

2025년 상반기 K-디지털 트레이닝

# 참조 타입

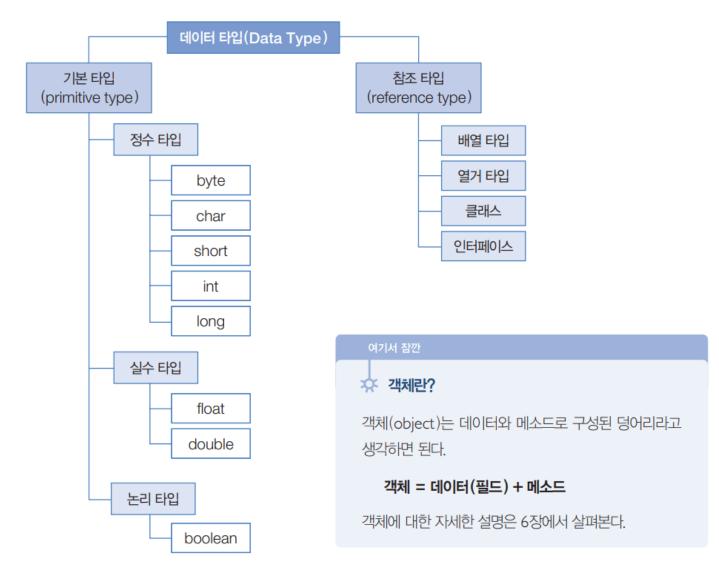
[KB] IT's Your Life



## **1 데이터 타입 분류**

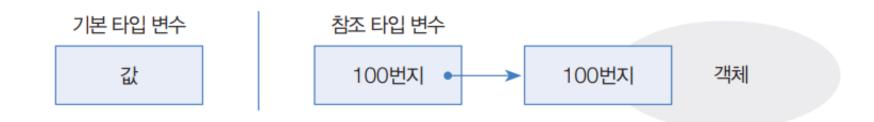
#### 🕜 참조 타입

- 객체의 번지를 참조하는 타입.
- 배열, 열거, 클래스, 인터페이스 타입



#### 기본 타입과 참조 타입

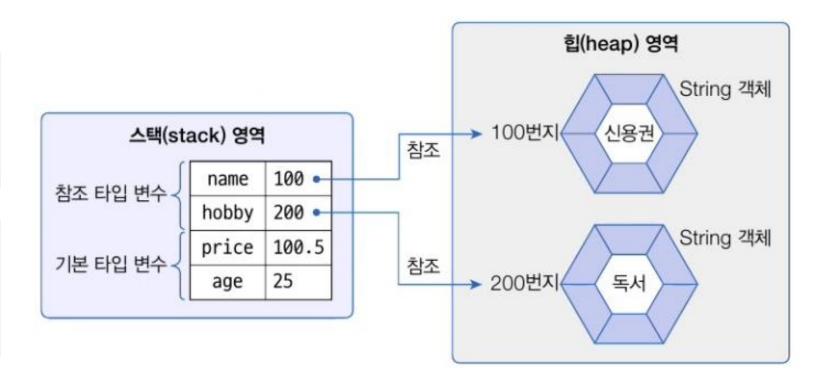
- o 기본 타입으로 선언된 변수
  - 값 자체를 저장
- 참조 타입으로 선언된 변수
  - 객체가 생성된 메모리 번지를 저장



### 기본 타입과 참조 타입

```
[기본 타입 변수]
int age = 25;
double price = 100.5;
```

```
[참조 타입 변수]
String name = "신용권";
String hobby = "독서";
```



#### 메소드, 힙, 스택 영역

○ JVM은 운영체제에서 할당받은 메모리 영역을 메소드 영역, 힙 영역, 스택 영역으로 구분해서 사용

#### ○ 메소드 영역

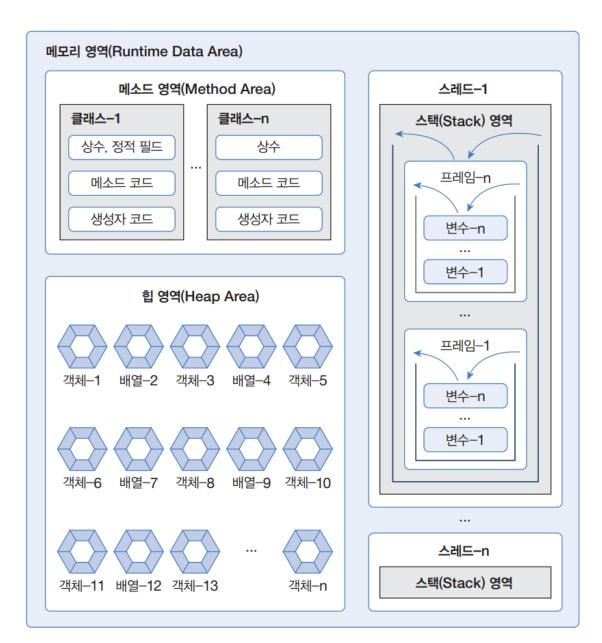
■ 바이트코드 파일을 읽은 내용이 저장되는 영역

#### o 힙 영역

객체가 생성되는 영역. 객체의 번지는 메소드 영역과 스택 영역의 상수와 변수에서 참조

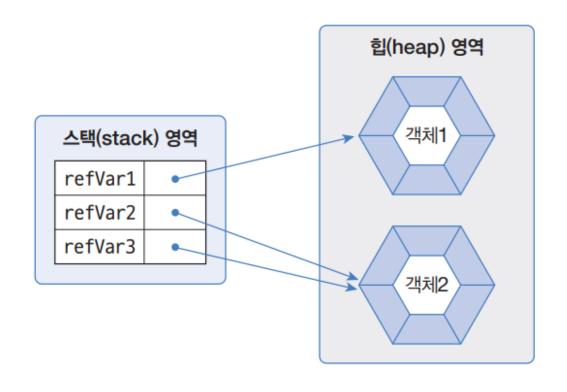
#### ○ 스택 영역

■ 메소드를 호출할 때마다 생성되는 프레임이 저장되는 영역



### ♡ ==, != 연산자

- ==,!= 연산자는 객체의 번지를 비교해 변수의 값이 같은지, 아닌지를 조사
- 번지가 같다면 동일한 객체를 참조하는 것이고, 다르다면 다른 객체를 참조하는 것



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

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

if( refVar2 == refVar3 ) { ... }
```

배열 객체

length: 2

123

배열 객체

length: 2

힙(heap) 영역

10번지

20번지

스택(stack) 영역

## 참조 타입 변수의 ==<u>, != 연산</u>

## ReferenceVariableCompareExample.java

