

배열 (Array)

▶ 참조 자료형(Reference Type)

데이터가 저장되어 있는 공간의 주소를 저장하는 자료형
기본형을 제외한 모든 자료형

주소를 저장하기위해 데이터 저장크기는 4Byte를 사용

✓ 자료형 구분

저장되는 값에 따라 기본자료형과 참조자료형으로 구분됨

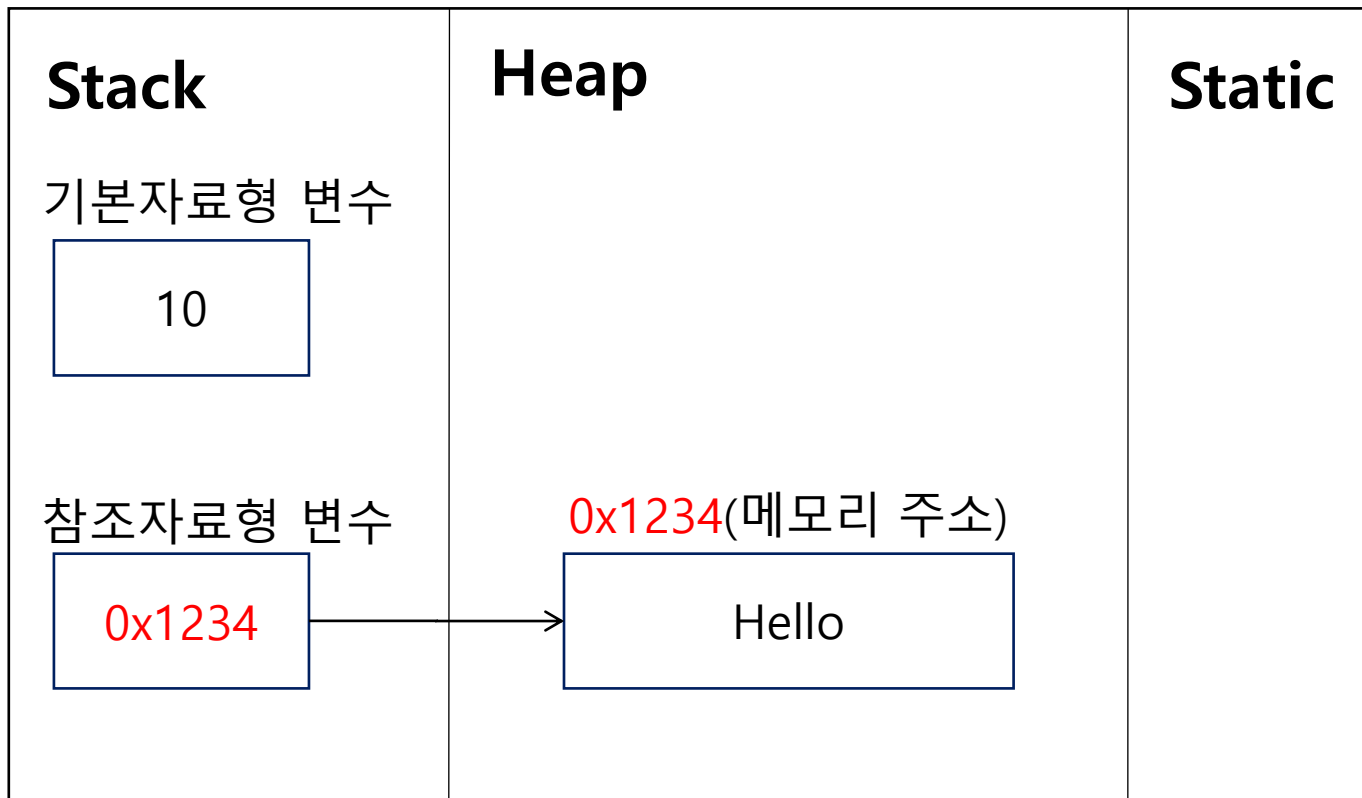
기본자료형 : 정수, 실수, 문자, 논리 리터럴

참조자료형 : 배열, 열거, 클래스, 인터페이스

▶ 참조 자료형(Reference Type)

기본자료형 변수는 메모리 Stack 영역에 만들어짐

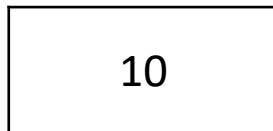
참조자료형 변수는 Stack 영역에, 실제 데이터는 Heap 영역에 저장됨



