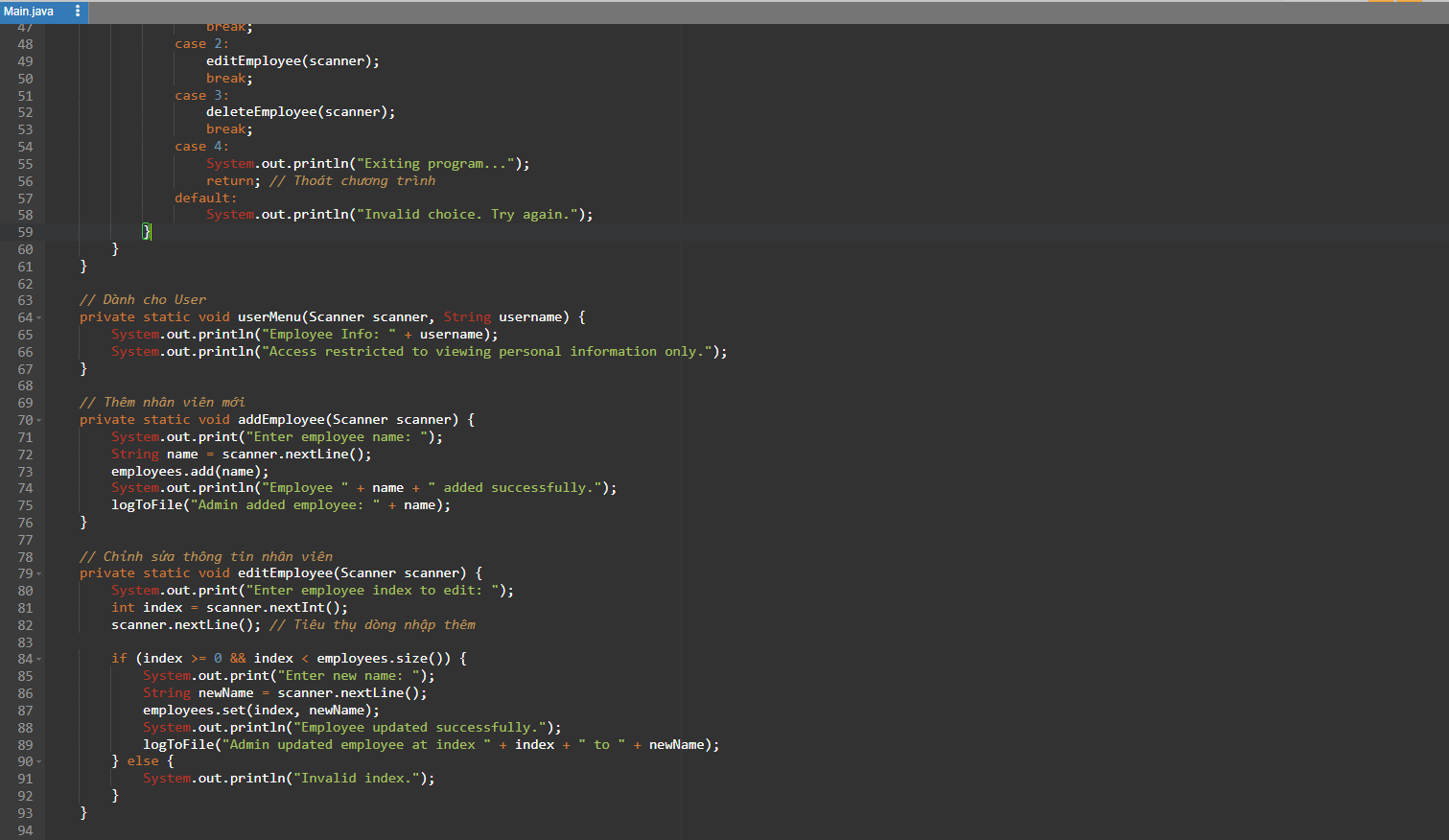
