<클래스>

1. 클래스란

: 우선 클래스는‘객체를 정의해놓은 것’또는 ‘객체의 설계도 또는 틀’이라고 정의할 수 있다. 클래스는 객체를 생성하는데 사용되며, 여기서 객체는 클래스에 정의된 대로 생성된다. 쉽게 말하면 클래스에는 특정 객체에 대한 속성과 기능이 정의되어 있으며, 클래스로부터 객체를 생성하면, 클래스에 정의된 속성과 기능을 가진 객체가 만들어지는 것이다. 이것을 흔히 붕어빵 틀과 붕어빵에 비유하곤 하는데, 붕어빵 틀에는 ‘붕어빵’이라는 객체의 특징들이 담겨져 있고, 이 붕어빵 틀을 이용해 붕어빵을 찍어내게 되면 붕어빵 틀에 있는 객체의 특징들이 새롭게 생성된 붕어빵에 담겨있는 것이다.

원이(가) 표시된 사진

자동 생성된 설명

\* 여기서 클래스로부터 객체를 만드는 과정을 클래스의 인스턴스화라고 하며, 어떤 클래스로부터 만들어진 객체를 그 클래스의 인스턴스라고 한다.

2. 상속이란

: 상속이란 클래스간의 코드를 재사용하고 조직화하는 방법이자, ‘부모’ 클래스에서 ‘자식 클래스로 속성과 메서드를 전달하는 과정이다. 예를 들자면 사람이란 클래스와 학생이란 클래스가 있다고 가정해보자. 사람이라는 클래스 내에는 사람을 나타내는 여러 속성과 행동들이 정의되어 있고, 학생이라는 클래스에도 역시 학생을 나타내는 여러 속성들이 정의되어 있을 것이다. 하지만 여기서 학생은 사람이라는 부류에 속하므로 학생 클래스와 사람 클래스가 공통적으로 가지고 있는 속성(이름, 나이, 소속 등)이 존재한다. 이 부분을 C++에서는 여러가지 이점을 고려해 학생 클래스 내부에서 따로 정의, 선언할 필요 없이 사람 클래스 내의 속성과 행동들을 물려받아 쓸 수 있도록 하는 것이 바로 ‘상속’이다.

3. 상속의 장점

- 3-1 : 코드의 재사용성 증가 -> 코딩 시간 절약, 코드의 일관성 유지

- 3-2 : 클래스 확장 용의

- 3-3 : 유지 보수성 향상 -> 상위 클래스만 수정하면 하위 클래스에 변경이 반영되기 때문

4. friend 클래스란

: friend 클래스는 특정 상황에서 클래스 내에 접근하지 못하도록 하는 private 제한을 필요한 경우 friend로 선언된 다른 클래스의 private 및 protected 멤버에 접근할 수 있도록 해주는 클래스이다. Friend 클래스는 friend class 클래스명으로 사용할 수 있다.

Ex) friend class Friend2;

5. static 멤버 변수

: static 멤버 변수란 모든 객체가 한 메모리를 공유하는 멤버 변수로, 객체별로 각각 할당되는 멤버가 아니라 모든 객체가 공유하는 멤버다. 때문에 메모리는 프로그램 시작때부터 차지하고 있어 메모리에 내내 값이 유지되며, 멤버 변수는 모든 객체가 공유해야 하므로 반드시 전역 범위에서 정의 및 초기화(자료형이름 클래스이름::static변수이름 = 초기화 할 값)를 해주어야 한다. 또한 모든 객체는 static 변수의 메모리를 공유하기 때문에 객체의 주소가 모두 동일하다.

\* static 멤버 함수 : 객체 없이도 호출할 수 있는 함수로 static 멤버 변수에만 접근이 가능하다.

