

# 객체지향 프로그래밍



[교재5장 보충교재2]  
**참조타입**  
(reference type)




고제욱 교수

2024.09.27



## 이것이 자바다(개정판)

교육 현장에서 가장 많이 쓰이는 JAVA 프로그래밍의 기본서

 한빛미디어  집필서  판매중



- 저자 : 신용권, 임경균
- 출간 : 2022-09-05
- 페이지 : 1008 쪽
- ISBN : 9791169210027
- 물류코드 : 11002
- 구판정보 : 이 도서는 <이것이 자바다>의 개정판입니다.

구판 정보 보기

- 저자 : 신용권
- 최초출간(1쇄 발행) : 2022-09-05
- 페이지 : 1008 쪽
- ISBN : 9791169210027
- 물류코드 : 11002

<https://product.kyobobook.co.kr/detail/S000061695652>

[https://www.hanbit.co.kr/store/books/look.php?p\\_code=B4861113361](https://www.hanbit.co.kr/store/books/look.php?p_code=B4861113361)

# 객체지향 프로그래밍



이미지 출처 : <https://www.brickowl.com/catalog/lego-computer-programmer-minifigure>

교재5장 보충교재

문자열 처리 메소드

## String 클래스 (java.lang.String)

java.lang 패키지에 속한 String 클래스임

## ❖ String 클래스 : 문자열을 저장하고 조작할 때 사용



//배열 전체를 String 객체 생성

```
String str = new String(byte[] bytes);
```

//지정한 문자셋으로 디코딩

```
String str = new String(byte[] bytes, String charsetName);
```

//배열의 offset 인덱스 위치부터 length 개 만큼 String 객체 생성

```
String str = new String(byte[] bytes, int offset, int length);
```

//지정한 문자셋으로 디코딩

```
String str = new String(byte[] bytes, int offset, int length, String charsetName)
```

파일의 내용을 읽거나,  
네트워크를 통해 받은 데이터는  
보통 byte[] 배열이므로  
이것을 문자열로 변환하기 위해 사용

### << 문자열 생성 방법 >>

- 문자열 리터럴은 자동으로 String 객체로 생성됨
- String 클래스의 다양한 생성자를 이용해서 직접 객체를 생성 가능

byte[] 배열을 매개변수로 받아서 문자열(string)로 변환하는 String 클래스의 생성자 활용 방법(왼쪽)

- 한글 1자를 UTF-8로 인코딩하면 3바이트가 되고,
- EUC-KR로 인코딩하면 2바이트가 됨

## 키보드로 부터 읽은 바이트 배열을 문자열(string)로 변환

```
byte[] bytes = new byte[100];
```

```
int readByteNo = System.in.read(bytes);
```

```
String str = new String(bytes, 0, readByteNo-2);
```

입력내용:

바이트 배열 내용 :

H	e	l	l	o	W	r	W	n
---	---	---	---	---	---	---	---	---

72	101	108	108	111	13	10
----	-----	-----	-----	-----	----	----

실제 입력된 내용

엔터키

```

1 package ch12.sec05;
2
3 import java.util.Arrays;
4
5 public class BytesToStringExample {
6     public static void main(String[] args) throws Exception {
7         String data = "자바";
8
9         //String -> byte 배열(기본: UTF-8 인코딩)
10        byte[] arr1 = data.getBytes();
11        //byte[] arr1 = data.getBytes("UTF-8");
12        System.out.println("arr1: " + Arrays.toString(arr1));
13
14        //byte 배열 -> String(기본: UTF-8 디코딩)
15        String str1 = new String(arr1);
16        //String str1 = new String(arr1, "UTF-8");
17        System.out.println("str1: " + str1);
18
19        //String -> byte 배열(EUC-KR 인코딩)
20        byte[] arr2 = data.getBytes("EUC-KR");
21        System.out.println("arr2: " + Arrays.toString(arr2));
22
23        //byte 배열 -> String(기본: UTF-8 디코딩)
24        String str2 = new String(arr2, "EUC-KR");
25        System.out.println("str2: " + str2);
26    }
27 }

```

getBytes() 메소드

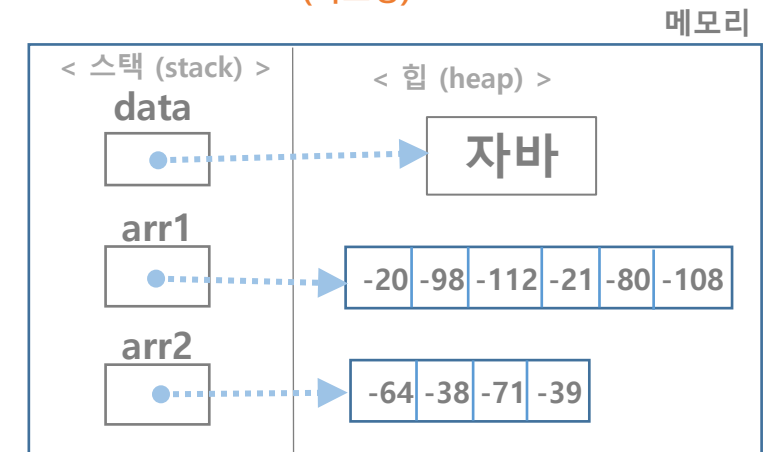
→ 메소드 매개변수로 주어진 문자셋으로 인코딩된 바이트 배열로 변환

인코딩

디코딩을 하려면, String 클래스의 생성자를 이용함  
new String(byte[] bytes, String charsetName)

인코딩

디코딩



< 실행결과 >

```

arr1: [-20, -98, -112, -21, -80, -108]
str1: 자바
arr2: [-64, -38, -71, -39]
str2: 자바

```



```

1 package ch12.sec05;
2
3 import java.util.Arrays;
4
5 public class BytesToStringExample {
6     public static void main(String[] args) throws Exception {
7         String data = "자바";
8
9         //String -> byte 배열(기본: UTF-8 인코딩)
10        //byte[] arr1 = data.getBytes();
11        byte[] arr1 = data.getBytes("UTF-8");
12        System.out.println("arr1: " + Arrays.toString(arr1));
13
14        //byte 배열 -> String(기본: UTF-8 디코딩)
15        //String str1 = new String(arr1);
16        String str1 = new String(arr1, "UTF-8");
17        System.out.println("str1: " + str1);
18
19        //String -> byte 배열(EUC-KR 인코딩)
20        byte[] arr2 = data.getBytes("EUC-KR");
21        System.out.println("arr2: " + Arrays.toString(arr2));
22
23        //byte 배열 -> String(기본: UTF-8 디코딩)
24        String str2 = new String(arr2, "EUC-KR");
25        System.out.println("str2: " + str2);
26    }
27 }

```

**getBytes() 메소드**  
→ 메소드 매개변수로 주어진 문자셋으로 인코딩된 바이트 배열로 변환

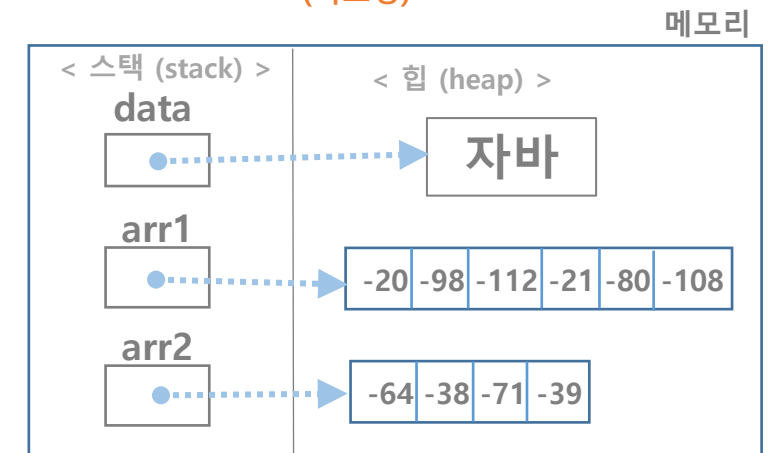
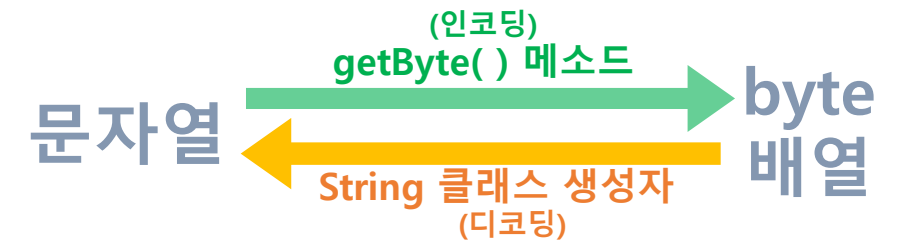
인코딩

디코딩을 하려면, String 클래스의 생성자를 이용함  
`new String(byte[] bytes, String charsetName)`

디코딩

인코딩

디코딩



```

arr1: [-20, -98, -112, -21, -80, -108]
str1: 자바
arr2: [-64, -38, -71, -39]
str2: 자바

