

# 메소드와 변수의 범위

06-1.

메소드에 대한 이해와 메소드의 정의

# Main 메소드에 대해서 우리가 아는 것과 모르는 것

메소드의 이름이 `main`이고 중괄호 내 문장들이 **순차적으로 실행**된다는 사실은 안다.

```
public static void main(String[] args) {  
    int num1 = 5;  
    int num2 = 7;  
    System.out.println("5 + 7 = " + (num1 + num2));  
}
```

`public`, `static`, `void` 선언이 의미하는 바는?

메소드 이름이 `main`인 이유는? **자바에서 정한 규칙: 프로그램의 시작은 `main`에서부터!**

`main` 옆에 있는 `(String[] args)`의 의미는?

# 메소드의 정의와 호출

```
public static void main(String[] args) {  
    System.out.println("프로그램의 시작");  
    hiEveryone(12); // 12를 전달하며 hiEveryone 호출  
    hiEveryone(13); // 13을 전달하며 hiEveryone 호출  
    System.out.println("프로그램의 끝");  
}
```

매개변수가 하나인 메소드의 정의

```
public static void hiEveryone(int age) {  
    System.out.println("좋은 아침입니다.");  
    System.out.println("제 나이는 " + age + "세 입니다.");  
}
```



```
C:\JavaStudy>java MethodDef  
프로그램의 시작  
좋은 아침입니다.  
제 나이는 12세 입니다.  
좋은 아침입니다.  
제 나이는 13세 입니다.  
프로그램의 끝  
C:\JavaStudy>
```

# 메소드의 호출

```
public static void main(String[ ] args) {  
    System.out.println("프로그램의 시작");  
    hiEveryone(12);  
    hiEveryone(13);  
    System.out.println("프로그램의 끝");  
}  
  
public static void hiEveryone(int age) {  
    System.out.println("좋은 아침입니다.");  
    System.out.println("제 나이는 .... ");  
}
```

변수 age로 12 전달

①

②

③

# 매개변수 0개, 2개인 메소드

```
public static void main(String[] args) {  
    double myHeight = 175.9;  
    hiEveryone(12, 12.5);  
    hiEveryone(13, myHeight);  
    byEveryone();  
}  
  
public static void hiEveryone(int age, double height) {  
    System.out.println("제 나이는 " + age + "세 입니다.");  
    System.out.println("저의 키는 " + height + "cm 입니다.");  
}  
  
public static void byEveryone() {  
    System.out.println("다음에 뵙겠습니다.");  
}
```

매개변수가 둘인 메소드의 정의

매개변수가 없는 메소드의 정의



```
명령 프롬프트  
C:\JavaStudy>java Method2Param  
제 나이는 12세 입니다.  
저의 키는 12.5cm 입니다.  
제 나이는 13세 입니다.  
저의 키는 175.9cm 입니다.  
다음에 뵙겠습니다.  
C:\JavaStudy>
```

# 값을 반환하는 메소드

void: 값을 반환하지 않음을 의미

```
public static void main(String[] args) {  
    int result;  
    result = adder(4, 5);    // adder가 반환하는 값을 result에 저장  
    System.out.println("4 + 5 = " + result);  
    System.out.println("3.5 x 3.5 = " + square(3.5));  
}
```

```
public static int adder(int num1, int num2) {  
    int addResult = num1 + num2;  
    return addResult;    // addResult의 값을 반환  
}
```

return: 값의 반환을 명령

```
public static double square(double num) {  
    return num * num;    // num * num의 결과를 반환  
}
```

명령 프롬프트

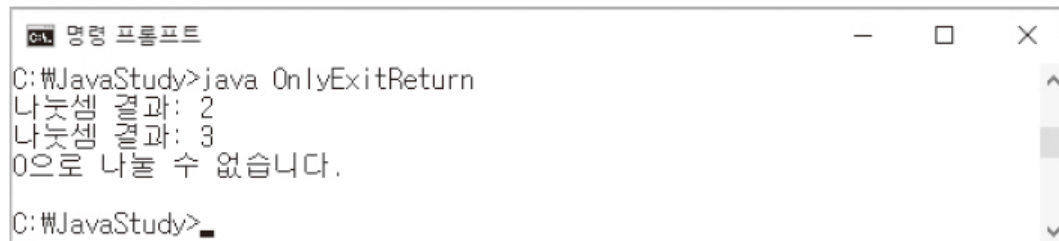
```
C:\JavaStudy>java MethodReturns  
4 + 5 = 9  
3.5 x 3.5 = 12.25  
  
C:\JavaStudy>
```

# return의 두가지 의미

```
public static void main(String[] args) {  
    divide(4, 2);  
    divide(6, 2);  
    divide(9, 0);  
}
```

메소드를 호출한 영역으로 값을 반환  
메소드의 종료

```
public static void divide(int num1, int num2) {  
    if(num2 == 0) {  
        System.out.println("0으로 나눌 수 없습니다.");  
        return; // 값의 반환 없이 메소드만 종료  
    }  
    System.out.println("나눗셈 결과: " + (num1 / num2));  
}
```



```
C:\JavaStudy>java OnlyExitReturn  
나눗셈 결과: 2  
나눗셈 결과: 3  
0으로 나눌 수 없습니다.  
C:\JavaStudy>
```



## 06-2. 변수의 스코프

# 가시성: 여기서는 저 변수가 보여요

```
if(...) {  
    int num = 5;  
    .... 지역변수 num  
}
```

```
매개변수 num  
public static void myFunc(int num) {  
    ....  
}  
지역변수의 범주에 포함되는 매개변수
```

```
for(int num = 1; num < 5; num++) {  
    ....  
}  
for문 내에서 유효한 지역변수 num
```

```
{ 속한 영역을 벗어나면 지역변수 소멸  
    int num2 = 33;  
    num2++;  
    System.out.println(num2);  
}
```

