참조타입

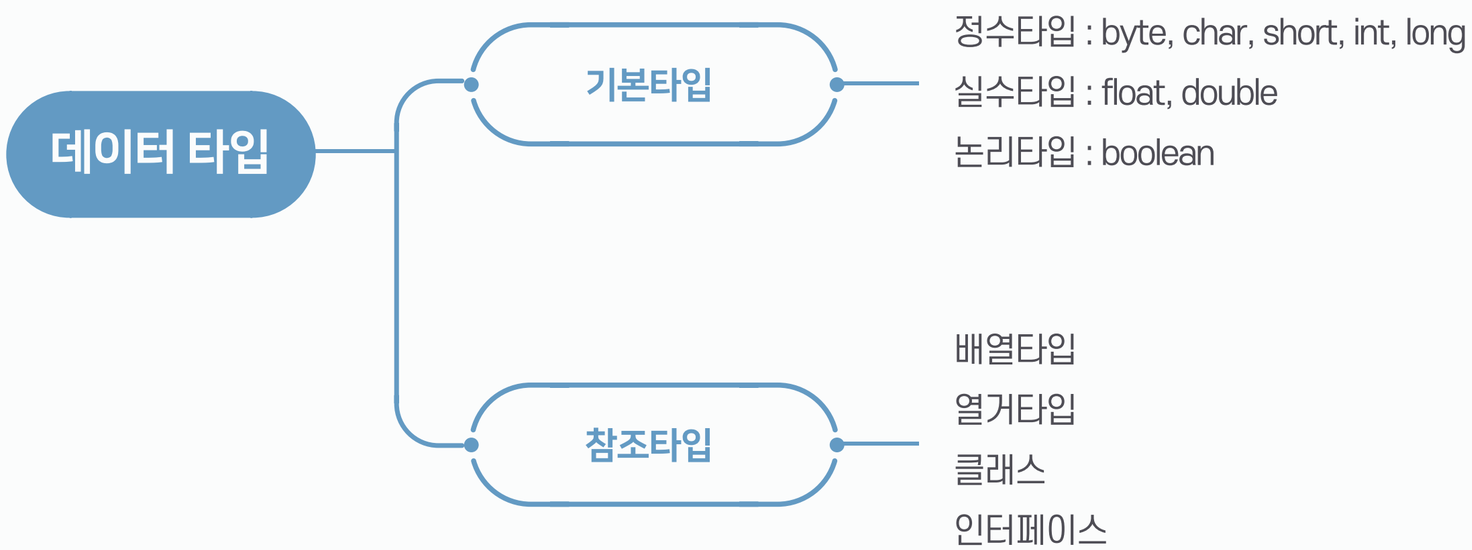
주효진

CONTENTS

Chapter 01

데이터 타입과 분류

데이터 타입 분류



- 기본타입은 선언된 변수에 실제 값을 저장
  참조타입은 선언된 변수에 메모리 번지 값을 저장하여 번지를 통해 객체를 참조

