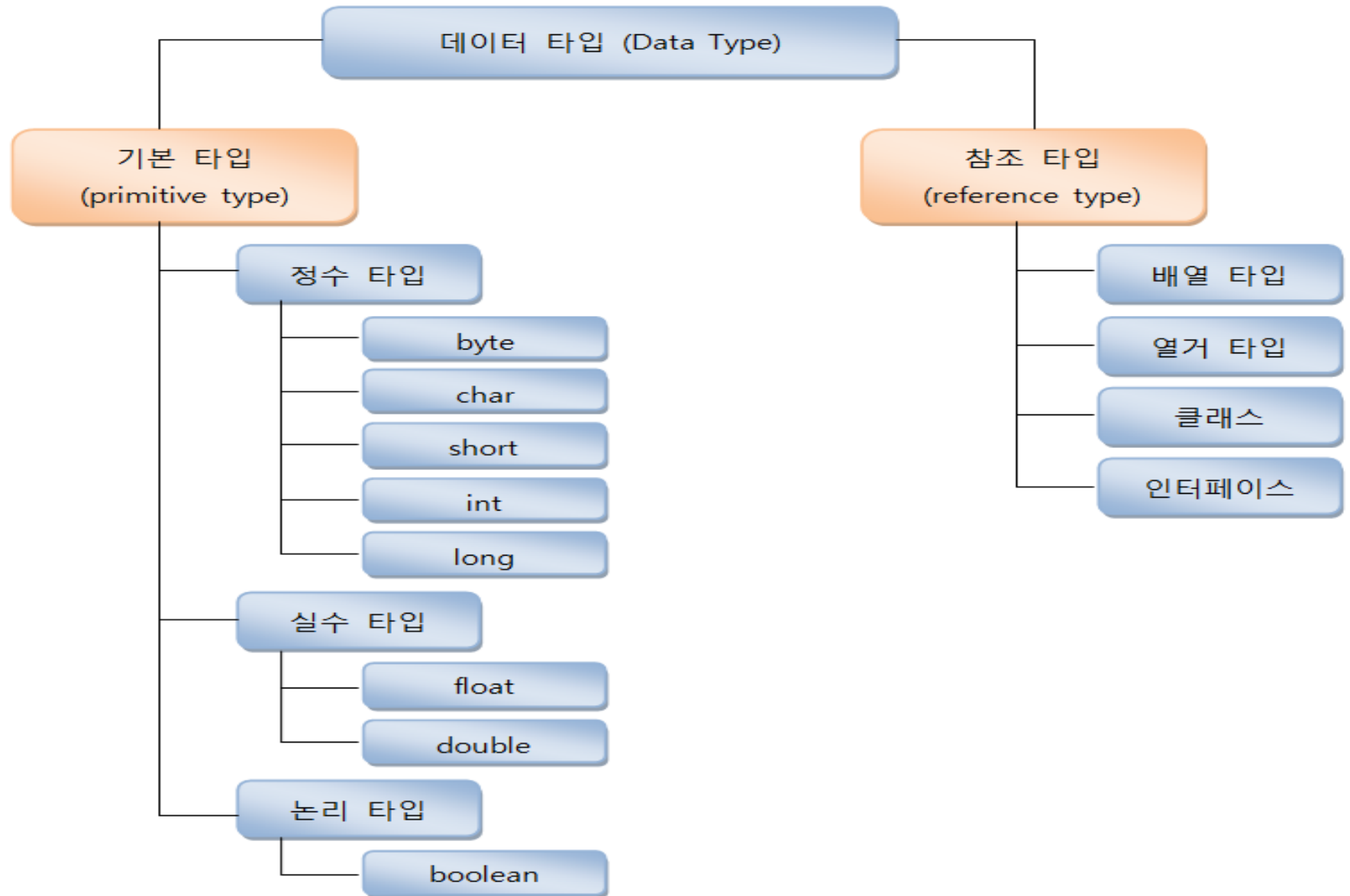


참조타입, 배열

강사 : 강병준

데이터 타입 분류



데이터 타입 분류

변수의 메모리 사용

- 기본 타입 변수 - 실제 값을 변수 안에 저장
- 참조 타입 변수 - 주소를 통해 객체 참조

[기본 타입 변수]

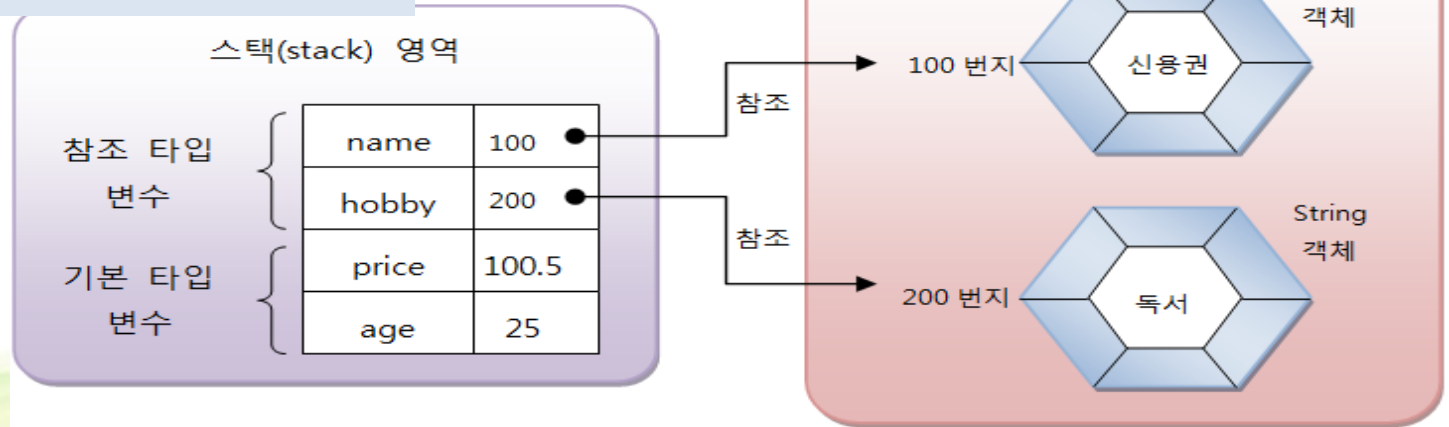
```
int age = 25;
```

```
double price = 100.5;
```

[참조 타입 변수]

```
String name = "신용권";
```

```
String hobby = "독서";
```

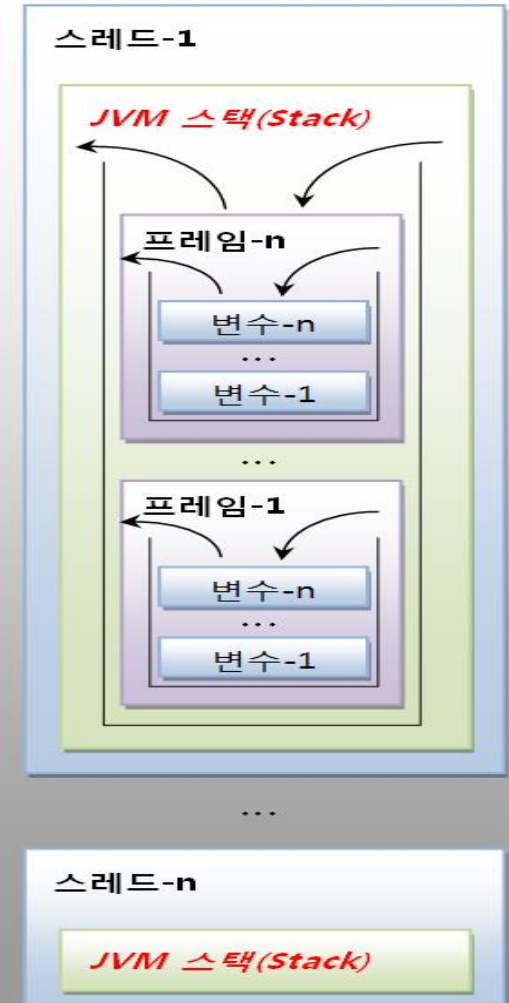
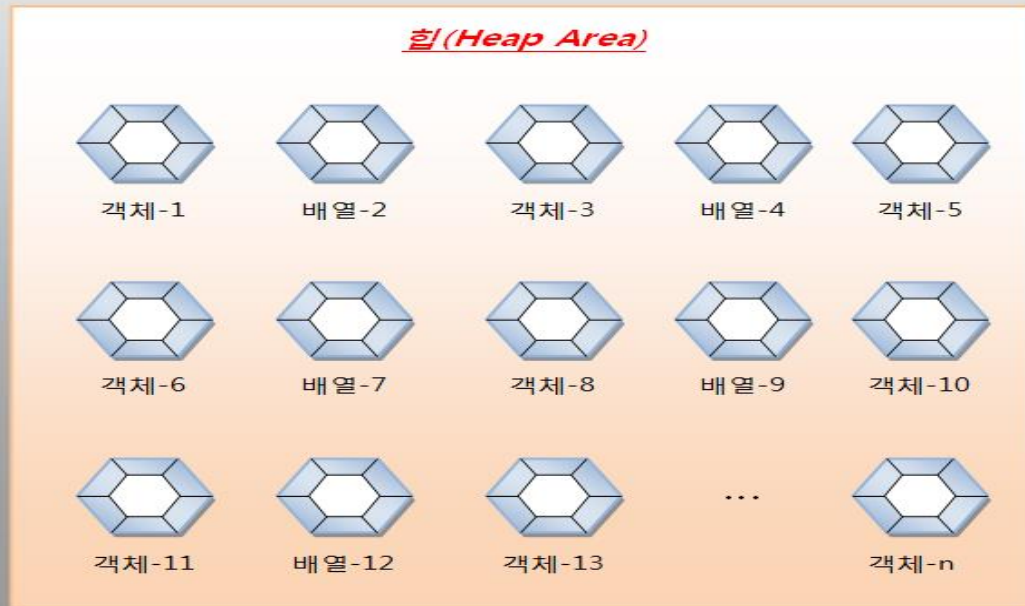
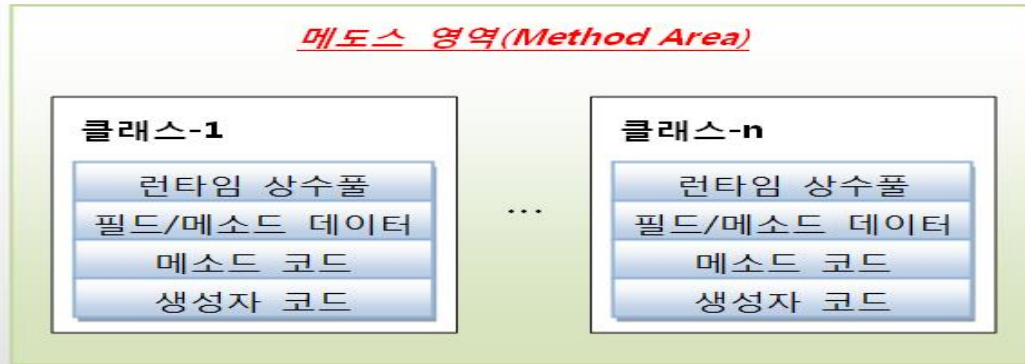


