TRƯỜNG ĐẠI HỌC XÂY DỰNG

KHOA CÔNG NGHỆ THÔNG TIN

**- - - - 🙡🕮🙣 - - - -**

ĐỒ ÁN

**PHƯƠNG PHÁP LẬP TRÌNH**

**HƯỚNG ĐỐI TƯỢNG**

**ĐỀ TÀI: QUẢN LÝ ĐIỂM THI SINH VIÊN**

Giảng viên hướng dẫn: **Thái Thị Nguyệt**

Nhóm môn học: **58PM2**

Nhóm sinh viên thực hiện:

* **Lương Văn Sáng**
* **Mai Thị Lan Anh**
* **Phạm Thị Thu Thuỳ**
* **Nguyễn Thị Tươi**

HÀ NỘI 06/2016

**MỤC LỤC**

[I. PHÁT BIỂU BÀI TOÁN 3](#_Toc453920781)

[II. THIẾT KẾ TỔNG QUAN 3](#_Toc453920782)

[1. Biểu đồ lớp 3](#_Toc453920783)

[2. Biểu đồ Use Case 4](#_Toc453920784)

[III. PHÂN TÍCH CHƯƠNG TRÌNH 5](#_Toc453920785)

[1. Quản lý sinh viên 5](#_Toc453920786)

[a. Hiện danh sách sinh viên 5](#_Toc453920787)

[b. Thêm mới sinh viên 6](#_Toc453920788)

[c. Sửa sinh viên 7](#_Toc453920789)

[d. Xóa sinh viên 8](#_Toc453920790)

[e. Tìm kiếm sinh viên 9](#_Toc453920791)

[2. Quản lý môn học 10](#_Toc453920792)

[a. Hiện danh sách môn học 10](#_Toc453920793)

[b. Thêm mới môn học 11](#_Toc453920794)

[c. Sửa môn học 12](#_Toc453920795)

[d. Xóa môn học 13](#_Toc453920796)

[e. Tìm kiếm môn học 14](#_Toc453920797)

[3. Quản lý kết quả 15](#_Toc453920798)

[a. Hiện danh sách kết quả 15](#_Toc453920799)

[b. Thêm mới kết quả 16](#_Toc453920800)

[c. Sửa kết quả 18](#_Toc453920801)

[d. Xóa kết quả 19](#_Toc453920802)

[e. Tìm kiếm kết quả 20](#_Toc453920803)

[4. Một số hình ảnh chương trình 21](#_Toc453920804)

[IV. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 24](#_Toc453920805)

[V. MẪU THIẾT KẾ: PROTOTYPE 25](#_Toc453920806)

[**1.** Mục đích 25](#_Toc453920807)

[**2.** Trình bày 25](#_Toc453920808)

[**3.** Khả năng áp dụng 26](#_Toc453920809)

[**4.** Cấu trúc 27](#_Toc453920810)

[**5.** Thành phần 27](#_Toc453920811)

[**6.** Tương tác 27](#_Toc453920812)

[**7.** Kết quả 27](#_Toc453920813)

[**8.** Cài đặt 28](#_Toc453920814)

[**9.** Ví dụ mẫu 30](#_Toc453920815)

[**10.** Các ứng dụng đã biết 33](#_Toc453920816)

[**11.** Những kiểu mẫu thiết kế có liên quan 34](#_Toc453920817)

[VI. TÀI LIỆU THAM KHẢO 34](#_Toc453920818)

[VII. LỜI KẾT 34](#_Toc453920819)

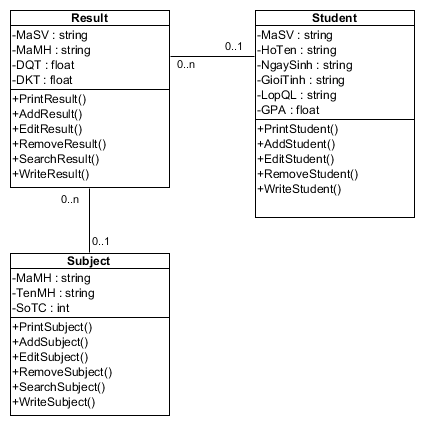
1. PHÁT BIỂU BÀI TOÁN

Chương trình quản lý điểm thi sinh viên có thể giúp công việc quản lý điểm thi của sinh viên trở nên dễ dàng, nhanh chóng, độ chính xác cao.

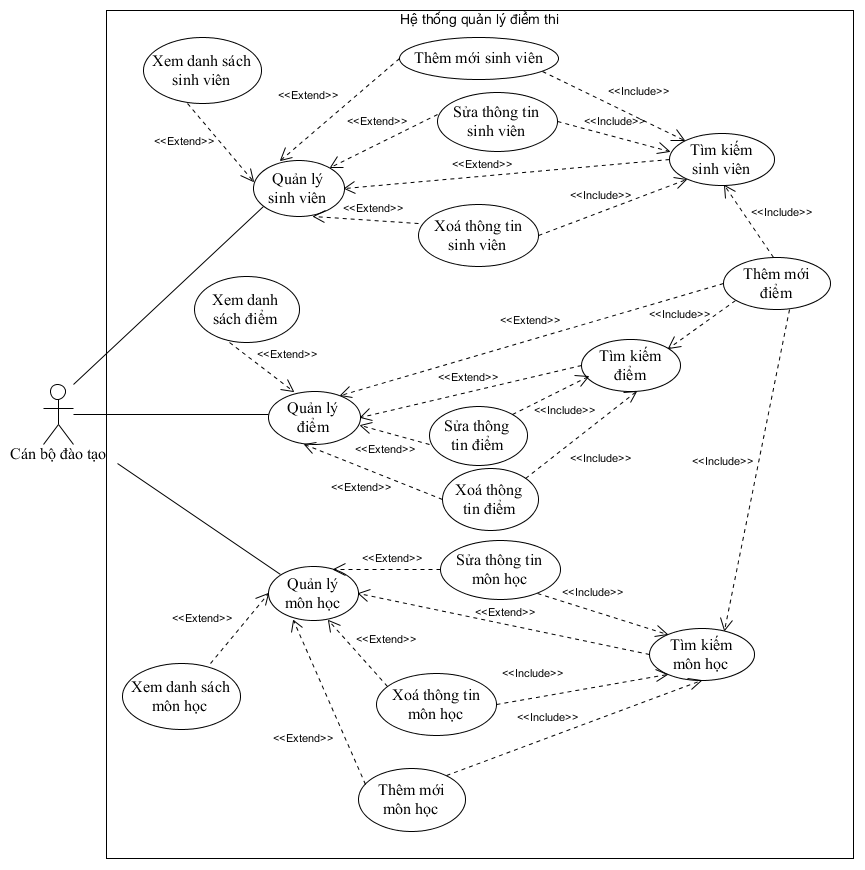
Các chức năng chính của chương trình:

* Quản lý sinh viên: Hiển thị danh sách sinh viên, thêm mới, sửa, xoá, tìm kiếm sinh viên
* Quản lý môn học: Hiển thị danh sách môn học, thêm mới, sửa, xoá, tìm kiếm môn học
* Quản lý kết quả: Hiển thị danh sách kết quả, thêm mới, sửa, xoá, tìm kiếm kết quả

1. THIẾT KẾ TỔNG QUAN
2. Biểu đồ lớp



1. Biểu đồ Use Case



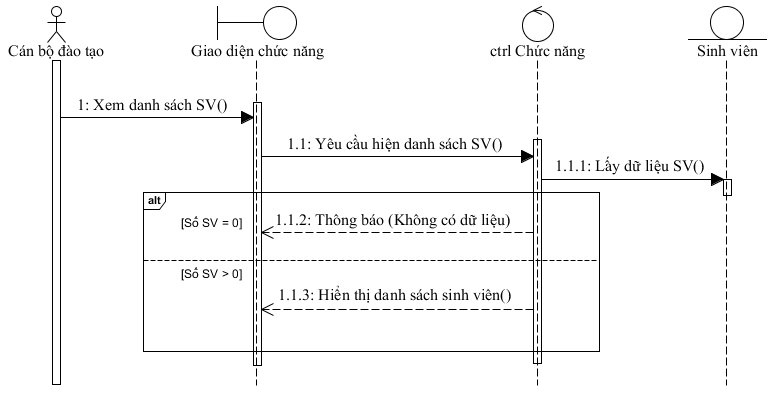
1. PHÂN TÍCH CHƯƠNG TRÌNH
2. Quản lý sinh viên
3. Hiện danh sách sinh viên

Đầu vào: Không có

Mô tả: Trong bảng chọn chức năng, nhập 1 để chọn chức năng “ Hiện danh sách sinh viên”

* Chương trình sẽ hiển thị ra màn hình bảng Danh sách sinh viên bao gồm các thông tin của sinh viên được chia theo các cột: Mã sinh viên, Họ tên, Ngày sinh, Giới tính, Lớp quản lí, Điểm trung bình

Biểu đồ trình tự:

