

# C++ Programming

## 5<sup>th</sup> Study: Object-Oriented Programming (1/8)

- Class and object
- Access modifier: public, protected, private
- Information hiding, Encapsulation

C++ Korea 옥찬호 (utilForever@gmail.com)



# Class and object

User-defined type such as structure

# Class and object

- 객체 지향 프로그래밍이란, 프로그램을 단순히 데이터와 처리 방법으로 나누지 않고, 수많은 '객체'라는 기본 단위로 나눈 뒤 이 객체들의 상호작용으로 서술하는 방식을 말함
- 쉽게 설명해보자면...
  - 객체 지향 프로그래밍은 캐릭터 중심의 소설 쓰기
  - 절차 지향 프로그래밍이 '시나리오를 구상하고, 시나리오에 맞는 캐릭터를 창조한다'는 전통적 의미의 소설쓰기에 해당한다면, 객체 지향 프로그래밍은 '캐릭터를 창조하고, 그 캐릭터에 맞는 시나리오를 구상한다'는 라이트 노벨식 소설쓰기에 해당함

# Class and object

- 문제 분석의 포커스 : “누가 무엇으로 어떻게”
  - 누가 : 객체
  - 무엇으로 : 객체 연산에 필요한 자료 = 멤버 변수
  - 어떻게 : 객체 연산 = 멤버 함수
- 객체(Object)
  - 객체지향 프로그래밍에서는 데이터와 기능을 하나로 묶어서 다룰 수 있는 객체라는 개념을 도입
  - 프로그램은 처리하는 절차보다도 동작되는 자료에 중점을 둔 객체, 그리고 객체 간의 상호관계로 표현

# Class and object

- 객체는 멤버 변수(특성, Attribute)와 이를 대상으로 처리하는 동작인 멤버 함수(메서드, Method)를 하나로 묶어 만든 요소로 프로그램을 구성하는 실체
- 구조적 프로그래밍에서는 변수와 함수가 합쳐져 프로그램을 만들지만, 객체 지향 프로그래밍에서는 멤버 변수와 멤버 함수가 합쳐져 하나의 객체를 만들고, 객체와 객체가 합쳐져 하나의 프로그램을 만듦

# Class and object

- C에서 구조체는, 서로 다른 타입의 변수들을 저장할 수 있는 사용자 정의 타입 → 하지만 함수를 저장할 수는 없었음
- 따라서 구조체를 매개변수로 받는 함수를 따로 만들어 처리
- C++에서는 멤버 변수랑 멤버 함수를 함께 담을 수 있는 클래스(Class)라는 개념을 도입
- 클래스를 통해 만들어지는 변수를 객체(Object)라고 부름
- 마치 클래스와 객체는 ‘붕어빵 틀’과 ‘붕어빵’ 관계와 같음!

# Class and object

- 클래스 선언 방법
  - `class 클래스 이름 { 멤버 변수; 멤버 함수; ... };`
  - `struct 클래스 이름 { 멤버 변수; 멤버 함수; ... };`
- C++에서 `class`  $\doteq$  `struct`
  - 즉, 구조체에서도 함수를 선언할 수 있다는 말!
  - 하지만 한 가지 차이점이 존재!, 기본적으로...
    - `class` : 클래스 밖에서 멤버 변수나 멤버 함수에 접근 불가능
    - `struct` : 클래스 밖에서 멤버 변수나 멤버 함수에 접근 가능

# Class and object

- 클래스의 멤버 함수
  - 함수의 본체 내용은 클래스 내부나 클래스 외부에서 구현
  - 클래스 내부에서 함수의 본체가 구현된 멤버 함수들은 모두 인라인 함수(Inline Function)로 정의
    - 인라인 함수 : 프로그램의 실행 속도를 높이기 위해 추가된 기능, C언어의 매크로 함수와 비슷하지만 훨씬 쓰기 간편하며 효율적으로 동작함  
자세한 내용은 <http://en.cppreference.com/w/cpp/language/inline> 참조
  - 함수의 본체가 긴 경우에 클래스의 멤버 함수를 클래스 내부가 아닌 클래스 외부에서 정의하는 것이 편리함
    - 클래스 바깥에서 리턴 타입 클래스 이름::함수 이름(매개변수)로 정의 가능  
예 : `void Elephant::wash() { ... }`