메모리 사용 영역

JVM이 사용하는 메모리 영역

- OS에서 할당 받은 메모리 영역(Runtime Data Area)을 세 영역으로 구분

Runtime Data Area



2절. 메모리 사용 영역

JVM이 사용하는 메모리 영역

– 메소드 영역

- JVM 시작할 때 생성
- 로딩된 클래스 바이트 코드 내용을 분석 후 저장
- 모든 스레드가 공유

– 힙 영역

- JVM 시작할 때 생성
- 객체/배열 저장
- 사용되지 않는 객체는 Garbage Collector 가 자동 제거

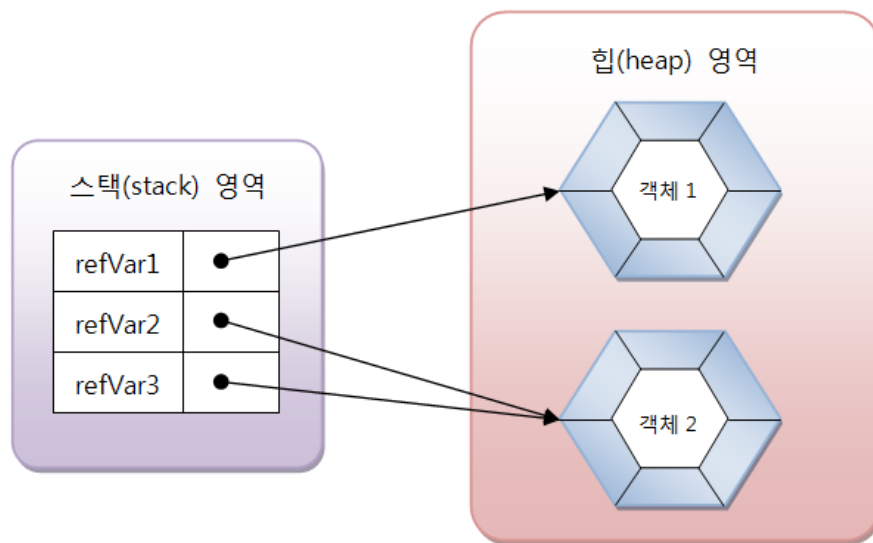
– JVM 스택

- 스레드 별 생성
- 메소드 호출할 때마다 Frame을 스택에 추가(push)
- 메소드 종료하면 Frame 제거(pop)

참조 변수의 ==, != 연산

변수의 값이 같은지 다른지 비교

- 기본 타입: byte, char, short, int, long, float, double, boolean
 - 의미 : 변수의 값이 같은지 다른지 조사
- 참조 타입: 배열, 열거, 클래스, 인터페이스
 - 의미 : 동일한 객체를 참조하는지 다른 객체를 참조하는지 조사



refVar1 == refVar2	결과: false
refVar1 != refVar2	결과: true

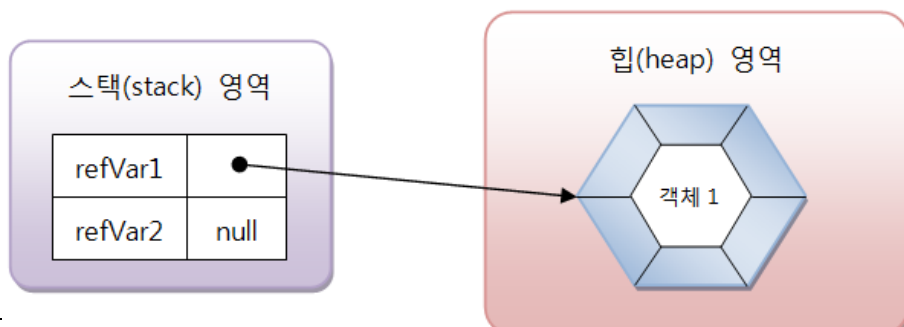
refVar2 == refVar3	결과: true
refVar2 != refVar3	결과: false

```
if( refVar2 == refVar3 ) { ... }
```

null과 NullPointerException

null(널)

- 변수가 참조하는 객체가 없을 경우 초기값으로 사용 가능
- 참조 타입의 변수에만 저장가능
- null로 초기화된 참조 변수는 스택 영역 생성



그림에서 refVar1은 힙 영역의 객체를 참조하므로 연산의 결과는 다음과 같다.

refVar1 == null	결과: false
refVar1 != null	결과: true

refVar2는 null 값을 가지므로 연산의 결과는 다음과 같다.

refVar2 == null	결과: true
refVar2 != null	결과: false

null과 NullPointerException

NullPointerException의 의미

- 예외(Exception)
 - 사용자의 잘못된 조작 이나 잘못된 코딩으로 인해 발생하는 프로그램 오류
- NullPointerException
 - 참조 변수가 null 값을 가지고 있을 때
 - 객체의 필드나 메소드를 사용하려고 했을 때 발생

```
int[] intArray = null;  
intArray[0] = 10;      //NullPointerException
```

```
String str = null;  
System.out.println("총 문자수: " + str.length()); //NullPointerException
```


배열이란?

● 배열

- 같은 형의 데이터들을 다수 개 저장할 수 있는 기억장소를 의미
- 하나의 이름으로 다수 개의 데이터를 사용할 수 있음
- 학생 10명 성적의 평균을 구하는 프로그램을 작성시 많은 변수를 사용하여 프로그램을 복잡하게 하여 오류를 발생시킬 수 있다.

```
int cskim, tbpark, seoh, phkim, cmkim, sdkim, cslee, hjkim, kylee, dhlee;  
int avg; cskim=99;
```

.....생략.....

```
dhlee=85;  
avg = cskim + tbpark + ....생략.... + dhlee / 10; System.out.println("과목  
평균 : " + avg);
```

배열

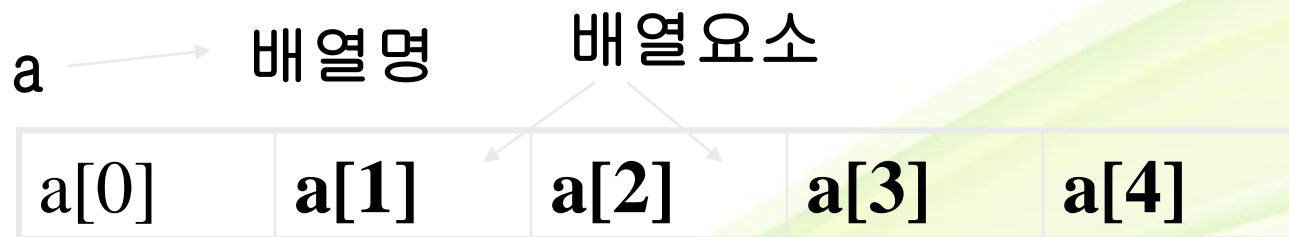
같은 종류의 데이터를 저장하기 위한 자료구조

배열을 객체로 취급

배열을 선언할때 배열의 크기를 명시하지 않는다.

[]는 앞에 오나 뒤에 오나 상관없다.

어떤 형으로 사용할 것인가를 정하고 배열변수를 정해준다.