```
arr1
package ch05.sec03;
                                                            arr2
                                                            arr3
public class ReferenceVariableCompareExample {
 public static void main(String[] args) {
   int[] arr1; //배열 변수 arr1 선언
   int[] arr2; //배열 변수 arr2 선언
   int[] arr3; //배열 변수 arr3 선언
   arr1 = new int[] { 1, 2, 3 }; //배열 { 1, 2, 3 }을 생성하고 arr1 변수에 대입
   arr2 = new int[] { 1, 2, 3 }; //배열 { 1, 2, 3 }을 생성하고 arr2 변수에 대입
   arr3 = arr2; //배열 변수 arr2의 값을 배열 변수 arr3에 대입
   System.out.println(arr1 == arr2); // arr1과 arr2 변수가 같은 배열을 참조하는지 검사
   System.out.println(arr2 == arr3); // arr2와 arr3 변수가 같은 배열을 참조하는지 검사
```

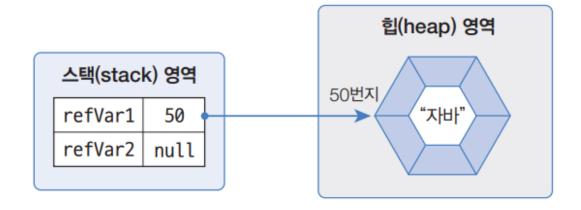
```
false
true
```

## 4 null과 NullPointerException

#### 🧿 null 값

- 참조 타입 변수는 아직 번지를 저장하고 있지 않다는 뜻
- o null도 초기값으로 사용할 수 있기 때문에 null로 초기화된 참조 변수는 스택 영역에 생성

```
String refVar1 = "자바";
String refVar2 = null;
```



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

refVar2 != null //결과: false

## 4 null과 NullPointerException

### NullPointerException

- 변수가 null인 상태에서 객체의 데이터나 메소드를 사용하려 할 때 발생하는 예외
- 참조 변수가 객체를 정확히 참조하도록 번지를 대입해야 해결됨

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

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

## NullPointerExceptionExample.java

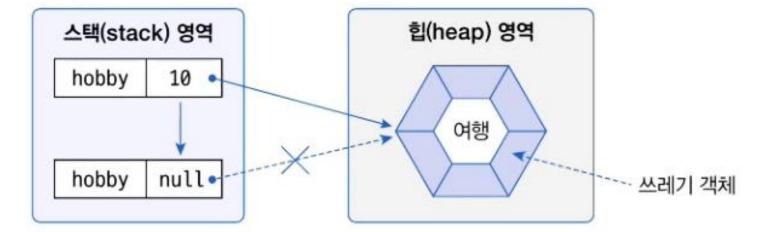
```
package ch05.sec04;
public class NullPointerExceptionExample {
 public static void main(String[] args) {
   int[] intArray = null;
   //intArray[0] = 10; //NullPointerException
   String str = null;
   //System.out.println("총 문자 수: " + str.length() );//NullPointerException
```

## 4 null과 NullPointerException

#### 🗸 null 값의 사용

○ 기존 값을 가진 상태에서 null을 대입하면 기존 값은 사용 불가 상태가 됨(쓰레기, garbage)

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

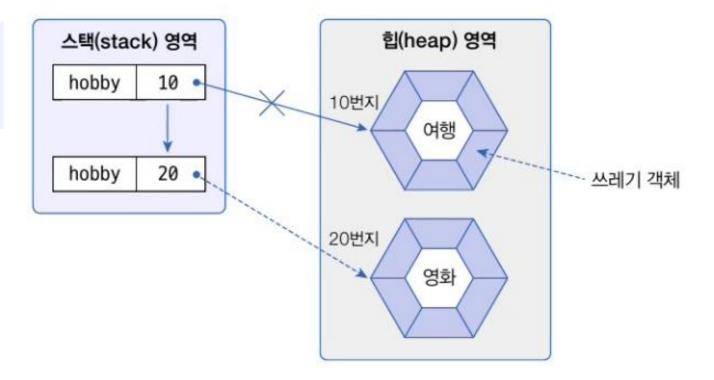


## 4 null과 NullPointerException

### Garbage Collector

- 더 이상 사용하지 않는 메모리 영역(garbage)을 회수 하는 역할
- 자바 가상 머신이 주기적으로 실행
  - 개발자는 메모리 회수에 신경쓸 필요 없음!

```
String hobby = "여행";
data = "영화";
```



## GarbageObjectExample.java

```
public class GarbageObjectExample {
  public static void main(String[] args) {
    String hobby = "여행";
    hobby = null; // "여행"에 해당하는 String 객체를 쓰레기로 만듦

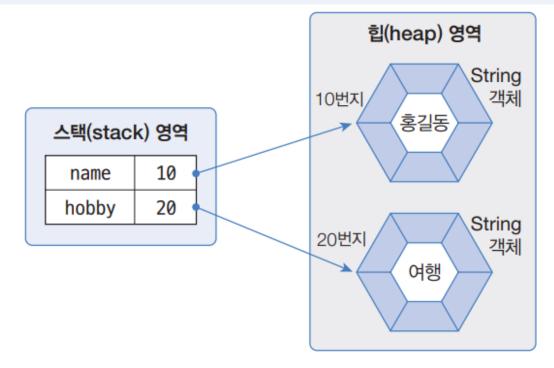
String kind1 = "자동차";
  String kind2 = kind1; // kind1 변수에 저장되어 있는 변지를 kind2 변수에 대입
  kind1 = null; // "자동차"에 해당하는 String 객체는 쓰레기가 아님
    System.out.println("kind2: " + kind2);
  }
}
```

kind2: 자동차

## String 타입

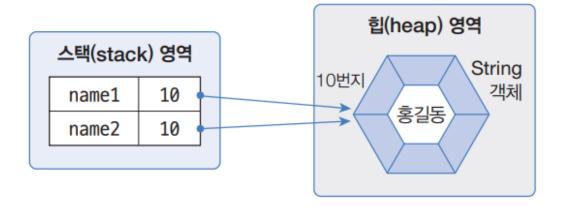
○ 문자열은 String 객체로 생성

```
String name; //String 타입 변수 name 선언
name = "홍길동"; //name 변수에 문자열 대입
String hobby = "여행"; //String 타입 변수 hobby를 선언하고 문자열 대입
```



### 문자열 비교

```
String name1 = "홍길동";
String name2 = "홍길동";
```

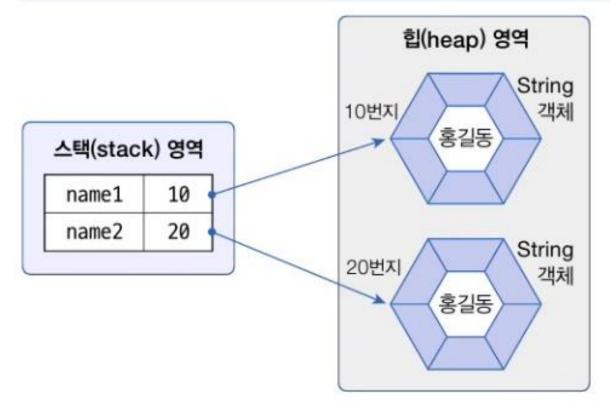


## 5 문자열(String) 타입

#### 🗸 문자열 비교

o new 연산자(객체 생성 연산자)로 직접 String 객체를 생성/대입 가능

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



#### 🗸 문자열 비교

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

```
name1 == name2 //결과: true
name1 == name3 //결과: false
```

```
boolean result = str1.equals(str2); //문자열이 같은지 검사(대소문자 구분)
원본 문자열 비교 문자열
boolean result != str1.equals(str2); //문자열이 다른지 검사
```

## EqualsExample.java

