

# Day03 (H1)

20150817

Variable Declration  
Function Parameter  
Method Function  
Static Function  
Creating Objects

Copyright (c) 2015 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

# \* 변수 (Variable)

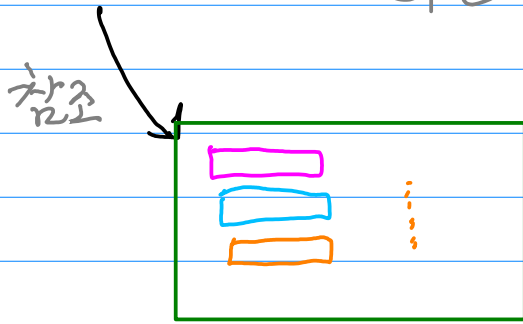
int      a ;

type 형      변수 이름

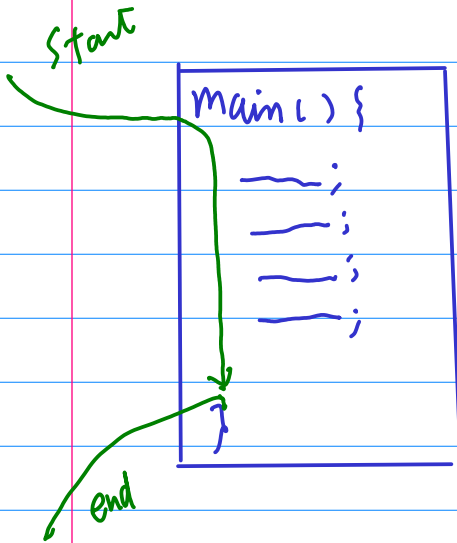
class (종류, 형)      변수 이름 ;

myclass      c1 ;

