

Java



Arrays class

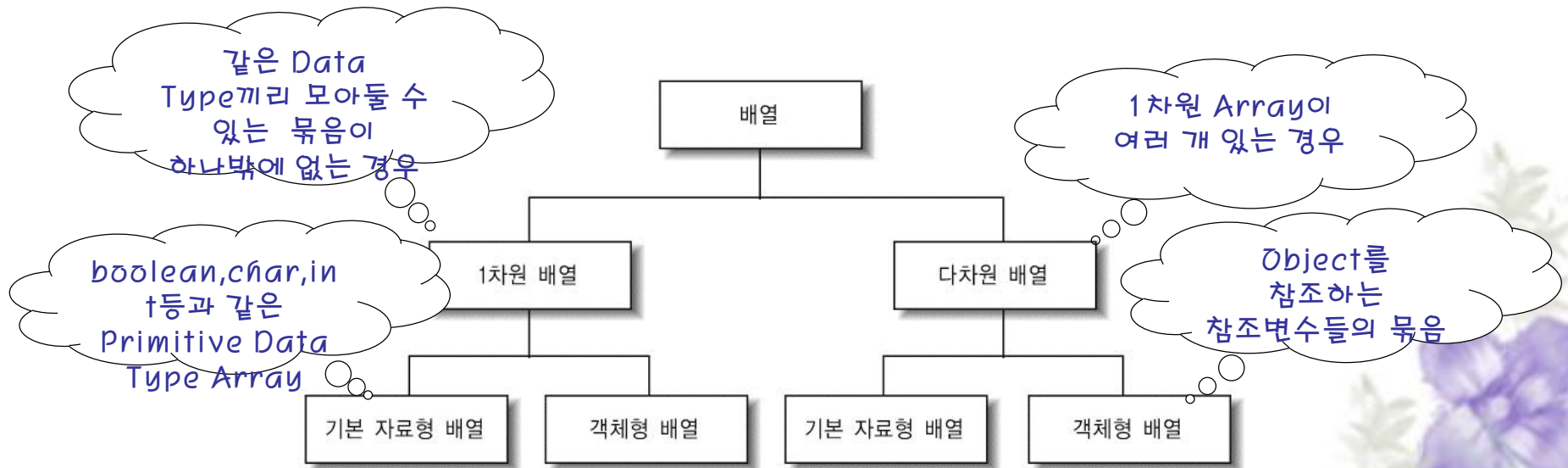
❖ Arrays

- Arrays class는 class내에서 제공하는 class method(static method)등을 이용하여 Array의 비교, Array의 정렬 또는 Array의 내용을 특정 값으로 채우고자 하는 등의 일을 수행하는 도구성 class이다.
- Arrays class는 Constructor가 없으며 API문서를 참조해 보면 다양한 기능들(Method들)이 Overloading 되어 있음을 알 수 있다.

Array

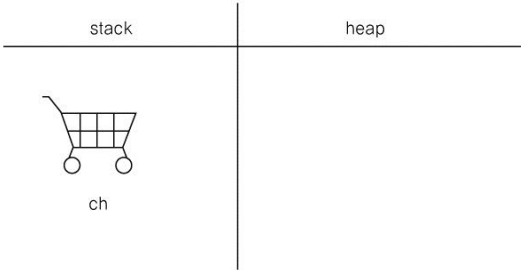
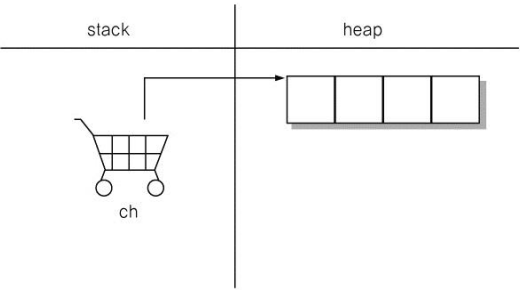
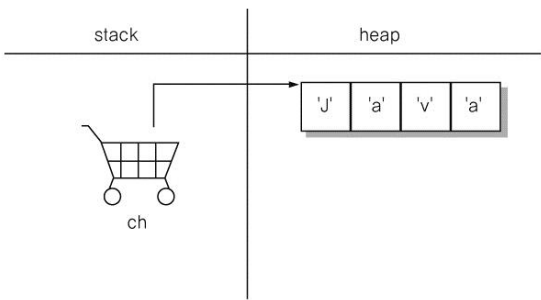
● Array

- ▶ Array는 같은 Data Type끼리 모아두는 하나의 묶음이다.
- ▶ 자바에서 하나의 배열은 하나의 객체로 인식된다.
- ▶ Array의 종류



Array

● Array의 단계적 작업

1. Array Declaration <code>char[] ch; 또는 char ch[];</code>	2. Array Creation <code>ch = new char[4];</code>	3. Array Initialization <code>ch[0]= 'j' ; ch[1]= 'a' ; ch[2]= 'v' ; ch[3]= 'a' ;</code>
		

