배열의 깊은 복사(Deep Copy) & 얕은 복사(Shallow Copy)

◎ 얕은복사(Shallow Copy) : 배열의 주소 값만 복사 (복사한 배열 수정 시, 원 배열도 함께 수정됨)
◎ 깊은복사(Deep Copy) : 배열의 실제 값을 새로운 메모리 공간에 복사 (원 배열 수정 안됨)

◈ 1차원 배열

얕은 복사(Shallow Copy)

원래 배열의 주소 값만 복사

복사한 배열을 수정하면 원래 배열도 함께 수정됨

단순 배열 변수 선언을 통한 복사방법 (int[] B = A;)

```
int[] A = {1,2,3,4,5};
int[] B = A;

B[0] = 1000;

System.out.println("A : " + Arrays.toString(A));
System.out.println("B : " + Arrays.toString(B));

// > A : [1000, 2, 3, 4, 5]
//   B : [1000, 2, 3, 4, 5]
```

깊은 복사(Deep Copy)

원래 배열의 실제 값을 복사

복사한 배열을 수정해도 원래 배열은 바뀌지 않음

새로운 메모리 공간에 복사

배열.clone()을 이용하여 깊은 복사

```
int[] A = {1,2,3,4,5};
int[] B = A.clone();
// clone() 메소드는 배열에만 사용하는 것이 좋다.
// 복사하는 대상이 배열이 아니면 복사 생성자, 복사 팩터리가 좋다.

B[0] = 1000;

System.out.println("A : " + Arrays.toString(A));
System.out.println("B : " + Arrays.toString(B));

// > A : [1, 2, 3, 4, 5]
//   B : [1000, 2, 3, 4, 5]
```

◈ 2차원 배열

얕은 복사(Shallow Copy)

단순 변수 선언 뿐만 아니라 A.clone()을 사용해도 얕은 복사 가능

```
int[][] A = {{1,2,3},
                        {4,5,6},
                        {7,8,9}};

int[][] B = A;
int[][] C = A.clone();
```

깊은 복사(Deep Copy)

이중 for문을 이용하여 깊은 복사 가능

반복문 + System.arraycopy() 를 사용하여 가능

System.arraycopy (원래 배열, 원래 배열 시작 위치, 새 배열, 새 배열 시작 위치, 길이)

```
int[][] A = {{1,2,3},
                        {4,5,6},
                        {7,8,9}};

// 2중 for문
for(int i = 0; i < A.length; i++){
        for(int j = 0 ; j < A[0].length; j++){
                B[i][j] = A[i][j];
        }
}

// 반복문 + System.arraycopy()
int[][] C = new int[A.length][A[0].length]; // C 선언
for(int i = 0; i < A.length; i++){
        System.arraycopy(A[i], 0, C[i], 0, C[i].length);
}
```