Chapter 05

참조 타입

05-1. 참조 타입과 참조 변수

❖ 목차

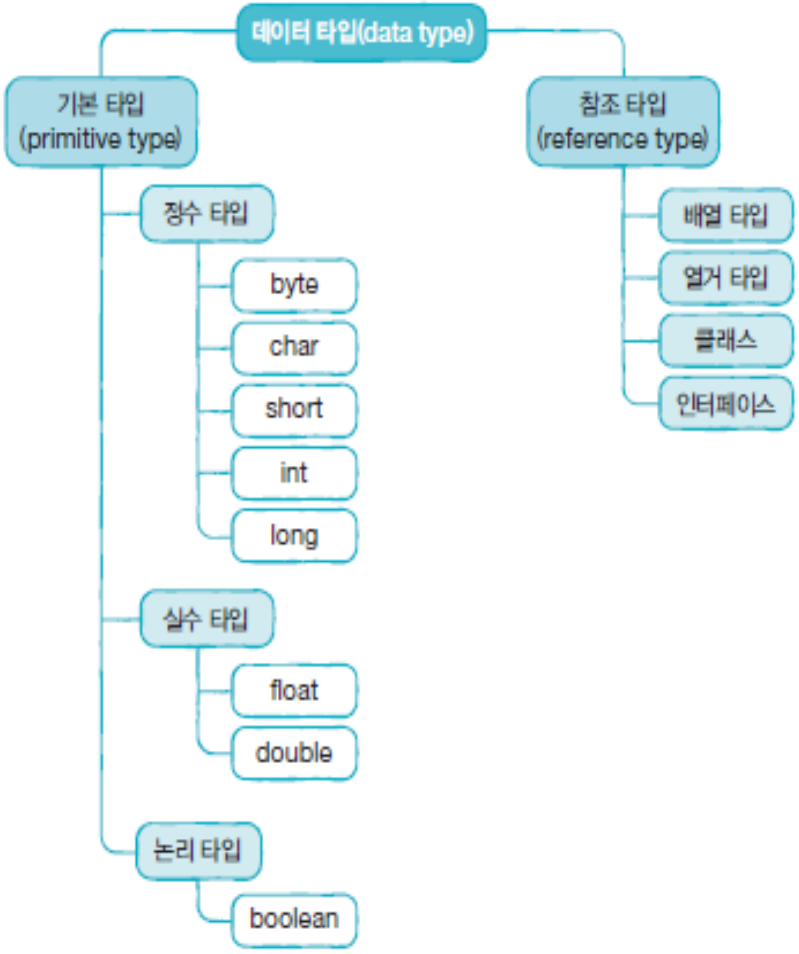
- 시작하기 전에
- 기본 타입과 참조 타입
- 메모리 사용 영역
- 참조 변수의 ==, != 연산
- null과 NullPointException
- String 타입
- 키워드로 끝내는 핵심 포인트

시작하기 전에

[핵심 키워드] : 기본 타입, 참조 타입, 메모리 사용 영역, 번지 비교, null, NullPointExcepti

[핵심 포인트]

참조 타입의 종류와 참조 변수의 역할을 정확히 이해한다.