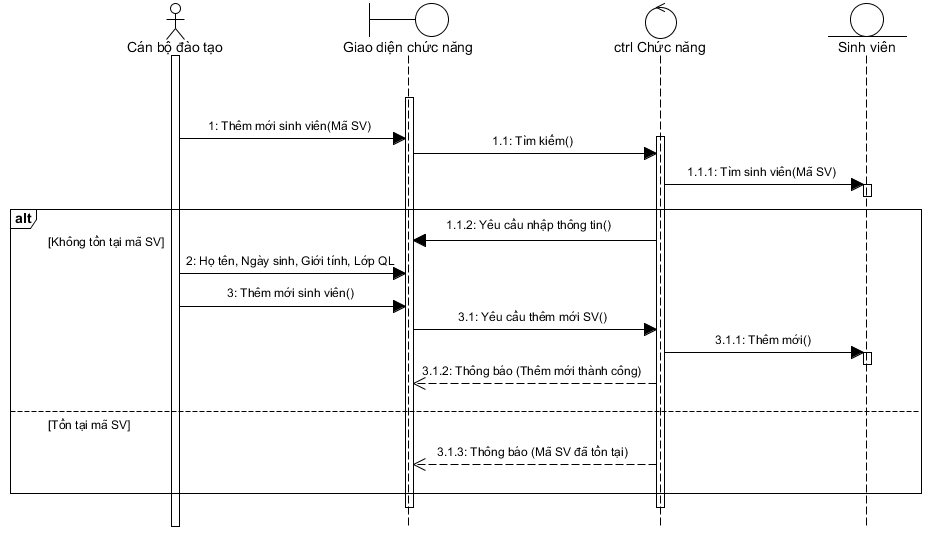


1. Thêm mới sinh viên

Đầu vào: Mã sinh viên, họ tên, ngày sinh, giới tính, lớp quản lí

Mô tả: Trong bảng chọn chức năng, nhập 4 để chọn chức năng “ Thêm mới sinh viên”

* Chương trình sẽ yêu cầu nhập vào Mã sinh viên:
  + Nếu mã sinh viên tồn tại, chương trình sẽ đưa ra thông báo “Mã sinh viên đã tồn tại trong hệ thống!” và yêu cầu nhập mã khác.
  + Nếu mã sinh viên chưa tồn tại, chương trình sẽ tiếp tục yêu cầu nhập vào Họ tên sinh viên, ngày sinh, giới tính và lớp quản lí.
* Sau khi nhập xong, chương trình sẽ lưu thông tin vừa nhập vào hệ thống và đưa ra thông báo “Thêm mới sinh viên thành công”.

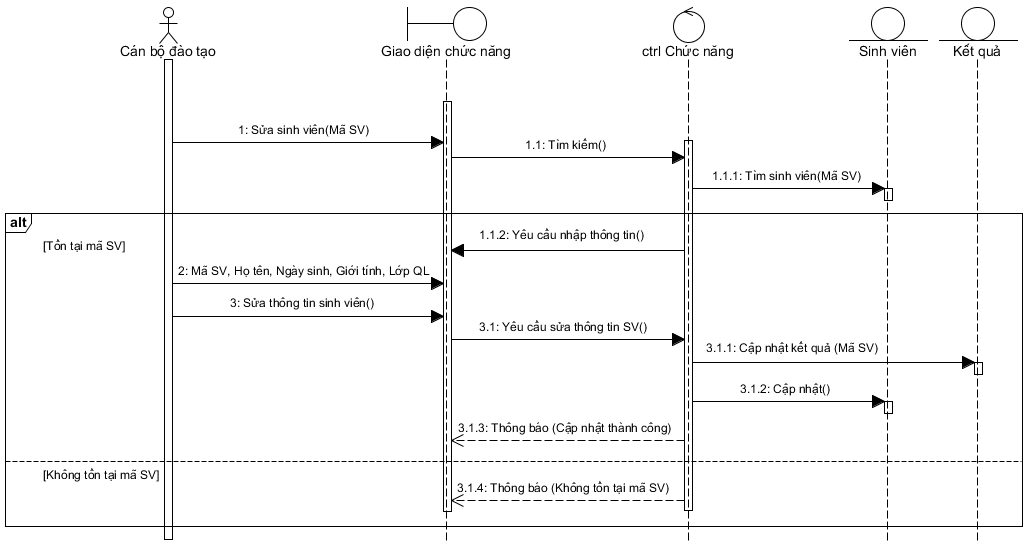
Biểu đồ trình tự:

1. Sửa sinh viên

Đầu vào: Mã sinh viên, Tên sinh viên, ngày sinh, giới tính và lớp quản lí.

Mô tả: Trong bảng chọn chức năng, nhập 7 để chọn chức năng “Sửa sinh viên”

* Chương trình sẽ yêu cầu nhập vào Mã sinh viên cần sửa:
  + Nếu mã sinh viên không tồn tại, chương trình sẽ đưa ra thông báo không tìm thấy sinh viên có mã sinh viên vừa nhập và yêu cầu nhập mã khác.
  + Nếu mã sinh viên có tồn tại trong hệ thống, chương trình sẽ tiếp tục yêu cầu nhập vào thông tin mới gồm: Mã sinh viên, Tên sinh viên, ngày sinh, giới tính và lớp quản lí.
* Sau khi nhập xong, chương trình sẽ lưu thông tin vừa nhập vào hệ thống, đồng thời tìm các kết quả có mã sinh viên cũ để cập nhật mã mới và đưa ra thông báo “ Cập nhật thông tin thành công”.

Biểu đồ trình tự:

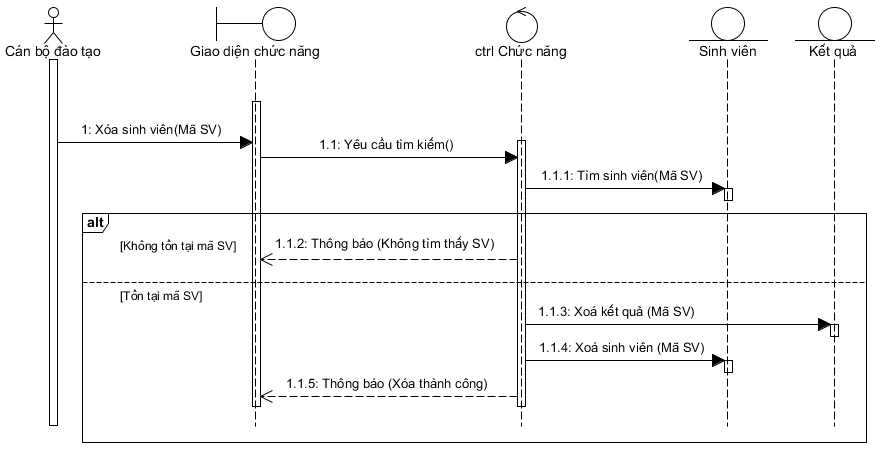
1. Xóa sinh viên

Đầu vào: Mã sinh viên

Mô tả: Trong bảng chọn chức năng, nhập 11 để chọn chức năng “Xóa sinh viên”

* Chương trình sẽ yêu cầu nhập vào Mã sinh viên cần xóa:
  + Nếu mã sinh viên không tồn tại, chương trình sẽ đưa ra thông báo không tìm thấy sinh viên có mã sinh viên vừa nhập và yêu cầu nhập mã khác.
  + Nếu mã sinh viên có tồn tại trong hệ thống, chương trình sẽ thực hiện xóa thông tin sinh viên, các kết quả của sinh viên có mã vừa nhập và hiện thị thông báo “ Xóa thành công sinh viên”

Biểu đồ trình tự:

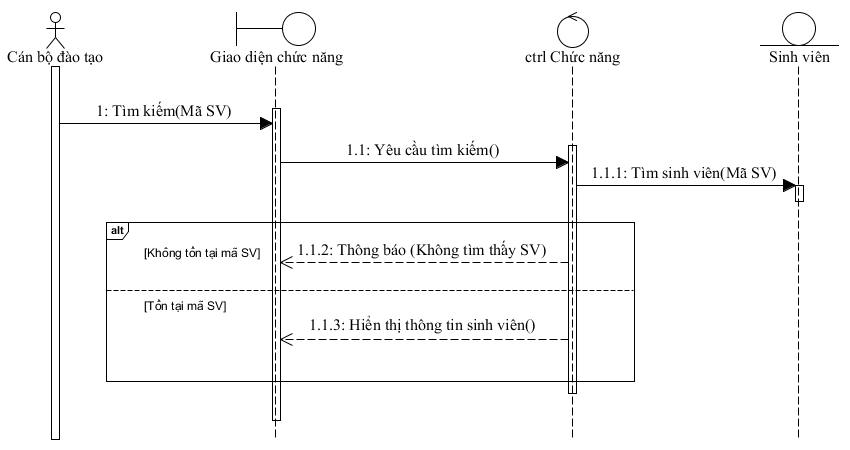


1. Tìm kiếm sinh viên

Đầu vào: Mã sinh viên

Mô tả: Trong bảng chọn chức năng, nhập 13 để chọn chức năng “Tìm kiếm sinh viên”

* Chương trình sẽ yêu cầu nhập vào Mã sinh viên cần tìm kiếm:
  + Nếu mã sinh viên không tồn tại, chương trình sẽ đưa ra thông báo không tìm thấy sinh viên có mã sinh viên vừa nhập.
  + Nếu mã sinh viên có tồn tại trong hệ thống, chương trình sẽ thực hiện hiển thị thông tin sinh viên có mã vừa nhập bao gồm: Mã sinh viên, Họ tên, Ngày sinh, Giới tính, Lớp quản lý và Điểm trung bình.

