값복사와 참조복사

💡 기본 자료형 변수간의 대입과 배열간의 대입은 서로 차이가 있다. 기본 자료형: `char`, `boolean`, `byte`, `short`, `int`, `long`, `float`, `double`

#01. 값 복사

기본 자료형 변수를 서로 대입할 경우의 현상.

1) 기본 자료형간의 대입

단순 복사가 발생하기 때문에 복사 후 원본이 변경되더라도 복사본에는 영향이 없다. (반대의 경우도 마찬가지)

기본 자료형 변수간의 복사 실험

Ex01_값복사

```
public class ValueCopy {
    public static void main(String[] args) {
        int a = 10; // 원본
        int b = a;  // 복사본
        System.out.println("a=" + a);
        System.out.println("b=" + b);
        System.out.println("----------");

        a += 10;    // 원본 수정
        System.out.println("a=" + a);
        System.out.println("b=" + b);
        System.out.println("----------");

        b -= 10;    // 복사본 수정
        System.out.println("a=" + a);
        System.out.println("b=" + b);
    }
}
```

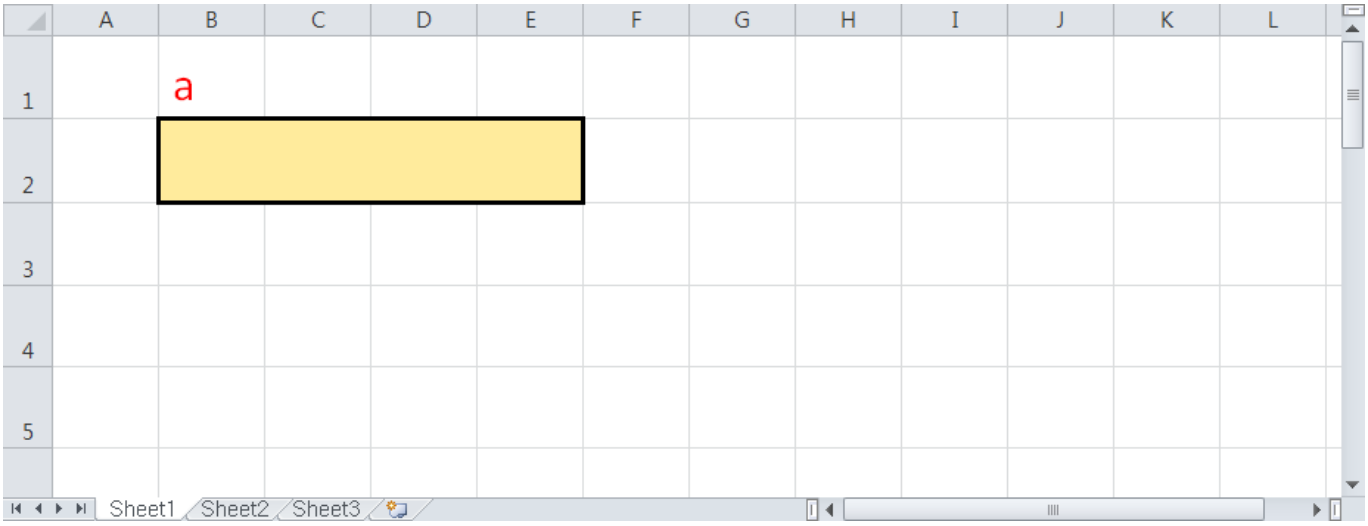
- 출력결과

```
a=10
b=10
----------
a=20
b=10
----------
a=20
b=0
```

