제 9장. 상속

학습 목표

- 상속의 기본 개념과 is-a 관계를 이해한다.
- 파생 클래스를 정의하는 방법을 알아보고, 파생 클래스의 객체를 사용하는 방법을 알아본다.
- 파생 클래스의 생성자에 대해서 알아본다.
- 접근 지정자와 접근 변경자에 대해서 알아본다.
- 다중 상속의 개념을 이해하고 다중 상속을 사용할 때의 주의사항을 알아본다.

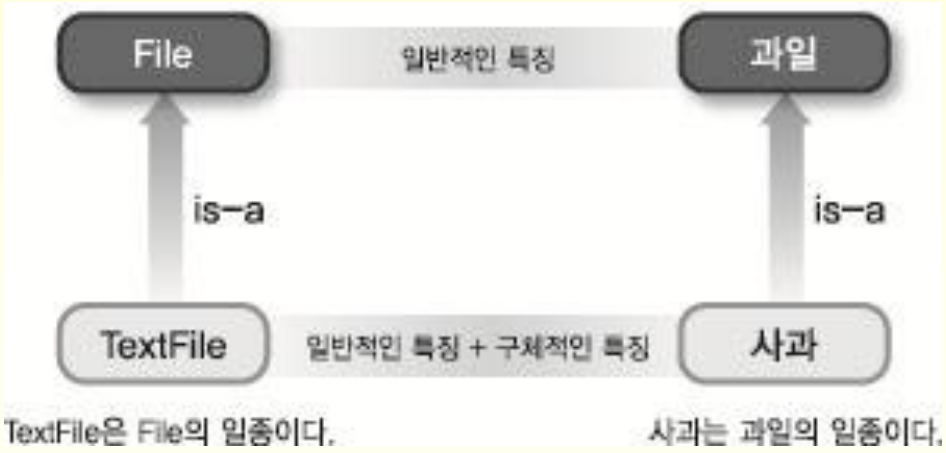
상속의 기본 개념

- 상속을 해주는 클래스를 기본 클래스, 상속을 받는 클래스를 파생 클래스라고 한다.

- 다른 클래스를 상속받아서 새로운 클래스를 정의할 때, 기존의 클래스에 구현되어 있는 기능은 새로운 클래스에 다시 구현할 필요가 없다.

is-a 관계

- 상속 관계에 있는 두 클래스 사이에는 is-a 관계가 성립한다.
  - "TextFile은 File이다(TextFile is a File)"의 관계가 성립한다. → TextFile 클래스는 File 클래스의 일종이다.
  - 일반적인 특징을 제공하는 기본 클래스에 구체적인 특징을 추가해서 이끌어낸 것이 파생 클래스이다.
  - 주의할 점은 is-a 관계는 한 방향으로만 성립한다는 것.



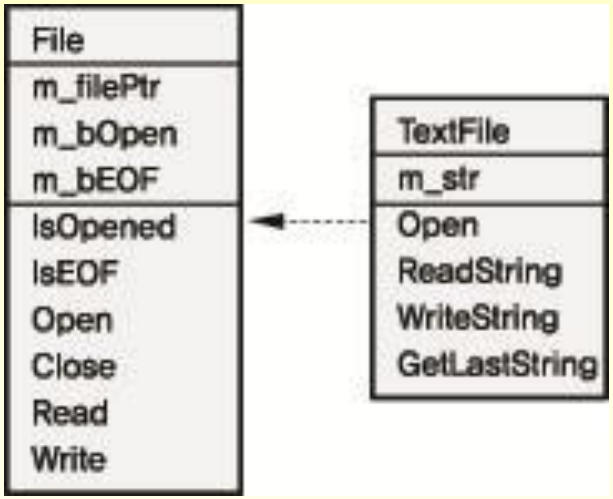
파생 클래스의 정의

- 파생 클래스를 정의하려면 클래스를 정의할 때 클래스 이름 다음에 콜론(:)을 쓰고 public 키워드와 함께 기본 클래스 이름을 적어준다.

```
class 파생클래스이름 : 접근변경자 기본클래스이름
{
};
```

- 파생 클래스의 멤버
  - 파생 클래스를 정의할 때 상속받는 멤버 변수는 다시 정의할 필요가 없음
  - 파생 클래스에는 새로 추가된 멤버 변수나 멤버 함수를 정의
  - 상속 받은 멤버 함수 중에서 기본 클래스와 다르게 처리할 멤버 함수를 재정의(overriding)

TextFile 클래스의 예




```
#include "File.h"          → 기본 클래스의 헤더 파일 포함

class TextFile : public File   → 기본 클래스 지정
{
protected:
  string m_str;            → 새로 추가된 멤버 변수

public:
  TextFile(void);
  ~TextFile(void);

  bool Open(const string& fileName,
            OPEN_MODE openMode);   → 재정의된 멤버 함수

  bool ReadString(string& s);
  bool WriteString(const string & s);
  string GetLastString() const;   → 새로 추가된 멤버 함수
};
```

