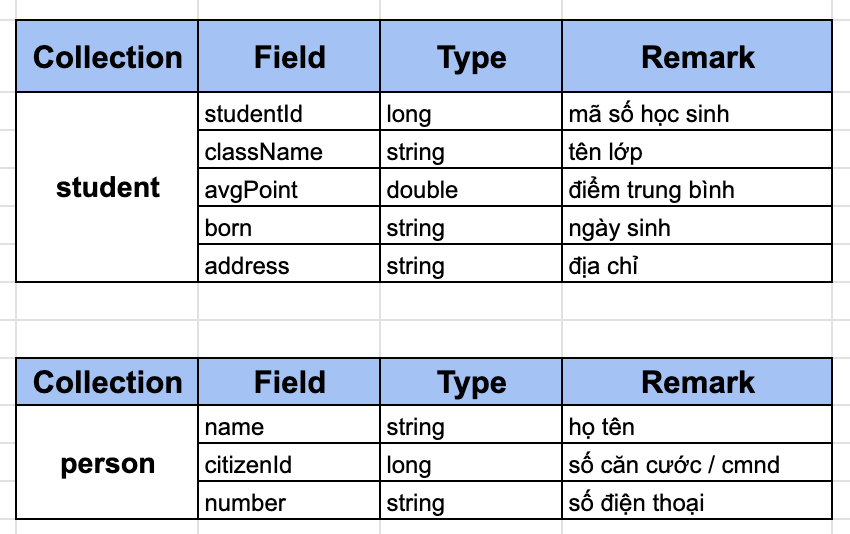
|  |
| --- |
| **TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**  **KHOA KỸ THUẬT - ĐIỆN TỬ VIỄN THÔNG**  A picture containing logo  Description automatically generated**•◊Δ◊•**  **BÁO CÁO ĐỒ ÁN OOP**  **TÊN ĐỀ TÀI:**  **Xây dựng chương trình quản lý học sinh bằng lập trình hướng đối tượng**  **GVHD: Lê Đức Trị**  **SVTH : Nguyễn Anh Tuấn – 20200400**  **Nguyễn Gia Phụng – 20200313**  **Nguyễn Hữu Luật – 20200256**  **LỚP : 20DTV2**  **Thành phố Hồ Chí Minh, ngày 15 tháng 3 năm 2023** |

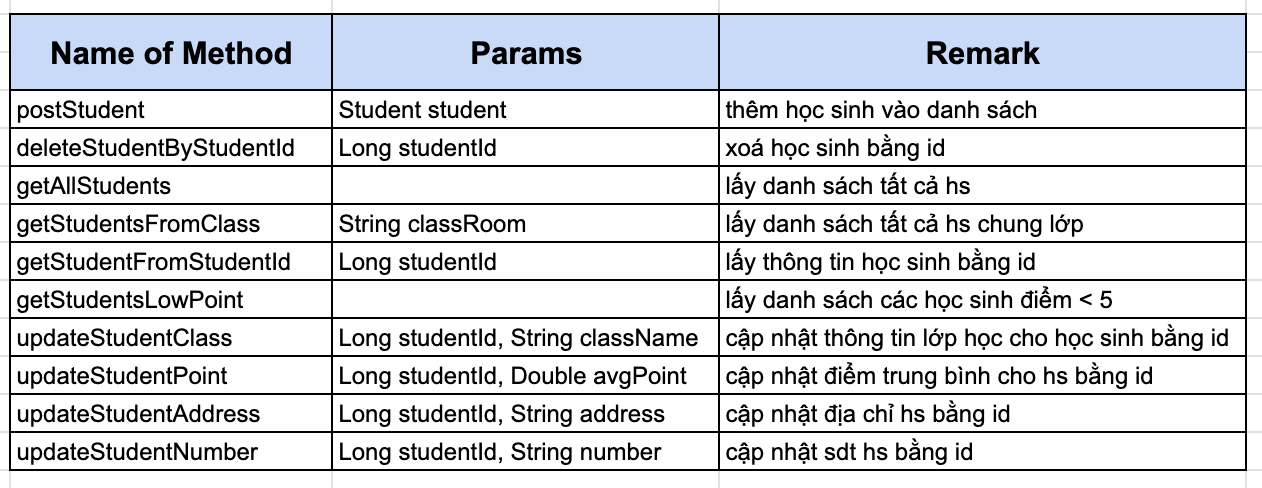
\*\*Ngôn ngữ sử dụng: Java, Javascript

1. **Phân tích chức năng:**

Song song với việc phát triển mô hình quản lí sinh viên, không chỉ quản lí sinh viên nói riêng mà còn quản lí con người nói chung trong cả trường học với mục đích mở rộng hơn ứng dụng thì chúng ta có class Person với các thuộc tính name, citizenId, number. Các thuộc tính của đối tượng Student cần thiết như ngày sinh, điểm trung bình, địa chỉ, id học sinh, từ đó khởi tạo bảng thuộc tính với đối tượng một cách trực quan:

Lúc này, đối tượng Student sẽ kế thừa các thuộc tính của Person để nhận về các thuộc tính của Person và các hàm, cầu nối thông qua khoá super(name,citizenId,number) khi khai báo phương thức khởi tạo Student.

Sau đó các actions được triển khai để quản lí danh sách sinh viên như CRUD ( create, read, update, delete a.k.a thêm bớt xoá sửa ):



Cùng với ngôn ngữ Java, SpringBoot phát triển và hỗ trợ Java một cách mạnh mẽ và để phát triển báo cáo một cách trực quan nhất có thể, việc xây dựng một server-side với Java SpringBoot và một client-side với JavaScript là rất hữu ích và thôi thúc tinh thần tìm hiểu. Bước đầu tiếp cận với mô hình design pattern MVC là ổn định và cơ bản để nắm bắt. Cấu trúc dự án:

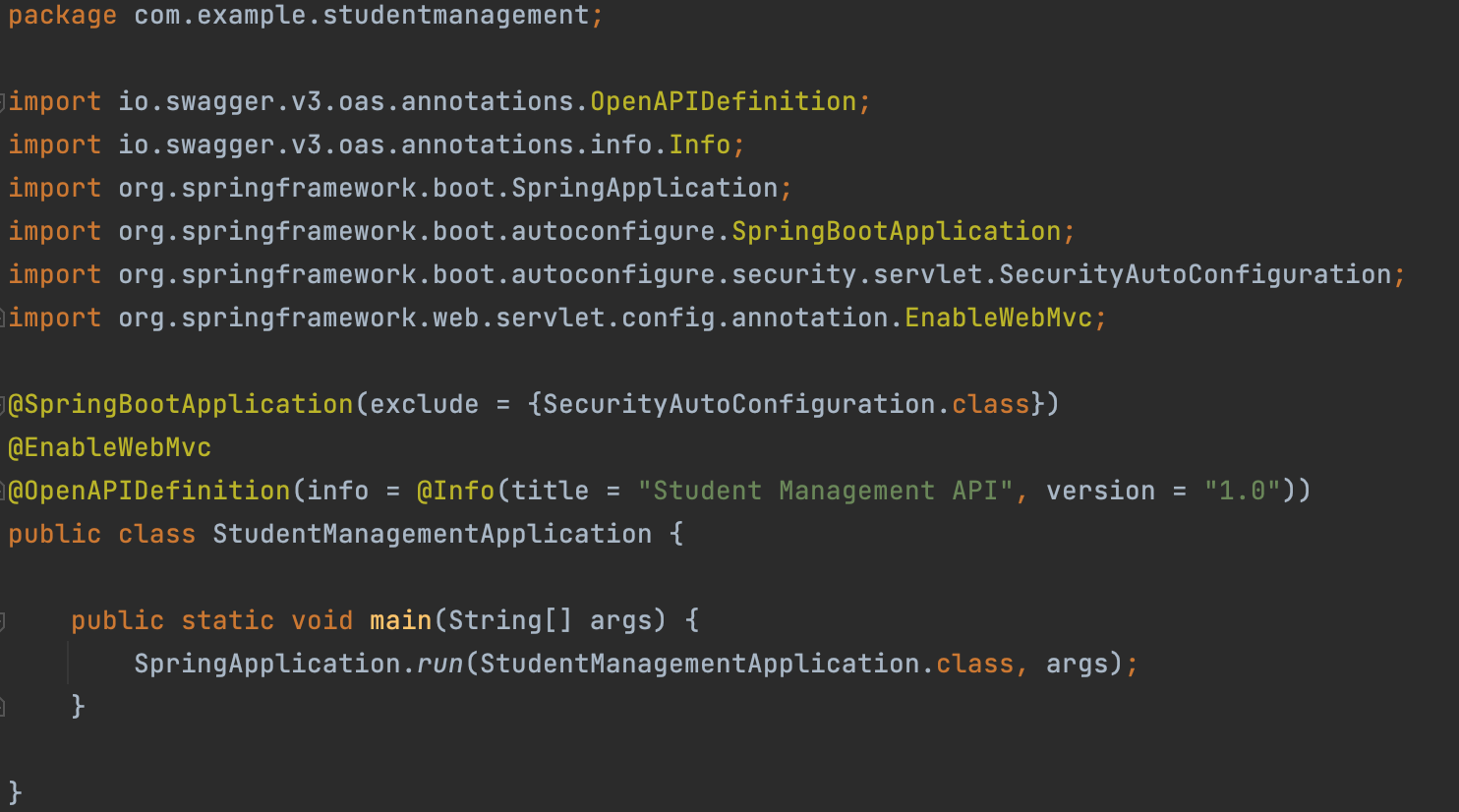


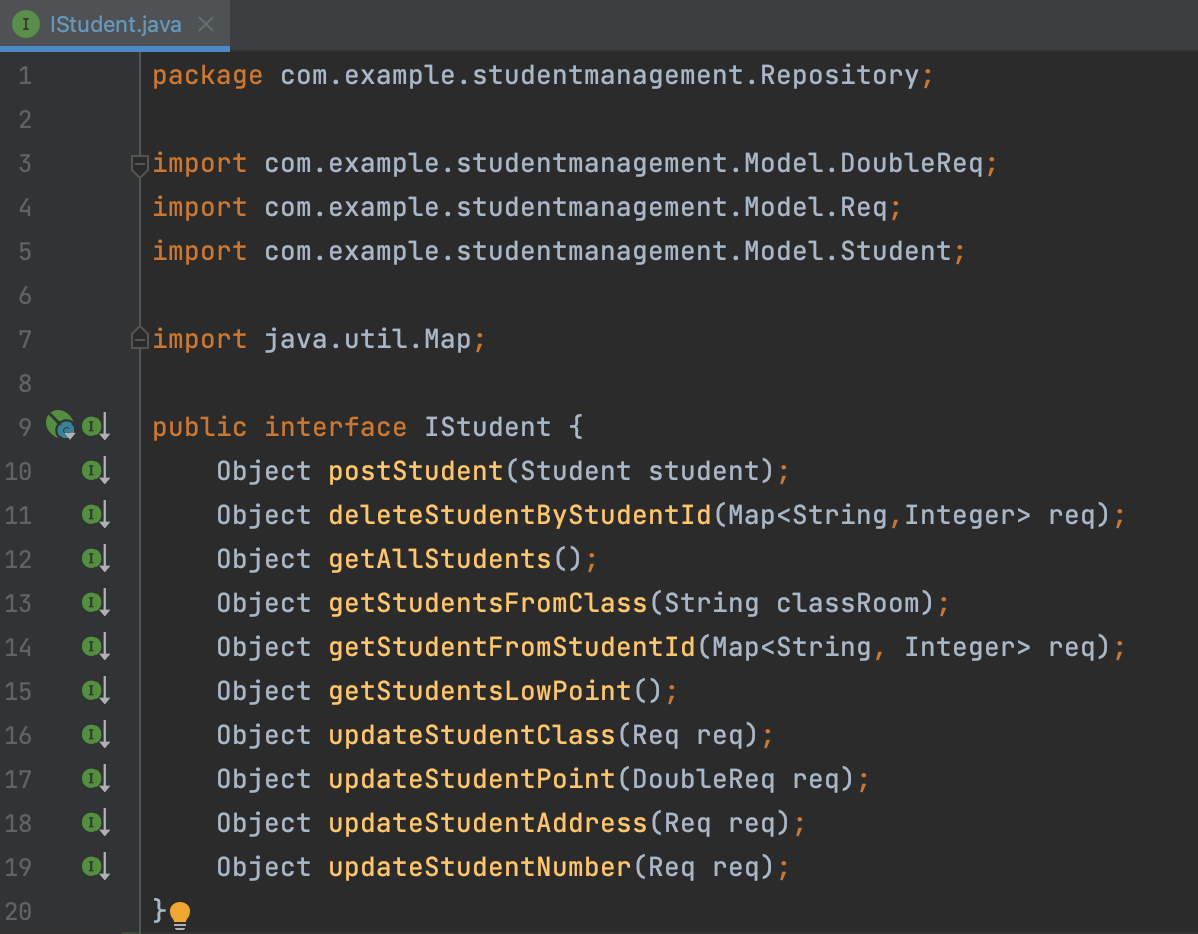
Trong đó:

* Package method là nơi định nghĩa các phương thức với endpoint API ví dụ như *abc.xyz/***post-student** thì post-student là endpoint API đại diện cho một phần tài nguyên trên máy chủ, cụ thể là server-side này thì với endpoint đó được gọi sẽ xử lí phương thức tương ứng được định nghĩa.
* Model là nơi chứa các class Object như Student, Person và các Response trả về client-side để xử lí dữ liệu.
* Repository là nơi chứa các triển khai của các hàm, phương thức để nối tới package method như trên, trong này sẽ là nơi xử lí dữ liệu chính. Class Istudent là class interface để triển khai cho StudentRepository và cả StudentService tránh tình trạng nhầm lẫn các method.
* Utils nơi chứa các xử lí logic kiểm tra vặt.
* StudentManagementApplication là một lớp chứa phương thức main() để khởi chạy ứng dụng này theo trình tự :
  + Tìm kiếm tất cả các phụ thuộc của ứng dụng và xác định các cấu hình mặc định.
  + Tự động cấu hình ứng dụng dựa trên các cấu hình mặc định và các cấu hình được xác định bởi người dùng trong các tệp cấu hình (config files).
  + Tạo ra các Bean của Spring phù hợp để xử lý các yêu cầu từ người dùng, bao gồm các Bean để xử lý các yêu cầu HTTP, Bean để xử lý các yêu cầu cơ sở dữ liệu, và các Bean khác để xử lý các tác vụ khác trong ứng dụng.
  + Kết nối các Bean lại với nhau để tạo ra một ứng dụng hoàn chỉnh.
  + Chạy ứng dụng và lắng nghe các yêu cầu từ người dùng để xử lý chúng.

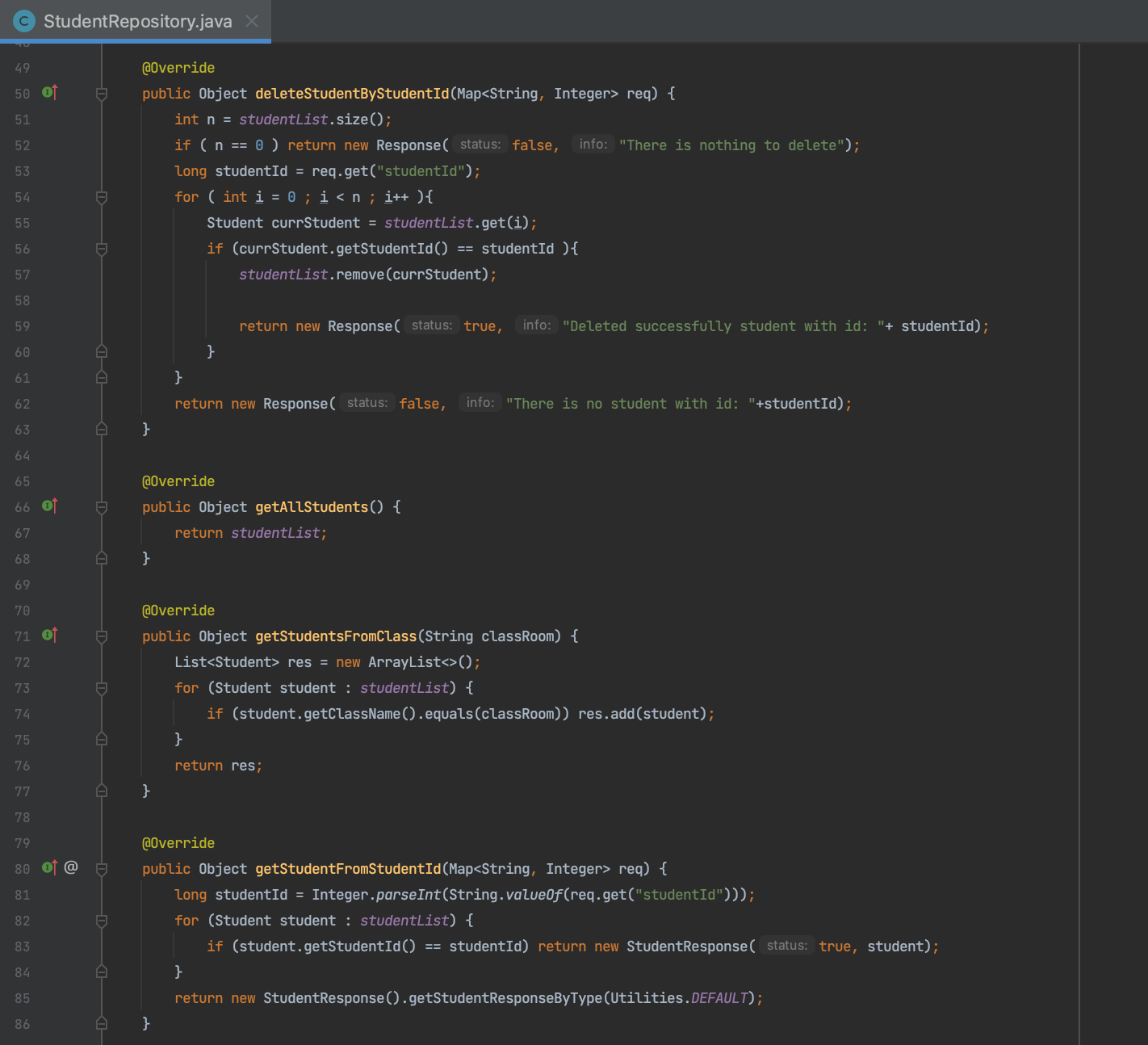
1. **Triển khai lập trình:**

File StudentManagementApplication.java:

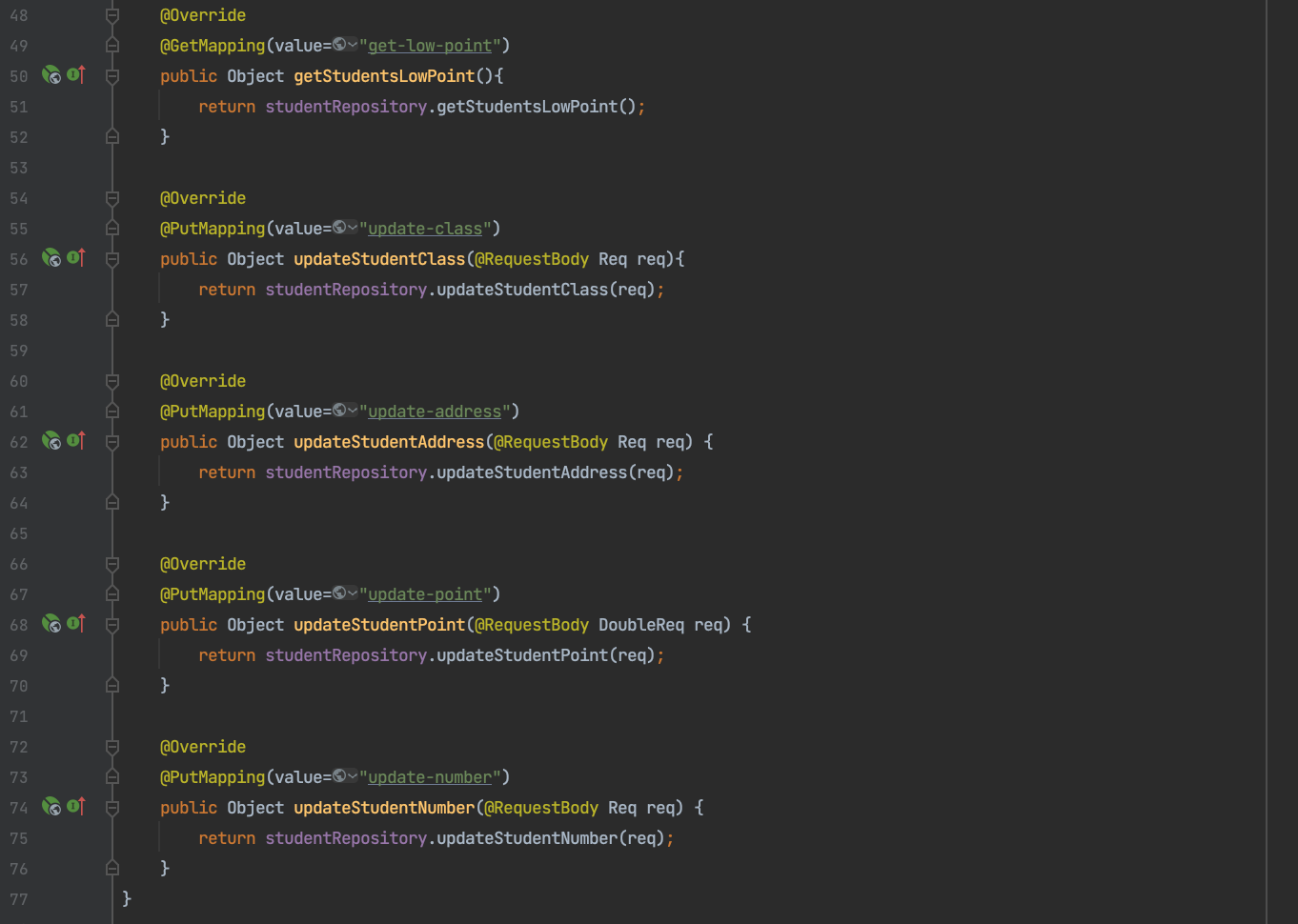
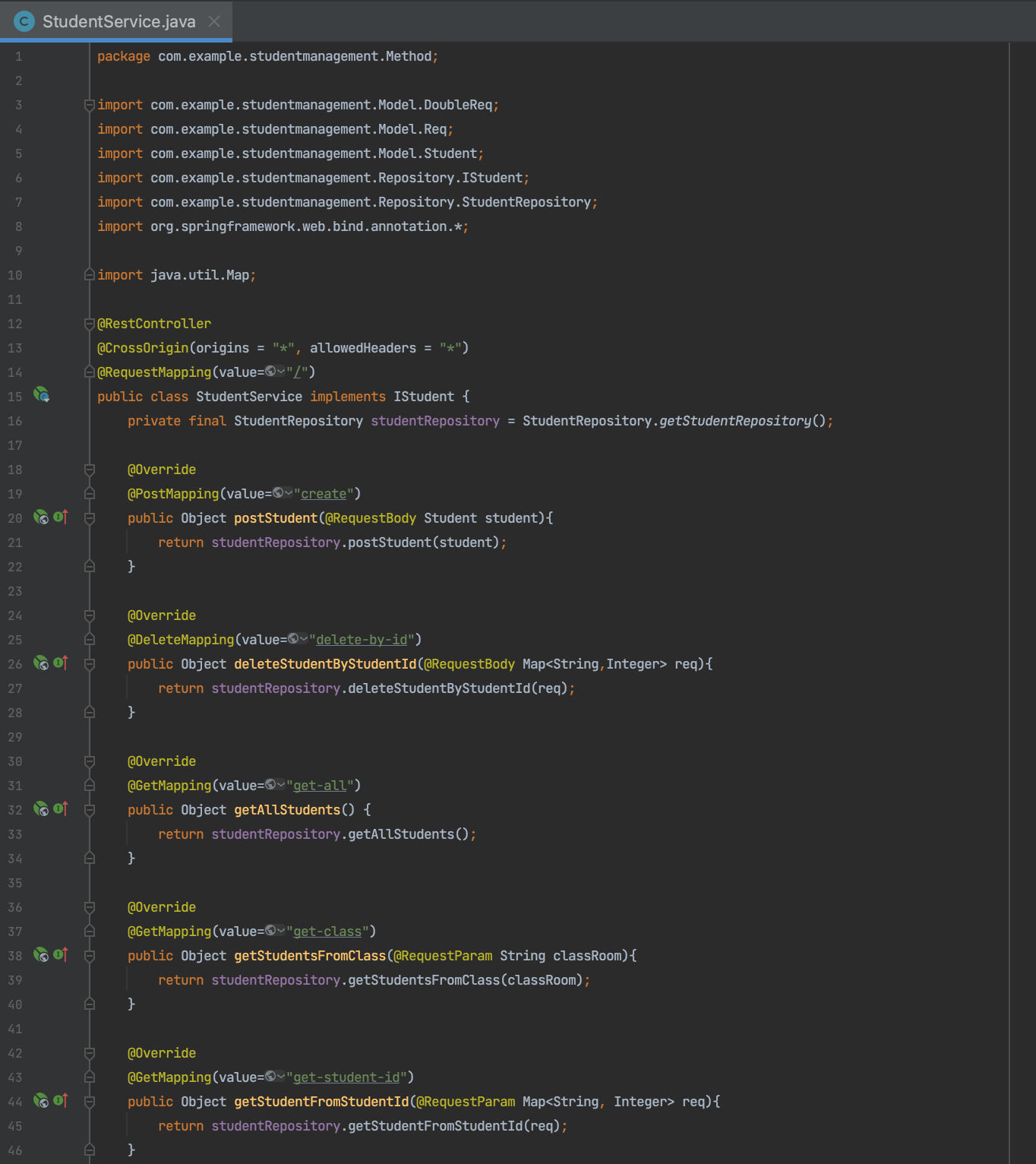


File Istudent.java: 

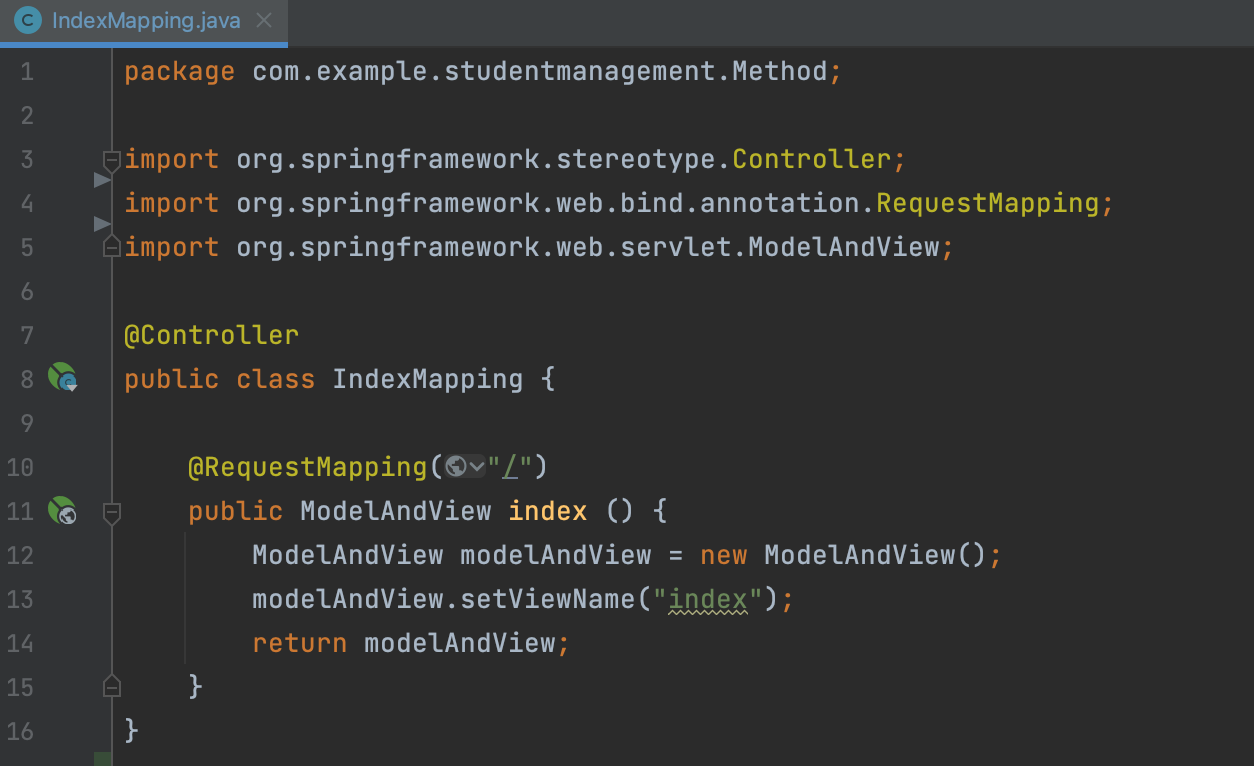
File StudentRepository.java: triển khai từ Istudent



