

3, 4주 2v

C++ STUDY

Week 3
Class & 객체

01

하나,

Class & 객체의 이해



02

두울,

Class 생성



03

세엣,

객체 생성

01

하나,

Class & 객체의 이해

02

두울,

Class 생성

03

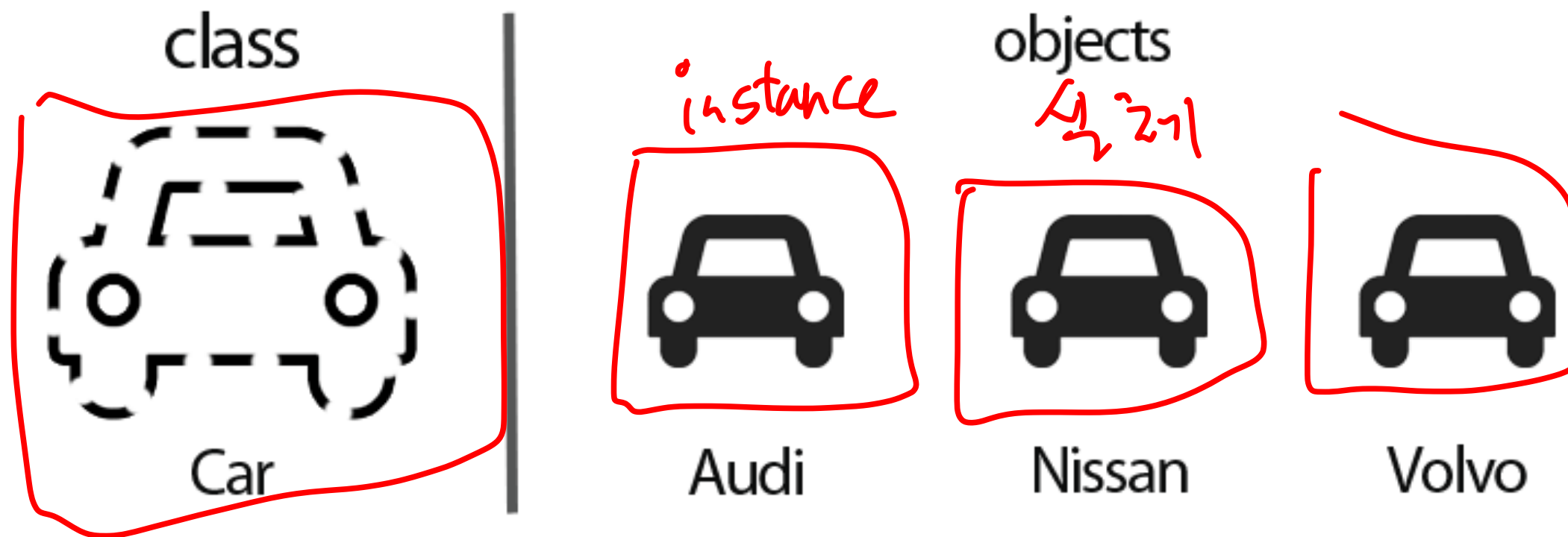
세엣,

객체 생성

Class & 객체

클래스 (class) : 객체를 정의하는 틀

객체 (object) : 클래스라는 틀에서 생겨난 '실체' (instance)



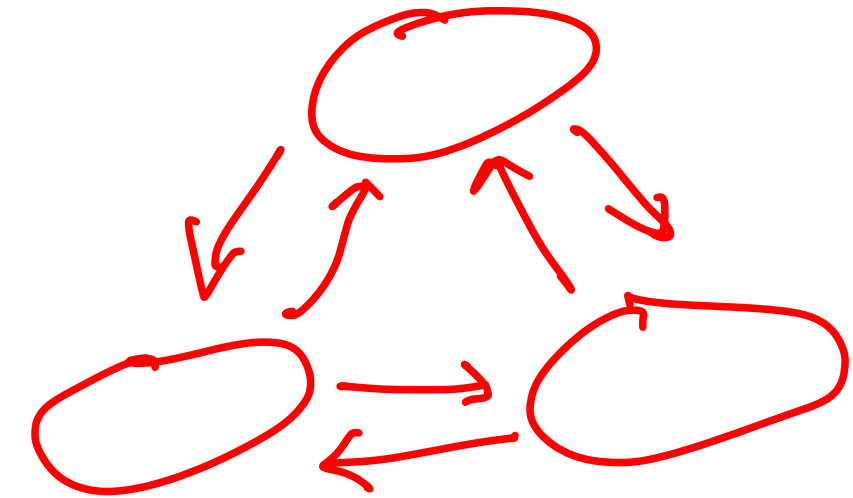
모든 것이 객체다.