❖ 기본 타입 (primitive type)
- 정수, 실수, 문자, 논리 리터럴 저장

❖ 참조 타입 (reference type)
- 객체(object)의 번지를 참조하는 타입
- 배열, 열거, 클래스, 인터페이스

기본 타입과 참조 타입

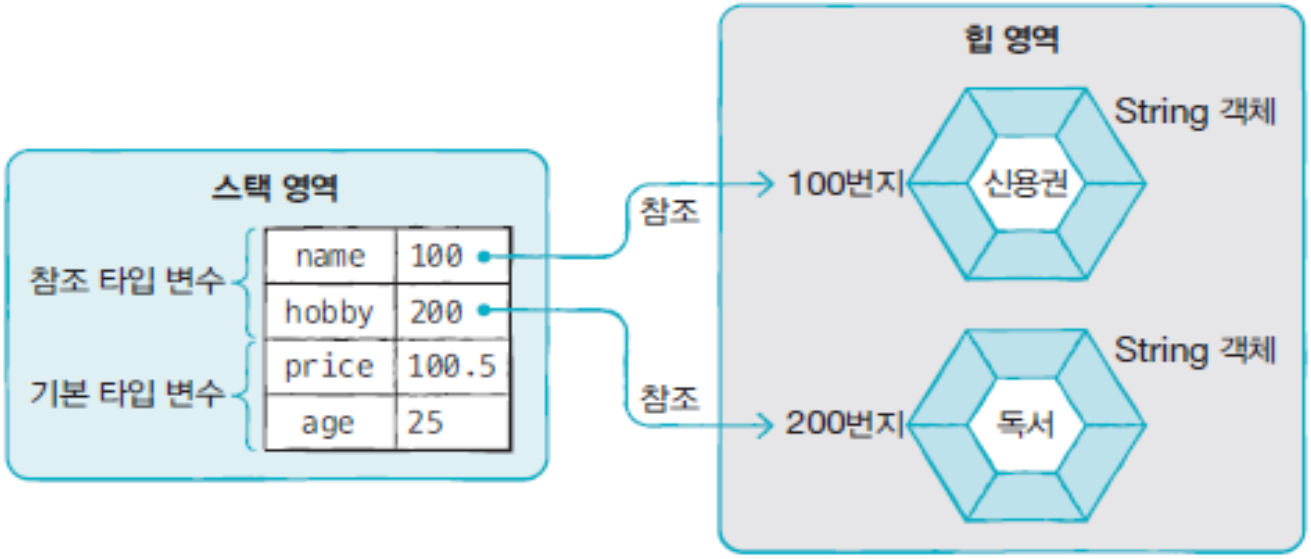
❖ 기본 타입 변수와 참조 타입 변수의 차이점

기본 타입 변수

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

참조 타입 변수

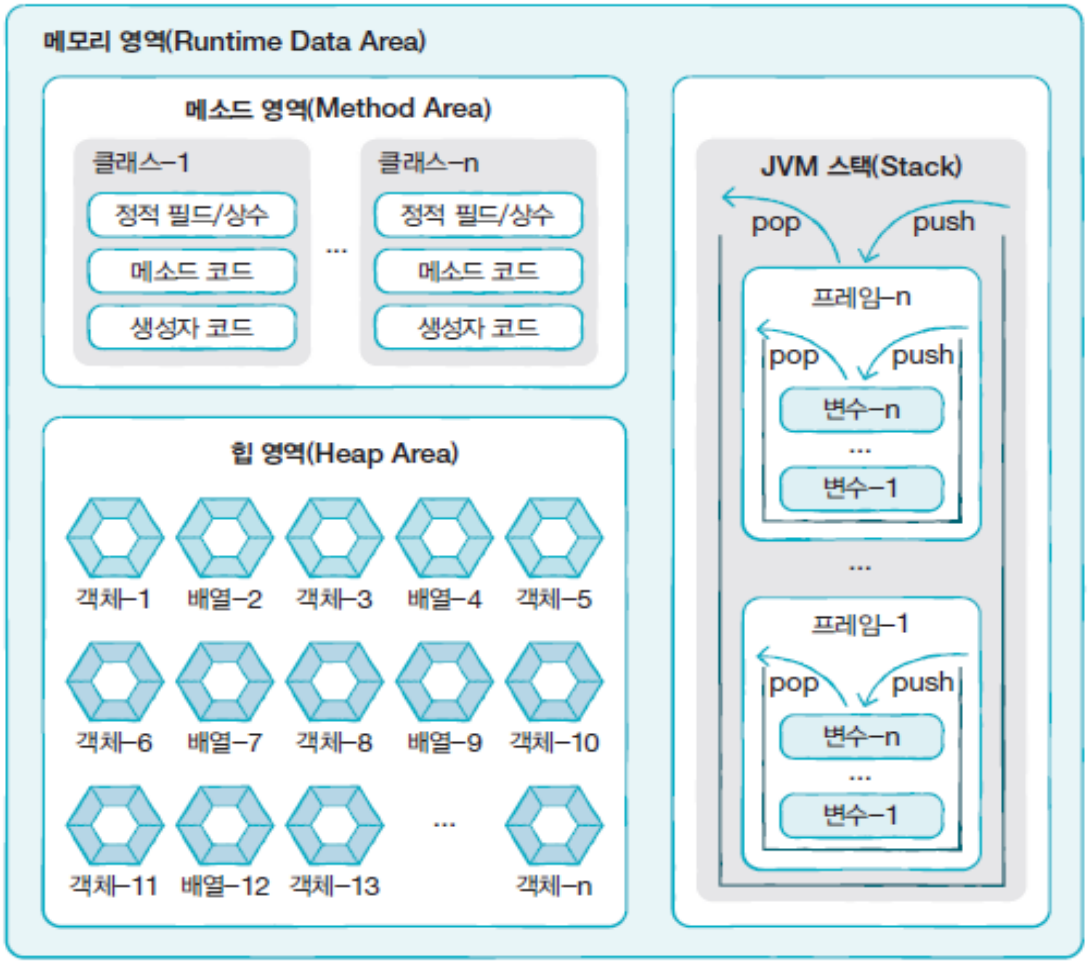
```
String name = "신용권";
String hobby = "독서";
```



메모리 사용 영역

❖ 메모리 사용 영역 (Runtime Data Area)

- 메소드 영역 (Method Area)
  - -클래스별로
  - 정적 필드(static field),
  - 상수(constant),
  - 생성자(constructor)
  - 메소드(method)
  - 코드 등을 분류해 저장
- 힙 영역 (Heap Area)
  - -객체와 배열이 생성되는 영역
- JVM 스택 영역
  - -메소드가 호출되면
  - 프레임이 추가되고,
  - -메소드 종료되면
  - 프레임이 제거됨

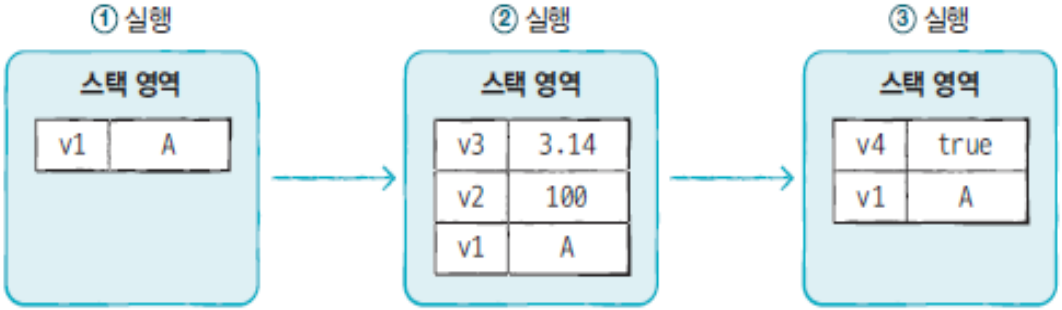


메모리 사용 영역

❖ JVM 스택 영역

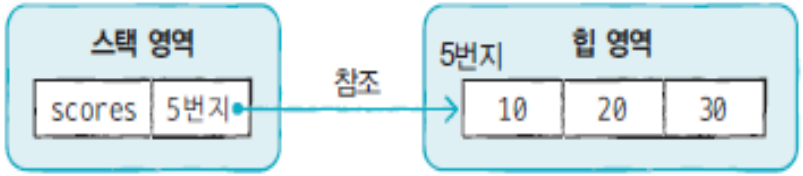
- 메소드를 호출할 때마다 프레임이 추가되고, 메소드가 종료되면 해당 프레임이 제거
  - 프레임 내부의 변수 스택 이해

```
char v1 = 'A';          ——— ①

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

boolean v4 = true;      ——— ③
```