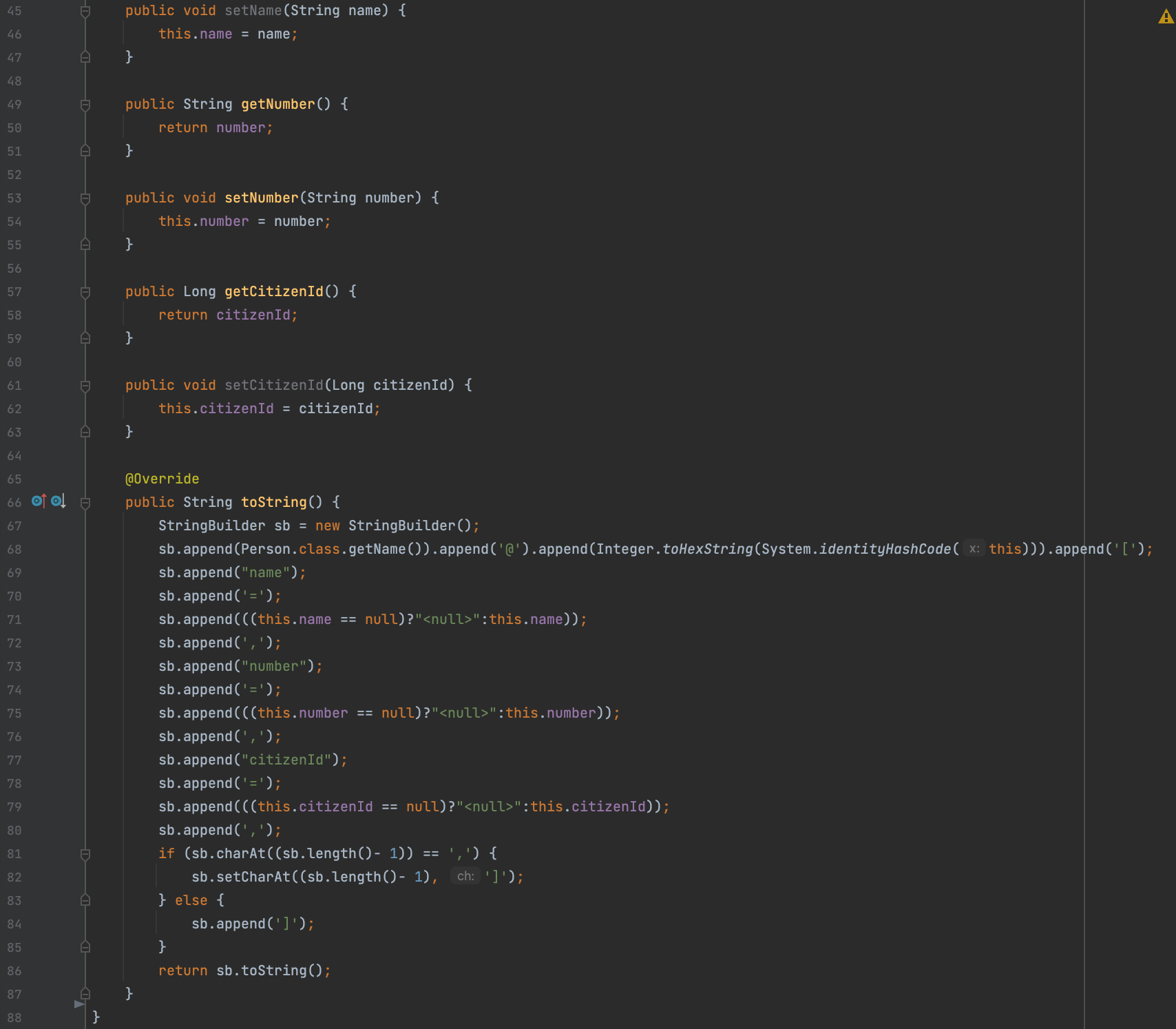
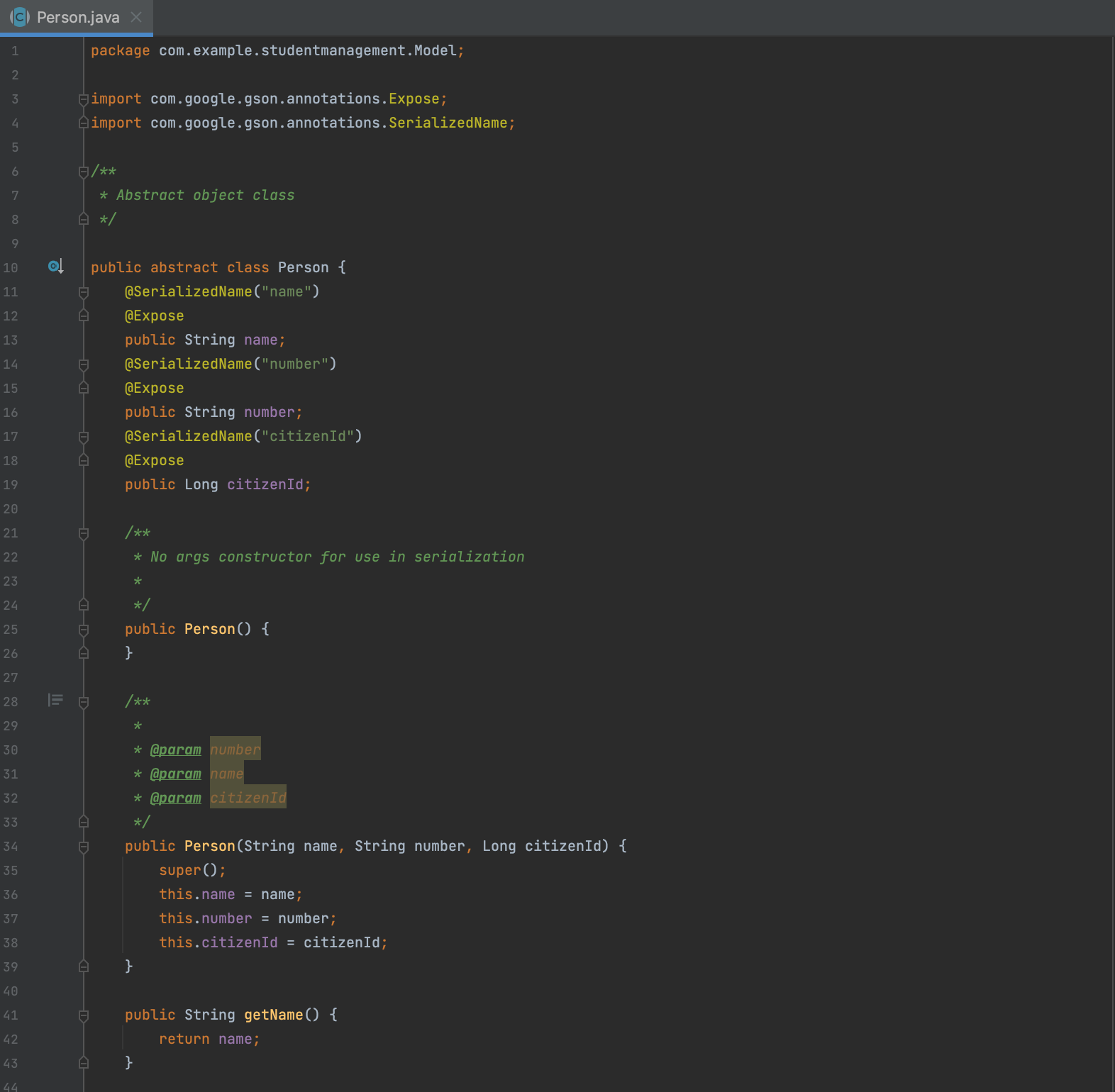
File StudentService.java triển khai từ IStudent:



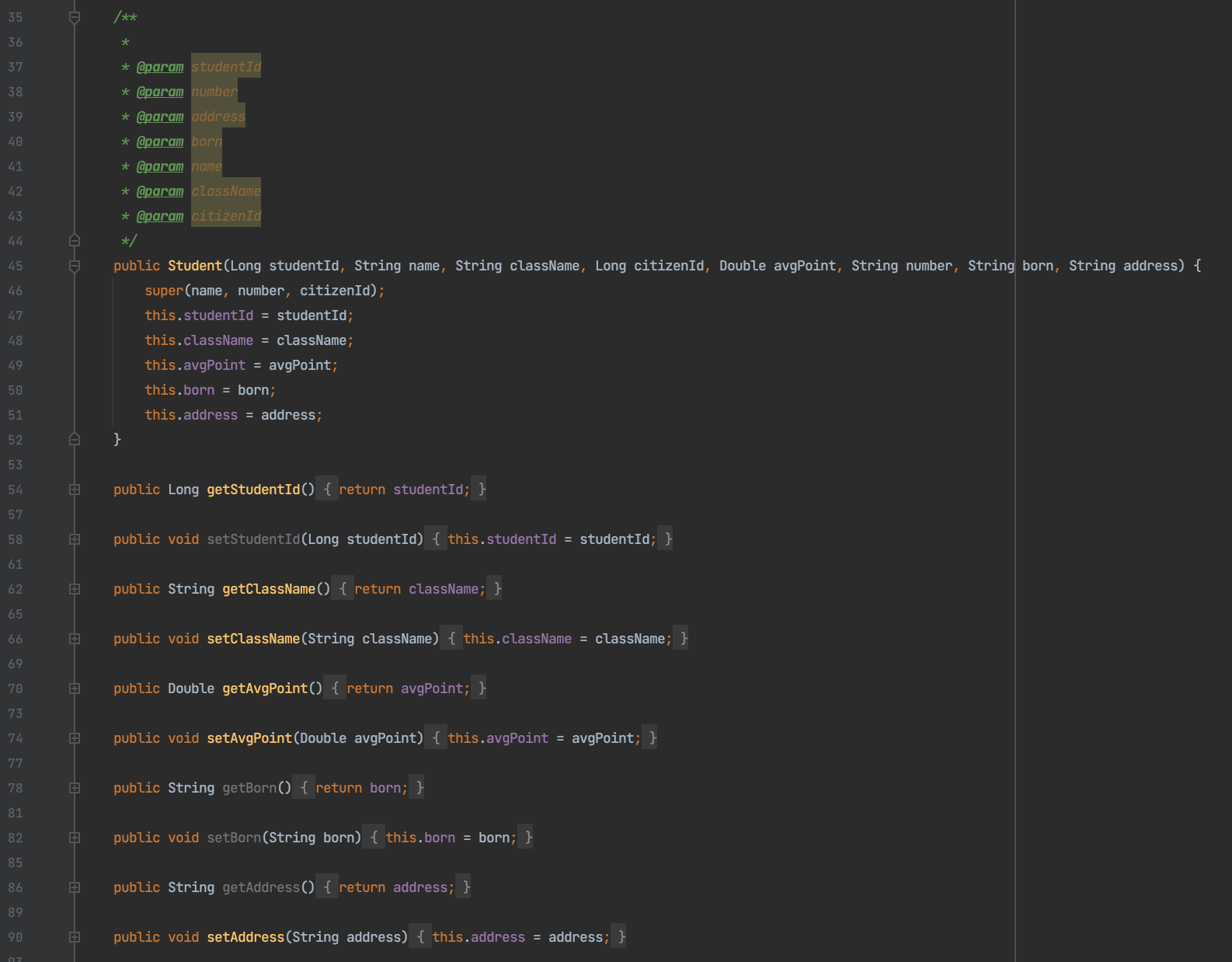
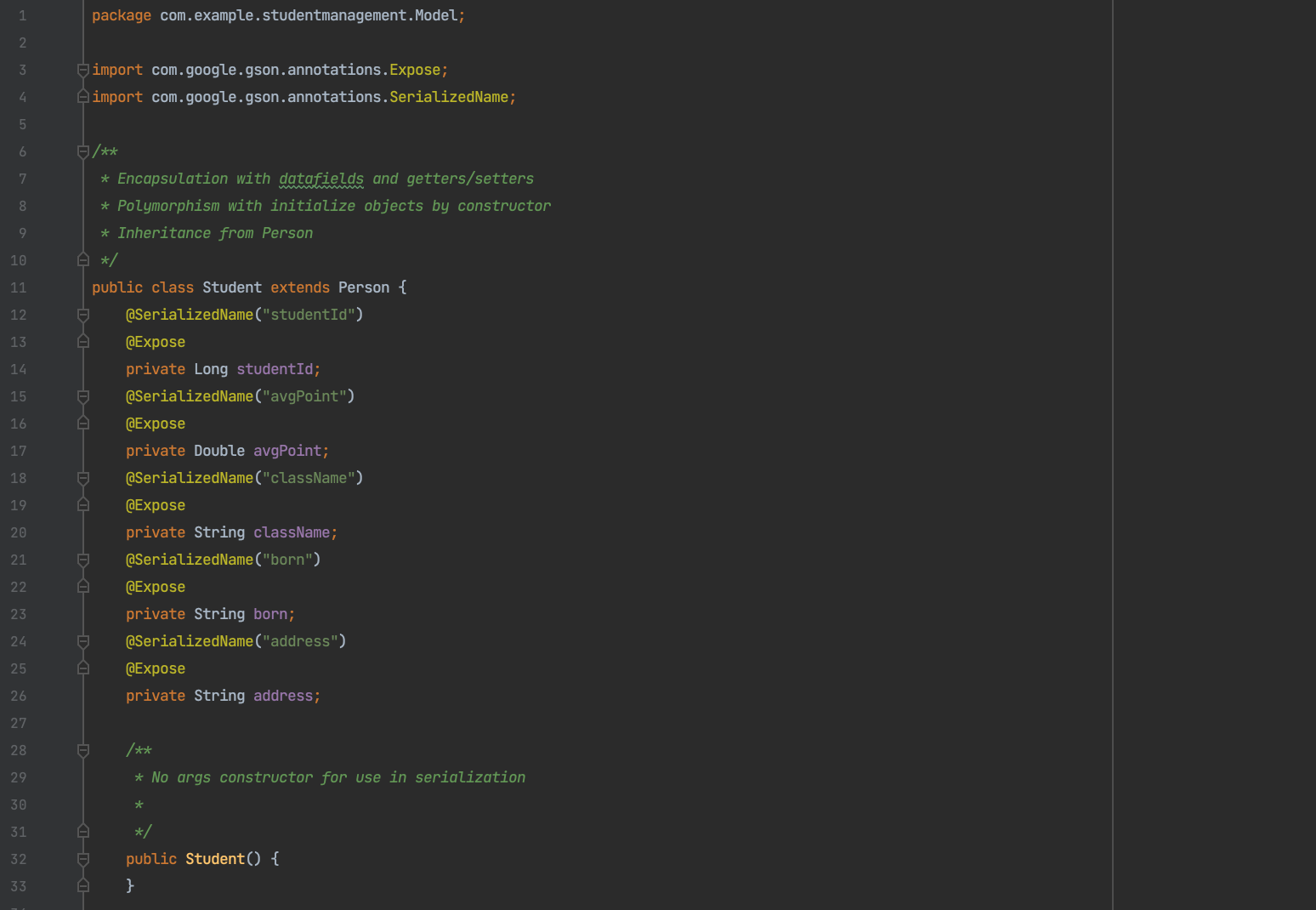
File IndexMapping.java:

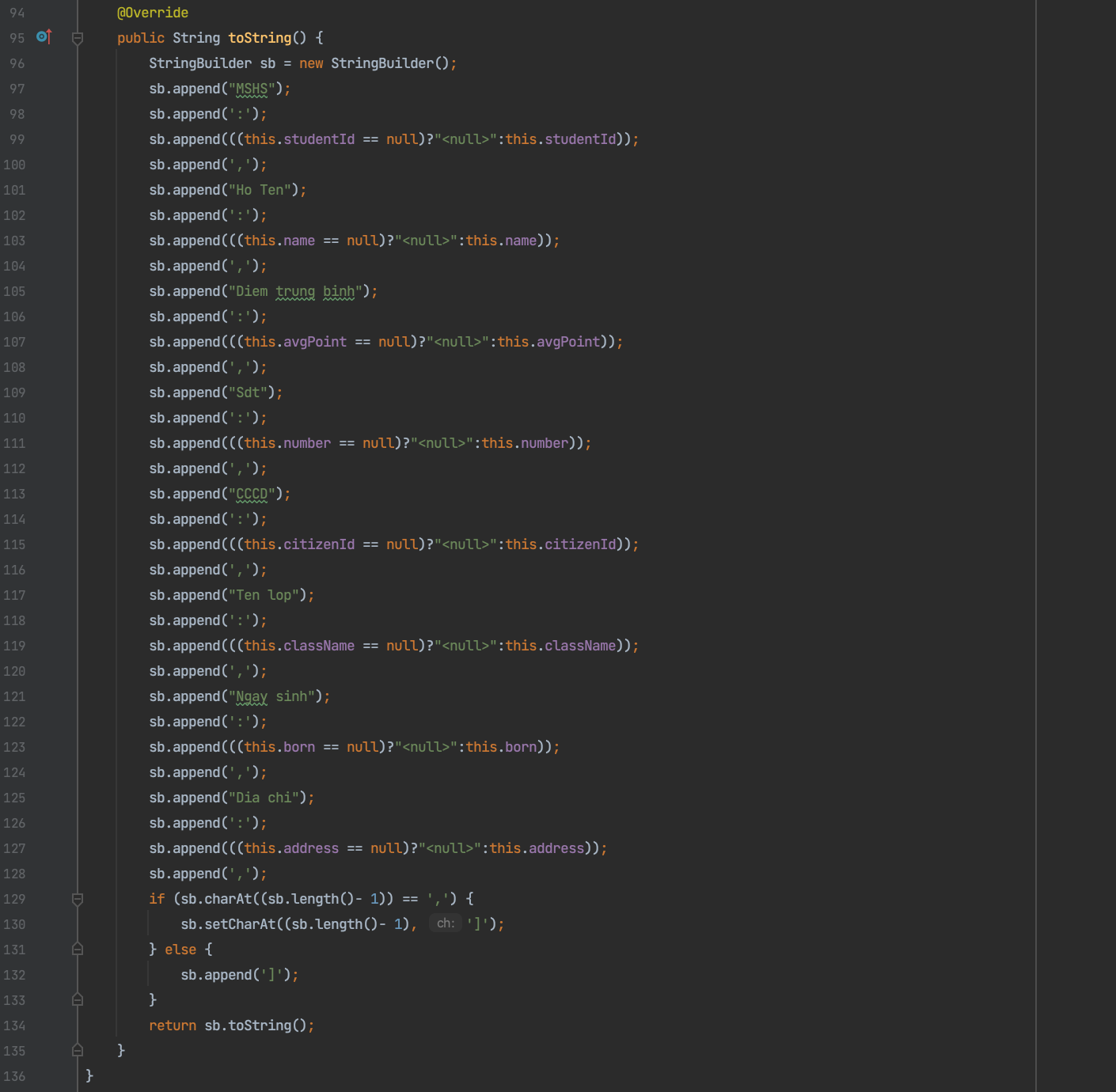
****

File Person.java:



File Student.java kế thừa lớp cha Person:

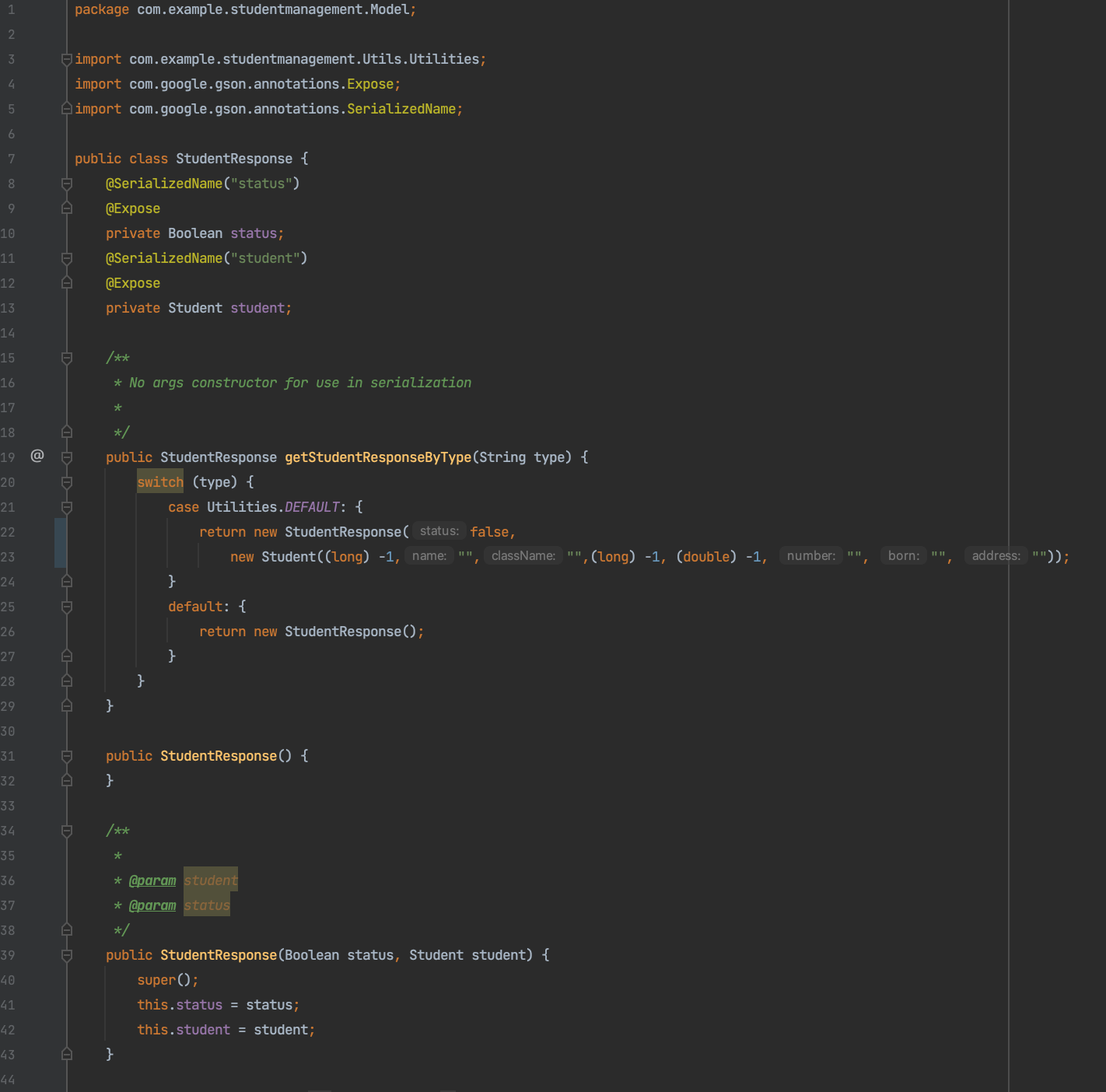


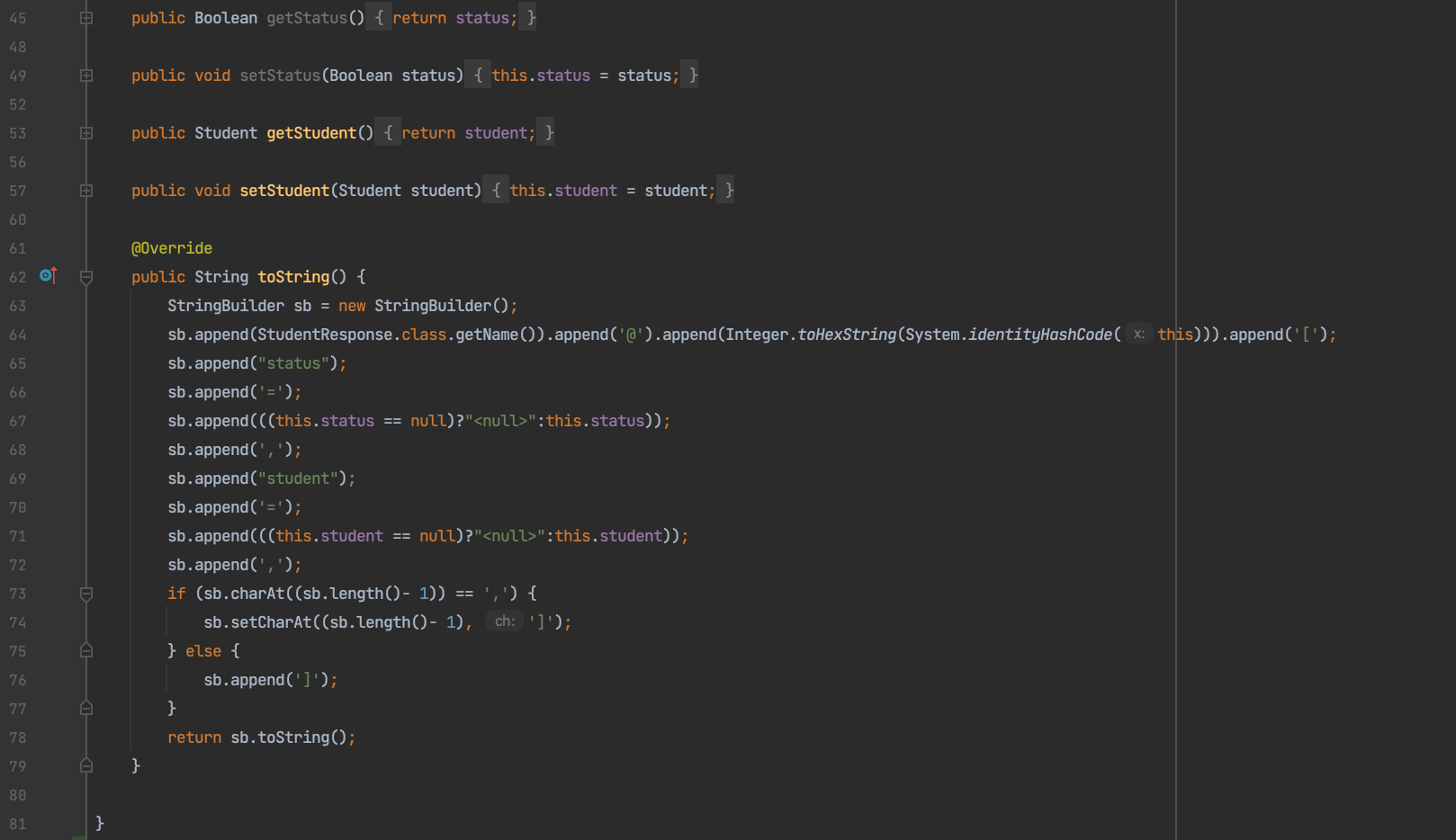


File Response.java

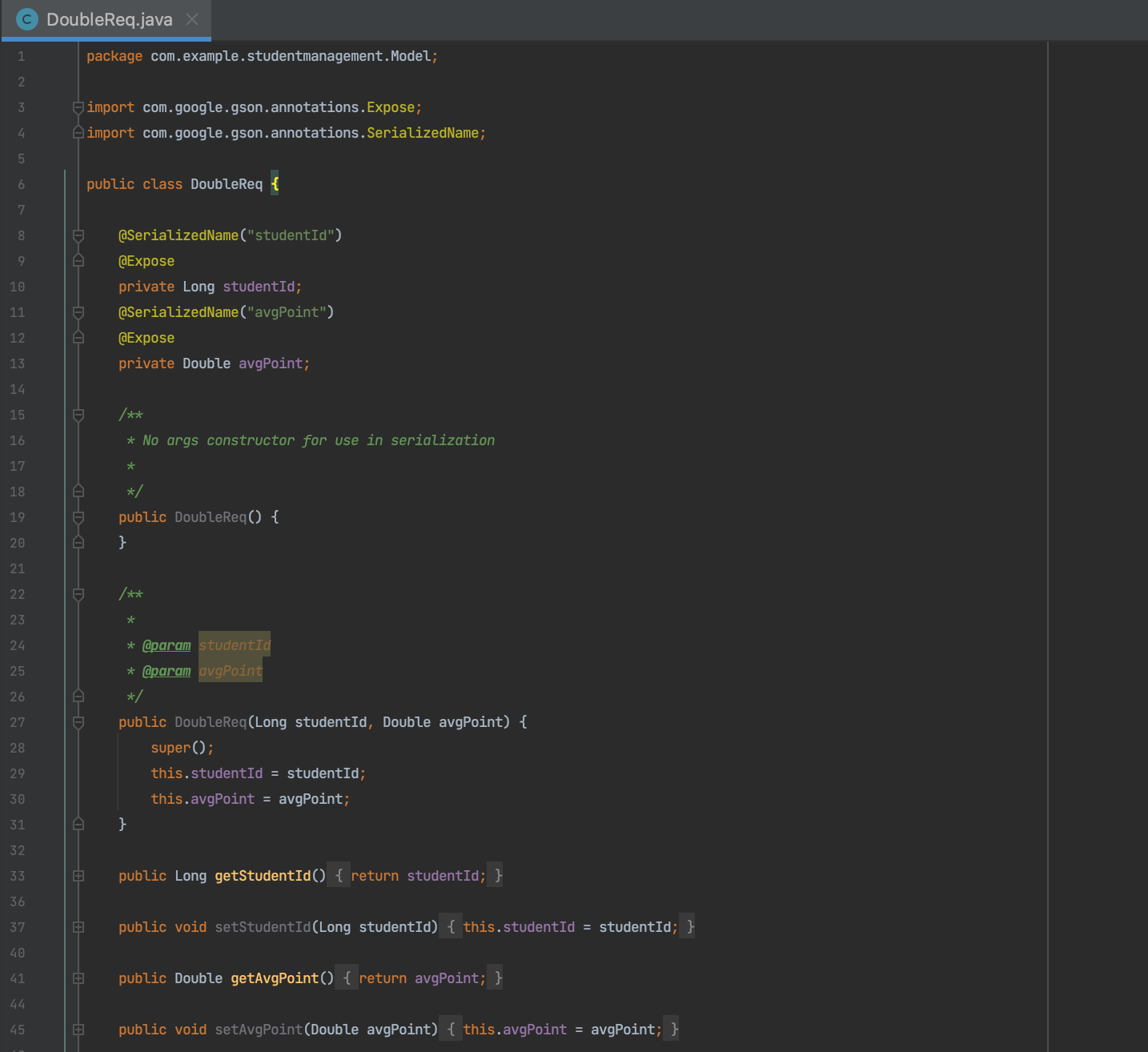


File StudentResponse.java:

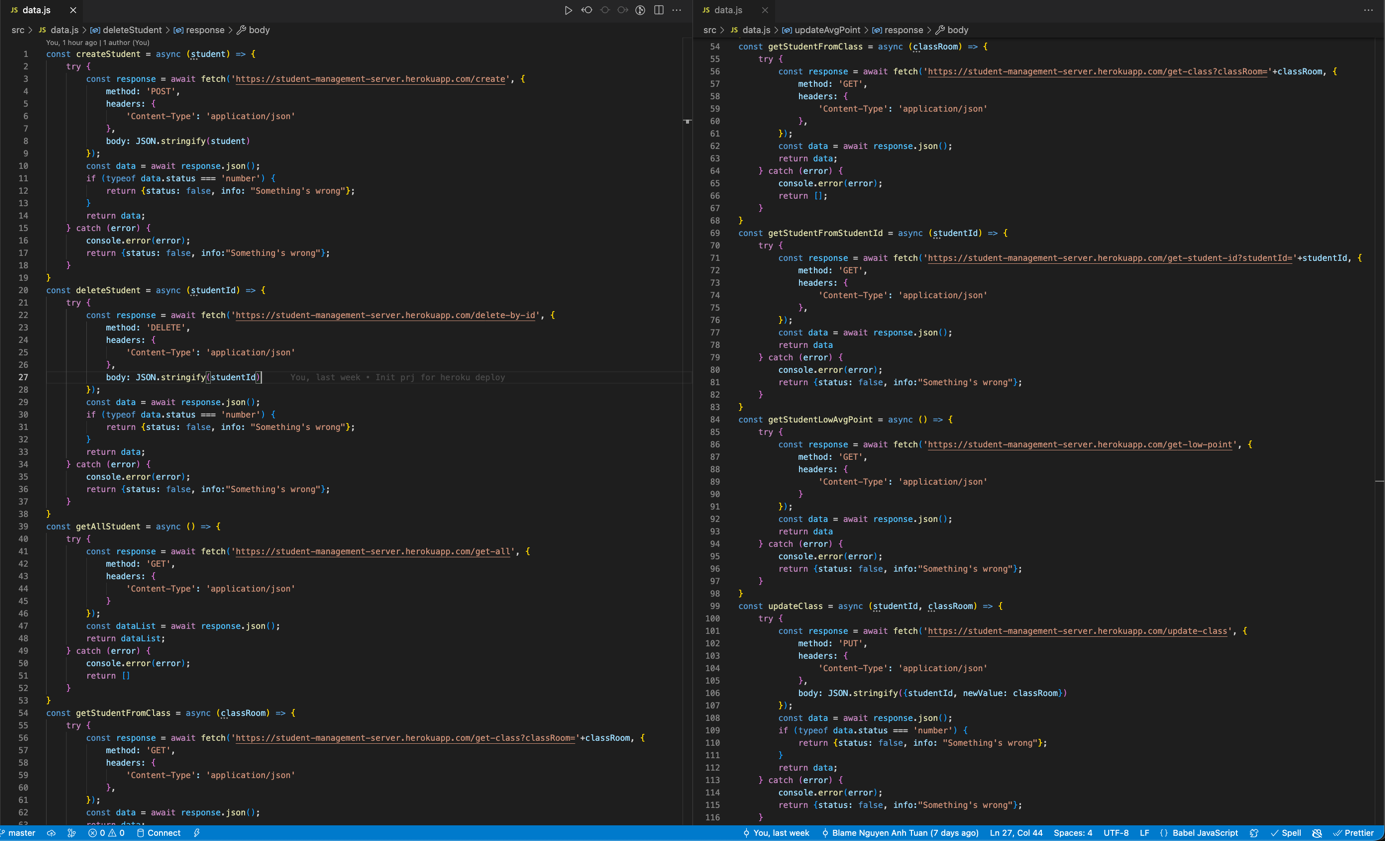


File Req.java và DoubleReq.java hỗ trợ các object truyền đi cho request







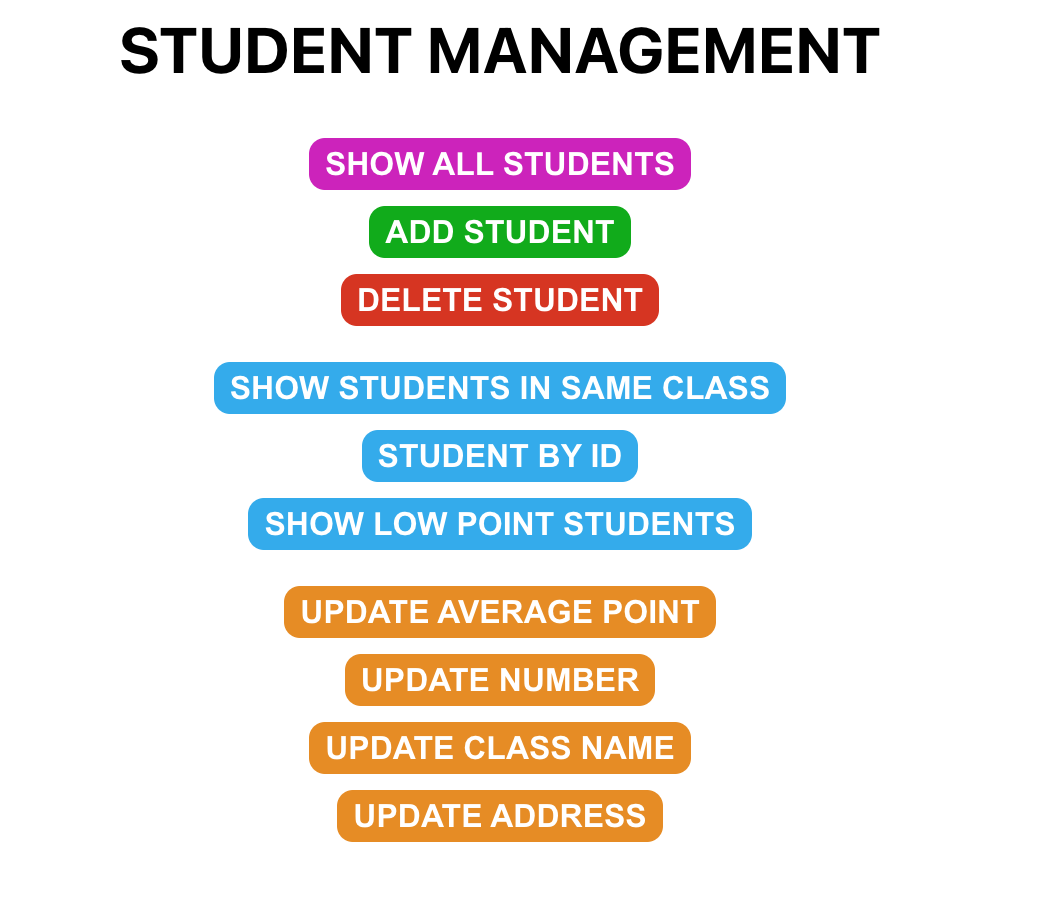
Về phần clinet-side, phần này không áp dụng gì liên quan đến OOP vì vậy em xin phép không đề cập code trong này vì khá dài, nội dung chủ yếu là tạo các nút và thao tác để giao tiếp với server-side này và hiển thị kết quả trả về, cụ thể code lấy data như sau: ….

Sau khi deploy bằng heroku thì ta có một trang web cho server-side và clinet-side như sau: [link demo server-side](https://student-management-server.herokuapp.com/) + [link demo client-side](https://student-management.herokuapp.com/)

