BTLT-Individual-W10-02

Đề bài:

SV review qua các bài lab của Thầy Quang, chỉ ra các nguyên lý **SOLID** được áp dụng trong các bài lab và bài tập mà bạn đã làm.

1. Lý thuyết về SOLID

S – Single Responsibility Principle (Nguyên tắc trách nghiệm duy nhất): Một lớp chỉ nên có một lý do để thay đổi. Nói cách khác, một lớp chỉ nên thực hiện một nhiệm vụ cụ thể.

O – Open/Closed Principle (Nguyên tắc mở/đóng): Phần mềm nên được thiết kế sao cho có thể mở rộng mà không cần phải sửa đổi mã nguồn đã có.

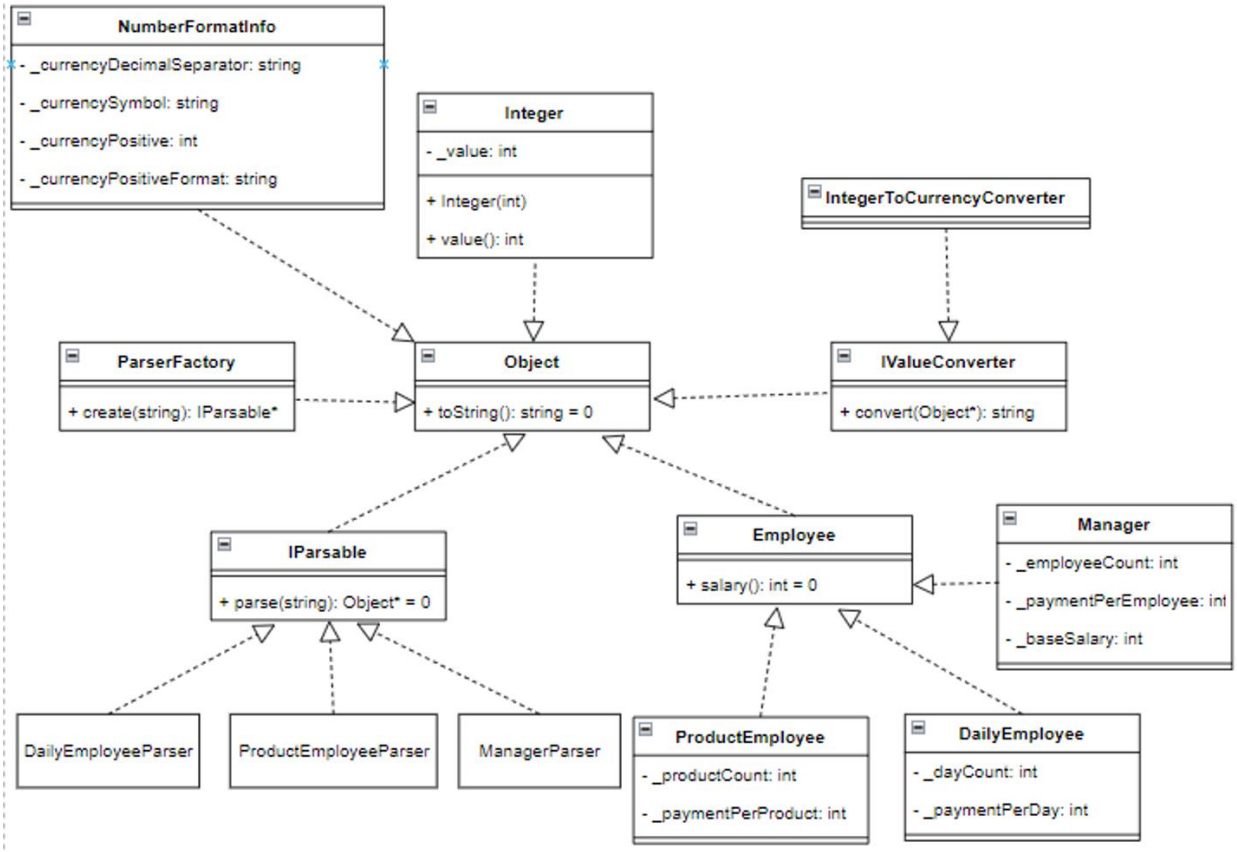
L – Liskov Substitution Principle (Nguyên tắc thay thế Liskov): Đối tượng của một lớp con có thể thay thế đối tượng của lớp cha mà không làm thay đổi tính đúng đắn của chương trình.