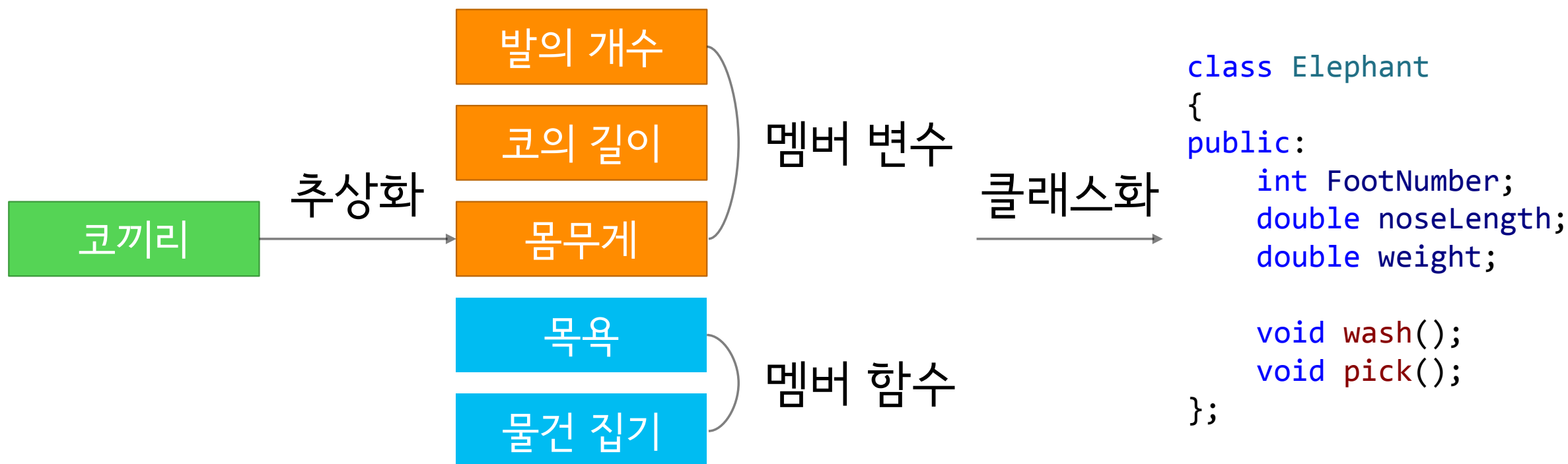
# Class and object

- 사물의 관찰 이후의 데이터 추상화
  - 현실 세계의 사물을 데이터적인 측면과 기능적인 측면을 통해서 정의
- 예 : 코끼리를 자료형으로 정의
  - 발이 4개 → 데이터 → 변수
  - 코의 길이가 5m 내외 → 데이터 → 변수
  - 몸무게는 1톤 이상 → 데이터 → 변수
  - 코를 이용해서 목욕을 함 → 기능 → 함수
  - 코를 이용해서 물건을 집기도 함 → 기능 → 함수

## 프로그램

# Class and object

- 데이터 추상화 이후의 클래스화
  - 추상화된 데이터를 가지고 사용자 정의 자료형을 정의



# Class and object

- 선언된 클래스를 사용해 객체를 생성
  - 클래스화 이후의 인스턴스화(Instantiation)

```
class Elephant
```

```
{
```

```
public:
```

```
    int FootNumber;
```

```
    double noseLength;
```

```
    double weight;
```

```
    void wash();
```

```
    void pick();
```

```
};
```

인스턴스화

```
int main()
```

```
{
```

```
    Elephant e = { 4, 4.96, 1.02 };
```

```
    // ...
```

```
}
```