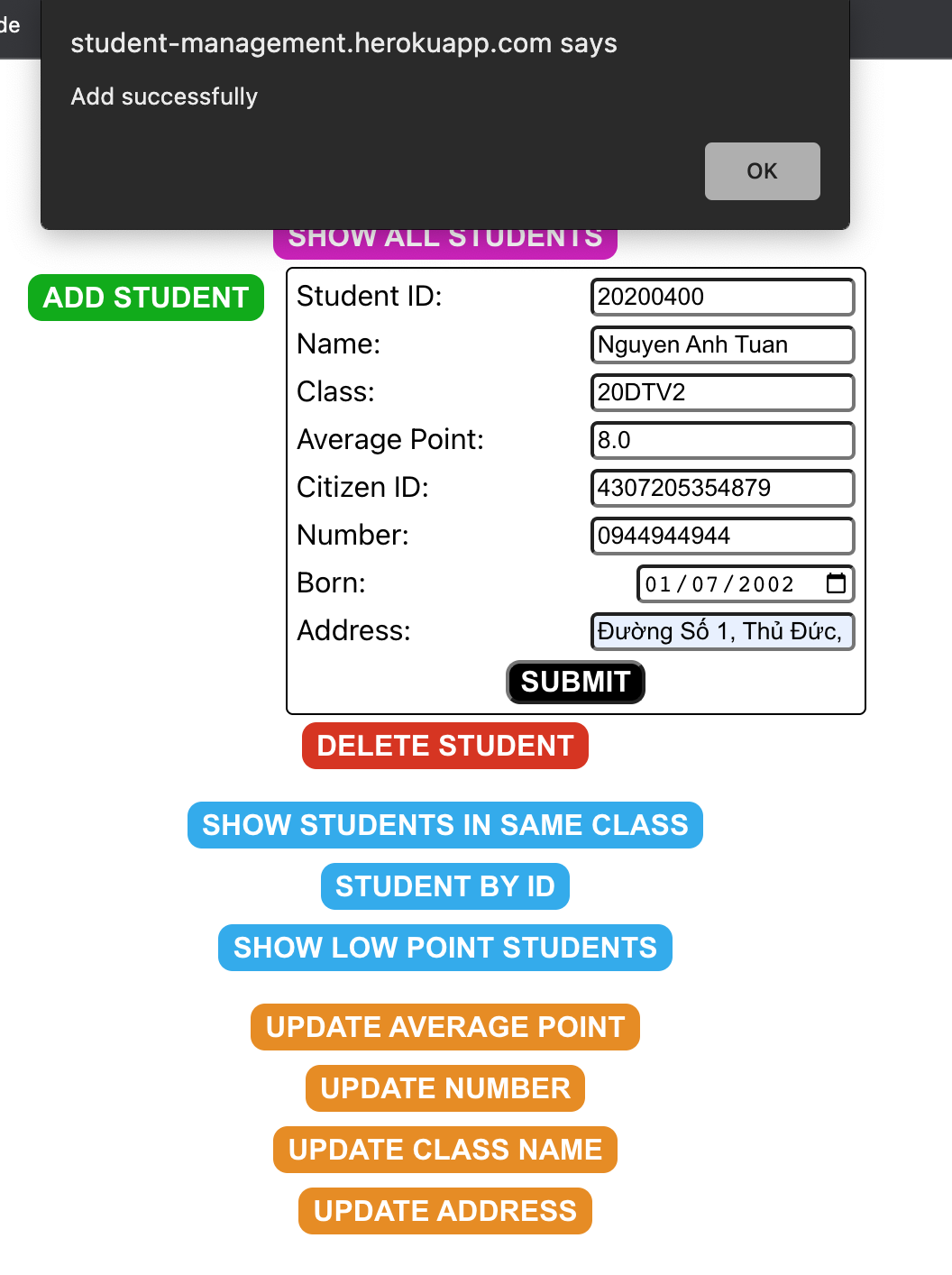
1. **Minh hoạ trực quan:**

Link sản phẩm: <https://student-management.herokuapp.com/> để có trải nghiệm thực tế tốt nhất.

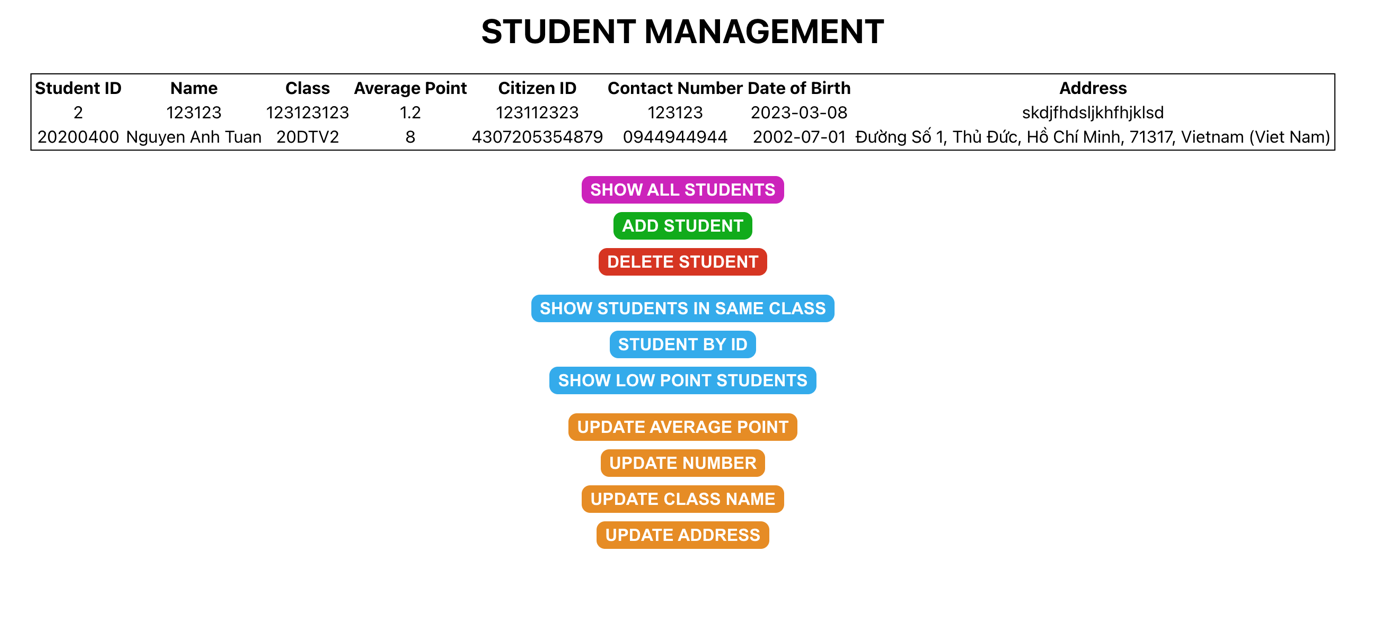
Giao diện:



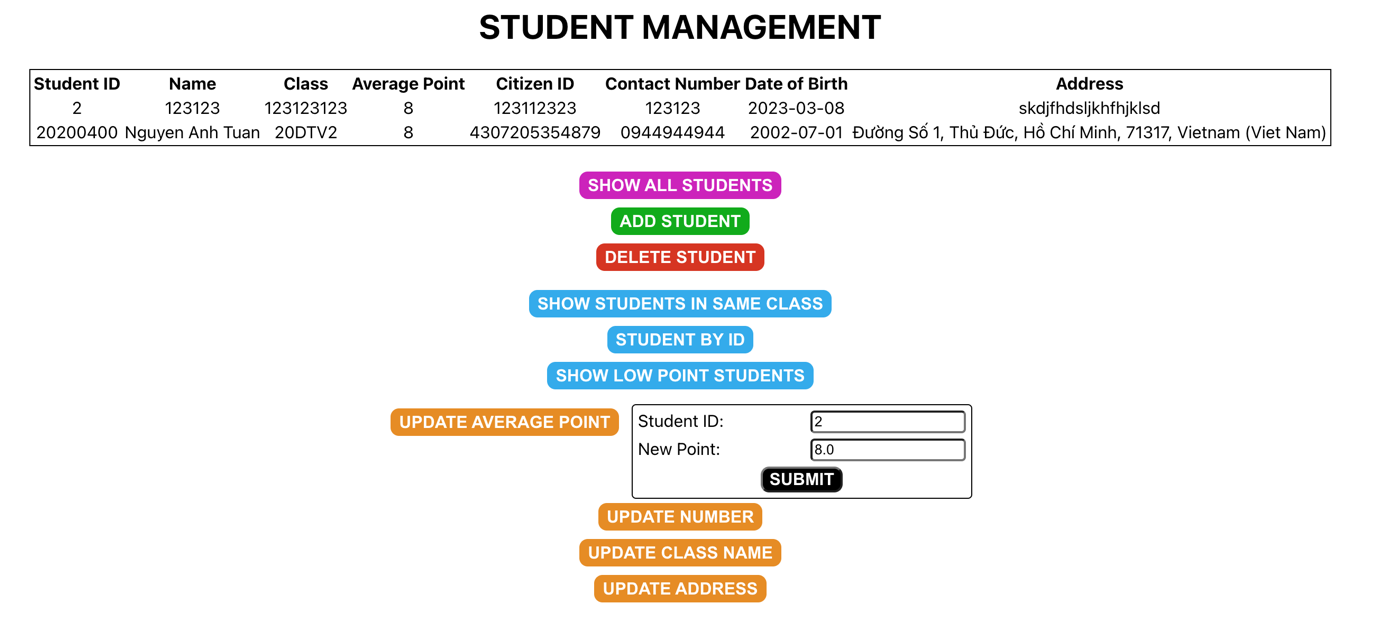
Thao tác thêm học sinh:



Xem danh sách học sinh:



Cập nhật điểm 8.0 cho học sinh có id là 2:



1. **Tổng kết:**

Các tính chất OOP được sử dụng:

* Đóng gói: để bảo vệ dữ liệu thuộc tính và cung cấp các getter, setter để truy cập và chỉnh sửa chúng một ách độc lập, điều này giúp đảm bảo truy cập ổn định.
* Đa hình: các phương thức khởi tạo học sinh, người, response… với các thuộc tính truyền vào.
* Kế thừa: lớp student kế thừa các thuộc tính của person như name, citizenid, address và các method đóng gói khởi tạo liên quan.
* Trừu tượng: dùng interface mô tả các phương thức trừu tượng triển khai ở repository và service

Cách vận hành: trong cách hình thành mảng học sinh, mảng này dùng collection ArrayList với pattern Singleton sẽ luôn khởi chạy khi server còn chạy không reset, khi reload vẫn không bị mất dữ liệu không cần apply database như SQL,…

Công nghệ áp dụng:

* Ngôn ngữ: Java, Javascript
* Framework: Spring, Reactjs
* Hosting: Heroku

|  |  |  |
| --- | --- | --- |
| **Họ tên** | **Công việc phân công** | **Mức độ hoàn thành** |
| Nguyễn Anh Tuấn | Phân bố công việc, đưa ra ý tưởng và code chính | 10/10 |
| Nguyễn Gia Phụng | Hỗ trợ và phát triển các tính năng giao diện | 9/10 |
| Nguyễn Hữu Luật | Xây dựng và đóng góp ý tưởng xây dựng giao diện + word | 9/10 |

|  |  |
| --- | --- |
| Nhận xét giáo viên |  |

*Đồ án này không tham khảo ở bất kỳ nguồn nào*