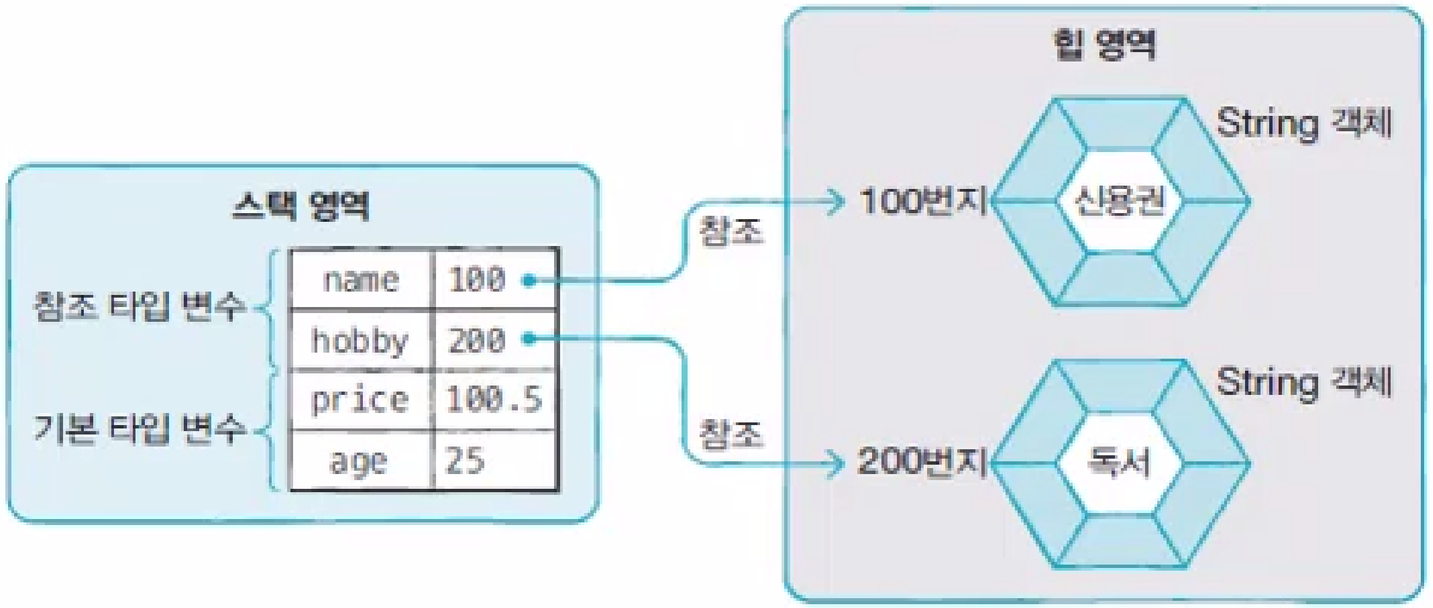
데이터 타입 분류

기본 타입 변수

```
int age = 25;
double price = 100.5;
```

참조 타입 변수

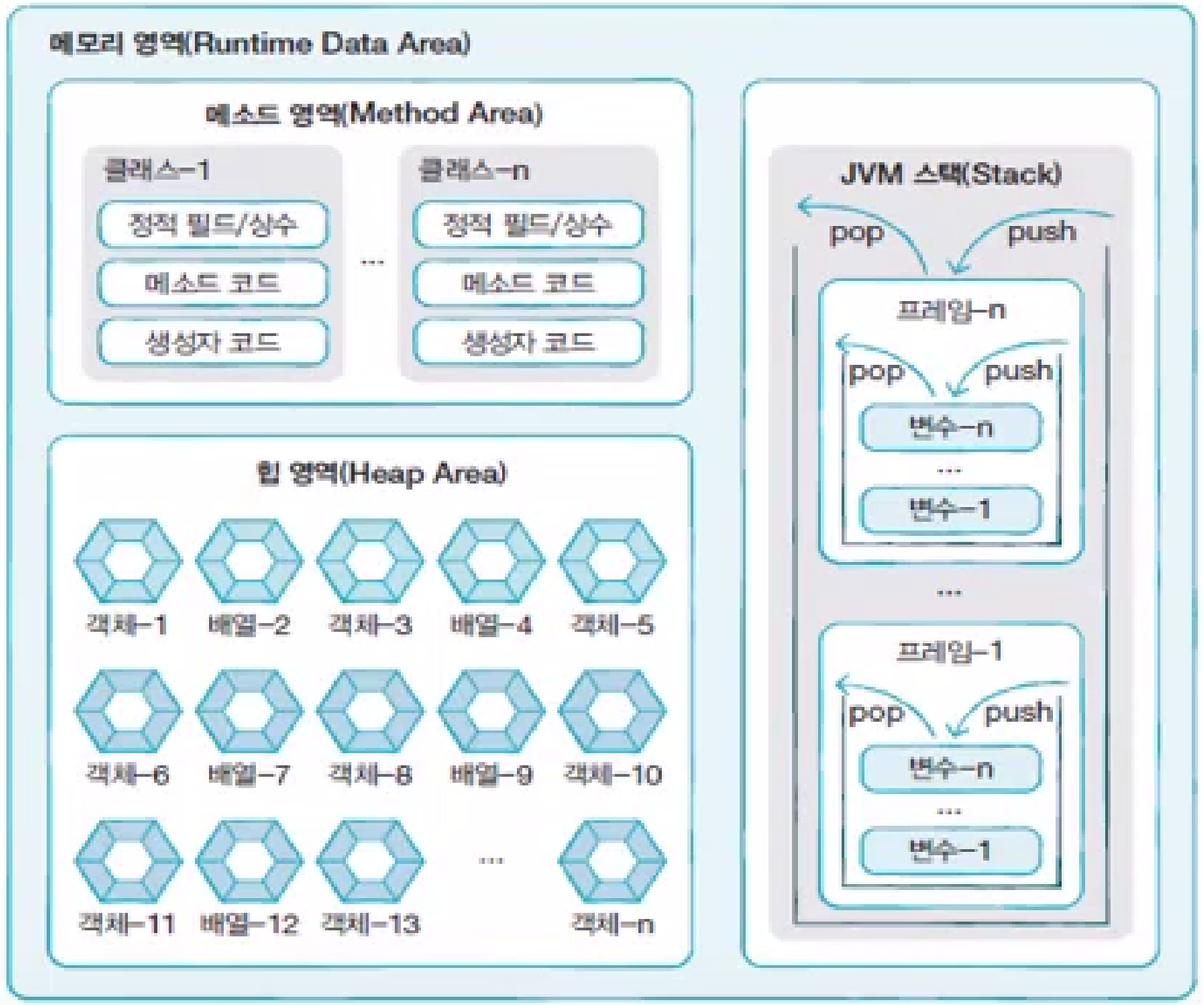
```
String name = "신용권";
String hobby = "독서";
```