# 지역변수 선언의 예

```
public static void main(String[] args) {  
    boolean ste = true;  
    int num1 = 11;
```

영역 1

```
    if(ste) {  
        // int num1 = 22;    // 주석 해제하면 컴파일 오류 발생  
        num1++;  
        System.out.println(num1);  
    }
```

영역 2

```
    {  
        int num2 = 33;  
        num2++;  
        System.out.println(num2);  
    }
```

영역 3

```
    // System.out.println(num2);    // 주석 해제하면 컴파일 오류 발생  
}
```

같은 영역 내에서 동일 이름의 변수 선언 불가

## 06-3. 메소드의 재귀 호출

# 수학적 측면에서의 재귀적인 사고

$$5! = 5 \times 4 \times 3 \times 2 \times 1$$

$$4! = 4 \times 3 \times 2 \times 1$$

$$3! = 3 \times 2 \times 1$$

$$2! = 2 \times 1$$

$$1! = 1$$



$$5! = 5 \times 4!$$

$$4! = 4 \times 3!$$

$$3! = 3 \times 2!$$

$$2! = 2 \times 1!$$

$$1! = 1$$



$$n! = n \times (n-1)!$$

이 문장을 코드로 그대로 옮기도록  
하는 것이 재귀 메소드의 정의

# 재귀의 함수식 정의

$$5! = 5 \times 4!$$

$$4! = 4 \times 3!$$

$$3! = 3 \times 2!$$

$$2! = 2 \times 1!$$

$$1! = 1$$

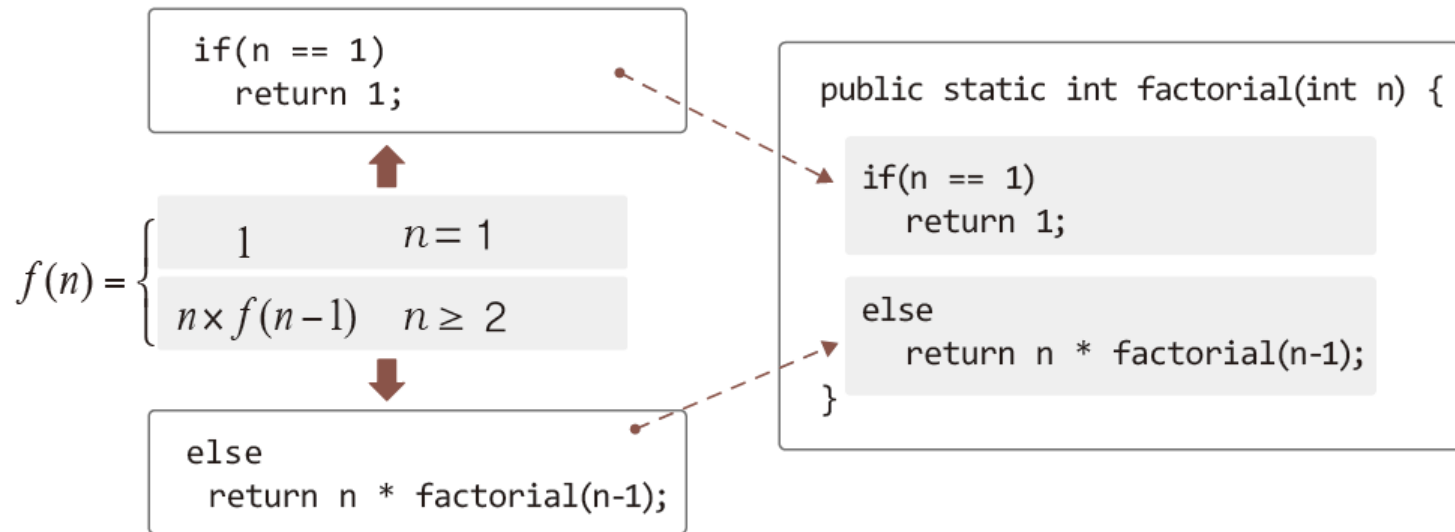


$$f(n) = \begin{cases} n \times f(n-1) & n \geq 2 \\ 1 & n = 1 \end{cases}$$

함수 f의 실행

함수 f의 정의

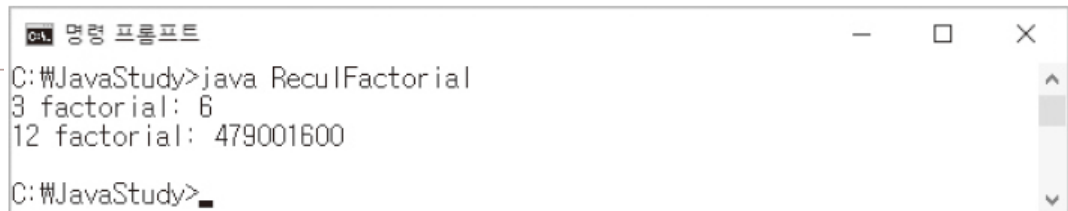
# 함수식 기반의 메소드 정의



# 팩토리얼 구현의 예

## ◆ ReculFactorial.java

```
1. class ReculFactorial {  
2.     public static void main(String[] args) {  
3.         System.out.println("3 factorial: " + factorial(3));  
4.         System.out.println("12 factorial: " + factorial(12));  
5.     }  
6.  
7.     public static int factorial(int n) {  
8.         if(n == 1)  
9.             return 1;  
10.        else  
11.            return n * factorial(n-1);  
12.    }  
13. }
```



```
C:\WJavaStudy>java ReculFactorial  
3 factorial: 6  
12 factorial: 479001600  
C:\WJavaStudy>
```



# 팩토리얼 구현의 예