```
package ch05.sec05;
public class EqualsExample {
   public static void main(String[] args) {
       String strVar1 = "홍길동";
       String strVar2 = "홍길동";
       if(strVar1 == strVar2) {
           System.out.println("strVar1과 strVar2는 참조가 같음");
       } else {
           System.out.println("strVar1과 strVar2는 참조가 다름");
       if(strVar1.equals(strVar2)) {
           System.out.println("strVar1과 strVar2는 문자열이 같음");
```

## EqualsExample.java

```
String strVar3 = new String("홍길동");
String strVar4 = new String("홍길동");
if(strVar3 == strVar4) {
   System.out.println("strVar3과 strVar4는 참조가 같음");
} else {
   System.out.println("strVar3과 strVar4는 참조가 다름");
if(strVar3.equals(strVar4)) {
   System.out.println("strVar3과 strVar4는 문자열이 같음");
```

```
strVar1과 strVar2는 참조가 같음
strVar1과 strVar2는 문자열이 같음
strVar3과 strVar4는 참조가 다름
strVar3과 strVar4는 문자열이 같음
```

## 문자열(String) 타입

## EmptyStringExample.java

```
public class EmptyStringExample {
  public static void main(String[] args) {
    String hobby = "";

    if(hobby.equals("")) {
       System.out.println("hobby 변수가 참조하는 String 객체는 빈 문자열");
    }
  }
}
```

hobby 변수가 참조하는 String 객체는 빈 문자열

#### 🗸 문자열 추출

o charAt() 메소드로 문자열에서 매개값으로 주어진 인덱스의 문자를 리턴해 특정 위치의 문자를 얻을 수 있음

```
String subject = "자바 프로그래밍";
char charValue = subject.charAt(3);
```



# ch05.sec05.CharAtExample.java

```
package ch05.sec05;
public class CharAtExample {
 public static void main(String[] args) {
   String ssn = "9506241230123";
   char sex = ssn.charAt(6);
   switch (sex) {
     case '1':
     case '3':
      System.out.println("남자입니다.");
      break;
     case '2':
     case '4':
      System.out.println("여자입니다.");
       break;
```

남자입니다.

### ☑ 문자열 길이

○ 문자열에서 문자의 개수를 얻고 싶다면 length() 메소드를 사용

```
String subject = "자바 프로그래밍";
int length = subject.length();
```



# LengthExample.java

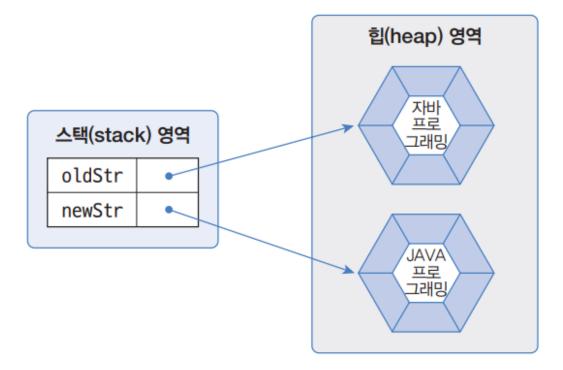
```
public class LengthExample {
  public static void main(String[] args) {
    String ssn = "9506241230123";
    int length = ssn.length();
    if(length == 13) {
        System.out.println("주민등록번호 자릿수가 맞습니다.");
    } else {
        System.out.println("주민등록번호 자릿수가 틀립니다.");
    }
}
```

주민등록번호 자릿수가 맞습니다.

#### 🗸 문자열 대체

o replace() 메소드는 기존 문자열은 그대로 두고, 대체한 새로운 문자열을 리턴

```
String oldStr = "자바 프로그래밍";
String newStr = oldStr.replace("자바", "JAVA");
```



## 문자열(String) 타입

## ReplaceExample.java

```
public class ReplaceExample {
  public static void main(String[] args) {
    String oldStr = "자바 문자열은 불변입니다. 자바 문자열은 String입니다.";
    String newStr = oldStr.replace("자바", "JAVA");

    System.out.println(oldStr);
    System.out.println(newStr);
  }
}
```

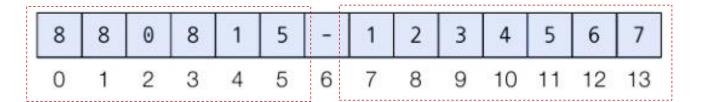
자바 문자열은 불변입니다. 자바 문자열은 String입니다. JAVA 문자열은 불변입니다. JAVA 문자열은 String입니다.

#### 💟 문자열 잘라내기

○ 문자열에서 특정 위치의 문자열을 잘라내어 가져오고 싶다면 substring() 메소드를 사용

메소드	설명
substring(int beginIndex)	beginIndex에서 끝까지 잘라내기
substring(int beginIndex, int endIndex)	beginIndex에서 endIndex 앞까지 잘라내기

```
String ssn = "880815-1234567";
String firstNum = ssn.substring(0, 6);
String secondNum = ssn.substring(7);
```



## 문자열(String) 타입

## SubStringExample.java

```
package ch05.sec05;

public class SubStringExample {
  public static void main(String[] args) {
    String ssn = "880815-1234567";

    String firstNum = ssn.substring(0, 6);
    System.out.println(firstNum);

    String secondNum = ssn.substring(7);
    System.out.println(secondNum);
  }
}
```

```
880815
1234567
```

#### 💟 문자열 찾기

○ 문자열에서 특정 문자열의 위치를 찾고자 할 때에는 indexOf() 메소드를 사용

```
String subject = "자바 프로그래밍";
int index = subject.indexOf("프로그래밍");
```

○ 포함되어 있지 않으면 -1 리턴

```
int index = subject.indexOf("프로그래밍");
if(index == -1) {
    //포함되어 있지 않은 경우
} else {
    //포함되어 있는 경우
}
```

○ 포함 여부 결과만 알고 싶은 경우 constains() 메소드 사용

```
boolean result = subject.contains("프로그래밍");
```



## IndexOfContainsExample.java

```
package ch05.sec05;
public class IndexOfContainsExample {
 public static void main(String[] args) {
   String subject = "자바 프로그래밍";
   int location = subject.indexOf("프로그래밍");
   System.out.println(location);
   String substring = subject.substring(location);
   System.out.println(substring);
   location = subject.index0f("자바");
   if(location != -1) {
    System.out.println("자바와 관련된 책이군요.");
   } else {
    System.out.println("자바와 관련 없는 책이군요.");
```

## IndexOfContainsExample.java

```
boolean result = subject.contains("자바");
if(result) {
    System.out.println("자바와 관련된 책이군요.");
} else {
    System.out.println("자바와 관련 없는 책이군요.");
}
}
}
```

```
3
프로그래밍
자바와 관련된 책이군요.
자바와 관련된 책이군요.
```

### ♡ 문자열 분리

○ 구분자가 있는 여러 개의 문자열을 분리할 때 split() 메소드를 사용

```
String board = "번호,제목,내용,글쓴이";
String[] arr = board.split(",");
```

arr[0]	arr[1]	arr[2]	arr[3]
"번호"	"제목"	"내용"	"성명"

# SplitExample.java