Chapter 02

메모리 사용 영역

메모리 사용 영역



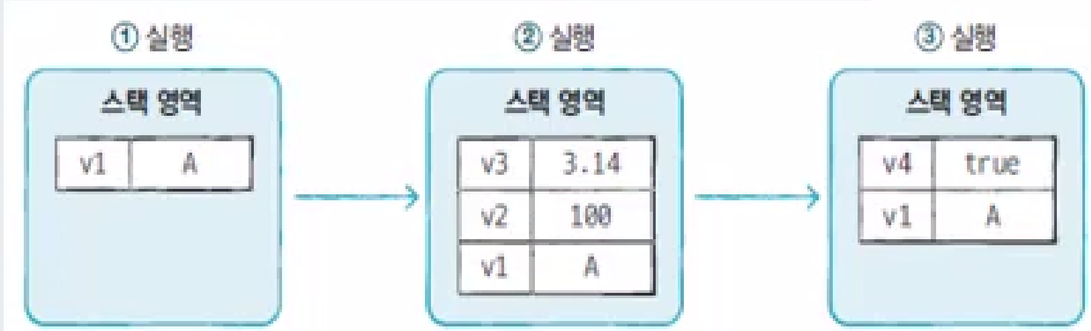
- 메소드 영역
  - 코드에서 사용되는 클래스별로 런타임 상수풀, 필드 데이터, 메소드 데이터, 메소드 코드, 생성자 코드 등을 분류해서 저장
  - 메소드 영역은 JVM이 시작할 때 생성되고 모든 스레드가 공유

- 힙 영역
  - 객체와 배열이 생성되는 영역. 생성된 객체와 배열은 JVM 스택 영역의 변수나 다른 객체의 필드에서 참조됨
  - 참조하는 변수나 필드가 없다면 의미 없는 객체가 되기 때문에 JVM은 Garbage Collecter를 실행시켜 자동으로 제거

- JVM 스택 영역
  - 각 스레드마다 하나씩 존재하며 스레드가 시작될 때 할당. 추가로 스레드를 생성하지 않았다면 main 스레드만 존재하여 스택도 하나

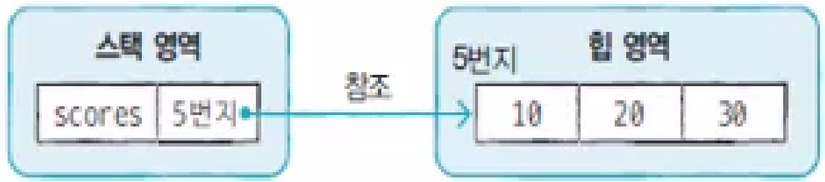
메모리 사용 영역

```
char v1 = 'A';          ——— ①

if (v1 == 'A') {
   int v2 = 100;
   double v3 = 3.14;    ——— ②
}

boolean v4 = true;      ——— ③
```



- 해당 코드를 실행하면 그림과 같은 순서로 스택에 데이터가 생성 및 소멸됨
  v2, v3는 if문 내부에서만 스택에 존재하고 if문을 벗어나면 소멸

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



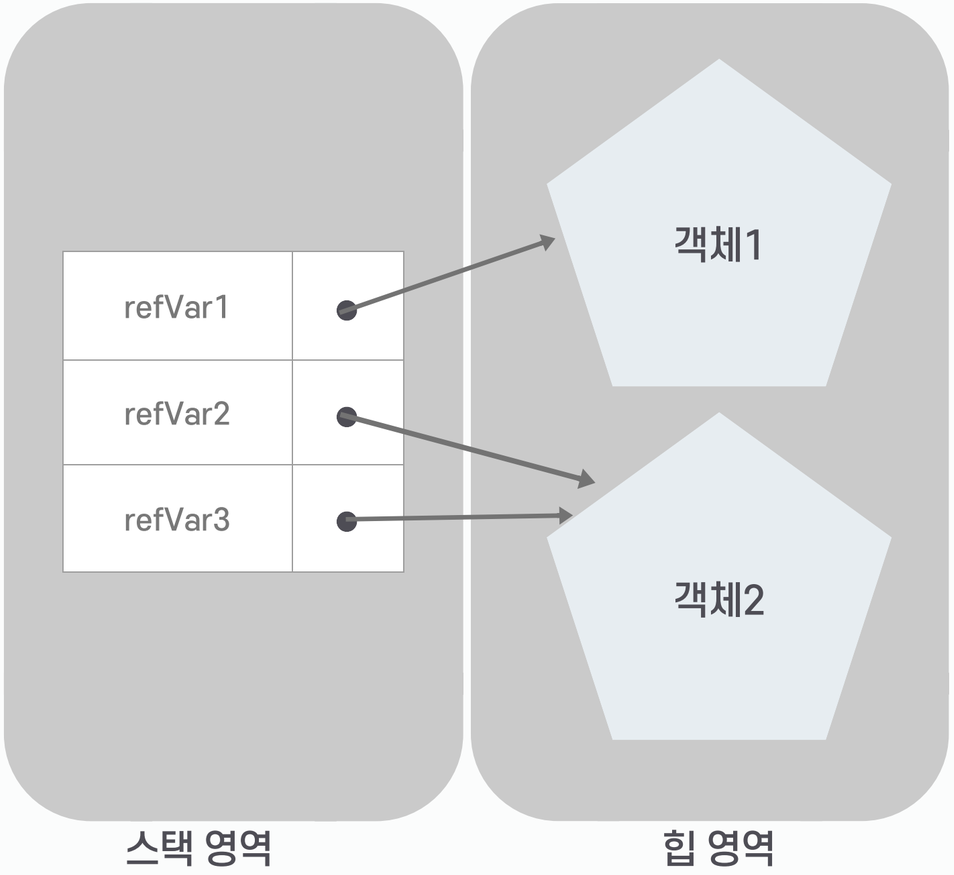
- scores도 스택 영역에 생성되지만 배열은 참조타입 변수이기 때문에 실질적인 값은 힙 영역에 생성되고
  그 생성된 배열의 주소가 스택에 저장