Biểu đồ trình tự:

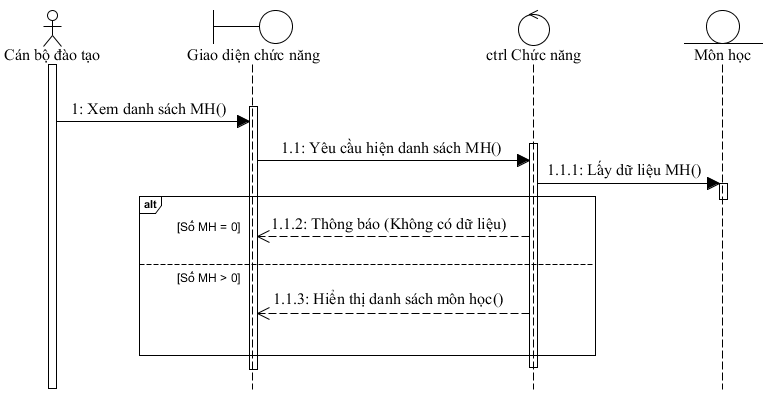
1. Quản lý môn học
2. Hiện danh sách môn học

Đầu vào: Không có

Mô tả: Trong bảng chọn chức năng, nhập 2 để chọn chức năng “Hiện danh sách môn học”

* Chương trình sẽ hiển thị ra màn hình bảng Danh sách môn học bao gồm các thông tin của môn học được chia theo các cột: Mã môn học, Tên môn học, Số tín chỉ.

Biểu đồ trình tự:

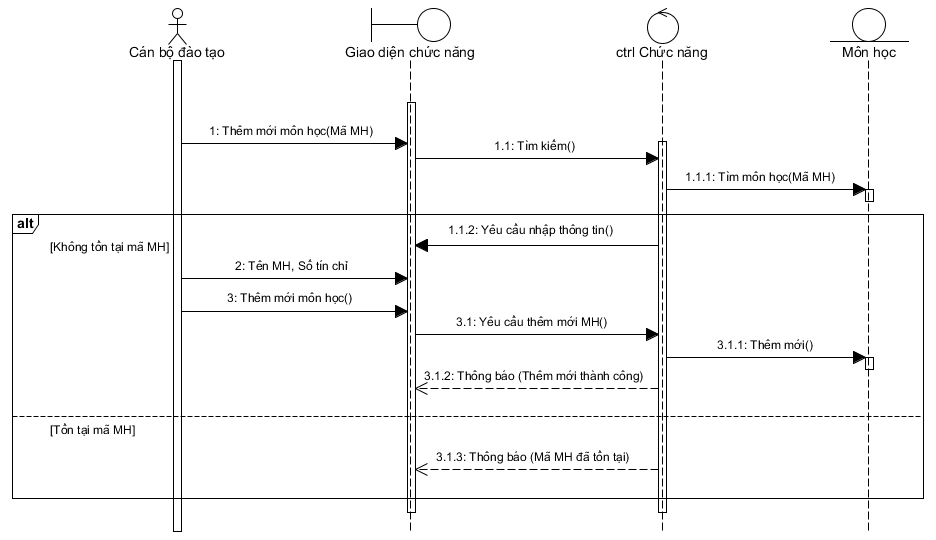


1. Thêm mới môn học

Đầu vào: Mã môn học, Tên môn học, Số tín chỉ.

Mô tả: Trong bảng chọn chức năng, nhập 5 để chọn chức năng “Thêm mới môn học”

* Chương trình sẽ yêu cầu nhập vào Mã môn học:
  + Nếu mã môn học đã tồn tại, chương trình sẽ đưa ra thông báo “Mã môn học đã tồn tại trong hệ thống!” và yêu cầu nhập mã khác.
  + Nếu mã môn học chưa tồn tại, chương trình sẽ tiếp tục yêu cầu nhập vào Tên môn học và Số tín chỉ.
* Sau khi nhập xong, chương trình sẽ lưu thông tin vừa nhập vào hệ thống và đưa ra thông báo “Thêm mới môn học thành công”.

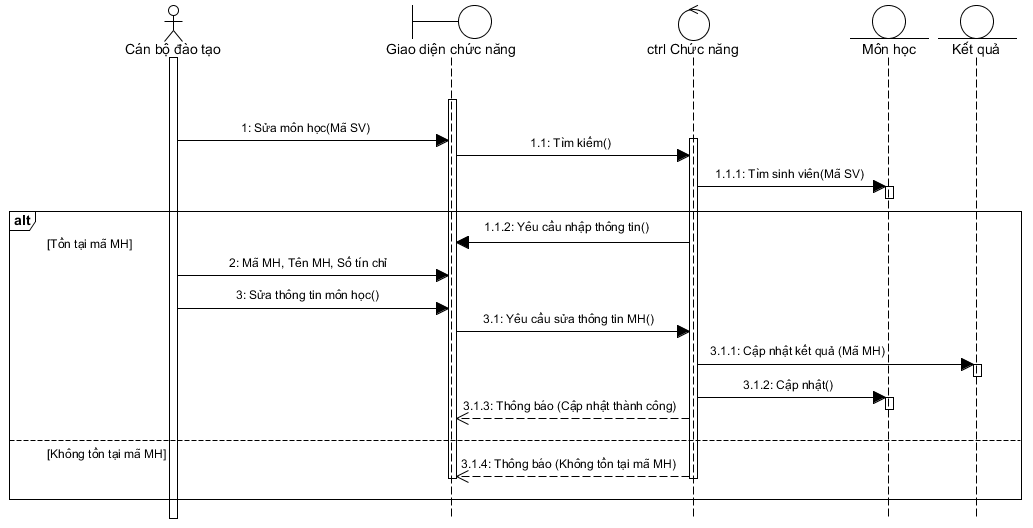
Biểu đồ trình tự:

1. Sửa môn học

Đầu vào: Mã môn học, Tên môn học, Số tín chỉ.

Mô tả: Trong bảng chọn chức năng, nhập 8 để chọn chức năng “Sửa môn học”

* Chương trình sẽ yêu cầu nhập vào Mã môn học cần sửa:
* Nếu mã môn học không tồn tại, chương trình sẽ đưa ra thông báo hông tìm thấy kết quả nào có mã môn học vừa nhập và yêu cầu nhập mã khác.
* Nếu mã môn học có tồn tại trong hệ thống, chương trình sẽ tiếp tục yêu cầu nhập vào thông tin mới gồm: Mã môn học, Tên môn học và Số tín chỉ.
* Sau khi nhập xong, chương trình sẽ lưu thông tin vừa nhập vào hệ thống, đồng thời tìm các kết quả có mã môn học cũ để cập nhật mã mới và đưa ra thông báo “Cập nhật thông tin thành công”.

Biểu đồ trình tự:

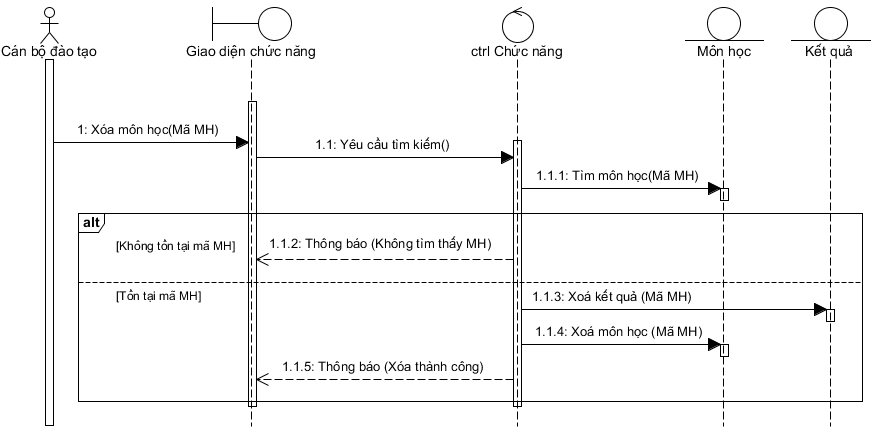
1. Xóa môn học

Đầu vào: Mã môn học

Mô tả: Trong bảng chọn chức năng, nhập 11 để chọn chức năng “Xóa môn học”

* Chương trình sẽ yêu cầu nhập vào Mã môn học cần xóa:
* Nếu mã môn học không tồn tại, chương trình sẽ đưa ra thông báo không tìm thấy kết quả nào có mã môn học vừa nhập và yêu cầu nhập mã khác.
* Nếu mã môn học có tồn tại trong hệ thống, chương trình sẽ thực hiện xóa thông tin môn học, các kết quả của môn học có mã vừa nhập và hiện thị thông báo “Xóa thành công môn học”

Biểu đồ trình tự:

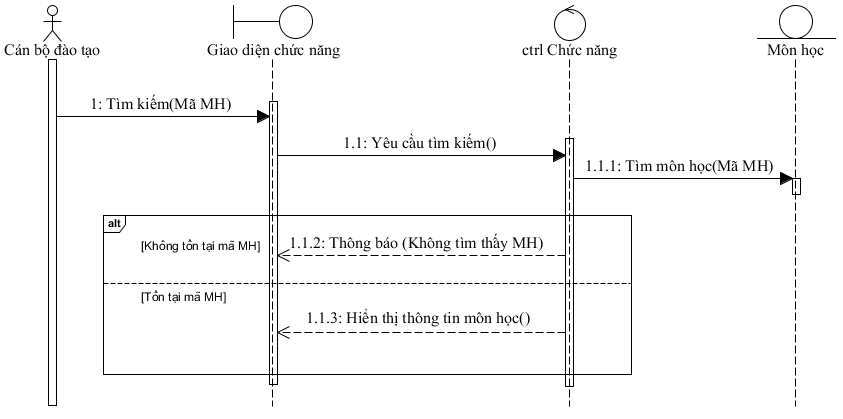


1. Tìm kiếm môn học

Đầu vào: Mã môn học

Mô tả: Trong bảng chọn chức năng, nhập 14 để chọn chức năng “Tìm kiếm môn học”

* Chương trình sẽ yêu cầu nhập vào Mã môn học cần tìm kiếm:
* Nếu mã môn học không tồn tại, chương trình sẽ đưa ra thông báo không tìm thấy kết quả nào có mã môn học vừa nhập.
* Nếu mã môn học có tồn tại trong hệ thống, chương trình sẽ thực hiện hiển thị thông tin môn học có mã vừa nhập gồm: Mã môn học, Tên môn học và Số tín chỉ.

