BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**HỌC PHẦN: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**

**ĐỀ TÀI: HỆ THỐNG ĐIỂM DANH CHO PHÉP GIÁO VIÊN ĐIỂM DANH SINH VIÊN TRONG LỚP**

|  |  |  |  |
| --- | --- | --- | --- |
| Mã Sinh Viên | Họ và Tên | Ngày Sinh | Lớp |
| 1771020649 | Nguyễn Trường Thịnh | 31/03/2005 | CNTT - 1704 |

### 

**Hà Nội, năm 2024**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**HỌC PHẦN: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**

**ĐỀ TÀI: HỆ THỐNG ĐIỂM DANH CHO PHÉP GIÁO VIÊN ĐIỂM DANH SINH VIÊN TRONG LỚP**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mã Sinh Viên | Họ và Tên | Ngày Sinh | Điểm | |
| Bằng Số | Bằng Chữ |
| 1771020649 | Nguyễn Trường Thịnh | 31/03/2005 |  |  |

### 

### CÁN BỘ CHẤM THI 1 CÁN BỘ CHẤM THI 2

**Hà Nội, năm 2024**

**LỜI NÓI ĐẦU**

(Nếu có)

Trong bối cảnh phát triển mạnh mẽ của công nghệ thông tin, việc hệ thống các giải pháp công nghệ vào giáo dục đang ngày càng trở nên quan trọng và cần thiết. Một trong những thách thức đối với các giáo viên và nhà quản lý giáo dục là việc điểm danh và quản lý sự có mặt của sinh viên trong các buổi học một cách hiệu quả. Quá trình điểm danh thủ công không chỉ tiêu tốn thời gian mà còn có khả năng gây ra sai sót, dẫn đến khó khăn trong việc theo dõi và đánh giá sự chuyên cần của sinh viên.

Nhằm giải quyết vấn đề này, hệ thống điểm danh tự động sử dụng hệ thống lập trình hướng đối tượng đã được phát triển. Hệ thống này cho phép giáo viên dễ dàng thực hiện điểm danh sinh viên ngay trong lớp học thông qua giao diện đơn giản và thân thiện. Hệ thống giúp giảm thiểu sai sót trong quá trình quản lý điểm danh, đồng thời cung cấp các công cụ hỗ trợ giáo viên theo dõi và xuất báo cáo về sự chuyên cần của sinh viên.

Trong quá trình phát triển hệ thống, chúng tôi đã áp dụng các nguyên lý của lập trình hướng đối tượng để xây dựng một hệ thống có cấu trúc rõ ràng, dễ bảo trì và mở rộng. Hệ thống bao gồm các thành phần như quản lý sinh viên, quản lý lớp học, và báo cáo điểm danh, tất cả được thiết kế dựa trên các đối tượng trong lập trình hướng đối tượng để đảm bảo tính mô đun và linh hoạt.

Với mong muốn góp phần nâng cao hiệu quả quản lý lớp học và cải thiện trải nghiệm giảng dạy của giáo viên, chúng tôi hy vọng rằng hệ thống điểm danh này sẽ trở thành một công cụ hữu ích và tiện lợi trong môi trường giáo dục hiện đại.

**MỤC LỤC**

[**CHƯƠNG 1. MÔ TẢ BÀI TOÁN**](#_Toc183256874) **5**

**1.1.** [**Giới thiệu**](#_Toc183256875) **5**

**1.2.** [**Chức năng chính**](#_Toc183256876) **5**

**CHƯƠNG 2. PHÂN TÍCH YÊU CẦU VÀ THIẾT KẾ HỆ THỐNG****9**

[**2.1. Phân tích yêu cầu**](#_Toc183256878) **9**

[**2.1.1. Yêu cầu chức năng**](#_Toc183256879) **9**

[**2.1.2. Yêu cầu phi chức năng**](#_Toc183256880) **10**

[**2.1.3. Yêu cầu giao diện người dùng**](#_Toc183256881) **11**

[**2.1.4. Các yếu tố kĩ thuật khác**](#_Toc183256882) **12**

[**2.1.5. Các ràng buộc**](#_Toc183256883) **12**

[**2.1.6. Các yêu cầu về bảo trì và hỗ trợ**](#_Toc183256884) **12**

[**2.2. Thiết kế hệ thống**](#_Toc183256885) **12**

[**2.3. Triển khai**](#_Toc183256886) **13**

[**2.4. Vận hành và bảo trì**](#_Toc183256887) **14**

[**CHƯƠNG 3. KẾT QUẢ THỰC HIỆN**](#_Toc183256888) **16**

[**3.1. Công nghệ đã sử dụng**](#_Toc183256889) **16**

[**3.2. Tiến độ thực hiện**](#_Toc183256890) **16**

[**3.3. Hình ảnh sản phẩm**](#_Toc183256891) **18**

[**KẾT LUẬN**](#_Toc183256892) **34**

**CHƯƠNG 1. MÔ TẢ BÀI TOÁN**

**1.1. Giới thiệu**

Xây dựng thành một hệ thống (Student Check) dành cho giáo viên và sinh viên, giúp giáo viên dễ dàng điểm danh sinh viên, giúp tiết kiệm thời gian và giảm thiểu sai sót trong quá trình điểm danh truyền thống. Thay vì thực hiện thủ công, hệ thống cung cấp các chức năng tiên tiến để điểm danh, cập nhật trạng thái sinh viên theo thời gian thực và tự động tạo báo cáo về sự có mặt của sinh viên trong các buổi học. Hệ thống sẽ được xây dựng bằng Java và chạy trên giao diện console (command-line).

**1.2. Chức năng chính**

Hệ thống Student Check cần có các chức năng sau:

### ****Chức năng đăng nhập và phân quyền****

* **Đăng nhập người dùng**: Hệ thống cho phép giảng viên và sinh viên đăng nhập bằng tài khoản cá nhân.
* **Phân quyền người dùng**: Phân quyền để đảm bảo chỉ giảng viên có thể điểm danh và quản lý danh sách lớp. Sinh viên chỉ có quyền xem kết quả điểm danh của mình.

### ****Tạo và quản lý lớp học****

* Giảng viên hoặc quản trị viên có thể tạo lớp học mới với các thông tin: mã lớp, tên lớp, khoa trực thuộc.
* Chỉnh sửa hoặc cập nhật thông tin lớp học như lịch học, tên lớp, và khoa.
* Xóa lớp học khi lớp không còn hoạt động.

1. **Quản lý khoa**

* Thêm, chỉnh sửa và xóa thông tin về các khoa trong trường.
* Liên kết các lớp học với khoa tương ứng để quản lý dễ dàng.

1. **Quản lý giảng viên**

**Thông tin giảng viên:**

* Hệ thống quản lý thông tin của từng giảng viên, bao gồm:
  + **Tên giảng viên**.
  + **Số điện thoại**.
  + **Email**.
* Cho phép quản trị viên thêm, chỉnh sửa, hoặc xóa thông tin giảng viên.
* **Liên kết giảng viên với các lớp học mà họ đang phụ trách.**

1. **Quản lý sinh viên**

**Mã sinh viên và thông tin sinh viên:**

* Mỗi sinh viên được xác định qua **mã sinh viên** duy nhất, đi kèm các thông tin cơ bản như tên, lớp học, và khoa trực thuộc.
* Hệ thống sẽ tự động cập nhật mã sinh viên vào lớp tương ứng khi tạo lớp hoặc cập nhật danh sách sinh viên.
* Giảng viên có thể xem danh sách sinh viên của từng lớp và cập nhật thông tin cá nhân của sinh viên nếu cần.