배열

같은 자료형의 변수를 하나의 묶음으로 다루는 것
배열의 각 변수공간에 접근하기 위해 순서번호(인덱스)를 사용
인덱스 번호는 0부터 시작됨

변수



num

→ `int num;`
`num = 10;`

`int[] arr = new int[5];`
5

arr

1 int=4byte
20byte

배열



arr[0]

arr[1]

arr[2]

arr[3]

arr[4]

▶ 배열 선언

배열공간의 주소 저장용 참조(Reference) 변수를 만드는 것

✓ 배열 선언

자료형[] 배열명 ;

자료형 배열명[] ;



✓ 배열 선언 예시

```
int[] arr;
```

```
int arr[];
```

▶ 배열 생성

실제 데이터가 저장될 배열 공간을 만드는 것

✓ 배열 공간 할당

```
자료형[ ] 배열명 = new 자료형[배열크기];
```

```
자료형 배열명[ ] = new 자료형[배열크기] ;
```

✓ 배열 공간 할당 예시

```
int[] arr = new int[5];
```

```
int arr[] = new int[5];
```

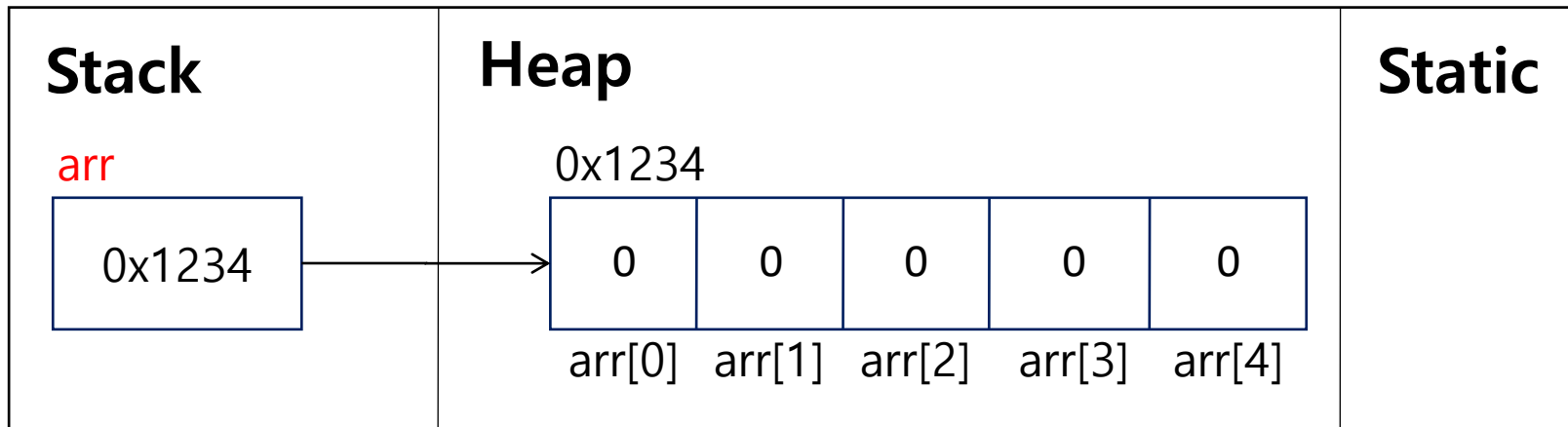
▶ 배열 저장구조

메모리의 Stack영역에 주소를 저장할 참조변수 할당

Heap영역에 실제 데이터가 저장될 배열 공간 할당

배열 공간의 주소를 이용해 인덱스를 참조하는 방식으로 값 처리

```
int[] arr = new int[4];
```



▶ 배열 초기화

✓ 인덱스를 이용한 초기화

```
int[] arr = new int[5];  
arr[0] = 10;  
arr[1] = 20;  
arr[2] = 30;  
arr[3] = 40;  
arr[4] = 50;
```

✓ 선언과 동시에 초기화

```
int[] arr1 = {10, 20, 30, 40, 50};  
int[] arr2 = new int[] {10, 20, 30, 40, 50};  
String fruit[] = {"사과", "포도", "참외"};
```