Biểu đồ trình tự:

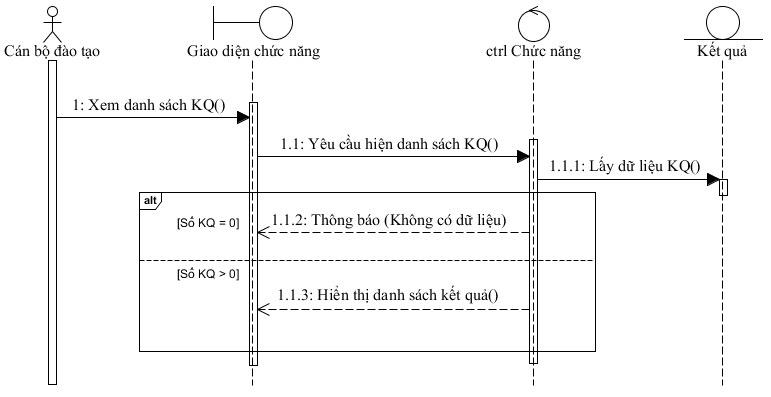
1. Quản lý kết quả
2. Hiện danh sách kết quả

Đầu vào: Không có

Mô tả: Trong bảng chọn chức năng, nhập 3 để chọn use case “Hiện danh sách kết quả”.

* Chương trình sẽ hiện ra 1 bảng danh sách các kết quả của sinh viên gồm: Mã MH, Mã SV, điểm quá trình, điểm kết thúc và điểm trung bình các môn của sinh viên.

Biểu đồ trình tự:



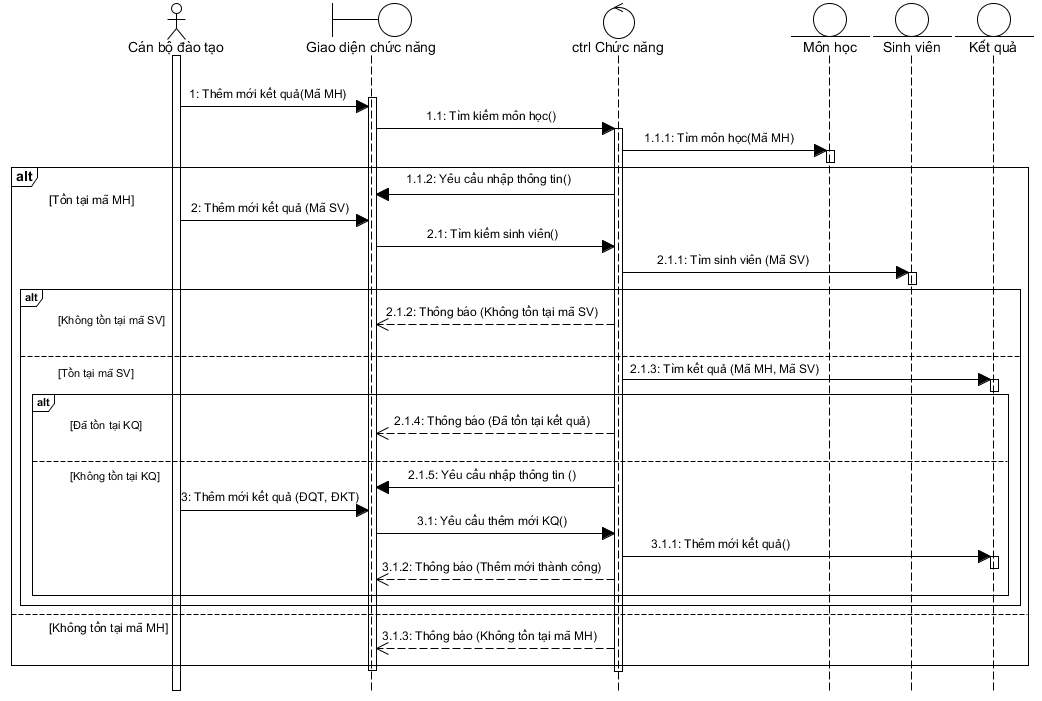
1. Thêm mới kết quả

Đầu vào: Mã môn học, mã sinh viên, điểm quá trình, điểm kết thúc

Mô tả: Trong bảng chọn chức năng, nhập 6 để chọn chức năng thêm mới kết quả. Có 2 chức năng con hiện ra là “Thêm mới kết quả theo mã sinh viên” và “Thêm mới kết quả theo mã môn học”.

* Trường hợp 1: Thêm mới kết quả theo mã sinh viên – Chọn 1
  + Chương trình sẽ yêu cầu nhập vào mã sinh viên cần nhập điểm, nếu nhập một mã sinh viên không tồn tại, chương trình đưa ra thông báo “Mã SV xxxxxxx không tồn tại trong hệ thống!” và yêu cầu nhập lại.
  + Nếu mã sinh viên có tồn tại, chương trình sẽ yêu cầu nhập số môn học cần nhập điểm. Sau đó, chương trình sẽ yêu cầu nhập mã môn học từng môn để nhập điểm, nếu mã môn học không tồn tại, chương trình sẽ đưa ra thông báo không tồn tại môn học với mã đó và yêu cầu nhập lại.
  + Nếu mã môn học có tồn tại, chương trình sẽ kiểm tra xem có tồn tại kết quả với mã sinh viên và mã môn học vừa nhập không, nếu tồn tại rồi thì đưa ra thông báo đã tồn tại kết quả với mã sinh viên và mã môn học vừa nhập.
  + Nếu chưa tồn tại kết quả với mã sinh viên và mã môn học đó trong hệ thống, chương trình sẽ yêu cầu nhập điểm quá trình và điểm kết thúc của sinh viên vào. Sau đó đưa thông tin vào hệ thống và thông báo “Thêm mới kết quả thành công!”
* Trường hợp 2: Thêm mới kết quả theo mã môn học – Chọn 2
  + Chương trình sẽ yêu cầu nhập vào mã môn học cần nhập điểm, nếu nhập một mã môn học không tồn tại, chương trình đưa ra thông báo “Mã MH yyyyy không tồn tại trong hệ thống” và yêu cầu nhập lại.
  + Nếu mã môn học có tồn tại, chương trình sẽ yêu cầu nhập số sinh viên cần nhập điểm. Sau đó, chương trình sẽ yêu cầu nhập mã sinh viên từng sinh viên để nhập điểm, nếu mã sinh viên không tồn tại, chương trình sẽ đưa ra thông báo không tồn tại sinh viên với mã đó và yêu cầu nhập lại.
  + Nếu mã sinh viên có tồn tại, chương trình sẽ kiểm tra xem có tồn tại kết quả với mã sinh viên và mã môn học vừa nhập không, nếu tồn tại rồi thì đưa ra thông báo đã tồn tại kết quả với mã sinh viên và mã môn học vừa nhập.
  + Nếu chưa tồn tại kết quả với mã sinh viên và mã môn học đó trong hệ thống, chương trình sẽ yêu cầu nhập điểm quá trình và điểm kết thúc của sinh viên vào. Sau đó đưa thông tin vào hệ thống và thông báo “Thêm mới kết quả thành công!”

Biểu đồ trình tự:



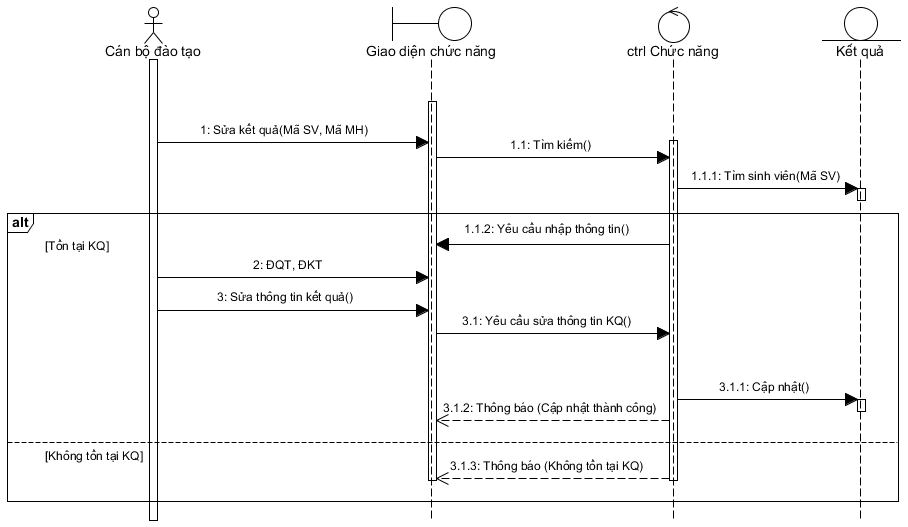
1. Sửa kết quả

Đầu vào: Mã sinh viên, mã môn học, điểm quá trình, điểm kiểm tra

Mô tả: Nhấn phím 9 để chọn use case “Sửa kết quả”.