1. **Điểm danh**

**Điểm danh hàng ngày**:

* Giảng viên có thể điểm danh sinh viên theo buổi học (ngày tháng cụ thể), chọn trạng thái cho từng sinh viên: **Có mặt**, **Vắng có phép**, **Vắng không phép**.
* Hệ thống sẽ lưu trữ thông tin điểm danh của từng sinh viên, bao gồm ngày, trạng thái, và ghi chú (nếu có).

**Điểm danh thủ công hoặc tự động**:

* Cho phép giảng viên chọn cách điểm danh thủ công (chọn trạng thái) hoặc tích hợp với hệ thống tự động (quét mã QR, RFID, nhận diện khuôn mặt).

**Chỉnh sửa điểm danh**:

* Giảng viên có thể chỉnh sửa trạng thái điểm danh của sinh viên sau buổi học nếu có thay đổi như cung cấp lý do vắng mặt hợp lệ.

1. **Quản lý ngày nghỉ**

**Ngày nghỉ có phép và không phép**:

* Hệ thống phân loại ngày nghỉ của sinh viên thành **có phép** và **không phép** dựa trên lý do giảng viên xác nhận.
* Giảng viên có thể thêm lý do nghỉ phép cho sinh viên, bao gồm thông tin về giấy phép hoặc lý do hợp lệ để xác nhận.

1. **Báo cáo và thống kê**

**Thống kê sự chuyên cần của sinh viên**:

* Hệ thống cung cấp các báo cáo chi tiết về tỉ lệ chuyên cần của sinh viên, bao gồm số ngày nghỉ có phép, không phép, và số buổi tham gia.

**Tạo báo cáo điểm danh**:

* Giảng viên có thể tạo báo cáo chuyên cần của sinh viên theo thời gian cụ thể (tuần, tháng, hoặc kỳ học).
* Báo cáo có thể được xuất dưới định dạng PDF hoặc Excel để giảng viên nộp cho nhà trường.

### ****Bảo mật và sao lưu dữ liệu****

* **Bảo mật dữ liệu**: Đảm bảo thông tin sinh viên và kết quả điểm danh được bảo mật, chỉ có quyền truy cập từ những người được ủy quyền.
* **Sao lưu dữ liệu**: Dữ liệu điểm danh được sao lưu định kỳ để tránh mất dữ liệu trong trường hợp hệ thống gặp sự cố.

### ****Chức năng trực tuyến và di động****

* **Phiên bản trên web và di động**: Hệ thống có thể được triển khai dưới dạng ứng dụng web và ứng dụng di động để giảng viên và sinh viên dễ dàng truy cập từ nhiều thiết bị.
* **Điểm danh trực tuyến**: Hỗ trợ điểm danh cho các lớp học trực tuyến, với việc theo dõi sự tham gia qua video, đăng nhập hoặc thông qua các phương pháp xác thực khác.

### ****Tích hợp với hệ thống quản lý khác****

* **Tích hợp với hệ thống quản lý học tập (LMS)**: Hệ thống điểm danh có thể tích hợp với các nền tảng quản lý học tập khác như Moodle hoặc Google Classroom để đồng bộ dữ liệu về sinh viên và lớp học.

**CHƯƠNG 2. PHÂN TÍCH YÊU CẦU VÀ THIẾT KẾ HỆ THỐNG**

**2.1. Phân tích yêu cầu:**

### 2.1.1. ****Yêu cầu chức năng****

#### 1. **Chức năng quản lý lớp học**

* **Tạo lớp học mới**: Giảng viên hoặc quản trị viên có thể tạo lớp học mới, nhập các thông tin như mã lớp, tên lớp, thời gian học, và danh sách sinh viên.
* **Chỉnh sửa lớp học**: Giảng viên có thể chỉnh sửa thông tin lớp học, bao gồm lịch học, thời gian học, và danh sách sinh viên.
* **Xóa lớp học**: Giảng viên hoặc quản trị viên có thể xóa lớp học khi không còn cần thiết.

2. **Chức năng điểm danh**

* **Điểm danh thủ công**: Giảng viên có thể điểm danh bằng cách chọn tên sinh viên có mặt, vắng mặt hoặc đến muộn trong danh sách sinh viên của lớp.
* **Điểm danh tự động**: Hệ thống có thể tích hợp với các công nghệ như quét mã QR, nhận diện khuôn mặt, hoặc sử dụng thẻ RFID để tự động điểm danh khi sinh viên vào lớp.
* **Xác nhận điểm danh**: Giảng viên có thể xác nhận danh sách sinh viên có mặt/vắng mặt trước khi lưu trữ kết quả điểm danh.

#### 3. **Chức năng quản lý điểm danh**

* **Xem lịch sử điểm danh**: Giảng viên và sinh viên có thể xem lại lịch sử điểm danh cho từng buổi học.
* **Chỉnh sửa kết quả điểm danh**: Giảng viên có thể chỉnh sửa kết quả điểm danh nếu có lỗi (ví dụ, sinh viên có lý do vắng mặt hợp lệ).
* **Thống kê điểm danh**: Hệ thống cung cấp các báo cáo và thống kê về sự chuyên cần của sinh viên, như số buổi vắng mặt, đi muộn, hoặc tham gia đầy đủ.

#### 4. **Thông báo và nhắc nhở**

* **Thông báo vắng mặt**: Khi sinh viên vắng mặt nhiều lần, hệ thống có thể gửi thông báo tự động đến sinh viên và giảng viên.
* **Nhắc nhở điểm danh**: Hệ thống gửi thông báo nhắc nhở giảng viên thực hiện điểm danh vào đầu mỗi buổi học.

#### 5. **Chức năng báo cáo**

* **Tạo báo cáo chuyên cần**: Hệ thống có thể tạo báo cáo về sự chuyên cần của từng sinh viên, bao gồm số lần vắng mặt, đi trễ, và tổng số buổi tham gia.
* **Xuất báo cáo**: Giảng viên có thể xuất báo cáo điểm danh dưới định dạng PDF hoặc Excel để lưu trữ hoặc nộp cho nhà trường.

#### 6. **Quản lý quyền người dùng**

* **Phân quyền người dùng**: Hệ thống có phân quyền cho các vai trò như quản trị viên, giảng viên và sinh viên. Quản trị viên có toàn quyền quản lý hệ thống, giảng viên có quyền quản lý lớp học và điểm danh, còn sinh viên chỉ có thể xem thông tin của mình.

**2.1.2. Yêu cầu phi chức năng**

#### 1. **Hiệu suất**

* Hệ thống cần có khả năng phản hồi nhanh chóng trong vòng 2-3 giây khi thực hiện các thao tác như điểm danh, xem báo cáo hoặc truy xuất dữ liệu sinh viên.
* Hỗ trợ hàng trăm giảng viên và sinh viên truy cập đồng thời trong các giờ cao điểm mà không làm giảm hiệu suất.

#### 2. **Bảo mật**

* Xác thực người dùng bằng tên đăng nhập và mật khẩu, với cơ chế phân quyền chặt chẽ cho từng vai trò.
* Dữ liệu về sinh viên và lịch sử điểm danh cần được mã hóa khi truyền tải và lưu trữ để tránh rò rỉ thông tin.
* Hỗ trợ bảo mật xác thực hai yếu tố (2FA) cho giảng viên và quản trị viên để tăng cường tính an toàn.

#### 3. **Khả năng mở rộng**

* Hệ thống cần dễ dàng mở rộng để quản lý nhiều lớp học, nhiều giảng viên và sinh viên khi quy mô tổ chức phát triển.
* Có thể tích hợp với các hệ thống quản lý khác như LMS (Learning Management System) hoặc hệ thống quản lý thông tin sinh viên (SIS) khi cần thiết.

#### 4. **Tính sẵn sàng**

* Hệ thống cần có thời gian hoạt động (uptime) cao, đạt ít nhất 99.9% để đảm bảo không bị gián đoạn trong quá trình giảng dạy.
* Cần có cơ chế dự phòng và khôi phục nhanh chóng khi hệ thống gặp sự cố.

