객체지향 프로그래밍(OOP)

생성일	@2023년 11월 23일 오후 1:46
태그	패러다임

객체 지향 프로그래밍이란?

- 객체들의 집합으로 프로그램의 상호작용을 표현하며 데이터를 객체로 취급하여 객체 내부에 선언된 메서드를 활용하는 방식
- 설계 소요 시간 🔺 처리 속도 🔻

특징

캡.상.추.다

캡슐화

- 객체의 속성과 메서드를 하나로 묶고 일부를 외부에 감추어 은닉하는 것
- 결합도 🔻 응집도 🔺

상속

- 상위클래스의 특성을 하위 클래스가 이어받아서 재사용하거나 확장하는 것
- 코드의 재사용성 🔺

추상화

- 복잡한 현실 세계의 객체를 간추려 표현하고, 핵심적인 특징을 추출하는 과정
- 중요한 부분을 강조하고 불필요한 부분은 숨김으로써, 프로그램의 이해도를 높이고 코드의 일반화를 가능하게 함

다형성

- 하나의 메서드나 클래스가 다양한 방법으로 동작하는 것

  (같은 형태지만 다른 기능을 함)
- **메서드 오버라이딩, 메서드 오버로딩**
- 코드 유연성, 확장성 🔺
- 객체 간에 유사한 동작을 공유할 수 있게 함

메서드 오버라이딩 VS 메서드 오버로딩

메서드 오버로딩

- 한 클래스 내에서 메서드 이름은 같으나 매개변수나 반환값이 다른 경우
- 컴파일 중 발생하는 정적 다형성

메서드 오버라이딩

- 상속 관계에서 부모 클래스의 메서드를 자식 클래스에서 재정의하는 것
- 메서드의 이름, 매개변수, 리턴값이 모두 같아야함
- 런타임 중 발생하는 동적 다형성

OOP 설계원칙

SOLID

각 원칙은 소프트웨어 디자인의 유연성, 확장성, 유지보수성을 향상시키는 데 목적이 있습니다

단일 책임 원칙(SLP, Single Responsibility Principle)

- 모든 클래스는 각각 하나의 책임만 가져야 한다
- 코드의 의존성 🔻 결합도 🔻
- 객체가 담당하는 동작 즉 책임이 많아질 수록 해당 객체의 변경에 따른 영향도의 양과 범위가 매우 커진다.

  단일 책임 원칙은 특정 객체의 책임 의존성 과중을 최대한 지양하기 위한 원칙이다.
- https://blog.itcode.dev/posts/2021/08/13/single-responsibility-principle

개방-폐쇄 원칙(OCP, Open Closed Principle)

- 유지 보수 사항이 생긴다면 코드를 쉽게 확장할 수 있어야하고 수정에는 닫혀있어야 한다
- 새로운 기능이 추가되거나 변경되더라도 기존의 코드를 수정하지 않고 확장할 수 있어야 한다.
- https://velog.io/@harinnnnn/OOP-객체지향-5대-원칙SOLID-개방-폐쇄-원칙-OCP

리스코프 치환 원칙(LSP, Liskov Substitution Principle)

- 파생 클래스는 기본 클래스로 교체 가능해야 한다. 즉, 기본 클래스 타입의 객체에 대해 파생 클래스의 객체를 사용해도 프로그램의 의미나 정확성이 변하지 않아야 한다.
- 부모 객체에 자식 객체를 넣어도 시스템이 문제없이 돌아가게 만드는 것
- 부모 객체 P와 자식 객체 S가 있을 때, P↔S 해도 문제가 없어야함
- https://velog.io/@harinnnnn/OOP-객체지향-5대-원칙SOLID-리스코프-치환-원칙-LSP

인터페이스 분리 원칙(ISP, Interface Segregation Principle)

- 하나의 일반적인 인터페이스보다 구체적인 여러 개의 인터페이스를 만들어야 하는 원칙
- 클라이언트는 자신이 사용하지 않는 메서드에 의존 관계를 맺으면 ❌

  즉, 큰 인터페이스를 작은 여러 개의 구체적인 인터페이스로 분리함으로써 클라이언트는 자신이 필요로 하는 메서드에만 의존하도록 하는 것.
- https://velog.io/@harinnnnn/OOP-객체지향-5대-원칙SOLID-인터페이스-분리-원칙-ISP

의존 역전 원칙(DIP, Dependency Inversion Principle)

- 객체는 구체적인 객체가 아닌 추상화에 의존해야 한다는 법칙입니다. 자신보다 변하기 쉬운 것에 의존해서는 안됩니다.
- https://velog.io/@harinnnnn/OOP-객체지향-5대-원칙SOLID-의존성-역전-원칙-DIP

예상 질문

❓객체 지향 프로그래밍에 대해 설명해주세요

객체들의 집합으로 프로그램의 상호작용을 표현하며 데이터를 객체로 취급하여 객체 내부에 선언된 메서드를 활용하는 방식입니다

특징으로는 불필요한 정보를 외부로부터 감추는 캡슐화, 부모 클래스가 자식 클래스에게 속성을 물려주어 코드 중복을 없애는 상속, 객체의 공통된 특징을 뽑아내는 추상화, 같은 형태이지만 다양한 기능을 하는 다형성이 있습니다.

❓오버라이딩과 오버로딩의 차이점에 대해 설명해주세요

오버로딩은 같은 이름을 사용해서 새로운 매셔드를 재정의 하는 것으로 매개변수나 반환값을 다르게 지정하여 메서드를 정의합니다.

오버라이딩은 부모클래스로부터 상속받은 기존의 메서드를 재정의하는 것으로 메서드의 이름, 매개변수, 반환값이 모두 같게하고 재정의해야 합니다.

❓OOP 설계 원칙에 대해 설명해주세요

모든 클래스는 각각 하나의 책임만 가져야 한다는 단일 책임 원칙,

유지 보수 사항이 생긴다면 코드를 쉽게 확장할 수 있어야하고 수정에는 닫혀있어야 한다는 개방-폐쇄 원칙,

파생 클래스는 기본 클래스로 교체 가능해야 한다는 리스코프 치환 원칙,

하나의 일반적인 인터페이스보다 구체적인 여러 개의 인터페이스를 만들어야 하는 인터페이스 분리 원칙,

객체는 구체적인 객체가 아닌 추상화에 의존해야 한다는 의존 역전 원칙이 있습니다.