2) 변수간의 값 복사 원리

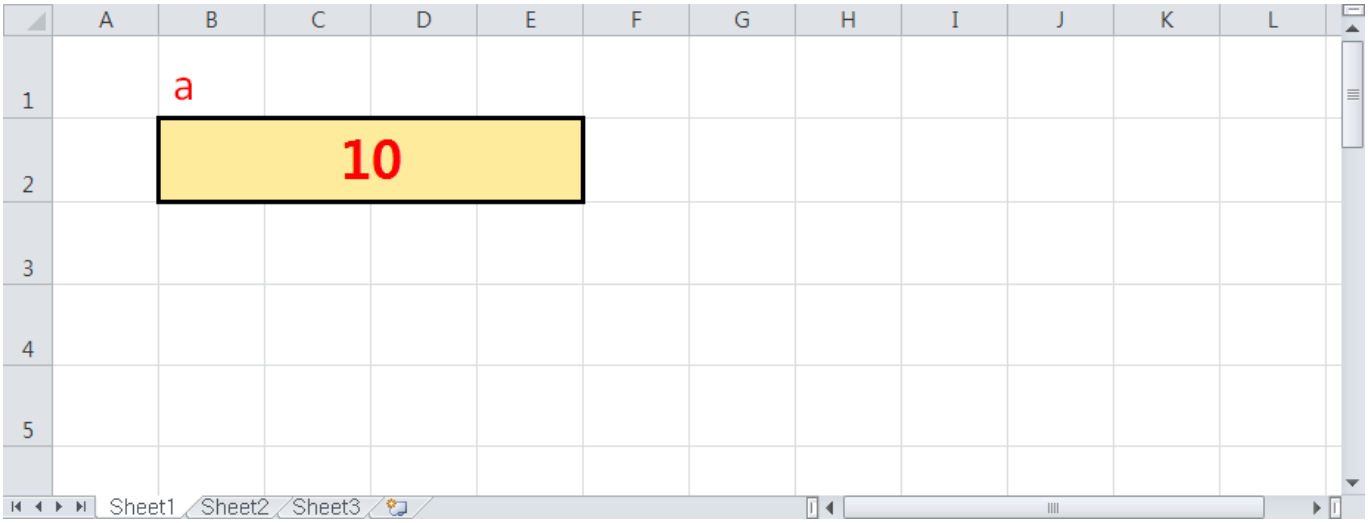
프로그램이 동작하는 동안 하나의 변수가 선언되면 컴퓨터는 해당 변수의 데이터 타입에 맞는 공간을 RAM 안에 점유한다. 예를 들어 `int`형의 변수 `a`를 선언한다면 int의 메모리 크기는 4byte 이므로 RAM 카드 안에서 4byte에 해당하는 영역을 점유하고 그 영역에 `a`라는 식별 이름을 적용한다.

```
int a;
```



선언한 변수에 값을 대입한다는 것은 점유해 둔 메모리 공간에 데이터를 기록한다는 의미이다.