* Hệ thống sẽ yêu cầu nhập thông tin cần chỉnh sửa bằng việc nhập mã sinh viên và mã môn học.
  + Nếu 1 trong 2 hay cả 2 loại mã sinh viên và mã môn học không tồn tại trong hệ thống, hệ thống đưa ra thông báo: “Không tìm thấy kết quả có MaSV = xxxxx, MaMH = yyyyy” mà người dùng vừa nhập, và lại hiện ra bảng chọn chức năng.
  + Nếu tồn tại mã sinh viên và mã môn học, thông báo “Nhập thông tin mới” hiện ra, cho phép người dùng nhập Điểm quá trình (DQT), điểm kết thúc (DKT). Cập nhật thành công, hệ thống thông báo: “Cập nhật thông tin thành công

Biểu đồ trạng thái:



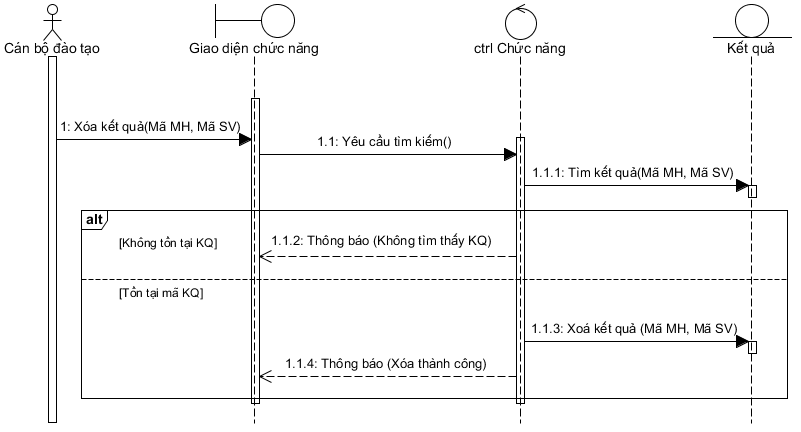
1. Xóa kết quả

Đầu vào: Mã sinh viên, mã môn học

Mô tả: Nhấn phím 12 để chọn use case “Xóa kết quả”.

* Hệ thống đưa ra thông báo “Nhập mã SV cần xóa kết quả”. Sau khi người dùng nhập mã sinh viên thì hệ thống đưa thông báo “ Nhập mã MH cần xóa kết quả”
  + Nếu kết quả của mã sinh viên và mã môn học không tồn tại trong hệ thống, hệ thống thông báo “Không tìm thấy kết quả có MaSV = xxxxx, MaMH= yyyy”. Và yêu cầu người dùng nhập lại.
  + Nếu kết quả của mã sinh viên, mã môn học có trong hệ thống, chương trình sẽ xoá kết quả khỏi hệ thống và thông báo “Xoá thành công”.

Biểu đồ trình tự:

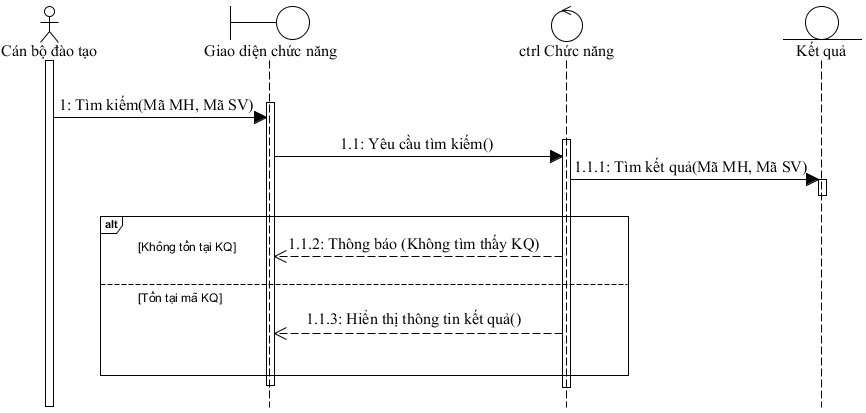


1. Tìm kiếm kết quả

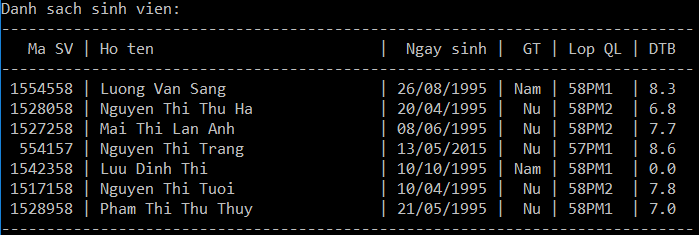
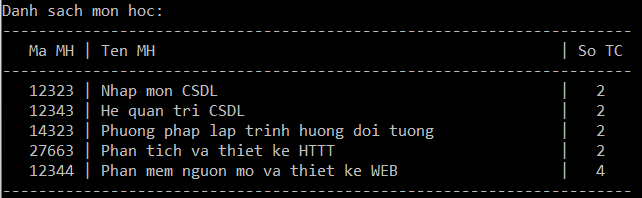
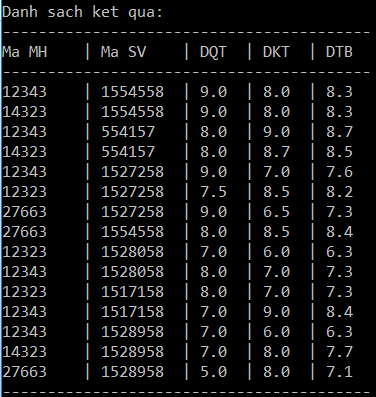
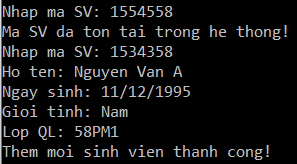
Đầu vào: Mã sinh viên, mã môn học

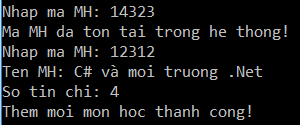
Mô tả: Nhấn chọn chức năng 15 “Tìm kiếm kết quả” từ bảng chọn chức năng.

* Hệ thống yêu cầu người dùng “Nhập MaMH cần tìm” và “Nhập MaSV cần tìm”.
  + Nếu Mã sinh viên và Mã môn học không tồn tại trong hệ thống, hệ thống thông báo: “không tìm thấy kết quả có MaSV= xxxxx, MaMH= yyyyy”. Bảng chọn chức năng hiện ra yêu cầu chọn lại.
  + Mã sinh viên và Mã môn học tồn tại: Hệ thống hiện ra 1 bảng bao gồm các thông tin về kết quả bao gồm Mã sinh viên, Mã môn học, Điểm quá trình, Điểm kết thúc, Điểm trung bình

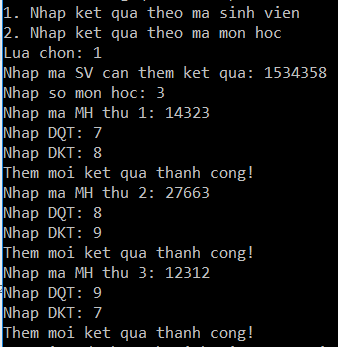
Biểu đồ trình tự:

1. Một số hình ảnh chương trình

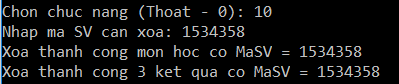
* Hiển thị danh sách sinh viên:
* Hiển thị danh sách môn học:
* Hiển thị danh sách kết quả:
* Thêm mới sinh viên:
* Thêm mới môn học:



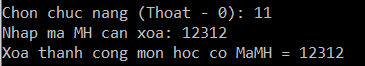
* Thêm mới kết quả:



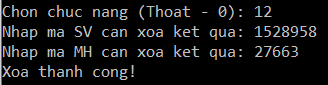
* Xoá sinh viên:



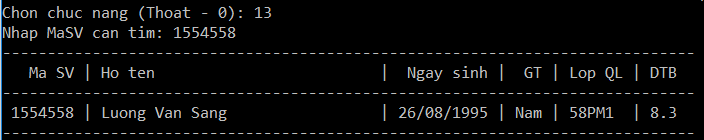
* Xoá môn học:



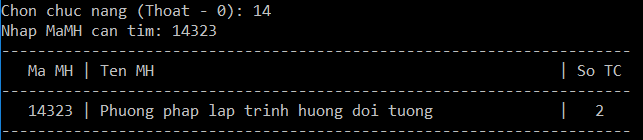
* Xoá kết quả:



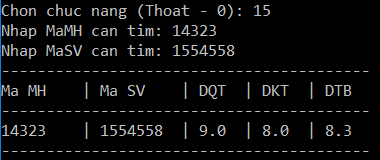
* Tìm kiếm sinh viên:



* Tìm kiếm môn học:



* Tìm kiếm kết quả:



1. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

* Kết luận:
  + Chương trình đã cung cấp được các chức năng cơ bản của quản lý điểm thi sinh viên là lưu trữ và xử lý các thông tin của sinh viên, môn học và điểm thi (kết quả).
  + Trình bày chương trình rõ ràng, dễ thêm mới và chỉnh sửa các tính năng
* Hướng phát triển:
  + Có thể sử dụng các công cụ khác để nâng cấp chương trình, sử dụng giao diện trực quan thay vì thao tác thông qua cửa sổ Console bất tiện.
  + Có thể thêm các Lớp mới để quản lý thêm các thông tin khác như Bộ môn, Khoa, v.v
  + Có thể thêm các tính năng mới để giúp chương trình hoàn thiện hơn như lưu trữ thông tin bằng một hệ quản trị cơ sở dữ liệu thay vì lưu ở dạng tệp như hiện tại.

1. MẪU THIẾT KẾ: PROTOTYPE

1. Mục đích

Đặc tả các loại đối tượng mà có thể tạo ra bằng cách dùng 1 thể hiện nguyên mẫu, và tạo ra các đối tượng mới bằng việc sao chép nguyên mẫu này.