6. const

: c++에서 const는 말 그대로 그 대상을 변경하지 않는 ‘상수’를 의미하며, “const + 클래스이름 + 객체이름”으로 상수 클래스를 선언할 수 있다. 이렇게 선언된 객체의 멤버변수는 상수 멤버 함수만 명시적으로 호출할 수 있으며, 상수 멤버 함수가 아닐 경우 컴파일 에러가 발생한다. 이때 상수 멤버 함수를 만들려면 const 키워드를 함수 선언의 매개 변수 목록 뒤에 추가하면 된다.

ex) int GetValue() const { return m\_Value; }(선언), int Something::GetValue() const(정의)

7. 접근 제어 지시자

: 접근 제어 지시자란 클래스의 멤버 변수나 멤버 함수에 대한 접근을 제어할 수 있는데, 이 접근 제어지시자에는 public, private, protected 세가지가 존재한다. 접근 제어 지시자를 통해 클래스 외부에서 접근 가능한 멤버와 접근 불가능한 멤버를 명확히 구분할 수 있으며, 이를 활용해 더 안정적이고 유지보수가 쉬운 코드를 작성할 수 있다.

- 7-1 public

: 해당 멤버에 대한 접근을 제한하지 않는다. 클래스 내외부에서 모두 접근 가능하며, 이를 통해 객체의 기능을 외부에 제공한다.

- 7-2 private

: 해당 멤버에 대해 외부 접근을 완전히 차단한다. 클래스 외부에서는 접근이 불가능하며, 클래스 멤버 함수만이 해당 멤버에 접근할 수 있다.

- 7-3 protected

: private와 비슷하지만 protected는 해당 클래스 및 그 클래스를 상속받은 자식클래스에서만 접근이 가능하다.

8. 상속의 종류

: 상속의 종류는 크게 단일 상속, 다중 상속, 가상 상속으로 나눌 수 있다. 상속받는 클래스의 수에 따라 단일 상속과 다중 상속이 결정되며, 다중 상속시 발생하는 ‘다이아몬드 문제’를 해결하기 위해 가상 상속을 사용할 수 있다.

8-1 단일 상속

: 한 개의 상위 클래스로부터만 상속을 받는 형식으로, 가장 기본적인 상속 구조이다.

8-2 다중 상속

: 한 클래스가 두개 이상의 상위 클래스로부터 상속을 받는 형식으로, 단일 상속에 비해 보다 다양한 기능을 결합할 수 있어 코드 재사용성과 효율성이 높아지며, 여러 상위 클래스의 속성을 동시에 가질 수 있기 때문에 다양한 기능들을 조합할 수 있는 장점을 가지고 있다. 하지만 다중 상속을 잘못 사용되면 프로그램의 복잡성을 높이고, 이해하기 어려운 코드를 생성할 수 있으므로 신중한 사용이 필요하다. 특히 다중 상속을 사용하면서 발생하는 2가지의 문제를 주의해야 하는데, 바로 ‘모호성 문제’와 ‘다이아몬드 문제’이다.

8-2-1 모호성 문제

: 다중 상속에서 상위 클래스에 동일한 이름의 멤버가 있을 경우 어떤 클래스의 멤버를 호출해야 할지 모호해지는 문제가 발생할 수 있다. 이 문제는 ‘명시적인 범위 지정’ 즉 자식 클래스에서 부모 클래스의 멤버를 참조할 때, 부모 클래스의 이름을 명시하여 어떤 부모 클래스의 멤버를 참조하는지 명확하게 해주는 방식으로 해결할 수 있다.

8-2-2 다이아몬드 문제

: 아래 그림과 같이 B클래스와 C클래스는 각각 A클래스에서 상속받고, D는 B클래스와 C클래스를 둘다 다중 상속받는다고 해보자, 우리는 1번 그림과 같이 A타입의 메모리가 하나만 존재할 것을 기대하지만 실제론 2번 그림처럼 A타입 메모리가 2개가 생성된다. 이 때문에 메모리 낭비 및 모호성이 발생하며, 이를 해결하기 위해 가상 상속을 사용한다.

도표, 직사각형, 라인, 디자인이(가) 표시된 사진

자동 생성된 설명

8-3 가상상속

: 가상 상속은 C++에서 다이아몬드 문제를 해결하기 위한 특별한 형태의 상속이다. C++에서 ‘virtual’ 키워드를 사용하여 상속을 선언하면, 기본 클래스는 단 한 번만 상속된다. 이로 인해 기본 클래스의 인스턴스는 하위 클래스에서 공유되며, 상속 받는 모든 경로에 상위 클래스의 단일 인스턴스만이 존재하도록 보장되고, 이를 통해 다이아몬드 문제가 해결된다.

도표, 라인, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

* class B : virtual public A, class C : virtual public A
* class D : public B, public C