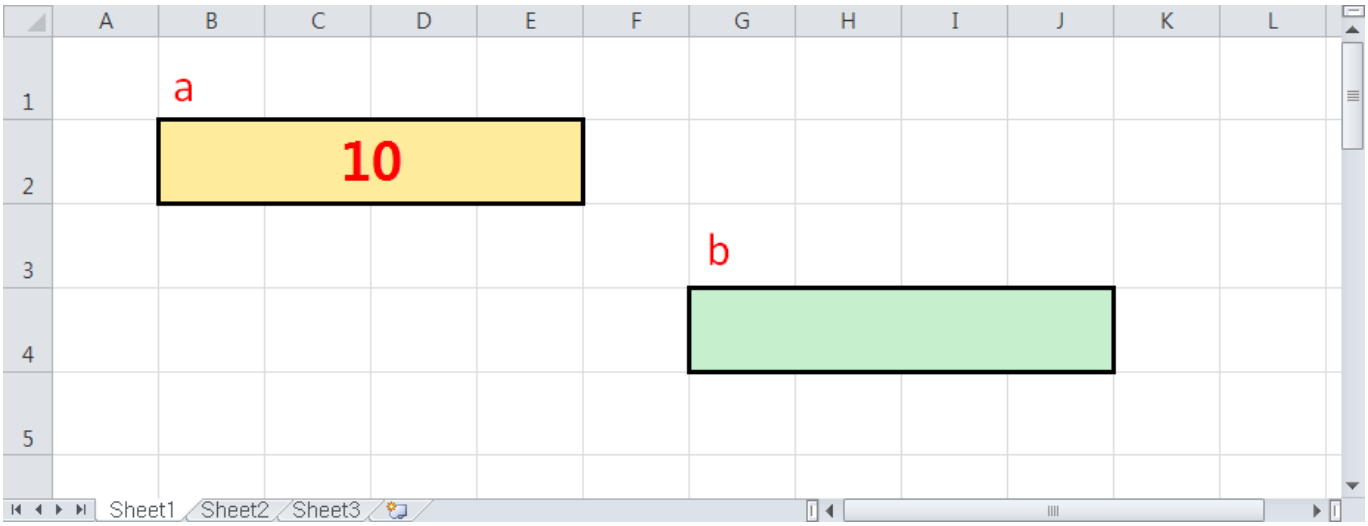
```
a = 10;
```



새로운 `int`형 변수 `b`를 선언하면 컴퓨터는 RAM 상의 무작위 위치에 4byte에 해당하는 공간을 점유한다.

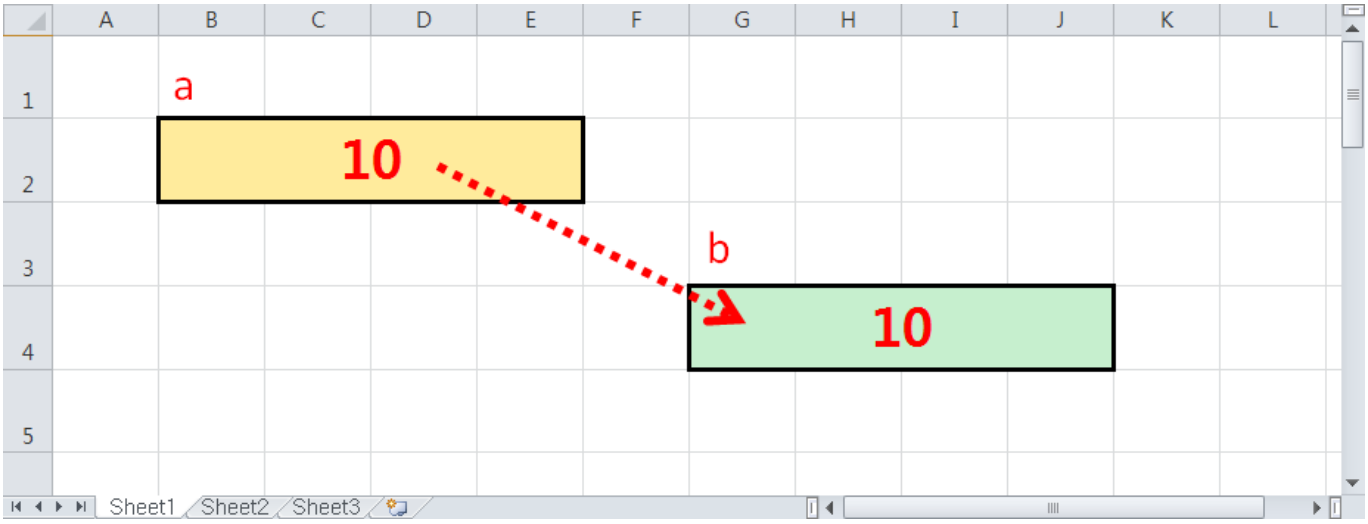
💡 메모리의 공간을 무작위로 사용한다고 해서 Random Access Memory 입니다.

```
int b;
```



a에 저장되어 있던 값을 b에 복사한다는 것은 메모리간의 값을 서로 복제한다는 의미이다.

```
b = a;
```



#02. 참조 복사

배열을 서로 대입할 경우의 현상.