Chapter 03

참조 변수의 ==, != 연산

참조 변수의 ==, != 연산

- 기본 타입의 변수는 값이 같은지 아닌지를 구분할 때 ==, != 연산을 하지만 참조 타입은 동일한 객체를 참조하는지 다른 참조하는지를 확인할 때 사용
  즉, 참조타입은 주소 값을 비교하는 것



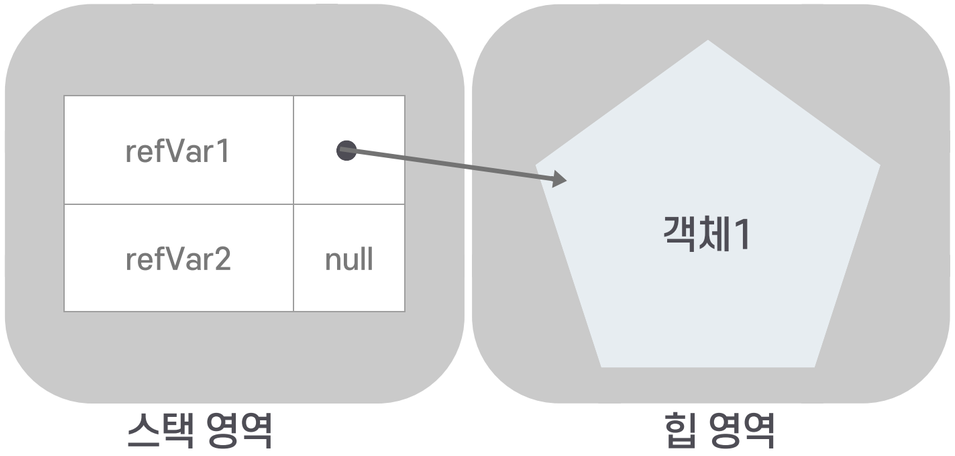
연산	결과
refVar1 == refVar2	false
refVar1 != refVar2	true
refVar2 == refVar3	true
refVar2 != refVar3	false

Chapter 04

null과 NullPointerException

null과 NullPointerException

- 참조 타입 변수는 힙 영역의 객체를 참조하지 않는다는 뜻의 null 값을 가질 수 있음



연산	결과
refVar1 == null	false
refVar1 != null	true
refVar2 == null	true
refVar2 != null	false

null과 NullPointerException

- NullPointerException

  참조 변수 사용 시 가장 많이 발생하는 오류

  null 값을 가지고 있는 참조 타입 변수는 힙 영역에 참조할 객체가 없기 때문

```
int[] arr = null;
arr[0] = 10; // NullpointException
```

배열 arr은 참조 타입 변수여서 null로 초기화가 가능

arr이 참조하는 배열 객체가 없기 때문에 데이터를 입력할 수 없음

```
String str = null;
System.out.print("문자 수 : " + str.length());// NullpointException
```

String은 클래스 타입으로 null로 초기화가 가능

str이 참조하는 String 객체가 없기 때문에 length() 메소드를 호출하면 오류 발생

Chapter 05

String 타입

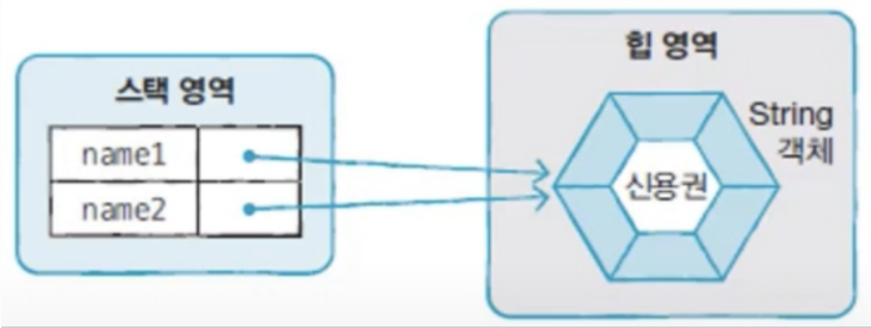
String 타입

- 선언

  String str;

  str = "문자열";