#### 5. **Tính di động**

* Hệ thống phải tương thích với nhiều loại thiết bị (máy tính, điện thoại di động, máy tính bảng) và hỗ trợ các hệ điều hành khác nhau (Windows, macOS, iOS, Android).
* Giao diện của hệ thống cần thiết kế theo chuẩn đáp ứng (responsive) để người dùng có thể dễ dàng thao tác trên các thiết bị di động.

### 2.1.3. ****Yêu cầu giao diện người dùng****

* **Giao diện trực quan**: Giao diện cần dễ sử dụng, thân thiện với người dùng, đặc biệt là giảng viên, giúp họ thao tác nhanh chóng khi điểm danh và xem báo cáo.
* **Thiết kế đáp ứng**: Giao diện phải tự động điều chỉnh kích thước và bố cục phù hợp với các màn hình khác nhau như máy tính để bàn, laptop, và điện thoại di động.
* **Hỗ trợ đa ngôn ngữ**: Nếu hệ thống phục vụ cho nhiều ngôn ngữ khác nhau, cần hỗ trợ tính năng đa ngôn ngữ để giảng viên và sinh viên có thể sử dụng hệ thống một cách thuận tiện.

### 2.1.4. ****Các yếu tố kỹ thuật khác****

* **Cơ sở dữ liệu**: Hệ thống cần sử dụng cơ sở dữ liệu hiệu quả để lưu trữ thông tin về sinh viên, lớp học, lịch học, và lịch sử điểm danh. Cơ sở dữ liệu cần được tối ưu hóa để truy xuất nhanh chóng.
* **Công nghệ phát triển**: Hệ thống có thể được phát triển bằng các công nghệ hiện đại như Java, PHP, Python hoặc các framework phát triển web phổ biến (React, Angular, Vue.js) cho phần frontend. Cần đảm bảo rằng hệ thống có thể mở rộng và bảo trì dễ dàng.

### 2.1.5. ****Các ràng buộc****

* **Ngân sách**: Hệ thống cần phát triển trong phạm vi ngân sách cho phép, đồng thời đảm bảo đầy đủ các tính năng cần thiết.
* **Thời gian phát triển**: Dự án cần hoàn thành trong khung thời gian cụ thể (ví dụ: 3-6 tháng) để đáp ứng yêu cầu của giảng viên và nhà trường.
* **Tích hợp với hệ thống hiện có**: Nếu trường đã có hệ thống quản lý sinh viên, cần tích hợp hệ thống điểm danh với hệ thống đó để đồng bộ thông tin.

### 2.1.6. ****Các yêu cầu về bảo trì và hỗ trợ****

* **Cập nhật và nâng cấp hệ thống**: Hệ thống cần có khả năng nâng cấp và cập nhật khi cần để bổ sung các tính năng mới hoặc sửa lỗi.
* **Hỗ trợ người dùng**: Cung cấp tài liệu hướng dẫn sử dụng cho giảng viên và sinh viên, cùng với dịch vụ hỗ trợ kỹ thuật nếu người dùng gặp sự cố trong quá trình sử dụng.

**2.2. Thiết kế hệ thống**

### 1. ****Xác định mục tiêu và yêu cầu:****

* **Mục tiêu chính:** Tự động hóa quá trình điểm danh, nâng cao hiệu quả quản lý lớp học, cung cấp dữ liệu chính xác về sự có mặt của sinh viên.
* **Yêu cầu:**
  + **Tính chính xác:** Đảm bảo thông tin điểm danh chính xác, không xảy ra sai sót.
  + **Hiệu quả:** Giảm thiểu thời gian điểm danh, tăng năng suất làm việc của giáo viên.
  + **Dễ sử dụng:** Giao diện thân thiện, dễ sử dụng cho cả giáo viên và sinh viên.
  + **Bảo mật:** Bảo vệ thông tin cá nhân của sinh viên.
  + **Linh hoạt:** Có thể tùy chỉnh để phù hợp với nhiều loại hình lớp học và trường học khác nhau.

### 2. ****Các tính năng chính:****

* **Điểm danh:**
  + **Mã sinh viên: Đọc mã sinh viên để được điểm danh**
  + **Họ tên:** Họ tên điểm danh cho sinh viên
* **Quản lý thông tin:**
  + Quản lý danh sách lớp học, sinh viên.
  + Lưu trữ lịch sử điểm danh.
  + Tạo báo cáo thống kê về sự vắng mặt của sinh viên.
* **Thông báo:**
  + Gửi thông báo đến phụ huynh khi sinh viên vắng mặt.
  + Gửi thông báo nhắc nhở sinh viên về việc điểm danh.
* **Tích hợp:**
  + Tích hợp với hệ thống quản lý học sinh của trường.
  + Tích hợp với các ứng dụng khác như lịch học, bảng tin.

**2.3. Triển khai**

### ****1. Giới thiệu****

Việc triển khai hệ thống điểm danh sinh viên bằng Java command line cho phép tạo ra một ứng dụng đơn giản, dễ sử dụng và không yêu cầu giao diện đồ họa phức tạp. Điều này đặc biệt hữu ích cho các trường hợp muốn có một giải pháp nhanh chóng và linh hoạt.

### ****2. Các bước thực hiện****

* **Lớp SinhVien:**
  + Mã sinh viên (String)
  + Tên sinh viên (String)
  + Lớp (String)
* **Lớp LopHoc:**
  + Mã lớp (String)
  + Tên lớp (String)
  + Danh sách sinh viên (List<SinhVien>)
* **Lớp DiemDanh:**
  + Mã lớp (String)
  + Ngày điểm danh (Date)
  + Danh sách sinh viên có mặt (List<String>)

**2.4. Vận hành và bảo trì**

**Giám sát hoạt động:**

* **Theo dõi thường xuyên:** Kiểm tra hệ thống hoạt động có ổn định không, có lỗi gì xảy ra không.
* **Kiểm tra kết quả điểm danh:** So sánh kết quả điểm danh trên hệ thống với sổ điểm danh truyền thống để đảm bảo tính chính xác.
* **Theo dõi hiệu năng:** Đánh giá thời gian phản hồi của hệ thống, khả năng chịu tải khi nhiều người sử dụng cùng lúc.

**Bảo dưỡng:**

* **Cập nhật phần mềm:** Cập nhật phiên bản mới của phần mềm để khắc phục lỗi, tăng tính năng.
* **Sao lưu dữ liệu:** Sao lưu dữ liệu thường xuyên để phòng ngừa mất mát dữ liệu trong trường hợp xảy ra sự cố.
* **Vệ sinh thiết bị:** Vệ sinh các thiết bị phần cứng như máy chủ, máy tính bảng, máy quét mã QR.

**Sửa chữa:**

* **Xử lý sự cố:** Khi gặp sự cố, nhanh chóng xác định nguyên nhân và tìm giải pháp khắc phục.
* **Liên hệ nhà cung cấp:** Nếu không thể tự sửa chữa, liên hệ với nhà cung cấp để được hỗ trợ.

**Nâng cấp:**

* **Mở rộng tính năng:** Thêm các tính năng mới như báo cáo thống kê chi tiết, tích hợp với các hệ thống khác.
* **Cải thiện hiệu năng:** Nâng cấp phần cứng, tối ưu hóa code để tăng tốc độ xử lý.

**CHƯƠNG 3. KẾT QUẢ THỰC HIỆN**

**3.1. Công nghệ đã sử dụng**

Ngôn ngữ Java: Là ngôn ngữ lập trình chính được sử dụng để xây dựng ứng dụng. Java được biết đến với tính hướng đối tượng, khả năng chạy trên nhiều nền tảng và cộng đồng người dùng lớn.