자바에서의 배열순서

배열의 선언 ➡ 배열의 메모리 할당(배열의 생성) ➡ 배열 요소의 이용

학생 5명의 성적을 저장해야 한다면

예 `int s1, s2, s3, s4, s5;`

만약 전교생(10000명)의 저장해야 한다면



배열

- 자료형이 동일한 여러 개의 값을 연이어 저장할 수 있도록 하는 기억 공간의 집합체(모임)

원소(Element)

- 배열에 저장된 각각의 값

색인, 인덱스 : index

- 배열의 원소에 접근하기 위한 첨자
- 만일 배열의 이름이 a 라면 배열 원소는 $a[0]$, $a[1]$, $a[2]$, ... 로 표시

1차원 배열

5개의 정수형을 저장하는 배열 선언

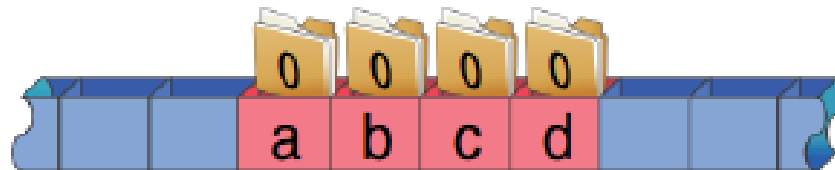
예

```
int [ ] a = new int [ 5 ];   int a[ ] = new int[5];  
①      ②      ③
```

- ① 각 원소에 저장할 값에 대한 자료형
- ② 배열명
- ③ 배열의 원소의 개수

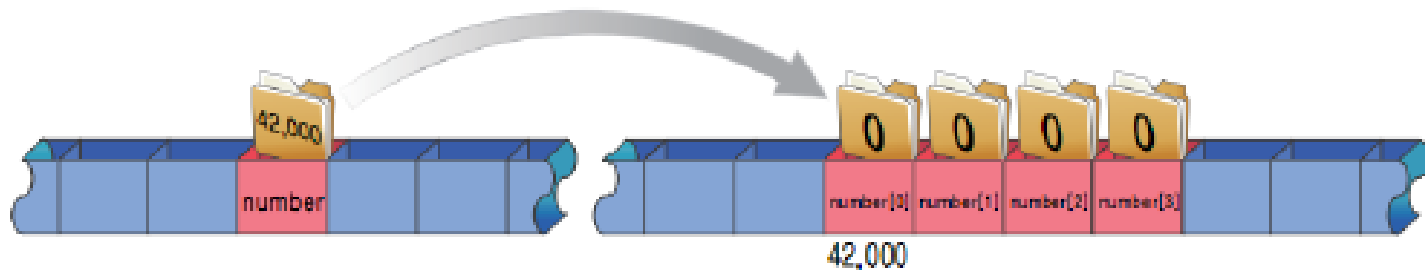
- 변수를 사용하는 경우와 배열을 사용하는 경우

```
int a = 0, b = 0, c = 0, d = 0
```



(a) 변수를 사용

```
int [ ] number = new int [4]
```



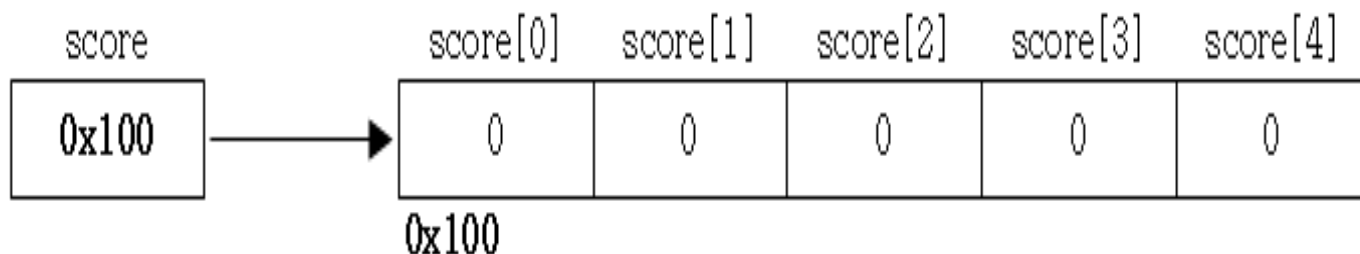
(b) 배열을 사용

배열의 선언과 생성

- 배열을 선언한다고 해서 값을 저장할 공간이 생성되는 것이 아니라 배열을 다루는데 필요한 변수가 생성된다.

```
int[] score;           // 배열을 선언한다. (생성된 배열을 다루는데 사용될 참조변수 선언)  
score = new int[5];    // 배열을 생성한다. (5개의 int값을 저장할 수 있는 공간생성)
```

[참고] 위의 두 문장은 `int[] score = new int[5];`와 같이 한 문장으로 줄여 쓸 수 있다.



배열요소의 이용

▶ 배열요소의 이용

배열요소는 배열변수명과 인덱스 두 가지를 이용한다.

※인덱스란? 배열의 맨 처음부터의 위치이다. 주의할 점은 자바에서의 인덱스는 0부터 시작한다는 것이다.

배열 변수명[인덱스]

Name[0]=3;



1차원 배열 초기값

초기 값을 콤마로 구분하여 여러 번 기술하고 이들을 중괄호 { }로 감싸 준다.

```
예 int [ ]a={10, 20, 30, 40, 50};
```

배열의 원소 값들을 출력하기

```
예 for(i=0; i<a.length; i++)  
    System.out.println( " a[ " + i + " ] = " +a[i]);
```

a.length

배열의 개수를 알아 낼 수 있다.

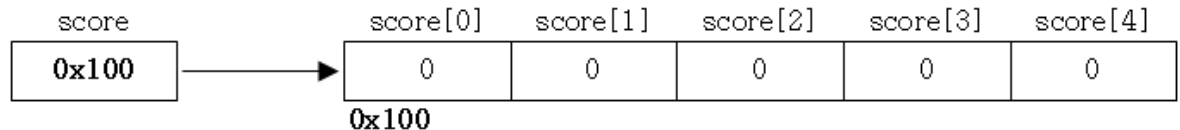
배열 선언 후 값 대입과 원소 출력하기

```
public class Ex01 {  
    public static void main(String[] args) {  
        int []a=new int [5];  
        int i;  
        a[0]=50;  
        a[1]=100;  
        a[2]=150;  
        a[3]=200;  
        a[4]=250;  
  
        for(i=0; i<5; i++)  
            System.out.println( (i+1)+" th a[" + i + "]= " +a[i]);  
    }  
}
```

배열의 초기화

- 생성된 배열에 처음으로 값을 저장하는 것

```
int[] score = new int[5]; // 크기가 5인 int형 배열을 생성한다.  
score[0] = 100;           // 각 요소에 직접 값을 저장한다.  
score[1] = 90;  
score[2] = 80;  
score[3] = 70;  
score[4] = 60;
```



```
int[] score = { 100, 90, 80, 70, 60}; // 1번
```

```
int[] score = new int[]{ 100, 90, 80, 70, 60}; // 2번
```

```
int[] score;
```

```
score = { 100, 90, 80, 70, 60}; // 에러 발생!!!
```

```
int[] score;
```

```
score = new int[]{ 100, 90, 80, 70, 60}; // OK
```

배열의 활용


- ▶ 배열에 값을 저장하고 읽어오기

```
score[3] = 100;    // 배열 score의 4번째 요소에 100을 저장한다.  
int value = score[3]; // 배열 score의 4번째 요소에 저장된 값을 읽어서 value에 저장.
```

- ▶ '배열이름.length'는 배열의 크기를 알려준다.

```
int[] score = { 100, 90, 80, 70, 60, 50 };
```

```
for(int i=0; i < 6; i++) {  
    System.out.println(score[i]);  
}
```



```
for(int i=0; i < score.length; i++) {  
    System.out.println(score[i]);  
}
```

차원 배열의 초기화

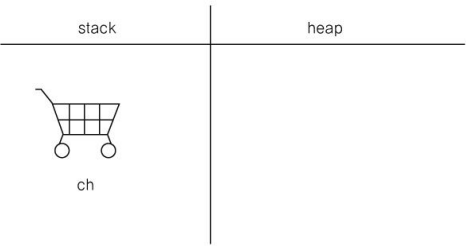
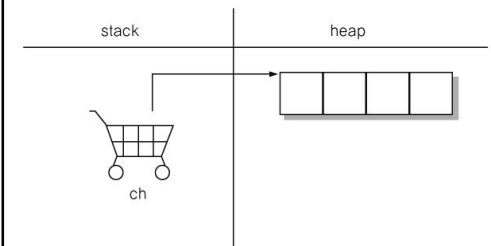
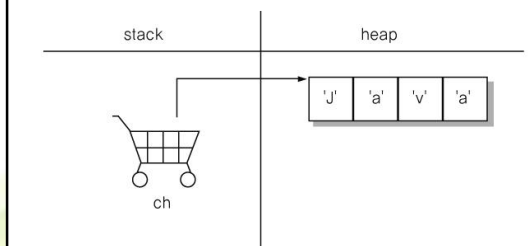
```
public class Ex02 {  
    public static void main(String[] args) {  
        int []a ={85, 90, 75, 100, 95};  
        int tot=0;  
        double avg;  
        int i;  
        for(i=0; i<5; i++)  
            tot+=a[i];  
  
        avg = (double)tot/5.0;  
        System.out.println(" 총합 = " + tot);  
        System.out.println(" 평균 = " + avg);  
    }  
}
```

배열 예제(1차원배열)

```
class ArrayTest {  
    public static void main(String[] arg){  
        int array[];  
        array = new int[10];  
  
        for(int i=0; i<10; i++)  
            array[i]=i+1;  
        for(int i=0; i<10; i++)  
            System.out.println "["+i+"]"+array[i]);  
    }  
}
```

