

# inheritance

EUnS

January 12, 2020

① inheritance

② virtual function

③ etc

# 기본적인 모습

```
1  class sample
2  {
3  public:
4  virtual ~sample();
5  };
6  class sampleInheritance : public sample
7  {
8  public:
9  };
10
```

# 상속

- 상속하는쪽 : 부모(super) 클래스 받는쪽: 자식(sub) 클래스
- 큰범위의 클래스(ex 사람) 작은 범위의 단위(ex 학생)으로 쪼개질때 큰범위의 정보를 포함하고 작아짐으로서 세세해지는 정보 또한 들어갈때 상속을 씬 is a 관계(a student is a person)
- 외부에서는 부모 클래스의 public + 자식 클래스의 public을 쓸 수 있다.
- 상속 방식
  - public : 외부에서 접근하는것처럼 부모클래스의 private,protect에 직접적인 접근 할 수 없음
  - protected : 상속만을 위해 존재하는 접근지정자이며 추가적인것은 찾아볼것.
  - private : 추가적인것은 찾아볼것.
- 일반적으로 public상속을 쓴다(나머지 두개는 본인도 써본적없음)

## 생성자 소멸자 순서

- 소멸자는 가상함수로 선언하여야한다.
- 부모생성자 - 자식 생성자 순으로 불린다.
- 자식 소멸자 - 부모 소멸자 순으로 불린다.

# 가상함수(virtual function)

```
1  class sample
2  {
3      public:
4          virtual ~sample();
5          virtual void f() { ; }
6          virtual void g() { ; }
7      };
8  class sampleInheritance : public sample
9  {
10     public:
11         void f() override
12         { ; }
13         void g() final { ; }
14     };
15
```

## 가상함수(virtual function)

- 상속 받는 클래스는 부모 클래스의 함수를 덮어 쓸 수 있다. 자식 클래스에 맞게 메소드를 다시 짜는것 이를 override라 한다.
- override할 부모 클래스 메소드에 virtual을 붙인다.  
자식클래스에는 붙여도되고 안붙여도 됨.
- c++ keyword : [참고](#)
  - override : override받는 메소드에 붙일 수 있다. override keyword를 붙인 멤버 함수가 override된 멤버함수가 아닐경우에 에러를 뱉는다.
  - final : 이 멤버 함수를 override하지 않으려고할때 붙일 수 있다. 상속받은 클래스가 이 멤버 함수를 override하면 에러를 뱉는다

## 가상함수 작동 방식

- 가상함수를 사용시 virtual function table이 생성된다.
- 가상함수가 들어간 클래스의 각 인스턴스는 가상함수테이블 (vtable)을 가르키는 포인터(virtual function pointer vptr : 4byte)를 가짐
- virtual 선언만하고 정의를 빼려면 virtual type function() = 0;

S



# 메모리 구조

Name	Value	Type
mae	{a=0 }	sampleinheritance
sample	{b=0 }	sample
_vfptr	0x00d83b34 (Project4.exe\void(*) sample::vftable[4])[0] {0x00d71c53 (Project4.exe\sampleinheritance::f(void)), ...}	void *
[0]	0x00d71c53 (Project4.exe\sampleinheritance::f(void))	void *
[1]	0x00d71c76 (Project4.exe\sampleinheritance::g(void))	void *
[2]	0x00d71c71 (Project4.exe\sample::j(void))	void *
b	0	int
a	0	int
sam	{b=0 }	sample
_vfptr	0x00d83b34 (Project4.exe\void(*) sample::vftable[4])[0] {0x00d71c58 (Project4.exe\sample::f(void)), ...}	void *
[0]	0x00d71c58 (Project4.exe\sample::f(void))	void *
[1]	0x00d71c6c (Project4.exe\sample::g(void))	void *
[2]	0x00d71c71 (Project4.exe\sample::j(void))	void *
b	0	int

Figure: vtbl

## up casting

- up casting : 부모 클래스의 자료형으로 자식 클래스를 가르키는것. 이때 소멸자를 가상함수로 선언하여야한다.
- down casting : up casting 된 부모 클래스의 포인터를 다시 자식 클래스 자료형 포인터로 가르키게 하는것.

```
1      super* supPtr = new sub(); // up casting
2      sub* subPtr1 = (sub*)supPtr; // down casting1
3      sub* subPtr2 = dynamic_cast<sub*>(supPtr); //
down casting2
```

# 찾아볼것

## casting

- C sytle
- *static\_cast* <> () : [참고](#)
- *dynamic\_cast* <> ()
- *reinterpret\_cast* <> () [참고](#)
- *const\_cast* <> () :

## *smart\_ptr*(C++11 )

- *unique\_ptr* <> ()
- *shared\_ptr* <> ()
- *weak\_ptr* <> ()

## enum class

# 다중 상속(Multiple inheritance)

- 직접 찾아볼것.

# 과제

## 과제 0