

배열

13-1.

1차원 배열의 이해와 활용

1차원 배열의 이해와 선언 방법

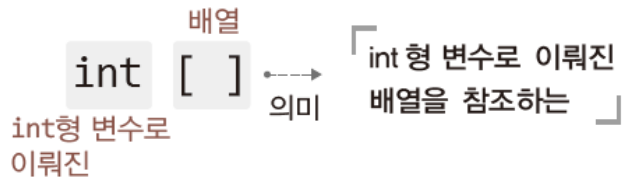
1차원 배열이란?

타입이 같은 둘 이상의 데이터를 저장할 수 있는 1차원 구조의 메모리 공간

1차원 배열의 선언 방법

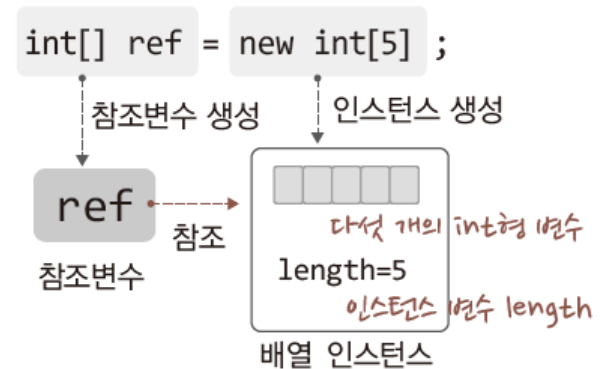
```
int[] ref = new int[5];    // 길이가 5인 int형 1차원 배열의 생성문
```

배열 선언문에 대한 세세한 이해와 결과



```
public static void main(String[] args) {  
    int[] ref;  
    ref = new int[5];  
    ....  
}
```

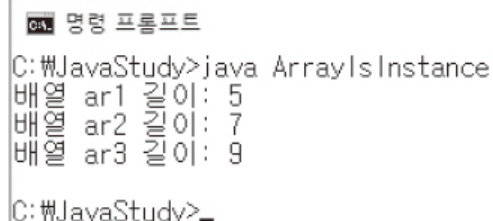
배열의 참조변수와 인스턴스의 선언도 분리 가능!



멤버 변수 `length`는 배열의 길이 정보 저장

1차원 배열의 예

```
public static void main(String[] args) {  
    // 길이가 5인 int형 1차원 배열의 생성  
    int[] ar1 = new int[5];  
  
    // 길이가 7인 double형 1차원 배열의 생성  
    double[] ar2 = new double[7];  
  
    // 배열의 참조변수와 인스턴스 생성 분리  
    float[] ar3;  
    ar3 = new float[9];  
  
    // 배열의 인스턴스 변수 접근  
    System.out.println("배열 ar1 길이: " + ar1.length);  
    System.out.println("배열 ar2 길이: " + ar2.length);  
    System.out.println("배열 ar3 길이: " + ar3.length);  
}
```



```
C:\JavaStudy>java ArrayIsInstance  
배열 ar1 길이: 5  
배열 ar2 길이: 7  
배열 ar3 길이: 9  
C:\JavaStudy>
```

인스턴스 대상 1차원 배열의 예

```
class Box {  
    private String conts;  
  
    Box(String cont) { this.conts = cont; }  
    public String toString() {  
        return conts;  
    }  
}
```

```
class ArrayIsInstance2 {  
    public static void main(String[] args) {  
        Box[] ar = new Box[5];    // 길이가 5인 Box형 1차원 배열의 생성  
        System.out.println("length : " + ar.length);    // length: 5  
    }  
}
```