c1은 객체를 가리키는  
참조 변수



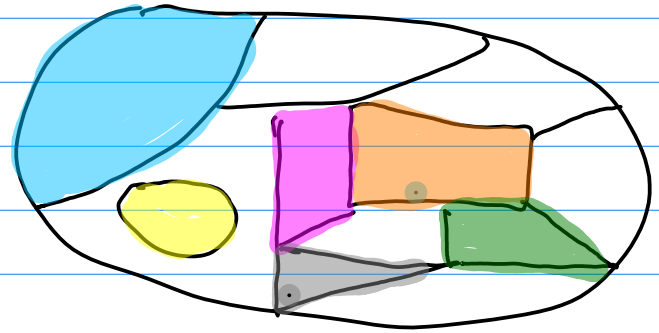
객체 object



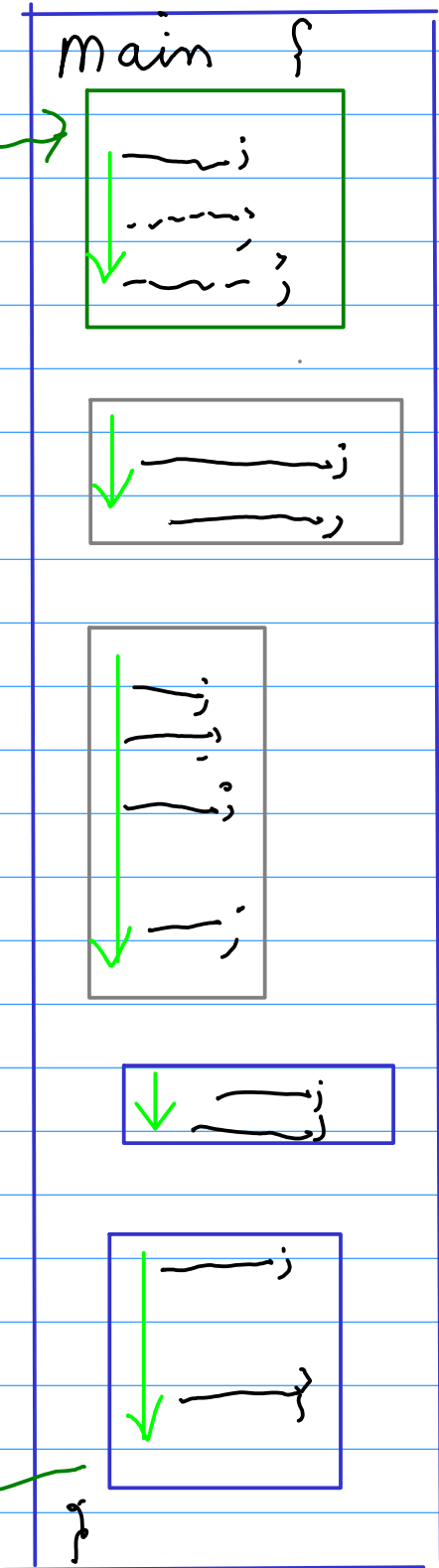
divide - and - conquer

① function

② class

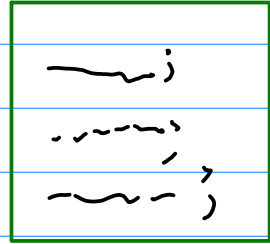


divide by { ① function → C  
② class → C++, Java

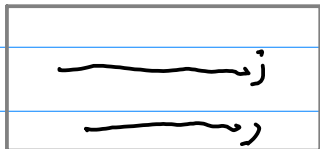


```
main() {
  func1();
  func2();
  func3();
  func4();
  func5();
}
```

func1()



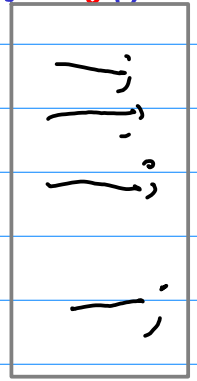
func2()



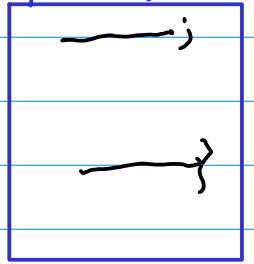
func4()



func3()



func5()



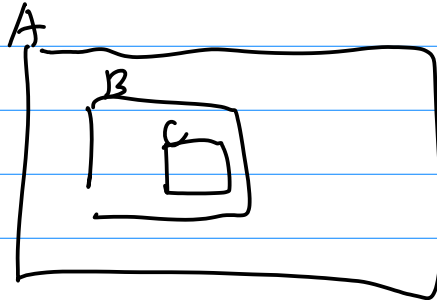
C:\A\B\C

계층적, 구조적

/home/young/A/B/C

Structure

A. B. C.



System.out • println ("hello");

상위계층      함수이름      입력

newline ();

동일이어야함

같은 class 내의 함수이름이다.

