객체지향프로그래밍
정리노트#6

09. 가상함수와 추상 클래스 (11주차)

가상함수 (virtual function) 란 virtual 키워드로 선언된 멤버 함수

함수 오버라이딩 (function overriding) 파생클래스에서 기본클래스의 가상함수를 재정의하는 것

함수 재정의 (redefine) = 컴파일 시간 다형성 (compile time polymorphim)

↓

오버라이딩 ⟹ 실행 시간 다형성 (run time polymorphism)

<오버라이딩 목적>

기본 클래스에 가상 함수를 만드는 목적은 파생 클래스들이
각자의 목적에 맞게 가상함수를 재정의함

동적 바인딩 (오버라이딩 함수가 목적인 호출)

<동적 바인딩 과정>

가상함수가 호출하는 코드를 컴파일할때, 컴파일러는 바인딩을 실행시간에 결정하도록 미룸.

가상함수 호출, 실행 중에 객체 내에 오버라이딩 된
가상함수를 동적으로 찾아 호출한다.

<동적 바인딩이 발생하는 경우>

- 기본 클래스 내의 멤버함수가 가상함수 호출
- 파생 클래스 내의 멤버함수가 가상함수호출
- Main() 과 같은 외부함수에서 기본클래스의 포인터로 가상함수호출
- 다른 클래스에서 가상함수호출

<동적 바인딩 사례>

(a)
```
Shape *pShape = new Shape();
pShape -> Paint(); // Paint()는 Shape의 draw() 함수를 호출한다.
```
Shape 클래스만 있는 경우로서, Shape의 draw() 가 실행

(b)
```
Shape *pShape = new Circle();
pShape -> Paint(); // Paint()는 Circle에서 오버라이딩한 draw() 함수를 호출한다.
```
Paint 함수 pShape이 가리키는 draw() 반환

동적 바인딩을 통해 Circle의 draw() 호출

<오버라이딩과 범위지정연산자 (::)>

(1) Main() 함수와 같이 클래스 외부에서 호출하는 경우
```
Circle circle;
Shape *pShape = &circle;
pShape -> Shape::draw(); // 정적 바인딩. Shape의 멤버함수 draw() 호출
```
(2) 클래스의 멤버함수에서도 범위지정 연산자를 이용하여 기본 클래스의 가상함수 호출하는 경우
```
class Circle : Public Shape {
Public:
    virtual void draw() {
        Shape::draw(); // 기본클래스 Shape의 draw() 실행. 정적바인딩
        // 확장된 기능 추가
    }
};
```

<가상소멸자>

- 소멸자를 가상함수로 선언하지 않은 경우
```
Base *p = new Derived();
delete *p;
```
→ Base타입 컴파일러는 ~Base() 소멸자를 호출하도록 컴파일

- 소멸자를 가상함수로 선언한경우

~Base() 에 대한 호출 → (실행중에 동적바인딩) ~Derived() 에 대한 호출 → ~Derived 실행

<순수 가상함수>

순수가상함수 (Pure virtual function)은 함수의 모드가 없고 선언만 있는 가상함수

추상클래스 ⇒ 불완전한 클래스

```
Shape shape1; // 컴파일 오류
Shape *p = new Shape(); // 컴파일오류
```

<이때 발생하는 오류 메시지>

error C2259: 'Shape': 추상클래스로 인스턴스를 생성할 수 없습니다