cmd 명령 프롬프트

C:\JavaStudy>java ArrayIsInstance2
length : 5

C:\JavaStudy>_

배열의 활용: 값의 저장과 참조

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

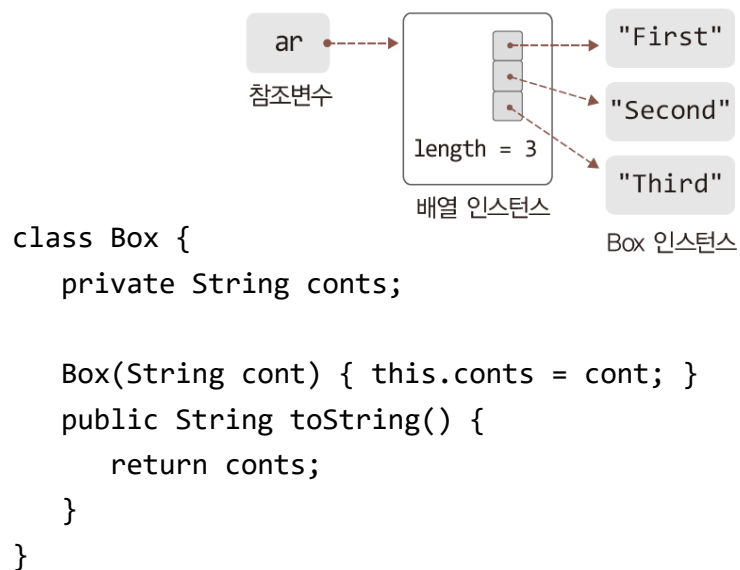
```
ar[0] = 7;           // 값의 저장: 첫 번째 요소
```

```
ar[1] = 8;           // 값의 저장: 두 번째 요소
```

```
ar[2] = 9;           // 값의 저장: 세 번째 요소
```

```
int num = ar[0] + ar[1] + ar[2];    // 값의 참조
```

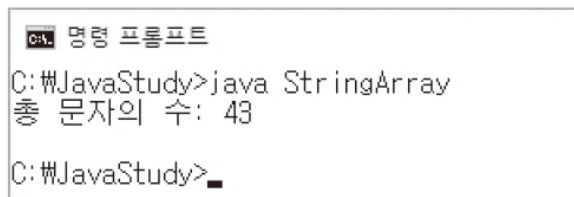
값의 저장과 참조의 예



```
public static void main(String[] args) {  
    Box[] ar = new Box[3];  
  
    // 배열에 인스턴스 저장  
    ar[0] = new Box("First");  
    ar[1] = new Box("Second");  
    ar[2] = new Box("Third");  
  
    // 저장된 인스턴스의 참조  
    System.out.println(ar[0]);  
    System.out.println(ar[1]);  
    System.out.println(ar[2]);  
}
```


배열 기반 반복문 활용의 예

```
public static void main(String[] args) {  
    String[] sr = new String[7];  
    sr[0] = new String("Java");  
    sr[1] = new String("System");  
    sr[2] = new String("Compiler");  
    sr[3] = new String("Park");  
    sr[4] = new String("Tree");  
    sr[5] = new String("Dinner");  
    sr[6] = new String("Brunch Cafe");  
  
    int cnum = 0;  
    for(int i = 0; i < sr.length; i++)  
        cnum += sr[i].length();  
  
    System.out.println("총 문자의 수: " + cnum);  
}
```



```
C:\JavaStudy>java StringArray  
총 문자의 수: 43  
C:\JavaStudy>
```

배열 요소는 반복문을 통해 순차적 접근이 가능하며,
이것은 배열이 가진 큰 장점 중 하나이다.

배열의 생성과 동시에 초기화

배열 생성

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

배열 생성 및 초기화1

```
int[] arr = new int[] {1, 2, 3};
```

배열 생성 및 초기화2

```
int[] arr = {1, 2, 3};
```

배열 대상 참조변수 선언의 두 가지 방법

```
int[] ar = new int[3];    // 조금 더 선호하는 방법
```

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

배열의 참조 값과 메소드

```
public static void main(String[] args) {  
    int[] ar = {1, 2, 3, 4, 5, 6, 7};  
    int sum = sumOfAry(ar);    // 배열의 참조 값 전달  
    ....  
}
```

```
static int sumOfAry(int[] ar) {  
    int sum = 0;  
    for(int i = 0; i < ar.length; i++)  
        sum += ar[i];  
    return sum;  
}
```

```
static int[] makeNewIntAry(int len) {  
    int[] ar = new int[len];  
    return ar;  
}
```