배열의 메모리

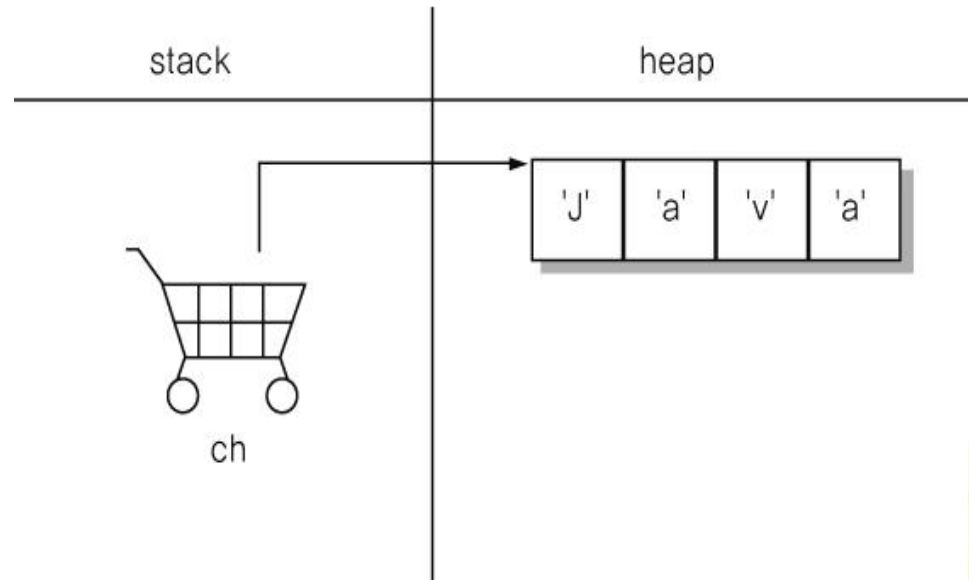
❖ 배열의 단계적 메모리 할당

1) 배열 선언 <code>char[] ch; 또는 char ch[];</code>	2) 배열 생성 <code>ch = new char[4];</code>	3) 배열 초기화 <code>ch[0]='J'; ch[1]='a'; ch[2]='v'; ch[3]='a';</code>
 <p>[그림 4-57] 배열 생성 1단계(선언)</p>	 <p>[그림 4-58] 배열 생성 2단계(생성)</p>	 <p>[그림 4-59] 배열 생성 3단계(초기화)</p>

배열의 메모리

1차원 배열

```
class ArrayEx1{  
    public static void main(String[] args){  
        char[] ch; // 배열 선언  
        ch = new char[4]; // 배열 생성  
        // 배열 초기화  
        ch[0] = 'J';  
        ch[1] = 'a';  
        ch[2] = 'v';  
        ch[3] = 'a';  
        // 배열 내용 출력  
        for(int i = 0 ; i < ch.length ; i++) {  
            System.out.println("ch[ "+i+" ] : "+ch[ i ] );  
        }  
    }  
}
```



배열의 메모리

● 객체형 배열

- 객체형 배열은 객체를 가리킬 수 있는 참조 값(주소)들의 묶음이다. 실제 값이 저장된 기본 자료 형과는 다르게 객체의 reference들의 집합인 것이다.
- 객체형 배열은 집집마다 우편함을 한곳에 모아둔 것과 같다. 각 우편함들은 나름대로 특정한 가정이라는 객체(Object)의 주소를 대신하는 것을 의미하며 이들의 묶음(집합)이 곧 reference배열 또는 객체형 배열이라 한다

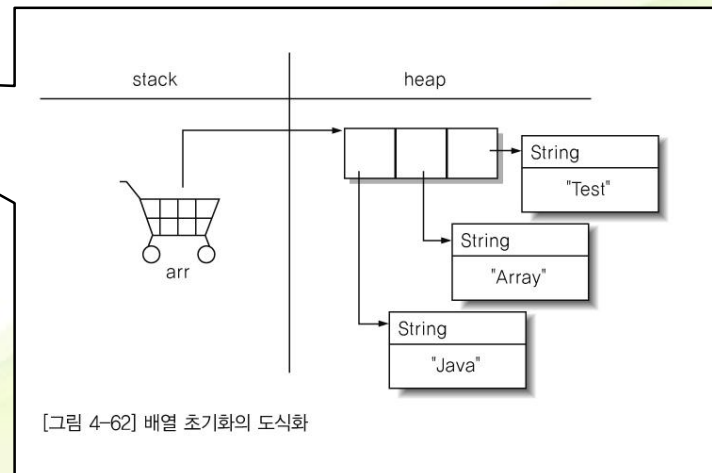
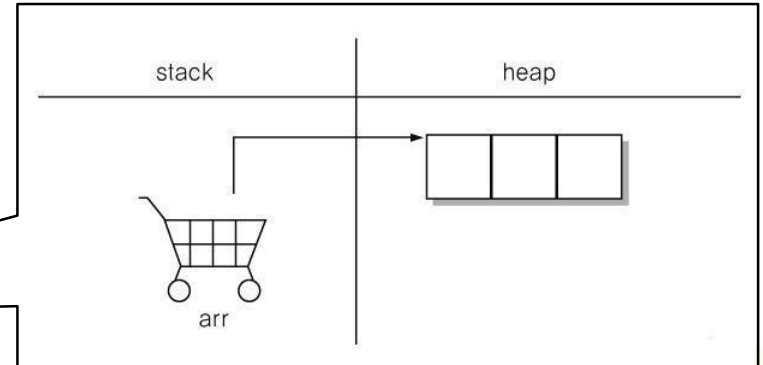
배열의 메모리

```
class ObjArrayEx1 {  
    public static void main(String[] args){
```

```
        String[] arr;  
        arr = new String[3];
```

```
        arr[0] = "Java ";  
        arr[1] = "Array ";  
        arr[2] = "Test";
```

```
    }  
}
```



배열 예제 (1차원 배열)

```
class ArrayTest2 //직접할당 {  
    public static void main(String[] arg){  
        int array[]={1,2,3,4,5,6,7,8,9,10};  
        for(int i=0; i<array.length; i++)  
            System.out.println("array["+i+"] :"+array[i]);  
    }  
}
```

배열 예제 (1차원 배열)

문제 1) 76, 45, 34, 89, 100, 50, 90, 92 8개의 값을 1차원 배열로 초기화하고 값에 합계와 평균 그리고 최대값과 최소값을 구하는 프로그램을 작성 하세요.

결과값 :

합계 = 576 평균 = 72

최대값은 = 100 최소값은 = 34

배열 예제 (1차원 배열)

//합계와 평균 최대값 최소값 구하기...

```
class ArrayTest3 {  
    public static void main(String[] arg){  
        int array[]={76,45,34,89,100,50,90,92};  
        int sum=0,avg=0,max=0,min=100;  
        for(int i=0; i<array.length; i++){  
            sum+=array[i];  
            if(array[i]>max) max=array[i];  
            if(array[i]<min) min=array[i];  
        }  
        avg=sum/array.length;  
        System.out.println("합계 = "+sum+"    평균 = "+avg);  
        System.out.println("최대값은 = "+max+"    최소값은 = "+min);  
    }  
}
```

정렬(선택정렬)

문제2) 76,45,34,89,100,50,90,92 8개의 값을 1차원 배열로 초기화 하고 이들 값들을 크기 순으로 나타내는 프로그램을 작성 하시오.

결과값 : 34 45 50 76 89 90 92 100

❖정렬은 데이터를 순서대로 나열하는 것을 의미합니다.

❖정렬방식은 작은 것부터 큰 순서대로 나열하는 오름차순 정렬과 큰 것에서 작은 순서대로 나열하는 내림차순 정렬이 있고 정렬을 하는 방법은 여러 가지가 있습니다.

❖선택정렬은 첫 번째 자리부터 마지막에서 두 번째 자리까지 자신보다 뒤에 있는 모든 자리들과 비교해서 다음 자료가 작으면 2개의 요소의 자리를 변경해주면 됩니다.

❖ 초기 상태 50 40 10 20 30

1Pass 10 50 40 20 30

2Pass 10 20 50 40 30

3Pass 10 20 30 50 40

4Pass 10 20 30 40 50

❖1번째 자리를 기준으로 2,3,4,5 번째 자리와 비교

❖1번째 자리는 제외하고2번째 자리를 기준으로 3,4,5 번째 자리와 비교

❖3번째 자리를 기준으로 4,5 번째 자리와 비교

❖4번째 자리를 기준으로 5번째 자리와 비교

❖2개의 요소 자리 변경(SWAP)

❖a 와 b 의 swap

temp = a; a = b; b = temp;

❖임시 변수를 만들어서 먼저 1 번째 자료의 값을 임시 변수에 넣고 2번째 자료의 값을 1 번째 자리로 복사하고 임시 변수에 들어있는 내용을 2번째 변수에 복사해 주면 됩니다.

배열 예제 (1차원 배열)

//오름차순 프로그램 작성

```
class ArrayTest4{
    public static void main(String[] arg){
        int array[]={76,45,34,89,100,50,90,92};
        int a=0;
        for(int i=0; i<array.length; i++){
            for(int j= i + 1; j<array.length; j++){
                if(array[i]>array[j]){
                    a=array[i];
                    array[i]=array[j];
                    array[j]=a;
                }
            }
        }
        for(int i=0; i<array.length; i++){
            System.out.print("  "+array[i]);
        }
    }
}
```

정렬(버블정렬)

- ❖ 버블 정렬은 n 개의 데이터가 있을 때 1부터 $n-1$ 번째 자료까지 $n-1$ 번 동안 다음 자료와 비교해가면서 정렬하는 방법입니다.
- ❖ 버블 정렬의 효과를 높이기 위해서는 비교 시 횟수만큼 빼면서 정렬하면 성능을 높일 수 있습니다.