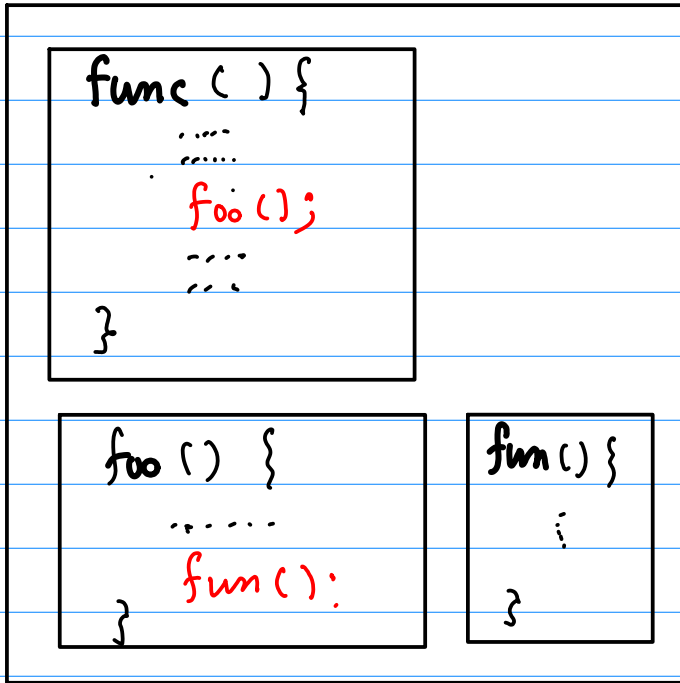
(한 class의 member 함수들)

```

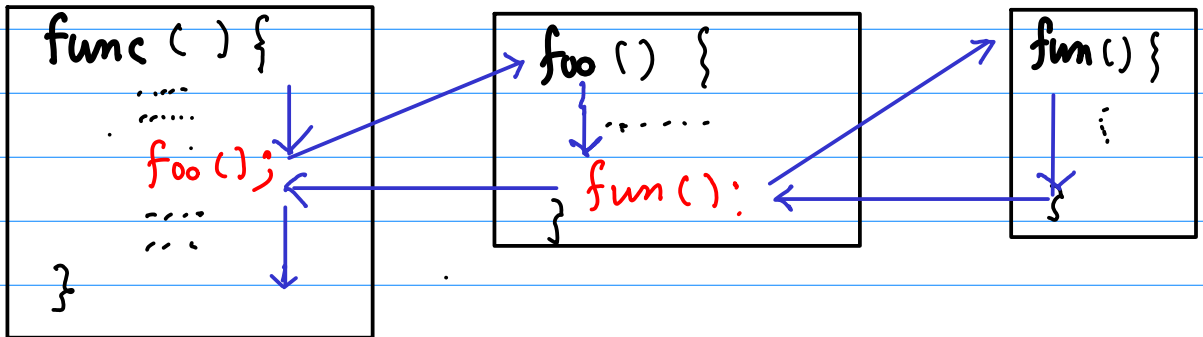
class MyTest {
    * ... newline() {
    }
    ..... main() {
        System.out.println();
    }
}
    
```

```

System
    out
        * ... println() {
        }
    
```



← program view



실행 순서

# Method 함수 & Static 함수

Class 이름

객체 이름

..... 객체를 가리키는 참조 변수

MyClass

C1 ;

↓  
pointer 나 같은

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 ( ) (C1)이 가리키는 객체의 member data 사용가능

(C2) . 함수이름 ( ) (C2)가 가리키는 객체의 member data 사용가능

MyClass . 함수이름 ( ) 객체와 상관없음  
Member data 사용불가



parameter ( 매개변수 )