배열의 참조 값 반환 가능

배열의 디폴트 초기화

기본 자료형 배열은 모든 요소 **0**으로 초기화

```
int[] ar = new int[10];
```

인스턴스 배열(참조변수 배열)은 모든 요소 **null**로 초기화

```
String[] ar = new String[10];
```

배열의 초기화 메소드

```
public static void fill(int[] a, int val)
```

→ 두 번째 인자로 전달된 값으로 배열 초기화

```
public static void fill(int[] a, int fromIndex, int toIndex, int val)
```

→ 인덱스 fromIndex ~ (toIndex-1)의 범위까지 val의 값으로 배열 초기화

java.util.Arrays 클래스에 정의되어 있는 메소드, 원하는 값으로 배열 전부 또는 일부를 채울 때 사용하는 메소드

배열 복사 메소드

```
public static void
```

```
    arraycopy(Object src, int srcPos, Object dest, int destPos, int length)
```

→ 복사 원본의 위치: 배열 src의 인덱스 srcPos

→ 복사 대상의 위치: 배열 dest의 인덱스 destPos

→ 복사할 요소의 수: length

java.lang.System 클래스에 정의되어 있는 메소드, 한 배열에 저장된 값을 다른 배열에 복사할 때 사용하는 메소드

배열 초기화와 복사의 예

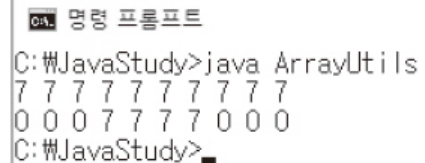
```
import java.util.Arrays;

class ArrayUtils {
    public static void main(String[] args) {
        int[] ar1 = new int[10];
        int[] ar2 = new int[10];

        Arrays.fill(ar1, 7);    // 배열 ar1을 7로 초기화
        System.arraycopy(ar1, 0, ar2, 3, 4);    // 배열 ar1을 ar2로 부분 복사

        for(int i = 0; i < ar1.length; i++)
            System.out.print(ar1[i] + " ");
        System.out.println(); // 단순 줄 바꿈

        for(int i = 0; i < ar2.length; i++)
            System.out.print(ar2[i] + " ");
    }
}
```



```
cmd 명령 프롬프트
C:\JavaStudy>java ArrayUtils
7 7 7 7 7 7 7 7 7
0 0 0 7 7 7 0 0 0
C:\JavaStudy>
```


main 메소드의 매개변수 선언

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

main을 호출해야 한다면 다음과 같이...

```
String[] arr = new String[] {"Coffee", "Milk", "Orange"};  
main(arr);
```

```
C:\JavaStudy>java Simple
```


```
String[] arr = new String[] { };  
main(arr);
```

```
C:\JavaStudy>java Simple Coffee Milk Orange
```

```
String[] arr = new String[] {"Coffee", "Milk", "Orange"};  
main(arr);
```

main의 매개변수로 인자를 전달하는 예

```
class Simple {  
    public static void main(String[] args) {  
        for(int i = 0; i < args.length; i++ )  
            System.out.println(args[i]);  
    }  
}
```



명령 프롬프트

```
C:\JavaStudy>java Simple Coffee Milk Orange  
Coffee  
Milk  
Orange
```

생성된 배열

```
String[] arr = new String[] {"Coffee", "Milk", "Orange"};
```

C:\JavaStudy>

13-2. enhanced for문

enhanced for문(for-each문)의 이해

코드의 특징: 배열 요소의 순차적 접근

```
int[] ar = {1, 2, 3, 4, 5};  
for(int i = 0; i < ar.length; i++) {  
    System.out.println(ar[i]);  
}
```