```

## [참고] getBytes( ) 메소드, indexOf( ) 메소드

### ■ 바이트 배열로 변환(getBytes())

- 시스템의 기본 문자셋으로 인코딩된 바이트 배열 얻기

```
byte[] bytes = "문자열".getBytes();
```

### ■ 특정 문자셋으로 인코딩 된 바이트 배열 얻기

```
try {  
    byte[] bytes = "문자열".getBytes("EUC-KR");  
    byte[] bytes = "문자열".getBytes("UTF-8");  
} catch (UnsupportedEncodingException e) {  
}
```

### [참고] 디코딩

```
String str = new String(byte[] bytes, String charsetName);
```

### ■ 문자열 찾기(indexOf())

- 매개값으로 주어진 문자열이 시작되는 인덱스 리턴
- 주어진 문자열이 포함되어 있지 않으면 -1 리턴

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

자	바		프	로	그	래	밍
0	1	2	3	4	5	6	7

- 특정 문자열이 포함되어 있는지의 여부에 따라서 실행 코드를 다르게 사용하고자 할 때 사용

### ■ 문자열 길이(length()) – 공백도 문자에 포함

총 8 문자

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

자	바		프	로	그	래	밍
0	1	2	3	4	5	6	7

```
import java.io.UnsupportedEncodingException;
```

```
public class StringGetBytesExample {
    public static void main(String[] args) {
        String str = "안녕하세요";
```

```
        byte[] bytes1 = str.getBytes(); ← 인코딩
        System.out.println("bytes1.length: " + bytes1.length);
        String str1 = new String(bytes1); ← 디코딩
        System.out.println("bytes1->String: " + str1);
        try {
            디코딩을 하려면, String 생성자를 이용함
            new String(byte[] bytes, String charsetName)
```

```
        byte[] bytes2 = str.getBytes("EUC-KR"); ← 인코딩
        System.out.println("bytes2.length: " + bytes2.length);
        String str2 = new String(bytes2, "EUC-KR"); ← 디코딩
        System.out.println("bytes2->String: " + str2);
```

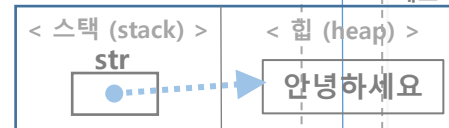
```
        byte[] bytes3 = str.getBytes("UTF-8"); ← 인코딩
        System.out.println("bytes3.length: " + bytes3.length);
        String str3 = new String(bytes3, "UTF-8"); ← 디코딩
        System.out.println("bytes3->String: " + str3);
```

```
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }
}
```

상기 메소드 사용시 발생하는  
예외 클래스임.  
(지원하지 않는 인코딩을 예외 처리함)

한글 1자를 UTF-8로 인코딩  
하면 3바이트가 되고, EUC-KR로  
인코딩하면 2바이트가 됨

getBytes() 메소드  
→ 메소드 매개변수로 주어진  
문자셋으로 인코딩된  
바이트 배열로 변환



```
public class StringIndexOfExample {
    public static void main(String[] args) {
        String subject = "자바 프로그래밍";

        int location = subject.indexOf("프로그래밍");
        System.out.println(location);

        if(subject.indexOf("자바") != -1) {
            System.out.println("자바와 관련된 책이군요");
        } else {
            System.out.println("자바와 관련없는 책이군요");
        }
    }
}
```

< 실행결과 >

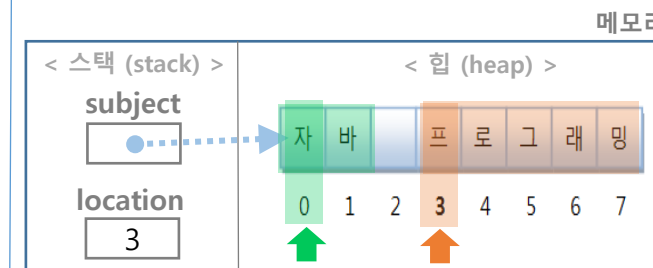
3  
자바와 관련된 책이군요

indexOf( ) 메소드

→ 매개값으로 주어진 문자열이 시작되는 인덱스 리턴  
→ 주어진 문자열이 포함되어 있지 않으면 -1을 리턴함

< 실행결과 >

```
bytes1.length: 10
bytes1->String: 안녕하세요
bytes2.length: 10
bytes2->String: 안녕하세요
bytes3.length: 15
bytes3->String: 안녕하세요
```





```
public class ByteToStringExample {
    public static void main(String[] args) {
        byte[] bytes = { 72, 101, 108, 108, 111, 32, 74, 97, 118, 97 };

        String str1 = new String(bytes);
        System.out.println(str1);

        String str2 = new String(bytes, 6, 4);
        System.out.println(str2);
    }
}
```

bytes, str1, str2 는 로컬변수임

< 실행결과 >

Hello Java  
Java

```
import java.io.IOException;

public class KeyboardToStringExample {
    public static void main(String[] args) throws IOException {
        byte[] bytes = new byte[100];

        System.out.print("입력: ");
        int readByteNo = System.in.read(bytes);

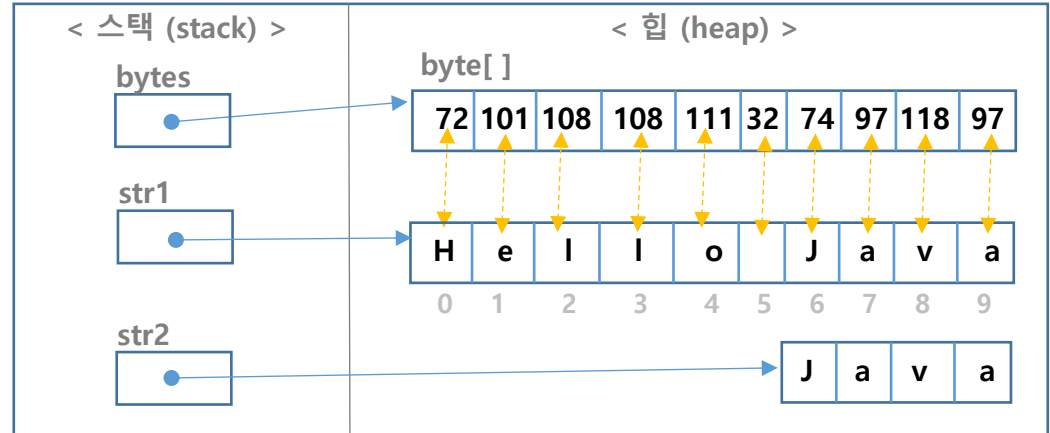
        String str = new String(bytes, 0, readByteNo-2);
        System.out.println(str);
    }
}
```

bytes, readByteNo, str 은  
로컬변수임

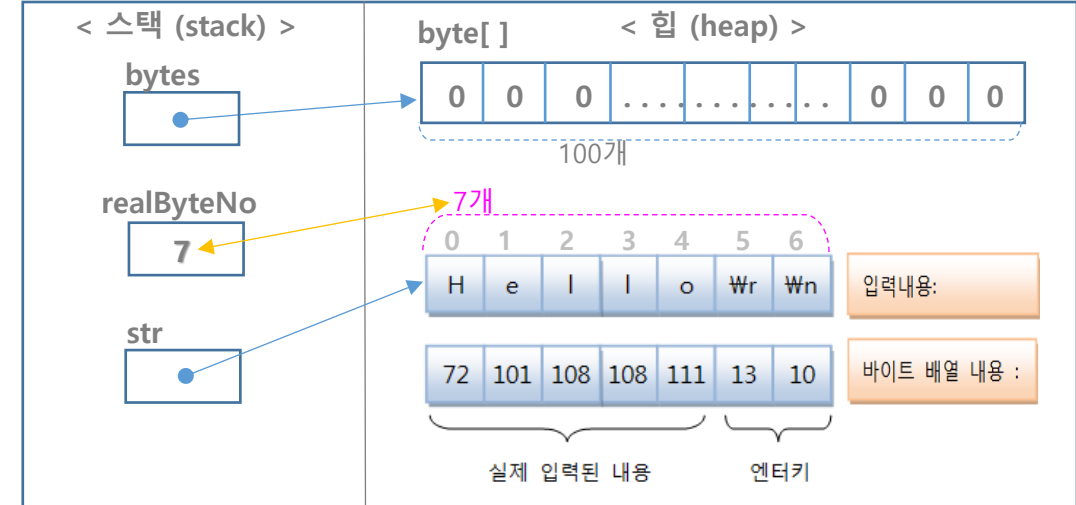
< 실행결과 >

입력: Hello  
Hello

메모리



메모리



# [참고] String 클래스의 메소드

- ❖ String 메소드
  - 문자열의 추출, 비교, 찾기, 분리, 변환 등과 같은 다양한 메소드 가짐
  - 사용 빈도 높은 메소드

리턴타입	메소드명(매개변수)	설명
char	charAt(int index)	특정 위치의 문자 리턴
boolean	equals(Object anObject)	두 문자열을 비교
byte[]	getBytes()	byte[]로 리턴
byte[]	getBytes(Charset charset)	주어진 문자셋으로 인코딩한 byte[]로 리턴
int	indexOf(String str)	문자열내에서 주어진 문자열의 위치를 리턴
int	length()	총 문자의 수를 리턴
String	replace(CharSequence target, CharSequence replacement)	target 부분을 replacement 로 대치한 새로운 문자열을 리턴
String	substring(int beginIndex)	beginIndex 위치에서 끝까지 잘라낸 새로운 문자열을 리턴
String	substring( int beginIndex, int endIndex)	beginIndex 위치에서 endIndex 전까지 잘라낸 새로운 문자열을 리턴
String	toLowerCase()	알파벳 소문자로 변환한 새로운 문자열을 리턴
String	toUpperCase()	알파벳 대문자로 변환한 새로운 문자열을 리턴
String	trim()	앞뒤 공백을 제거한 새로운 문자열을 리턴
String	valueOf(int i) valueOf(double d)	기본 타입값을 문자열로 리턴