1) 배열간의 대입

배열간의 대입은 배열의 변수 이름에 원소들을 참조시키기 때문에 복사본을 수정할 경우 원본도 함께 수정된다. (반대의 경우도 마찬가지)

배열간의 복사 실험

Ex02_배열복사

```
public class ArrayCopy {
    public static void main(String[] args) {
        int[] origin = {1, 2};
        int[] copy = origin;

        System.out.println("origin[0]=" + origin[0]);
        System.out.println("origin[1]=" + origin[1]);
        System.out.println("copy[0]=" + copy[0]);
        System.out.println("copy[1]=" + copy[1]);
        System.out.println("----------");

        copy[0] += 100;
        copy[1] += 200;

        System.out.println("origin[0]=" + origin[0]);
        System.out.println("origin[1]=" + origin[1]);
        System.out.println("copy[0]=" + copy[0]);
        System.out.println("copy[1]=" + copy[1]);
    }
}
```

- 출력결과

```
origin[0]=1
origin[1]=2
copy[0]=1
copy[1]=2
----------
origin[0]=101
origin[1]=202
copy[0]=101
copy[1]=202
```

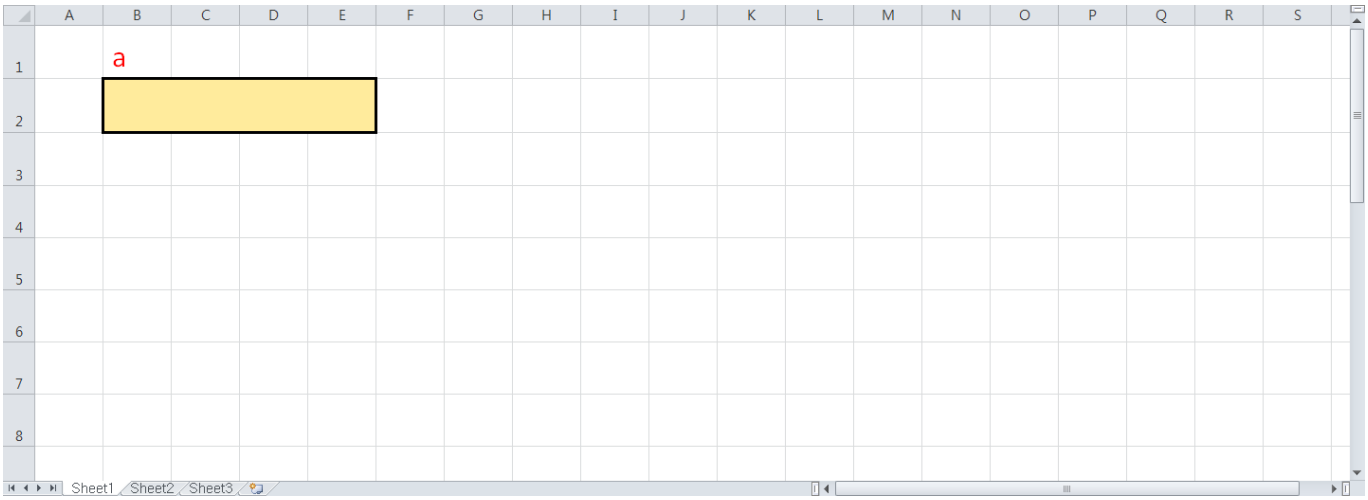
2) 배열간의 참조 복사 원리

컴퓨터의 메모리(RAM)은 각각의 칸에 접근할 수 있는 위치값을 int 형으로 관리한다.

모든 형태의 배열은 데이터 타입이 어떤 것이건 무조건 정수형 값을 저장할 수 있는 메모리 공간을 생성한다.
즉 모든 배열 자체의 메모리 크기는 4byte 이다. 32bit = 4byte
64bit = 8byte

배열을 선언할 때 데이터 타입으로 명시되는 `int[]`라는 것은 이 변수에 정수형 배열의 위치만을 저장할 수 있게 용도를 제한한다는 의미이다.

```
int[] a;
```

