# Day03 (H1)

20150817

Variable Declration
Function Parameter
Method Function
Static Function
Creating Objects

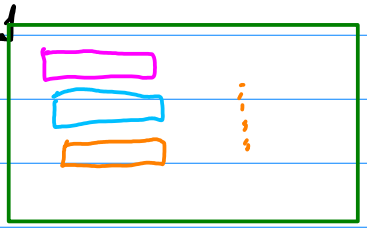
# * 변수 (Variable)

| int | a | ; |
|---|---|---|
| type 형 | 변수이름 | |

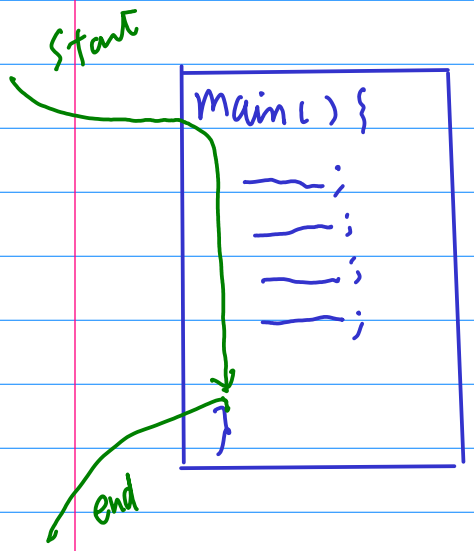
| class (종류,형) | 변수이름 | ; |
|---|---|---|
| myclass | c1 | ; |