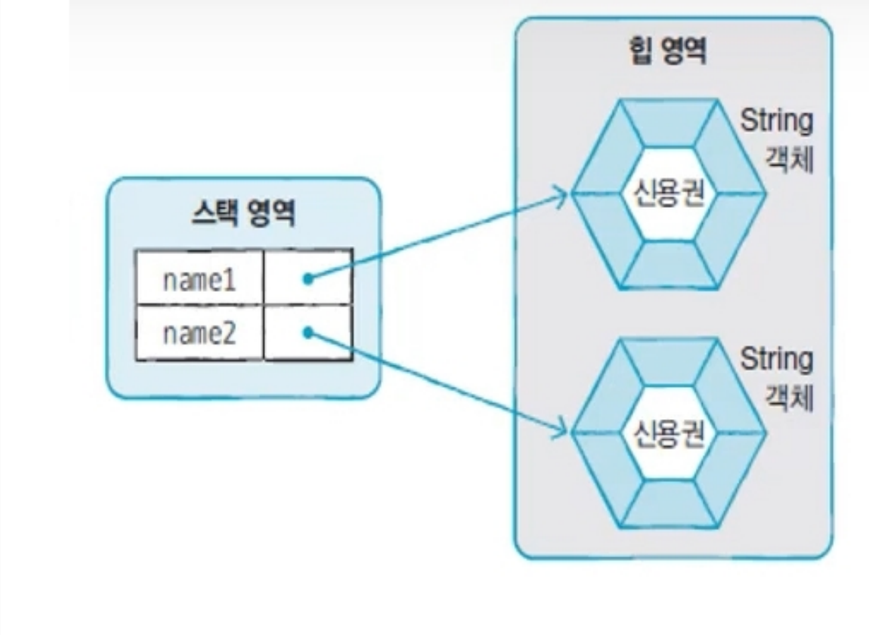
  String str = "문자열";



String name1 = "신용권"

String name2 = "신용권"

String 타입



String name1 = new String("신용권")
String name2 = new String("신용권")

- 서로 참조하는 객체가 같은 지를 확인할 때는 == 연산자를 사용하여 확인
  name1 == name2

  문자열이 같은지 확인을 하기 위해서는 equals() 메소드를 사용해야함
  name1.equals(name2)

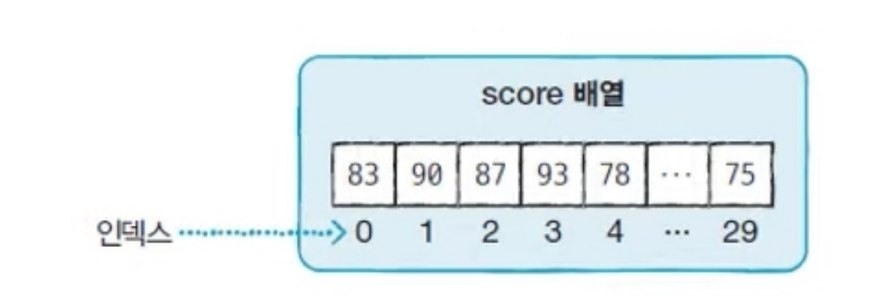
- name1 = null;
  name1에 null 값을 대입하면 더 이상 String 객체를 참조하지 않게되고
  참조를 잃은 String 객체는 쓰레기 객체로 취급하여 Garbage Collector로 자동 제거

Chapter 06

배열 타입

배열 타입

- 배열은 같은 타입의 데이터를 연속된 공간에 나열시키고 각 데이터에 인덱스를 부여해 놓은 자료구조
  같은 타입의 데이터만 저장할 수 있어서 int 배열에는 int 타입의 데이터만 저장이 가능함. 데이터 타입이 다르면 타입 불일치 오류 발생
  선언과 동시에 데이터 타입과 배열의 길이가 정해지기 때문에 수정을 위해선 새로운 배열을 생성하고 데이터를 복사해야 함



- scores[인덱스]
  배열명[인덱스]를 입력하여 데이터에 접근 가능

배열 타입

- 선언

  iint[] arr;

  int arr[];

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

  int[] arr = new int[3]

  arr[0] = 4 // 배열 arr의 0번 인덱스의 데이터를 4로 수정

- 배열의 길이

  arr.length;

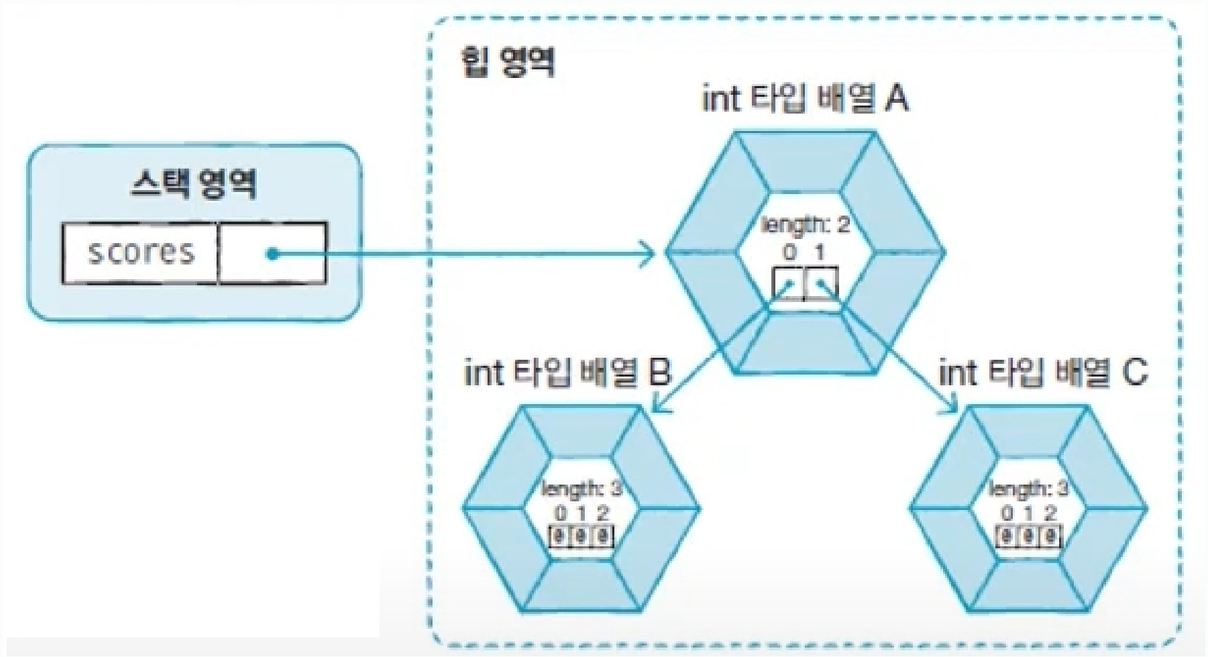
  - length 메소드는 배열의 길이를 확인하는 메소드로 읽기 전용이기 때문에 값을 바꿀 수 없다

