Chapter 05

참조 타입

05-2. 배열

혼자 공부하는 자바 (신용권 저)

시작하기 전에

[핵심 키워드] : 배열, 인덱스, 배열 길이, 배열 선언, 배열 생성, 다차원 배열, 향상된 for문

[핵심 포인트]

많은 양의 데이터를 적은 코드로 처리하는 배열에 대해 알아본다.

❖ 많은 양의 데이터를 다루는 프로그램

- ex) 학생 30명의 성적을 저장하고 평균값을 구하려면?

```
int score1 = 83;
int score2 = 90;
int score3 = 87;
...
int score30= 75;
```

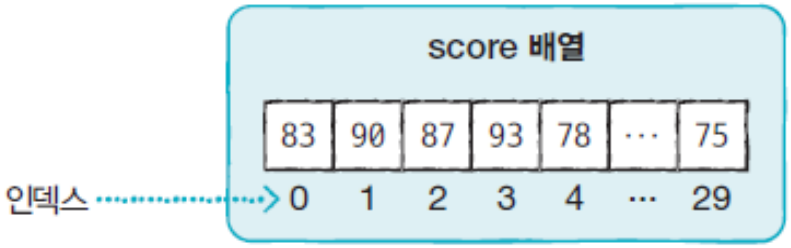
```
int sum = score1;
sum += score2;
sum += score3;
...
sum += score30;
int avg = sum / 30;
```

배열이란?

❖ 배열

- 데이터를 연속된 공간에 나열하고 각 데이터에 인덱스(Index) 부여한 자료구조
- 같은 타입의 데이터만 저장할 수 있음
- 한 번 생성된 배열은 길이를 늘리거나 줄일 수 없음

- score 배열



- for문을 이용한 배열 처리

```
int sum = 0;
for(int i=0; i<30; i++) {
   sum += score[i];
}
int avg = sum / 30;
```

배열 선언

❖ 배열 변수 선언

```
int[] intArray;
double[] doubleArray;
String[] strArray;
```

```
int intArray[];
double doubleArray[];
String strArray[];
```

- 참조할 배열 객체 없는 경우 배열 변수는 null 값으로 초기화

```
타입[] 변수 = null;
```

❖ 배열 생성

- 값 목록으로 배열 생성

```
타입[] 변수 = { 값0, 값1, 값2, 값3, … };
```

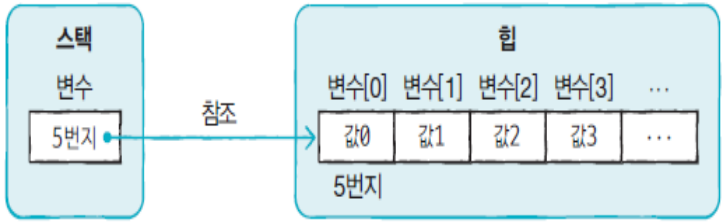
- new 연산자를 이용해서 배열 생성

```
int[] scores = new int[30];
```

배열 생성

❖ 값 목록을 이용해서 배열 생성

```
타입[] 변수 = { 값0, 값1, 값2, 값3, … };
```




```
int[] scores = { 90, 95, 87, 93, ... };
```

scores[1] = 100

- 배열 변수 선언한 뒤에는 다른 실행문에서 값 목록으로 배열 생성 불가능

```
타입[] 변수;
변수 = { 값0, 값1, 값2, 값3, … };  //컴파일 에러
```

- 배열 변수 미리 선언한 후 값 목록이 나중에 결정되는 경우
  - new 연산자 사용하여 값 목록 지정

```
변수 = new 타입[] { 값0, 값1, 값2, 값3, … };
```

```
String[] names = null;
names = new String[] { "신용권", "홍길동", "감자바" };
```

배열 생성

❖ new 연산자로 배열 생성

```
타입[] 변수 = new 타입[길이];
```

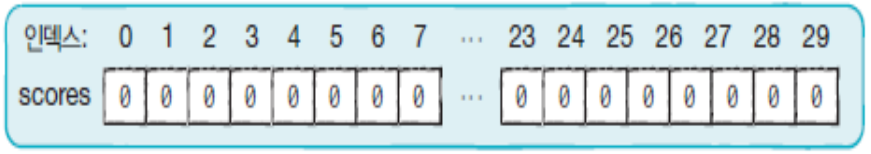
- 배열 변수가 선언된 경우

```
타입[] 변수 = null;
변수 = new 타입[길이];
```