- 문자 추출(charAt())
  - 매개값으로 주어진 인덱스의 문자 리턴

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

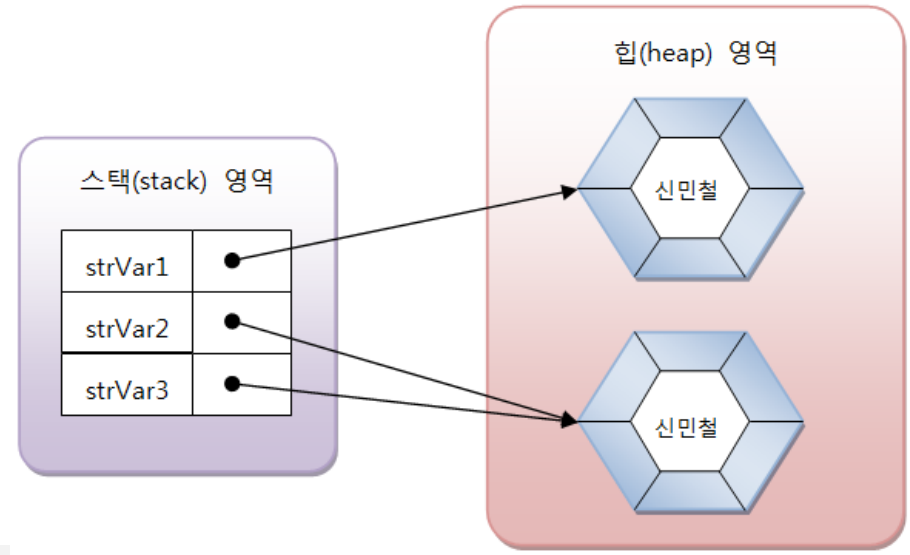


- 문자열 비교(equals())
  - 문자열을 비교할 때 == 연산자를 사용하면 원하지 않는 결과 발생!

```
String strVar1 = new String("신민철");  
String strVar2 = "신민철";  
String strVar3 = "신민철";
```

```
strVar1 == strVar2 → false  
strVar2 == strVar3 → true
```

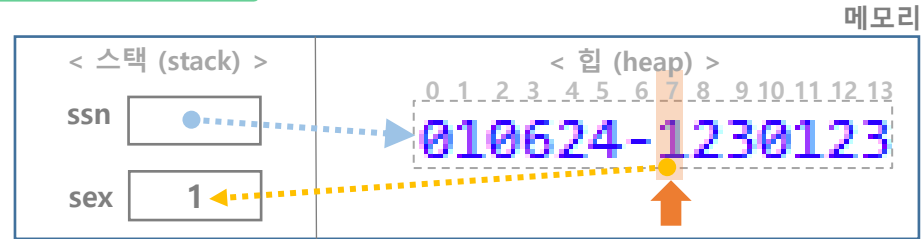
```
strVar1.equals(strVar2) → true  
strVar2.equals(strVar3) → true
```



```
public class StringCharAtExample {
    public static void main(String[] args) {
        String ssn = "010624-1230123";
        char sex = ssn.charAt(7);
        switch (sex) {
            case '1':
            case '3':
                System.out.println("남자 입니다.");
                break;
            case '2':
            case '4':
                System.out.println("여자 입니다.");
                break;
        }
    }
}
```

charAt( ) 메소드  
→ 매개값으로 주어진 인덱스의 문자를 리턴함 (문자추출 메소드)

남자 입니다.

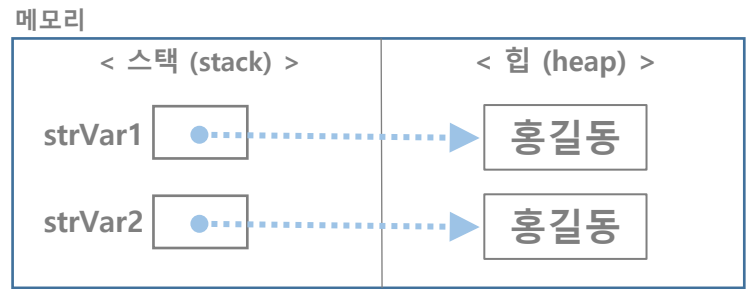


```
public class StringEqualsExample {
    public static void main(String[] args) {
        String strVar1 = new String("신민철");
        String strVar2 = "신민철";
        if(strVar1 == strVar2) {
            System.out.println("같은 String 객체를 참조");
        } else {
            System.out.println("다른 String 객체를 참조");
        }
        if(strVar1.equals(strVar2)) {
            System.out.println("같은 문자열을 가짐");
        } else {
            System.out.println("다른 문자열을 가짐");
        }
    }
}
```

== 연산시 좌변(l-value)과 우변(r-value) 이 문자열 객체인 경우, 각 변수에 저장된 메모리 번지를 비교함

equals( ) 메소드  
→ 두 String 객체의 문자열만을 비교할 때 사용

다른 String 객체를 참조  
같은 문자열을 가짐

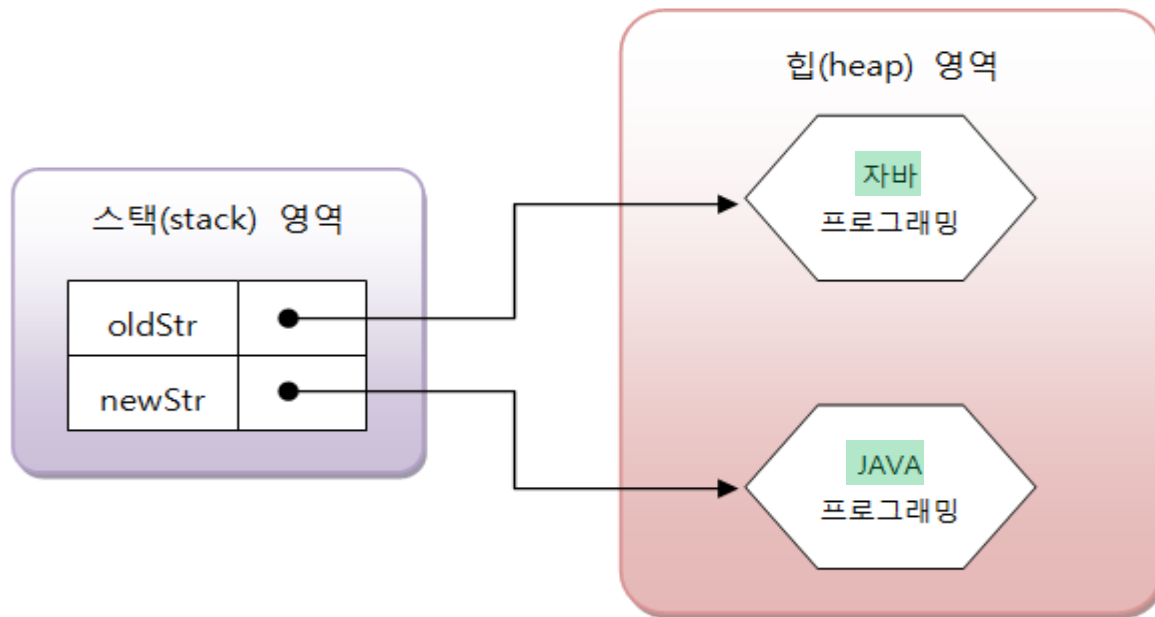


## [참고] String 메소드 – replace( ), substring( )

### ■ 문자열 대치(replace())

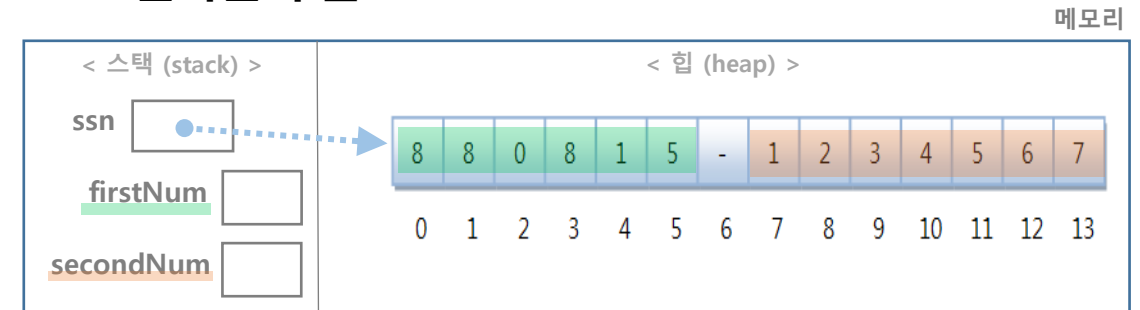
- 첫 번째 매개값인 문자열 찾을
- 두 번째 매개값인 문자열로 대치
- 새로운 문자열 리턴