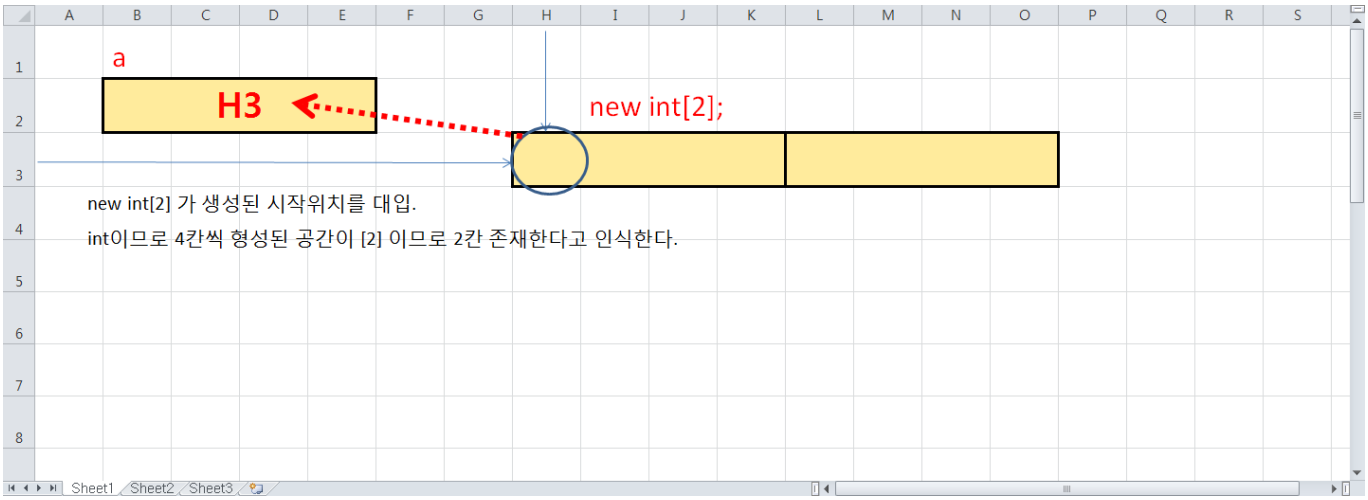


new 키워드를 사용하여 배열의 크기를 할당한다는 의미는 변수가 가리키는 위치부터 몇 칸으로 구성된 배열이 존재한다는 것을 의미한다.

```
a = new int[2];
```

즉, 위의 코드는 메모리 어딘가에 int형 변수 2개가 묶인 배열이 존재하는데 그 배열의 시작위치를 a라는 변수에 저장하겠다는 의미이다.

아래 그림과 같이 H3라는 위치값이 저장되었다면 총 8byte의 메모리 공간이 H3부터 연속적으로 사용되고 a라는 변수를 통해 H3부터 8칸을 new int가 지정한 대로 4칸씩 나누어 읽게 된다.



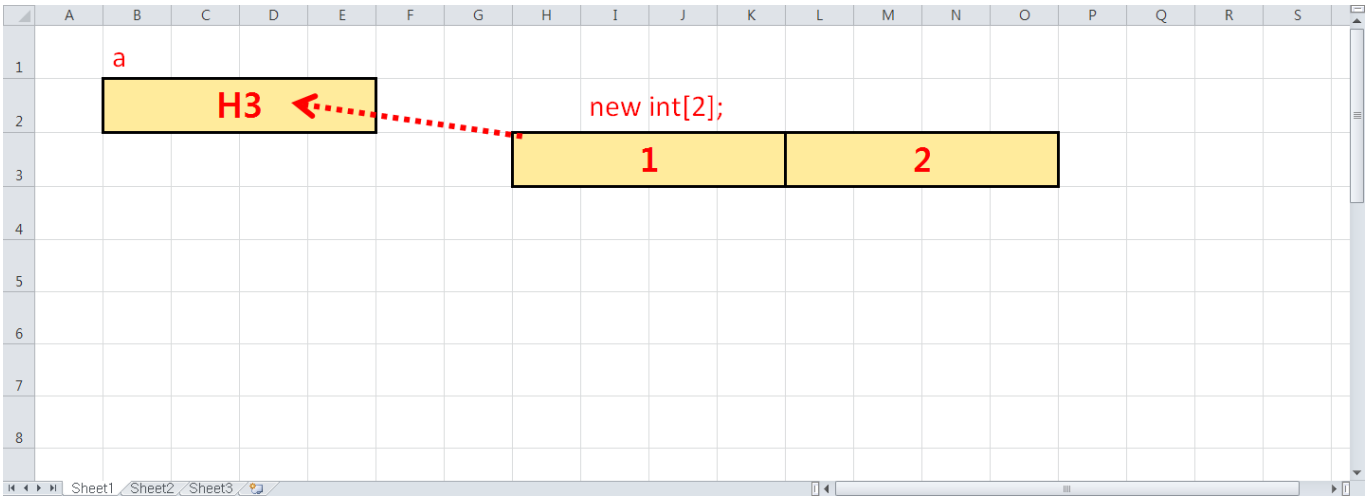

```
double[] b = new double[3];
             double = 8byte
```