```
package ch05.sec05;
public class SplitExample {
 public static void main(String[] args) {
   String board = "1,자바 학습,참조 타입 String을 학습합니다.,홍길동";
   //문자열 분리
   String[] tokens = board.split(",");
   //인덱스별로 읽기
   System.out.println("번호: " + tokens[0]);
                                             번호: 1
   System.out.println("제목: " + tokens[1]);
                                             제목: 자바 학습
   System.out.println("내용: " + tokens[2]);
                                             내용: 참조 타입 String을 학습합니다.
   System.out.println("성명: " + tokens[3]);
                                             성명: 홍길동
   System.out.println();
   //for 문을 이용한 읽기
                                             자바 학습
   for(int i=0; i<tokens.length; i++) {</pre>
                                             참조 타입 String을 학습합니다.
    System.out.println(tokens[i]);
                                             홍길동
```

#### 💟 배열의 필요성

- 변수는 하나의 값만 저장
- 데이터 개수 만큼 변수가 필요
  - 너무 많은 변수명이 필요함

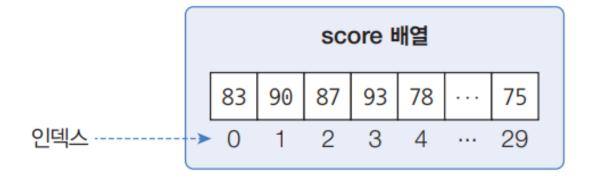
■ 변수가 너무 많으면 연산이 힘들어짐

```
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;
```

#### 🗸 배열

- 연속된 공간에 값을 나열시키고, 각 값에 인덱스를 부여해 놓은 자료구조
- 인덱스는 대괄호 []와 함께 사용하여 각 항목의 값을 읽거나 저장하는데 사용



■ for 문을 이용해 인덱스를 조정하며 접근 → 연산이 간결해짐

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

## 6 배열(Array) 타입

#### 💟 배열의 특징

- o 같은 타입의 값만 관리
- 길이를 늘리거나 주일 수 없음

#### 💟 배열 변수 선언

○ 두 가지 형태로 작성

타입[] 변수;

■ 첫 번째가 관례적인 표기

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

타입 변수[];

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

#### ☑ 배열 변수 선언

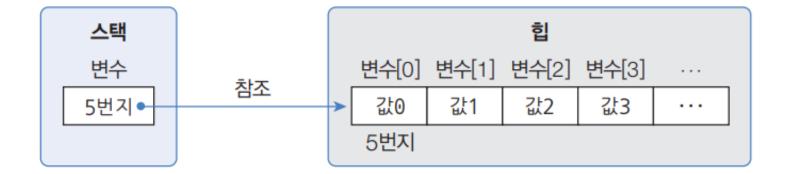
- 배열은 힙 영역에 생성되고 배열 변수는 힙 영역의 배열 주소를 저장
- 참조할 배열이 없다면 배열 변수도 null로 초기화할 수 있다

```
타입[] 변수 = null;
```

#### ☑ 값 목록으로 배열 생성

○ 배열에 저장될 값의 목록이 있다면, 다음과 같이 간단하게 배열을 생성할 수 있음

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



```
String[] season = { "Spring", "Summer", "Fall", "Winter" };
season[1] = "여름";
```

#### ArrayCreateByValueListExample1.java

```
package ch05.sec06;
public class ArrayCreateByValueListExample1 {
 public static void main(String[] args) {
   //배열 변수 선언과 배열 생성
   String[] season = { "Spring", "Summer", "Fall", "Winter" };
   //배열의 항목값 읽기
   System.out.println("season[0] : " + season[0]);
   System.out.println("season[1] : " + season[1]);
   System.out.println("season[2] : " + season[2]);
   System.out.println("season[3] : " + season[3]);
   //인덱스 1번 항목의 값 변경
   season[1] = "여름";
   System.out.println("season[1] : " + season[1]);
   System.out.println();
```

#### .ArrayCreateByValueListExample1.java

```
//배열 변수 선언과 배열 생성
int[] scores = { 83, 90, 87 };
//총합과 평균 구하기
int sum = 0;
for(int i=0; i<3; i++) {
 sum += scores[i];
System.out.println("총합 : " + sum);
double avg = (double) sum / 3;
System.out.println("평균 : " + avg);
```

season[0] : Spring season[1] : Summer season[2] : Fall

season[3]: Winter season[1] : 여름

총합: 260

평균: 86.666666666667

#### ☑ 값 목록으로 배열 생성

- 배열 변수를 선언한 시점과 값 목록이 대입되는 시점이 다르다면 new 타입[]을 중괄호 앞에 붙여줌.
- 타입은 배열 변수를 선언할 때 사용한 타입과 동일하게 지정

```
타입[] 변수;
변수 = { 값0, 값1, 값2, 값3, ... }; //컴파일 에러
변수 = new 타입[] { 값0, 값1, 값2, 값3, ... };
String[] names = null;
names = new String[] { "신용권", "홍길동", "감자바" };
```

#### ♥ 값 목록으로 배열 생성

```
//메소드 선언
void printItem(int[] scores) { … }

//잘못된 메소드 호출
printItem( {95, 85, 90} ); //컴파일 에러
```

```
//올바른 메소드 호출
printItem( new int[] {95, 85, 90} );
```

#### U

## ArrayCreateByValueListExample2.java

```
package ch05.sec06;

public class ArrayCreateByValueListExample2 {

    //printItem() 메소드 선언
    public static void printItem( int[] scores ) {

        //매개변수가 참조하는 배열의 항목을 출력
        for(int i=0; i<3; i++) {

            System.out.println("score[" + i + "]: " + scores[i]);
        }
    }
```

## 6 배열(Array) 타입

## ArrayCreateByValueListExample2.java

```
public static void main(String[] args) {
 //배열 변수 선언
 int[] scores;
 //배열 변수에 배열을 대입
 scores = new int[] { 83, 90, 87 };
 //배열 항목의 총합을 구하고 출력
 int sum1 = 0;
 for(int i=0; i<3; i++) {
  sum1 += scores[i];
 System.out.println("총합 : " + sum1);
 //배열을 매개값으로 주고, printItem() 메소드 호출
 printItem( new int[] { 83, 90, 87 } );
```

```
총합: 260
score[0]: 83
score[1]: 90
score[2]: 87
```

```
//배열을 매개값으로 주고, printItem() 메소드 호출
 printItem( new int[] { 83, 90, 87 } );
//printItem() 메소드 선언
public static void printItem( int[] scores ) {
 //매개변수가 참조하는 배열의 항목을 출력
 for(int i=0; i<3; i++) {
    System.out.println("score[" + i + "]: " + scores[i]);
```

#### new 연산자로 배열 생성

o new 연산자로 값의 목록은 없지만 향후 값들을 저장할 목적으로 배열을 미리 생성

```
타입[] 변수 = new 타입[길이];
타입[] 변수 = null;
변수 = new 타입[길이];
int[] intArray = new int[5];
```

#### ☑ new 연산자로 배열 생성

o new 연산자로 배열을 처음 생성하면 배열 항목은 기본값으로 초기화

데이터 타입		초기값
기본 타입	byte[] char[] short[] int[] long[] float[] double[]	0 '\u0000' 0 0 0L 0.0F 0.0 false
참조 타입	클래스[] 인터페이스[]	null null

## 6 배열(Array) 타입

#### new 연산자로 배열 생성