OOP (Object-Oriented Programming): Toàn bộ ứng dụng được xây dựng dựa trên các lớp và đối tượng. Các lớp như, quanLySinhVien, quanLyLopHoc, quanLyKhoa, quanLyDiemDanh đại diện cho các thực thể trong thế giới thực.

Collections: Sử dụng ArrayList để lưu trữ danh sách các đối tượng như sinh viên, lớp học, giảng viên. Đây là một trong những cách phổ biến để quản lý tập hợp các đối tượng trong Java.

File I/O: Sử dụng các lớp File, FileWriter, Scanner để đọc và ghi dữ liệu vào các tệp văn bản. Điều này giúp lưu trữ thông tin một cách bền vững.

Exception Handling: Sử dụng *try-catch* để xử lý các ngoại lệ có thể xảy ra, như lỗi khi mở file, lỗi khi ghi file.

**3.2. Tiến độ thực hiện**

[**https://github.com/thinh3132005/NhomHNguyenTruongThinh**](https://github.com/thinh3132005/NhomHNguyenTruongThinh)

**B1. Tạo dự án mới:**

* Mở IntelliJ IDEA.
* Chọn "Create New Project".
* Chọn "Java" làm ngôn ngữ lập trình.
* Chọn JDK phù hợp (ví dụ: JDK 11 hoặc mới hơn).
* Nhập tên dự án (ví dụ: "TodoApp").
* Chọn vị trí lưu trữ dự án.
* Nhấn "Finish" để tạo dự án.

**B2. Tạo các package:**

* Trong cửa sổ "Project", click chuột phải vào thư mục "src".
* Chọn "New" -> "Package".
* Tạo các package sau:
  + model: chứa các lớp Task và TaskList.
  + view: chứa lớp TaskView.
  + controller: chứa lớp TaskController.
  + handler: chứa lớp GoogleSheetHandler.

**B3. Tạo các lớp:**

* Trong mỗi package, click chuột phải và chọn "New" -> "Java Class" để tạo các lớp tương ứng.
* Cài đặt các thuộc tính và phương thức cho từng lớp dựa trên thiết kế đã phân tích.

**B4. Cài đặt thư viện Google Sheets API:**

* Mở file pom.xml (nếu bạn đang sử dụng Maven) hoặc build.gradle (nếu bạn đang sử dụng Gradle).
* Thêm dependency cho thư viện Google API Client Library for Java. Ví dụ, trong Maven:
* IntelliJ IDEA sẽ tự động tải về và thêm thư viện vào dự án.

**B5. Viết code:**

* Bắt đầu viết code cho từng lớp, thực hiện các chức năng của ứng dụng:
  + **Task:** cài đặt các thuộc tính và phương thức cơ bản.
  + **TaskList:** cài đặt các phương thức để quản lý danh sách công việc.
  + **GoogleSheetHandler:** cài đặt các phương thức để kết nối, đọc, ghi, cập nhật, xóa dữ liệu trên Google Sheet.
  + **TaskView:** cài đặt các phương thức để hiển thị giao diện console và tương tác với người dùng.
  + **TaskController:** cài đặt logic xử lý yêu cầu của người dùng, điều phối các lớp khác để hoàn thành các chức năng.

**B6. Xử lý xác thực Google Sheets:**

* Tạo tài khoản dịch vụ trên Google Cloud Platform và tải xuống file JSON chứa khóa API.
* Tham khảo tài liệu của Google Sheets API để biết cách sử dụng file JSON này để xác thực ứng dụng của bạn.
* Cài đặt code xử lý xác thực trong GoogleSheetHandler.

**B7. Chạy và kiểm thử:**

* Chạy ứng dụng từ IntelliJ IDEA bằng cách click chuột phải vào lớp Main (hoặc lớp chứa phương thức main) và chọn **"Run 'Main.main()'"**.
* Kiểm tra các chức năng của ứng dụng, sửa lỗi và hoàn thiện code.

**B8. Triển khai (tùy chọn):**

* Nếu muốn đóng gói ứng dụng thành file JAR để dễ dàng chia sẻ và chạy trên các máy khác, bạn có thể sử dụng chức năng "Build Artifacts" của IntelliJ IDEA.

**3.3. Hình ảnh sản phẩm**

import java.io.File;  
import java.io.FileNotFoundException;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;  
import java.io.FileWriter;  
import java.io.IOException;  
public class Main {  
 private static int *i*;  
 private static quanLyKhoa[] *danhSachKhoa*;  
 private static List<quanLyKhoa> *danhSachKhoa1*;

**import java.io.File;:** Import lớp File từ gói java.io. Lớp này được sử dụng để đại diện cho một file trên hệ thống.

**import java.io.FileNotFoundException;:** Import ngoại lệ FileNotFoundException. Ngoại lệ này sẽ được ném ra khi không tìm thấy file.

**import java.util.ArrayList;:** Import lớp ArrayList từ gói java.util. Đây là một lớp dùng để tạo danh sách các đối tượng, cho phép thêm, xóa, sửa các phần tử trong danh sách.

**import java.util.List;:** Import interface List từ gói java.util. Interface này định nghĩa các phương thức cơ bản để làm việc với danh sách các đối tượng.

**import java.util.Scanner;:** Import lớp Scanner từ gói java.util. Lớp này được sử dụng để đọc dữ liệu từ bàn phím hoặc từ một file.

**import java.io.FileWriter;:** Import lớp FileWriter từ gói java.io. Lớp này được sử dụng để ghi dữ liệu vào một file.

**import java.io.IOException;:** Import ngoại lệ IOException. Ngoại lệ này sẽ được ném ra khi có lỗi xảy ra trong quá trình làm việc với file.

**import Explain.Test.Document;:** Import lớp Document từ gói Explain.Test. Lớp này có thể là một lớp tùy chỉnh được định nghĩa trong dự án của bạn, và nó chứa các thông tin hoặc chức năng liên quan đến tài liệu.

**import Explain.Test.Fix;:** Tương tự như trên, import lớp Fix từ gói Explain.Test. Lớp này có thể chứa các chức năng để sửa chữa hoặc cải thiện dữ liệu.

public static void main(String[] args) {  
 List<dangNhap> danhSachNguoiDung = new ArrayList<>();  
 danhSachNguoiDung.add(new dangNhap("admin", "1"));  
 danhSachNguoiDung.add(new dangNhap("thinh", "1"));  
 danhSachNguoiDung.add(new dangNhap("giang", "1"));  
 danhSachNguoiDung.add(new dangNhap("huy", "1"));  
  
 Scanner scanner = new Scanner(System.*in*);  
 boolean dangNhapThanhCong = false;  
  
 while (!dangNhapThanhCong) {  
 System.*out*.print("Nhập tài khoản: ");  
 String taiKhoanNhap = scanner.nextLine();  
  
 System.*out*.print("Nhập mật khẩu: ");  
 String matKhauNhap = scanner.nextLine();  
  
 for (dangNhap nguoiDung : danhSachNguoiDung) {  
 if (nguoiDung.kiemTradangNhap(taiKhoanNhap, matKhauNhap)) {  
 dangNhapThanhCong = true;  
 System.*out*.println("Đăng nhập thành công!");  
 *hienThiMenu*(scanner, nguoiDung.getTaiKhoan());  
 break;  
 }  
 }  
  
 if (!dangNhapThanhCong) {  
 System.*out*.println("Sai tài khoản hoặc mật khẩu! Vui lòng thử lại.");  
 }

****Tạo danh sách người dùng:****

List<dangNhap> danhSachNguoiDung = new ArrayList<>();: Tạo một danh sách (List) để lưu trữ thông tin đăng nhập của các người dùng. Mỗi phần tử trong danh sách là một đối tượng của lớp dangNhap.

