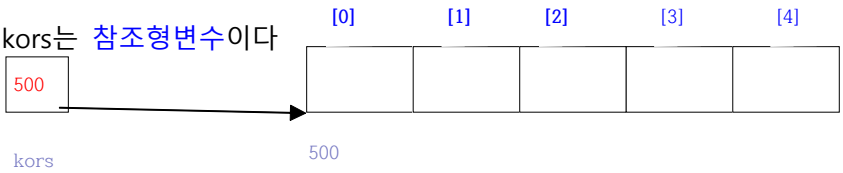
■배열 (기억장소의 집합, 자료형이 동일, 기억장소가 연속적으로 확보됨)

많은 기억장소가 필요로 할 때 배열사용 고려
(변수를 1000개 선언하고 사용하는 것은 불편)
입력과 처리가 불일치 할 때
(예) 50명 학생의 성적에 대해 총점 평균을 구하고 성적순으로 정렬한 다음 출력하시오

배열선언 (반드시 개수정보를 알아야 함)
int[] **kors** = **new** int[5];

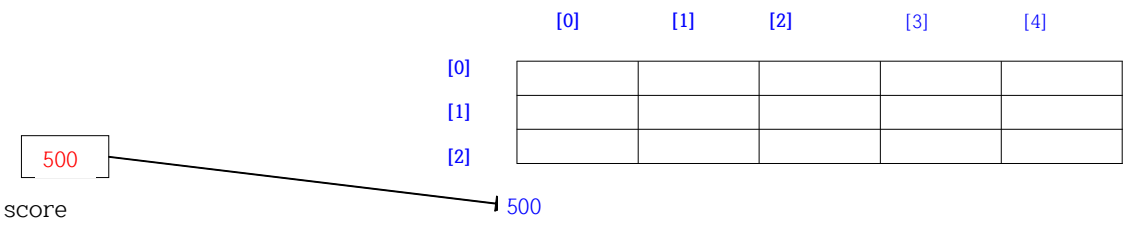


배열요소 접근시 첨자(index)사용
기준으로부터 떨어진 거리(offset)의미
0 1 2의 의미:
자료형의 크기 0만큼 떨어진 위치
자료형의 크기 1만큼 떨어진 위치
자료형이 크기 2만큼 떨어진 위치 의미함

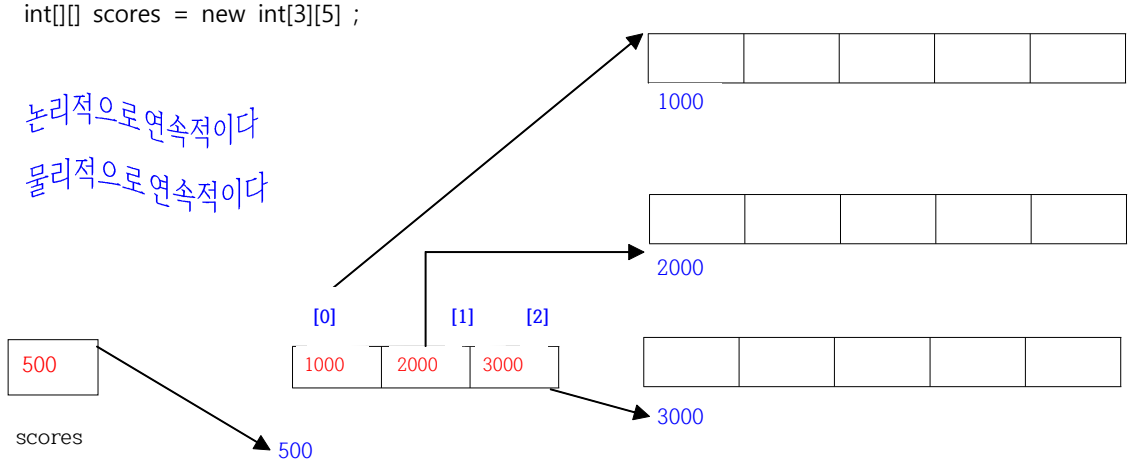
배열의 요소 한 개: 기억정소 한 개와 동일함

kors[0]=100
kors[1]=90
kors[2]=70
kors[3]=90
kors[4]=95

2차원 배열
각 학생의 성적 국어 영어 수학을 2차원 배열로 관리



int[][] scores = new int[3][5] ;



//0행 국어점수 배열
scores[0][0]=80 scores[0][1]=90 scores[0][2]=100 scores[0][3]=70 scores[0][4]=80

//1행 영어점수 배열
scores[1][0]=80 scores[1][1]=80 scores[1][2]=100 scores[1][3]=80 scores[1][4]=100

//2행 수학점수 배열
scores[2][0]=80 scores[2][1]=100 scores[2][2]=100 scores[2][3]=90 scores[2][4]=90

■배열과 반복문
1차원 배열과 반복문
2차원 배열과 다중 반복문 (중첩반복문)

문제) 국어점수 5개 입력후 평균 및 학생점수 모두 출력하시오
문제) 5행 5열의 배열을 만들고 1~25까지 저장한 후 모든 배열의 요소를 출력하시오
(단 입력과 출력을 분리하시오)

메모리

(메모리사용량이 정해지는 시점을 기준으로 분류했을 때)

