2차원 배열

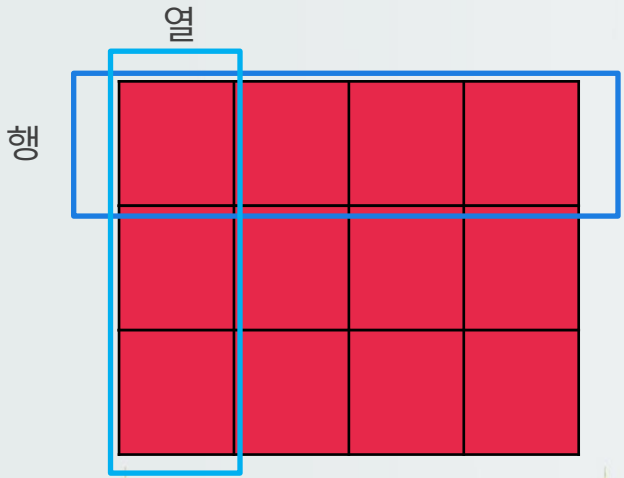
2차원 배열

2차원 배열

2차원 배열

→ 1차원 배열 안에 다른 배열을 넣은 것
→ 배열은 저장된 값 마다 인덱스 번호 두 개로 설정되고 앞번호는 행, 뒷 번호는 열

인덱스 사용법 → arr[행번호][열번호]



→ 배열의 행번호와 열번호는 0번부터 시작
→ 예시의 그림은 3행 4열 배열로
→ arr[0][0]부터 arr[2][3]까지 총 12개 인덱스

2차원 배열

2차원 배열의 선언

→ 주소 값을 가지지 않은 변수 생성(Stack)

자료형 [][] 변수명; → **int [][] arr;**

자료형 변수명[][]; → **int arr [][];**

2차원 배열의 할당

→ 배열객체 생성 후 변수에 주소 값 할당(Heap)

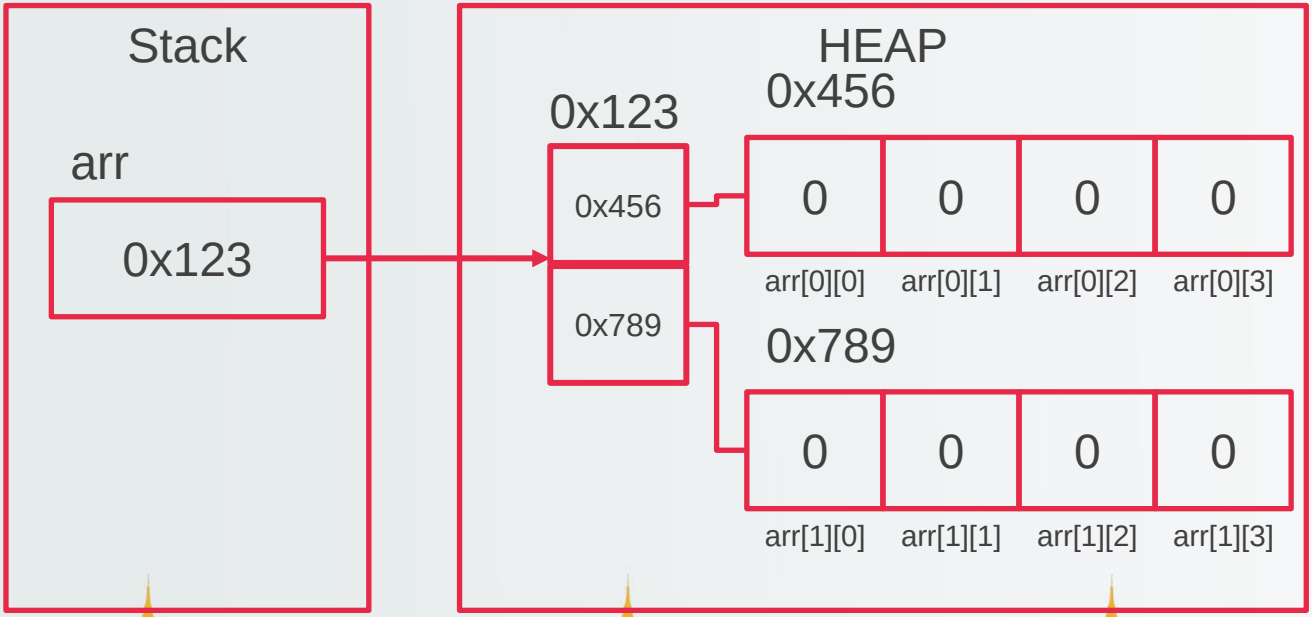
변수명 = new 자료형[행크기][열크기];→ arr = new int[2][3];

→ 2행 3열 크기의 2차원배열

2차원 배열

2차원 배열의 저장 구조

int [][] arr = new int[2][4]; → 2행 4열 배열 생성



2차원 배열

2차원 배열의 초기화

→ 배열을 선언 후 할당하게 되면 각 자료형 별 기본값이 들어 감
→ 숫자형 변수 모두 0, boolean : false, char : ‘ ’
→ 선언과 동시에 값을 기록 할 수 있음

int[][] arr ={{1,2,3},{4,5,6}};

1	2	3
4	5	6

String[][] arr = {{“김”, “이”}, {“박”, “최”}};

김	이
박	최

2차원 배열

2차원 배열 값 기록

1. 인덱스를 이용한 직접 값 기록

```
int [][] arr = new int[2][2];
arr[0][0] = 1;
arr[0][1] = 2;
arr[1][0] = 3;
arr[1][1] = 4;
```

2. for문을 이용한 값 기록

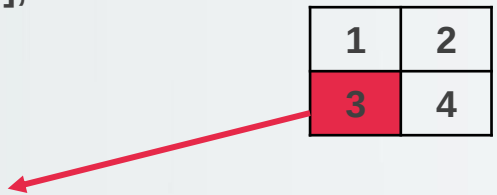
```
int [][] arr = new int[5][5];
int k= 0;
for(int i=0;i<arr.length;i++){
    for(int j=0;j<arr[i].length;j++){
        arr[i][j] = k;
        k++;
    }
}
```

2차원 배열

2차원 배열 값 출력

1. 인덱스를 이용한 직접 값 출력

```
int [][] arr = new int[2][2];
arr[0][0] = 1;
arr[0][1] = 2;
arr[1][0] = 3;
arr[1][1] = 4;
System.out.println(arr[0][1]);
```



2. for문을 이용한 값 출력

```
for(int i=0;i<arr.length;i++){
    for(int j=0;j<arr[i].length;j++){
        System.out.print(arr[i][j]+ “ ”);
    }
    System.out.println();
}
```

가변배열

가변배열

가변 배열

→ 2차원 배열 선언 시 마지막 열크기를 지정하지 않고, 추 후 각기 다른 길이의 배열을 생성함으로써, 고정된 형태가 아닌 보다 유동적인 가변 배열을 구성할 수 있다.
→ 각 행 별로 열의 길이가 다르다는 것 이외에는 2차원 배열과 동일

자료형[][] 변수명 = new 자료형[행크기][];

```
int [][] arr = new int[3][];
arr[0] = new int[3];
arr[1] = new int[2];
arr[2] = new int[4];
```

가변배열

가변 배열의 저장 구조