**Nhập thông tin đăng nhập từ người dùng:**

Scanner scanner = new Scanner(System.in);: Tạo một đối tượng Scanner để đọc dữ liệu nhập từ bàn phím.

while (!dangNhapThanhCong) { ... }: Vòng lặp này sẽ tiếp tục cho đến khi người dùng đăng nhập thành công.

Bên trong vòng lặp, chương trình sẽ yêu cầu người dùng nhập tài khoản và mật khẩu.

****Kiểm tra thông tin đăng nhập:****

for (dangNhap nguoiDung : danhSachNguoiDung) { ... }: Vòng lặp này duyệt qua từng phần tử trong danh sách người dùng.

if (nguoiDung.kiemTradangNhap(taiKhoanNhap, matKhauNhap)) { ... }: Kiểm tra xem thông tin đăng nhập mà người dùng nhập vào có khớp với thông tin của bất kỳ người dùng nào trong danh sách hay không. Phương thức kiemTradangNhap của lớp dangNhap sẽ thực hiện việc so sánh này.

Nếu tìm thấy một người dùng có thông tin trùng khớp, biến dangNhapThanhCong sẽ được gán giá trị true và vòng lặp sẽ dừng lại.

**Hiển thị thông báo và gọi menu:**

Nếu đăng nhập thành công, chương trình sẽ in ra thông báo "Đăng nhập thành công!" và gọi hàm hienThiMenu để hiển thị menu cho người dùng.

Giải thích thêm về lớp dangNhap:

Lớp dangNhap (mặc dù không được định nghĩa trong đoạn code này) sẽ chứa các thuộc tính như taiKhoan và matKhau.

Phương thức kiemTradangNhap của lớp dangNhap sẽ so sánh tài khoản và mật khẩu được truyền vào với các thuộc tính của đối tượng.

public static void quanLySinhVienMenu(Scanner scanner, List<quanLySinhVien> danhSachSinhVien) {  
 System.*out*.print("Nhập số lượng sinh viên cần thêm: ");  
 int soLuong = scanner.nextInt();  
 scanner.nextLine();  
  
 for (int i = 0; i < soLuong; i++) {  
 System.*out*.println("\nNhập thông tin sinh viên thứ " + (i + 1) + ":");  
  
 System.*out*.print("Họ tên: ");  
 String hoTen = scanner.nextLine();  
  
 System.*out*.print("Lớp: ");  
 String lop = scanner.nextLine();  
  
 System.*out*.print("Khoa: ");  
 String khoa = scanner.nextLine();  
  
 System.*out*.print("Số điện thoại: ");  
 String soDienThoai = scanner.nextLine();  
  
 System.*out*.print("Gmail: ");  
 String gmail = scanner.nextLine();  
  
 System.*out*.print("Mã sinh viên: ");  
 String maSinhVien = scanner.nextLine();  
  
 danhSachSinhVien.add(new quanLySinhVien(hoTen, lop, khoa, soDienThoai, gmail, maSinhVien));  
 }  
  
 *ghiDanhSachSinhVienVaoFile*(danhSachSinhVien, "E:\\BTL HDT\\data\\danhsachsinhvien.txt");  
 System.*out*.println("Cập nhật danh sách sinh viên thành công!");  
}

**Khai báo hàm:**

**public static void quanLySinhVienMenu(Scanner scanner, List<quanLySinhVien> danhSachSinhVien):**

**public:** Hàm có thể được truy cập từ bất kỳ lớp nào.

**static:** Hàm thuộc về lớp chứ không thuộc về đối tượng của lớp.

**void:** Hàm không trả về giá trị.

**quanLySinhVienMenu:** Tên của hàm, gợi ý rằng hàm này thực hiện chức năng quản lý menu cho sinh viên.

**Scanner scanner:** Đối tượng Scanner để nhận dữ liệu nhập từ người dùng.

**List<quanLySinhVien> danhSachSinhVien:** Một danh sách các đối tượng sinh viên, nơi mà thông tin sinh viên mới sẽ được thêm vào.

**Nhập số lượng sinh viên:**

**System.out.print("Nhập số lượng sinh viên cần thêm: ");:** In ra màn hình một thông báo yêu cầu người dùng nhập số lượng sinh viên cần thêm.

**int soLuong = scanner.nextInt();:** Đọc giá trị số nguyên từ bàn phím và gán vào biến soLuong.

**Vòng lặp nhập thông tin sinh viên:**

**for (int i = 0; i < soLuong; i++) { ... }:** Vòng lặp này sẽ lặp lại soLuong lần, mỗi lần nhập thông tin của một sinh viên.

Bên trong vòng lặp, chương trình sẽ in ra các thông báo để yêu cầu người dùng nhập từng thông tin của sinh viên như: họ tên, lớp, khoa, số điện thoại, gmail.

Dữ liệu nhập vào từ bàn phím sẽ được gán vào các biến tương ứng.

public static void ghiDanhSachSinhVienVaoFile(List<quanLySinhVien> danhSach, String filePath) {  
 try (FileWriter writer = new FileWriter(filePath)) {  
 for (quanLySinhVien sinhVien : danhSach) {  
 writer.write(sinhVien.toString() + "\n");  
 }  
 } catch (IOException e) {  
 System.*out*.println("Đã xảy ra lỗi khi ghi file: " + e.getMessage());  
 }  
}

**Chức năng:** Đoạn code này thực hiện việc ghi thông tin của các sinh viên trong một danh sách vào một file văn bản.

**Phân tích từng phần:**

**Khai báo hàm:**

**public static void ghiDanhSachSinhVienVaoFile(List<quanLySinhVien> danhSach, String filePath):**

**public:** Hàm có thể được truy cập từ bất kỳ đâu trong chương trình.

**static:** Hàm thuộc về lớp chứ không thuộc về đối tượng của lớp.

**void**: Hàm không trả về giá trị.

**ghiDanhSachSinhVienVaoFile**: Tên hàm gợi ý chức năng của hàm là ghi danh sách sinh viên vào file.

**List<quanLySinhVien> danhSach**: Tham số đầu vào là một danh sách các đối tượng sinh viên.

**String filePath**: Tham số thứ hai là một chuỗi đại diện cho đường dẫn đến file cần ghi.

**Thực hiện ghi file:**

**try (FileWriter writer = new FileWriter(filePath)) { ... }:**

**try-with-resources:** Cấu trúc này đảm bảo rằng file sẽ được đóng tự động khi kết thúc khối lệnh try hoặc khi có ngoại lệ xảy ra.

**FileWriter writer = new FileWriter(filePath):** Tạo một đối tượng FileWriter để ghi dữ liệu vào file tại đường dẫn filePath.

**Vòng lặp:**

**for (quanLySinhVien sinhVien : danhSach) { ... }:** Vòng lặp này duyệt qua từng phần tử (sinh viên) trong danh sách danhSach.

**writer.write(sinhVien.toString() + "\n");:**

**sinhVien.toString():** Gọi phương thức toString() của đối tượng sinh viên để chuyển đổi thông tin của sinh viên thành một chuỗi ký tự.

**writer.write(...):** Ghi chuỗi vừa tạo vào file. Thêm ký tự \n để xuống dòng sau khi ghi xong thông tin của một sinh viên.

**Xử lý ngoại lệ:**

**catch (IOException e) { ... }:** Nếu xảy ra lỗi khi ghi file (ví dụ: file không tồn tại, không đủ quyền truy cập), ngoại lệ IOException sẽ được bắt và xử lý.