Array

■ 1st Array Ex

```
01 class ArrayEx1{
02     public static void main(String[] args){
03         char[] cñ; //배열 선언
04         cñ = new char[4]; //배열 생성
05
06         //배열 초기화
07         cñ[0] = 'j';
08         cñ[1] = 'a';
09         cñ[2] = 'v';
10         cñ[3] = 'a';
11
12         //배열 내용 출력
13         for(int i = 0 ; i < cñ.length ; i++)
14             System.out.println("cñ["+i+"]:"+cñ[i]);
15     }
16 }
```

Array

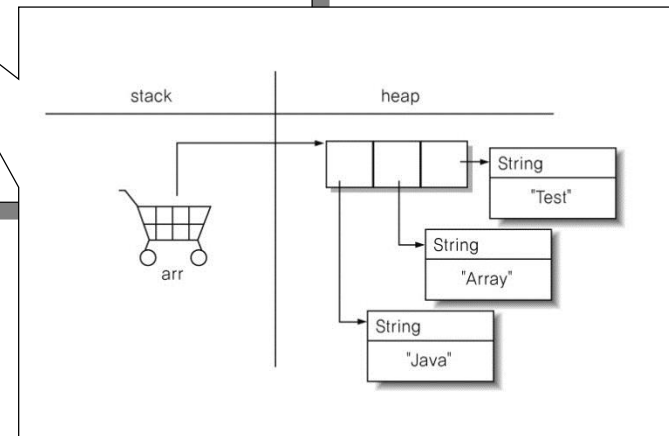
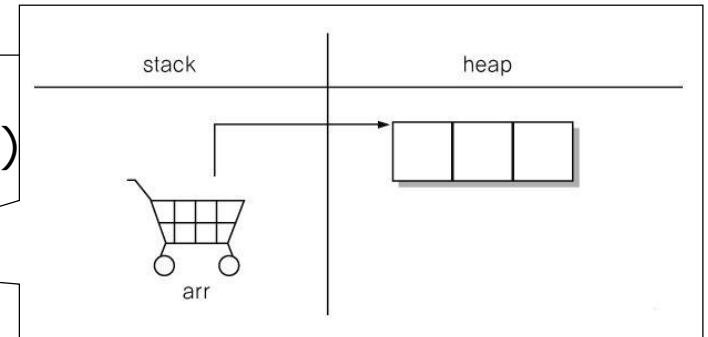
❖ Object type Array

- Object type Array는 Object를 가리킬 수 있는 reference value(Address)들의 묶음이다,
실제 값이 저장된 Primitive Type과는 다르게 Object의 reference들의 집합인 것이다.
- Object type Array는 집집마다 우편함을 한곳에 모아둔 것과 같다. 각 우편함들은 나름대로 특정한 가정이라는 Object의 Address를 대신하는 것을 의미하며 이들의 묶음(집합)이 곧 reference Array 또는 Object type Array이라 한다

Array

■ Object type Array

```
01 class ObjArrayEx1 {  
02     public static void main(String[] args)  
03     {  
04         String[] arr;  
05         arr = new String[3];  
06         arr[0] = "Java ";  
07         arr[1] = "Array ";  
08         arr[2] = "Test";  
09     }  
10 }
```



자바의 기초 문법

04. 배열의 선언, 생성, 이용

배열의 필요성

- 개별적인 변수와 배열

a) 10개의 다른 이름을 갖는 변수들

이름:	a	b	c	d	e	f	g	h	i	j
	7	9	100	32	5	8	6	72	27	81

b) 10개의 데이터를 저장할 수 있는 배열

이름:	num									
	7	9	100	32	5	8	6	72	27	81
인덱스:	0	1	2	3	4	5	6	7	8	9

04. 배열의 선언, 생성, 이용

배열의 선언 (1차원 배열)

- 배열 변수 선언문의 형식 (1)

타입 식별자[];

↑ ↑

배열 항목의 타입 배열 변수의 이름

[예]

```
int        arr[];  
float     num[];  
String    strArr[];
```

- 배열 변수 선언문의 형식 (2)

타입[] 식별자;

↑ ↑

배열 항목의 타입 배열 변수의 이름

[예]

```
int[]        arr;  
float[]     num;  
String[]    strArr;
```

04. 배열의 선언, 생성, 이용

배열의 생성 (1차원 배열)

- 배열은 선언뿐만 아니라 생성을 해야만 사용할 수 있음
- 배열 생성식의 형식

new 타입 [크기];

↑ ↑

배열 항목의 타입 배열 항목의 수

[예]

```
arr = new int[10];  
num = new float[5];  
strArr = new String[3];
```

04. 배열의 선언, 생성, 이용

배열의 이용 (1차원 배열)

- 배열 이름과 인덱스를 이용하면 배열 항목을 단일 변수처럼 사용 가능
- 배열 항목을 가리키는 식

배열이름[인덱스];

↑ ↑

배열 변수의 이름 배열 항목의 위치

[예]

```
arr[0] = 12;  
num[3] = num[1] + num[2];  
System.out.println(strArr[2]);
```