컴퓨터, 책, 건물, 사람, 의자, 자동차, 카메라, TV 등등

객체는 캡슐화 된다.

객체의 구성요소들을 캡슐로 싸서 보호하고 볼 수 없게 한다. 은닉성

예) 컴퓨터의 케이스, 사람의 피부



하지만 일부 요소는 공개가 필요하다.

객체들이 서로 정보를 교환하고 통신하기 위해 일부 요소의 공개 노출이 필요하다.

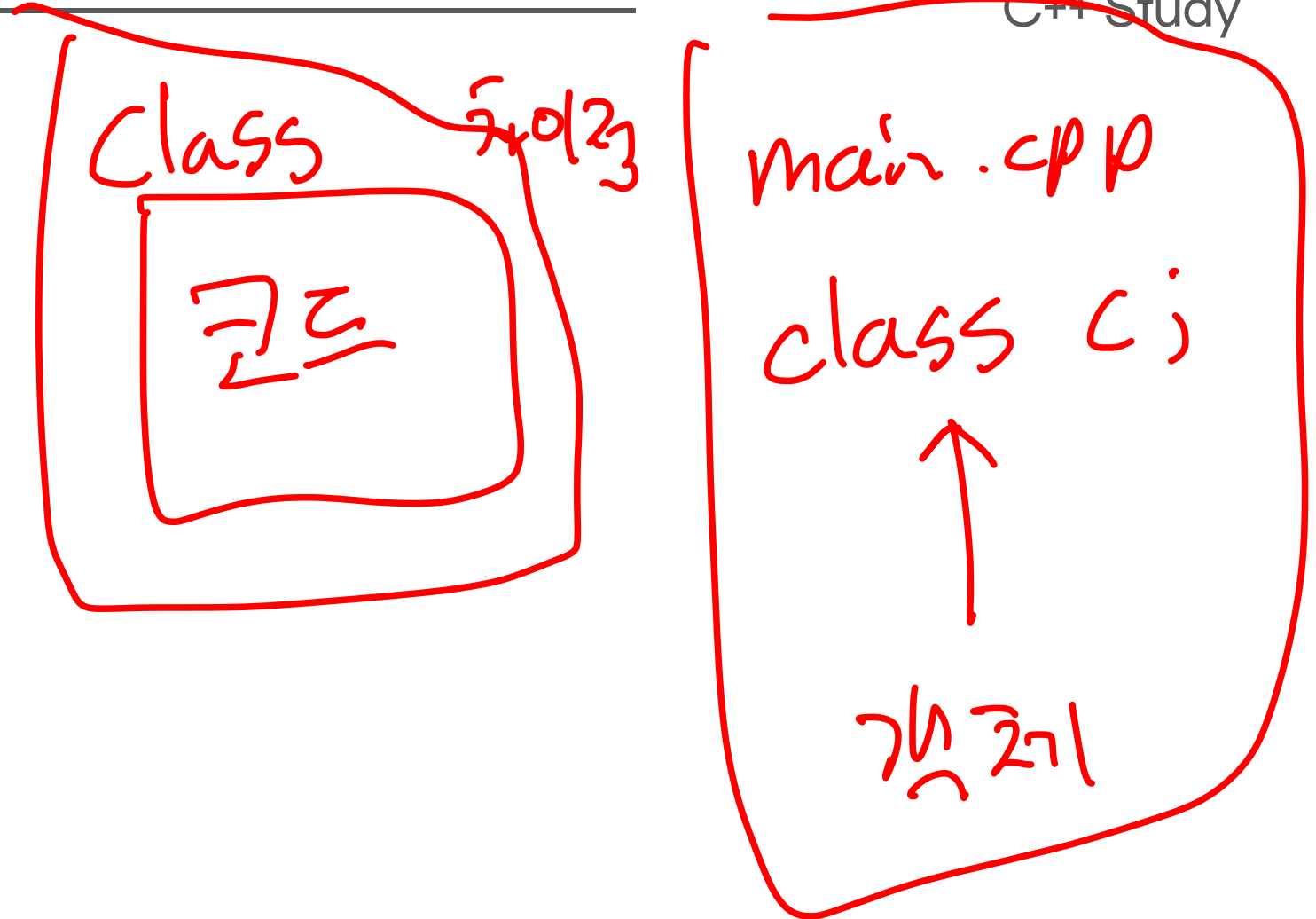
예) 리모컨 - TV, 사람의 뇌 - 눈

객체의 구성

멤버 변수 + 멤버 함수

그래서 객체란?

클래스라는 틀에서 찍어내어, 멤버 변수 메모리와 멤버 함수 코드를 가지고,
프로그램이 실행되는 동안 실존하는 실체 또는 인스턴스(instance)



01

하나,

Class & 객체의 이해

02

두울,

Class 생성

03

세엣,

객체 생성

클래스 만들기

```
class Circle
{
public:
    Circle();      생성자
    ~Circle();     소멸자

    // 인라인 함수
    int getRadius()
    {
        return radius;    멤버함수
    }
    void setRadius(int radius);

private:
    int radius;         멤버변수
};

void Circle::setRadius(int radius)    정의
{
    // this를 사용하여 radius란 이름은 같지만 멤버 변수와 인자를 구분할 수 있다.
    this->radius = radius;
}

// 생성자
Circle::Circle() // : radius(0) <- 멤버 초기화 리스트
{
    radius = 0;
}

// 소멸자
Circle::~~Circle()
{
}
```


클래스 만들기

```
private:  
    int radius;
```

변수 선언 (바로 `radius = 5`, 이런 식으로 초기화 불가능)

```
void setRadius(int radius);
```

함수 원형 선언 (리턴 타입, 매개 변수 리스트등이 모두 선언되어야 함)

```
void Circle::setRadius(int radius) 중의  
{  
    // this를 사용하여 radius란 이름은 같지만 멤버 변수와 인자를 구분할 수 있다.  
    this->radius = radius;  
}
```

함수 구현부 (앞에 클래스이름`::`을 붙혀주어 클래스의 멤버 함수에 접근 후 구현)

`::` -> 범위 지정 연산자 (같은 이름의 함수가 다른 클래스에 존재할 수 있기 때문)

접근 지정자

```
class Circle
{
    public:
        Circle();
        ~Circle();

        // 인라인 함수
        int getRadius()
        {
            return radius;
        }
        void setRadius(int radius);

    private:
        int radius;
};
```