main ( String [] args ) {  
    ~~~~~ ;  
    ~~~~~ ;  
    ~~~~~ ;  
}

n lines ( int n ) {  
    ~~~~~  
    매개변수 n 은 여기서 사용가능  
}

main (     ) {  
    ⋮  
    nlines ( 20 ) ;  
    ⋮  
}

main 함수에서

함수 nlines 를 call 할 때

입력으로 20 을 넣어줌

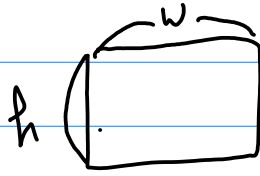
nlines 함수는 입력으로

받은 20 을 매개변수 n 이

대입하여 사용

# \* Class

Rectangle

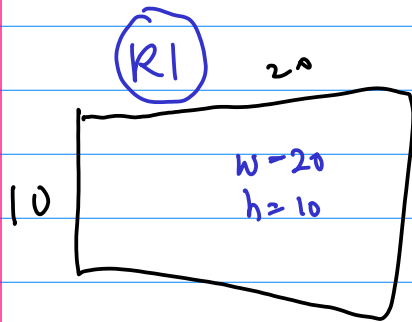


width  
height

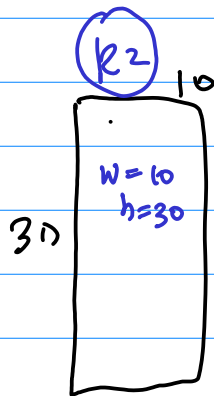
$$\text{area} = w \times h$$

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

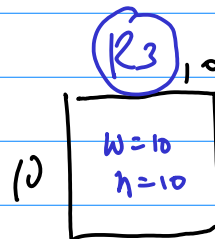
7th mo of 09/1



$$\text{area} = w \times h = \underline{200}$$



$$\text{area} = 30 \times 10 = 300$$



$$\text{area} = 10 \times 10 = 100$$

```
class Rect {
```

member data

```
int w;
```

member data

```
int h;
```

member  $\rightarrow$   $\leftarrow$

```
int area ( ) { return (w * h); }
```

```
}
```

같은 class의 member data

return 하는 값이 int type

{ field  
| method

```
class Rect
```

```
...
```

```
int area ( ) {  
    return (w * h);  
}
```

```
int area (int x, int y) {  
    return (x * y);  
}
```

```
void display ( ) {  
    area ( );  
}
```

같은 class의 member  $w, h$  data는

$(\downarrow, \downarrow)$   
parameter로 받을 필요 없이

각각 method 내에서 사용 가능

같은 class의 member 함수도

⊙ 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 ( );
```

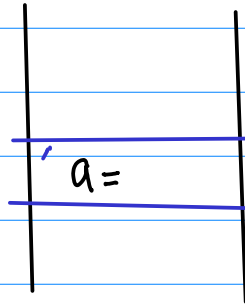
```
c1.area (10, 20);
```

다른 class에서 area()를  
호출하기 위해서

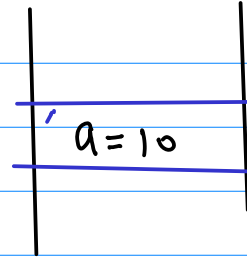
- ① 객체 생성 new
- ② 객체 참조 명사로 호출

# Creating Objects (객체 생성)

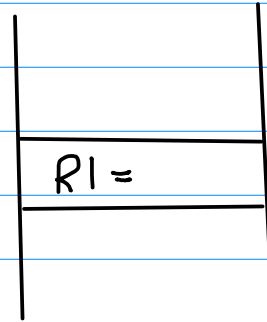
int a;



a = 10;

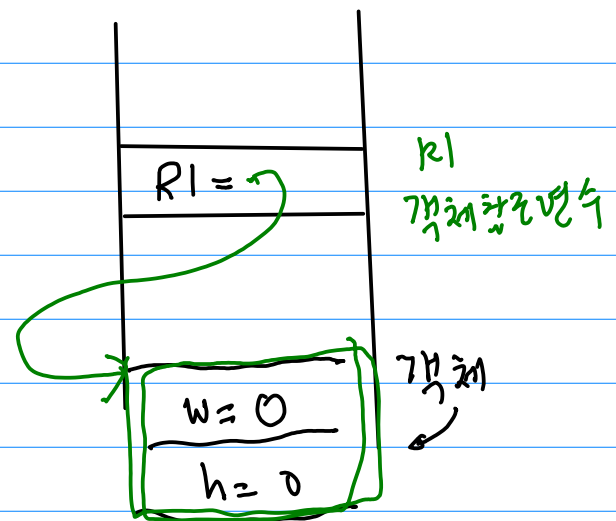


Rect R1



R1 = new Rect();

class 이걸라  
같은 함수  
(= 생성과 함수)



```
Class Rect {  
  ...  
}
```

```
new Rect();
```

함수 call

class 내부라 같은 함수 call

⇒ 생성자 함수 call

⇒ 객체가 memory에 생성 될 때 리턴이 (반만

자동적으로 불러오는 함수

## 초기화 (변수)

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

⇒

```
int a = 10;
```

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

⇒

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



# Concatenation

System.out.println ( "R1.W = " + R1.W )

문자열 + 숫자 21

문자열 + 문자열

R1.W = 21