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



### ■ 문자열 잘라내기(substring())

- substring(int beginIndex, int endIndex)  
→ 주어진 시작(beginIndex) 부터 끝 인덱스의 바로 직전(endIndex-1)까지의 문자열 추출
- substring(int beginIndex)  
→ 주어진 인덱스(beginIndex) 이후 부터 끝까지 문자열 추출

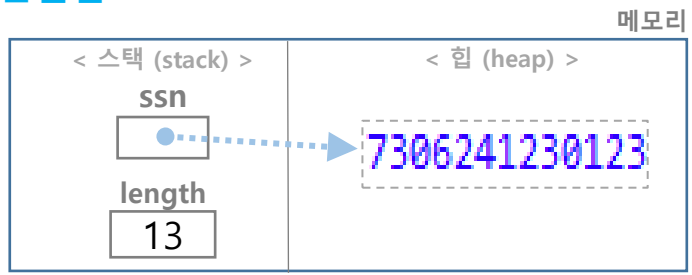


```
String ssn = "880815-1234567";  
String firstNum = ssn.substring(0, 6);  
String secondNum = ssn.substring(7);
```



```
public class StringLengthExample {
    public static void main(String[] args) {
        String ssn = "7306241230123";
        int length = ssn.length();
        if(length == 13) {
            System.out.println("주민번호 자리수가 맞습니다.");
        } else {
            System.out.println("주민번호 자리수가 틀립니다.");
        }
    }
}
```

length( ) 메소드  
→ 문자열의 길이(문자열의 문자의 수)를 반환함  
→ 문자열의 중간에 공백이 있다면, 이 공백도 문자열의 길이에 포함됨.



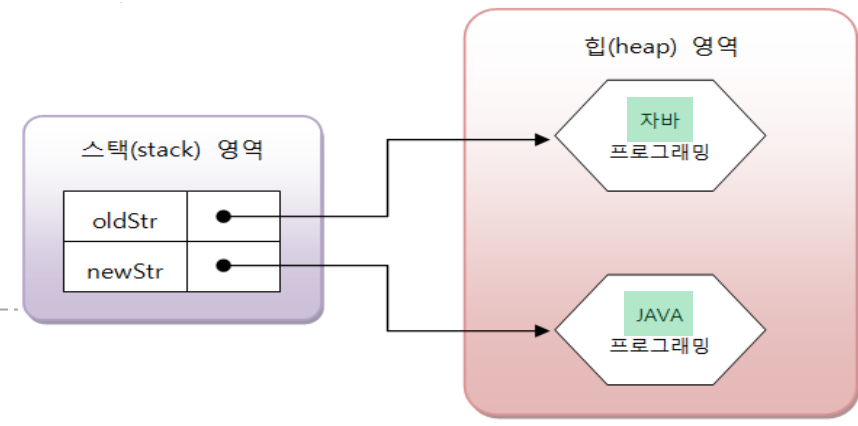
< 실행결과 >  
주민번호 자리수가 맞습니다.

```
public class StringReplaceExample {
    public static void main(String[] args) {
        String oldStr = "자바는 객체지향언어 입니다. 자바는 풍부한 API를 지원합니다.";
        String newStr = oldStr.replace("자바", "JAVA");

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

replace( ) 메소드  
→ 첫번째 매개변수의 문자열을 찾아서, 두번째 매개변수의 문자열로 대치(대체)하는 새로운 문자열을 생성하여 리턴함.

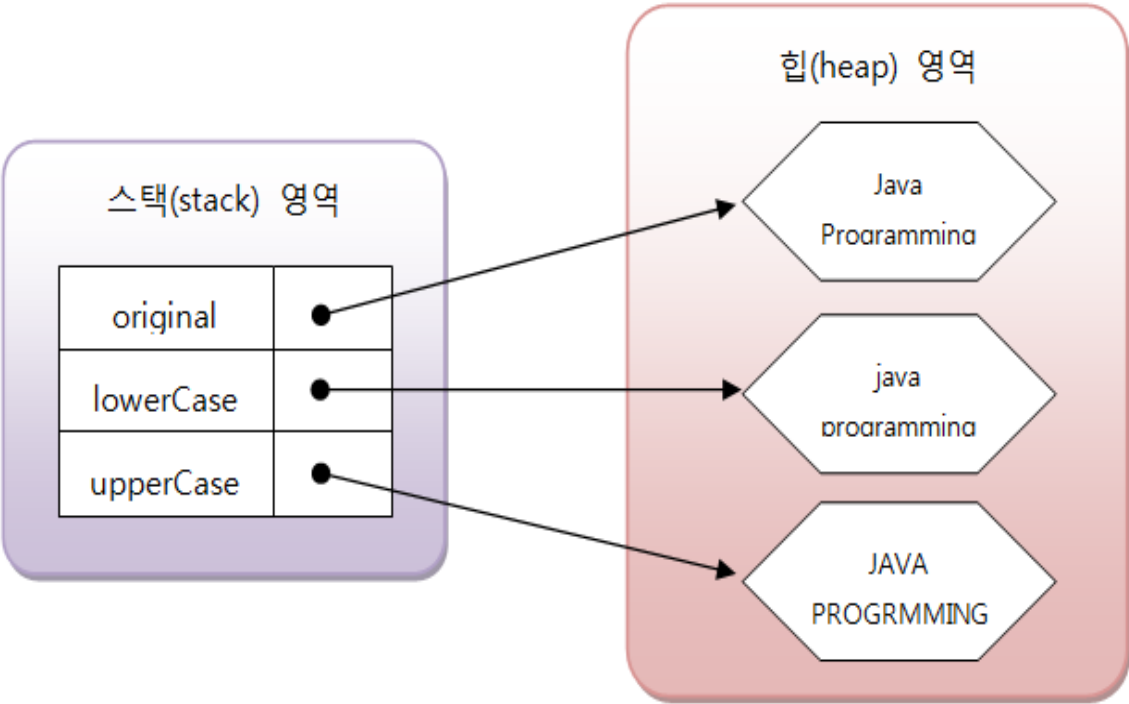
< 실행결과 >  
자바는 객체지향언어 입니다. 자바는 풍부한 API를 지원합니다.  
JAVA는 객체지향언어 입니다. JAVA는 풍부한 API를 지원합니다.



# [참고] toLowerCase( ), toUpperCase( ), trim( ), ValueOf( ) 메소드

- 알파벳 소·대문자 변경  
(toLowerCase(), toUpperCase())

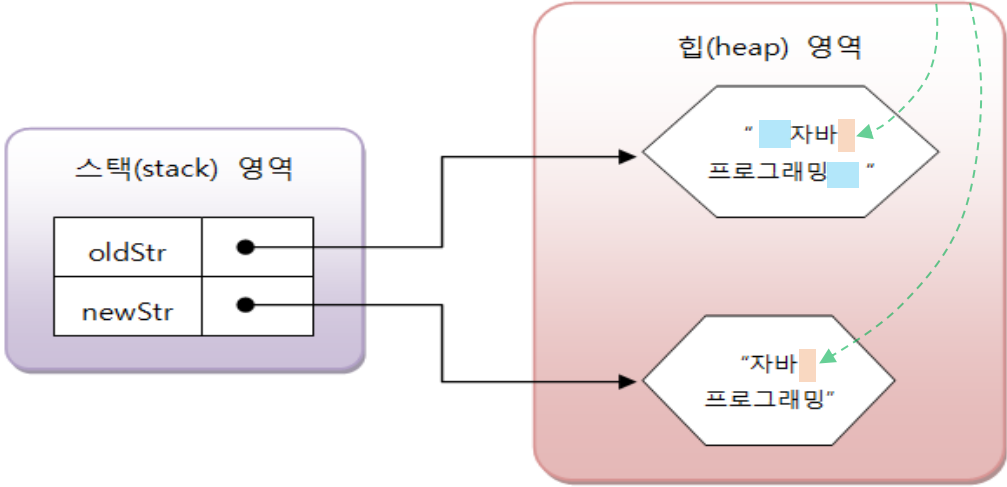
```
String original = "Java Programming";  
String lowerCase = original.toLowerCase();  
String upperCase = original.toUpperCase();
```



- ❖ 문자열 앞뒤 공백 잘라내기(trim())

```
String oldStr = " 자바 프로그래밍 ";  
String newStr = oldStr.trim();
```

trim( ) 메소드는  
문자열의 중간에  
있는 공백을  
없애지는 않음



- 문자열 변환(valueOf())

- valueOf( ) 메소드의  
매개변수값으로 받은  
기본 타입의 값을  
문자열로 변환

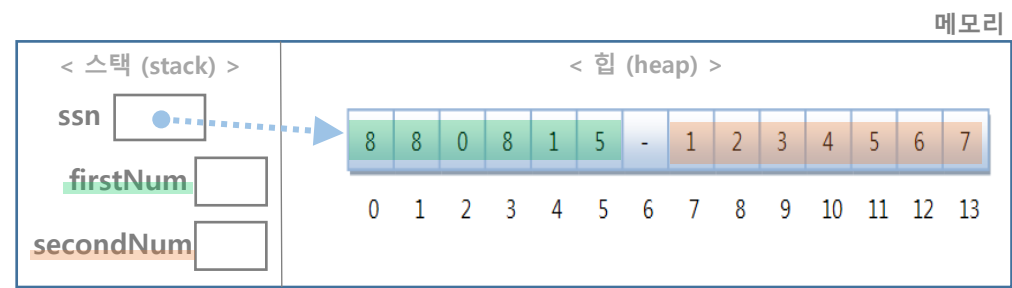
```
static String valueOf(boolean b)  
static String valueOf(char c)  
static String valueOf(int i)  
static String valueOf(long l)  
static String valueOf(double d)  
static String valueOf(float f)
```