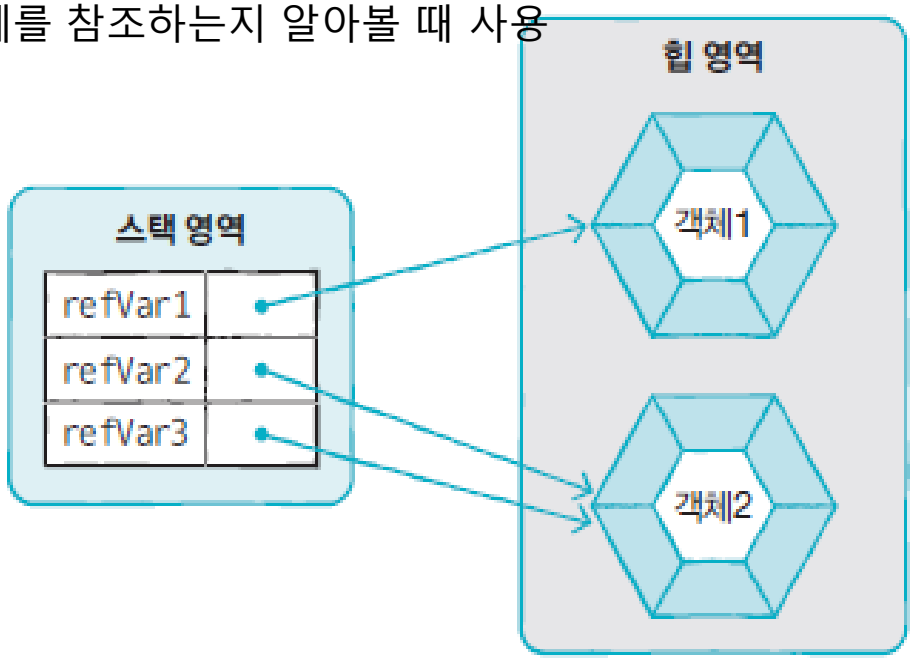
  - 참조 타입 변수는 스택 영역에 힙 영역에 생성된 객체의 주소 가짐

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



참조 변수의 ==, != 연산

❖ 참조 타입 변수 간의 ==, != 연산
- 동일 객체를 참조하는지, 다른 객체를 참조하는지 알아볼 때 사용
- 번지 값의 비교
  - ==
  - 같으면 true
  - 다르면 false
  - !=
  - 같으면 false
  - 다르면 true

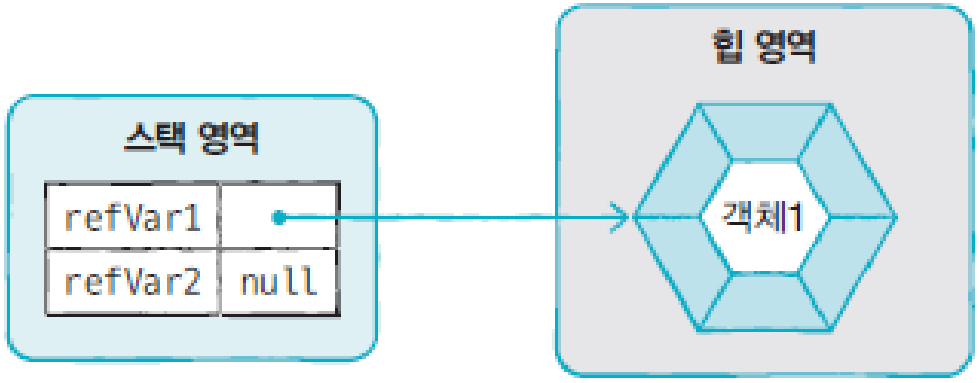



```
refVar1 == refVar2    //결과: false
refVar1 != refVar2    //결과: true
```

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

null과 NullPointerException

- 참조 타입 변수는 객체를 참조하지 않는다는 뜻으로 null 값 가질 수 있음
  - null로 초기화된 참조변수도 스택 영역에 생성




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

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

null과 NullPointerException

❖ 예외 (Exception)

- 프로그램 실행 도중 발생하는 오류

❖ NullPointException

- 참조 타입 변수가 null 상태에서 존재하지 않는 객체의 데이터나 메소드 사용할 경우 발생
- 해당 참조 변수가 객체를 참조하도록 수정하여 해결

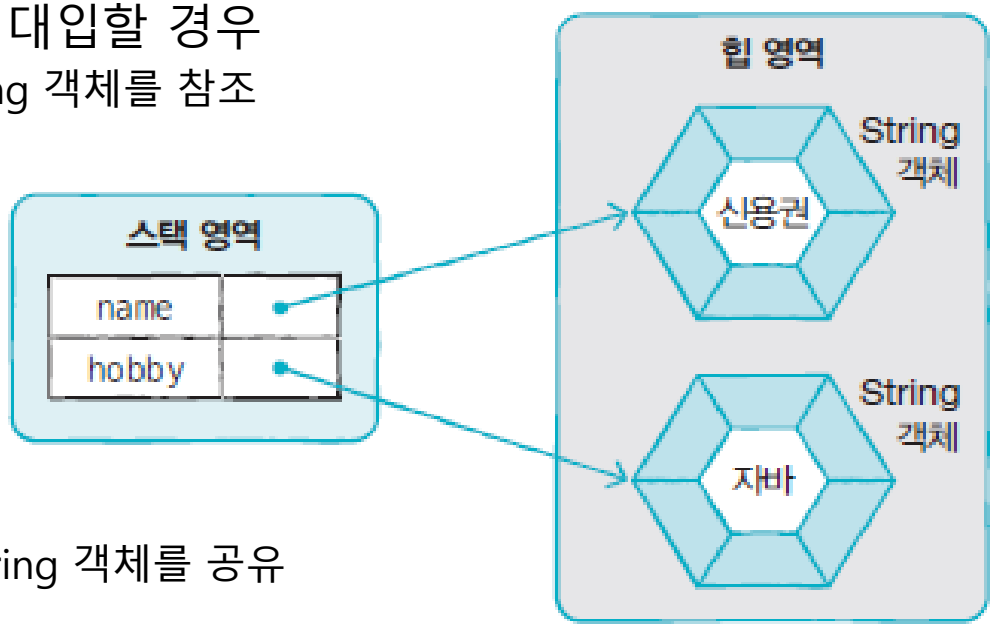
```
int[] intArray = null;
intArray[0] = 10;    //NullPointerException
```