자바의 기초 문법

04. 배열의 선언, 생성, 이용

배열의 선언, 생성, 이용

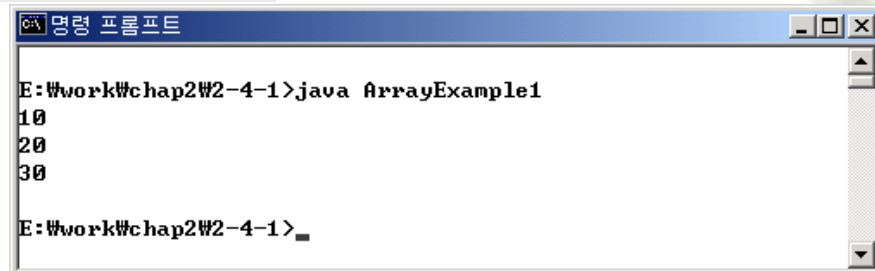
[예제 2-16] 1차원 배열의 사용 예

```
1  class ArrayExample1 {  
2      public static void main(String args[]) {  
3          int arr[];  
4          arr = new int[10];  
5          arr[0] = 10;  
6          arr[1] = 20;  
7          arr[2] = arr[0] + arr[1];  
8          System.out.println(arr[0]);  
9          System.out.println(arr[1]);  
10         System.out.println(arr[2]);  
11     }  
12 }
```

배열 변수를 선언합니다.

배열을 생성합니다.

배열 항목을 사용합니다.

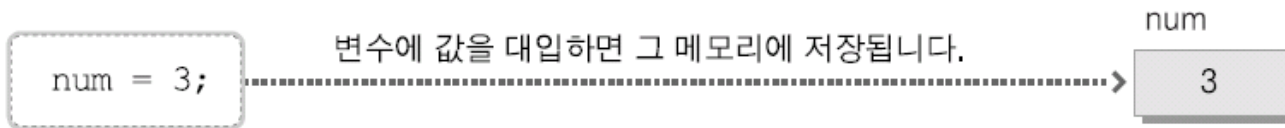
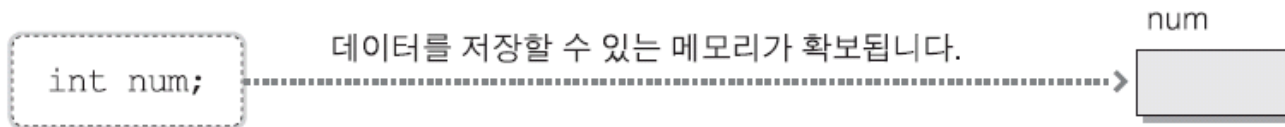


```
명령 프롬프트  
E:\work\chap2\2-4-1>java ArrayExample1  
10  
20  
30  
E:\work\chap2\2-4-1>
```

04. 배열의 선언, 생성, 이용

배열을 생성해야 하는 이유

- 단일 변수의 메모리

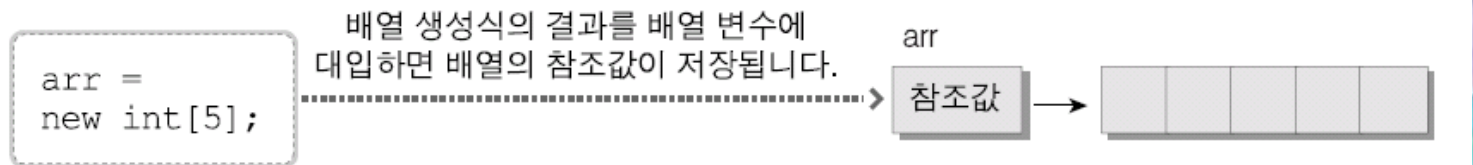
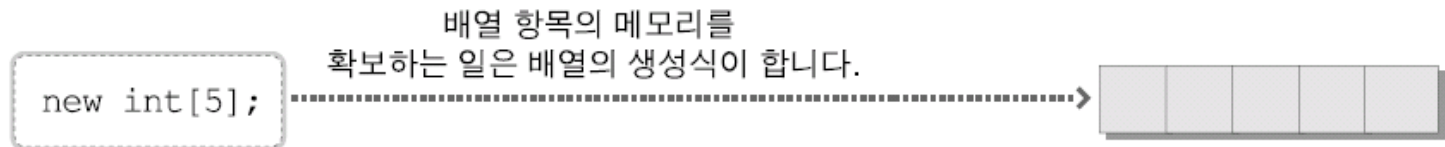
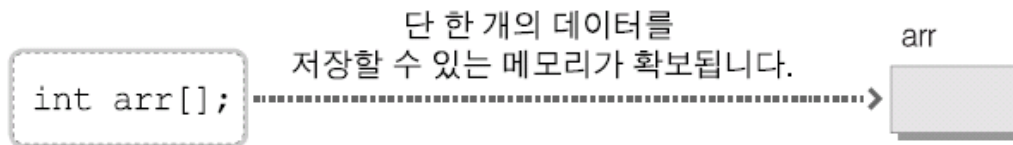