▶ 배열

✓ 배열 예시

```
int[] arr = new int[5];  
arr[0] = 10;  
arr[1] = 20;  
arr[2] = 30;  
arr[3] = 40;  
arr[4] = 50;  
System.out.println(arr[0]);  
System.out.println(arr[1]);  
System.out.println(arr[2]);  
System.out.println(arr[3]);  
System.out.println(arr[4]);
```

▶ 배열

✓ 배열 예시

```
int[] arr = new int[5];  
for(int i=0; i<arr.length; i++){  
    arr[i] = (i+1)*10;  
}  
for(int i=0; i<arr.length; i++) {  
    System.out.println(i+" 인덱스 값 : "+arr[i]);  
}
```

* index가 순차적으로 증가함에 따라
저장할 값이 규칙적이라면
반복문을 통해 배열 초기화 가능

----- 실행 결과 -----

```
0 인덱스 값 : 10  
1 인덱스 값 : 20  
2 인덱스 값 : 30  
3 인덱스 값 : 40  
4 인덱스 값 : 50
```

▶ 배열

✓ 배열 예시

```
int[] arr = new int[5];  
for(int i=0; i<arr.length; i++){  
    arr[i] = (i+1)*10;  
}  
System.out.println(arr);  
//배열의 시작 주소 값 출력
```

▶ 배열

- 5개의 수를 입력 받아 합을 구하는 프로그램 만들기(배열 이용)

=====출력=====

1번째 수 입력 : 10

2번째 수 입력 : 20

3번째 수 입력 : 30

4번째 수 입력 : 40

5번째 수 입력 : 50

$10 + 20 + 30 + 40 + 50 = 150$

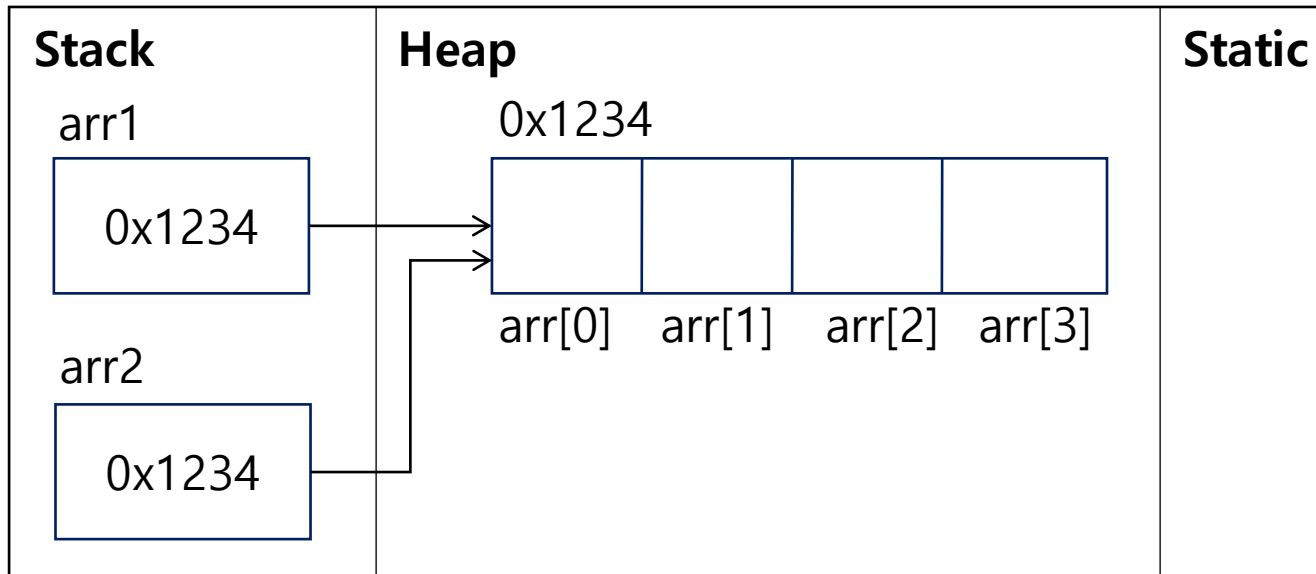
▶ 배열 복사

✓ 얕은 복사

객체의 주소 값만 가져와 참조형 변수에 저장하고 하나의 객체를 두 변수가 참조하는 것

```
int[] arr1 = new int[4];
```

```
int[] arr2 = arr1;
```