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

String 타입

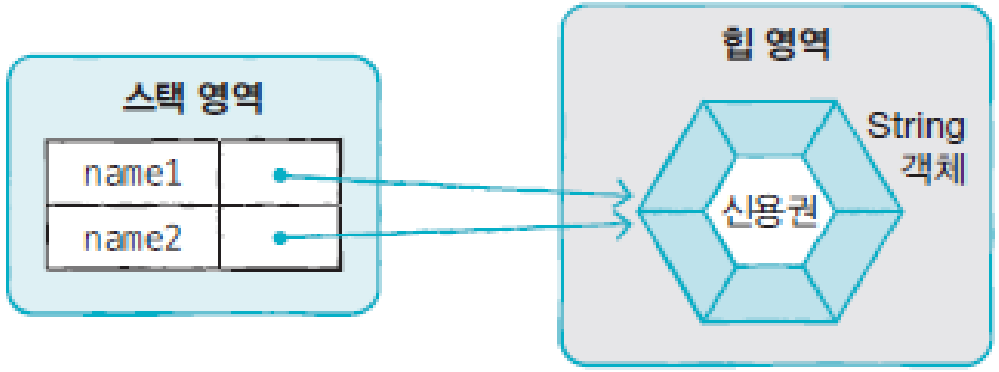
❖ String 변수에 문자열 리터럴을 대입할 경우
- String 객체로 생성되고 변수가 String 객체를 참조

String name = "신용권";
String hobby = "자바";



- 문자열 리터럴 동일한 경우 같은 String 객체를 공유

```
String name1 = "신용권";
String name2 = "신용권";
```

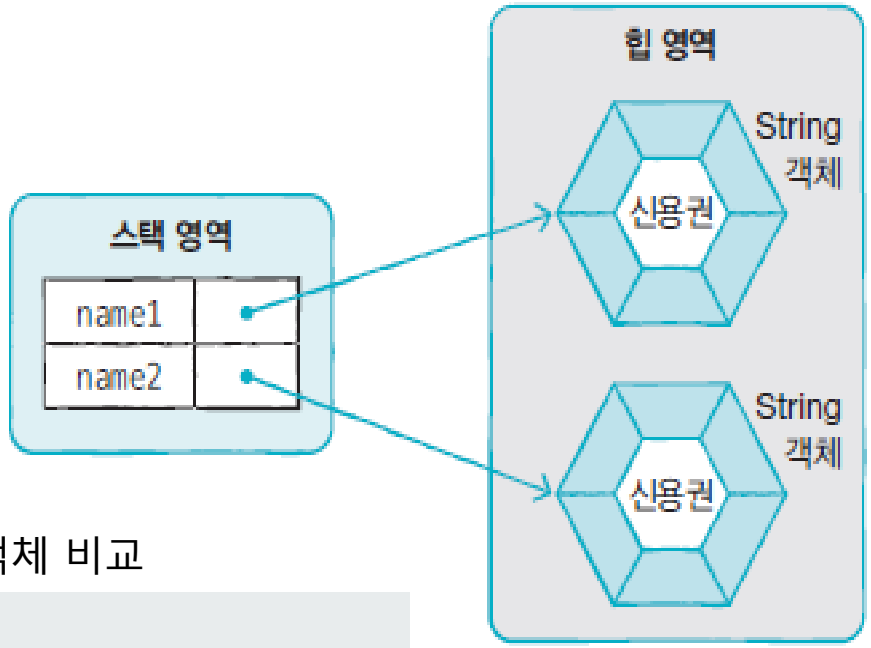


String 타입

❖ new 연산자

- 객체 생성 연산자
- 힙 영역에 새로운 String 객체를 생성

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



- 문자열 리터럴과 new 연산자로 생성된 객체 비교

```
String name1 = "신용권";
String name2 = "신용권";
String name3 = new String("신용권");
```

  - name1 == name2 : true
  - name1 == name3 : false

String 타입

❖ 문자열 비교

- == : 번지 비교 (X)
- equals(): 문자열 비교 (O)

```
boolean result = str1.equals(str2);
                 ↑           ↑
              원본 문자열   비교 문자열
```