**System.out.println("Đã xảy ra lỗi khi ghi file: " + e.getMessage());:** In ra thông báo lỗi để người dùng biết.

public static void quanLyLopHocMenu(Scanner scanner, List<quanLyLopHoc> danhSachLopHoc) {  
 int luaChonLopHoc;  
  
 do {  
 System.*out*.println("1. Thêm lớp học");  
 System.*out*.println("2. Sửa lớp học");  
 System.*out*.println("3. Xóa lớp học");  
 System.*out*.println("4. Quay lại");  
 System.*out*.print("Nhập lựa chọn của bạn: ");  
 luaChonLopHoc = scanner.nextInt();  
 scanner.nextLine();  
  
 switch (luaChonLopHoc) {  
 case 1:  
 *themLopHoc*(scanner, danhSachLopHoc);  
 break;  
 case 2:  
 *chinhSuaLopHoc*(scanner, danhSachLopHoc);  
 break;  
 case 3:  
 *xoaLopHoc*(scanner, danhSachLopHoc);  
 break;  
 case 4:  
 return;  
 default:  
 System.*out*.println("Lựa chọn không hợp lệ. Vui lòng thử lại.");  
 }  
 } while (luaChonLopHoc != 4);  
}

**Khai báo hàm:**

public static void **ghiDanhSachSinhVienVaoFile(List<quanLySinhVien> danhSach, String filePath):**

**public**: Hàm này có thể được truy cập từ bất kỳ đâu trong chương trình.

**static**: Hàm thuộc về lớp chứ không thuộc về đối tượng của lớp.

**void**: Hàm không trả về giá trị.

**ghiDanhSachSinhVienVaoFile**: Tên hàm mô tả rõ ràng chức năng của nó.

**List<quanLySinhVien> danhSach**: Tham số đầu vào là một danh sách các đối tượng sinh viên. Mỗi đối tượng trong danh sách này đại diện cho một sinh viên.

**String filePath**: Tham số thứ hai là một chuỗi đại diện cho đường dẫn đầy đủ của file mà chúng ta muốn ghi dữ liệu vào.

**Mở file để ghi:**

**try (FileWriter writer = new FileWriter(filePath)) { ... }:**

**try-with-resources**: Cấu trúc này đảm bảo rằng file sẽ được đóng tự động khi kết thúc khối lệnh try hoặc khi có ngoại lệ xảy ra.

**FileWriter writer = new FileWriter(filePath**): Tạo một đối tượng FileWriter để ghi dữ liệu vào file. Đối tượng này sẽ được liên kết với file có đường dẫn được chỉ định bởi filePath.

Duyệt và ghi thông tin sinh viên:

**for (quanLySinhVien sinhVien : danhSach) { ... }**: Vòng lặp này sẽ duyệt qua từng phần tử (sinh viên) trong danh sách danhSach.

**writer.write(sinhVien.toString() + "\n");:**

**sinhVien.toString():** Gọi phương thức **toString()** của đối tượng sinh viên để chuyển đổi thông tin của sinh viên thành một chuỗi ký tự. Phương thức này thường được định nghĩa trong lớp **quanLySinhVien** để trả về một chuỗi chứa các thông tin cần thiết về sinh viên (ví dụ: mã sinh viên, họ tên, ngày sinh, ...).

**writer.write(...):** Ghi chuỗi vừa tạo vào file. Ký tự \n được thêm vào cuối để xuống dòng, tạo ra một dòng mới cho mỗi sinh viên.

**Xử lý ngoại lệ:**

**catch (IOException e) { ... }:** Nếu có bất kỳ lỗi nào xảy ra trong quá trình ghi file (ví dụ: file không tồn tại, không đủ quyền truy cập, lỗi I/O), ngoại lệ IOException sẽ được bắt và xử lý.

**System.out.println("Đã xảy ra lỗi khi ghi file: " + e.getMessage());:** In ra thông báo lỗi để người dùng biết. Thông báo này sẽ bao gồm một mô tả chi tiết về lỗi, giúp cho việc tìm và sửa lỗi trở nên dễ dàng hơn.

public static void themLopHoc(Scanner scanner, List<quanLyLopHoc> danhSachLopHoc) {  
 System.*out*.print("Nhập số lượng lớp học cần thêm: ");  
 int soLuong = scanner.nextInt();  
 scanner.nextLine();   
  
 for (int i = 0; i < soLuong; i++) {  
 System.*out*.println("\nNhập thông tin lớp học thứ " + (i + 1) + ":");  
  
 System.*out*.print("Lịch học: ");  
 String lichHoc = scanner.nextLine();  
  
 System.*out*.print("Khoa: ");  
 String khoa = scanner.nextLine();  
  
 System.*out*.print("Mã lớp: ");  
 String maLop = scanner.nextLine();  
  
 System.*out*.print("Tên lớp: ");  
 String tenLop = scanner.nextLine();  
  
   
 quanLyLopHoc lopHoc = new quanLyLopHoc(lichHoc, khoa, maLop, tenLop);  
 danhSachLopHoc.add(lopHoc);

**Chức năng:**

Đoạn code này thực hiện việc thêm một số lượng lớp học nhất định vào một danh sách các lớp học đã tồn tại. Thông tin về mỗi lớp học (lịch học, khoa, mã lớp, tên lớp) sẽ được nhập từ bàn phím và sau đó được lưu trữ vào danh sách.

**Phân tích từng phần:**

**Khai báo hàm:**

**public static void themLopHoc(Scanner scanner, List<quanLyLopHoc> danhSachLopHoc):**

**public:** Hàm này có thể được truy cập từ bất kỳ đâu trong chương trình.

**static:** Hàm thuộc về lớp chứ không thuộc về đối tượng của lớp.

**void:** Hàm không trả về giá trị.

**themLopHoc:** Tên hàm mô tả rõ ràng chức năng của nó là thêm lớp học.

**Scanner scanner:** Tham số đầu vào là một đối tượng Scanner để đọc dữ liệu nhập từ bàn phím.

**List<quanLyLopHoc> danhSachLopHoc:** Tham số thứ hai là một danh sách các đối tượng lớp học, nơi mà các lớp học mới sẽ được thêm vào.

**Nhập số lượng lớp học:**

**System.out.print("Nhập số lượng lớp học cần thêm: ");:** In ra màn hình một thông báo yêu cầu người dùng nhập số lượng lớp học cần thêm.

**int soLuong = scanner.nextInt();:** Đọc giá trị số nguyên từ bàn phím và gán vào biến soLuong.

**scanner.nextLine()**;: Dòng này dùng để đọc bỏ ký tự xuống dòng thừa sau khi nhập số lượng, tránh gây ra lỗi trong các lần nhập tiếp theo.

**Vòng lặp nhập thông tin lớp học:**

**for (int i = 0; i < soLuong; i++) { ... }**: Vòng lặp này sẽ lặp lại soLuong lần, mỗi lần nhập thông tin của một lớp học.

Bên trong vòng lặp, chương trình sẽ in ra các thông báo để yêu cầu người dùng nhập từng thông tin của lớp học như: lịch học, khoa, mã lớp, tên lớp.

Dữ liệu nhập vào từ bàn phím sẽ được gán vào các biến tương ứng.

**Tạo đối tượng lớp học và thêm vào danh sách:**

**quanLyLopHoc lopHoc = new quanLyLopHoc(lichHoc, khoa, maLop, tenLop);:** Tạo một đối tượng **quanLyLopHoc** mới với các thông tin vừa nhập được.