정적메모리 : 컴파일 시점에 메모리 사용량이 정해질 때

동적메모리 : 실행시 메모리의 사용량이 정해질 때

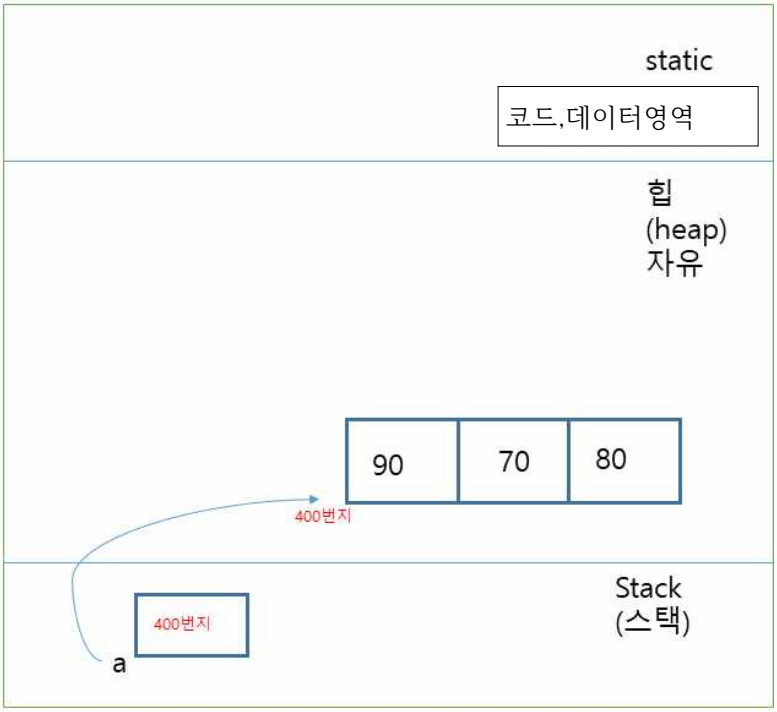
(프로그램을 실행시켜봐야 메모리를 얼마나 사용할지가 결정됨)

```
//정적으로 메모리 쓰는 방법 (컴파일시점에 메모리를 쓰는 양이 정해짐)
                int a1,b1,c1,d1,f1;
                int a2,b2,c2,d2,f2;
                int a3,b3,c3,d3,f3;
                int a4,b4,c4,d4,f4;
                int a5,b5,c5,d5,f5;
                int a6,b6,c6,d6,f6;
                int a7,b7,c7,d7,f7;

                // int arr[100] ;
                // 이런 표현이 아예안된다. 자바는 정적인 형태로 배열생성 못함
                //동적으로 메모리를 쓰는 방법: 실행시에 배열의 크기가 결정



                System.out.println("배열의 크기를 입력하세요 : ");
                Scanner sc = new Scanner(System.in);
                int cnt = sc.nextInt();
                int  arr[];
                arr = new int[cnt];
                System.out.println( arr.length);


                for(int i=0; i< arr.length ; i++)
                {
                        System.out.print( i+1 +"번째 입력:  ");
                        arr[i]= sc.nextInt();
                }

                for(int i=0; i< arr.length ; i++)
                        System.out.println( arr[i]);

        }
}
```

동적메모리 요청방식
new 키워드 이용해서
메모리요청함

```
int[]  a = new int[3];

      a[0]=90;
      a[1]=70;
      a[2]=80;
```

메모리 해제는 직접하지 않고 JVM에서 해 줌
(가비지 컬렉터)



자바는 배열부터 동적메모리 강제한다

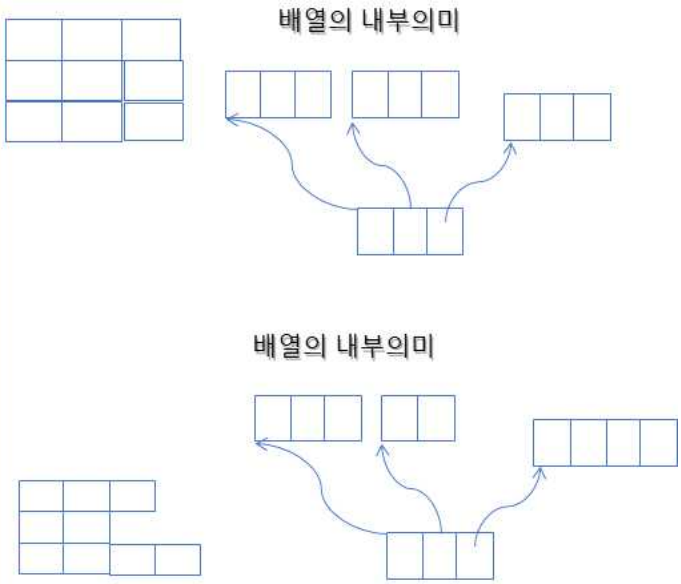
2차원배열:

1.고정길이 배열선언

```
int[][] a;
a = new [3][3];
```

2. 가변길이 배열선언

```
int[][] a;
a= new[3][ ];

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



■메모리사용
-static
-heap
-stack
영역을 나눠서 사용함 (메모리사용의 효율을 높이기 위해 영역을 나눠서 사용함

static : 코드, 데이터 , static변수 (전역영역) , 프로그램의 시작과 끝까지 함께 사용됨

heap : 동적메모리 , new에 의해 메모리 요청함
(프로그래머에 의해 요청과 반납이 이루어지는 영역임) , (즉시반납)

stack : 지역변수사용 (자동영역) ,요청과 반납이 자동으로 이루어짐 (자동반납)

메모리요청
메모리반납

heap 메모리영역

c언어

```
int *korp = malloc( sizeof(int) *4 ) ;
free(korp);
```

c++

```
int *korp = new int[4];
delete[]  korp;
```

자바

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

메모리해제는 별도로 하지 않음 (JVM이 대신해줌, Garbage collector) :참조를 잃은 공간이 대상이됨