1. Trình bày

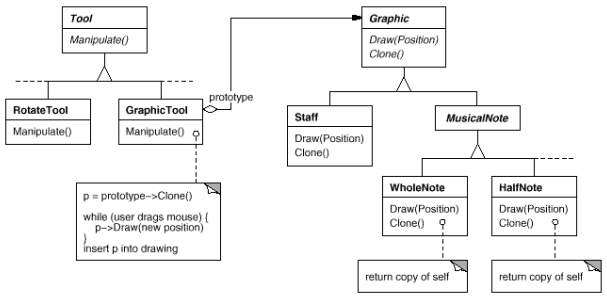
Bạn có thể xây dựng 1 trình chỉnh sửa cho các bản nhạc bằng cách tuỳ chỉnh một framework chung cho các trình chỉnh sửa đồ họa và thêm các đối tượng mới như các nốt chính, các nốt lặng, các dòng kẻ. Framework trình chỉnh sửa có thể có 1 bảng các công cụ để thêm các đối tượng cho bản nhạc. Bảng này cũng có thể gồm các công cụ để lựa chọn, di chuyển và các thao tác khác với các đối tượng âm nhạc. Người dùng sẽ chọn công cụ nốt đen và dùng nó để thêm nốt đen vào bản nhạc. Hoặc ta có thể dịch chuyển nốt nhạc lên xuống trên các dòng kẻ nhạc để thay đổi cao độ.

Giả sử rằng framework cung cấp một lớp trừu tượng Graphic cho các thành phần đồ hoạ, giống như nốt và dòng kẻ. Hơn nữa, nó sẽ cung cấp cho ta 1 lớp trừu tượng Tool để xác định các công cụ giống như trong bảng công cụ ở trên. Framework cũng xác định trước một lớp con GraphicTool cho những công cụ để tạo ra trường các đối tượng đồ hoạ và thêm chúng vào tài liệu.

Nhưng những đại diện GraphicTool này có 1 vấn đề về thiết kế framework. Các lớp cho nốt và dòng kẻ thì phù hợp với ứng dụng, nhưng lớp GraphicTool lại thuộc về framework. Nó không làm thế nào để tạo ra các thể hiện cho các lớp âm nhạc của chúng ta để thêm vào bản nhạc. Chúng ta có thể tạo lớp con của GraphicTool cho mỗi lọai đối tượng âm nhạc, nhưng điều này sẽ tạo ra nhiều lớp con mà chỉ khác nhau ở loại đối tượng âm nhạc mà chúng thể hiện. Ta biết rằng biên soạn các đối tượng là sự thay thế linh hoạt cho việc tạo ra các lớp con. Vấn đề đặt ra là, làm thế nào mà framework sử dụng nó để tham số hoá các thể hiện của GraphicTool bằng các lớp Graphic mà chúng sẽ được tạo?

Giải pháp nằm ở việc khiến GraphicTool tạo ra 1 Graphic mới bằng cách sao chép 1 thể hiện của lớp con Graphic. Chúng ta có thể gọi thể hiện này là prototype. GraphicTool được tham số hoá bởi prototype nó cần sao chép và thêm vào tài liệu. Nếu tất cả các lớp con Graphic đều có toán tử Clone thì GraphicTool có thể nhân bản bất kì loại Graphic nào.

Trong trình chỉnh sửa của chúng ta, mỗi công cụ thiết kế ra 1 đối tượng âm nhạc là 1 thể hiện của GraphicTool mà được khởi tạo với các prototype khác nhau. Mỗi thể hiện của GraphicTool sẽ sinh ra 1 đối tượng âm nhạc bằng việc nhân bản từ nguyên mẫu và thêm chúng vào bản nhạc.



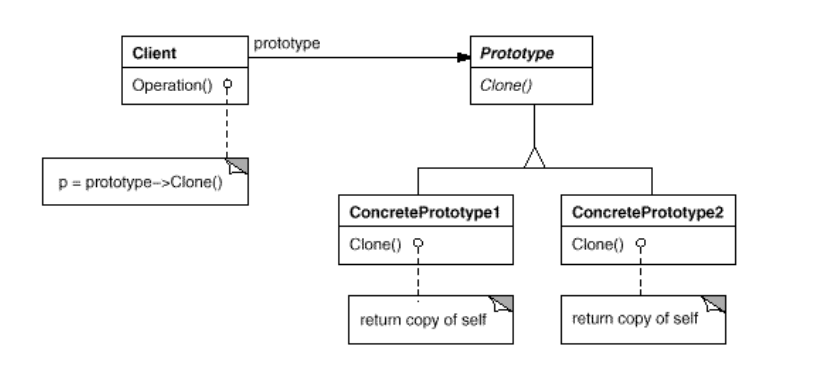
Chúng ta có thể sử dụng mẫu thiết kế Prototype để giảm số lớp cần thêm vào. Chúng ta có các lớp riêng biệt cho các nốt và bán nốt. Nhưng điều đó có thể không cần thiết. Thay vào đó, chúng có thể là thể hiện của cùng 1 lớp được khởi tạo bởi các ảnh nhị phân và quãng khác nhau. Một công cụ để tạo ra các nốt trở thành một GraphicTool có nguyên mẫu là 1 MusicalNote được khởi tạo thành 1 nốt chính. Điều này có thể giảm đi số lượng lớp đáng kể. Nó cũng khiến việc thêm các loại nốt mới vào trong trình chỉnh sửa dễ dàng hơn.

1. Khả năng áp dụng

Sử dụng mẫu thiết kế Prototype khi hệ thống cần phải độc lập trong việc các sản phẩm của nó được cấu thành, tạo ra, thể hiện và

* Khi các lớp cần muốn thể hiện được đặc tả khi chạy chương trình, ví dụ như tải động; hoặc
* Tránh xây dựng 1 lớp phân cấp bậc của nhà máy ngang hàng với xây dựng 1 lớp phân cấp bậc sản phẩm; hoặc
* Khi các thể hiện của 1 lớp có thể chỉ có 1 trong số ít các tổ hợp trạng thái khác nhau. Cũng thuận tiện hơn để thiết lập và nhân bản 1 số lượng lớn những nguyên mẫu phù hợp so với việc thể hiện thủ công 1 lớp mỗi lần 1 trạng thái riêng biệt

1. Cấu trúc



1. Thành phần

* Prototype (Graphic):
  + Khai báo một giao diện để nhân bản chính nó.
* ConcretePrototype (Staff, WholeNote, HaftNote):
  + Thực hiện một thao tác để tự nhân bản nó.
* Client (GraphicTool):
  + Yêu cầu 1 prototype nhân bản chính nó để tạo một đối tượng mới.

1. Tương tác

Một client yêu cầu 1 prototype tự nhân bản chính nó.

1. Kết quả

Prototype có rất nhiều hệ quả giống với Abstract Factory(99) và Builder(110): Nó ẩn các lớp sản phẩm cụ thể từ client, bằng cách đó làm giảm số lượng tên mà các client biết. Hơn thế nữa, những mẫu thiết kế này để client làm việc với những lớp ứng dụng cụ thể mà không có sự thay đổi nào.

Các lợi ích khác của mẫu thiết kế Prototype được liệt kê sau đây:

1. *Thêm mới và loại bỏ các sản phẩm trong thời gian chạy*. Prototypes cho phép bạn kết hợp một lớp sản phẩm cụ thể vào một hệ thống bằng cách đơn giản là đưa cho client một ví dụ điển hình. Cách này linh hoạt hơn một chút so với các mẫu thiết kế tạo lập khác, vì client có thể cài đặt và loại bỏ các nguyên mẫu trong thời gian chạy.
2. *Xác định các đối tượng mới bằng cách biến đổi giá trị.* Các hệ thống biến đổi động cao cho phép bạn định nghĩa cách xử lý mới thông qua bản chất của đổi tượng bằng cách chỉ rõ biến của đối tượng, để lấy ví dụ chứ không phải để định nghĩa một lớp mới. Bạn sẽ dễ dàng định nghĩa những tính chất mới của đối tượng bằng cách lấy thể hiện cụ thể của những lớp đang tồn tại và ghi nhận nó như một bản mẫu của những đối tượng client. Môt client có thể đưa ra cách xử lý mới bằng cách gán chức năng cho nguyên mẫu.

Loại thiết kế này để người dùng định nghĩa các lớp mới mà không phải lập trình. Thực tế, nhân bản prototype cũng tương tự với việc thể hiện một lớp. Mẫu thiết kế Prototype có thể giảm thiểu 1 số lượng lớn các lớp mà hệ thống cần. Trong trình chỉnh sửa âm nhạc của chúng ta, một lớp GraphicTool có thể tạo ra đa dạng các đối tượng âm nhạc.

1. *Xác định các đối tượng mới bằng cách biến đổi cấu trúc.* Có rất nhiều ứng dụng xây dựng đối tượng từ những phần và những thành phần con. Ví dụ như các trình biên tập cho thiết kế mạch, xây dựng các mạch từ các mạch con. Để thuận tiện, các ứng dụng như vậy cho phép bạn khởi tạo các cấu trúc phức tạp, các cấu trúc do người dùng tự định nghĩa, và sử dụng lại các mạch con cụ thể.

Các mô hình nguyên mẫu cũng hỗ trợ điều này rất nhiều. Chúng tôi chỉ đơn giản là thêm các mạch con như một nguyên mẫu vào các yếu tố mạch có sẵn. Miễn là các đối tượng mạch tổng hợp thực hiện Clone như một sự sao chép “sâu” (deep copy), thì những mạch có cấu trúc khác nhau có thể là các prototype.