c1은 객체를 가리키는 참조 변수

참조



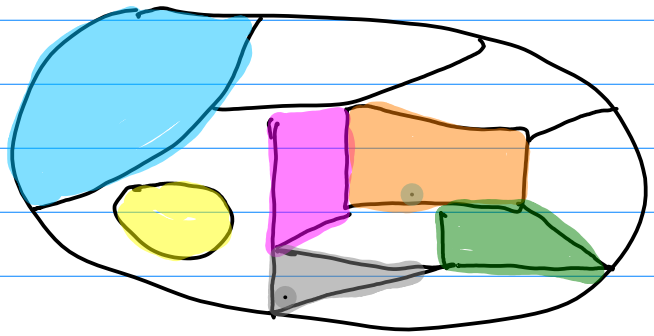
객체 Object




divide-and-conquer

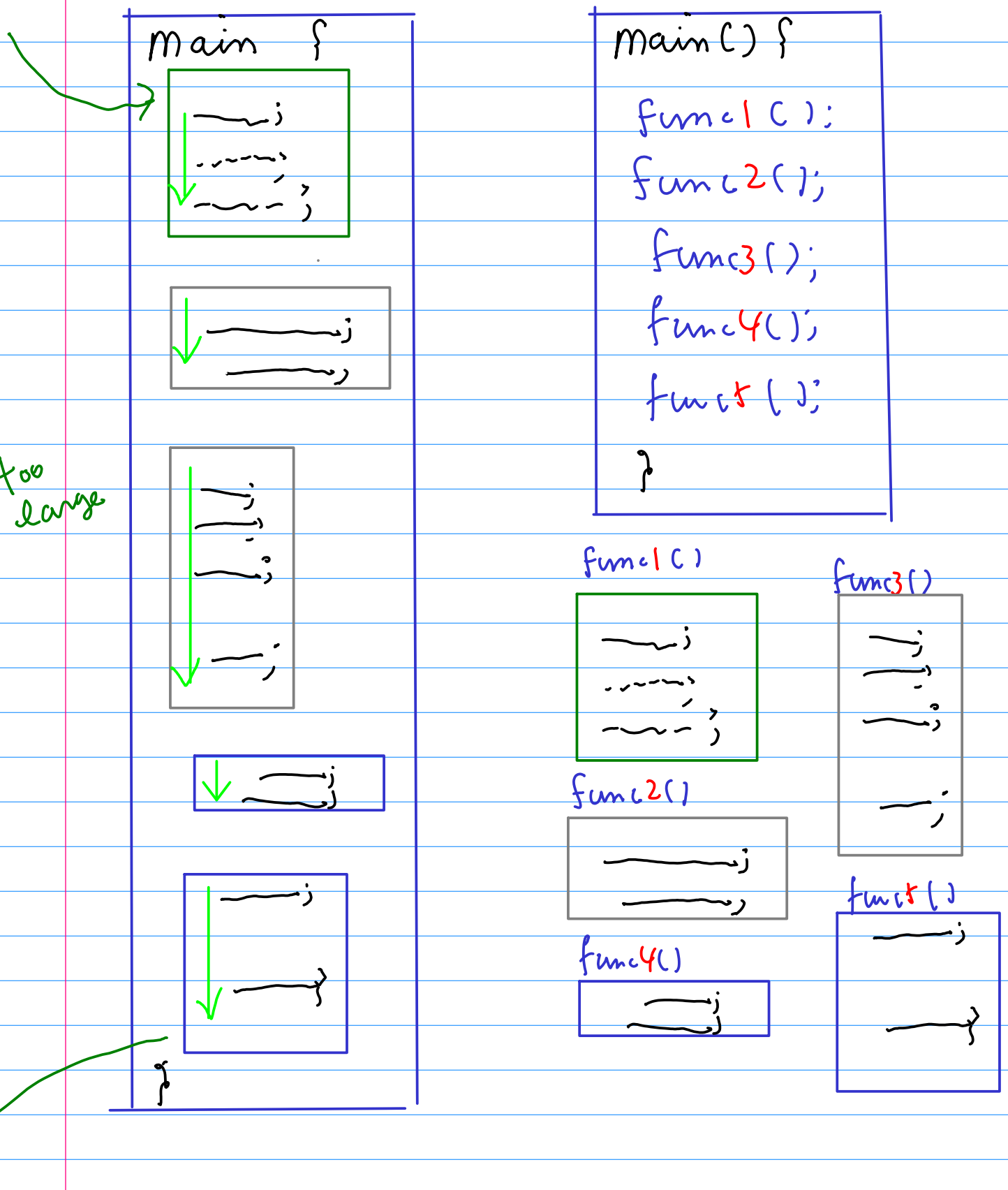
① function

② class



divide by { ① function ⋯> C

② class ⋯> C++, Java

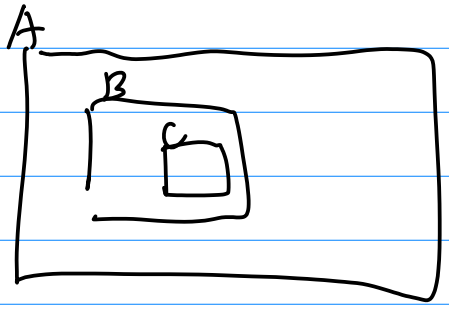

main {
too large
main() {
func1();
func2();
func3();
func4();
func5();
}
func1()
func2()
func3()
func4()
func5()

C:\A\B\C 계층적, 구조적

/home/young/A/B/C

Structure

A. B. C.



System.out . println ("Hello");

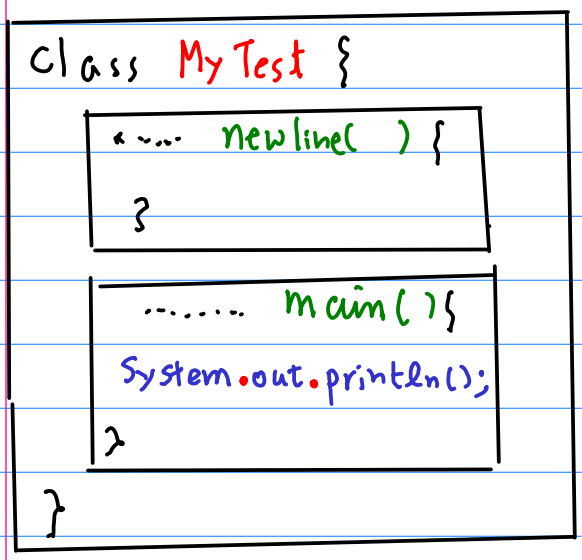
상위계층 . 함수이름 . 입력

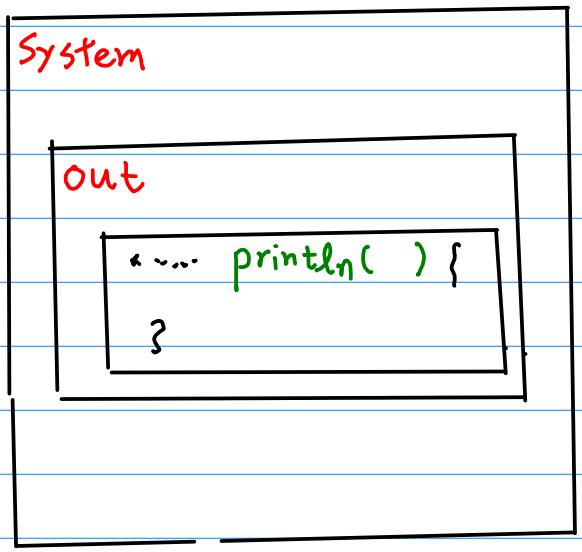
newLine ();