```
int[] scores = new int[30];
인덱스: 0 1 2 3 4 5 6 7 ... 23 24 25 26 27 28 29
scores 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0
```

## ArrayCreateByNewExample.java

System.out.println("\n");

```
package ch05.sec06;
public class ArrayCreateByNewExample {
 public static void main(String[] args) {
   //배열 변수 선언과 배열 생성
   int[] arr1 = new int[3];
   //배열 항목의 초기값 출력
   for(int i=0; i<3; i++) {
    System.out.print("arr1[" + i + "] : " + arr1[i] + ", ");
   System.out.println();
   //배열 항목의 값 변경
   arr1[0] = 10;
   arr1[1] = 20;
                                                  arr1[0]: 0, arr1[1]: 0, arr1[2]: 0,
   arr1[2] = 30;
                                                  arr1[0]: 10, arr1[1]: 20, arr1[2]: 30,
   //배열 항목의 변경 값 출력
   for(int i=0; i<3; i++) {
    System.out.print("arr1[" + i + "] : " + arr1[i] + ", ");
```

#### ArrayCreateByNewExample.java

```
//배열 변수 선언과 배열 생성
double[] arr2 = new double[3];
//배열 항목의 초기값 출력
for(int i=0; i<3; i++) {
 System.out.print("arr2[" + i + "] : " + arr2[i] + ", ");
System.out.println();
//배열 항목의 값 변경
arr2[0] = 0.1;
arr2[1] = 0.2;
                                               arr2[0]: 0.0, arr2[1]: 0.0, arr2[2]: 0.0,
arr2[2] = 0.3;
                                               arr2[0]: 0.1, arr2[1]: 0.2, arr2[2]: 0.3,
//배열 항목의 변경 값 출력
for(int i=0; i<3; i++) {
 System.out.print("arr2[" + i + "] : " + arr2[i] + ", ");
System.out.println("\n");
```

## ArrayCreateByNewExample.java

```
//배열 변수 선언과 배열 생성
String[] arr3 = new String[3];
//배열 항목의 초기값 출력
for(int i=0; i<3; i++) {
 System.out.print("arr3[" + i + "] : " + arr3[i] + ", ");
System.out.println();
//배열 항목의 값 변경
arr3[0] = "1월";
arr3[1] = "2월";
                                             arr3[0]: null, arr3[1]: null, arr3[2]: null,
                                             arr3[0] : 1월, arr3[1] : 2월, arr3[2] : 3월,
arr3[2] = "3월";
//배열 항목의 변경값 출력
for(int i=0; i<3; i++) {
 System.out.print("arr3[" + i + "] : " + arr3[i] + ", ");
```

#### 🕜 배열 길이

- 배열의 길이란 배열에 저장할 수 있는 항목 수
- 코드에서 배열의 길이를 얻으려면 도트(.) 연산자를 사용해서 참조하는 배열의 length 필드를 읽음

```
배열변수.length;
```

o 배열의 length 필드는 읽기만 가능하므로 값을 변경할 수는 없음

```
intArray.length = 10; //컴파일 에러 발생
```

○ 배열 길이는 for 문을 사용해서 전체 배열 항목을 반복할 때 많이 사용

## ArrayLengthExample.java

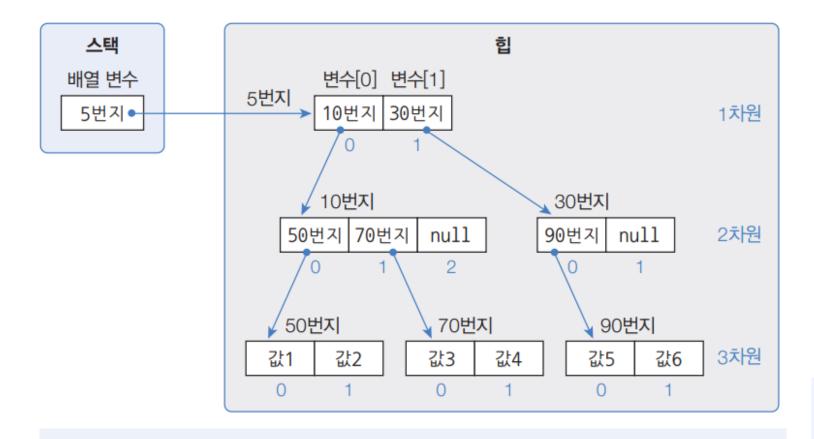
```
package ch05.sec06;
public class ArrayLengthExample {
 public static void main(String[] args) {
   //배열 변수 선언과 배열 대입
   int[] scores = { 84, 90, 96 }; // scores.length => 3
   //배열 항목의 총합 구하기
   int sum = 0;
   for(int i=0; i<scores.length; i++) {</pre>
    sum += scores[i];
   System.out.println("총합 : " + sum);
   //배열 항목의 평균 구하기
   double avg = (double) sum / scores.length;
   System.out.println("평균 : " + avg);
```

총합: 270 평균 : 90.0

#### 7 다 차원 배열

#### 🕜 다차원 배열

○ 배열 항목에는 또 다른 배열이 대입된 배열



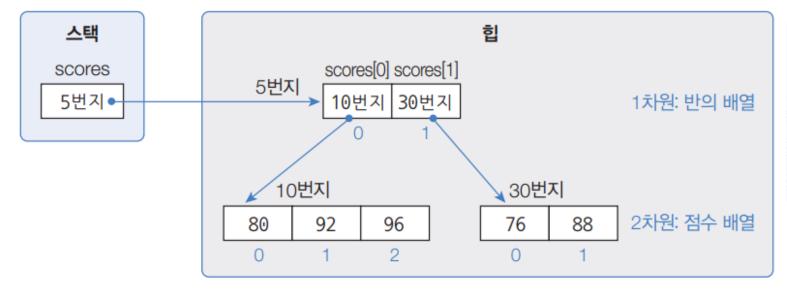
변수[1차원인덱스][2차원인덱스]...[N차원인덱스]

변수[0][0][0] //값1 변수[0][1][0] //값3 변수[1][0][1] //값6

#### ☑ 값 목록으로 다차원 배열 생성

- 배열 변수 선언 시 타입 뒤에 대괄호 []를 차원의 수만큼 붙이고,
- 값 목록도 마찬가지로 차원의 수만큼 중괄호를 중첩

#### 💟 값 목록으로 다차원 배열 생성



```
int score = scores[0][2]; //96
int score = scores[1][1]; //88
scores.length //반의 수: 2
scores[0].length //첫 번째 반의 학생 수: 3
scores[1].length //두 번째 반의 학생 수: 2
```