1. *Làm giảm sự phân lớp.* Factory Method (121) thường tạo ra một hệ thống các lớp Creator song song với hệ thống phân cấp lớp sản phẩm. Mô hình nguyên mẫu cho phép bạn sao chép một nguyên mẫu thay vì yêu cầu phương pháp Factory để tạo ra một đối tượng mới. Do đó bạn hoàn toàn không cần một hệ thống phân cấp lớp Creator. Lợi ích này áp dụng chủ yếu với các ngôn ngữ như C ++ không xử lý các lớp như là đối tượng lớp thứ nhất. Các ngôn ngữ, như Smalltalk và Objective C, có xử lý nhưng đưa ra ít lợi ích hơn, vì bạn luôn có thể sử dụng một lớp đối tượng như một sự tạo ra. Các lớp đối tượng vốn dĩ đã hoạt động như nguyên mẫu trong những ngôn ngữ này.
2. *Cấu hình một ứng dụng với những lớp động*. Một vài môi trường “run- time” cho phép bạn tải động các lớp vào một ứng dụng. Các mẫu thiết kế Prototype là chìa khóa để khai thác một cách khéo léo một ngôn ngữ như C++.

Một ứng dụng mà muốn tạo ra các ví dụ về một lớp tải động sẽ không thể để dính dáng tới cấu trúc tĩnh của nó. Thay vào đó, môi trường “run- time” tự động tạo ra một thể hiện của mỗi lớp khi nó được nạp và nó ghi nhận ví dụ với một quản lý nguyên mẫu (xem phần Cài đặt). Sau đó, các ứng dụng có thể yêu cầu quản lý nguyên mẫu cho các ví dụ của các lớp vừa được nạp, các lớp mà không được liên kết với chương trình từ đầu. Framework [WGM88] của các ứng dụng ET++ có một hệ thống “run- time” sử dụng cách này.

Nhiệm vụ chính của mô hình Prototype là mỗi lớp con của Prototype phải cài đặt một hành động Clone, điều này có thể khó khăn. Ví dụ, thêm Clone là việc khó khăn phải cân nhắc khi các lớp đã tồn tại. Thực hiện Clone có thể khó khăn khi bên trong chúng bao gồm các đối tượng không hỗ trợ sao chép hoặc có tham chiếu vòng .

1. Cài đặt

Prototype đặc biệt hữu ích với những ngôn ngữ tĩnh giống như C++, nơi mà các lớp không phải là các đối tượng và 1 chút hoặc không có loại thông tin nào có sẵn trong thời gian chạy chương trình. Nó kém quan trọng hơn trong những ngôn ngữ giống như SmallTalk hay Objective C, những ngôn ngữ mà chung quy lại là cung cấp tới prototype 1 vài thứ gì đó (ví dụ 1 đối tượng của lớp) để tạo ra những thực thể cho mỗi lớp. Kiểu này được xây dựng trở thành ngôn ngữ dựa trên prototype giống như Self [US1987] ,nơi mà mọi sự sáng tạo đối tượng diễn ra bằng việc nhân bản 1 prototype.

Xem xét những vấn đề dưới đây khi thực hiện nguyên mẫu:

1. *Sử dụng 1 trình quản lý nguyên mẫu*: Khi 1 số lượng lớn nguyên mẫu trong hệ thống không bị cố định, (điều đó nghĩa là chúng có thể được tạo ra và phá hủy 1 cách linh hoạt), cần lưu lại 1 sổ đăng kí của những nguyên mẫu có sẵn. Máy khách sẽ không tự quản lý các nguyên mẫu mà sẽ lưu trữ và khôi phục lại chúng từ sổ đăng kí. 1 khách sẽ yêu cầu 1 nguyên mẫu từ 1 sổ đăng kí trước khi nhân bản nó. Chúng ta gọi sổ đăng kí này là 1 trình quản lý nguyên mẫu.

1 trình quản lý nguyên mẫu là 1 kho liên kết trả về nguyên mẫu phù hợp với khoá nhận được. Nó có những thao tác để đăng kí 1 nguyên mẫu dưới 1 khoá và hủy đăng kí nó. Máy khách có thể thay đổi hoặc thâm chí duyệt thông qua sổ đăng kí tại thời gian chạy chương trình. Điều này cho phép khách gia hạn và lấy đi bản kiểm kê(inventory) trên hệ thống mà không cần viết code.

1. *Thực hiện thao tác nhân bản*: Phần khó nhất của kiểu nguyên mẫu là thực thi thao tác nhân bản 1 cách chính xác. Điều này đặc biệt khó khăn khi cấu trúc đối tượng chứa những tham chiếu vòng.

Phần lớn các ngôn ngữ cung cấp vài hỗ trợ cho việc sao chép đối tượng. Ví dụ, SmallTalk cung cấp 1 bản sao thực thi được kế thừa từ tất cả các lớp con của lớp đối tượng. C++ cung cấp 1 hàm khởi tạo sao chép (copy constructor). Nhưng những cơ sở này không giải quyết được vấn đề “shallow copy và deep copy” [GR83]. Vấn đề đó là, việc nhân bản 1 đối tượng sẽ lần lượt nhân bản các biến thực thể của nó, hay đối tượng nhân bản và đối tượng gốc cùng chia sẻ các biến?

Shallow copy rất đơn giản và thường đầy đủ, và đó chính xác là những gì mà Smalltalk cung cấp mặc định. 1 hàm khởi tạo sao chép (copy constructor) mặc định trong C++ thực hiện 1 bản sao member-wise, nghĩa là con trỏ sẽ đươc chia sẻ cho bản sao và cả bản gốc. Nhưng việc nhân bản nguyên mẫu cùng với cấu trúc phức tạp lại yêu cầu 1 deep copy, bởi vì bản nhân bản và bản gốc là tách biệt. Vì vậy cần phải đảm bảo rằng những thành phần của bản sao phải là nhân bản của những thành phần nguyên mẫu. Nhân bản buộc bạn phải xác định được điều gì sẽ xảy ra nếu thứ nào đó được chia sẻ.

Nếu những đối tượng trong hệ thống cung cấp những thao tác Lưu và tải (save and load), thì bạn có thể sử dụng chúng để cung cấp 1 cài đặt mặc định của Clone 1 cách đơn giản bằng cách lưu đối tượng và tải chúng về ngay lập tức. Thao tác lưu 1 đối tượng vào vùng đệm ô nhớ , và Load tạo ra bản sao bằng việc xây dựng lại 1 đối tượng từ bộ đệm.

1. *Khởi tạo nhân bản*: Trong khi một số máy khách hài lòng với việc nhân bản, thì 1 số khác vẫn muốn khởi tạo 1 vài hoặc tất cả những trạng thái bên trong giá trị mà họ chọn. Nhìn chung bạn không thể truyền những giá trị này trong thao tác Clone, bởi vì số của chúng sẽ thay đổi giữa các lớp nguyên mẫu. 1 vài nguyên mẫu cần nhiều tham số khởi tạo, 1 số khác thì không. Truyền tham số trong thao tác Clone ngăn cản 1 giao diện nhân bản thống nhất.

Cũng có thể trong trường hợp các lớp nguyên mẫu của bạn hoàn toàn đã xác định những thao tác để cấu tạo (tái tạo) key cho những trạng thái. Nếu như vậy, khách có thể sử dụng những thao tác này ngay lập tức sau khi nhân bản. Nếu không, bạn có thể giới thiệu 1 thao tác khởi tạo (xem phần Sample code) cái mà coi những biến khởi tạo giống như đối số và thiết lập trạng thái nhân bản nội bộ cho phù hợp. Cẩn thận vớ thao tác Clone deep copy, những bản sao có thể cần được xóa (hoặc là bị xóa rõ ràng, hoặc trong hàm Initialize) trước khi khởi tạo lại chúng.

1. Ví dụ mẫu

Ta sẽ định nghĩa 1 lớp con MazePrototypeFactory của lớp MazeFactory. Lớp MazePrototypeFactory sẽ được khởi tạo với các bản mẫu của các đối tượng mà nó sẽ tạo ra cốt để chúng ta không phải tạo ra lớp con của nó chỉ để thay đổi các lớp của tường hoặc phòng mà nó tạo ra.

MazePrototypeFactory mở rộng giao diện MazeFactory với hàm khởi tạo dùng các nguyên mẫu làm đối số:

class MazePrototypeFactory : public MazeFactory {

public:

MazePrototypeFactory(Maze\*, Wall\*, Room\*, Door\*);

virtual Maze\* MakeMaze() const;

virtual Room\* MakeRoom(int) const;

virtual Wall\* MakeWall() const;

virtual Door\* MakeDoor(Room\*, Room\*) const;

private:

Maze\* \_prototypeMaze;

Room\* \_prototypeRoom;

Wall\* \_prototypeWall;

Door\* \_prototypeDoor;

};

Hàm khởi tạo mới đơn giản chỉ khởi tạo các nguyên mẫu của nó:

MazePrototypeFactory::MazePrototypeFactory (

Maze\* m, Wall\* w, Room\* r, Door\* d

) {

\_prototypeMaze = m;

\_prototypeWall = w;

\_prototypeRoom = r;

\_prototypeDoor = d;

}

Các hàm thành phần dùng để khởi tạo các bức tường, các căn phòng và các cánh cửa thì giống nhau: Mỗi hàm sẽ sao chép 1 kiểu mẫu và sau đó sẽ khởi tạo nó. Sau đây là định nghĩa của hàm MakeWall và MakeDoor:

Wall\* MazePrototypeFactory::MakeWall () const {

return \_prototypeWall->Clone();