분류	타입	초기값
기본 타입(정수)	byte[] char[] short[] int[] long[]	0 '\u0000' 0 0 0L
기본 타입(실수)	float[] double[]	0.0F 0.0
기본 타입(논리)	boolean[]	false
참조 타입	클래스[]	null
	인터페이스[]	null

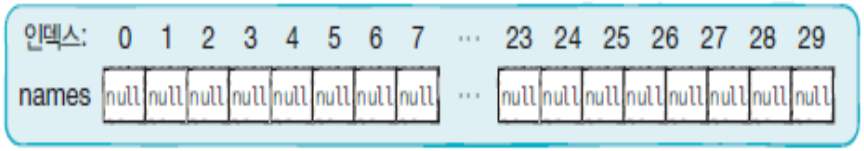
- new 연산자로 배열 처음 생성할 때 배열은 자동적으로 기본값으로 초기화됨
  - int 배열

```
int[] scores = new int[30];
```



  - String 배열

```
String[] names = new String[30];
```



- 배열 생성 후 특정 인덱스 위치에 새 값 저장

```
scores[0] = 83;
scores[1] = 90;
```

- 배열 길이: 배열에 저장할 수 있는 전체 요소

```
int[] intArray = { 10, 20, 30 };
int num = intArray.length;
```

명령 라인 입력

❖ main() 메소드의 String[] args 매개변수

- 실행할 때 명령라인 매개값을 주지 않았을 경우
  - 길이 0인 String 배열 생성 후 main() 메소드 호출

```
String[] args = { };
```

main() 메소드 호출 시 전달

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

- 실행할 때 명령라인 매개값을 주었을 경우

```
[JDK 11 이후 버전] java -p . -m 모듈명/패키지.클래스 문자열0 문자열1 문자열2 … 문자열n-1
[JDK 8 이전 버전]  java 패키지.클래스 문자열0 문자열1 문자열2 … 문자열n-1
```

```
String[] args = { 문자열0, 문자열1, … , 문자열n-1 };
```

main() 메소드 호출 시 전달

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

명령 라인 입력 예제

```
1  package sec02.exam05;
2
3  public class MainStringArrayArgument {
4      public static void main(String[] args) {
5          if(args.length != 2) {
6              System.out.println("값의 수가 부족합니다.");
7              System.exit(0);
8          }
9
10         String strNum1 = args[0];
11         String strNum2 = args[1];
12
13         int num1 = Integer.parseInt(strNum1);
14         int num2 = Integer.parseInt(strNum2);
15
16         int result = num1 + num2;
17         System.out.println(num1 + " + " + num2 + " = " + result);
18     }
19 }
20
```

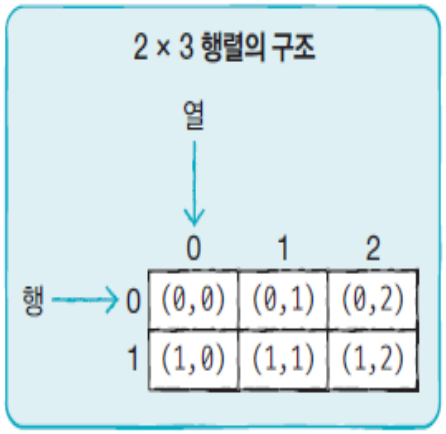
[Run] ➔ [Run Configurations] ➔ [Arguments]