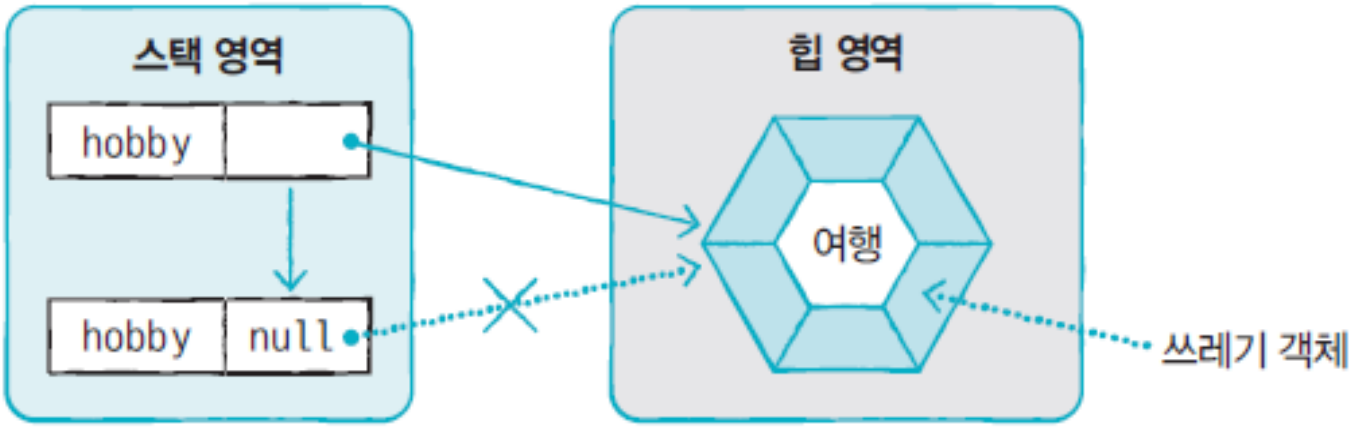
String 타입

- String 변수 초기값으로 null 대입
  - String 변수가 참조하는 객체가 없음을 의미

```
String hobby = null;
```

```
String hobby = "여행";
hobby = null;
```

- 참조를 잃은 String 객체는 쓰레기 수집기 (Garbage Collector) 통해 메모리에서 자동 제거



String 타입

```
1  package sec01.exam01;
2
3  public class StringEqualsExample {
4      public static void main(String[] args) {
5          String strVar1 = "이순신";
6          String strVar2 = "이순신";
7
8          if(strVar1 == strVar2){
9              System.out.println("strVar1과 strVar2는 참조가 같음");
10         } else {
11             System.out.println("strVar1과 strVar2는 참조가 다름");
12         }
13
14         if(strVar1.equals(strVar2)){
15             System.out.println("strVar1과 strVar2는 문자열이 같음");
16         }
17
18         String strVar3 = new String("세종대왕");
19         String strVar4 = new String("세종대왕");
```

String 타입

```
20
21              if(strVar3  ==  strVar4){
22                      System.out.println("strVar3과  strVar4는  참조가  같음");
23              }  else  {
24                      System.out.println("strVar3과  strVar4는  참조가  다름");
25              }
26
27              if(strVar3.equals(strVar4)){
28                      System.out.println("strVar3과  strVar4는  문자열이   같음");
29              }
30      }
31  }
```

키워드로 끝내는 핵심 포인트

- 기본 타입: byte, short, char, int, long, float, double, boolean 타입.
  변수에 값 자체를 저장

- 참조 타입: 기본 타입을 제외한 배열, 열거, 클래스, 인터페이스.
  변수에 객체의 번지 저장

- 메모리 사용 영역 : JVM은 운영체제에서 할당받은 메모리 영역을
  메소드 영역, 힙 영역, 스택 영역으로 구분해서 사용.

- 번지 비교 : 비교 연산자(==, !=)가 기본 타입에서 사용되면 값을 비교하지만
  참조 타입에서 사용되면 번지를 비교

- null : 참조 변수는 객체를 참조하지 않는다는 뜻으로 null 값을 가질 수 있음

- NullPointException : 참조 변수가 null 일 때
  참조 변수를 통해 존재하지도 않는 객체를 사용하려고 할 경우 발생

Chapter 05

참조 타입

05-2. 배열

❖ 목차

- 시작하기 전에
- 배열이란
- 배열 선언
- 배열 생성
- 배열 길이
- 명령 라인 입력
- 다차원 배열
- 객체를 참조하는 배열
- 배열 복사
- 향상된 for문
- 키워드 핵심 포인트

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;
```

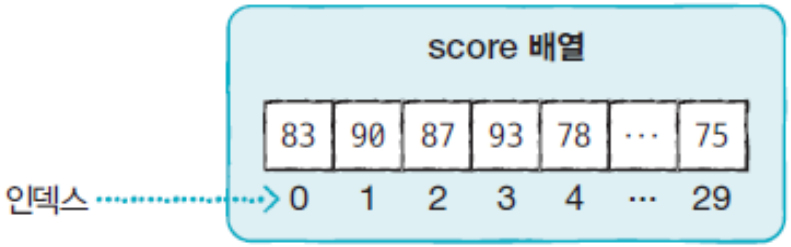
```
int sum = score1;
sum += score2;
sum += score3;
...
sum += score30;
int avg = sum / 30;
```

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, … };
```

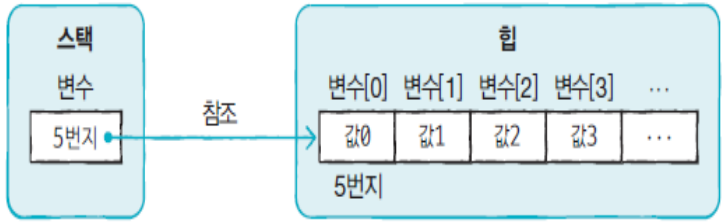
- new 연산자를 이용해서 배열 생성

```
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

```
1  package sec02.exam01;
2
3  public class ArrayCreateByValueListExample1 {
4          public static void main(String[] args) {
5                  int[] scores = { 83,90,87};
6
7                  System.out.println("scores[0] : " + scores[0]);
8                  System.out.println("scores[1] : " + scores[1]);
9                  System.out.println("scores[2] : " + scores[2]);
10
11                 int sum = 0;
12                 for(int i=0; i<3; i++) {
13                         sum += scores[i];
14                 }
15                 System.out.println("총합 : " + sum);
16                 double avg = (double) sum / 3;
17                 System.out.println("평균 : " + avg);
18         }
19 }
```

배열 생성

❖ new 연산자로 배열 생성