파생 클래스의 객체 생성 및 사용

- 파생 클래스의 객체를 생성하면 항상 기본 클래스로부터 상속받은 멤버 변수가 먼저 메모리에 할당되고 그 다음에 파생 클래스에 추가된 멤버 변수가 메모리에 할당된다.


```
int main()
{
    TextFile file1;
    file1.Open(“test.txt”, File::out);    // 재정의된 멤버 함수 호출
    file1.WriteString("안녕하세요.");       // 새로 추가된 멤버 호출
    file1.Close();                        // 상속받은 멤버 호출
}
```

파생 클래스의 멤버

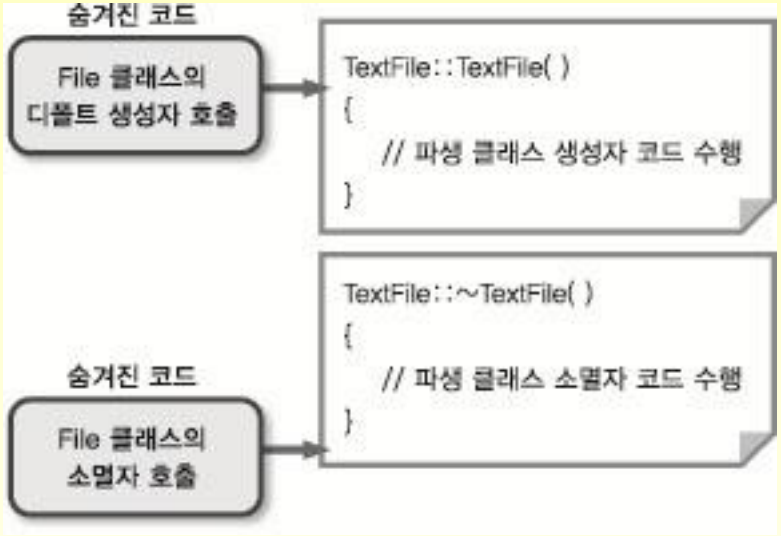
- 기본 클래스로부터 상속받은 멤버
- 새로 추가된 멤버
- 재정의된 멤버 함수
  - 상속받는 멤버 함수와 원형이 같은 함수를 정의하는 것
  - 멤버 변수는 재정의하지 않는다.

파생 클래스의 생성자와 소멸자 (1)

- 파생 클래스의 객체를 생성하면 파생 클래스 생성자는 항상 내부적으로 기본 클래스 생성자를 이용한다.
  - 기본 클래스로부터 상속받은 멤버 변수를 초기화하는데 기본 클래스의 생성자를 이용

- 파생 클래스의 객체가 소멸될 때 파생 클래스 소멸자는 내부적으로 기본 클래스의 소멸자를 호출한다.

파생 클래스의 생성자와 소멸자 (2)



- 기본 클래스 생성자 중 인자 있는 생성자를 호출하려면 파생 클래스 생성자에서 초기화 리스트를 사용한다.

```
TextFile::TextFile(const string& fileName, OPEN_MODE openMode)
: File(fileName, openMode)    초기화 리스트
{
}
```

접근 지정자

- 접근 지정자는 멤버의 접근 권한을 설정한다.
- 접근 지정자 설정 기준
  - 클래스 밖으로 공개되어야 하는 멤버는 public으로 지정한다.
  - 클래스 안에 숨겨둘 멤버는 protected로 지정한다.
  - 파생 클래스에도 숨겨야 하는 멤버는 private으로 지정한다.
- 일반적으로 멤버 변수는 protected나 private으로 지정하고, 멤버 함수는 public으로 지정한다.
  - 사용 상의 편이성 때문에 멤버 변수를 public으로 지정할 수도 있다.
- 클래스 안에 숨기는 멤버는 주로 protected으로 설정한다.
  - 파생 클래스에게 반드시 숨겨야만 하는 멤버는 거의 없기 때문에 private보다는 protected로 지정

접근 변경자

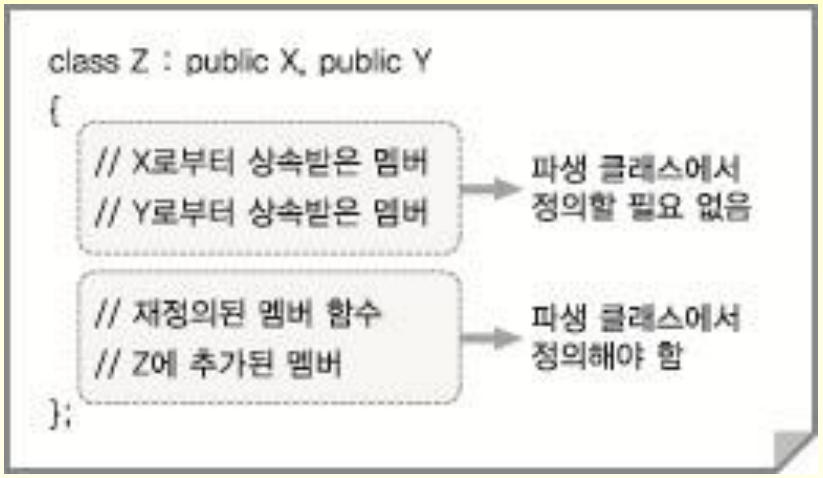
- 기본 클래스로부터 상속 받은 멤버의 접근 권한을 변경한다.
- 주로 public 상속을 사용한다.