public : 클래스 외부로부터 접근 허용

protected : 자식만 접근 가능

private : 자신 말고 아무도 접근 불가

저번 시간에 배웠던 캡슐화, 은닉성

Get / Set 함수

```
class Circle
{
public:
    Circle();
    ~Circle();

    // 인라인 함수
    int getRadius()
    {
        return radius;
    }
    void setRadius(int radius);

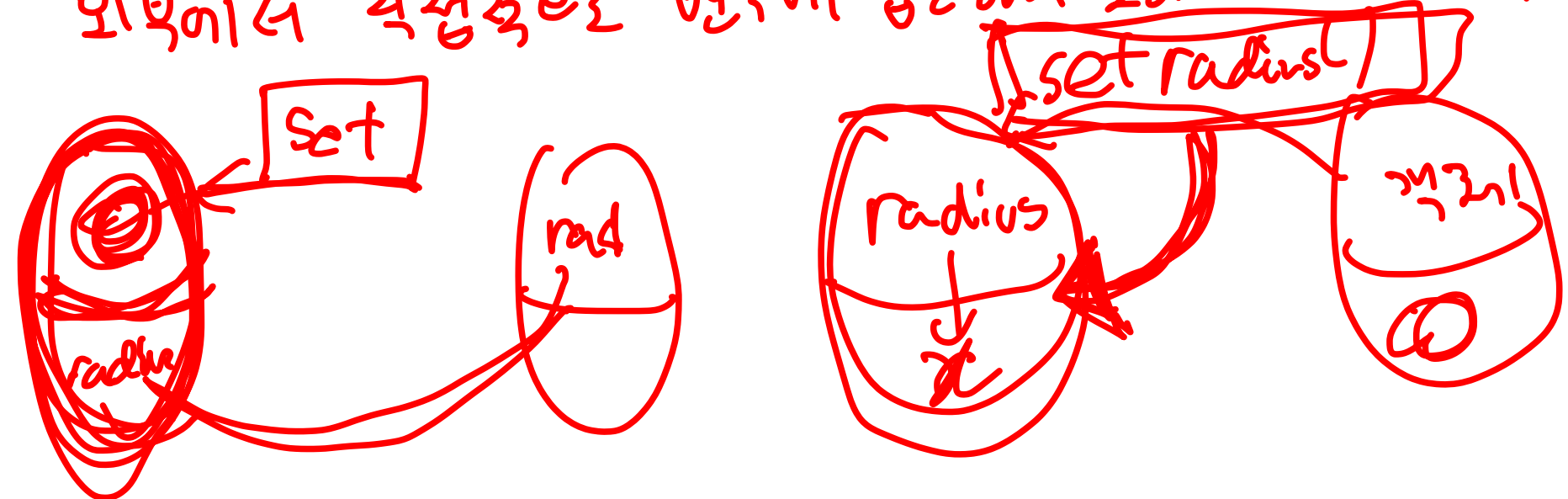
private:
    int radius;
};
```

Get 함수 : private으로 선언된 멤버 변수를 반환

Set 함수 : 외부에서 멤버 변수를 변경

왜 굳이?

외부에서 직접적으로 변수가 접근되어 못하게 하기 위해



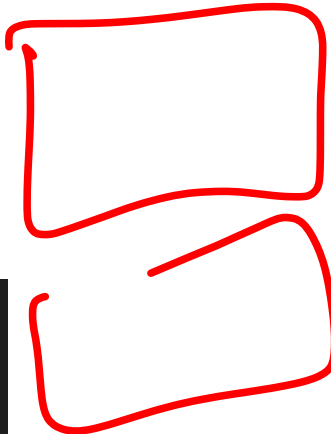
인라인 함수

```
class Circle
{
public:
    Circle();
    ~Circle();

    // 인라인 함수
    int getRadius()
    {
        return radius;
    }
    void setRadius(int radius);

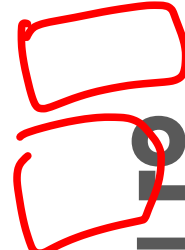
private:
    int radius;
};
```

public



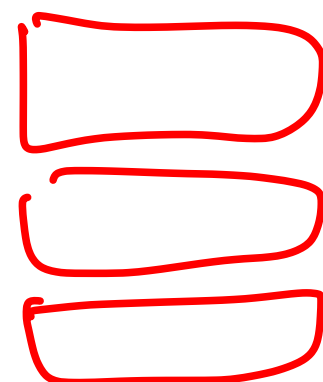
간단한 함수를 클래스내에
구현하여 실행 속도를 향상시킨다.

private



인라인 함수가 많아질 경우
클래스 내의 코드 길이가 늘어난다.

Circle.cpp



this 포인터

radius

int r
int radius

circle
!!
this

```
void Circle::setRadius(int radius)
{
    // this를 사용하여 radius란 이름은 같지만 멤버 변수와 인자를 구분할 수 있다.
    this->radius = radius;
}
```

circle → radius

this는 객체 자신에 대한 포인터이다.

멤버 함수 내에서만 사용된다.

변수 초기화 시 많이 사용된다.

return this를 하게되면 class* 가 반환 된다.

생성자 (Constructor)

생성자라는 특별한 멤버 함수를 통해 객체를 생성할 때 객체를 초기화해준다.

문법

```
Circle();  
Circle(int radius);
```

리턴 타입을 선언하지 않는다.
오직 한번만 실행된다.

구현

```
// 생성자  
Circle::Circle() // : radius(0) <- 멤버 초기화 리스트  
{  
    radius = 0;  
}  
Circle::Circle(int radius) // : radius(radius)  
{  
    this->radius = radius;  
}
```

기본 생성자 (매개변수 X)

매개변수가 있는 생성자

생성자의 목적은 객체가 생성될 때 필요한 초기 작업을 하기 위함이다.