자바의 기초 문법

04. 배열의 선언, 생성, 이용

배열을 생성해야 하는 이유

- 배열의 메모리



04. 배열의 선언, 생성, 이용

배열의 선언 (2차원 배열)

- 배열 변수 선언문의 형식 (1)

타입 식별자 [] [] ;

↑ ↑

배열 항목의 타입 배열 변수의 이름

[예]

```
int        arr[][];  
float     num[][];  
String    strArr[][];
```

- 배열 변수 선언문의 형식 (2)

타입 [] [] 식별자 ;

↑ ↑

배열 항목의 타입 배열 변수의 이름

[예]

```
int[][]     arr;  
float[][]   num;  
String[][]  strArr;
```

04. 배열의 선언, 생성, 이용

배열의 생성 (2차원 배열)

- 배열 생성식의 형식

`new 타입 [크기1] [크기2];`

배열 항목의 타입 행의 수 열의 수

[예]

```
arr = new int[10][10];  
num = new float[5][2];  
strArr = new String[3][15];
```

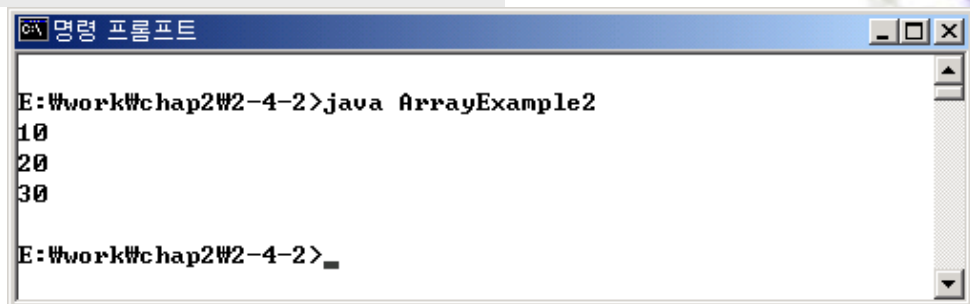

자바의 기초 문법

04. 배열의 선언, 생성, 이용

배열의 선언, 생성, 이용

[예제 2-17] 2차원 배열의 사용 예

```
1  class ArrayExample2 {  
2      public static void main(String args[]) {  
3          int table[][] = new int[3][4];  
4          table[0][0] = 10;  
5          table[1][1] = 20;  
6          table[2][3] = table[0][0] + table[1][1];  
7          System.out.println(table[0][0]);  
8          System.out.println(table[1][1]);  
9          System.out.println(table[2][3]);  
10     }  
11 }
```



```
명령 프롬프트  
E:\work\chap2\2-4-2>java ArrayExample2  
10  
20  
30  
E:\work\chap2\2-4-2>
```

04. 배열의 선언, 생성, 이용 효율적인 코딩 방법

- 배열 선언과 초기화를 함께 하는 명령문

```
int arr[];  
arr = new int[10];  
arr[0] = 10;  
arr[1] = 20;  
arr[2] = 30;  
arr[3] = 40;  
arr[4] = 50;  
arr[5] = 60;  
arr[6] = 70;  
arr[7] = 80;  
arr[8] = 90;  
arr[9] = 100;
```



```
int arr[] = new int[10];  
arr[0] = 10;  
arr[1] = 20;  
arr[2] = 30;  
arr[3] = 40;  
arr[4] = 50;  
arr[5] = 60;  
arr[6] = 70;  
arr[7] = 80;  
arr[8] = 90;  
arr[9] = 100;
```



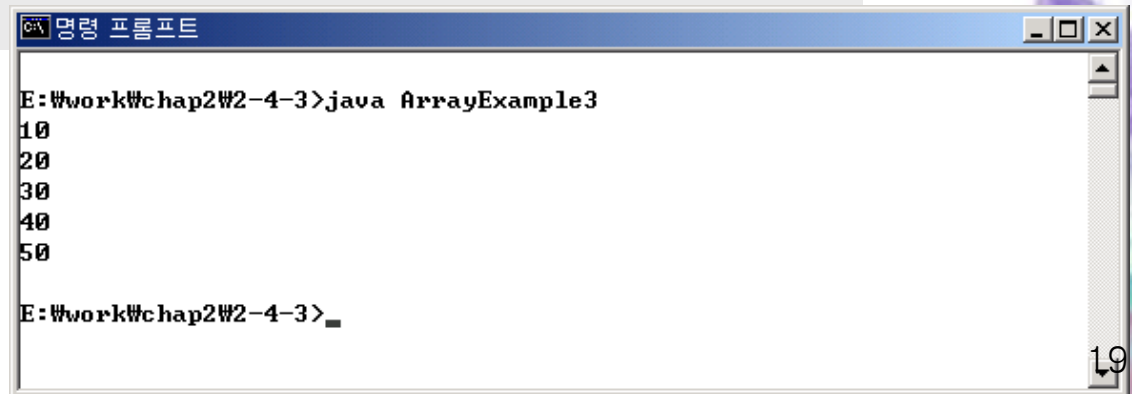
```
int arr[] = { 10, 20, 30, 40, 50,  
             60, 70, 80, 90, 100 };
```