시작

7	4	11	9	2
---	---	----	---	---

4	7	11	9	2
---	---	----	---	---

4	7	11	9	2
---	---	----	---	---

4	7	9	11	2
---	---	---	----	---

4	7	9	2	11
---	---	---	---	----

1 패스 완료 (4번 비교)

4	7	9	2	11
---	---	---	---	----

4	7	9	2	11
---	---	---	---	----

4	7	2	9	11
---	---	---	---	----

2 패스 완료 (3번 비교)

4	7	2	9	11
---	---	---	---	----

4	2	7	9	11
---	---	---	---	----

3 패스 완료 (2번 비교)

2	4	7	9	11
---	---	---	---	----

4 패스 완료 (1번 비교)

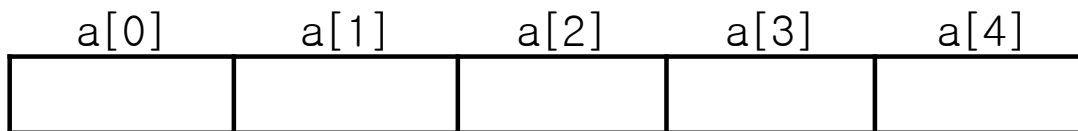
정렬(버블정렬)

```
public class BubbleSort {  
    public static void main(String[] args) {  
        int[] a = {76,45,34,89,100,50,90,92};  
        int temp;  
        for(int i = 0; i < a.length - 1; i++) {  
            for (int j=0; j < a.length-(i+1); j++) {  
                if (a[j]>a[j+1]) {  
                    temp = a[j];  
                    a[j] = a[j+1];  
                    a[j+1] = temp;  
                }  
            }  
        }  
        for(int i : a) {  
            System.out.print(i+"\t");  
        }  
    }  
}
```

2차원 배열

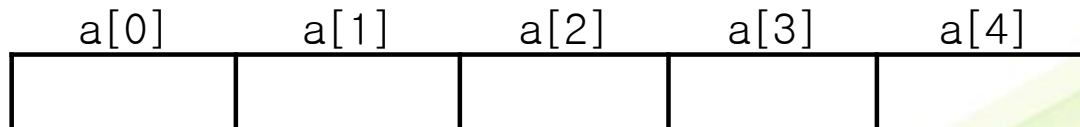
1차원 배열

예 `int [] a = new int [5];`

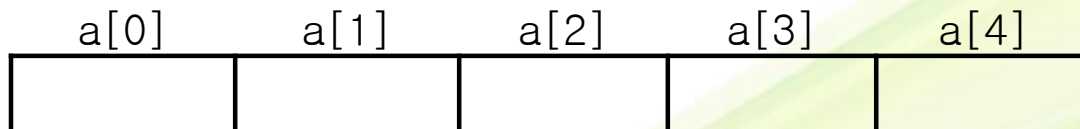


2차원 배열은 1차원 배열을 여러 개 묶어서 사용한 형태로 이해

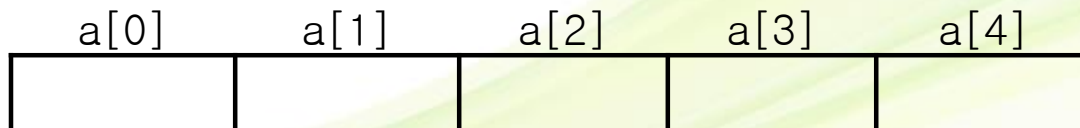
1반



2반

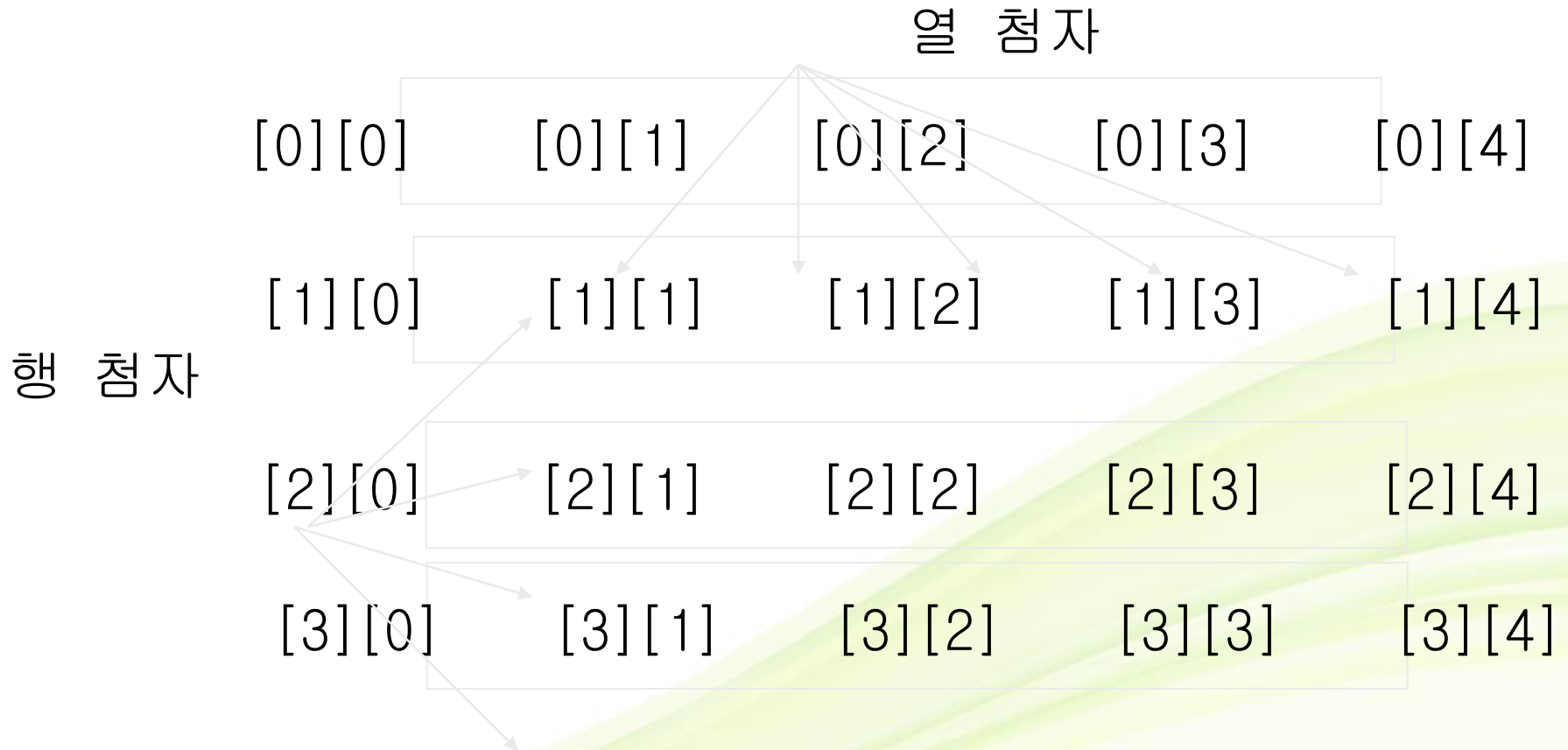


3반



다차원 배열 선언

- 데이터형 배열변수명[][]; 혹은 데이터형[][] 배열변수명;
int name[][]; 또는 int[][] name; (권장)



배열의 생성

배열의 생성

```
array-name = new type[size];
```

예 : `int name[][] = new int[4][5];`(정적)

`int name[][] = new int[3][];`(동적)

※ `int name[5]`, `int name = new int[][3]` 은 가능하지 않다.

직접: `int[] name = {1,2,3,4}`, `int[][] name={{1,2},{2,3}}`

Tip : 자바의 배열은 한번 생성된 후에는 크기가 변할 수 없다

※ `name`은 정수형 데이터 5개를 갖고 있는 배열의 레퍼런스 변수이다.

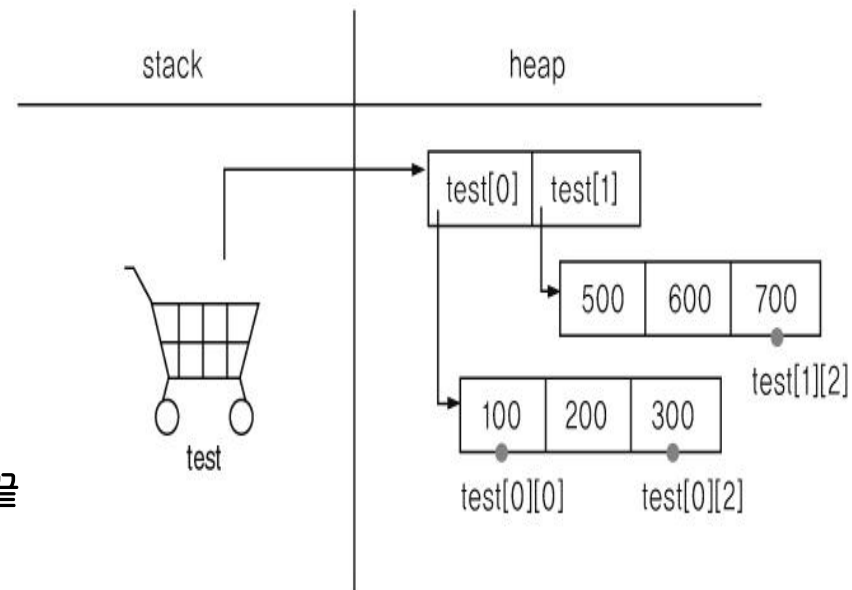
—레퍼런스 변수란? 기본 데이터를 값으로 갖는 것이 아니고, 메모리상이 다른 객체나 배열을 가리키고 있는 변수이다.