만약 위와 같이 구현되었다면 아래와 같은 의미가 되는 것이다.

1. b라는 정수형 변수가 생성되는데 이 변수에는 double 형 배열의 위치만 저장할 수 있다.
2. double형 배열 3칸을 어딘가에 만들고 그 시작위치를 b에 저장한다.
3. b가 저장하고 있는 위치부터 8byte씩 3개 이므로 24칸의 메모리가 사용된다.

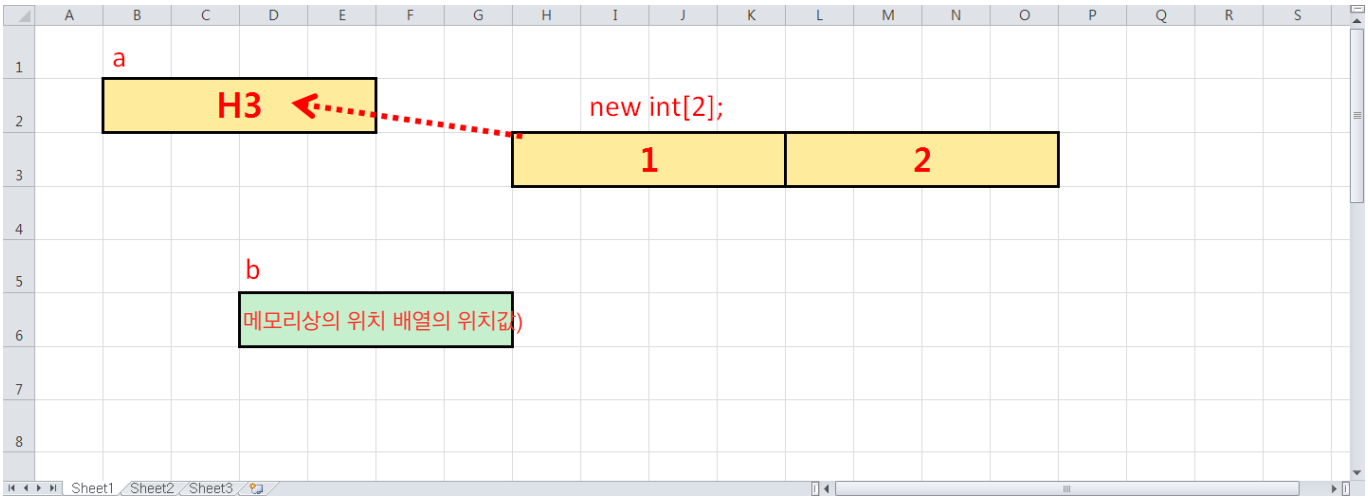
결국 배열 a의 0번에 값을 대입한다는 것은 a라는 변수에 저장된 메모리 위치를 찾아가서 new int에 따라 4칸씩 나누어 서 그 중 0번째 영역에 1을 넣는다는 의미가 된다.

```
a[0] = 1;
a[1] = 2;
```