```
package ch05.sec07;
public class MultidimensionalArrayByValueListExample {
 public static void main(String[] args) {
   //2차원 배열 생성
   int[][] scores = {
      { 80, 90, 96 },
      { 76, 88 }
   };
   //배열의 길이
   System.out.println("1차원 배열 길이(반의 수): " + scores.length);
   System.out.println("2차원 배열 길이(첫 번째 반의 학생 수): " + scores[0].length);
   System.out.println("2차원 배열 길이(두 번째 반의 학생 수): " + scores[1].length);
                                                    1차원 배열 길이(반의 수): 2
   //첫 번째 반의 세 번째 학생의 점수 읽기
                                                    2차원 배열 길이(첫 번째 반의 학생 수): 3
   System.out.println("scores[0][2]: " + scores[0][2]);
                                                     2차원 배열 길이(두 번째 반의 학생 수): 2
   //두 번째 반의 두 번째 학생의 점수 읽기
                                                    scores[0][2]: 96
   System.out.println("scores[1][1]: " + scores[1][1]);
                                                    scores[1][1]: 88
```

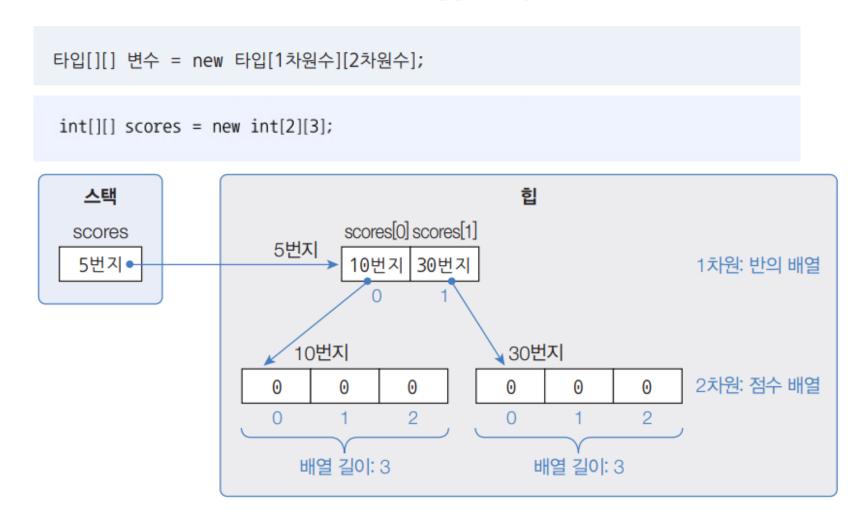
```
//첫 번째 반의 평균 점수 구하기
int class1Sum = 0;
for(int i=0; i<scores[0].length; i++) {</pre>
 class1Sum += scores[0][i];
double class1Avg = (double) class1Sum / scores[0].length;
System.out.println("첫 번째 반의 평균 점수: " + class1Avg);
//두 번째 반의 평균 점수 구하기
int class2Sum = 0;
for(int i=0; i<scores[1].length; i++) {</pre>
 class2Sum += scores[1][i];
double class2Avg = (double) class2Sum / scores[1].length;
System.out.println("두 번째 반의 평균 점수: " + class2Avg);
```

```
//전체 학생의 평균 점수 구하기
int totalStudent = 0;
int totalSum = 0;
for(int i=0; i<scores.length; i++) { //반의 수만큼 반복
 totalStudent += scores[i].length; //반의 학생 수 합산
 for(int k=0; k<scores[i].length; k++) { //해당 반의 학생 수만큼 반복
  totalSum += scores[i][k]; //학생 점수 합산
double totalAvg = (double) totalSum / totalStudent;
System.out.println("전체 학생의 평균 점수: " + totalAvg);
```

전체 학생의 평균 점수: 86.0

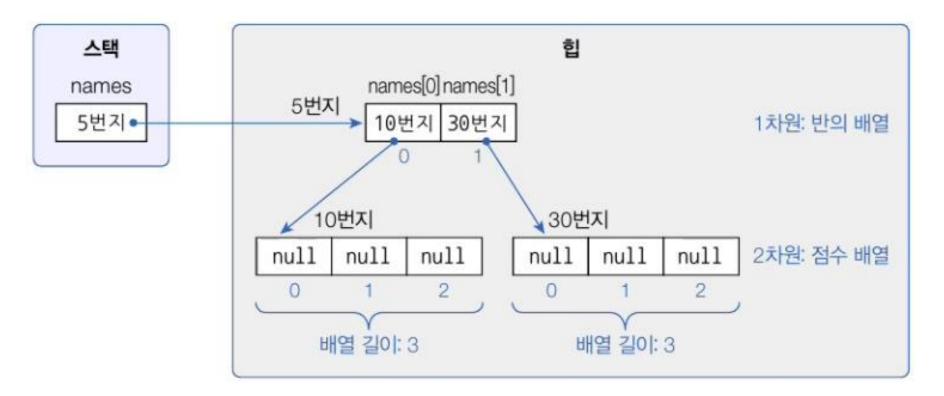
#### new 연산자로 다차원 배열 생성

- 배열 변수 선언 시 타입 뒤에 대괄호 [ ]를 차원의 수만큼 붙이고,
- o new 타입 뒤에도 차원의 수만큼 대괄호 [ ]를 작성



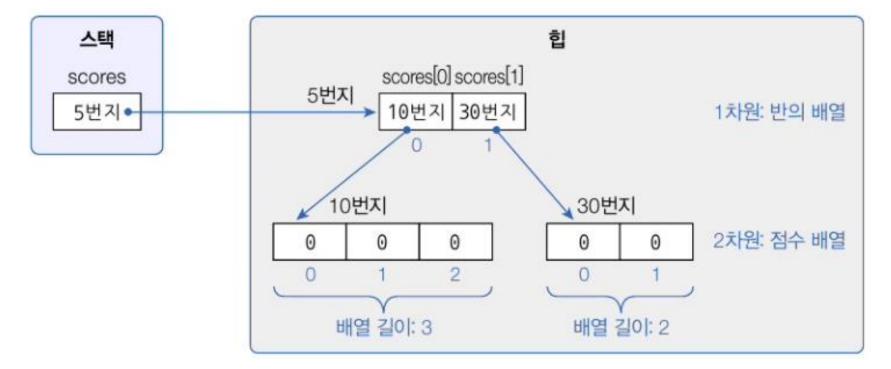
#### new 연산자로 다차원 배열 생성





#### new 연산자로 다차원 배열 생성

```
int[][] scores = new int[2][];
scores[0] = new int[3]; //첫 번째 반의 학생 수가 3명
scores[1] = new int[2]; //두 번째 반의 학생 수가 2명
```



```
mathScores[0][0]: 0
mathScores[0][1]: 0
mathScores[0][2]: 0
mathScores[1][0]: 0
mathScores[1][1]: 0
mathScores[1][2]: 0
```

```
//배열 항목 값 변경
mathScores[0][0] = 80;
mathScores[0][1] = 83;
mathScores[0][2] = 85;
mathScores[1][0] = 86;
mathScores[1][1] = 90;
mathScores[1][2] = 92;
//전체 학생의 수학 평균 구하기
int totalStudent = 0;
int totalMathSum = 0;
for (int i = 0; i < mathScores.length; i++) {
 totalStudent += mathScores[i].length; //반의 학생 수 합산
 for (int k = 0; k < mathScores[i].length; k++) { //해당 반의 학생 수만큼 반복
   totalMathSum += mathScores[i][k]; //학생 점수 합산
double totalMathAvg = (double) totalMathSum / totalStudent;
System.out.println("전체 학생의 수학 평균 점수: " + totalMathAvg);
System.out.println();
                                 전체 학생의 수학 평균 점수: 86.0
```

```
//각 반의 학생 수가 다를 경우 점수 저장을 위한 2차원 배열 생성 int[][] englishScores = new int[2][]; englishScores[0] = new int[2]; englishScores[1] = new int[3]; //배열 항목 초기값 출력 for (int i = 0; i < englishScores.length; i++) { //반의 수만큼 반복 for (int k = 0; k < englishScores[i].length; k++) { // 해당 반의 학생 수만큼 반복 System.out.println("englishScores[" + i + "][" + k + "]: " + englishScores[i][k]); } } System.out.println();
```