```
타입[] 변수 = new 타입[길이];
```

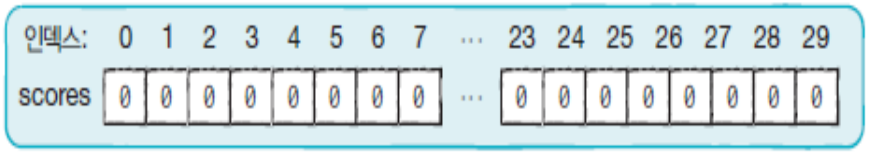
- 배열 변수가 선언된 경우

```
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

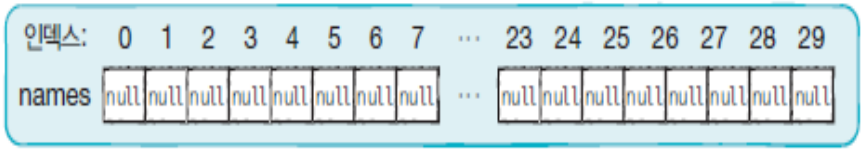
- new 연산자로 배열 처음 생성할 때 배열은 자동적으로 기본값으로 초기화됨
  - int 배열

```
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 수

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

배열 길이

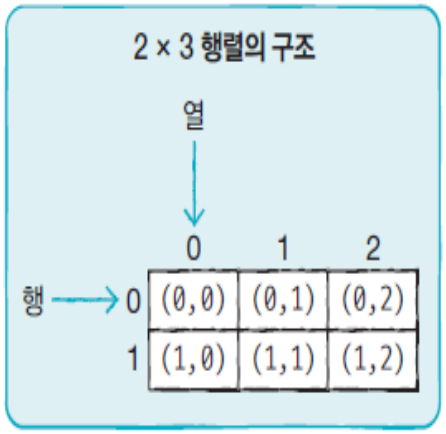
```java
package sec02.exam04;

public class ArrayLengthExample {
	public static void main(String[] args) {
		int[] scores = { 83,90,87};

		int sum = 0;
		for(int i=0; i<scores.length; i++) {
			sum += scores[i];
		}
		System.out.println("총합 :" + sum);

		double avg = (double) sum / scores.length;
		System.out.println("평균 :" + avg);
	}
}
```

다차원 배열

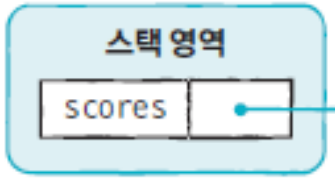
❖ 2차원 배열

- 행렬 구조

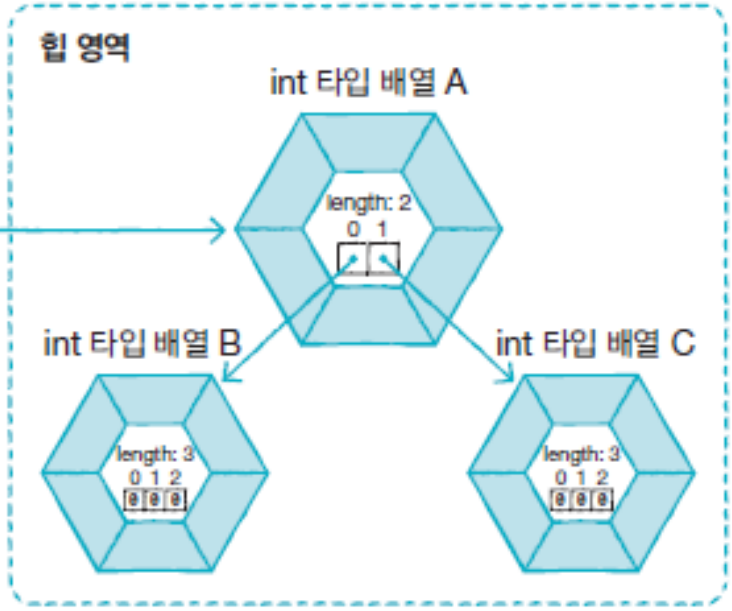
```
int[][] scores = new int[2][3];
```



- - 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)
```