멤버 초기화 리스트

대입이 아닌 선언과 동시에 초기화를 해주는 리스트

```
private:
    int radius;
    const int area;
```

const int형 변수 선언

```
Circle::Circle()
{
    area = 0;
    radius = 1;
}
```

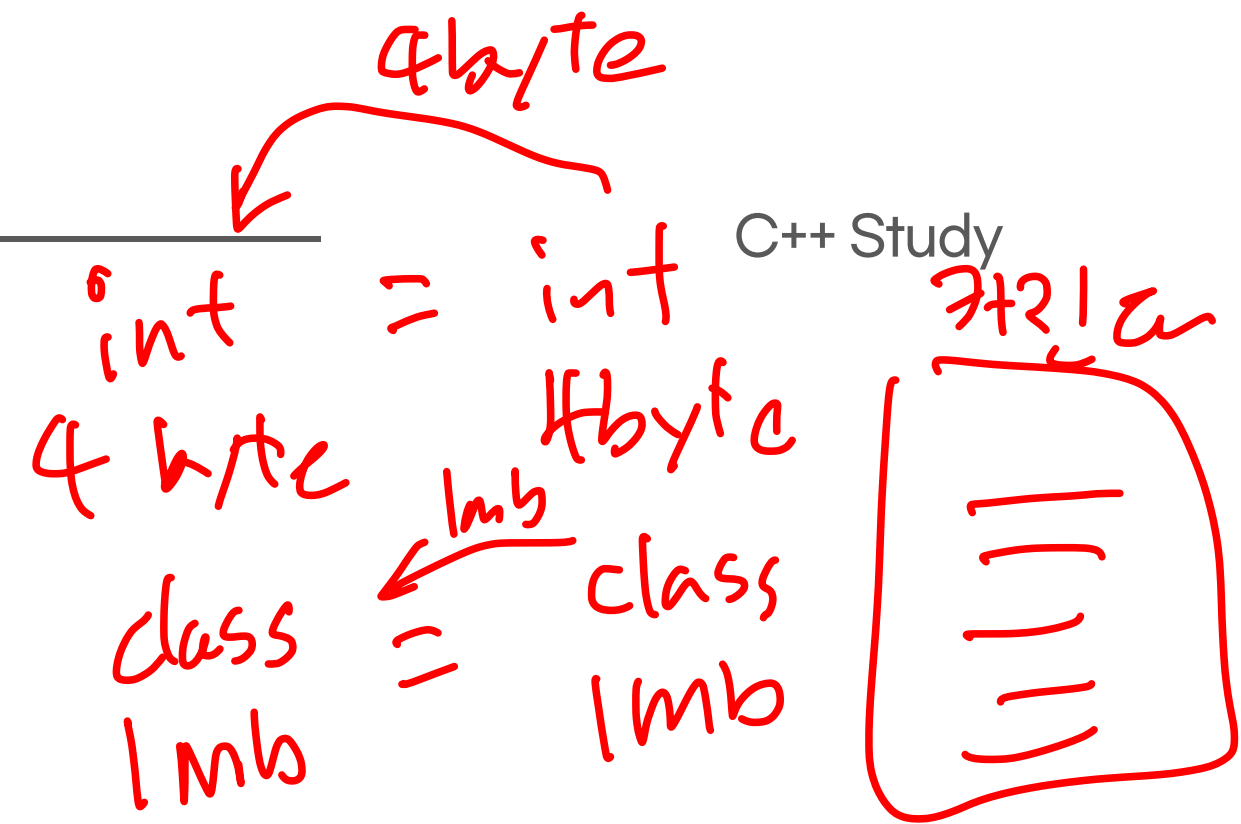
대입으로
초기화 불가

```
Circle::Circle() : radius(1), area(0)
{
}
// 변수(값)
```

```
public:
    Scene()
    : GameObject{ nullptr, "root", "root", nullptr, {1, 1}, {0, 0}, Position::zeros } isCompleted(false),
      enemySpawnTimer(0), enemyFireTimer(0), input(Input::GetInstance()), nextPivotDiff(0), nextSpaceDiff(0),
      boundary(nullptr), helicopter(nullptr), enemy(nullptr), timeBoard(nullptr), time(0.f)
    {
```

나중엔 이렇게 많아진다!