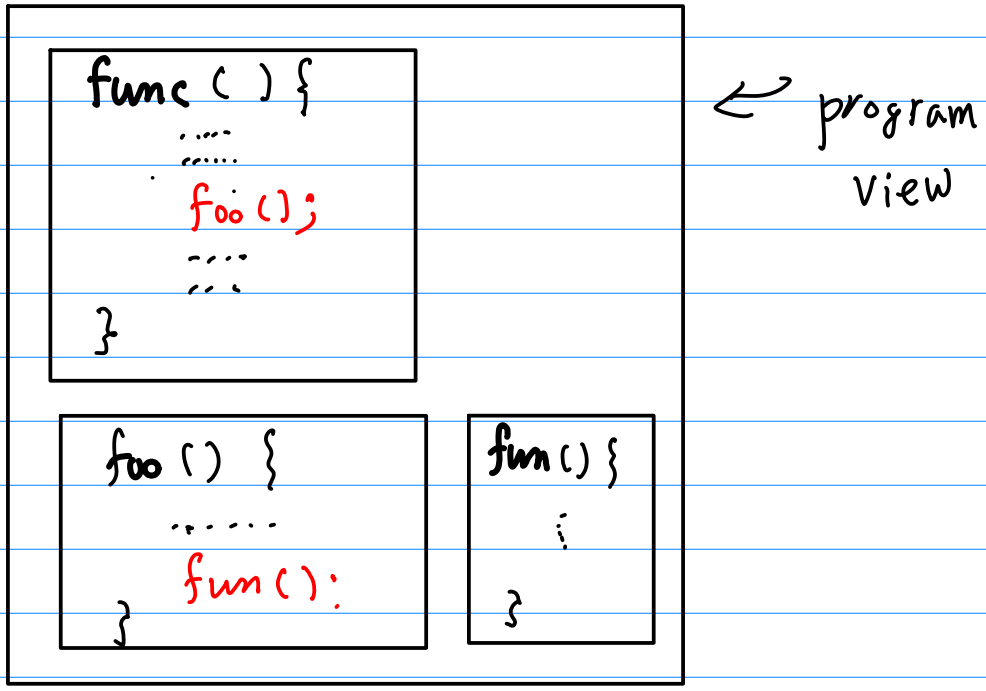
입력이없음 . 함수이름

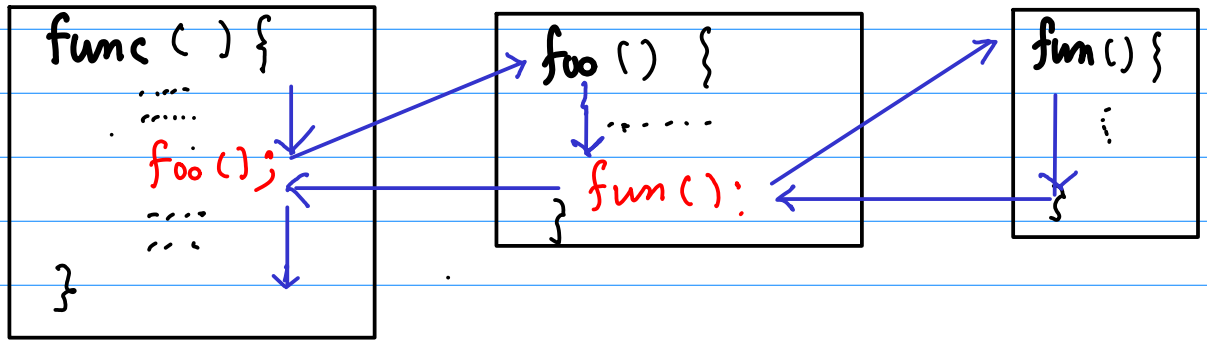
같은 class 내의 함수이므로 . 없다.