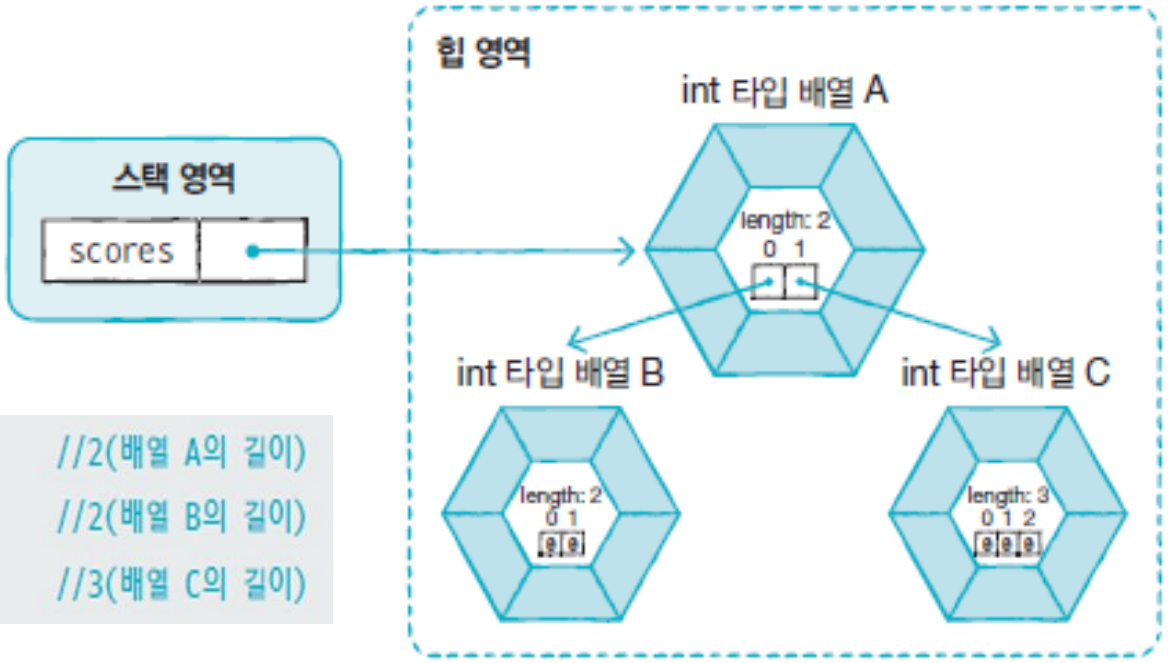
다차원 배열

- 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
```

배열 속의 배열

```
package sec02.exam06;

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

int[][] mathScores = new int[2][3];
for(int i=0; i<mathScores.length; i++) {
for(int k=0; k<mathScores[i].length; k++) {
System.out.println("mathScores["+i+"]["+k+"]="
+mathScores[i][k]);
}
}
System.out.println();
int[][] englishScores = new int[2][];
englishScores[0] = new int[2];
englishScores[1] = new int[3];
```

배열 속의 배열

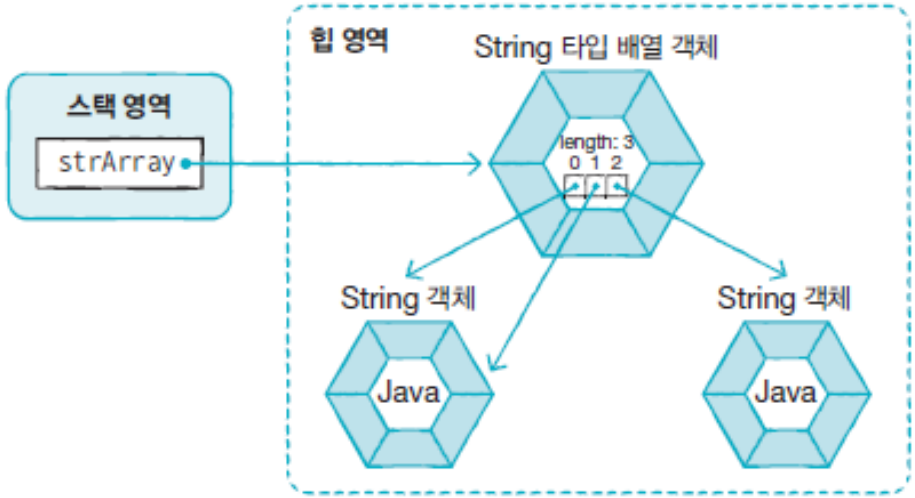
```
for(int i=0; i<englishScores.length; i++) {
for(int k=0; k<englishScores[i].length; k++) {
System.out.println("englishScores["+i+"]["+k+"]=" + englishScores[i][k]);
}
}
System.out.println();

int[][] javaScores = { {95, 80}, {92, 96, 80} };
for(int i=0; i<javaScores.length; i++) {
for(int k=0; k<javaScores[i].length; k++) {
System.out.println("javaScores["+i+"]["+k+"]="
+javaScores[i][k]);
}
}
}
}
```

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)
```

객체를 참조하는 배열

```
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 }
```

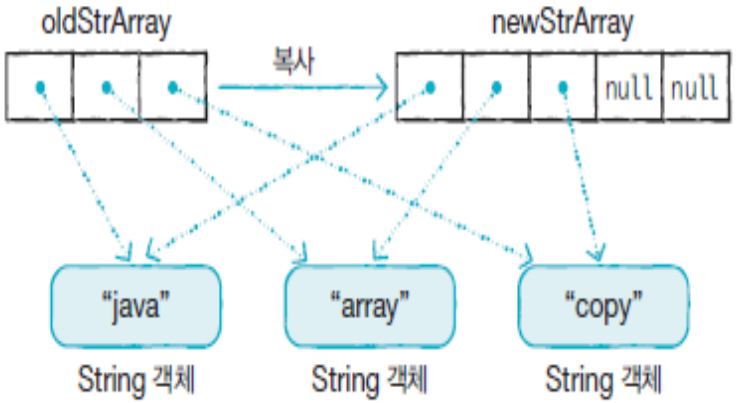
배열 복사

❖ 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 );
```



for문으로 배열 복사

```
1  package sec02.exam08;
2
3  public class ArrayCopyByForExample {
4      public static void main(String[] args) {
5          int[] oldIntArray = { 1,2,3};
6          int[] newIntArray = new int[5];
7
8          for(int i=0; i<oldIntArray.length; i++) {
9              newIntArray[i] = oldIntArray[i];
10         }
11
12         for(int i=0; i<newIntArray.length; i++) {
13             System.out.print(newIntArray[i] + ", ");
14         }
15     }
16 }
```

System.arraycopy()로 배열 복사