# Class and object: Example

C

```
typedef struct Elephant {
    int FootNumber;
    double noseLength;
    double weight;
} Elephant;

void wash() { printf("Wash!\n"); }
void pick() { printf("Pick!\n"); }

int main()
{
    Elephant e = { 4, 4.96, 1.02 };

    wash(); pick();
}
```

C++

```
class Elephant {
public:
    int FootNumber;
    double noseLength;
    double weight;

    void wash() { cout << "Wash!\n"; }
    void pick() { cout << "Pick!\n"; }
};

int main()
{
    Elephant e = { 4, 4.96, 1.02 };

    e.wash(); e.pick();
}
```

# Class and object: Example

## class

```
class Elephant {  
    int FootNumber;  
    double noseLength;  
    double weight;  
  
    void wash() { cout << "Wash!\n"; }  
    void pick() { cout << "Pick!\n"; }  
};  
  
int main()  
{  
    Elephant e = { 4, 4.96, 1.02 };  
  
    e.wash(); e.pick();  
}
```

## struct

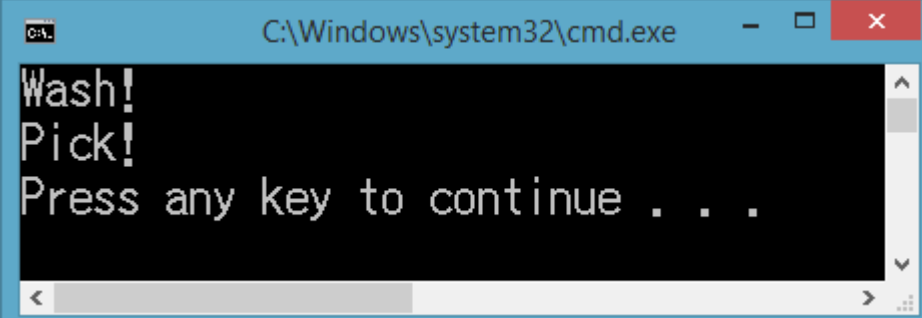
```
struct Elephant {  
    int FootNumber;  
    double noseLength;  
    double weight;  
  
    void wash() { cout << "Wash!\n"; }  
    void pick() { cout << "Pick!\n"; }  
};  
  
int main()  
{  
    Elephant e = { 4, 4.96, 1.02 };  
  
    e.wash(); e.pick();  
}
```

# Class and object: Example

class

	Code	Description
		IntelliSense: too many initializer values
		IntelliSense: function "Elephant::wash" (declared at line 15) is inaccessible
		IntelliSense: function "Elephant::pick" (declared at line 16) is inaccessible
✖	C2440	'initializing': cannot convert from 'braced-init-list' to 'Elephant'
✖	C2248	'Elephant::wash': cannot access private member declared in class 'Elephant'
✖	C2248	'Elephant::pick': cannot access private member declared in class 'Elephant'

struct



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a black background with white text. The text displayed is: "Wash!", "Pick!", and "Press any key to continue . . .". The window includes standard Windows window controls (minimize, maximize, close) in the top right corner and a scrollbar on the right side.

# Class and object: Example

```
class Elephant {  
public:  
    int FootNumber;  
    double noseLength;  
    double weight;  
  
    void wash() { cout << "Wash!\n"; }  
    void pick() { cout << "Pick!\n"; }  
};
```

```
class Elephant {  
public:  
    int FootNumber;  
    double noseLength;  
    double weight;  
  
    void wash();  
    void pick();  
};  
  
void Elephant::wash()  
{ cout << "Wash!\n"; }  
void Elephant::pick()  
{ cout << "Pick!\n"; }
```

# Access modifier: public, protected, private

Keywords that set the accessibility of classes, methods, and other members



# Access modifier

- class와 struct를 사용한 결과가 각각 달랐던 이유?
  - class : 클래스 밖에서 멤버 변수나 멤버 함수에 접근 불가능
  - struct : 클래스 밖에서 멤버 변수나 멤버 함수에 접근 가능
- C++에서는 접근 제한자(Access Modifier)를 통해 멤버 변수나 멤버 함수에 접근할 수 있는 범위를 정함
  - private : 자신의 멤버 함수 외에는 접근할 수 없음
  - protected : 자식 클래스의 멤버 함수로부터의 접근만 허용
  - public : 모든 곳으로부터의 접근을 허용