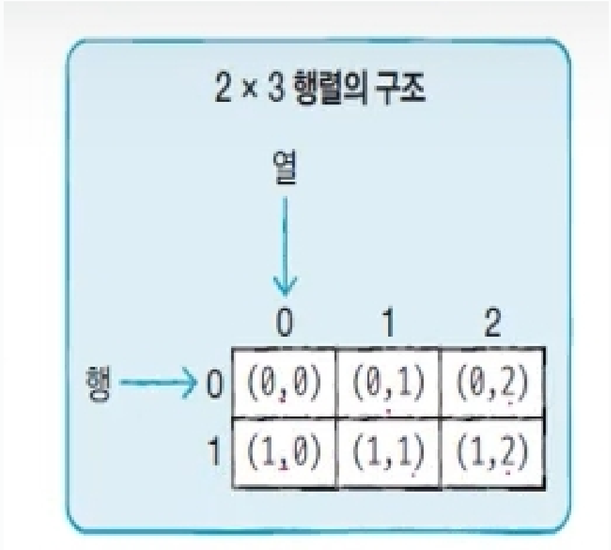
  arr.length = 3 // 오류

배열 타입

- 다차원 배열

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



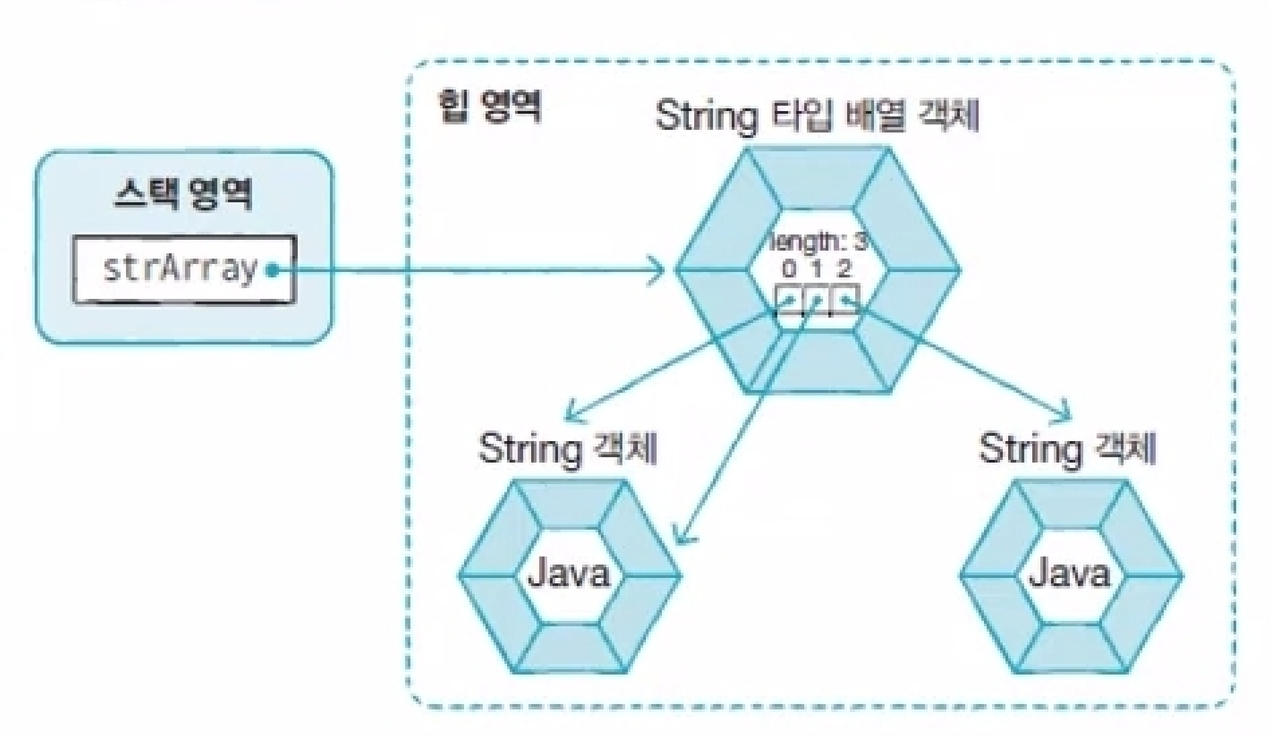


배열 타입

- 객체를 참조하는 배열

  기본 타입의 배열은 각 항목에 직접 값을 가지고 있지만 참조 타입의 배열은 각 항목에 객체의 번지를 가지고 있기 때문에