(한 class의 member 함수들)






실행 순서

# Method 함수 & Static 함수

Class 이름. 객체 이름. ..... 객체를 가리키는 참조 변수
↓
pointer 와 같은

MyClass C1 ;

모두 같은 class 안에서 정의된 함수

★ 객체 이름. 함수이름( ) ----> method 함수

Class 이름. 함수이름( ) ----> Static 함수

★ C1. 함수이름( ) ----> method 함수

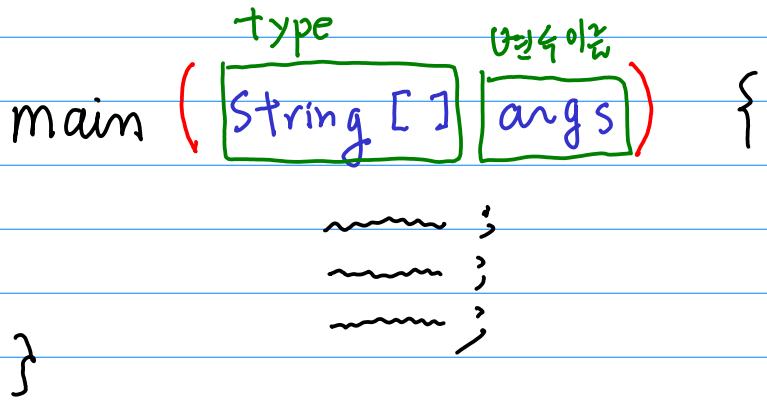
MyClass. 함수이름( ) ----> Static 함수

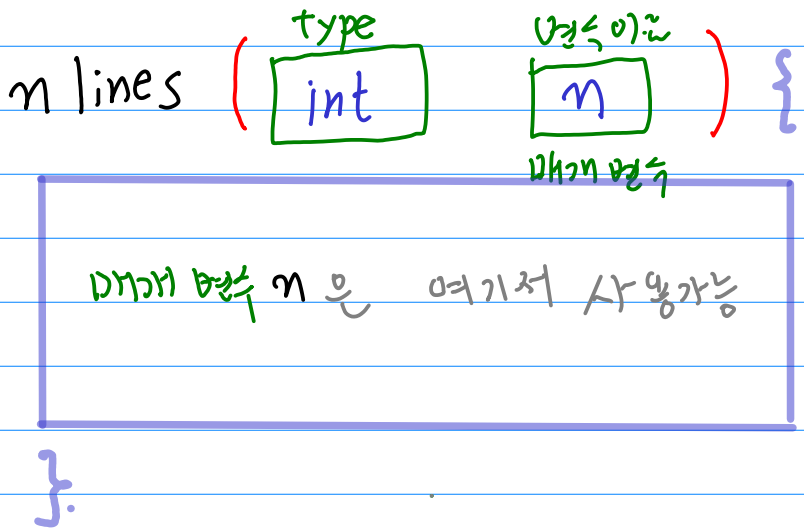
★ C1. 함수이름( ) C1이 가리키는 객체의 member data 사용가능

C2. 함수이름( ) C2가 가리키는 객체의 member data 사용가능

MyClass. 함수이름( ) 객체와 상관없음
member data 사용불가

# parameter (매개변수)



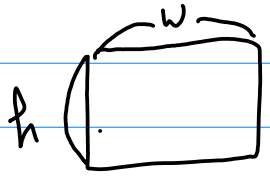

```
main (      ) {
        :
    nlines ( 20 );
        :
}
```