다차원 배열

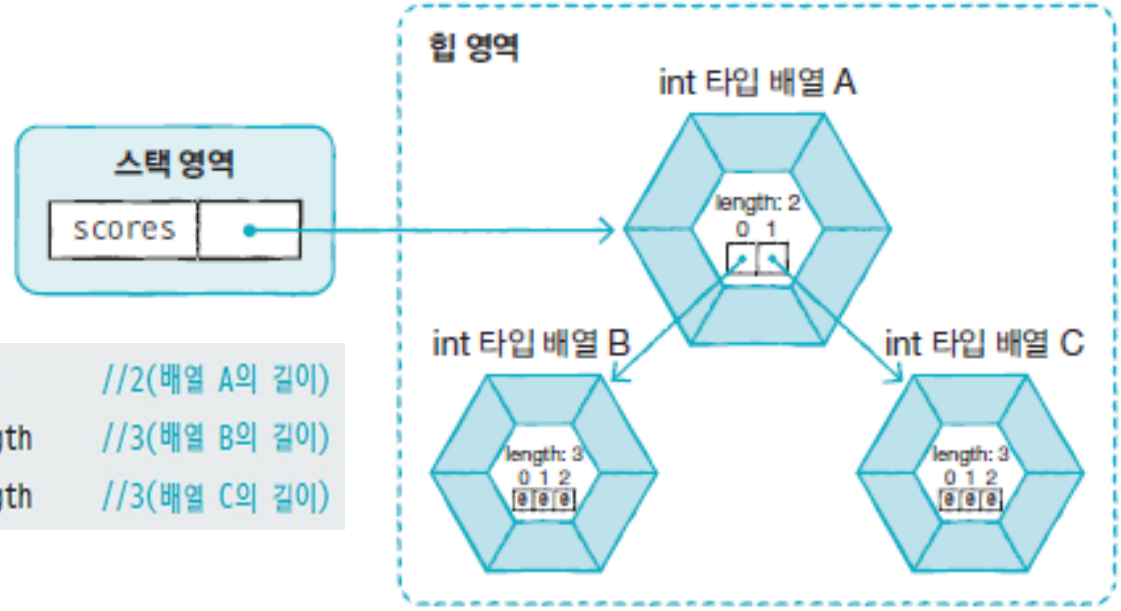
❖ 2차원 배열

- 행렬 구조

```
int[][] scores = new int[2][3];
```



- - 구현 방법: 1차원 배열이 다시 1차원 배열을 참조




```
scores.length       //2(배열 A의 길이)
scores[0].length    //3(배열 B의 길이)
scores[1].length    //3(배열 C의 길이)
```