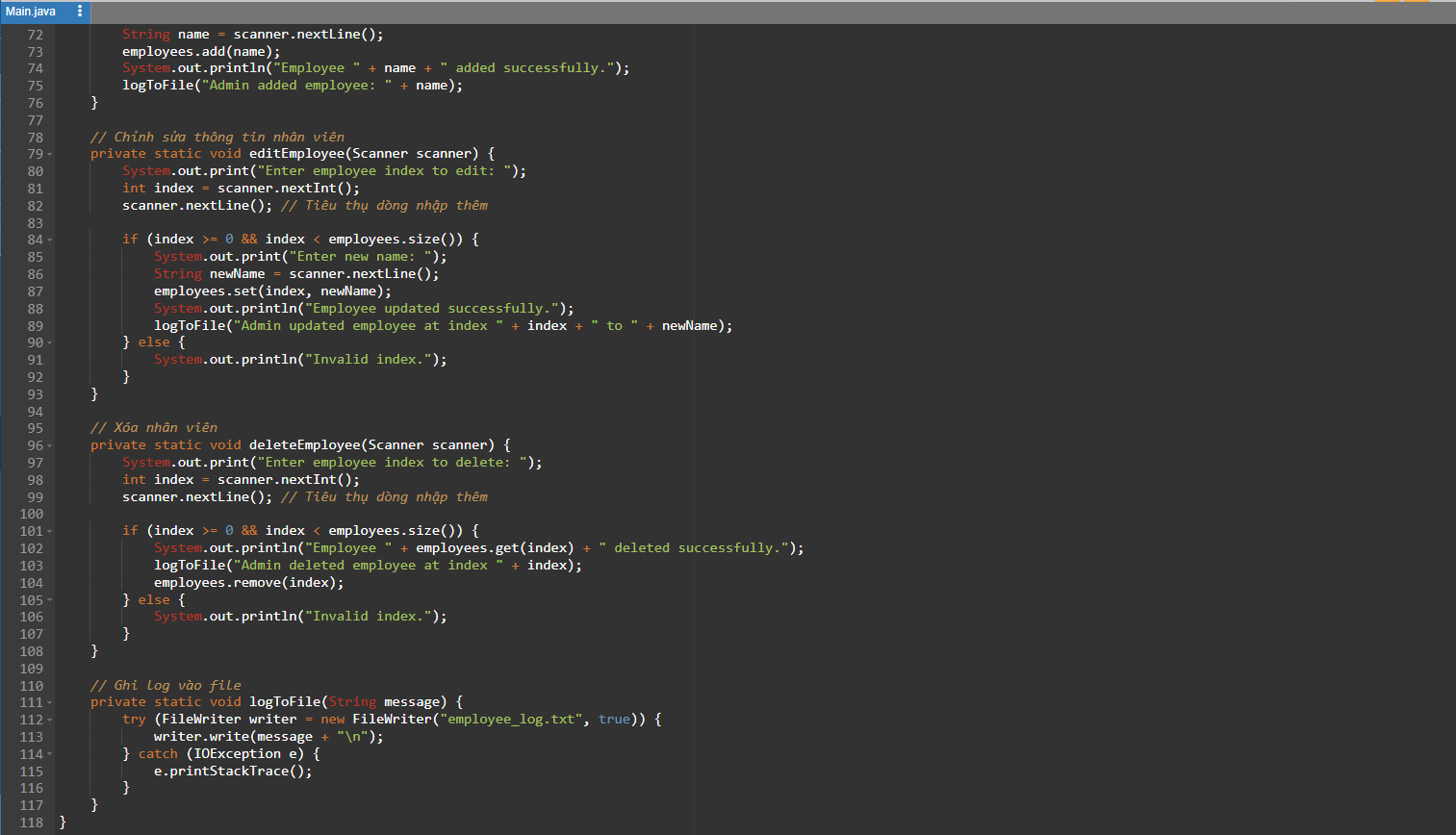
**Yêu cầu:**