멤버 초기화 리스트로 초기화



일반적인 자료형에선 성능차이가 크지 않지만 사용자가 정의한 타입의 경우엔 대입으로 초기화를 할 시 성능 차이가 더욱 많이날 수 있다. 애용하자!

소멸자 Destructor

객체 소멸시 호출되는 함수

```
// 소멸자 (객체 소멸시 1번 실행)
Circle::~Circle()
{
    cout << "radius : " << radius << " OUT" << endl;
}
```

동작하는 방식 delete 함수 → 메모리 반환

객체가 사라질 때 필요한 마무리 작업을 위함이다. (예 : 메모리 반환, 파일 닫기)

생성자 소멸자 호출 순서 중요!!

01

하나,

Class & 객체의 이해

02

두울,

클래스 생성

03

세엣,

객체 생성

객체 선언

```
Circle* p = new Circle; // p라는 포인터가 새로 생성된 Circle 객체를 가리킴
p->getArea(); // (*). = ->

delete p;

Circle circleArray[3]; // 하나의 배열에 3개의 Circle 객체 생성

circleArray[0].setRadius(1);
circleArray[1].setRadius(2);
circleArray[2].setRadius(3);

// 배열 활용
for (int i = 0; i < 3; i++)
{
    cout << i << "번째 원의 면적 : " << circleArray[i].getArea() << endl;
}

Circle* ptr; // 포인터 선언
ptr = circleArray; // p가 Circle 배열의 시작 주소를 가리킴

for (int i = 0; i < 3; i++)
{
    // 1.
    cout << i << "번째 원의 면적 : " << ptr[i].getArea() << endl; // ptr로 접근가능
    // 2.
    cout << i << "번째 원의 면적 : " << ptr->getArea() << endl; // ptr로 접근가능
    ptr++;
}
```

객체 포인터 선언

```
Circle* p = new Circle; // p라는 포인터가 새로 생성된 Circle 객체를 가리킴
p->getArea(); // (*p). = p->

delete p;
```

() 있어도 되고 없어도 됨

같은 자료형의 p라는 포인터를 선언하여 객체를 새로 생성하고,
p라는 포인터가 그 객체를 가리키게 함. *new*

동적 할당을 했으면 반드시 반환을 시켜줘야 한다.

**p*.getArea() == p->getArea()

**p*는 값이기 때문에 (첫 시간에 엄청나게 강조한 부분)
circle.

객체 배열 선언

기본 생성자와

```
Circle circleArray[3]; // 하나의 배열에 3개의 Circle 객체 생성

circleArray[0].setRadius(1);
circleArray[1].setRadius(2);
circleArray[2].setRadius(3);

// 배열 활용
for (int i = 0; i < 3; i++)
{
    cout << i << "번째 원의 면적 : " << circleArray[i].getArea() << endl;
}
```

같은 자료형의 객체들을 배열로 선언하여 생성

*객체 배열로 선언할 경우 반드시 기본 생성자 호출
(Circle circleArray[3](10) 이런거 불가능)

```
Circle newCircleArray[3] = { Circle(10), Circle(20), Circle() };
```

이 방법을 쓸 수 있다!

배열로 만들 경우 한꺼번에 처리하기 쉬워진다.