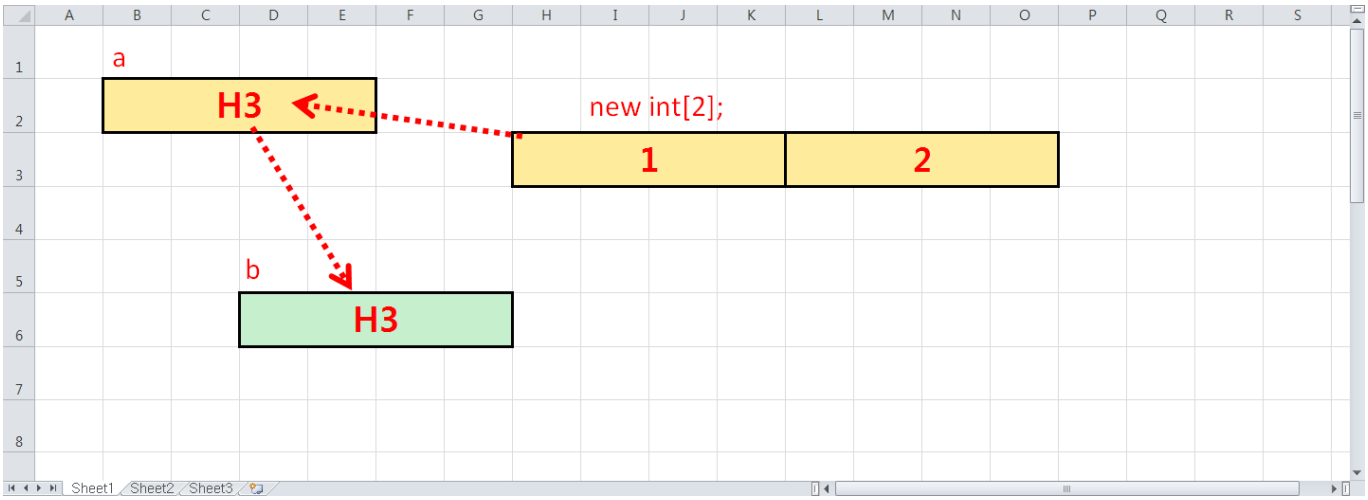
한편, 새로운 정수형 배열 b를 선언하면 배열의 위치값을 저장할 수 있는 정수형 메모리 공간을 생성한다.

```
int[] b;
```



배열 b에 배열 a를 복사하면 a에 저장되어 있던 위치값인 H3를 복사한다.

```
b = a;
```



결국 a의 0번째와 b의 0번째는 같은 메모리 위치를 가리킨다.

3) 배열을 값복사 형태로 처리하는 방법

배열의 각 원소를 개별적으로 복사하기

Ex03_배열의_값복사

```
public class ArrayCopy2 {
    public static void main(String[] args) {
        int[] origin = {1, 2};

        // 원본과 동일한 사이즈의 배열 생성
        int[] copy = new int[origin.length]; 원본의 길이만큼 만들어야함  원본의 원소가 2개여서 카피도 2개

        // 각각의 원소를 개별적으로 복사해야 한다.
        copy[0] = origin[0];
        copy[1] = origin[1];

        System.out.println("origin[0]=" + origin[0]);
        System.out.println("origin[1]=" + origin[1]);
        System.out.println("copy[0]=" + copy[0]);
        System.out.println("copy[1]=" + copy[1]);
        System.out.println("----------");

        // 복사본을 수정하더라도 원본의 변화가 없다.
        copy[0] += 100;
        copy[1] += 200;

        System.out.println("origin[0]=" + origin[0]);
        System.out.println("origin[1]=" + origin[1]);
        System.out.println("copy[0]=" + copy[0]);
        System.out.println("copy[1]=" + copy[1]);
    }
}
```