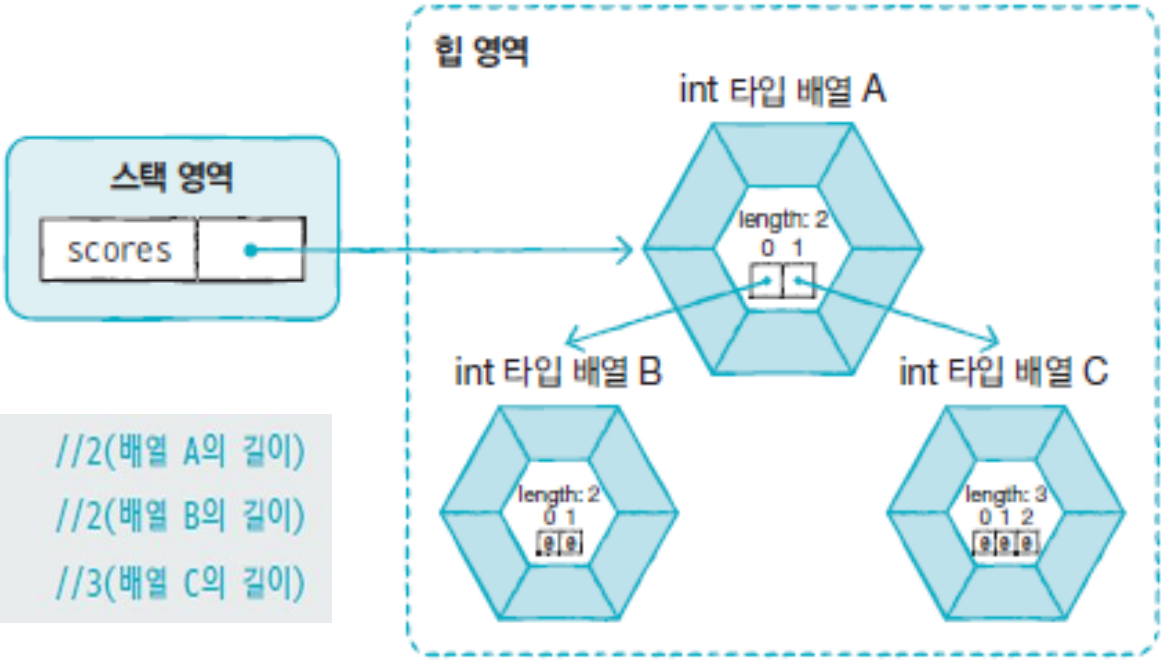
다차원 배열

- 계단식 구조

```
int[][] scores = new int[2][];
scores[0] = new int[2];
scores[1] = new int[3];
```

0	1

0	1	2




```
scores.length       //2(배열 A의 길이)
scores[0].length    //2(배열 B의 길이)
scores[1].length    //3(배열 C의 길이)
```

다차원 배열

- 값 목록을 이용한 2차원 배열 생성

```
타입[ ][ ] 변수 = { {값1, 값2, …}, {값1, 값2, …}, … };
                    ↑              ↑
                 그룹 0 값목록   그룹 1 값목록
```

```
int[ ][ ] scores = { {95, 80}, {92, 96} };
```

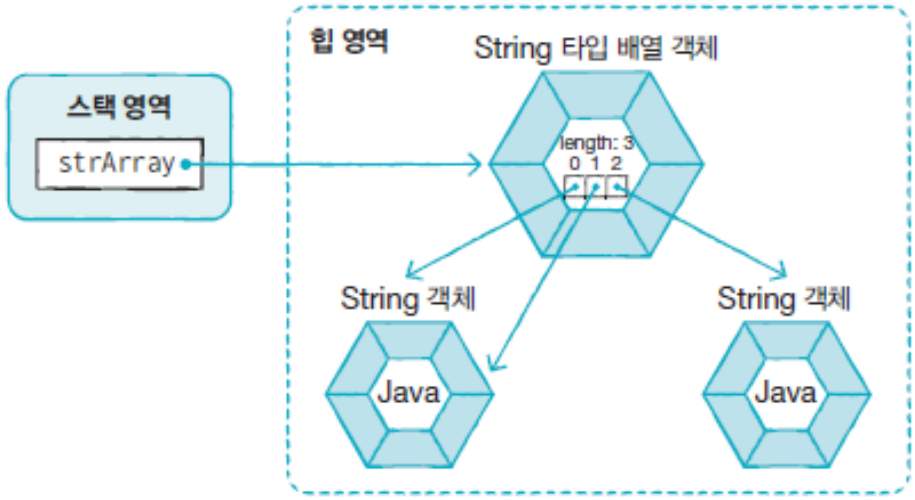
```
int score = scores[0][0];  //95
int score = scores[1][1];  //96
```

객체를 참조하는 배열

❖ 참조 타입 배열

- 요소에 값(정수, 실수, 논리값)을 저장하지 않고, 객체의 번지를 가지고 있음

```
String[] strArray = new String[3];
strArray[0] = "Java";
strArray[1] = "Java";
strArray[2] = new String("Java");
```




```
System.out.println( strArray[0] == strArray[1] );        //true (같은 객체를 참조)
System.out.println( strArray[0] == strArray[2] );        //false (다른 객체를 참조)
System.out.println( strArray[0].equals(strArray[2]) );   //true (문자열이 동일)
```

참조 타입 예제

```
1  package sec02.exam07;
2
3  public class ArrayReferenceObjectExample {
4      public static void main(String[] args) {
5          String[] strArray = new String[3];
6          strArray[0] = "Java";
7          strArray[1] = "Java";
8          strArray[2] = new String("Java");
9
10         System.out.println( strArray[0] == strArray[1]);
11         System.out.println( strArray[0] == strArray[2] );
12         System.out.println( strArray[0].equals(strArray[2]) );
13     }
14 }
15
```