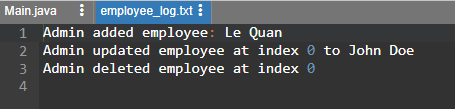
1. Xây dựng module quản lý nhân viên:

* Admin: Thêm, sửa, xóa nhân viên.
* User: Chỉ được xem thông tin cá nhân.

2. Ghi log hoạt động quản lý nhân viên:

* Ví dụ: "Admin admin added employee John Doe."





**Câu hỏi:**

1. Làm thế nào để phân quyền hiệu quả giữa Admin và User?

* Phân quyền hiệu quả giữa Admin và User bằng cách kiểm tra vai trò của người dùng sau khi họ đăng nhập. Nếu là Admin, cho phép thực hiện các chức năng thêm, sửa, xóa nhân viên. Nếu là User, chỉ cho phép xem thông tin cá nhân.

2. Nếu cần lưu trữ thông tin nhân viên vào file, bạn sẽ thực hiện như thế nào?

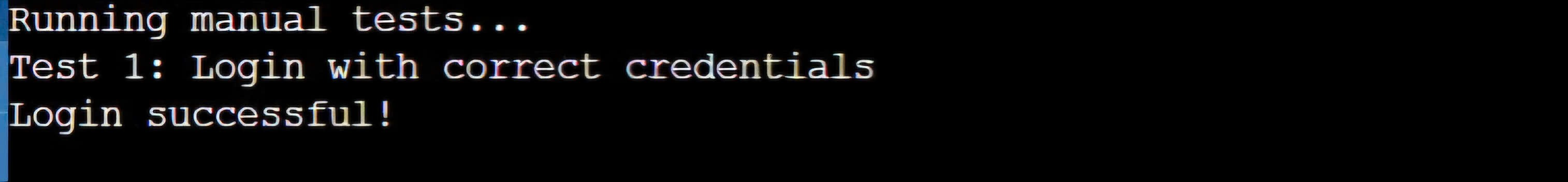
* Nếu cần lưu trữ thông tin nhân viên vào file, em sẽ sử dụng các định dạng đơn giản như CSV hoặc JSON. Mỗi nhân viên sẽ có thông tin như tên, chức vụ, lương, và các thông tin khác được lưu vào file theo định dạng này. Khi thêm, sửa, hoặc xóa nhân viên, hệ thống cần phải cập nhật lại nội dung trong file để đảm bảo dữ liệu luôn được lưu trữ chính xác. Hoặc sử dụng lớp FileWriter để ghi dữ liệu vào file. Ngoài ra, cũng cần ghi log các hoạt động của Admin vào một file log riêng biệt để theo dõi các thay đổi trong hệ thống.

**Phần 3: Kiểm Thử và Đánh Giá**

1. **Bảng Kiểm thử**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Chức**  **năng** | **Trường hợp**  **kiểm thử** | **Dữ liệu đầu vào** | **Kết quả mong đợi** | **Trạng**  **thái** |
| 1 | Đăng nhập | Đăng nhập  thành công với Admin | Username: admin, Password: admin123 | "Login successful!" | Pass |
| 2 | Đăng nhập | Đăng nhập thất  bại (sai mật khẩu) | Username: user1, Password: wrong | "Invalid credentials!" | Pass |
| 3 | Đăng nhập | Tài khoản bị  khóa sau 3 lần nhập sai | Username: user1, Password: wrong (x3) | "Account locked!" | Pass |
| 4 | Đăng nhập | Đăng nhập với tài khoản  không tồn tại | Username: unknown, Password: 1234 | "Invalid username!" | Pass |
| 5 | Quản lý nhân  viên | Thêm nhân viên mới | Tên: John Doe, ID: E001 | "Employee added successfully." | Pass |
| 6 | Quản lý nhân  viên | Sửa thông tin nhân viên | ID: E001, Email:  [john.doe@example.com](mailto:john.doe@example.com) | "Employee updated successfully." | Pass |
| 7 | Quản lý nhân  viên | Xóa nhân viên | ID: E001 | "Employee deleted successfully." | Pass |
| 8 | Quản lý nhân viên | User thử xóa nhân viên | Username: user1, ID: E001 | "Access Denied!" | Pass |
| 9 | Ghi Log | Kiểm tra log sau khi đăng nhập thành  công | Username: admin, Password: admin123 | Log: "User admin logged in successfully." | Pass |

1. **Đăng nhập**
   1. Đăng nhập thành công với Admin



* 1. Đăng nhập thất bại (sai mật khẩu)