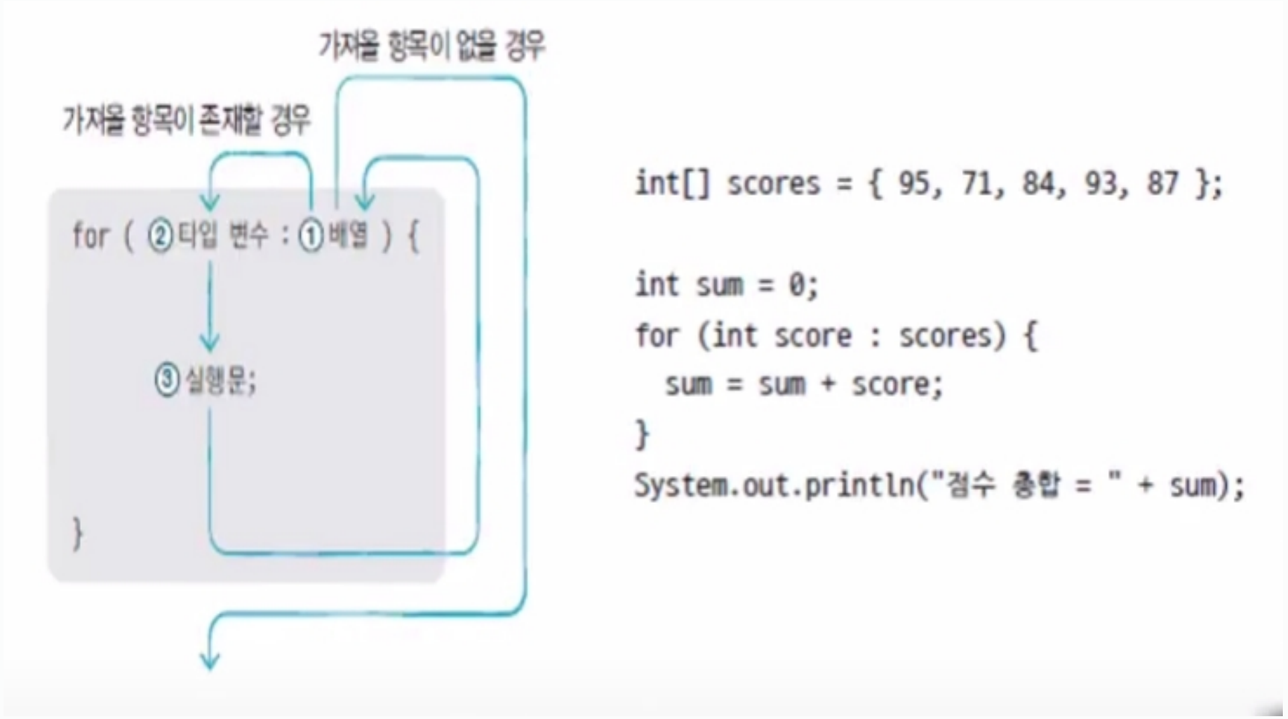
  String[] 배열은 각 문자열을 가지고 있는 것이 아니라 문자열 객체의 주소를 저장하게 됨



배열 타입

- 배열 복사
  for문을 통해 인덱스마다 복사하는 방법과 System.arraycopy()를 이용한 복사
  System.arraycopy(arr1, 0, arr2, 0, arr1.length); //순서대로 원본 배열, 복사할 항목의 시작 인덱스, 복사할 새 배열, 붙여넣을 인덱스, 복사할 개수