자바의 기초 문법

04. 배열의 선언, 생성, 이용 효율적인 코딩 방법

[예제 2-18] 배열 항목을 초기화하는 배열 변수 선언문 (1차원 배열)

```
1  class ArrayExample3 {  
2      public static void main(String args[]) {  
3          int arr[] = { 10, 20, 30, 40, 50 };  
4          System.out.println(arr[0]);  
5          System.out.println(arr[1]);  
6          System.out.println(arr[2]);  
7          System.out.println(arr[3]);  
8          System.out.println(arr[4]);  
9      }  
10 }
```



```
E:\work\chap2\2-4-3>java ArrayExample3  
10  
20  
30  
40  
50  
  
E:\work\chap2\2-4-3>
```

자바의 기초 문법

04. 배열의 선언, 생성, 이용 효율적인 코딩 방법

[예제 2-19] 배열 항목을 초기화하는 배열 변수 선언문 (2차원 배열)

```
1  class ArrayExample4 {  
2      public static void main(String args[]) {  
3          int table[][] = { { 1, 2, 3, 4 }, { 5, 6, 7, 8 }, { 9, 10, 11, 12 } };  
4          System.out.println(table[0][0]);  
5          System.out.println(table[1][1]);  
6          System.out.println(table[2][3]);  
7      }  
8  }
```



```
명령 프롬프트  
E:\work\chap2\2-4-3>java ArrayExample4  
1  
6  
12  
E:\work\chap2\2-4-3>
```

04. 배열의 선언, 생성, 이용

배열의 항목 수

- 배열의 항목수는 <배열이름>.length라는 식으로 알 수 있음
- [예제 2-20] 배열의 항목 수를 출력하는 프로그램

```
1  class ArrayExample5 {  
2      public static void main(String args[]) {  
3          int arr[] = { 21, 2, 4, 3, 4, 65, 7, 178, 3, 5, 45, 78, 2, 0, 32, 75, 92 };  
4          System.out.println(arr.length);  
5      }  
6  }
```



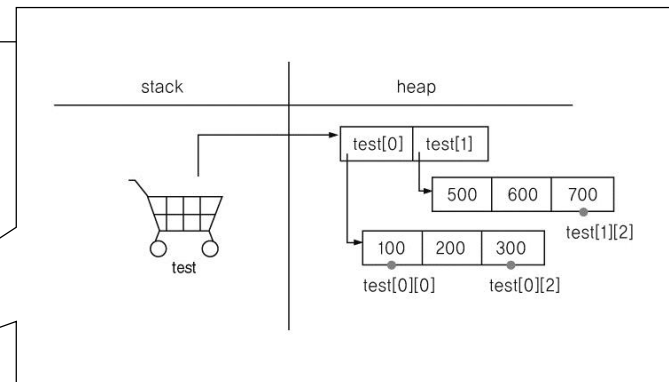
```
명령 프롬프트  
E:\work\chap2\2-4-4>java ArrayExample5  
17  
E:\work\chap2\2-4-4>
```

Array

다차원 Array

➡ Array가 여러 개 모인 것을 다차원 Array이라 한다.

```
01 class ObjArrayEx4 {  
02     public static void main(String[] args){  
03         int[][] test; // 다차원 배열 선언  
04         test = new int[2][3];  
05         test[0][0] = 100;  
06         test[0][1] = 200;  
07         test[0][2] = 300;  
08         //----- 1행 초기화 끝  
09         test[1][0] = 500;  
10         test[1][1] = 600;  
11         test[1][2] = 700;  
12         //----- 2행 초기화 끝  
13     }  
14 }
```



Arrays class

※ Arrays 클래스에서 제공하는 메서드

반환형	메서드명	설명
static <T> List <T>	asList(T... a)	지정된 배열을 기본으로 하는 고정 길이의 배열을 반환
static int	binarySearch (Object [] a, Object key)	이진 검색 방법을 사용하여 전달받은 인자 중 배열에서 두 번째 인자인 key 객체를 검색한다. 그리고 검색된 객체의 index를 반환
static boolean	equals(Object [] a, Object [] a2)	인자로 전달된 2개의 객체형 배열이 서로 같을 경우에 true를 반환
static void	fill(Object [] a, Object val)	전달된 인자 중 객체형 배열의 각 요소를 전달된 인자 중 val이라는 객체로 모두 대입
	fill(Object [] a, int fromIndex, int toIndex, Object val)	전달받은 인자 중 객체형 배열의 fromIndex에서 toIndex까지 지정된 범위에 있는 각 요소를 4번째 인자인 val이라는 객체로 모두 대입
static int	hashCode (Object [] a)	전달된 배열 인자의 내용에 근거하는 해시코드를 반환
static void	sort(Object [] a)	요소의 자연적(기본적) 순서에 따라, 전달된 인자의 배열을 오름차순으로 정렬
	sort(T[] a, Comparator <? super T> c)	전달된 Comparator가 가리키는 순서에 따라 전달된 T a 객체 배열을 정렬
static String	toString (Object [] a)	전달된 배열 내용들의 캐릭터 행 표현을 반환