```
public class StringSubstringExample {
    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);
    }
}
```

**substring( ) 메소드**  
→ 첫번째 매개변수의 인덱스 부터 시작하여  
두번째 매개변수의 인덱스 바로 직전까지의 문자열을 추출함.



- 문자열 잘라내기(substring())
  - substring(int beginIndex, int endIndex)  
→ 주어진 시작(beginIndex) 부터 끝 인덱스의 바로 직전(endIndex-1)까지의 문자열 추출
  - substring(int beginIndex)  
→ 주어진 인덱스(beginIndex) 이후 부터 끝까지 문자열 추출

< 실행결과 >  
880815  
1234567

```
public class StringToLowerUpperCaseExample {
    public static void main(String[] args) {
        String str1 = "Java Programming";
        String str2 = "JAVA Programming";

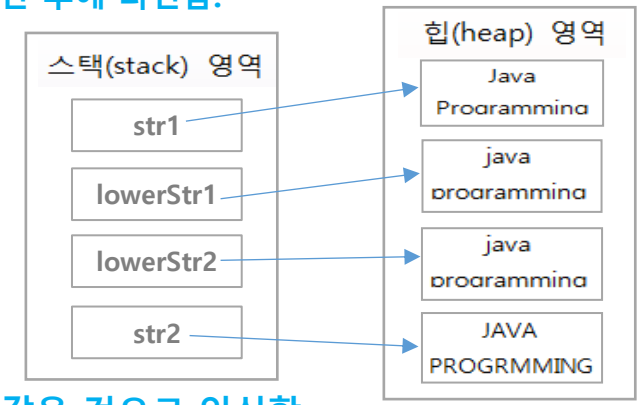
        System.out.println(str1.equals(str2));

        String lowerStr1 = str1.toLowerCase();
        String lowerStr2 = str2.toLowerCase();
        System.out.println(lowerStr1.equals(lowerStr2));

        System.out.println(str1.equalsIgnoreCase(str2));
    }
}
```

**toLowerCase( ) / toUpperCase( ) 메소드**  
→ 매개변수로 주어진 문자열을 모두 소문자(대문자)로 바꾼 새로운 문자열을 생성한 후에 리턴함.

< 실행결과 >  
false  
true  
true



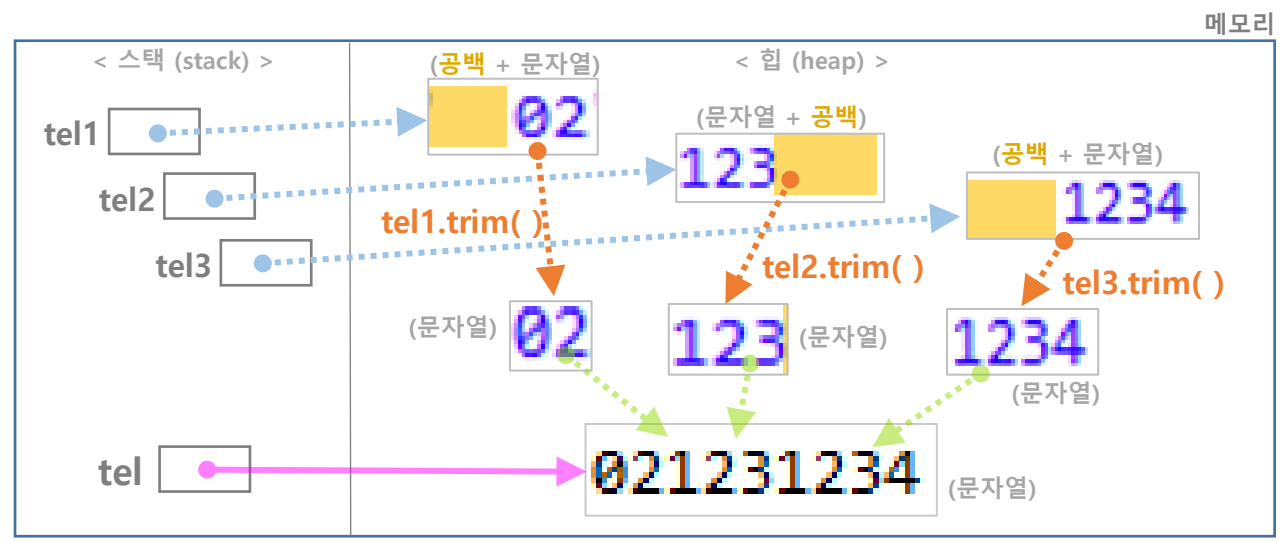
**equalsIgnoreCase( ) 메소드**  
→ 영문 대소문자의 차이를 무시하고 같은 것으로 인식함.  
(예, 영문 대문자 A와 소문자 a 를 (대소문자 차이를 무시하고) 같은 것으로 간주하여 인식함.)

```
public class StringTrimExample {
    public static void main(String[] args) {
        String tel1 = " 02";
        String tel2 = "123 ";
        String tel3 = " 1234 ";

        String tel = tel1.trim() + tel2.trim() + tel3.trim();
        System.out.println(tel);
    }
}
```

**trim( ) 메소드**  
→ trim( ) 메소드의 매개변수로 전달(제공)되는 문자열의 앞뒤 공백을 제거한 새로운 문자열을 생성하여 리턴함  
→ trim( ) 메소드의 매개변수로 전달(제공)되는 문자열의 앞뒤의 공백만 제거할 뿐,中间的 공백은 제거하지 않는다.

< 실행결과 >  
**021231234**

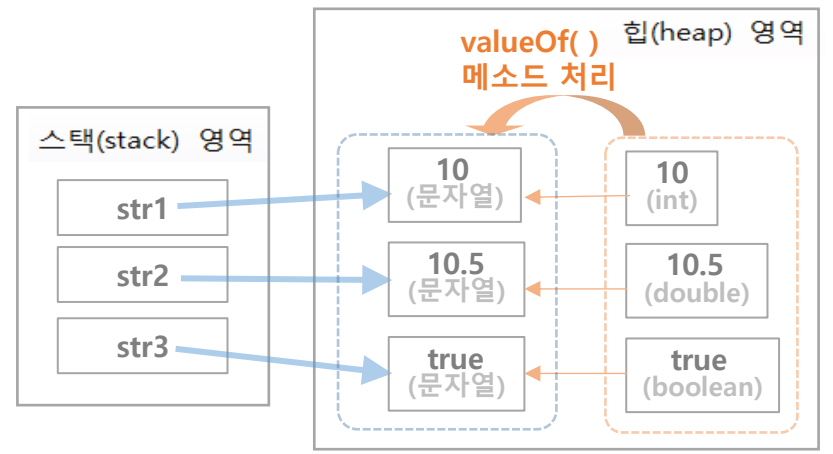


```
public class StringValueOfExample {
    public static void main(String[] args) {
        String str1 = String.valueOf(10);
        String str2 = String.valueOf(10.5);
        String str3 = String.valueOf(true);

        System.out.println(str1);
        System.out.println(str2);
        System.out.println(str3);
    }
}
```

**valueOf(-) 메소드**  
→ valueOf( ) 메소드의 매개변수로 전달(제공)되는 기본타입의 값을 문자열로 변환한다.

< 실행결과 >  
**10**  
**10.5**  
**true**



# 객체지향 프로그래밍



이미지 출처 : <https://www.brickowl.com/catalog/lego-computer-programmer-minifigure>

교재12장 보충교재  
java.base 모듈

## [참고] Arrays 클래스 (java.util.Arrays)

java.util 패키지에 속한 Arrays 클래스임



❖ Arrays

- 배열 조작 기능을 가지고 있는 클래스
  - 배열 복사, 항목 정렬, 항목 검색
- 제공하는 정적 메소드

■ 배열 복사

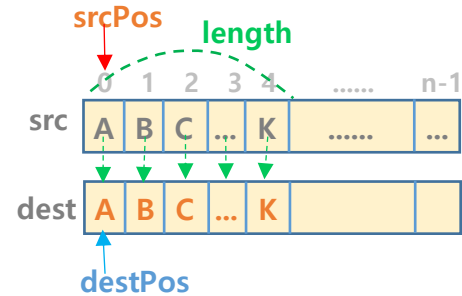
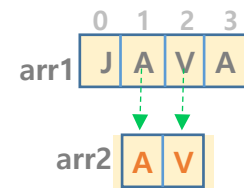
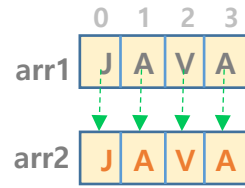
- Arrays.copyOf(원본배열, 복사할 길이)
  - 배열 인덱스 0 ~ (복사할 길이-1)까지 항목 복사
  - 복사할 길이는 원본 배열의 길이보다 커도 되며 타겟 배열의 길이