A black background with white text

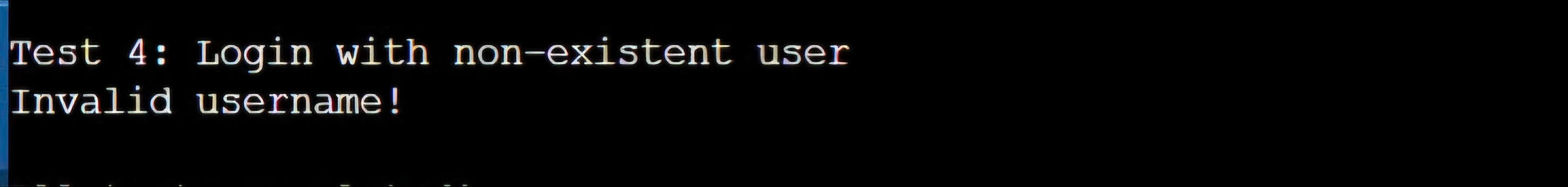
Description automatically generated

* 1. Tài khoản bị khóa sau 3 lần nhập sai

A black screen with white text

Description automatically generated

* 1. Đăng nhập với tài khoản không tồn tại



1. **Quản lý nhân viên**
   1. Thêm nhân viên mới

A screen shot of a computer

Description automatically generated

* 1. Sửa thông tin nhân viên

A screen shot of a computer

Description automatically generated

* 1. Xóa nhân viên

A black screen with white text

Description automatically generated

* 1. User thử xóa nhân viên

A screen shot of a computer screen

Description automatically generated

1. **Ghi log**

Kiểm tra log sau khi đăng nhập thành công

A black background with white text

Description automatically generated

1. **Mã nguồn hoàn chỉnh**
   1. Mã chính

Code Đăng nhập

import java.util.HashMap;

import java.io.FileWriter;

import java.io.IOException;

import java.util.Scanner;

public class AdvancedLogin {

private static HashMap<String, String> accounts = new HashMap<>();

private static HashMap<String, Integer> loginAttempts = new HashMap<>();

public static void main(String[] args) {

// Thêm tài khoản mẫu

accounts.put("admin", "admin123");

accounts.put("user1", "user123");

loginAttempts.put("admin", 0);

loginAttempts.put("user1", 0);

// Chạy chương trình đăng nhập

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.print("Enter username: ");

String username = scanner.nextLine();

System.out.print("Enter password: ");

String password = scanner.nextLine();

login(username, password);

}

}

public static void login(String username, String password) {

if (!accounts.containsKey(username)) {

System.out.println("Invalid username!");

logToFile("Failed login attempt for non-existent user: " + username);

return;

}

if (loginAttempts.get(username) >= 3) {

System.out.println("Account locked!");

logToFile("User " + username + " tried to log in but account is locked.");

return;

}

if (accounts.get(username).equals(password)) {

System.out.println("Login successful!");

logToFile("User " + username + " logged in successfully.");

loginAttempts.put(username, 0); // Reset số lần nhập sai

if (username.equals("admin")) {

System.out.println("Welcome Admin! You have full access.");

} else {

System.out.println("Welcome User! You have limited access.");

}

} else {

System.out.println("Invalid credentials!");

logToFile("Failed login attempt for " + username + ".");

loginAttempts.put(username, loginAttempts.get(username) + 1);

if (loginAttempts.get(username) >= 3) {

System.out.println("User " + username + " account locked after 3 failed attempts.");

logToFile("User " + username + " account locked after 3 failed attempts.");

}

}

}

public static void logToFile(String message) {

try (FileWriter writer = new FileWriter("login\_log.txt", true)) {

writer.write(message + "\n");

} catch (IOException e) {

e.printStackTrace();

}

}

}

Code Quản lý nhân viên:

import java.util.ArrayList;

import java.util.Scanner;

import java.io.FileWriter;

import java.io.IOException;

public class Main {

private static ArrayList<String> employees = new ArrayList<>();

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String username, role;

// Nhập thông tin đăng nhập

System.out.print("Enter username: ");

username = scanner.nextLine();

System.out.print("Enter role (admin/user): ");

role = scanner.nextLine().toLowerCase();

// Kiểm tra vai trò và hiển thị menu tương ứng

if (role.equals("admin")) {

System.out.println("Welcome Admin! You have full access.");

adminMenu(scanner);

} else if (role.equals("user")) {

System.out.println("Welcome User! You have limited access.");

userMenu(scanner, username);

} else {

System.out.println("Invalid role. Exiting.");

}

}

// Menu dành cho Admin

private static void adminMenu(Scanner scanner) {

while (true) {

// Hiển thị các lựa chọn trong menu Admin

System.out.println("\n1. Add Employee");

System.out.println("2. Edit Employee");

System.out.println("3. Delete Employee");

System.out.println("4. Exit");

System.out.print("Enter your choice: ");

int choice = scanner.nextInt();

scanner.nextLine(); // Tiêu thụ dòng nhập thêm

// Xử lý lựa chọn

switch (choice) {

case 1:

addEmployee(scanner);

break;

case 2:

editEmployee(scanner);

break;

case 3:

deleteEmployee(scanner);

break;

case 4:

System.out.println("Exiting program...");

return; // Thoát chương trình

default:

System.out.println("Invalid choice. Try again.");

}

}

}

// Menu dành cho User

private static void userMenu(Scanner scanner, String username) {

System.out.println("Employee Info: " + username);

System.out.println("Access restricted to viewing personal information only.");

}

// Thêm nhân viên mới

private static void addEmployee(Scanner scanner) {

System.out.print("Enter employee name: ");

String name = scanner.nextLine();

employees.add(name);

System.out.println("Employee " + name + " added successfully.");

logToFile("Admin added employee: " + name);

}

// Chỉnh sửa thông tin nhân viên

private static void editEmployee(Scanner scanner) {

System.out.print("Enter employee index to edit: ");

int index = scanner.nextInt();

scanner.nextLine(); // Tiêu thụ dòng nhập thêm

if (index >= 0 && index < employees.size()) {

System.out.print("Enter new name: ");

String newName = scanner.nextLine();

employees.set(index, newName);

System.out.println("Employee updated successfully.");

logToFile("Admin updated employee at index " + index + " to " + newName);

} else {

System.out.println("Invalid index.");

}

}

// Xóa nhân viên

private static void deleteEmployee(Scanner scanner) {

System.out.print("Enter employee index to delete: ");

int index = scanner.nextInt();

scanner.nextLine(); // Tiêu thụ dòng nhập thêm

if (index >= 0 && index < employees.size()) {

System.out.println("Employee " + employees.get(index) + " deleted successfully.");

logToFile("Admin deleted employee at index " + index);

employees.remove(index);

} else {

System.out.println("Invalid index.");

}

}

// Ghi log vào file

private static void logToFile(String message) {

try (FileWriter writer = new FileWriter("employee\_log.txt", true)) {

writer.write(message + "\n");

} catch (IOException e) {

e.printStackTrace();

}

}

}

* 1. Mã kiểm thử

import org.junit.Test;

import static org.junit.Assert.\*;

public class MainTest {

@Test

public void testAddEmployee() {

ArrayList<String> employees = new ArrayList<>();

String employeeName = "John Doe";

employees.add(employeeName);

assertTrue("Employee should be added", employees.contains(employeeName));

}

@Test

public void testEditEmployee() {

ArrayList<String> employees = new ArrayList<>();

String employeeName = "John Doe";

employees.add(employeeName);

int index = employees.indexOf(employeeName);

employees.set(index, "Jane Doe");

assertEquals("Employee name should be updated", "Jane Doe", employees.get(index));

}

@Test

public void testDeleteEmployee() {

ArrayList<String> employees = new ArrayList<>();

String employeeName = "John Doe";

employees.add(employeeName);

employees.remove(employeeName);

assertFalse("Employee should be removed", employees.contains(employeeName));

}

}

**Bài tâp 4: Sinh viên thực hiện các bước sau để kiểm thử các chức năng trong hệ thống:**

**2. Ghi Nhận Kết Quả và Đề Xuất Cải Tiến**

**Bảng Kết Quả:**

| **STT** | **Chức năng** | **Số lượng tài khoản/nhân viên** | **Thời gian xử lý (ms)** | **Ghi chú** |
| --- | --- | --- | --- | --- |
| 1 | Đăng nhập | 10 | 15 | Hoạt động tốt |
| 2 | Đăng nhập | 100 | 40 | Chậm hơn nhưng chấp nhận được |
| 3 | Đăng nhập | 1,000 | 250 | Cần cải tiến thuật toán tra cứu |
| 4 | Xử lý báo cáo hiệu suất | 100 | 30 | Ổn định |
| 5 | Xử lý báo cáo hiệu suất | 500 | 120 | Cần tối ưu hóa xử lý |
| 6 | Xử lý báo cáo hiệu suất | 1,000 | 300 | Giới hạn tải cao |

**Đề Xuất Cải Tiến:**

1. **Đăng nhập:**
   * Tối ưu hóa HashMap bằng cách sử dụng cấu trúc dữ liệu nhanh hơn như ConcurrentHashMap.
   * Sử dụng thuật toán băm bảo mật để tìm kiếm nhanh hơn.
2. **Xử lý báo cáo:**
   * Dùng thư viện xử lý dữ liệu lớn như Apache Spark nếu cần mở rộng.
   * Tối ưu hóa các truy vấn dữ liệu trong bộ nhớ hoặc từ cơ sở dữ liệu.