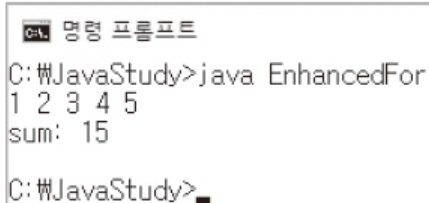
위 유형의 코드는 for-each문으로 다음과 같이 구성 가능

```
int[] ar = {1, 2, 3, 4, 5};  
for(int e : ar) {  
    System.out.println(e);  
}
```

코드의 양이 줄고 배열의 길이와 요소에 신경 쓸 필요 없다.

for-each문의 예

```
public static void main(String[] args) {  
    int[] ar = {1, 2, 3, 4, 5};  
  
    // 배열 요소 전체 출력  
    for(int e: ar) {  
        System.out.print(e + " ");  
    }  
    System.out.println(); // 단순 줄 바꿈을 목적으로  
  
    int sum = 0;  
  
    // 배열 요소의 전체 합 출력  
    for(int e: ar) {  
        sum += e;  
    }  
    System.out.println("sum: " + sum);  
}
```

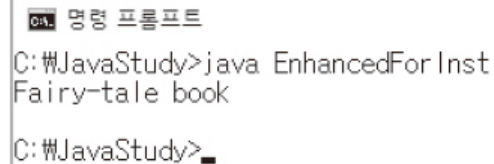


A screenshot of a Windows command prompt window titled "명령 프롬프트". The window shows the execution of a Java program. The command entered is "C:\WJavaStudy>java EnhancedFor". The output displayed is "1 2 3 4 5" followed by "sum: 15". The prompt "C:\WJavaStudy>" is shown again at the bottom.

```
C:\WJavaStudy>java EnhancedFor  
1 2 3 4 5  
sum: 15  
C:\WJavaStudy>
```

인스턴스 배열 대상 for-each문의 예

```
public static void main(String[] args) {  
    Box[] ar = new Box[5];  
    ar[0] = new Box(101, "Coffee");  
    ar[1] = new Box(202, "Computer");  
    ar[2] = new Box(303, "Apple");  
    ar[3] = new Box(404, "Dress");  
    ar[4] = new Box(505, "Fairy-tale book");  
  
    // 배열에서 번호가 505인 Box를 찾아 그 내용물을 출력하는 반복문  
    for(Box e: ar) {  
        if(e.getBoxNum() == 505)  
            System.out.println(e);  
    }  
}
```



```
C:\JavaStudy>java EnhancedForInst  
Fairy-tale book  
C:\JavaStudy>
```

13-3.

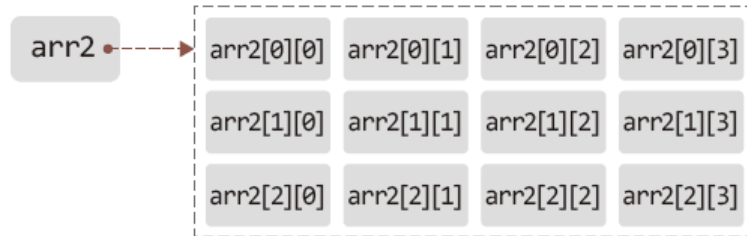
다차원 배열의 이해와 활용

2차원 배열의 생성

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



```
int[][] arr2 = new int[3][4]
```



2차원 배열의 접근

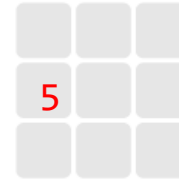
```
int[][] arr = new int[3][3]
```



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



```
arr[1][0] = 5;
```



```
arr[2][2] = 9;
```



```
arr[0][1] = 7;
```