배열

다차원 배열

1차원 배열이 여러 개 모인 것을 다차원 배열이라 한다.

```
class ObjArrayEx4 {  
    public static void main(String[] args){  
        int[][] test; // 다차원 배열 선언  
        test = new int[2][3];  
        test[0][0] = 100;  
        test[0][1] = 200;  
        test[0][2] = 300;  
        //----- 1행 초기화 끝  
        test[1][0] = 500;  
        test[1][1] = 600;  
        test[1][2] = 700;  
        //----- 2행 초기화 끝  
    }  
}
```



[그림 4-68] 다차원 배열의 구조와 접근 index값

배열

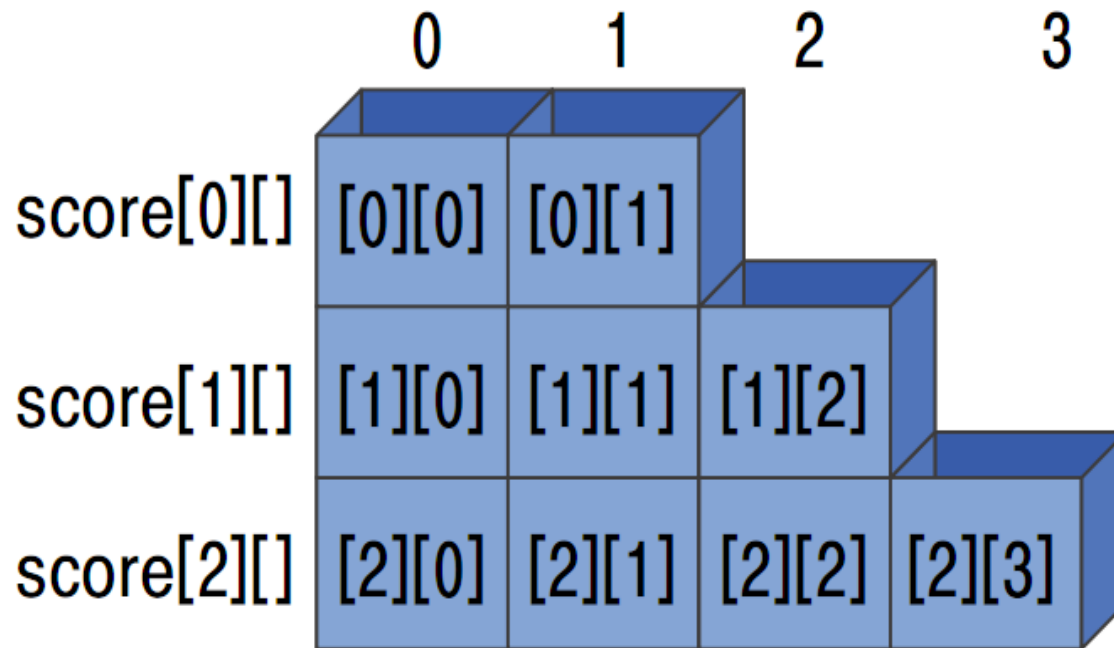
```
class ArrayTest1 { // 이차원 배열
    public static void main(String args[]) {
        int a[][]={{1,2,3,4,5},{11,12,13,14,15},{21,22,23,24,25}};
        for(int i=0;i<a.length;i++) {
            for(int j=0; j<a[i].length; j++) {
                System.out.print(a[i][j]+" \t");
            }
            System.out.println(); }
    }
```

```
class ArrayTest2 {
    public static void main(String args[]) {
        int a[][]={{10,20,30},{40,50,60},{70,80,90}};
        int sum=0;
        for(int i=0;i<a.length;i++) {
            for(int j=0;j<a[i].length;j++) {
                System.out.print(a[i][j] + " ");
                sum+=a[i][j];
            }
            System.out.println(" :"+sum);
            sum=0; // 클리어
        }
    }
```

```
class ArrayTest3 {  
    public static void main(String args[]) {  
        double d[][]={{188.8, 166.6},{179.8, 158.6}};  
        double sum=0.0;  
        for(int i=0;i<d.length;i++) {  
            for(int j=0;j<d[i].length;j++) {  
                System.out.print(d[i][j]+" \t");  
                sum+=d[i][j];  
            }  
            System.out.println(" : "+sum+" : "+(double)sum/d[i].length);  
            sum=0.0;  
        }  
    }  
}
```

배열의 선언과 생성

- 각 행의 요소의 개수가 다른 배열의 형태



가변배열

- 다차원 배열에서 마지막 차수의 크기를 지정하지 않고 각각 다르게 지정.

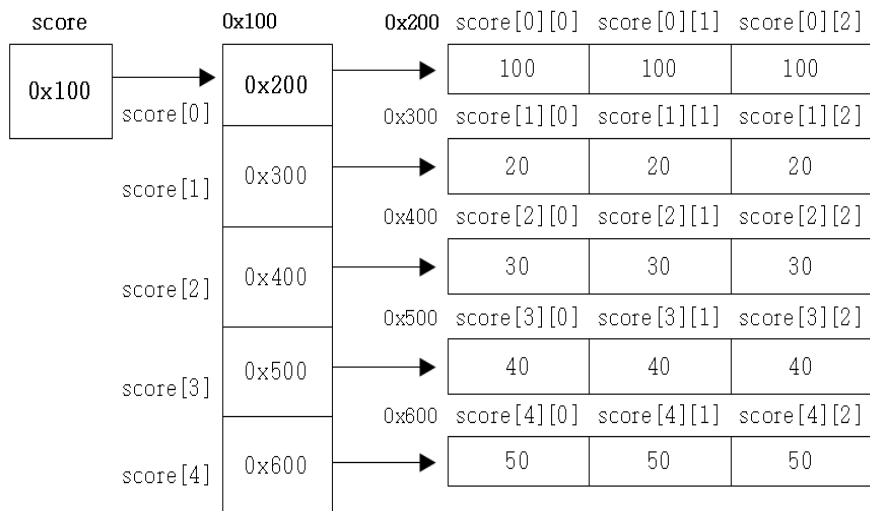
```
int[][] score = new int[5][3];    // 5행 3열의 2차원 배열을 생성한다.
```

```
int[][] score = new int[5][];  
score[0] = new int[3];  
score[1] = new int[3];  
score[2] = new int[3];  
score[3] = new int[3];  
score[4] = new int[3];
```

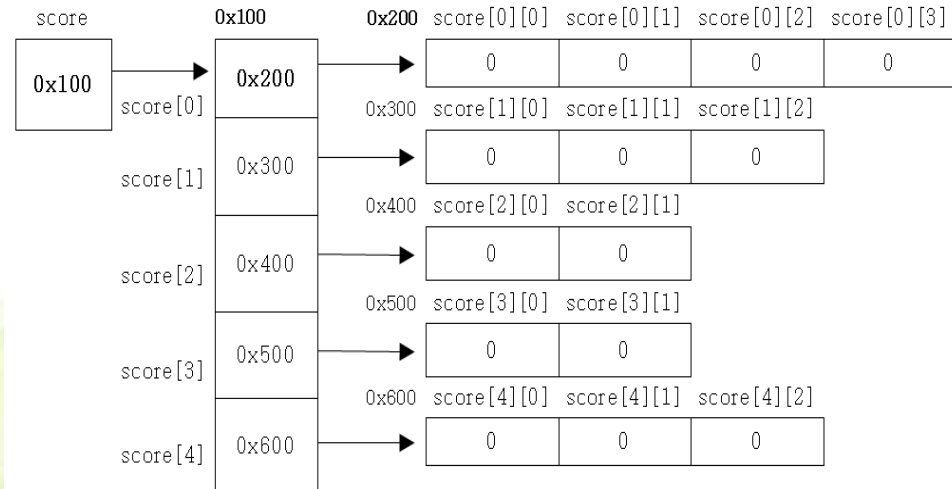
```
int[][] score =  
{  
    {100, 100, 100},  
    { 20,  20,  20},  
    { 30,  30,  30},  
    { 40,  40,  40},  
    { 50,  50,  50},  
};
```

```
int[][] score = new int[5][];  
score[0] = new int[4];  
score[1] = new int[3];  
score[2] = new int[2];  
score[3] = new int[2];  
score[4] = new int[3];
```

```
int[][] score =  
{  
    {100, 100, 100, 100},  
    { 20,  20,  20},  
    { 30,  30},  
    { 40,  40},  
    { 50,  50,  50},  
};
```