- 향상된 for문
  배열 및 컬렉션 객체를 쉽게 처리할 목적으로 사용
  배열 항목의 개수만큼 반복하고 자동적으로 종료



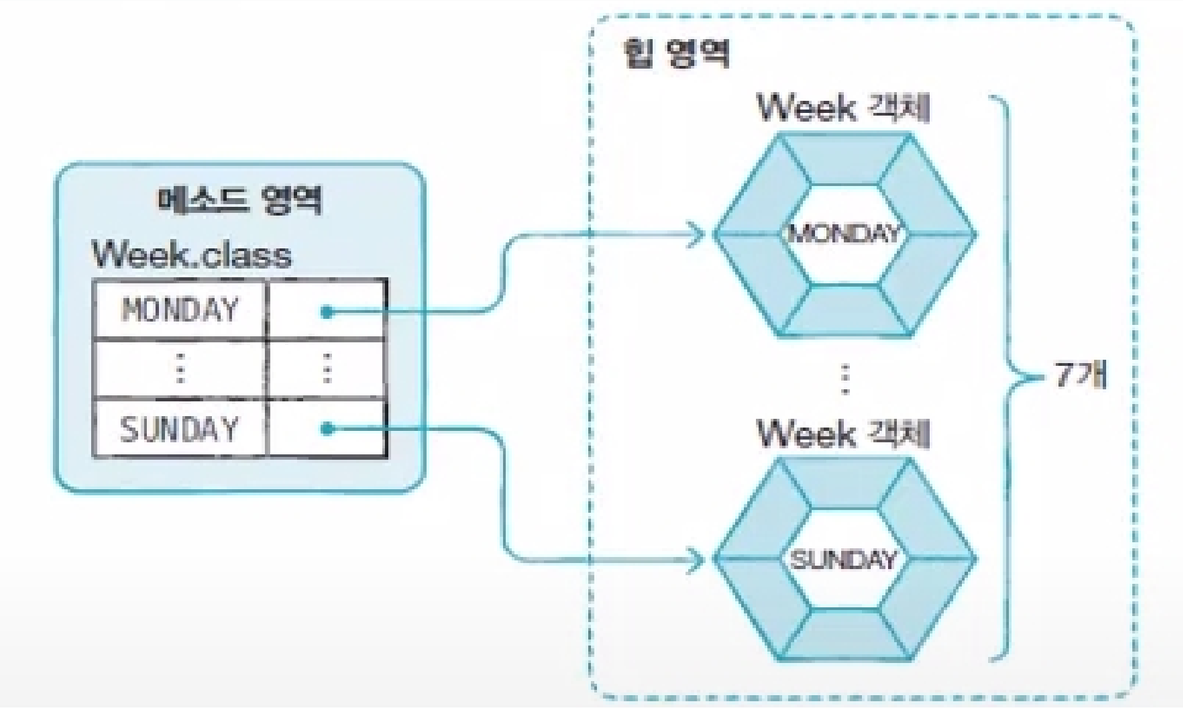

```
int[] scores = { 95, 71, 84, 93, 87 };

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

Chapter 07

열거 타입

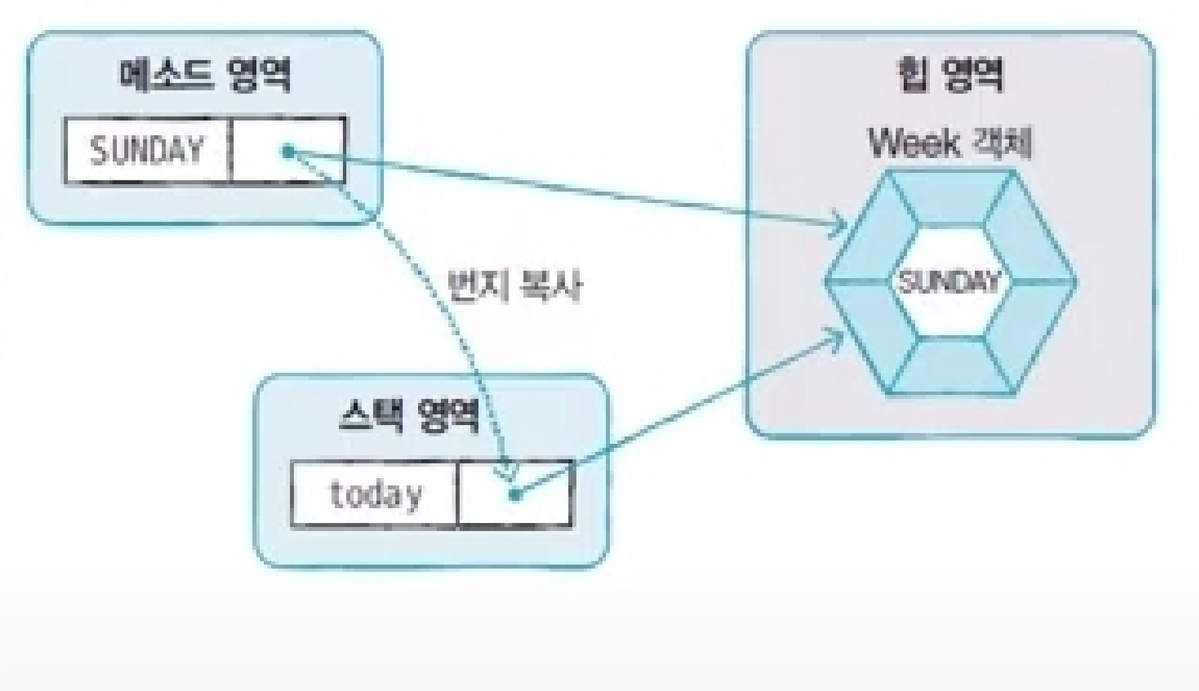
열거 타입

- 한정된 값만 갖는 데이터 타입
  예) 요일에 대한 데이터 월,화,수,목,금,토,일 / 계절에 대한 봄, 여름, 가을, 겨울
  public enum Week {MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY}



열거 타입

- Week today = Week.SUNDAY;
  today == Week.SUNDAY; //true
  Week today1 = Week.SUNDAY;
  today == today1 //true



감사합니다.