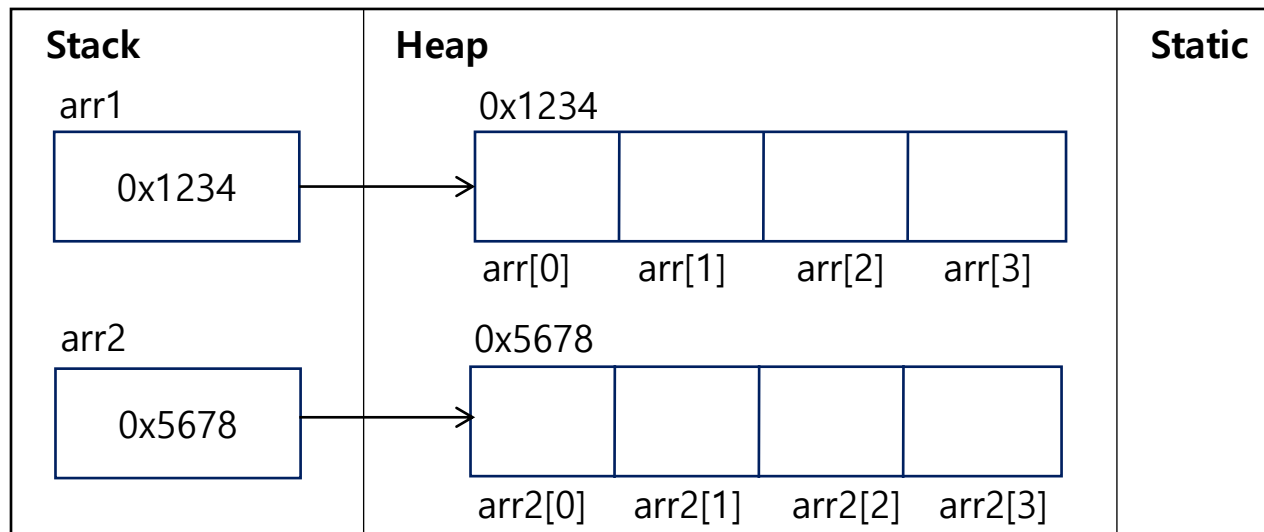
▶ 배열 복사

✓ 깊은 복사

새로운 배열 객체를 생성하여 기존 배열의 데이터를 복사하는 것

```
for(int i = 0; i < arr1.length; i++) {
    arr2[i] = arr1[i];
}

System.arraycopy(arr1, 0, arr2, 0, arr1.length);
arr2 = Arrays.copyOf(arr1, arr1.length);
arr2 = arr1.clone();
```

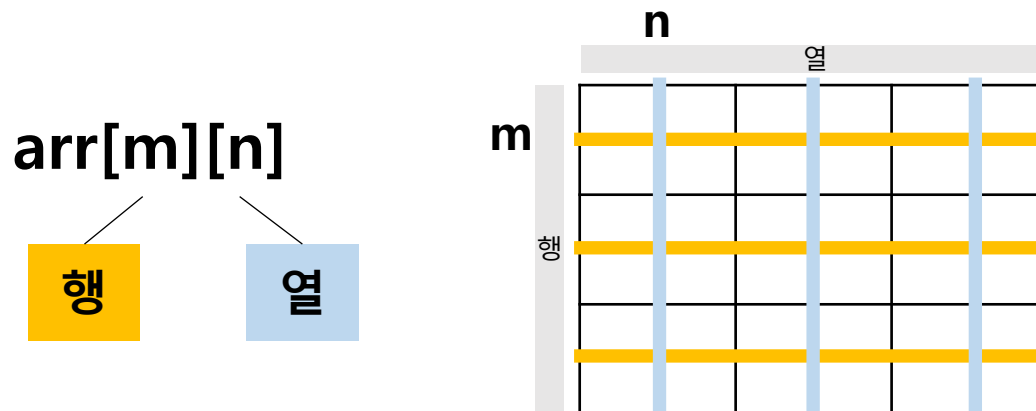


2차원 배열

▶ 2차원 배열

자료형이 같은 1차원 배열의 묶음으로 배열 안에 다른 배열 존재
 2차원 배열은 할당된 공간마다 인덱스 번호 두 개 부여
 (앞 번호는 행, 뒷 번호는 열 ([0][0]))

✓ 인덱스 값 이해



- m값이 올라가면
행이 아래로 가고
- n값이 올라가면
열이 옆으로 이동

▶ 2차원 배열 선언과 할당

✓ 배열 선언

자료형[][] 배열명 ;