```
englishScores[0][0]: 0
englishScores[0][1]: 0
englishScores[1][0]: 0
englishScores[1][1]: 0
englishScores[1][2]: 0
```

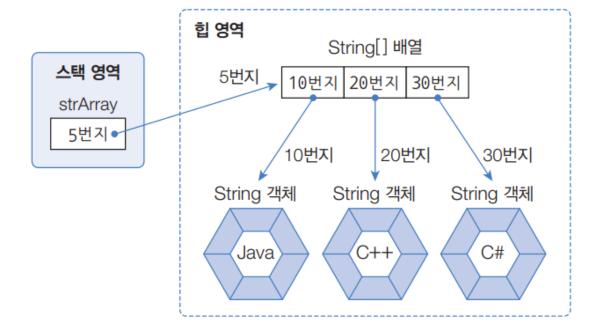
```
//배열 항목 값 변경
englishScores[0][0] = 90;
englishScores[0][1] = 91;
englishScores[1][0] = 92;
englishScores[1][1] = 93;
englishScores[1][2] = 94;
//전체 학생의 영어 평균 구하기
totalStudent = 0;
int totalEnglishSum = 0;
for (int i = 0; i < englishScores.length; i++) { //반의 수만큼 반복
 totalStudent += englishScores[i].length; //반의 학생 수 합산
 for (int k = 0; k < englishScores[i].length; k++) { // 해당 반의 학생 수만큼 반복
   totalEnglishSum += englishScores[i][k]; //학생 점수 합산
double totalEnglishAvg = (double) totalEnglishSum / totalStudent;
System.out.println("전체 학생의 영어 평균 점수: " + totalEnglishAvg);
```

전체 학생의 영어 평균 점수: 92.0

#### ☑ 배열에서 객체 참조하기

- 기본 타입(byte, char, short, int, long, float, double, boolean) 배열은 각 항목에 값을 직접 저장
- 참조 타입(클래스, 인터페이스) 배열은 각 항목에 객체의 번지를 저장

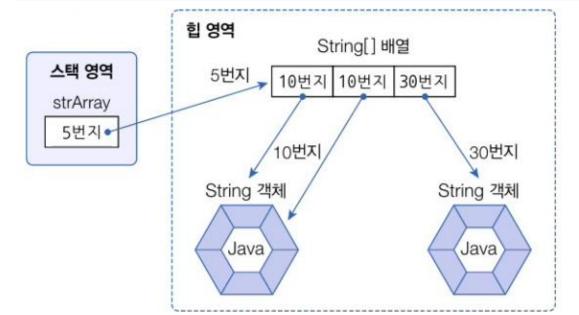
```
String[] strArray = new String[3];
strArray[0] = "Java";
strArray[1] = "C++";
strArray[2] = "C#";
```



# ☑ 배열에서 객체 참조하기

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

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



#### ArrayReferenceObjectExample.java

```
package ch05.sec08;
public class ArrayReferenceObjectExample {
 public static void main(String[] args) {
   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 객체 참조
   System.out.println( strArray[0] == strArray[2] ); //false: 다른 객체를 참조
   System.out.println( strArray[0].equals(strArray[2]) ); //true: 문자열이 동일
```

```
true
false
true
```

#### 💟 배열 복사하기

- 배열은 한 번 생성하면 길이를 변경할 수 없음.
- 더 많은 저장 공간이 필요하다면 더 큰 길이의 배열을 새로 만들고 이전 배열로부터 항목들을 복사



## ArrayCopyByForExample.java

```
package ch05.sec09;
public class ArrayCopyByForExample {
 public static void main(String[] args) {
   //길이 3인 배열
   int[] oldIntArray = { 1, 2, 3 };
   //길이 5인 배열을 새로 생성
   int[] newIntArray = new int[5];
   //배열 항목 복사
   for(int i=0; i<oldIntArray.length; i++) {</pre>
     newIntArray[i] = oldIntArray[i];
   //배열 항목 출력
   for(int i=0; i<newIntArray.length; i++) {</pre>
     System.out.print(newIntArray[i] + ", ");
```

1, 2, 3, 0, 0,

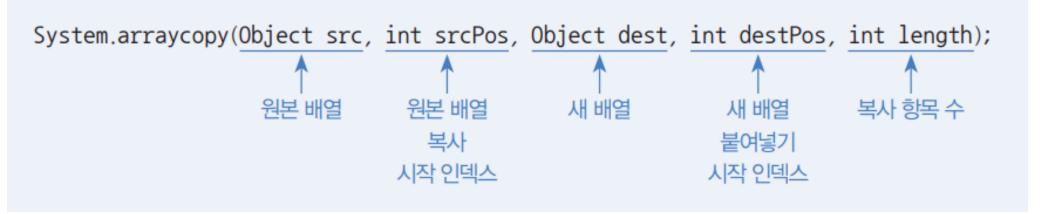
#### 9 배열 복시

#### 🧿 배열 복사하기

○ System의 arraycopy() 메소드를 이용해 배열 복사 가능

해당 함수는 얕은 복사인가?





○ 원본 배열 arr1을 새 배열 arr2에 모두 복사하기

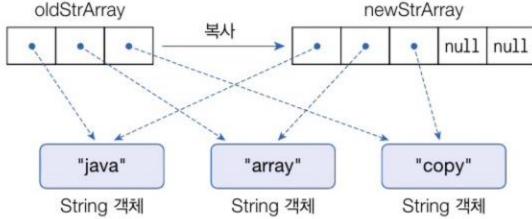
System.arraycopy(arr1, 0, arr2, 0, arr1.length);

객체 복사는?? 깊은 복사는?

## ArrayCopyExample.java

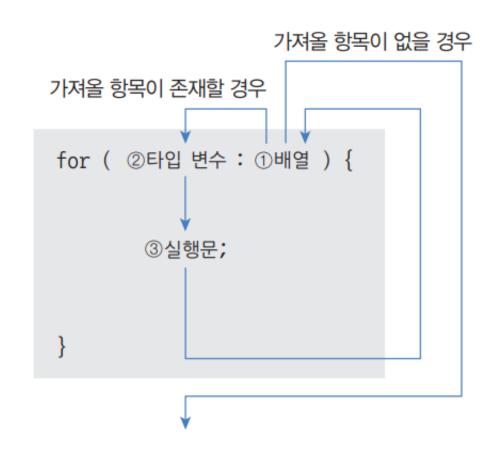
```
package ch05.sec09;
public class ArrayCopyExample {
 public static void main(String[] args) {
   //길이 3인 배열
   String[] oldStrArray = { "java", "array", "copy" };
   //길이 5인 배열을 새로 생성
   String[] newStrArray = new String[5];
   //배열 항목 복사
   System.arraycopy( oldStrArray, 0, newStrArray, 0, oldStrArray.length);
   //배열 항목 출력
   for(int i=0; i<newStrArray.length; i++) {</pre>
                                                             oldStrArray
                                                                                           newStrArray
     System.out.print(newStrArray[i] + ", ");
                                                                            복사
```

java, array, copy, null, null,



#### 10 배열 항목 반복을 위한 향상된 for 문

- 😕 배열 및 컬렉션 처리에 용이한 for 문
  - 카운터 변수와 증감식을 사용하지 않고, 항목의 개수만큼 반복한 후 자동으로 for 문을 빠져나감



for 문이 실행되면

- ①배열에서 가져올 항목이 있을 경우
- ②변수에 항목을 저장,
- ③실행문을 실행

다시 반복해서 ①배열에서 가져올 다음 항목이 존재하면

② → ③ → ①로 진행하고,

가져올 다음 항목이 없으면 for 문을 종료

#### 10 배열 항목 반복을 위한 향상된 for 문

## AdvancedForExample.java