```
1  package sec02.exam09;
2
3  public  class ArrayCopyExample {
4          public  static  void main(String[]  args)  {
5                  String[]  oldStrArray  =  {  "java",  "array",  "copy"  };
6                  String[]  newStrArray  =  new String[5];
7
8                  System.arraycopy(  oldStrArray,  0,  newStrArray,  0,  oldStrArray.length);
9
10                 for(int  i=0;  i<newStrArray.length;  i++)  {
11                         System.out.print(newStrArray[i]  +  ", ");
12                 }
13         }
14 }
```

키워드로 끝내는 핵심 포인트

- 배열: 같은 타입의 데이터를 연속된 공간에 나열하고, 각 데이터에 인덱스 부여한 자료구조
- 인덱스: 0부터 시작해 0~(배열길이 -1)까지 범위 가짐
- 배열 선언 : 타입[] 변수; 형태로 선언.
- 배열 생성 : { 값1, 값2, 값3, ... }과 같이 값 목록으로 생성하거나, new 타입[길이];로 생성
- 배열 길이 : 배열에 저장될 수 있는 항목의 수. 배열변수.length
- 다차원 배열 : 타입[][] 변수 = new타입[길이1][길이2]; 로 생성.

Chapter 05

참조 타입

05-3. 열거 타입

❖ 목차

- 시작하기 전에
- 열거 타입 선언
- 열거 타입 변수
- 키워드로 확인하는 핵심 포인트

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 { … }
```

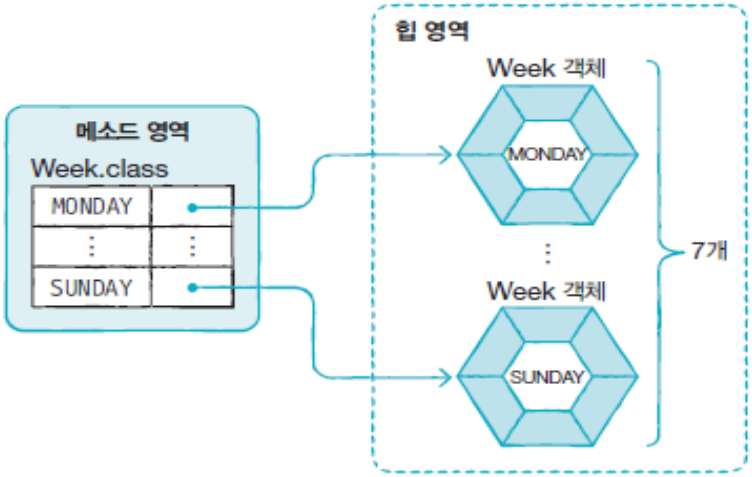
- 열거 상수 선언
  - Week.java

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

열거 타입 이름 ↑ Week

열거 상수 ↑ MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY

- 열거 상수는 열거 객체로 생



열거타입 선언

```
1  package sec03.exam01;
2
3  public  enum  Week {
4          MONDAY,
5          TUESDAY,
6          WEDNESDAY,
7          THURSDAY,
8          FRIDAY,
9          SATURDAY,
10         SUNDAY
11         }
```

열거 타입 변수

❖ 열거 타입 변수 선언

```
열거타입 변수;
```

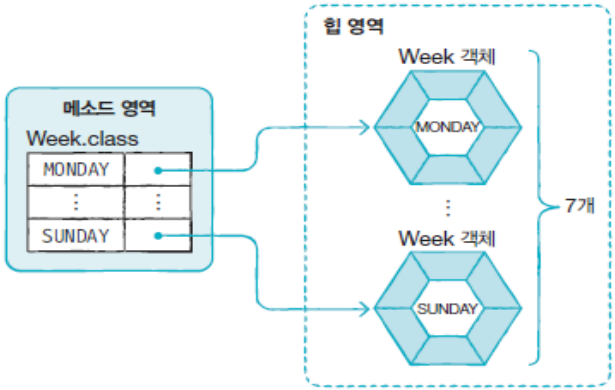
```
Week today;
Week reservationDay;
```

Week.java

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

열거 타입 이름

열거 상수



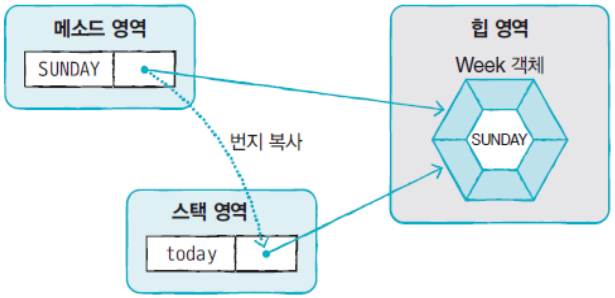
❖ 열거 상수 저장

```
열거타입 변수 = 열거타입.열거상수;
```

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

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

```
Week birthday = null;
```



열거타입 이용 예제

```
1  package sec03.exam02;
2
3  import java.util.Calendar;
6
7  public class EnumWeekExample {
8      public static void main(String[] args) {
9          Week today = null;
10
11         Calendar cal = Calendar.getInstance();
12         int week = cal.get(Calendar.DAY_OF_WEEK);
13
14         switch(week) {
15             case 1:
16                 today = Week.SUNDAY; break;
17             case 2:
18                 today = Week.MONDAY; break;
19             case 3:
20                 today = Week.TUESDAY; break;
21             case 4:
22                 today = Week.WEDNESDAY; break;
23             case 5:
24                 today = Week.THURSDAY; break;
25             case 6:
26                 today = Week.FRIDAY; break;
27             case 7:
28                 today = Week.SATURDAY; break;
29         }
```

열거타입 이용 예제

```
31              System.out.println("오늘 요일: "+ today);
32
33              if(today == Week.SUNDAY) {
34                  System.out.println("일요일에는 TV 시청을 합니다.");
35              } else {
36                  System.out.println("월요일부터 금요일까지는 열심히 자바공부를 합니다.");
37              }
38          }
39      }
```

키워드로 끝내는 핵심 포인트

- 열거 타입: 몇 가지로 한정된 값을 가지는 타입
- 열거 타입 선언 : enum 타입 { 상수, 상수, ... }
- 열거 상수 : 열거 타입 선언 때 주어진 상수
- 열거 타입 변수 : 열거 타입으로 선언된 변수, 열거 상수(한정된 값) 중 하나를 대입.

Thank You !