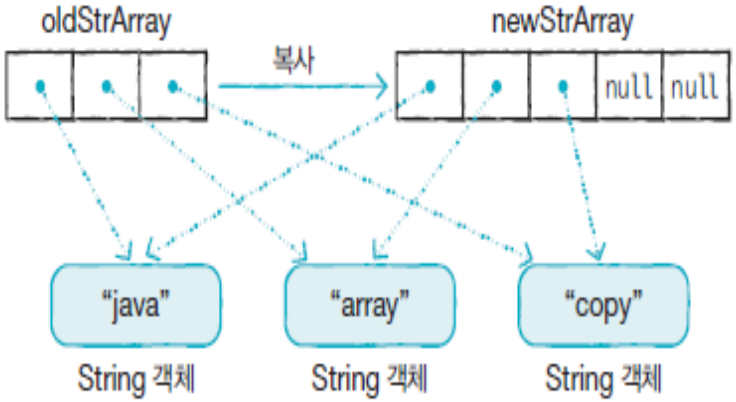
배열 복사

❖ 배열 복사

- for문을 이용해서 요소 하나 하나를 복사
- System.arraycopy()를 이용한 복사

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

```
String[] oldStrArray = { "java", "array", "copy" };
String[] newStrArray = new String[5];

System.arraycopy( oldStrArray, 0, newStrArray, 0, oldStrArray.length );
```



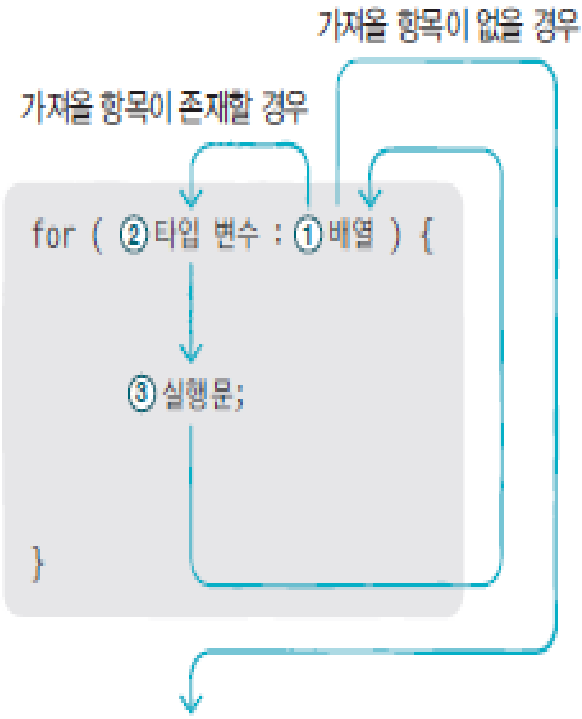
배열 복사 예제

```java
package sec02.exam09;

public class ArrayCopyExample {
    public static void main(String[] args) {
        String[] oldStrArray = { "java", "array", "copy" };
        String[] newStrArray = new String[5];

        System.arraycopy( oldStrArray, 0, newStrArray, 0, oldStrArray.length);

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

향상된 for문

❖ 향상된 for문

- 배열이나 컬렉션을 좀 더 쉽게 처리
- 반복 실행 위해 루프 카운터 변수나 증감식 사용하지 않음




```
int[] scores = { 95, 71, 84, 93, 87 };

int sum = 0;
for (int score : scores) {
  sum = sum + score;
}
System.out.println("점수 총합 = " + sum);
```

Chapter 05

참조 타입

05-3. 열거 타입

혼자 공부하는 자바 (신용권 저)

시작하기 전에

[핵심 키워드] : 열거 타입, 열거 타입 선언, 열거 상수, 열거 타입 변수

[핵심 포인트]

데이터 중에는 몇 가지로 한정된 값만을 갖는 경우가 있다.
이러한 한정된 값을 갖는 타입을 열거 타입이라고 한다.

❖ 열거 타입

- 열거 상수(한정된 값) 를 저장하는 타입