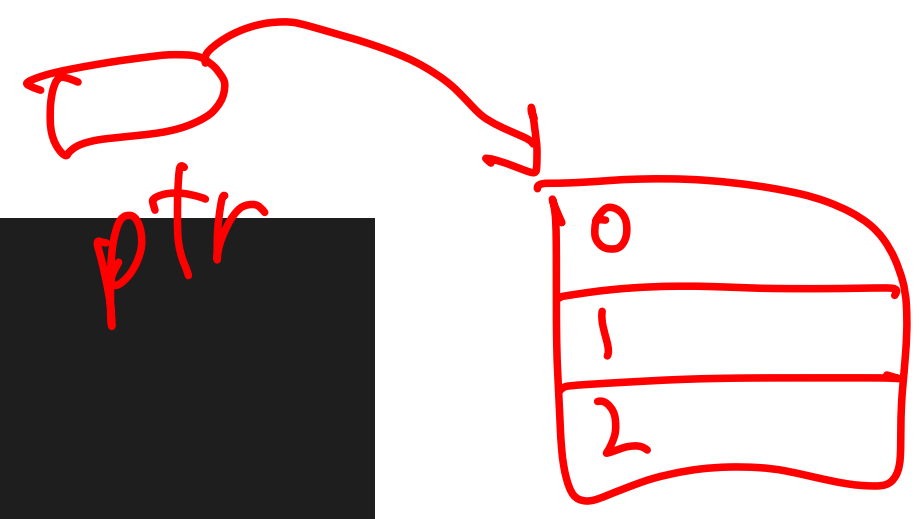
객체 포인터 배열 선언

```
Circle* ptr; // 포인터 선언
ptr = circleArray; // p가 Circle 배열의 시작 주소를 가리킴

for (int i = 0; i < 3; i++)
{
    // 1.
    cout << i << "번째 원의 면적 : " << ptr[i].getArea() << endl; // ptr로 접근가능
    // 2.
    cout << i << "번째 원의 면적 : " << (*ptr).getArea() << endl; // ptr로 접근가능
    ptr++;
}
```

Handwritten annotations in red:

- A bracket under `Circle*` in the first line.
- A bracket under `ptr` in the first line, with an arrow pointing to a diagram of an array.
- A bracket under `ptr` in the second line.
- A bracket under `ptr[i]` in the third line, with a note "배열 [i]" and "X X" next to it.
- A bracket under `(*ptr)` in the fourth line, with a note "ptr->" and "ptr++" next to it.
- A note "ptr도" next to `ptr++;` in the fifth line.



같은 자료형의 포인터 변수가 배열의 시작 주소를 가리키게 한다.

배열의 이름 대신 ptr변수로 사용이 가능하다.

ptr++로 다음 원소로 이동 가능하다.

동적할당

포인터 초기화

자료형* 포인터변수 = new 자료형();
delete 포인터 변수;

```
Circle* cir = new Circle(); // 클래스 동적 생성
cout << cir->getArea() << endl;
delete cir; // 객체 반환
cout << p << endl; // 반환하여도 포인터는 그대로 가리키고 있다.
p = 0; // 포인터 초기화
cout << p << endl; // 확인
```

```
Circle* cirs = new Circle[3]; // 클래스 배열 동적 생성
delete[] cirs; // 객체 배열 반환
cirs = 0; // 포인터 초기화
```

delete[] cirs

확인
11VS11
포인터
5VS5

프로그램

11VS11
or
5VS5

배열 11 5X

5VS5

5VS5

11VS11

11VS11

11VS11

11VS11

전역 변수
Static 변수

데이터 영역

0x3D...
프로그램 할당
Circle

힙 영역

지역 변수
매개 변수
cir

스택 영역

delete

NULL

과제

1. Circle 클래스를 구현하고 원의 갯수를 사용자에게 입력받고 각각 원의 반지름을 사용자에게 입력받은 뒤 면적이 100보다 큰 원은 몇개인지 출력
입력된 원의 개수만큼 동적으로 배열을 할당 받아야한다.

출력 예)

원의 개수 : 3

원 1의 반지름 : 10

원 2의 반지름 : 15

원 3의 반지름 : 2

면적이 100보다 큰 원은 2개 입니다.

심화과제

n명이 하는 한글 끝말잇기 게임을 만들자. 선수의 수를 입력받고
각 선수들의 이름을 입력받아 게임을 시작한다.

출력 예) 끝말잇기 게임을 시작합니다. 게임에 참가하는 인원은 몇명입니까? 2

참가자의 이름을 입력하세요 >> 방찬웅

참가자의 이름을 입력하세요 >> 홍길동

시작하는 단어는 아버지입니다.

방찬웅>> 지상

홍길동>> 상수도

방찬웅>> 토마토

방찬웅이 졌습니다.

WordGame Class) 소동
Player Class
main - 게임 시작

문자열은 string class 객체로 만들어 string class 안에서 제공해주는
메소드를 활용하여 비교. (파이썬 리스트같은 클래스로 생각하면 아주 쉽다)

Q&A 및 토크

감사합니다!