

# 12.29 木 \_ 매개 변수

## 기본형 매개변수와 참조형 매개변수

- **기본형(원시형) 매개변수 (Primitive)**

- 변수의 값을 읽기만 할 수 있다.(read only)
- boolean, char, byte, short, int, long, float, double
- 실제 값을 저장

- **참조형 매개변수 (Reference)**

- 변수의 값을 읽고 변경할 수 있다.(read & write)
- 기본형을 제외한 나머지 (String, System 등)
- 객체의 주소를 저장 (4byte, 0x00000000~0xffffffff)

## 반환 값이 없어 에러!!

```
package test;

class Bar{
    static int doSomething() {    => 정상 작동 시키려면 int 를 void 로 변경해야 한다
        // 반환 값이 있어야 하는데 없다.....
    }
}

public class test4 {

    public static void main(String[] args) {

    }

}
```

```
package test;
```

```

class Bar{
    static void doSomething(int argA) {
        System.out.println(argA);
    }
}

public class test4 {

    public static void main(String[] args) {
        Bar.doSomething(3);
    }

}

```

**스레드** - 일꾼

main 문부터 하나하나 읽어가면서 코드 실행시킨다



인자 값은 매개 변수에 **복사**가 되어서 들어온다 (call by value)

매서드를 호출할 때 넣어주는 input값 자체를 인자 값이라고 한

#### 기본형(Primitive) 매개변수 사용예제

```

public class SimpleTest {
    public static void main(String[] args) {
        Data d = new Data();
        d.x = 10;
        change(d.x);
        System.out.println("After change(): " + d.x);
    }

    static void change(int x) {
        x = 1000;
        System.out.println("change(): " + x);
    }
}

class Data {
    int x;
}

```

Call by value  
Call by reference

change(): 1000  
After change(): 10

#### 참조형(Reference) 매개변수 사용예제

```

public class SimpleTest {
    public static void main(String[] args) {
        Data d = new Data();
        d.x = 10;
        change(d);
        System.out.println("After change(): " + d.x);
    }

    static void change(Data d) {
        d.x = 1000;
        System.out.println("change(): " + d.x);
    }
}

class Data {
    int x;
}

```

change(): 1000  
After change(): 1000

```

package test;

class Bar{
    static void doSomething(int argA) {
        argA = 5;
        System.out.println(argA);
    }
}

```

```

}

public class test4 {

    public static void main(String[] args) {

        int x =3;

        Bar.doSomething(x);  // 메소드가 종료되면 호출한 이곳으로 다시 돌아온다

        System.out.println(x);

    }

}

```

## Call by reference

```

package test;

class Bar{
    static void doSomething(Foo argFooObj) {
        argFooObj.x = 10;

    }
}

class Foo{

    int x =3;
}

public class test4 {

    public static void main(String[] args) {

        Foo f1 = new Foo();

        Bar.doSomething(f1);  // 메소드가 종료되면 호출한 이곳으로 다시 돌아온다

        System.out.println(f1.x);

    }

}

```

# Call by value - 프리미티브

콜바이 레퍼런스 → 레퍼런스 값이 넘어간다

```
package test;

class Bar{
    static void doSomething(Foo argFooObj , int argY) {
        argY =20 ;
        argFooObj.x = 10;
    }
}

class Foo{
    int x =3;
}

public class test4 {

    public static void main(String[] args) {

        Foo f1 = new Foo();

        int y =10 ;

        Bar.doSomething(f1,y);  // 메소드가 종료되면 호출한 이곳으로 다시 돌아온다

        System.out.println(f1.x + ":" +y);

    }

}
```

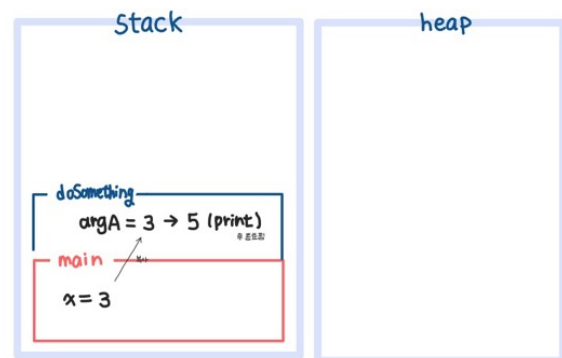
```

class Bar {
    static void doSomething (int argA) {
        argA = 5;
        sysout (argA)
    }
}

public class MyProject {
    public static void main (String args[]) {
        int x = 3;
        Bar. doSomething (x);
        sysout (x);
    }
}

```

## Call by Value



```

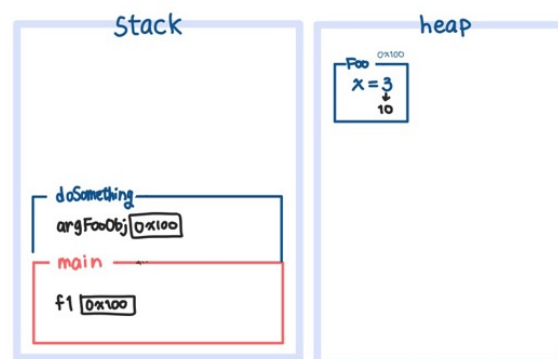
class Bar {
    static void doSomething (foo argFooObj) {
        argFooObj.x = 10;
    }
}

class Foo {
    int x = 3;
}

public class MyProject {
    public static void main (String args[]) {
        Foo f1 = new Foo();
        Bar. doSomething (f1);
        sysout (f1.x); // 10이 출력됨!
    }
}

```

## Call by reference



## Deep copy vs Shallow copy

```

package Test;

// Method

class Bar {
    int x = 3;
}

// Deep copy vs Shallow copy
public class MyProject {
    public static void main(String args[]) {

        Bar b1 = new Bar();
        b1.x = 10;
        Bar b2 = b1; // Shallow copy

        // <<----
        Bar b3 = new Bar();
        b3.x = b1.x;
        // ---->> Deep copy
    }
}

```

```

package Test;

// Method

class Bar {
    int x = 3;
}

// Deep copy vs Shallow copy
public class MyProject {
    public static void main(String args[]) {

        Bar b1 = new Bar();
        b1.f1 = new Foo();
        b1.k = new int[5];

        // b1을 Deep copy 하라~!
        Bar b2 = new Bar();
        b2.f1 = b1.f1;
        b2.k = b1.k;
    }
}

```