```
char[] arr1 = {'J', 'A', 'V', 'A'};
char[] arr2 = Arrays.copyOf(arr1, arr1.length);
```

- copyOfRange(원본 배열, 시작 인덱스, 끝 인덱스)
  - 시작 인덱스 ~ (끝 인덱스-1)까지 항목 복사

```
char[] arr1 = {'J', 'A', 'V', 'A'};
char[] arr2 = Arrays.copyOfRange(arr1, 1, 3);
```

리턴타입	메소드 이름	설명
int	binarySearch(배열, 찾는값)	전체 배열 항목에서 찾는값이 있는 인덱스 리턴
타겟배열	copyOf(원본배열, 복사할길이)	원본배열의 0 번 인덱스에서 복사할 길이만큼 복사한 배열 리턴, 복사할 길이는 원본배열의 길이보다 크도 되며, 타겟배열의 길이가 된다.
타겟배열	copyOfRange(원본배열, 시작인덱스, 끝인덱스)	원본배열의 시작인덱스에서 끝인덱스까지 복사한 배열 리턴
boolean	deepEquals(배열, 배열)	두 배열의 깊은 비교(중첩 배열의 항목까지 비교)
boolean	equals(배열, 배열)	얕은 비교(중첩 배열의 항목은 비교하지 않음)
void	fill(배열, 값)	전체 배열 항목에 동일한 값을 저장
void	fill(배열, 시작인덱스, 끝인덱스, 값)	시작인덱스부터 끝인덱스까지의 항목에만 동일한 값을 저장
void	sort(배열)	배열의 전체 항목을 올림차순으로 정렬



• System.arraycopy()

```
System.arraycopy(Object src, int srcPos, Object dest, int destPos, int length)
                    원본배열   원본시작인덱스   타겟배열   타겟시작인덱스   복사개수
```

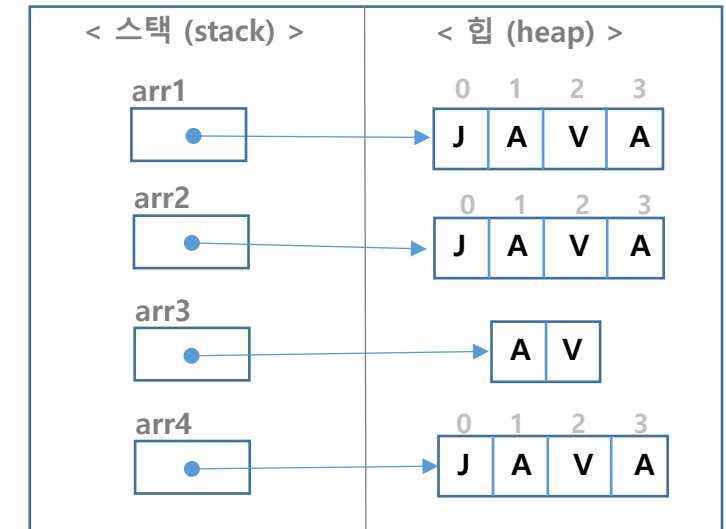
```
import java.util.Arrays;

public class ArrayCopyExample {
    public static void main(String[] args) {
        char[] arr1 = {'J', 'A', 'V', 'A'};

        //방법1
        char[] arr2 = Arrays.copyOf(arr1, arr1.length);
        System.out.println(Arrays.toString(arr2));

        //방법2
        char[] arr3 = Arrays.copyOfRange(arr1, 1, 3);
        System.out.println(Arrays.toString(arr3));

        //방법3
        char[] arr4 = new char[arr1.length];
        System.arraycopy(arr1, 0, arr4, 0, arr1.length);
        for(int i=0; i<arr4.length; i++) {
            System.out.println("arr4[" + i + "]=" + arr4[i]);
        }
    }
}
```



&lt; 실행결과 &gt;

```
[J, A, V, A]
[A, V]
arr4[0]=J
arr4[1]=A
arr4[2]=V
arr4[3]=A
```

배열 항목 비교

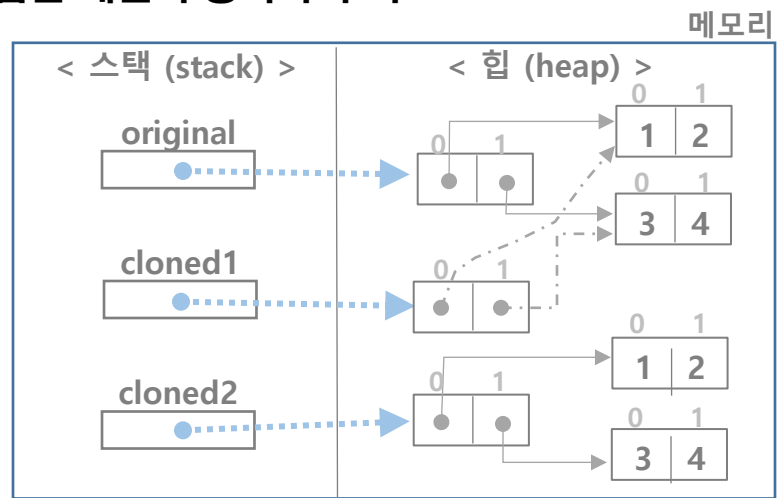
- Arrays.equals(배열, 배열) - 1차 항목의 값만 비교
- Arrays.deepEquals(배열, 배열) - 중첩된 배열의 항목까지 비교

```
import java.util.Arrays;

public class EqualsExample {
    public static void main(String[] args) {
        int[][] original = { {1,2}, {3,4} };

        //얕은 복사후 비교
        System.out.println("[얕은 복제후 비교]");
        int[][] cloned1 = Arrays.copyOf(original, original.length);
        System.out.println("배열 번지 비교: " + original.equals(cloned1));
        System.out.println("1차 배열 항목값 비교: " + Arrays.equals(original, cloned1));
        System.out.println("중첩 배열 항목값 비교: " + Arrays.deepEquals(original, cloned1));

        //깊은 복사후 비교
        System.out.println("\n[깊은 복제후 비교]");
        int[][] cloned2 = Arrays.copyOf(original, original.length);
        cloned2[0] = Arrays.copyOf(original[0], original[0].length);
        cloned2[1] = Arrays.copyOf(original[1], original[1].length);
        System.out.println("배열 번지 비교: " + original.equals(cloned2));
        System.out.println("1차 배열 항목값 비교: " + Arrays.equals(original, cloned2));
        System.out.println("중첩 배열 항목값 비교: " + Arrays.deepEquals(original, cloned2));
    }
}
```



< 실행결과 >

[얕은 복제후 비교]  
배열 번지 비교: false  
1차 배열 항목값 비교: true  
중첩 배열 항목값 비교: true

[깊은 복제후 비교]  
배열 번지 비교: false  
1차 배열 항목값 비교: false  
중첩 배열 항목값 비교: true

```
import java.util.Arrays;

public class SortExample {
    public static void main(String[] args) {
        int[] scores = { 99, 97, 98 };
        Arrays.sort(scores);
        for(int i=0; i<scores.length; i++) {
            System.out.println("scores[" + i + "]= " + scores[i]);
        }
        System.out.println();

        String[] names = { "홍길동", "박동수", "김민수" };
        Arrays.sort(names);
        for(int i=0; i<names.length; i++) {
            System.out.println("names[" + i + "]= " + names[i]);
        }
        System.out.println();

        Member m1 = new Member("홍길동");
        Member m2 = new Member("박동수");
        Member m3 = new Member("김민수");
        Member[] members = { m1, m2, m3 };
        Arrays.sort(members);
        for(int i=0; i<members.length; i++) {
            System.out.println("members[" + i + "].name= " + members[i].name);
        }
    }
}
```

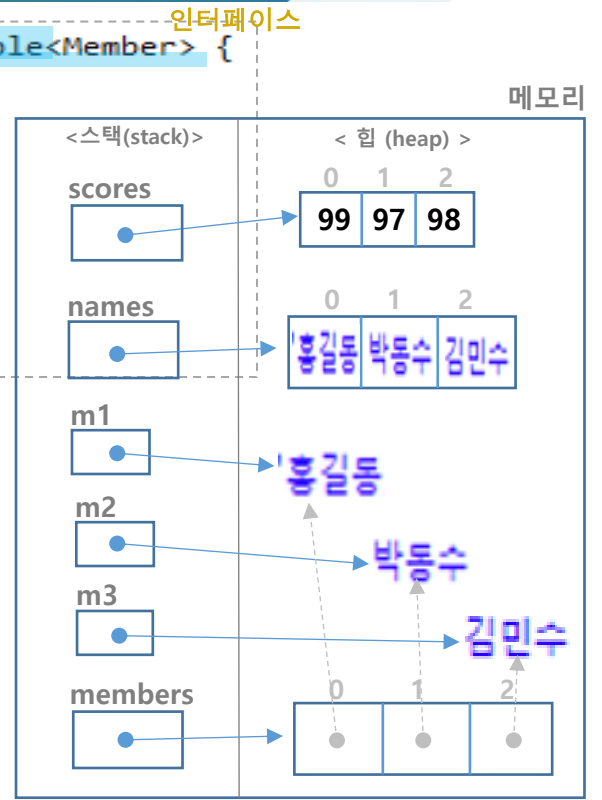
```
public class Member implements Comparable<Member> {
    String name;
    Member(String name) {
        this.name = name;
    }
    @Override
    public int compareTo(Member o) {
        return name.compareTo(o.name);
    }
}
```

< 실행결과 >