자료형 배열명[][] ;

자료형[] 배열명[] ;

✓ 배열 할당

자료형[][] 배열명 = new 자료형[행크기][열크기];

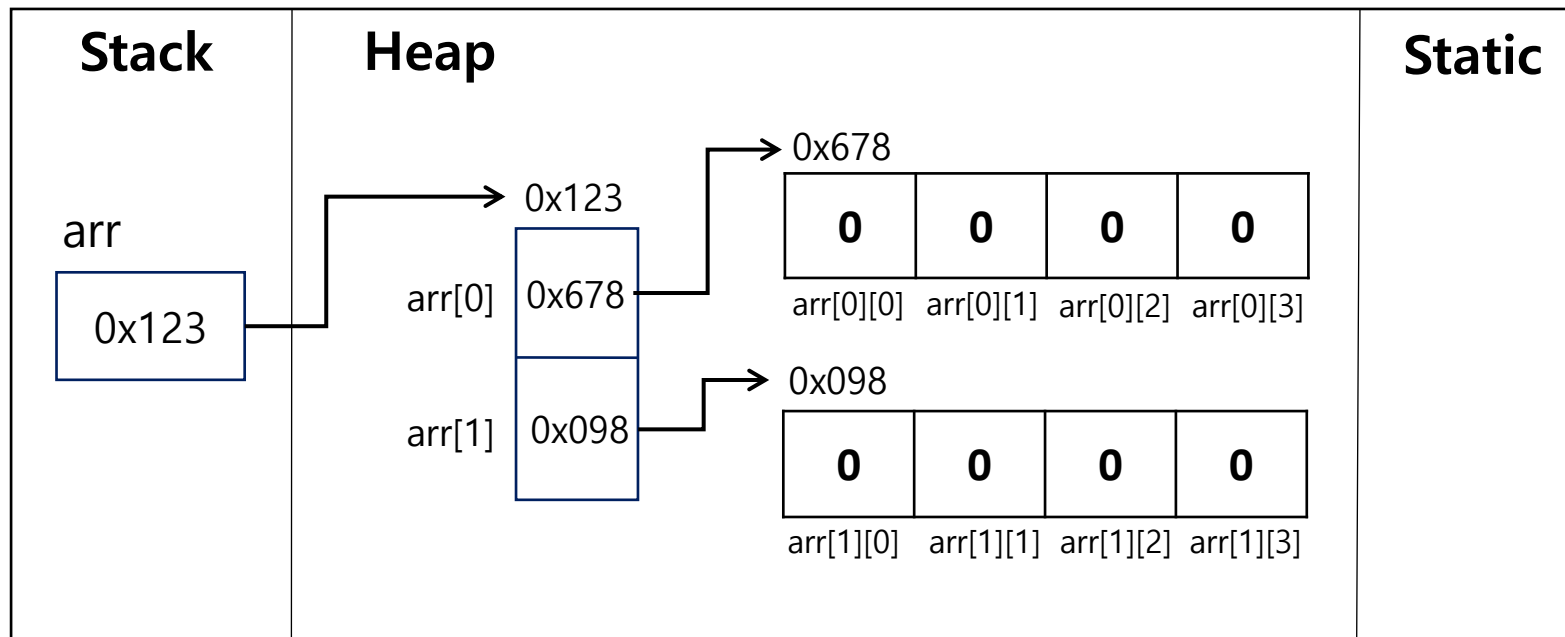
자료형 배열명[][] = new 자료형[행크기][열크기] ;

자료형[] 배열명[] = new 자료형[행크기][열크기] ;

ex) `int[][] arr = new int[3][4];`
`int arr[][] = new int[3][4];`

▶ 2차원 배열 구조

```
int [][] arr=new int[2][4];
```



▶ 2차원 배열 초기화

✓ 인덱스를 이용한 초기화

ex) `arr[0][0] = 1;`

`arr[1][1] = 2;`

✓ for문을 이용한 초기화

ex)

```
for(int i = 0; i < arr.length; i++) {  
    for(int j = 0; j < arr[i].length; j++) {  
        arr[i][j] = j;  
    }  
}
```

✓ 선언과 동시에 초기화

ex) `int[][] arr = {{1, 2, 3, 4}, {5, 6, 7, 8}};`

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

`String fruit[][] = {{ "사과", "딸기", "석류",
 "바나나", "참외", "레몬" }};`