```
package ch05.sec10;
public class AdvancedForExample {
 public static void main(String[] args) {
   //배열 변수 선언과 배열 생성
   int[] scores = { 95, 71, 84, 93, 87 };
   //배열 항목 전체 합 구하기
   int sum = 0;
   for (int score : scores) {
    sum = sum + score;
   System.out.println("점수 총합 = " + sum);
   //배열 항목 전체 평균 구하기
   double avg = (double) sum / scores.length;
   System.out.println("점수 평균 = " + avg);
```

```
점수 총합 = 430
점수 평균 = 86.0
```

#### String[] args 매개변수의 필요성

- 자바 프로그램을 실행하기 위해 main() 메소드를 작성하면서 문자열 배열 형태인 String[] args 매개변수가 필요
- 프로그램 실행 시 입력값이 부족하면 길이가 0인 String 배열 참조

```
java Sum 10 20
```

```
{ "10", "20" };
main() 메소드 호출 시 전달
public static void main(String[] args) { … }
```

```
String x = args[0];
String y = args[1];
int x = Integer.parseInt(args[0]);
int y = Integer.parseInt(args[1]);
```

#### String[] args 매개변수

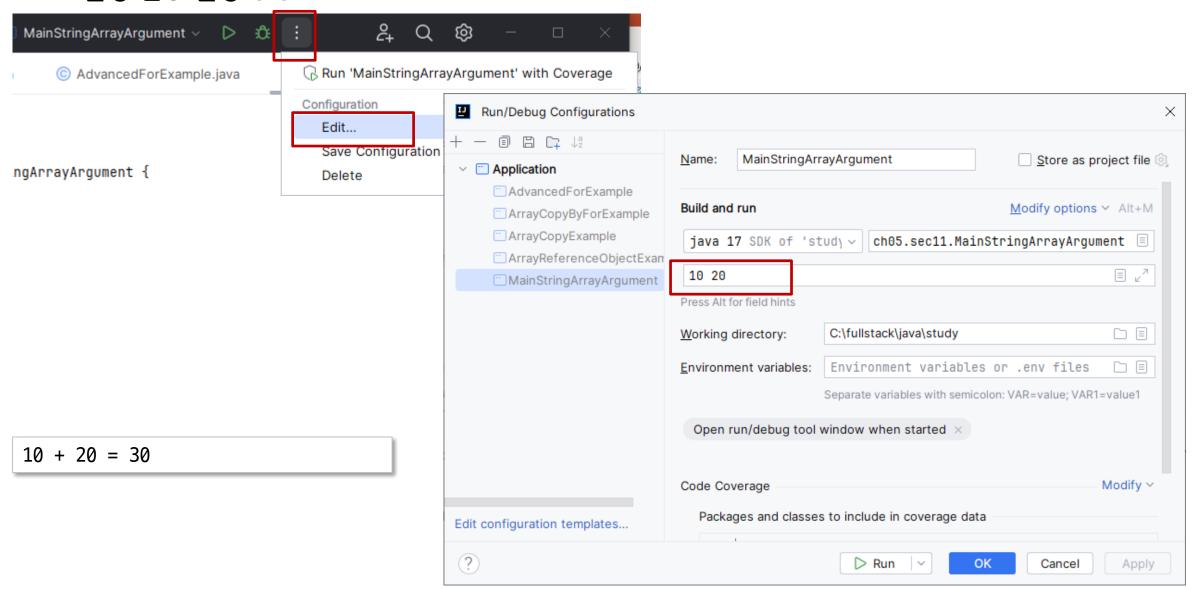
- 매개변수가 지정되지 않은 경우 0java Sum
- o 매개변수의 개수가 지정되어 있는 경우 검사

```
if(args.length != 2) {
System.out.println("실행 시 두 개의 값이 필요합니다.");
}
```

## MainStringArrayArgument.java

```
package ch05.sec11;
public class MainStringArrayArgument {
 public static void main(String[] args) {
   if(args.length != 2) {
     System.out.println("프로그램 입력값이 부족");
     System.exit(0);
   String strNum1 = args[0];
   String strNum2 = args[1];
   int num1 = Integer.parseInt(strNum1);
   int num2 = Integer.parseInt(strNum2);
   int result = num1 + num2;
   System.out.println(num1 + " + " + num2 + " = " + result);
```

#### ☑ 실행 인수 설정하기

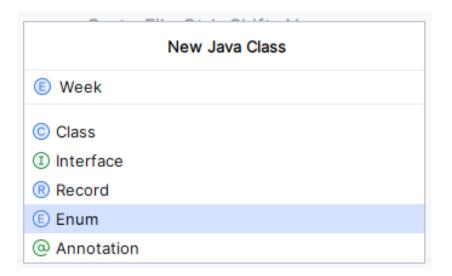


#### 🤝 한정된 값으로 이루어진 Enum 타입

- 요일, 계절처럼 한정된 값을 갖는 타입
- 먼저 열거 타입 이름으로 소스 파일(.java)을 생성하고 한정된 값을 코드로 정의
- 열거 타입 이름은 첫 문자를 대문자로 하고 캐멀 스타일로 지어주는 것이 관례

## ✓ Week.java

```
package ch05.sec12;
                     ➤ 열거 타입 이름
public enum Week {
  MONDAY,
  TUESDAY,
  WEDNESDAY,
                       열거 상수 목록(한정된 값 목록)
  THURSDAY,
  FRIDAY,
  SATURDAY,
  SUNDAY
```



## WeekExample.java

```
package ch05.sec12;
import java.util.Calendar;
public class WeekExample {
 public static void main(String[] args) {
   //Week 열거 타입 변수 선언
   Week today = null;
   //Calendar 얻기
   Calendar cal = Calendar.getInstance();
   //오늘의 요일 얻기(1~7)
   int week = cal.get(Calendar.DAY_OF_WEEK);
```

## WeekExample.java

```
//숫자를 열거 상수로 변환해서 변수에 대입
switch(week) {
 case 1: today = Week.SUNDAY ; break;
 case 2: today = Week.MONDAY ; break;
 case 3: today = Week.TUESDAY ; break;
 case 4: today = Week.WEDNESDAY ; break;
 case 5: today = Week.THURSDAY ; break;
 case 6: today = Week.FRIDAY ; break;
 case 7: today = Week.SATURDAY ; break;
//열거 타입 변수를 사용
if(today == Week.SUNDAY) {
 System.out.println("일요일에는 축구를 합니다.");
} else {
 System.out.println("열심히 자바를 공부합니다.");
```

열심히 자바를 공부합니다.