```
scores[0]=97
scores[1]=98
scores[2]=99

names[0]=김민수
names[1]=박동수
names[2]=홍길동

members[0].name=김민수
members[1].name=박동수
members[2].name=홍길동
```



배열 항목 정렬

- Arrays.sort(배열)
  - 항목 오름차순으로 정렬
  - 기본 타입이거나 String 배열을 자동 정렬
- 사용자 정의 클래스 배열은 Comparable 인터페이스를 구현해야만 정렬됨

<< 위의 그림 설명 >>  
Array 클래스의 sort() 메소드로 정렬을 하기 이전의 배열들임.  
(정렬이 안된 원래 상태의 배열임)

```
import java.util.Arrays;

public class SearchExample {
    public static void main(String[] args) {
        //기본 타입값 검색 scores 는 로컬변수임
        int[] scores = { 99, 97, 98 };
        Arrays.sort(scores);
        int index = Arrays.binarySearch(scores, 99);
        System.out.println("찾은 인덱스: " + index);

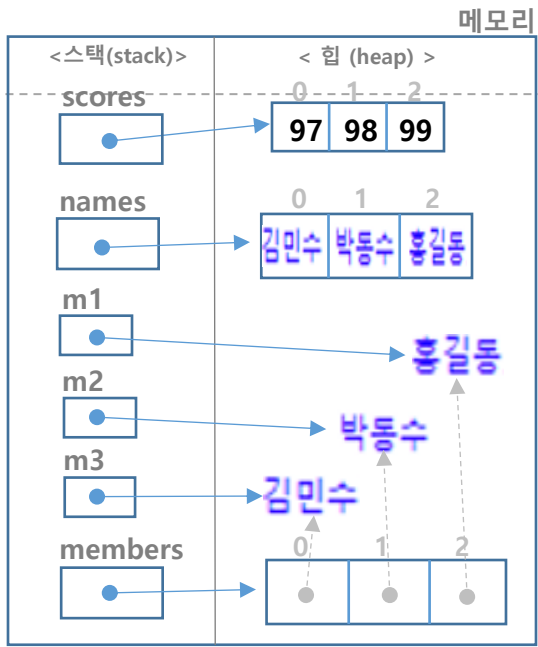
        //문자열 검색 names 는 로컬변수임
        String[] names = { "홍길동", "박동수", "김민수" };
        Arrays.sort(names);
        index = Arrays.binarySearch(names, "홍길동");
        System.out.println("찾은 인덱스: " + index);

        //객체 검색
        Member m1 = new Member("홍길동");
        Member m2 = new Member("박동수");
        Member m3 = new Member("김민수");
        Member[] members = { m1, m2, m3 };
        Arrays.sort(members);
        index = Arrays.binarySearch(members, m1);
        System.out.println("찾은 인덱스: " + index);
    }
}
```

```
public class Member implements Comparable<Member> {
    필드 { String name;
    생성자 { Member(String name) {
        this.name = name;
    }
    메소드 { @Override
        public int compareTo(Member o) {
            return name.compareTo(o.name);
        }
    }
}
```

< 실행결과 >

찾은 인덱스: 2  
찾은 인덱스: 2  
찾은 인덱스: 2



배열 항목 검색

- 특정 값 위치한 인덱스 얻는 것
- Arrays.sort(배열)로 먼저 정렬해야 함
- Arrays.binarySearch(배열, 찾는 값) 메소드로 항목을 찾아야 함

<< 위의 그림 설명 >>  
Array 클래스의 sort() 메소드에 의해 이미 정렬된 배열들임.



# 객체지향 프로그래밍



이미지 출처 : <https://www.brickowl.com/catalog/lego-computer-programmer-minifigure>

교재12장 보충교재  
java.base 모듈

## Date, Calendar 클래스

(java.util.Date, java.util.Calendar)

Date 클래스와 Calendar 클래스 모두  
java.util 패키지 소속임.

## ❖ Date 클래스

12장, 교재 p532

- 날짜를 표현하는 클래스
- 날짜 정보를 객체간에 주고 받을 때 주로 사용
- Date 클래스의 생성자인 Date( )만 주로 사용됨  
(나머지 오버로딩된 생성자들은 deprecated 되어 사용 안함)  
→ Date( ) 생성자는 컴퓨터의 현재 날짜를 읽어서 Date 객체로 만든다

## ❖ Calendar 클래스

12장, 교재 p533~534

- 달력을 표현하는 추상 클래스  
→ 지역별, 문화별 역법 차이를 감안하여 특정 역법의 달력은 자식 클래스에서 구현하도록 설계됨.
- 특정 역법을 사용하는 경우가 아니면, 직접 하위(자식) 클래스를 만들 필요는 없음
- Calender 클래스의 정적(static) 메소드인 getInstance( ) 메소드 이용  
→ 컴퓨터(OS)에 설정되어 있는 시간대(TimeZone)를 기준으로 Calendar 하위 객체 생성

```
Calendar now = Calendar.getInstance();
```

## ■ 날짜 및 시간 정보 얻기

- Calender 클래스가 제공하는 날짜 및 시간에 대한 정보를 얻기 위해서는 get( ) 메소드를 이용함
- get( ) 메소드의 매개값으로 Calender 클래스에 정의된 상수를 주면, 상수가 의미하는 값을 반환(리턴)함.

```
int year    = now.get(Calendar.YEAR);           //년도를 리턴
int month   = now.get(Calendar.MONTH) + 1;      //월을 리턴
int day     = now.get(Calendar.DAY_OF_MONTH);    //일을 리턴
int week    = now.get(Calendar.DAY_OF_WEEK);     //요일을 리턴
int amPm    = now.get(Calendar.AM_PM);          //오전/오후를 리턴
int hour    = now.get(Calendar.HOUR);            //시를 리턴
int minute  = now.get(Calendar.MINUTE);          //분을 리턴
int second  = now.get(Calendar.SECOND);          //초를 리턴
```

# Date 클래스

메모리

```
import java.text.*;
import java.util.*;
```

12장, 교재 p533

```
public class DateExample {
    public static void main(String[] args) {
        Date now = new Date();
        String strNow1 = now.toString();
        System.out.println(strNow1);

        SimpleDateFormat sdf = new SimpleDateFormat("yyyy년 MM월 dd일 hh시 mm분 ss초");
        String strNow2 = sdf.format(now);
        System.out.println(strNow2);
    }
}
```

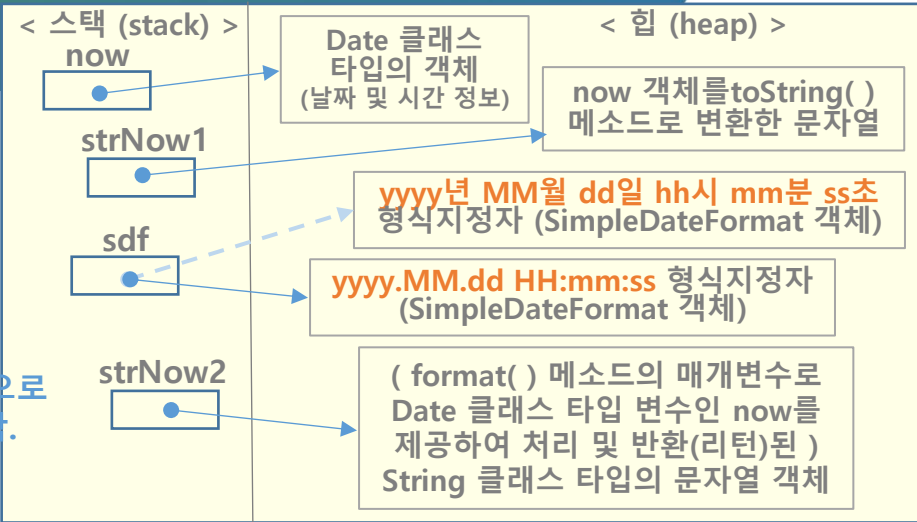
현재 날짜를 문자열로 얻고 싶다면  
toString() 메소드 사용  
→ 날짜 및 시간 정보가 영문으로 출력됨

현재 날짜 정보를 문자열로 얻을 때, 원하는 날짜 형식으로  
얻고 싶다면, SimpleDateFormat 클래스를 이용해야 함.

< 실행결과 >

Wed Feb 02 13:08:14 KST 2022  
2022년 02월 02일 01시 08분 14초

"(따옴표) 안의 내용(형식지정자)을  
yyyy.MM.dd HH:mm:ss 로 바꾸면



< 주의 > 매번 실행할 때마다 실행결과와  
값이 다르게 나올 수 있음.

< 실행결과 > Wed Feb 02 13:08:14 KST 2022  
2022.02.02 01:08:14 로 실행결과가 나타남