**danhSachLopHoc.add(lopHoc);:** Thêm đối tượng lớp học mới vào danh sách các lớp học.

System.*out*.print("Nhập số lượng lịch học: ");  
 int soLuonglichHoc = scanner.nextInt();  
 scanner.nextLine();   
  
 for (int j = 0; j < soLuonglichHoc; j++) {  
 System.*out*.println("\nNhập thông tin lịch học thứ " + (j + 1) + ":");  
  
 System.*out*.print("Ngày: ");  
 String ngay = scanner.nextLine();  
  
 System.*out*.print("Buổi học: ");  
 String buoiHoc = scanner.nextLine();  
  
 lopHoc.themLichHoc(new lichHoc(ngay, buoiHoc));  
 }  
 }  
  
  
 *ghiDanhSachLopHocVaoFile*(danhSachLopHoc, "E:\\BTL HDT\\data\\quanlylophoc.txt");  
 System.*out*.println("Cập nhật danh sách lớp học thành công!");  
}  
  
public static void xoaLopHoc(Scanner scanner, List<quanLyLopHoc> danhSachLopHoc) {  
 System.*out*.print("Nhập mã lớp học cần xóa: ");  
 String maLopHoc = scanner.nextLine();  
  
 for (int i = 0; i < danhSachLopHoc.size(); i++) {  
 if (danhSachLopHoc.get(i).getMaLop().equals(maLopHoc)) {  
 danhSachLopHoc.remove(i);  
 System.*out*.println("Xóa lớp học thành công!");  
 *ghiDanhSachLopHocVaoFile*(danhSachLopHoc, "E:\\BTL HDT\\data\\quanlylophoc.txt");  
 return;  
 }  
 }  
  
 System.*out*.println("Mã lớp học không tồn tại!");  
}

**Thêm lịch học cho lớp học**

**Chức năng:** Phần này cho phép người dùng nhập thông tin về lịch học cho một lớp học và thêm vào đối tượng lớp học đó.

**Nhập số lượng lịch học:** Người dùng nhập vào số lượng lịch học cần thêm cho lớp học.

**Vòng lặp nhập thông tin:**

Vòng lặp **for** sẽ lặp lại theo số lượng lịch học đã nhập.

Trong mỗi vòng lặp, người dùng sẽ nhập thông tin về ngày và buổi học.

**Tạo đối tượng lịch học:**

Mỗi lần nhập thông tin xong, một đối tượng **lichHoc** mới được tạo với ngày và buổi học vừa nhập.

**Thêm lịch học vào lớp học:**

Đối tượng **lichHoc** mới được thêm vào danh sách các lịch học của lớp học thông qua phương thức **themLichHoc().**

**Ghi danh sách lớp học vào file**

**Chức năng:** Sau khi cập nhật thông tin lớp học, đoạn code này sẽ ghi lại toàn bộ danh sách lớp học vào một file văn bản để lưu trữ.

**Gọi hàm ghi file:** Hàm **ghiDanhSachLopHocVaoFile()** được gọi để thực hiện việc ghi file.

**Đường dẫn file:** Đường dẫn đến file được chỉ định rõ ràng.

**Xóa lớp học:**

**Chức năng:** Phần này cho phép người dùng xóa một lớp học khỏi danh sách dựa trên mã lớp.

**Nhập mã lớp học:** Người dùng nhập vào mã lớp học cần xóa.

**Tìm lớp học:** Vòng lặp for duyệt qua từng lớp học trong danh sách để tìm lớp học có mã trùng khớp với mã lớp vừa nhập.

**Xóa lớp học:** Nếu tìm thấy lớp học, nó sẽ được xóa khỏi danh sách.

**Ghi lại danh sách:** Sau khi xóa, danh sách lớp học được cập nhật sẽ được ghi lại vào file.

public static void chinhSuaLopHoc(Scanner scanner, List<quanLyLopHoc> danhSachLopHoc) {  
 System.*out*.print("Nhập mã lớp học cần chỉnh sửa: ");  
 String maLopHoc = scanner.nextLine();  
  
 for (quanLyLopHoc lopHoc : danhSachLopHoc) {  
 if (lopHoc.getMaLop().equals(maLopHoc)) {  
 System.*out*.println("Lớp học tìm thấy: " + lopHoc);  
 System.*out*.println("Chọn thông tin muốn sửa:");  
 System.*out*.println("1. Lịch học");  
 System.*out*.println("2. Khoa");  
 System.*out*.println("3. Tên lớp");  
 System.*out*.print("Nhập lựa chọn của bạn: ");  
 int luaChon = scanner.nextInt();  
 scanner.nextLine();   
  
 switch (luaChon) {  
 case 1:  
 System.*out*.print("Nhập lịch học mới: ");  
 lopHoc.setLichHoc(scanner.nextLine());  
 break;  
 case 2:  
 System.*out*.print("Nhập khoa mới: ");  
 lopHoc.setKhoa(scanner.nextLine());  
 break;  
 case 3:  
 System.*out*.print("Nhập tên lớp mới: ");  
 lopHoc.setTenLop(scanner.nextLine());  
 break;  
 default:  
 System.*out*.println("Lựa chọn không hợp lệ.");  
 return;  
 }  
  
 *ghiDanhSachLopHocVaoFile*(danhSachLopHoc, "E:\\BTL HDT\\data\\quanlylophoc.txt");  
 System.*out*.println("Cập nhật thông tin lớp học thành công!");  
 return;  
 }  
 }  
  
 System.*out*.println("Mã lớp học không tồn tại!");  
}

**System.out.print("Nhập mã lớp học cần chỉnh sửa:** ");: In ra màn hình yêu cầu người dùng nhập mã lớp học.

**String maLopHoc = scanner.nextLine();:** Đọc mã lớp học từ bàn phím và gán vào biến **maLopHoc.**

**for (quanLyLopHoc lopHoc : danhSachLopHoc) { ... }:** Vòng lặp duyệt qua từng lớp học trong danh sách.

**if (lopHoc.getMaLop().equals(maLopHoc)) { ... }:** Kiểm tra xem mã lớp của lớp học hiện tại có trùng với mã lớp cần tìm không.

**switch (luaChon) { ... }:** Cấu trúc điều khiển để thực hiện các hành động khác nhau dựa trên lựa chọn của người dùng.

**lopHoc.setLichHoc(scanner.nextLine());,lopHoc.setKhoa(scanner.nextLine());,** **lopHoc.setTenLop(scanner.nextLine())**;: Gọi các phương thức để cập nhật thông tin của lớp học.

**ghiDanhSachLopHocVaoFile(danhSachLopHoc,"E:\\BTL HDT\\data\\quanlylophoc.txt");:** Gọi hàm để ghi danh sách lớp học vào file.

public static void ghiDanhSachLopHocVaoFile(List<quanLyLopHoc> danhSach, String filePath) {  
 try (FileWriter writer = new FileWriter(filePath)) {  
 for (quanLyLopHoc lopHoc : danhSach) {  
 writer.write(lopHoc.toString() + "\n");  
  
 for (lichHoc lich : lopHoc.getLichHoc()) {  
 writer.write("\t" + lich.toString() + "\n");  
 }  
 }  
 } catch (IOException e) {  
 System.*out*.println("Đã xảy ra lỗi khi ghi file: " + e.getMessage());  
 }  
}

**Khai báo hàm:**

**public static void ghiDanhSachLopHocVaoFile(List<quanLyLopHoc> danhSach, String filePath):**

**public:** Hàm có thể được truy cập từ bất kỳ đâu trong chương trình.

**static:** Hàm thuộc về lớp chứ không thuộc về đối tượng của lớp.

**void:** Hàm không trả về giá trị.

**ghiDanhSachLopHocVaoFile**: Tên hàm mô tả rõ ràng chức năng của nó.

**List<quanLyLopHoc> danhSach**: Tham số đầu vào là một danh sách các đối tượng lớp học.

**String filePath**: Tham số thứ hai là một chuỗi đại diện cho đường dẫn đến file mà chúng ta muốn ghi dữ liệu vào.

**Mở file để ghi:**

**try (FileWriter writer = new FileWriter(filePath)) { ... }:**

**try-with-resources:** Cấu trúc này đảm bảo rằng file sẽ được đóng tự động khi kết thúc khối lệnh try hoặc khi có ngoại lệ xảy ra.