접근 변경자	기본 클래스의 private 멤버	기본 클래스의 protected 멤버	기본 클래스의 public 멤버
private 상속	파생 클래스에서 접근 불가	파생 클래스의 private 멤버로 간주	파생 클래스의 private 멤버로 간주
protected 상속	파생 클래스에서 접근 불가	파생 클래스의 protected 멤버로 간주	파생 클래스의 protected 멤버로 간주
public 상속	파생 클래스에서 접근 불가	파생 클래스의 protected 멤버로 간주	파생 클래스의 public 멤버로 간주

- 접근 변경자도 접근 지정자처럼 디폴트로 private을 사용하므로 반드시 public을 명시적으로 지정해야 한다.

다중 상속

- 둘 이상의 기본 클래스를 동시에 상속받는 기능



다중 상속의 이름 충돌 문제

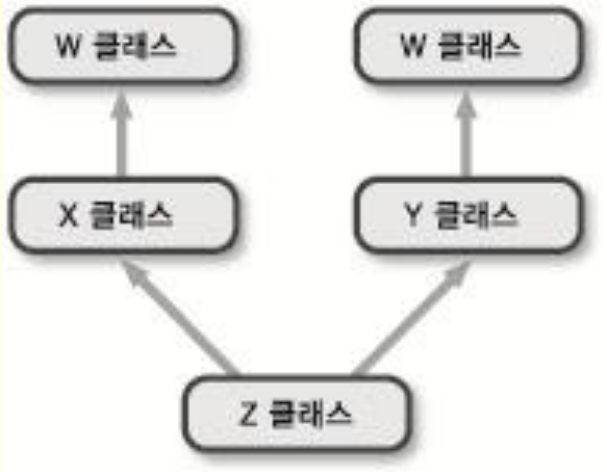
```
class X
{   ...
protected:
    int m_data;
public:
    void func()
    {  cout << "X::func 호출□n";
    }
};

class Y
{   ...
protected:
    int m_data;
public:
    void func()
    {  cout << "Y::func 호출□n";
    }
};
```

```
class Z : public X, public Y
{
public:
    void test()
    {
        cout << "Z::test 호출□n";
        //m_data = 10;      // 컴파일 에러
        X::m_data = 10;
        //func();           // 컴파일 에러
        Y::func();
    }
};
```

중복 상속

- 여러 기본 클래스를 다중 상속 받는 클래스를 정의할 때, 각각의 기본 클래스가 같은 클래스를 상속받는 경우를 말한다.



정리

- 기본 클래스의 특징을 이어받아 구체적인 특징을 추가로 제공하는 파생 클래스를 정의하는 기능이 바로 상속이다.
- 파생 클래스는 기본 클래스로부터 상속받은 멤버와 새로 추가된 멤버, 재정의된 멤버 함수를 갖는다.
- 파생 클래스 객체를 생성하면 항상 기본 클래스로부터 상속받은 멤버 변수부터 메모리에 할당하고 파생 클래스에 추가된 멤버 변수가 그 다음에 할당된다.
- 파생 클래스의 생성자는 기본 클래스로부터 상속받은 멤버를 초기화하기 위해서 기본 클래스 생성자를 이용한다. 파생 클래스 소멸자는 기본 클래스로부터 상속받은 멤버를 정리하기 위해서 기본 클래스 소멸자를 이용한다.
- 기본 클래스를 상속받을 때 기본 클래스 이름 앞에 private, protected, public 키워드를 사용하는데, 이것을 접근 변경자라고 한다. 상속의 is-a 관계는 public 상속을 통해서 얻어진다.
- 기본 클래스가 둘 이상인 경우 다중 상속이라고 한다.

실습과제

- 9장 실습과제 1, 2 [200점]
- 10장 실습과제 [100점]
- 도형 문제 [9, 10장 공통]
  - 삼각형, 사다리꼴, 평행사변형, 직사각형, 정사각형, 마름모, 원 등의 도형 클래스 (클래스 4개 이상 / 도형 클래스마다 100점 부여)
  - 이름, 색깔, 영역(왼쪽 최하단 점과 오른쪽 최상단 점으로 표현하는 직사각형), 중심점, 면적 등의 공통 성질을 갖는 도형 클래스 (300점)
  - 도형들 중 가장 면적이 큰 객체의 이름을 출력하는 함수 (200점)
  - main()에서는 도형 추가, 도형 삭제, 도형 리스트(또는 배열, vector) 출력, 최대 면적 도형 찾기, 개별 도형의 정보 출력 등을 작성하여 각 구현한 객체들의 변수나 기능 확인 (300점)
- 제출 기한 : 11월 20일 자정