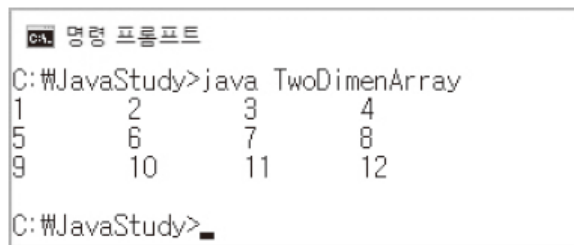


2차원 배열의 예

```
public static void main(String[] args) {  
    int[][] arr = new int[3][4];  
    int num = 1;  
  
    // 배열에 값을 저장  
    for(int i = 0; i < 3; i++) {  
        for(int j = 0; j < 4; j++) {  
            arr[i][j] = num;  
            num++;  
        }  
    }  
  
    // 배열에 저장된 값을 출력  
    for(int i = 0; i < 3; i++) {  
        for(int j = 0; j < 4; j++) {  
            System.out.print(arr[i][j] + "\\t");  
        }  
        System.out.println();  
    }  
}
```

2차원 배열 요소 전체의 순차적 접근은 중첩된 반복문으로...

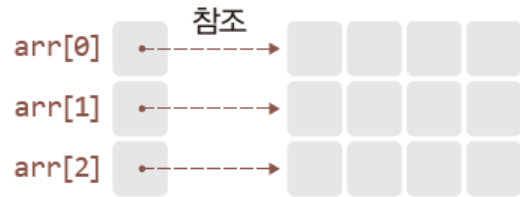
for문의 중첩으로...



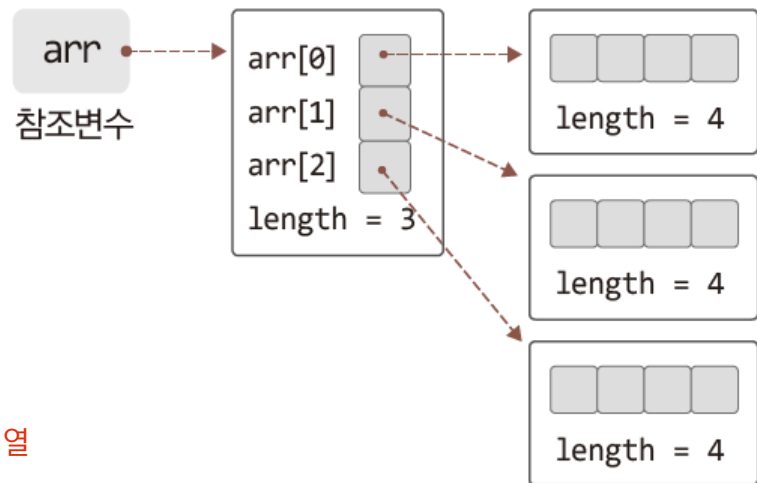
```
C:\JavaStudy>java TwoDimenArray  
1      2      3      4  
5      6      7      8  
9      10     11     12  
  
C:\JavaStudy>
```

2차원 배열의 실제 구조

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



다수의 1차원 배열을 엮어서 구성이 되는 2차원 배열

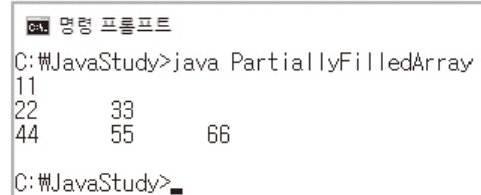


2차원 배열의 초기화

```
int[][] arr = {  
    {11, 22, 33},  
    {44, 55, 66},  
    {77, 88, 99}  
};
```

```
int[][] arr = {  
    {11},  
    {22, 33},  
    {44, 55, 66}  
};
```

```
public static void main(String[] args) {  
    int[][] arr = {  
        {11},  
        {22, 33},  
        {44, 55, 66}  
    };  
  
    // 배열의 구조대로 내용 출력  
    for(int i = 0; i < arr.length; i++) {  
        for(int j = 0; j < arr[i].length; j++) {  
            System.out.print(arr[i][j] + "\t");  
        }  
        System.out.println();  
    }  
}
```



```
명령 프롬프트  
C:\JavaStudy>java PartiallyFilledArray  
11  
22    33  
44    55    66  
C:\JavaStudy>
```