【그림 5-2】 2차원 배열



【그림 5-3】 가변배열

배열의 복사

▶ for문을 이용한 배열의 복사

```
int[] number = {1,2,3,4,5};  
int[] newNumber = new int[10];
```

```
for(int i=0; i<number.length;i++) {  
    newNumber[i] = number[i]; // 배열 number의 값을 newNumber에 저장한다.  
}
```

number

1	2	3	4	5
---	---	---	---	---

newNumber

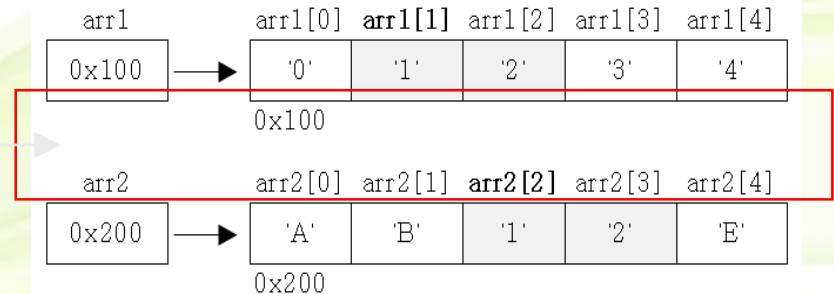
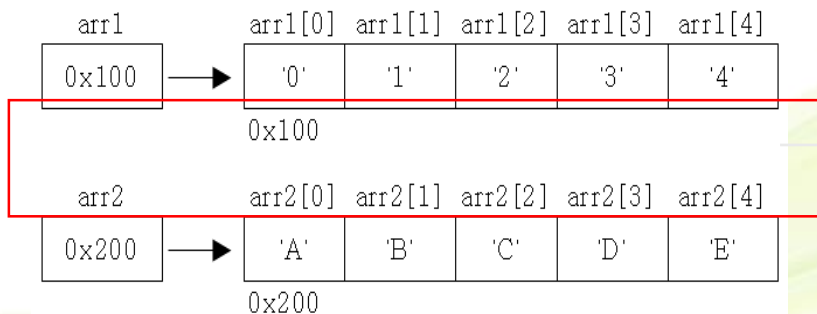
0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---

▶ System.arraycopy()를 이용한 배열의 복사

```
System.arraycopy(arr1, 0, arr2, 0, arr1.length);
```

arr1[0]에서 arr2[0]으로 arr1.length개의 데이터를 복사

```
System.arraycopy(arr1, 1, arr2, 2, 2);
```



배열의 복사

```
class ArrayEx12 {  
    public static void main(String[] args) {  
        char[] abc = { 'A', 'B', 'C', 'D'};  
        char[] number = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};  
        System.out.println(new String(abc));  
        System.out.println(new String(number));  
  
        // 배열 abc와 number를 붙여서 하나의 배열(result)로 만든다.  
        char[] result = new char[abc.length+number.length];  
        System.arraycopy(abc, 0, result, 0, abc.length);  
        System.arraycopy(number, 0, result, abc.length, number.length);  
        System.out.println(new String(result));  
  
        // 배열 abc을 배열 number의 첫 번째 위치부터 배열 abc의 크기만큼 복사  
        System.arraycopy(abc, 0, number, 0, abc.length);  
        System.out.println(new String(number));  
  
        // number의 인덱스6 위치에 3개를 복사  
        System.arraycopy(abc, 0, number, 6, 3);  
        System.out.println(new String(number));  
    }  
}
```

배열의 복사

%. System.arraycopy메소드

```
arraycopy(srcArray,srcStartIndex,dstArray,dstStartIndex,copyLength) ;
```

```
public class ArrayTest6 {
```

```
    public static void main(String[] args) {
```

```
        String cities[] = {"서울","대구","춘천","울산","광주","천안"};
```

```
        String nations[] = new String[]{"대한민국","미국","영국","일본","프랑스"};
```

```
        String newArray[] = new String[cities.length+nations.length];
```

```
        System.arraycopy(cities, 0, newArray, 0, cities.length);
```

```
        System.arraycopy(nations, 0, newArray, cities.length, nations.length);
```

```
        for(String str:newArray){
```

```
            System.out.println(str);
```

```
        }
```

```
    }
```

```
}
```

배열의 복사

```
public class ArrayCopy1 {  
    public static void main(String[] args) {  
        int[] num = {34,56,43,7,76};  
        int[] cpy = new int[7];  
        // cpy = num;  
        // System.arraycopy(num, 0, cpy, 0, num.length);  
        // System.arraycopy(num, 0, cpy, 1, num.length);  
        // System.arraycopy(num,1,cpy,0,num.length-1);  
        System.arraycopy(num,1,cpy,2,2);  
        for (int i=0;i<cpy.length;i++) {  
            System.out.println(cpy[i]);  
        }  
    }  
}
```

사용자 입력받기 - 커맨드라인

```
class ArrayEx13 {  
    public static void main(String[] args) {  
        System.out.println("매개변수의 개수:"+args.length);  
        for(int i=0;i< args.length;i++) {  
            System.out.println("args[" + i + "] = \""+ args[i] + "\"");  
        }  
    }  
}
```

```

class ArrayEx15 {
    public static void main(String[] args) {
        if (args.length != 3) {
            System.out.println("usage: java ArrayEx15 NUM1 OP NUM2");
            System.exit(0);
        }
        int num1 = Integer.parseInt(args[0]); // 문자열을 숫자로 변환한다.
        char op = args[1].charAt(0);          // 문자열을 문자(char)로 변환한다.
        int num2 = Integer.parseInt(args[2]);
        int result = 0;
        switch(op) { // switch문의 수식에는 byte, short, char, int가 올 수 있다.
            case '+': result = num1 + num2;      break;
            case '-': result = num1 - num2;      break;
            case 'x': result = num1 * num2;      break;
            // *는 .classpath를 가르키므로 x로 변경하여 사용하였음
            case '/': result = num1 / num2;      break;
            default : System.out.println("지원되지 않는 연산입니다.");
        }
        System.out.println("결과:"+result);
    }
}

```

성적표

이름	국어	영어	수학	총점	평균
----	----	----	----	----	----

원더걸스	90	80	70		
------	----	----	----	--	--

비스트	76	86	90		
-----	----	----	----	--	--

투피엠	90	78	90		
-----	----	----	----	--	--

소녀시대	80	80	80		
------	----	----	----	--	--

합계					
----	--	--	--	--	--

매출현황

제품명	1월	2월	3월	4월	합계	평균
-----	----	----	----	----	----	----

냉장고	250	170	300	780		
-----	-----	-----	-----	-----	--	--

테레비	170	120	150	220		
-----	-----	-----	-----	-----	--	--

청소기	450	230	400	250		
-----	-----	-----	-----	-----	--	--

총계						
----	--	--	--	--	--	--


```

public class ArrayTest9 {
    public static void main(String[] args) {
        String[] m = {"이름", "국어", "영어", "수학", "총점", "평균"};
        String[] name = {"소녀시대", "이하이", "투엔니원", "비스트"};
        int[][] score = {{80,70,90},{70,90,90},{80,70,80},{90,90,70}};
        int[] tot = new int[3];          int sum = 0, avg = 0;
        System.out.println("성적표 \n");
        for(int i = 0; i<m.length;i++) System.out.print(m[i]+"\\t");
        System.out.println("\\n-----");
        for(int i = 0; i < score.length; i++) {
            System.out.print(name[i] + "\\t");
            for(int j = 0 ; j < score[i].length; j++) {
                System.out.print(score[i][j] + "\\t");
                sum += score[i][j];          tot[j] += score[i][j];
            }
            avg = sum / score[i].length;
            System.out.println(sum + "\\t" + avg);          sum = 0;
        }
        System.out.println("-----");
        System.out.print("총계" + "\\t");
        for(int j = 0 ; j < 3 ; j++) {
            System.out.print(tot[j] + "\\t"); sum += tot[j];
        }
        avg = sum / 12;          System.out.println(sum + "\\t" + avg);
    }
}

```

연습문제

1. 배열에 담긴 값을 더하는 프로그램을 작성

```
int[] arr = { 10, 20, 30, 40, 50}
```

2. 2차원 배열에 저장된 값의 합계를 구하시오

```
int[][] arr = { { 5, 5, 5, 5, 5}, {10, 10, 10, 10, 10},  
                {20, 20, 20, 20, 20}, {30, 30, 30, 30, 30}};
```

3. 거스름돈 2680을 500, 100, 50, 10짜리 동전으로 줄때
몇개씩 주어야 하나

힌트

```
coinUnit[i] + “원 ;” + money/coinUnit[i]  
money = money%coinUnit[i];
```

연습문제

```
import java.util.Scanner;
public class Ex04 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int year = 0;
        do {
            System.out.println("년도를 입력하세요");
            year = sc.nextInt();
            if (year%4==0 && year%100 !=0 || year%400==0)
                System.out.println("윤년입니다");
            else System.out.println("평년입니다");
        } while(year != 0);
        System.out.println("프로그램 종료");
        sc.close();
    }
}
```

연습문제

```
public class Ex05 {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int temp;          int[] lotto = new int[6];  
        for(int i = 0; i < 6; i++) {  
            lotto[i] = (int)(Math.random()*45) + 1;  
            for(int j=i-1;j>=0;j--) {  
                if (lotto[j]==lotto[i]) { i--;    break;    }  
            }  
        }  
        for(int i = 0; i < lotto.length;i++) {  
            for (int j=i+1;j<lotto.length;j++) {  
                if (lotto[i] > lotto[j]) {  
                    temp=lotto[i]; lotto[i]=lotto[j]; lotto[j]=temp;  
                }  
            }  
        }  
        for (int i=0;i < lotto.length;i++) { System.out.println(lotto[i]); }  
        sc.close();  
    }  
}
```

연습문제

```
public class SecArr3 {  
    public static void main(String[] args) {  
        int[][] score = {{67,78,98},{78,98,65},{78,56,90}};  
        int tot=0, max=0, min=100,tmax=0, tmin=100;  
        float avg;  
        for(int[] f : score) {  
            for(int s : f) {  
                tot += s;  
                if (max < s) max = s;  
                if (min > s) min = s;  
                if (tmax < s) tmax = s;  
                if (tmin > s) tmin = s;  
            }  
            avg = tot / 3.0f;  
            System.out.println("총점 : "+tot+", 평균 : "+avg);  
            System.out.println("최고 : "+max+", 최소 : "+min);  
            tot = 0; max = 0; min = 100;  
        }  
        System.out.println("전체최고 : "+tmax+", 전체최소 : "+tmin);  
    }  
}
```