**FileWriter writer = new FileWriter(filePath):** Tạo một đối tượng **FileWriter** để ghi dữ liệu vào file tại đường dẫn **filePath.**

**Duyệt danh sách lớp học:**

**for (quanLyLopHoc lopHoc : danhSach) { ... }**: Vòng lặp duyệt qua từng lớp học trong danh sách.

**Ghi thông tin lớp học:**

**writer.write(lopHoc.toString() + "\n");:**

**lopHoc.toString():** Gọi phương thức **toString()** của đối tượng lớp học để chuyển đổi thông tin của lớp học thành một chuỗi ký tự.

**writer.write(...):** Ghi chuỗi vừa tạo vào file. Thêm ký tự \n để xuống dòng.

**Duyệt danh sách lịch học:**

**for (lichHoc lich : lopHoc.getLichHoc()) { ... }:** Vòng lặp lặp qua từng lịch học trong danh sách lịch học của lớp học hiện tại.

**writer.write("\t" + lich.toString() + "\n");**: Ghi thông tin của lịch học vào file, thêm một **tab (\t)** để tạo thụt đầu dòng.

**Xử lý ngoại lệ:**

**catch (IOException e) { ... }:** Nếu có bất kỳ lỗi nào xảy ra trong quá trình ghi file (ví dụ: file không tồn tại, không đủ quyền truy cập, lỗi I/O), ngoại lệ IOException sẽ được bắt và xử lý.

**System.out.println("Đã xảy ra lỗi khi ghi file: " + e.getMessage());:** In ra thông báo lỗi để người dùng biết.

**KẾT LUẬN**

Một trong những lợi thế rõ rệt của việc sử dụng Java để xây dựng hệ thống điểm danh sinh viên qua command line là tính hiệu quả và tiện lợi trong việc triển khai. Với Java, việc xử lý dữ liệu và giao tiếp với người dùng qua dòng lệnh là rất đơn giản và dễ hiểu. Chương trình có thể yêu cầu giảng viên hoặc sinh viên nhập thông tin vào hệ thống, sau đó tự động ghi nhận và lưu trữ kết quả điểm danh. Một hệ thống điểm danh được xây dựng bằng Java có thể hỗ trợ các thao tác như nhập danh sách sinh viên, kiểm tra sự có mặt của từng sinh viên, và lưu trữ kết quả dưới dạng file văn bản hoặc cơ sở dữ liệu.

Java cung cấp các thư viện và công cụ giúp lập trình viên dễ dàng thao tác với dữ liệu và thực hiện các phép toán logic, làm cho chương trình điểm danh sinh viên trở nên nhanh chóng và chính xác. Việc thực hiện trên dòng lệnh cũng giúp giảm thiểu sự phức tạp của giao diện đồ họa, đặc biệt trong các môi trường không yêu cầu giao diện người dùng phức tạp.

Dù hệ thống điểm danh trên dòng lệnh không yêu cầu giao diện đồ họa phức tạp, nó vẫn có thể đảm bảo được tính bảo mật khi lưu trữ thông tin. Java hỗ trợ nhiều phương thức bảo mật và mã hóa thông qua các thư viện như Java Cryptography Architecture (JCA), giúp bảo vệ dữ liệu của sinh viên khỏi sự truy cập trái phép. Tuy nhiên, cần lưu ý rằng bảo mật trong hệ thống này còn phụ thuộc vào cách triển khai và quản lý dữ liệu sau khi được ghi nhận. Để đảm bảo an toàn, các dữ liệu có thể được mã hóa và lưu trữ trên cơ sở dữ liệu an toàn hoặc trong các file có quyền truy cập hạn chế.

Hệ thống điểm danh sinh viên triển khai bằng Java rất linh hoạt và có thể dễ dàng phát triển thêm các tính năng mới. Ví dụ, hệ thống có thể được mở rộng để bao gồm các chức năng như tự động gửi thông báo qua email hoặc SMS cho giảng viên khi có sinh viên vắng mặt hoặc để tạo báo cáo tự động về tỷ lệ tham gia của sinh viên.

Hệ thống có thể dễ dàng tích hợp với các công cụ quản lý học tập khác hoặc kết nối với cơ sở dữ liệu để cung cấp thông tin chi tiết hơn về hành vi học tập của sinh viên. Với việc sử dụng Java, các lập trình viên có thể tối ưu hóa và cải tiến hệ thống theo yêu cầu của trường học hoặc giảng viên, mở ra nhiều khả năng ứng dụng thực tế.

Mặc dù hệ thống điểm danh sinh viên viết bằng Java qua command line mang lại nhiều lợi ích, nhưng cũng không thiếu các thách thức. Đầu tiên, việc triển khai hệ thống trên môi trường dòng lệnh yêu cầu người dùng có một số kiến thức cơ bản về lập trình và sử dụng dòng lệnh, điều này có thể gây khó khăn cho những người không quen với công nghệ.

Bên cạnh đó, hệ thống này cũng có thể gặp phải một số vấn đề về bảo trì, đặc biệt khi dữ liệu ngày càng lớn và yêu cầu khả năng quản lý dữ liệu tốt hơn. Để khắc phục vấn đề này, việc tích hợp cơ sở dữ liệu hoặc nâng cấp hệ thống lên các giao diện người dùng đồ họa có thể là một giải pháp khả thi, mặc dù điều này sẽ làm tăng độ phức tạp của hệ thống.

Một hệ thống điểm danh sinh viên viết bằng Java giúp dễ dàng theo dõi và kiểm tra sự có mặt của sinh viên trong mỗi buổi học. Tất cả dữ liệu về điểm danh sẽ được lưu trữ dưới dạng các file văn bản hoặc cơ sở dữ liệu, giúp người dùng dễ dàng kiểm tra và truy xuất thông tin khi cần thiết. Hệ thống này cũng giúp giảm thiểu khả năng gian lận trong quá trình điểm danh, vì dữ liệu được ghi nhận tự động và chính xác.

Hệ thống điểm danh sinh viên triển khai bằng Java trên môi trường dòng lệnh mang lại nhiều lợi ích như tiết kiệm thời gian, tính chính xác cao và khả năng theo dõi sự có mặt của sinh viên một cách minh bạch. Với khả năng linh hoạt và dễ phát triển, hệ thống này có thể được mở rộng để phục vụ nhiều mục đích khác nhau trong môi trường giáo dục. Tuy nhiên, hệ thống cũng đối mặt với một số thách thức, đặc biệt về mặt bảo trì và yêu cầu người dùng có kiến thức cơ bản về công nghệ. Dù vậy, nếu được triển khai đúng cách, hệ thống điểm danh sinh viên bằng Java trên dòng lệnh là một công cụ hữu ích giúp cải thiện chất lượng quản lý lớp học và tạo ra một môi trường học tập hiệu quả hơn.

**DANH MỤC TÀI LIỆU THAM KHẢO**

1. Design Patterns: Elements of Reusable Object-Oriented Software, Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (Gang of Four), 1994, Addison-Wesley

[2]. Giáo trình C++ & Lập trình hướng đối tượng, TS. Trần Anh Thắng; TS. Nguyễn Phương Huy(2024), NXB Khoa Học & Kỹ Thuật

[3]. James F. Kurose and Keith W. Ross (2013), Computer Networking: A top-down approach sixth Edition, Pearson Education.

**PHỤ LỤC**

[4]. Nguyễn Hồng Sơn (2007), *Giáo trình hệ thống Mạng máy tính CCNA* (Semester 1), NXB Lao động xã hội.

[5]. Phạm Quốc Hùng (2017), *Đề cương bài giảng Mạng máy tính*, Đại học SPKT Hưng Yên.

[6]. James F. Kurose and Keith W. Ross (2013), *Computer Networking: A top-down approach sixth Edition*, Pearson Education.