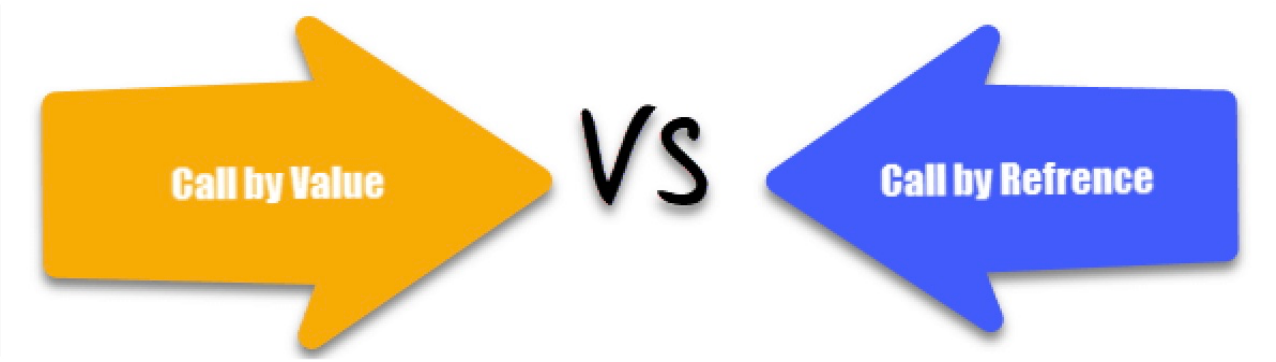
Call by Value vs Call by Reference



Call by Value와 Call by Reference는 함수 호출 방식을 나타내는 용어입니다.

- Call by Value는 값에 의한 호출 방식을 말합니다.
- Call by Reference는 참조에 의한 호출 방식을 말합니다.

함수 호출 방식은 함수가 호출될 때 매개변수(parameter)를 전달하는 방법을 의미합니다. 함수가 호출될 때 매개변수는 호출한 코드 블록에서 선언한 변수의 값을 복사하여 전달됩니다.

Call by Value에서는 값을 복사하여 전달합니다. 따라서 함수 내에서 매개변수의 값을 변경하더라도 호출한 코드 블록에서는 변경된 값이 반영되지 않습니다.

반면에 Call by Reference에서는 값이 아니라 메모리 주소를 전달합니다. 따라서 함수 내에서 매개변수가 참조하는 값이 변경되면 호출한 코드 블록에서도 변경된 값이 반영됩니다.

Call by Value와 Call by Reference는 각각 장단점

Call by Value

장점

- 값을 복사하기 때문에 원본 데이터가 보호됩니다.
- 다른 스레드에서 값을 변경해도 영향을 받지 않습니다.

단점

- 값이 복사되기 때문에, 큰 데이터를 전달할 때 메모리 사용량이 증가할 수 있습니다.

Call by Reference

장점

- 값을 복사하지 않기 때문에, 메모리 사용량이 적습니다.
- 호출된 함수에서 전달된 인수에 대한 변경이 가능합니다.

단점

- 전달된 값이 변경될 수 있기 때문에, 호출된 함수에서 예상치 못한 결과가 발생할 수 있습니다.
- 여러 스레드가 값을 변경할 경우, 동기화 문제가 발생할 수 있습니다.

따라서, 어떤 호출 방식을 선택할지는 상황에 따라 다르며, 프로그램의 성격과 목적에 따라 적절한 호출 방식을 선택해야 합니다.

Java는 Call by Value인가? Call by Reference인가?

```
public class CallTests {
    private static void add(int a){
        a += 1;
    }

    private static void addArr1(int[] arr){
        for (int i = 0; i < arr.length; i++) {
            arr[i] += 1;
        }
    }

    private static void addArr2(int[] arr){
        int length = arr.length;
        arr = new int[length];
    }

    @Test
    @DisplayName("Call by Value Test")
    void callByValue(){
        int a = 1;
        add(a);
        assertEquals(1, a); // pass
    }

    @Test
    @DisplayName("Call by Reference Test1")
    void callByReference1(){
        int[] arr = new int[2];
        addArr1(arr);
        assertEquals(1, arr[0]); // pass
        assertEquals(1, arr[1]); // pass
    }

    @Test
    @DisplayName("Call by Reference Test2")
    void callByReference2(){
        int[] arr = new int[2];
        addArr1(arr);
        addArr2(arr);
        assertEquals(0, arr[0]); // fail
        assertEquals(0, arr[1]); // fail
    }
}
```