main 함수에서
함수 nlines를 call 할때
입력으로 (20)을 넣어줌

nlines 함수는 입력으로
받은 20을 매개변수 (n)에
대입하여 사용

# * Class

Rectangle

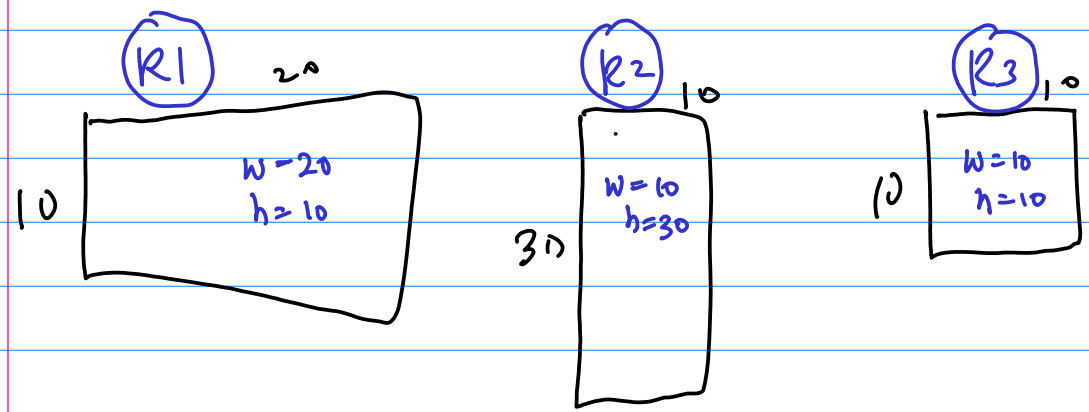



width
height

area = w x h

```
Class Rect {
    int w;
    int h;
}
```

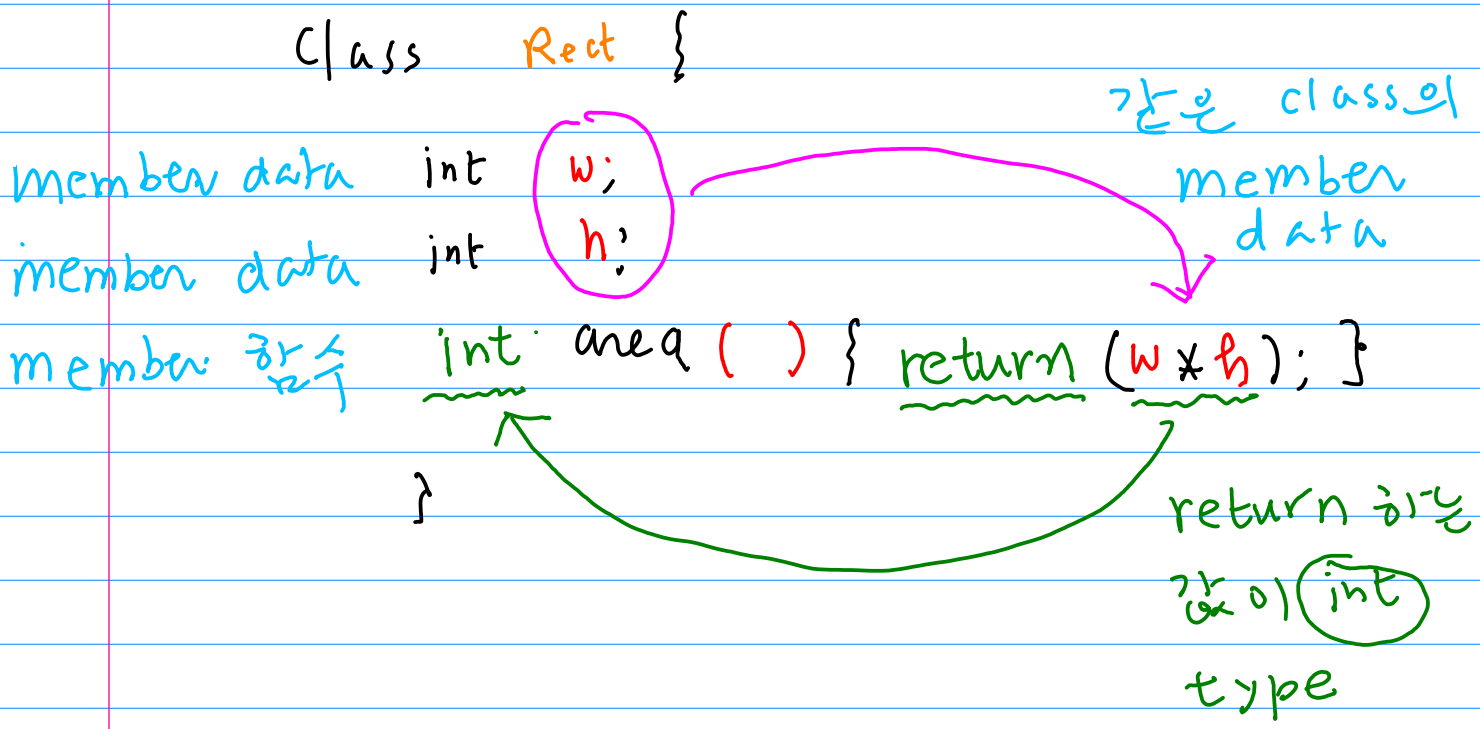
객체 생성 예시




area = w x 20 = 200

area = 30 x 10
= 300

area = 10 x 10
= 100




{ field
  method

```
class Rect
  ...
  int area ( ) {
      return (w * h);
  }
  int area (int x, int y) {
      return (x * y);
  }
  void display ( ) {
      area ( ) ;
  }
```