Arrays class

○ Array 채우기

- 앞서 살펴본 class method(static method)들을 가지고 Arrays의 능력을 살펴볼 것이며 우선 Array fill를 알아본다. Data Type만 일치한다면 Array의 일부분 또는 전체를 원하는 값 또는 Object로 채울 수가 있다.

```
01 import java.util.Arrays;
02 import static java.lang.System.out;
03 public class ArraysEx1 {
04
05     public static void main(String[] args) {
06         String[] ar = {"fill()", "in", "the", "Arrays"};
07
08         Arrays.fill(ar, "SunAe"); //Array의 요소들을 "SunAe"로
채운다.
09         for(String n : ar)
10             out.print(n+",");
11
12         out.println("\n-----");
13         Arrays.fill(ar, 1, 3, "♥");
14         for(String n : ar)
15             out.print(n+",");
16     }
17 }
```


Arrays class

❖ Array간의 비교

- Array과 Array간의 내용을 비교한다는 것은 두 Array이 가지는 요소들이 서로 일치하는가? 하는 기능을 두고 얘기 하는 것이다.
- Arrays class의 equals() method는 두 Array의 길이와 각각의 요소들이 일치하는지를 검증하여 모두 일치할 경우에만 true를 반환해 주는 method다.

```
01 import java.util.Arrays;
02 import static java.lang.System.out;
03 public class ArraysEx2 {
04
05     public static void main(String[] args) {
06         String[] ar1 = {"fill()", "in", "the", "Arrays"};
07         String[] ar2 = {"fill()", "in", "", "Arrays"};
08
09         if(!Arrays.equals(ar1, ar2))
10             out.println("두 Array이 다릅니다.");
11         out.println("-----");
12     }
```

Arrays class

```
13         Arrays.fill(ar2, 2, 3, "the");
14         if(!Arrays.equals(ar1, ar2))
15             out.println("두 Array이 다릅니다.");
16         else
17             out.println("두 Array이 같습니다.");
18     }
19 }
```

- Arraysclass 의 equals()method 는 두 Array 의 내용이 일치하는지를 알아내는 method이다. 이는 두 Array의 요소들의 수가 같은지를 먼저 파악하고 두 Array이 같은 순서로 같은 요소들을 포함하고 있는 경우에 true를 반환하는 method이다. 또는 두 Array이 서로 null값을 참조하고 있다면 equals()method는 true를 반환한다.

Arrays class

○ Array간의 정렬

- Array의 정렬이라는 것은 Array내의 요소들이 서로 크고 작은 것임을 비교하여 작은 것에서부터 큰 것까지 나열되는 것을 말한다.

```
01 import java.util.Arrays;
02 import static java.lang.System.out;
03 public class ArraysEx3 {
04
05     public static void main(String[] args) {
06         int[] ar1 = {20,4,12,1,5};
07
08         Arrays.sort(ar1);
09         for(int n : ar1)
10             out.print(n+",");
11     }
12 }
```

Arrays class

○ Array간의 정렬

- Array 또는 Collection 내에 Object(Object)들이 저장되어 정렬 기준이 모호하여 비교가 되지 않을 경우에는 Programmer가 정렬에 대한 기준을 다음 두 가지 방법 중 하나를 구현하면서 제시해야 한다.
- [방법 1]

※ java.lang.Comparable 인터페이스

인터페이스	설명
public interface Comparable<T>	이 인터페이스를 구현하는 각 클래스의 객체에 정렬 기준을 의미한다. 이는 클래스의 자연적 정렬 기준이라고 불리며 compareTo() 메서드는 자연적 비교 메서드라고도 불린다. 이 인터페이스를 구현하는 객체는 Arrays.sort()나 Collections.sort() 등에 의해 자동적으로 정렬을 이룰 수 있다.

※ Comparable 인터페이스의 메서드

반환형	메서드명	설명
int	compareTo(T o)	인터페이스를 구현한 현재 객체와 인자로 전달된 객체가 정렬을 위한 비교를 한다.

Arrays class

■ [방법 2]

※ java.util.Comparator 인터페이스

인터페이스	설명
public interface Comparator<T>	Comparator를 정렬 메서드인 Arrays.sort()나 Collections.sort() 등에 건네주면 정렬순서를 정확하게 제어할 수 있다. 또 Comparator를 사용하면 TreeSet 또는 TreeMap이라고 하는 자료구조의 순서를 제어할 수도 있다.