```
import java.util.TimeZone;
```

12장, 교재 p537

```
public class PrintTimeZoneID {
    public static void main(String[] args) {
        String[] availableIDs = TimeZone.getAvailableIDs();
        for(String id : availableIDs) {
            System.out.println(id);
        }
    }
}
```

America/Los\_Angeles와 같은 시간대ID는  
TimeZone.getAvailableIDs()라는 메소드가  
반환(리턴)하는 값 중 하나를 사용한다.

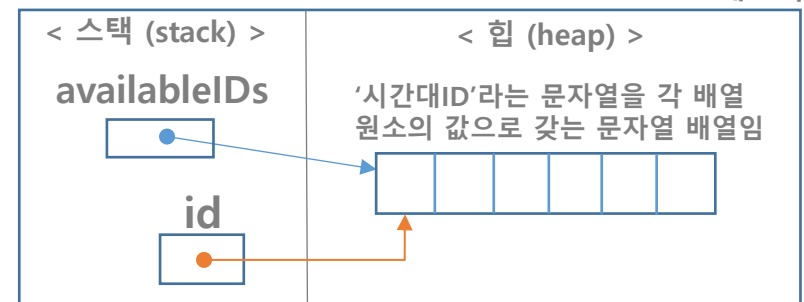
< 실행결과 >

Africa/Abidjan  
Africa/Accra  
Africa/Addis\_Ababa  
Africa/Algiers  
Africa/Asmara  
Africa/Asmera  
Africa/Bamako  
Africa/Bangui

교재 p537의 예제는  
TimeZone.getAvailableIDs() 메소드가  
반환하는 시간대ID를 모두 출력한다.

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

메모리



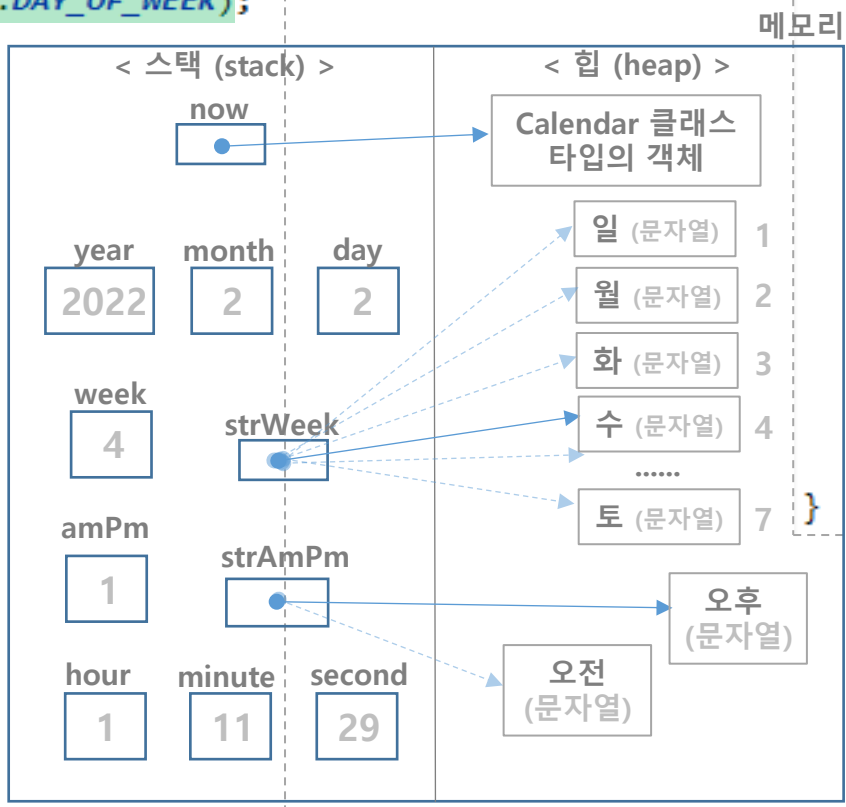
```
import java.util.*;

public class CalendarExample {
    public static void main(String[] args) {
        Calendar now = Calendar.getInstance();

        int year      = now.get(Calendar.YEAR);
        int month     = now.get(Calendar.MONTH) + 1;
        int day       = now.get(Calendar.DAY_OF_MONTH);

        int week      = now.get(Calendar.DAY_OF_WEEK);
        String strWeek = null;
        switch(week) {
            case Calendar.MONDAY:
                strWeek = "월";
                break;
            case Calendar.TUESDAY:
                strWeek = "화";
                break;
            case Calendar.WEDNESDAY:
                strWeek = "수";
                break;
            case Calendar.THURSDAY:
                strWeek = "목";
                break;
            case Calendar.FRIDAY:
                strWeek = "금";
                break;
            case Calendar.SATURDAY:
                strWeek = "토";
                break;
            default:
                strWeek = "일";
        }
    }
}
```

Calendar.AM의 값은 정수 0 (오전)  
Calendar.PM의 값은 정수 1 (오후) 이다.



```
int amPm = now.get(Calendar.AM_PM);
String strAmPm = null;
if(amPm == Calendar.AM) {
    strAmPm = "오전";
} else {
    strAmPm = "오후";
}

int hour      = now.get(Calendar.HOUR);
int minute    = now.get(Calendar.MINUTE);
int second    = now.get(Calendar.SECOND);

System.out.print(year + "년 ");
System.out.print(month + "월 ");
System.out.println(day + "일 ");
System.out.print(strWeek + "요일 ");
System.out.println(strAmPm + " ");
System.out.print(hour + "시 ");
System.out.print(minute + "분 ");
System.out.println(second + "초 ");
}
```

< 실행결과 >

2022년 2월 2일  
수요일 오후  
1시 11분 29초

< 주의 >  
매번 실행할 때마다  
실행결과 값이  
다르게 나올 수 있음.

```

1 package ch12.sec08;
2
3 import java.util.Calendar;
4 import java.util.TimeZone;
5
6 public class LosAngelesExample {
7     public static void main(String[] args) {
8         TimeZone timeZone = TimeZone.getTimeZone("America/Los_Angeles");
9         Calendar now = Calendar.getInstance( timeZone );
10
11         int amPm = now.get(Calendar.AM_PM); Calendar.AM의 값은 정수 0 (오전)
12         String strAmPm = null; Calendar.PM의 값은 정수 1 (오후) 이다.
13         if(amPm == Calendar.AM) {
14             strAmPm = "오전";
15         } else {
16             strAmPm = "오후";
17         }
18         int hour = now.get(Calendar.HOUR);
19         int minute = now.get(Calendar.MINUTE);
20         int second = now.get(Calendar.SECOND);
21
22         System.out.print(strAmPm + " ");
23         System.out.print(hour + "시 ");
24         System.out.print(minute + "분 ");
25         System.out.println(second + "초 ");
26     }
27 }

```

< 실행결과 >

```

Console X
<terminated> LosA
오전 6시 4분 9초

```

<주의>  
매번 실행할 때마다 실행결과 값이 다르게 나올 수 있음.

## ■ 다른 시간대의 Calendar 객체 얻기

\* Calener 클래스의 오버로딩(overloading)된

다른 getInstance( ) 메소드를 이용

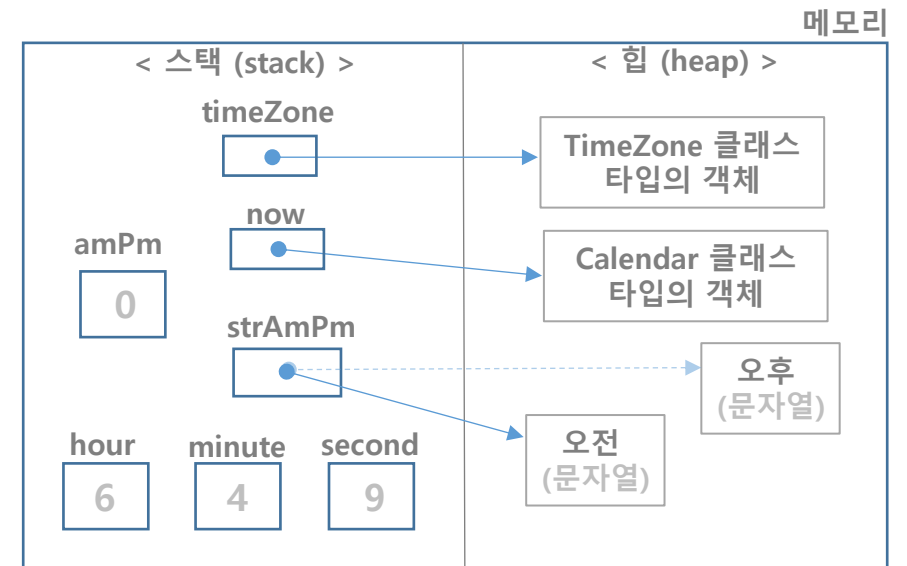
→ 다른 국가나 지역의 시간대의 Calendar 객체를 얻을 수 있음

\* 알고 싶은 시간대의 TimeZone 클래스 타입 객체를 얻어서,  
이 객체를 getInstance( ) 메소드의 매개값으로 넘겨주면 됨.

```

TimeZone timeZon = TimeZone.getTimeZone("America/Los_Angeles");
Calendar now = Calendar.getInstance( timeZon );

```







[교재5장 보충교재]  
**참조타입(reference type)**

[교재12장 보충교재]  
**java.base 모듈**

- 끝 -