- 출력결과

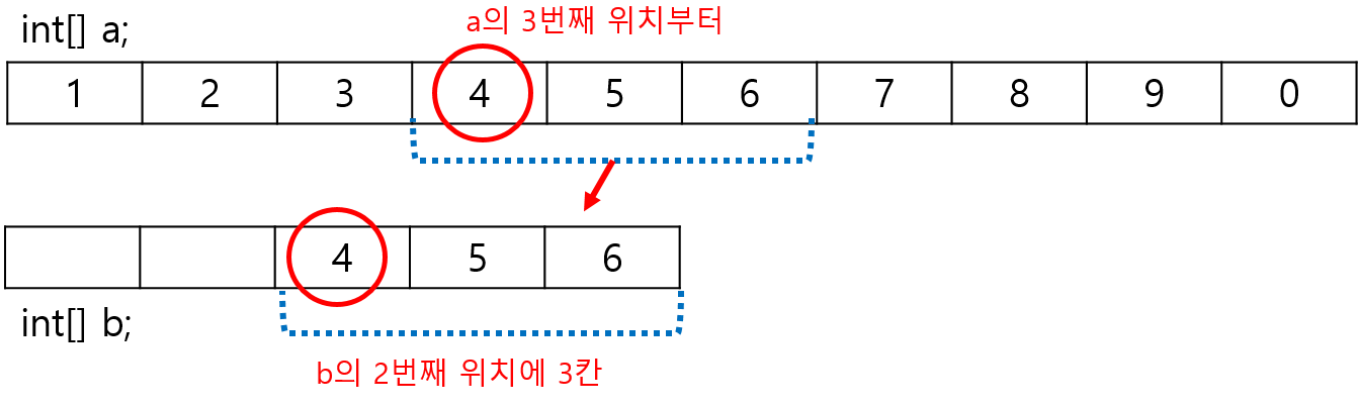
```
origin[0]=1
origin[1]=2
copy[0]=1
copy[1]=2
----------
origin[0]=1
origin[1]=2
copy[0]=101
copy[1]=202
```

자바에서 제공하는 기능을 사용하기

```
System.arraycopy(원본배열, 원본의 복사 시작 위치,
                 복사될 배열, 복사가 시작될 위치, 복사할 값의 길이);
```

사용 예

```
int[] a = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 };
int[] b = new int[5];
          a.length
// a의 3번째 위치부터 b의 2번째 위치에 3칸을 복사
System.arraycopy(a, 3, b, 2, 3);
                  a,0,b,0,a.length 으로 해도됨
```



Ex04_ArrayCopy

```
public class ArrayCopy3 {
    public static void main(String[] args) {
        int[] origin = {1, 2, 3, 4, 5};
        // 원본과 동일한 사이즈의 배열 생성
        int[] copy = new int[origin.length];

        // origin의 0번째 부터 copy의 1번째에 3칸을 복사
        System.arraycopy(origin, 0, copy, 1, 3);
```

```
        for (int i=0; i<origin.length; i++) {
            System.out.printf("origin[%d]=%d \t copy[%d]=%d\n",
                              i, origin[i], i, copy[i]);
        }

        System.out.println("-----------");

        // 주로 다음과 같이 사용됨 --> 통째로 복사
        System.arraycopy(origin, 0, copy, 0, origin.length);

        for (int i=0; i<origin.length; i++) {
            System.out.printf("origin[%d]=%d \t copy[%d]=%d\n",
                              i, origin[i], i, copy[i]);
        }
    }
}
```

- 출력결과

```
origin[0]=1      copy[0]=0
origin[1]=2      copy[1]=1
origin[2]=3      copy[2]=2
origin[3]=4      copy[3]=3
origin[4]=5      copy[4]=0
-----------
origin[0]=1      copy[0]=1
origin[1]=2      copy[1]=2
origin[2]=3      copy[2]=3
origin[3]=4      copy[3]=4
origin[4]=5      copy[4]=5
```

참고

💡 다른 설명에서는 다음과 같이 설명하기도 합니다.

- 값 복사 = 깊은 복사 (Deep Copy)
- 참조 복사 = 얕은 복사 (Shallow Copy)