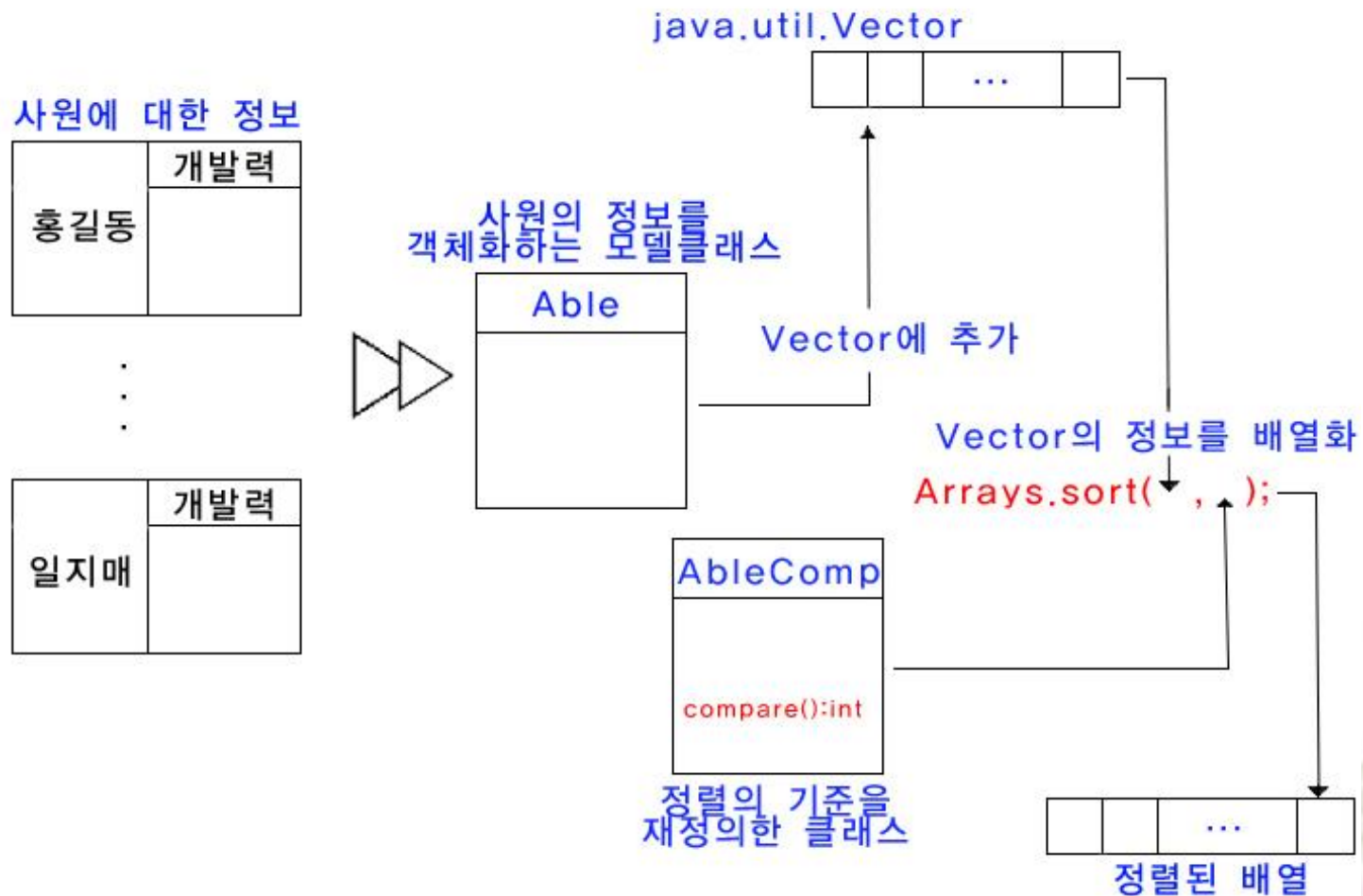
※ Comparator 인터페이스의 메서드

반환형	메서드명	설명
int	compare(T o1, T o2)	정렬을 위해 인자로 전달된 두 개의 객체를 비교
boolean	equals(Object obj)	인자로 전달된 obj 객체가 현재의 Comparator와 동일한지 비교

- Comparator는 외부의 Arrays.sort()와 같은 method에서 Object 비교를 하여 정렬의 순서가 정해지므로 Object의 정렬 기준을 compare() method를 재정의하여 정해 주어야 한다,

argument로 전달된 o1 Object가 더 크다면 양수(1)를 반환하고 반대로 o1 Object가 o2 Object보다 더 작다면 음수(-1)를 반환 해야 한다, 그리고 두 Object가 같다면 0을 반환하는 규칙으로 재 정의를 해야 하는 것이다,

Arrays class



Arrays class

- 각 사원들의 정보를 Object화 시키는 모델 class

```
01 class Able{
02     String empno;
03     int net,ejb,xml,total;
04     public Able(int total){ //Constructor
05         this.total = total;
06     }
07     public Able(String no,int n,int e,int x){
//Constructor
08         empno = no;
09         net = n;
10         ejb = e;
11         xml = x;
12         total = n+e+x;
13     }
14     public int getTotal(){
15         return total;
16     }
17 }
```

Arrays class

- 정렬의 기준을 재정의한 class

```
01 import java.util.Comparator;
02     class           AbleComp           implements
Comparator<Able>{
03
04     public int compare(Able obj1, Able obj2){
05         int var = 0;
06
07         if(obj1.getTotal()           >
obj2.getTotal())
08             var = 1;
09         else if(obj1.getTotal()           <
obj2.getTotal())
10             var = -1;
11         return var;
12     }
13 }
```