}

Door\* MazePrototypeFactory::MakeDoor (Room\* r1, Room \*r2) const {

Door\* door = \_prototypeDoor->Clone();

door->Initialize(r1, r2);

return door;

}

Chúng ta có thể sử dụng lớp MazePrototypeFactory để tạo ra 1 mê cung theo nguyên mẫu hoặc theo mặc định chỉ bằng việc khởi tạo nó với các nguyên mẫu của các thành phần đơn giản của mê cung:

MazeGame game;

MazePrototypeFactory simpleMazeFactory(

new Maze, new Wall, new Room, new Door

);

Maze\* maze = game.CreateMaze(simpleMazeFactory);

Để thay đổi loại mê cung, chúng ta khởi tạo MazePrototype với 1 tập các nguyên mẫu khác. Lời gọi sau sẽ tạo ra 1 mê cung với 1 cánh cửa bị đánh bom (BombedDoor) và 1 căn phòng có 1 quả bom (RoomWithABomb):

MazePrototypeFactory bombedMazeFactory(

new Maze, new BombedWall,

new RoomWithABomb, new Door

);

Một đối tượng mà có thể được dùng như 1 nguyên mẫu, ví dụ như 1 thực thể của Wall, cần hỗ trợ thao tác Clone. Nó cũng phải có 1 hàm khởi tạo sao chép để dùng cho việc sao chép. Nó cũng có thể cần 1 hàm để tái khởi tạo trạng thái nội bộ. Chúng ta sẽ thêm hàm Initialize vào lớp Door để các client khởi tạo các căn phòng đã sao chép.

So sánh định nghĩa sau của lớp Door với định nghĩa của lớp Door ở trang 96:

class Door : public MapSite {

public:

Door();

Door(const Door&) ;

virtual void Initialize(Room\*, Room\*);

virtual Door\* Clone() const;

virtual void Enter();

Room\* OtherSideFrom(Room\*);

private:

Room\* \_room1;

Room\* \_room2;

};

Door::Door (const Door& other) {

\_room1 = other.\_room1;

\_room2 = other.\_room2;

}

void Door::Initialize (Room\* r1, Room\* r2) {

\_room1 = r1;

\_room2 = r2;

}

Door\* Door::Clone () const {

return new Door(\*this);

}

Lớp con BombedWall phải viết chồng hàm Clone và thực thi 1 hàm khởi tạo sao chép phản hồi.

class BombedWall : public Wall {

public:

BombedWall();

BombedWall(const BombedWall&);

virtual Wall\* Clone() const;

bool HasBomb();

private:

bool \_bomb;

};

BombedWall::BombedWall (const BombedWall& other) : Wall(other) {

\_bomb = other.\_bomb;

}

Wall\* BombedWall::Clone () const {

return new BombedWall(\*this);

}

Mặc dù hàm BombedWall::Clone() trả về 1 con trỏ Wall\*, các thực thi của nó lại trả về một con trỏ đến 1 thể hiện của một lớp con, đó là BombedWall\*. Chúng ta định nghĩa hàm Clone như thế này trong lớp cơ sở để đảm bảo rằng các clients mà sao chép nguyên mẫu không cần phải biết về các lớp con. Người dùng không nên ép kiểu của giá trị trả về của Clone về kiểu mong muốn.

Trong Smalltalk, bạn có thể tái sử dụng phương thức sao chép chuẩn được kế thừa từ Object để sao chép bất kỳ MapSite. Bạn có thể dùng lớp MazeFactory để tạo ra nguyên mẫu bạn cần. ví dụ bạn có thể tạo ra một căn phòng bằng cách cung cấp tên #room. Lớp MazeFactory có bộ từ điển danh sách đến các nguyên mẫu. Cách tạo phương thức make: method của nó sẽ như sau:

make: partName

^ (partCatalog at: partName) copy

Các phương thức phù hợp đã được cho trước để khởi tạo lớp MazeFactory với các nguyên mẫu, bạn có thể tạo 1 mê cung đơn giản với đoạn code sau:

CreateMaze

on: (MazeFactory new

with: Door new named: #door;

with: Wall new named: #wall;

with: Room new named: #room;

yourself)

mà định nghĩa của phương thức on: class cho hàm CreateMaze sẽ là:

on: aFactory

| room1 room2 |

room1 := (aFactory make: #room) location: 1@1.

room2 := (aFactory make: #room) location: 2@1.

door := (aFactory make: #door) from: room1 to: room2.

room1

atSide: #north put: (aFactory make: #wall);

atSide: #east put: door;

atSide: #south put: (aFactory make: #wall);

atSide: #west put: (aFactory make: #wall).

room2

atSide: #north put: (aFactory make: #wall);

atSide: #east put: (aFactory make: #wall);

atSide: #south put: (aFactory make: #wall);

atSide: #west put: door.

^ Maze new

addRoom: room1;

addRoom: room2;

yourself

1. Các ứng dụng đã biết

Có thể ví dụ đầu tiên của kiểu nguyên mẫu ở trong hệ thống Ivan Sutherland's Sketchpad [Sut63]. Ứng dụng đầu tiên của kiểu này được biết đến rộng rãi trong ngôn ngữ hướng đối tượng ThingLab, nơi mà người dùng có thể tạo ra 1 đối tượng tổng hợp và sau đó thúc đẩy nó vào 1 nguyên mẫu bằng việc cài đặt nó trong thư viện của những đối tượng có thể tái sử dụng. [Bor81]. Golberg và Robsson đề cập những nguyên mẫu như 1 kiểu [GR83], nhưng Coplien [Cop92] đưa ra nhiều mô tả hoàn chỉnh hơn cả. Anh ấy mô tả những thành ngữ này liên quan tới kiểu nguyên mẫu dành cho C++ và đưa ra rất nhiều ví dụ và các biến thể.

Etgdb là 1 trình gỡ lỗi front-end dựa trên ET++ cung cấp 1 giao diện chỉ và click (point and click) đến các trình gỡ lỗi đường hướng khác nhau. Mỗi trình gỡ lỗi có 2 lớp con DebuggerAdaptor tương ứng (corresponding). Ví dụ,GdbAdaptor làm cho etgdb phù hợp với cú pháp lênh của gdb GNU, trong khi SuDbxAdaptor làm cho etgdb phù hợp với trình gỡ lỗi Sun’s dbx/ Etgdb không chứa 1 tập các lớp DebuggerAdaptor được mã hóa cứng. Thay vào đó, nó đọc tên của bộ tùy chỉnh/bộ chuyển đổi(adaptor) để sử dụng từ 1 biến số môi trường, tìm kiếm 1 nguyên mẫu với những cái tên được chỉ định trong bảng toàn cầu, và sau đó nhân bản nguyên mẫu. Những trình gỡ lỗi mới có thể được thêm vào etgdb bằng việc kết nối nó với DebuggerAdaptor làm việc cho trình gỡ lỗi đó.

“Thư viện chuyên môn tương tác” trong chế độ Composer lưu trữ những nguyên mẫu của các đối tượng, cái mà hỗ trợ phần lớn kĩ thuật tương tác [Sha90]. Bất kì 1 kĩ thuật tương tác nào được tạo ra bởi chế độ Composer cũng được sử dụng như 1 nguyên mẫu bằng cách đặt chúng trong 1 thư viện. Kiểu nguyên mẫu cho phép chế độ Composer hỗ trợ 1 tập kĩ thuật tương tác không giới hạn.

Ví dụ trình chỉnh sửa âm nhạc được thảo luận trước đó dựa trên khuôn vẽ Unidraw [VL90]

1. Những kiểu mẫu thiết kế có liên quan

Nguyên mẫu và Abstract Factory (99) đang cạnh tranh nhau về 1 vài kiểu ở 1 vài cách, như chúng ta sẽ thảo luận vào cuối chương này. Tuy nhiên chúng vẫn có thể được sử dụng cùng nhau. 1 Abstract Factory có thể lưu trữ 1 tập các nguyên mẫu để nhân bản và trả lại những đối tượng sản phẩm.

Thiết kế ngoài việc sử dụng nhiều Composite (183) thì chúng còn thường xuyên hưởng lợi ích từ nguyên mẫu.

1. TÀI LIỆU THAM KHẢO

Trong quá trình thực hiện đồ án này thì chúng em có tham khảo tài liệu thông qua một số nguồn khác như:

* + - 1. Design Patterns : Elements of Reusable Object-Oriented Software
* Các tác giả: Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
  + - 1. Các trang hỗ trợ, các trang chia sẻ kiến thức trên Internet.

1. LỜI KẾT

Trên đây là toàn bộ nội dung đề tài chương trình quản lý điểm thi sinh viên được viết bằng ngôn ngữ C++. Trong thời gian tìm hiểu và thực hiện đề tài, chúng em đã cố gắng hoàn thành nghiêm túc công việc. Tuy nhiên, do khả năng, tầm hiểu biết của bản thân còn hạn chế và thời gian có hạn nên đề tài không thể tránh khỏi những thiếu sót, chưa thực sự đáp ứng đầy đủ yêu cầu của công việc. Chúng em rất mong được sự quan tâm, đóng góp ý kiến của các thầy cô giáo và các bạn để chương trình này có thể đạt hiệu quả cao hơn và chúng em có thể rút ra được nhiều kinh nghiệm quý báu để làm các đồ án sau.

Chúng em xin chân thành cảm ơn cô Thái Thị Nguyệt đã nhiệt tình giúp đỡ chúng em trong quá trình hoàn thiện đề tài này.

Các thành viên trong nhóm:

* Lương Văn Sáng
* Mai Thị Lan Anh
* Phạm Thị Thu Thuỳ
* Nguyễn Thị Tươi