같은 class의 member data는 (w, h)
parameter로 (x, y) 받을 필요 없이
직접 method 내부에서 사용 가능

같은 class의 member 함수로
⊙ 없이 호출 가능

```
class Rect
  ...
  int area ( ) {
      return (w * h);
  }
  int area (int x, int y) {
      return (x * y);
  }
  void display ( ) {
      area ( ) ;
  }
```

```
class RectTest

  main ( )

  Rect c1 = new Rect();
  c1.area ( );
  c1.area (10, 20);
```

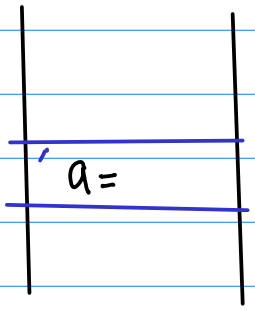
다른 class에서 area()를 호출하기 위해서는

① 객체 생성 new

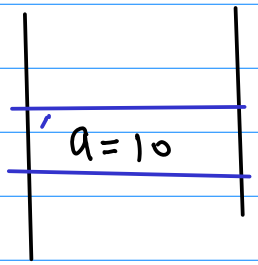
② 객체 참조 변수로 호출

# Creating Objects (객체 생성)

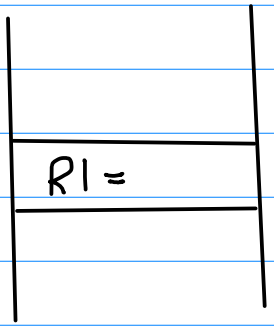
```
int a;
```




```
a = 10;
```

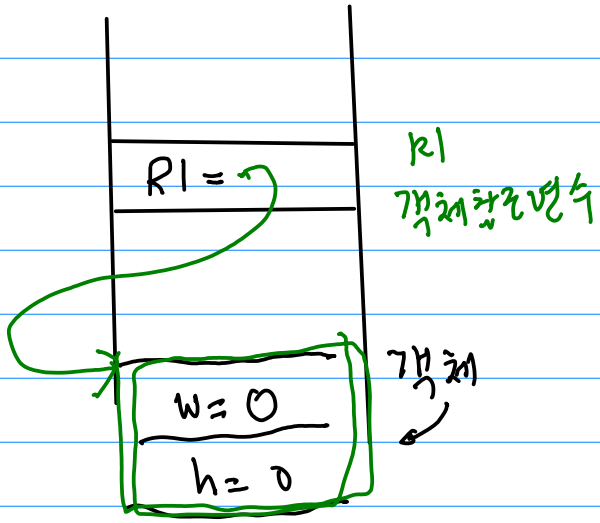


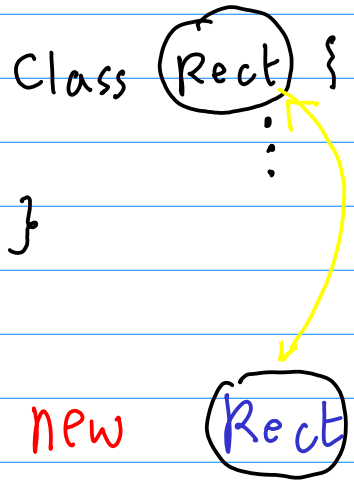

```
Rect R1
```




```
R1 = new Rect();
```

Class 이름과
똑같은 함수
(= 생성자 함수)






함수 call

Class 명과 같은 함수 call

⇒ 생성자 함수 call

⇒ 객체가 memory에 생성 될때 처음에 1번만 자동적으로 불러지는 함수

초기화 (변수)

```
int a;
a = 10;
```

⇒

```
int a = 10;
```

```
Rect R1;
R1 = new Rect();
```

⇒

```
Rect R1 = new Rect();
```

# Concatenation

System.out.println( "rl.w =" + rl.w )

"rl.w =" : 문자열

rl.w : 숫자형

문자열 + 숫자형 → 문자열

rl.w = 2