```
Week today;
```

```
today = Week.FRIDAY;
```

Week.java

```
public enum Week {
    MONDAY,
    TUESDAY,
    WEDNESDAY,
    THURSDAY,
    FRIDAY,
    SATURDAY,
    SUNDAY
}
```

열거 타입 이름

열거 상수

열거 타입 선언

❖ 열거 타입 선언

- 소스파일(.java) 생성
- 열거타입 선언

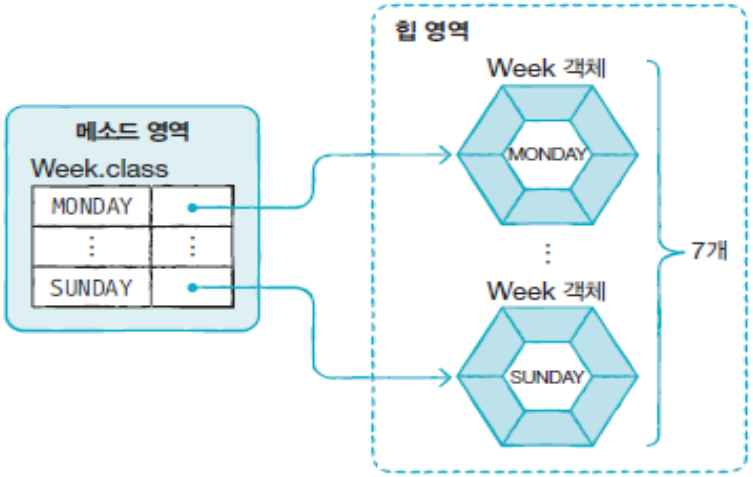
```
public enum 열거타입이름 { … }
```

- 열거 상수 선언
  - Week.java

```
public enum Week { MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY }
```

열거 타입 이름 ↑ — 열거 상수 ↑

- 열거 상수는 열거 객체로 생성



열거 타입 변수

❖ 열거 타입 변수 선언

```
열거타입 변수;
```

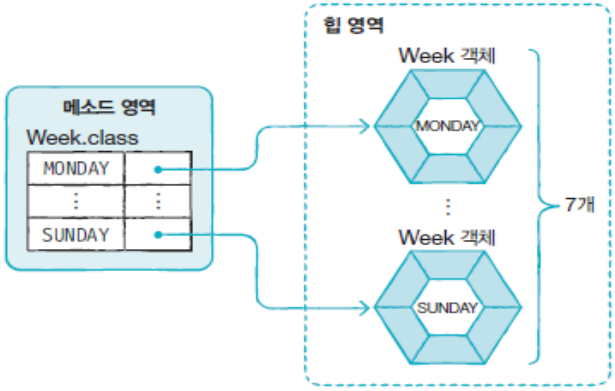
```
Week today;
Week reservationDay;
```

Week.java

```
public enum Week {
    MONDAY,
    TUESDAY,
    WEDNESDAY,
    THURSDAY,
    FRIDAY,
    SATURDAY,
    SUNDAY
}
```

열거 타입 이름

열거 상수



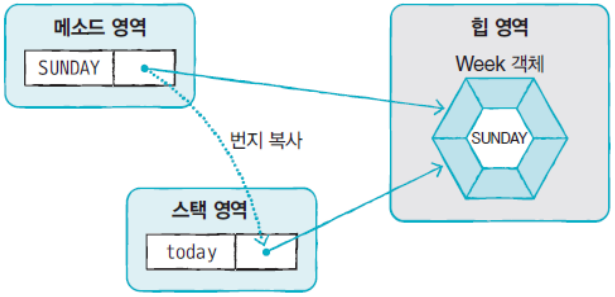
❖ 열거 상수 저장

```
열거타입 변수 = 열거타입.열거상수;
```

```
Week today = Week.SUNDAY;
```

```
today == Week.SUNDAY;   //true
```

```
Week birthday = null;
```




혼자 공부하는 자바
혼자 공부하는 자바
1:1 과외처럼 공부하고 싶을 때
작심삼일, 하다가 포기하게 될 때
어제 배운 내용이 생각나지 않을 때
신용권 지음
JAVA 8&11 지원
245개의 손코딩 예제
무료 동영상 강의
별책 용어 노트
한빛미디어

Thank You!