I – Interface Segregation Principle (Nguyên tắc phân chia giao diện): Một giao diện chỉ nên được thiết kể dành riêng cho các khách hàng của nó. Khách hàng không nên ép buộc phải sử dụng các phương thức mà họ không cần.

D – Dependency Inversion Principle (Nguyên tắc phụ thuộc đảo ngược): Các module cấp cao không nên phụ thuộc vào các module cấp thấp. Cả hai nên phụ thuộc vào các Abstraction. Điều này giúp giảm sự ràng buộc giữa các module và làm cho mã nguồn dễ dàng bảo trì và mở rộng hơn.

2. Review các bài Lab:

a. Lab 08 – Polymorphism



Single Responsibility Principle (SRP): Các class đảm nhiệm các nhiệm vụ cụ thể

- Các lớp như Manager, DailyEmployee, ProductEmployee, DailyEmployeeParser, ProductEmployeeParser, và ManagerParser đều có một trách nhiệm duy nhất:
- Manager, DailyEmployee, ProductEmployee: Mỗi lớp này đại diện cho một loại nhân viên và chỉ quản lý thông tin liên quan đến loại nhân viên đó. DailyEmployeeParser, ProductEmployeeParser, ManagerParser: Mỗi lớp này chịu trách nhiệm phân tích và tạo đối tượng từ chuỗi đầu vào liên quan đến loại nhân viên cụ thể.

Open/Closed Principle (OCP): Chương trình được thiết kế để có thể mở rộng mà không cần thay đổi mã nguồn đã có

- Các lớp DailyEmployeeParser, ProductEmployeeParser, và ManagerParser đều kế thừa từ lớp IParsable. Điều này cho phép dễ dàng thêm mới các lớp parser mà không cần phải sửa đổi mã nguồn của các lớp hiện có.
- Lớp ParserFactory có thể mở rộng bằng cách thêm các parser mới vào _container mà không cần sửa đổi mã nguồn cốt lõi của lớp này.

Liskov Substitution Principle (LSP): các đối tượng của lớp con phải có thể thay thế cho các đối tượng của lớp cha mà không làm thay đổi tính đúng đắn của chương trình.

- Các lớp Manager, DailyEmployee, ProductEmployee kế thừa từ Employee và tất cả đều có triển khai phương thức salary(). Do đó, đối tượng của các lớp con này có thể thay thế cho đối tượng của lớp cha Employee mà không gây ra lỗi.

Interface Segregation Principle (ISP): không nên buộc các lớp phải triển khai các giao diện mà chúng không sử dụng. Thay vì đó, nên tách giao diện lớn thành các giao diện nhỏ hơn.

- Giao diện IParsable chỉ chứa các phương thức liên quan đến việc phân tích chuỗi đầu vào (parse) và tên của đối tượng đã phân tích (parsedObjectName). Điều này đảm bảo rằng các lớp triển khai IParsable chỉ cần thực hiện các phương thức liên quan đến trách nhiệm của chúng.

Dependency Inversion Principle (DIP): các mô-đun cấp cao không nên phụ thuộc vào các mô-đun cấp thấp. Cả hai nên phụ thuộc vào các trừu tượng (abstract). Trừu tượng không nên phụ thuộc vào chi tiết.

- Lớp ParserFactory phụ thuộc vào giao diện IParsable thay vì phụ thuộc vào các lớp cụ thể như DailyEmployeeParser, ProductEmployeeParser, hay ManagerParser. Điều này cho phép thay đổi hoặc thêm mới các parser mà không cần thay đổi mã nguồn của ParserFactory.