Arrays class

- 앞의 compare() method 같은 경우는 외부의 Arrays.sort() method 나 Collections.sort() 등에 의해 자동 호출되어 각 Object들간의 정렬 순서가 정해진다. 정의된 Able class와 AbleComp class를 이용한 정렬해 보자

```
01 import java.util.Arrays;
02 import java.util.Vector;
03 import static java.lang.System.out;
04 class AbleEx1{
05     public static void main(String[] args) {
06         Vector<Able> v = new Vector<Able>(2,5);
07
08         //AbleObject 생성 및 Vector에 추가
09         Able a1 = new Able("B123",90,75,70);
10         Able a2 = new Able("T723",60,90,80);
11         Able a3 = new Able("A427",85,80,80);
12         Able a4 = new Able("G533",90,90,60);
13         v.addElement(a1);
14         v.addElement(a2);
15         v.addElement(a3);
16         v.addElement(a4);
```

Arrays class

```
18         out.println("----- Sort 전 -----");
19         for(Able n : v)
20             out.println(n.empty()+" "+n.getTotal());
21
22         Able[] a = new Able[v.size()];
23         v.copyInto(a);
24         AbleComp comp = new AbleComp();
25         Arrays.sort(a, comp);
26         out.println("----- Sort 후 -----");
27         // v = new Vector<Able>(Arrays.asList(a));
28         for(Able n : a)
29             out.println(n.empty()+" "+n.getTotal());
30     }
31 }
```

Properties class

❖ Properties

Properties class는 말 그대로 속성들을 모아서 하나의 Object로 만들기 위해 제공되는 class이다

❖ Properties의 필요성

일반적으로 사용하는 각자의 Computer도 마찬가지이며 그 안에서 구동 되는 모든 Program들도 각각의 속성들을 가지고 있다. Computer가 켜질 때 또는 Program이 시작되기 전에 여러 개의 속성들 중 원하는 속성들을 미리 인식되게 하여 전반적인 실행 환경을 조율하고 보다 신속한 처리속도를 가져오는데 목적을 두고 있다.

※ Properties 클래스 생성자

생성자명	설명
Properties()	기본값도 없는 새로운 Properties 객체를 생성한다.

Properties class

- Properties Object에서 속성들을 저장 관리하는데 필요한 주요 method들이다,

※ Properties 클래스의 주요 메서드

반환형	메서드명	설명
String	getProperty (String key)	인자로 전달된 key를 가지는 속성값을 찾아 반환
void	list (PrintWriter out)	인자로 전달된 출력 스트림을 통해 속성 목록들을 출력
	load(InputStream inStream)	인자로 전달된 입력 스트림으로부터 키와 요소가 한 쌍으로 구성된 속성 목록들을 읽어들이어 현 Properties 객체에 저장
Enumeration <?>	propertyNames()	속성 목록에 있는 모든 속성의 key값들을 열거형 객체로 반환
Object	setProperty (String key, String value)	현 Properties 객체의 속성 목록에 인자로 전달된 key와 value를 한 쌍으로 구성되어 저장. 내부적으로는 Hashtable의 put() 메서드가 호출
void	store (OutputStream out, String comments)	모든 속성들을 load() 메서드를 사용해 Properties 테이블에 로드하고 적절한 포맷과 인자로 전달된 출력 스트림을 통해 출력

Properties class

```
01 import java.util.Properties;
02 import java.util.Enumeration;
03 import static java.lang.System.out;
04 public class PropertiesEx1 {
05     public static void main(String[] args) {
06         Properties prop = new Properties();
07
08         prop.put("UserName","Michael"); //속성 저장하기
09         prop.setProperty("lovely","SunAe");
10
11         prop.setProperty("dbDRV","oracle.jdbc.driver.OracleDriver");
12
13         String user = prop.getProperty("UserName"); //속성
14         String love = prop.getProperty("lovely");
15         String db_drv = prop.getProperty("dbDRV");
16         out.println(user);
17         out.println(love);
18         out.println(db_drv);
19         out.println("----- keys -----");
```

Properties class

```
20      //키 값들만 얻어내기
21      Enumeration keys = prop.propertyNames();
22      while(keys.hasMoreElements())
23          out.println(keys.nextElement());
24
25      prop.list(System.out); //속성 목록 출력하기
26  }
27 }
```

Properties class

- 이 외에도 Properties를 사용하는 작은 예로는 바로 파일의 경로를 정의할 때 Windows 환경에서는 경로의 구분자를 '\\' 를 사용하지만 Unix 나 Linux환경에서는 '/' 를 사용한다.

이런 속성 하나로 인해 Windows환경에서는 잘 구동 되던 Program이 Linux로 환경이 변경된 후 갑자기 오류가 발생하는 원인이 된다. 이럴 때 사용할 수 있는 것이 바로 현재 컴퓨터의 속성에서 '경로 구분자' 를 어떻게 사용하는지를 알아내는 `System.getProperty("file.separator")` 라는 코드로 해결할 수 있다.

이제

많은

일을

이루고

싶습니다~