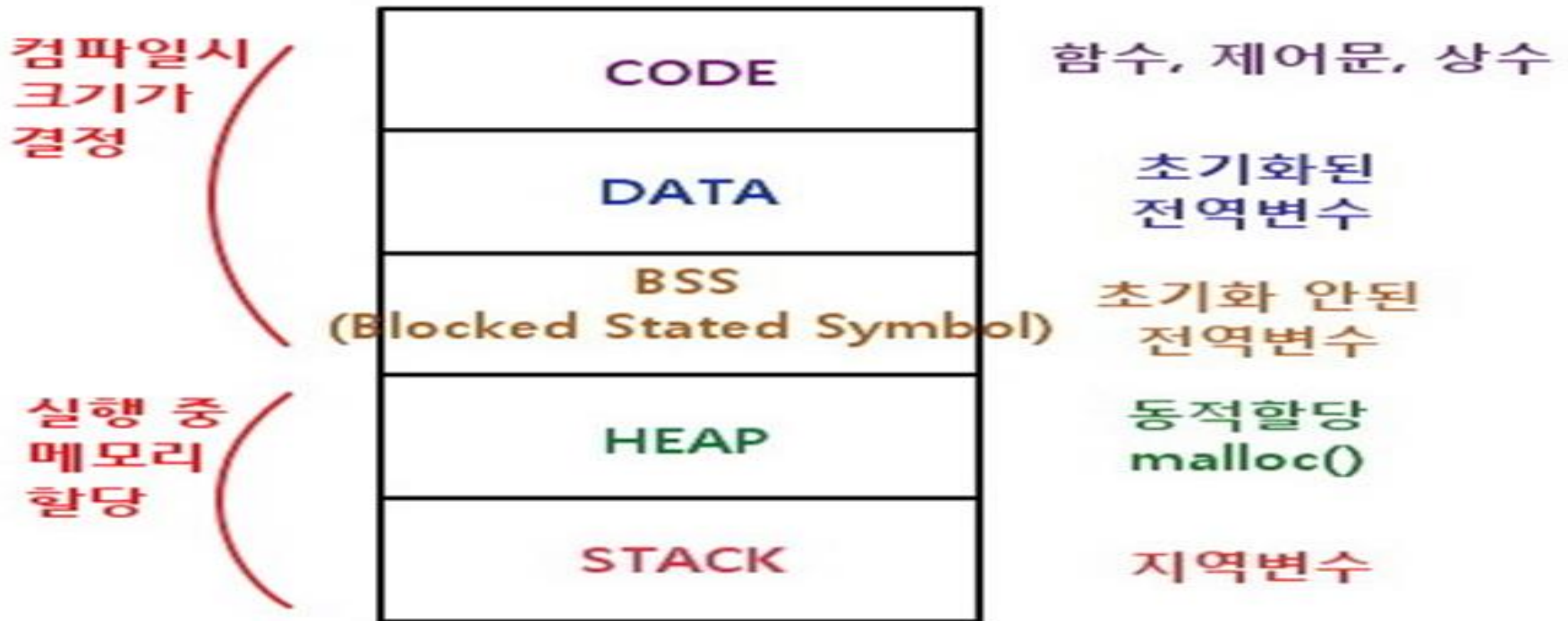
프로세스의 메모리 영역

설정

32비트 시스템에서 프로세스(실행 중인 프로그램) 생성시 4GB의 메모리를 할당 받을 수 있는데, 이는 램만으로는 충당하기엔 턱없이 부족하다.

그래서 운영체제는 램과 하드디스크를 하나로 묶어 가상 메모리로 관리한다.

대부분의 시스템에서는 주로 페이징(Paging)이라는 기법으로 가상 메모리를 관리한다. 가상 메모리는 프로세스당 아래와 같은 구조를 가지는데, 모두 5대영역으로 나뉜다. 아래 그림과 같이 데이터를 분류하여 각 영역에 저장하는 것이다.



메모리 영역

운영체제에서 메모리 영역은 각종 데이터의 저장공간이다. 그런데 데이터 별로 용량이 다르고 타입이 다르고 용도가 다르다

1. 코드영역 : 원시소스와 명령어 등이 저장되는 영역이다. 우리가 프로그램을 짜면 소스가 모두 코드영역에 `text`파일로 저장된다
2. 데이터 영역은 전역 변수와 `static` 변수가 할당되는 영역이다. 이 영역에 할당되는 변수들은 일반적으로 프로그램의 시작과 동시에 할당되고, 프로그램이 종료되어야만 메모리에서 소멸된다. 즉, 데이터 영역에 할당된 변수는 프로그램이 종료될 때까지 계속 존재한다는 특징을 지닌다
3. **Heap**영역 : 프로그래머가 메모리를 동적 할당할 때 저장이 되는 영역이다. 프로그래밍을 할 때 일반 변수를 선언하는 것은 정적인 것으로 사용자가 메모리 크기를 유동적으로 조절할 수가 없지만 동적 할당을 하면 사용자가 필요한 만큼 메모리를 잡을 수 있다
4. 스택 영역 : 지역변수, 전달인자, 지역함수 등 임시적인 성격의 데이터가 저장되는 곳이다. 지역변수가 왜 임시적이냐 해당 지역에서만 가능하니까. 스택 영역은 약 1MB정도이기 때문에 용량을 많이 차지하는 변수를 선언할 때는 동적 할당으로 **Heap**영역에 메모리를 잡으면 된다

메모리 영역은 크게 두 가지로 나눌 수 있는데 컴파일시 크기가 고정되는 **code, data, bss** 영역과 실행시할 때 메모리가 할당되었다 반납되는 **heap, stack**영역으로 나눌 수 있다.

① 메모리 할당이 고정되는 영역

* 코드 영역 - 코드영역은 실행 파일을 구성하는 명령어들이 올라가는 메모리 영역으로 함수, 제어문, 상수 등이 여기에 지정된다.

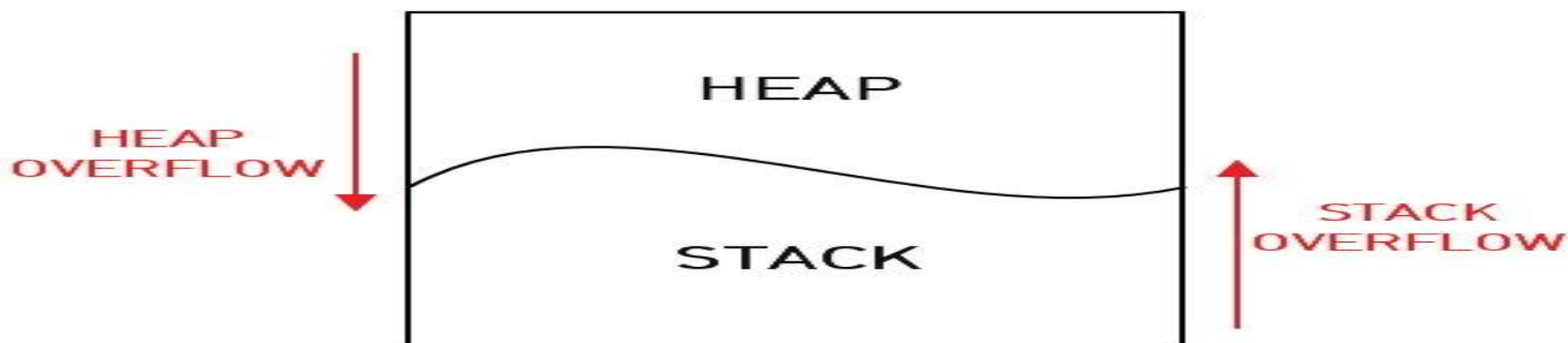
* 데이터 영역 & BSS- 데이터 영역은 BSS와 함께 묶어서 데이터 영역으로 칭하기도 하는데 이는 전역변수와 Static변수가 지정되는 영역이다. 초기화 되지 않은 전역변수들은 BSS에 지정된다.

② 실행 중에 메모리를 할당하는 영역(할당과 반납이 이루어짐)

* **HEAP** 영역 - **HEAP** 영역은 **malloc(), calloc()** 등으로 프로그래머가 자율적으로 메모리 크기를 할당할 수 있는 영역이다. 위의 함수들은 **free()**함수로 할당된 영역을 반납해 줘야하므로 동적 할당 영역에 속한다.

• **STACK** 영역 - **STACK** 영역은 지역변수가 할당되는 영역으로 함수가 호출되면 할당되었다 함수의 종료할 때 반납되는 영역이다.

위의 **HEAP**과 **STACK**영역은 사실 같은 공간을 공유한다. **HEAP**이 메모리 위쪽 주소부터 할당되면 **STACK**은 아래쪽 부터 할당되는 식이다. 그래서 각 영역이 상대 공간을 침범하는 일이 발생할 수 있는데 이를 각각 **HEAP OVERFLOW, STACK OVERFLOW**라고 칭한다.



데이터의 종류와 메모리 영역

변수의 종류

전역변수 : `main()` 함수 밖에서 선언

지역변수 : 메소드 안에서 선언

일반변수 : 일반적으로 사용하는 변수

정적변수 : 선언할 때 한번만 초기화 되고 이후는 대입만 함

	생성시기	소멸시기
전역변수	프로그램이 시작할 때	프로그램이 끝날 때
지역변수	지역에서 선언할 때	해당지역이 끝날 때
일반변수	지역에서 선언할 때	해당지역이 끝날 때
정적변수	프로그램이 시작할 때	프로그램이 끝날 때