# Access modifier: Example

```
class Parent
{
private:
    int priv;
protected:
    int prot;
public:
    int pub;
    void AccessAllMembers();
};

void Parent::AccessAllMembers()
{
    priv = 100; // Success
    prot = 100; // Success
    pub = 100;  // Success
}
```

```
class Child : public Parent {
public:
    void AccessParents() {
        int n;
        n = priv; // Fail
        n = prot; // Success
        n = pub;  // Success
    }
};

int main() {
    Parent parent;
    int n;
    n = parent.priv; // Fail
    n = parent.prot; // Fail
    n = parent.pub;  // Success
}
```

# Access modifier: Example

## class

```
class Elephant {  
    [private:]  
        int FootNumber;  
        double noseLength;  
        double weight;  
  
        void wash() { cout << "Wash!\n"; }  
        void pick() { cout << "Pick!\n"; }  
};  
  
int main()  
{  
    Elephant e = { 4, 4.96, 1.02 };  
  
    e.wash(); e.pick();  
}
```

## struct

```
struct Elephant {  
    [public:]  
        int FootNumber;  
        double noseLength;  
        double weight;  
  
        void wash() { cout << "Wash!\n"; }  
        void pick() { cout << "Pick!\n"; }  
};  
  
int main()  
{  
    Elephant e = { 4, 4.96, 1.02 };  
  
    e.wash(); e.pick();  
}
```

# Information hiding, Encapsulation

The process of compartmentalizing the elements of an abstraction that constitute its structure and behavior

# Information hiding, Encapsulation

- 정보 은닉(Information Hiding)
  - 프로그램을 사용하는 사용자가 알아야 하는 것은 프로그램을 사용하는 방법이지, 프로그램의 내부 동작이나 상세 구조가 아님
  - 사용자가 굳이 알 필요가 없는 정보를 숨김으로써, 사용자는 최소한의 정보만으로 프로그램을 쉽게 사용할 수 있어야 함
  - C++에서는 클래스의 정보 은닉 기능을 지원하기 위해 접근 제한자 (private, protected, public) 키워드를 제공
  - 하지만 간접적 접근 경로를 제공해야 함 → 숨길 멤버와 공개할 멤버의 블록을 구성 → 공개된 멤버는 외부에서 자유롭게 읽을 수 있지만, 숨겨진 멤버를 참조하려고 시도하면 오류 발생하도록 처리

# Information hiding, Encapsulation

- 정보 은닉(Information Hiding)
  - 간접적 접근 경로를 제공하는 방법 : 게터(Getter)와 세터(Setter)
    - 게터(Getter) : 값을 읽기 위한 멤버 함수
    - 세터(Setter) : 값을 쓰기 위한 멤버 함수
  - 예를 들어, 멤버 변수 `weight`에 대해서...
    - 게터 : `double getWeight();`
    - 세터 : `void setWeight(double _weight);`
  - 참고하자면 일반적으로...
    - 멤버 변수는 `private`, 멤버 함수는 `public`
    - 하지만 프로그램 설계에 따라 달라질 수 있으니 참고할 것!

# Information hiding, Encapsulation

- 캡슐화(Encapsulation)
  - 관련있는 데이터와 함수를 하나의 단위로 묶는 것
- 캡슐화와 정보 은닉, 과연 똑같은 것일까?

```
class Point
{
public:
    double x;
    double y;
};
```

정보 은닉은 되지 않았지만,  
캡슐화는 되어 있는 코드

```
class Point {
private:
    double x;
    double y;

public:
    double getX();
    double getY();
    void setX(double _x);
    void setY(double _y);
};
```

정보 은닉과 캡슐화  
모두 되어 있는 코드