8장 상속

상속의 목적 및 장점

- 클래스의 간결화
- 구조적 관계 파악
- 재활용과 확장을 통한 생산성 향상

캐스팅
업캐스팅

- 파생클래스의 객체를 기본클래스의 포인터로 가리키는것

다운캐스팅
-기본 클래스 포인터가 가르키고있는 객체를 파생클래스의 포인터로 가르키는것

접근지정자
protect 멤버는 본인의 자식에게, 즉 상속을 주는? 멤버 클래스에게만 접근을 허용.

상속의 목적 및 장점:

1. **클래스의 간결화:** 상속을 통해 기존 클래스의 특성을 그대로 가져와 새로운 클래스를 정의할 수 있습니다. 이렇게 함으로써 코드 중복을 피하고, 코드의 간결성을 유지할 수 있습니다.
2. **구조적 관계 파악:** 상속은 클래스 간의 계층 구조를 형성합니다. 이를 통해 클래스 간의 관계를 시각적으로 파악하고, 코드의 구조를 더욱 명확하게 만듭니다.
3. **재활용과 확장을 통한 생산성 향상:** 기존 클래스를 재활용하여 새로운 클래스를 만들 수 있으므로 코드의 생산성이 향상됩니다. 또한, 새로운 기능이나 속성을 추가하여 클래스를 확장할 수 있다.

캐스팅:

1. **업캐스팅 (Upcasting):**
   - 파생 클래스의 객체를 기본 클래스의 포인터로 가리키는 것.
   - 부모 클래스로의 포인터 또는 참조를 통해 파생 클래스의 객체에 접근할 수 있다.
2. **다운캐스팅 (Downcasting):**
   - 기본 클래스 포인터가 가리키고 있는 객체를 파생 클래스의 포인터로 가리키는 것.

접근지정자:

- `protected` 멤버는 자식 클래스에게만 접근을 허용한다.
- 이것은 부모 클래스의 멤버 중에서 자식 클래스에서만 사용되어야 하고 외부에서는 직접 접근되면 안 될 때 사용된다.
- 캡슐화를 유지하면서 상속을 통한 유연성을 제공한다

다중 상속:

- 다중 상속은 하나의 파생 클래스가 여러 개의 기본 클래스로부터 상속받는 것을 말합니다.
- 장점:
  - 코드 재사용이 증가

    - 여러 클래스로부터 상속받아 새로운 클래스를 만든다.
    - 다양한 기능을 하나의 클래스에서 모아 사용할 수 있다.
- 단점:
    - 다이아몬드 문제: 여러 경로를 통해 같은 클래스를 상속받을 때 발생하는 모호성
    - 이를 해결하기 위해 가상 상속을 사용
    - 클래스 간의 복잡한 관계로 코드를 이해하고 유지보수하기 어려울 수 있다.

가상 상속:

- 가상 상속은 다이아몬드 문제를 해결하기 위한 메커니즘,
- 가상 상속을 받은 클래스는 공통의 기본 클래스를 딱 한 번만 상속한다.
- `virtual` 키워드를 사용하여 가상 상속을 선언한다.
- 가상 상속을 받은 클래스는 중복으로 상속되지 않고,
- 가상 상속을 사용하지 않은 클래스는 중복으로 상속된다.
- 예시:

  class A {};
  class B : public virtual A {};
  class C : public virtual A {};
  class D : public B, public C {};`
- 이 경우, `D`는 `A`를 중복으로 상속받지 않는다.

주제생각 및 고민, 토론한 내용.

***주제 : 다중 상속과 단일 상속 중 어떤 것이 더 효율적인가?**

-> 다중상속은 코드 재사용성을 극대화한다. 또한 다른 여러 클래스들로부터 동시에 상속을 받을수 있다. 유연한 코드를 짜는데 좋을듯함.

-> 단일 상속은 코드를 더 간결하게 만들어준다 이로써 유지보수가 용이하며, 서로다른 클래스간의 관계가 명확할 것이다.

-> 다중상속은 다이아몬드 문제로 인해 모호성을 야기할 수 있다는 문제가 있다는것이 단점!

-> 단일, 다중 상속은 각각 장단점을 가지고 있으므로 상황에 따라 다른 상속을 주는것이 올바르다고 생각함.

-> 간단한 프로젝트에서의 클래스간의 관계로는 단일상속을사용하는것이 직관적으로 보임으로 효율적일것 같고, 큰 프로젝트에서는 다중상속의 다이아몬드 문제를 해결한다는 가정하에. 다중상속을 사용하는것이 더욱 효율적인 코드를 짤 수 있을것이라 생각함.