11. 기본 API 클래스

과목명: Java 애플리케이션 개발

Contents

Contents

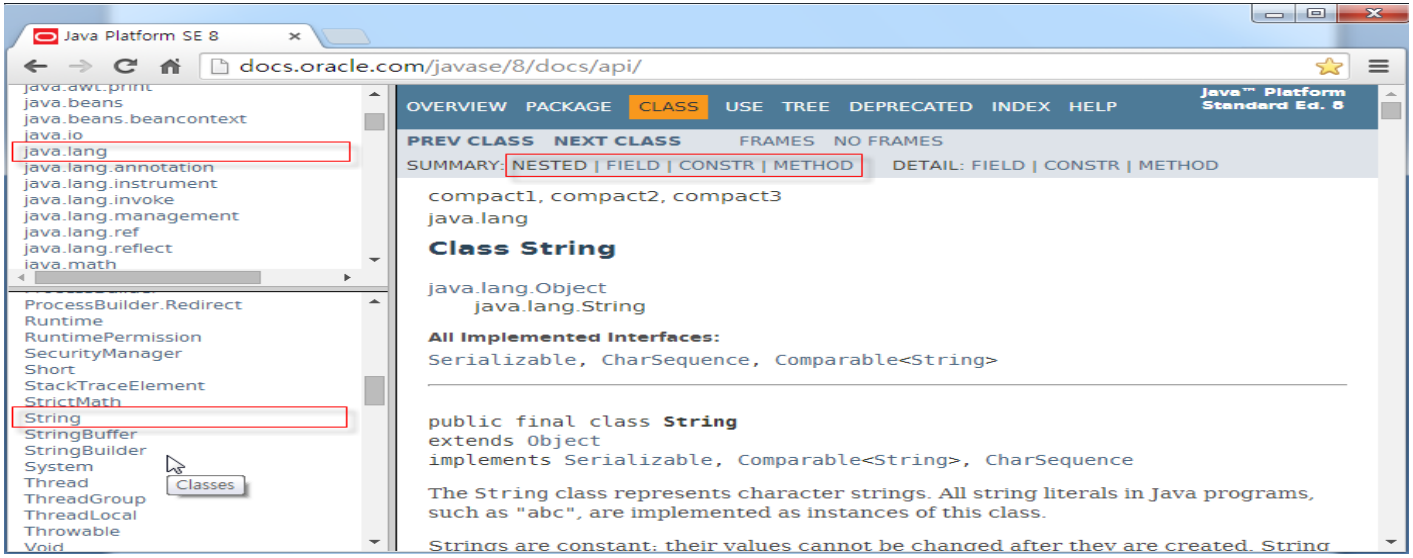
1절. 자바 API 도큐먼트

❖ 자바 API란?

- 자바에서 기본적으로 제공하는 라이브러리(library)
- 프로그램 개발에 자주 사용되는 클래스 및 인터페이스 모음

❖ API 도큐먼트 (p.454~456)

- 쉽게 API 찾아 이용할 수 있도록 문서화한 것
- HTML 페이지로 작성되어 있어 웹 브라우저로 바로 볼 수 있음
  - http://docs.oracle.com/javase/8/docs/api/



2절. java.lang과 java.util 패키지

❖ java.lang 패키지

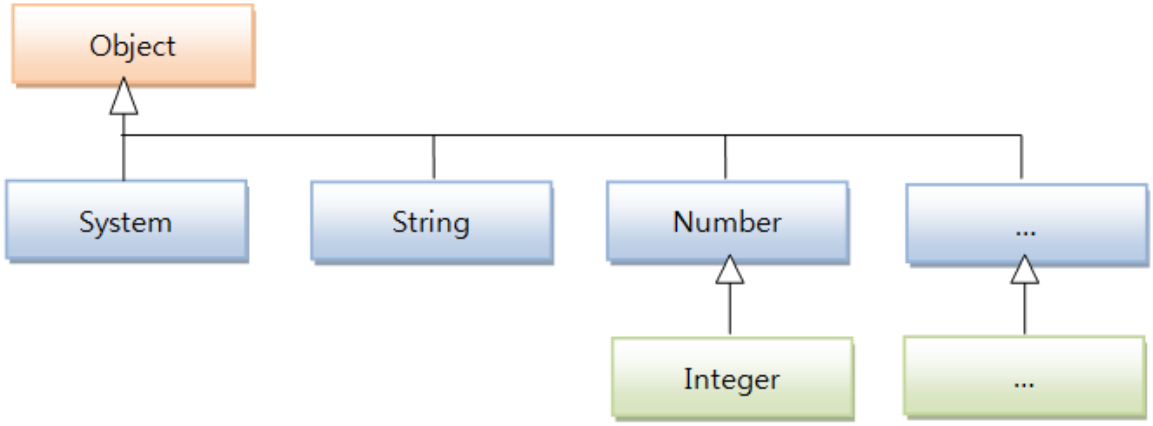
- 자바 프로그램의 기본적인 클래스를 담은 패키지
- 포함된 클래스와 인터페이스는 import 없이 사용
- 주요 클래스

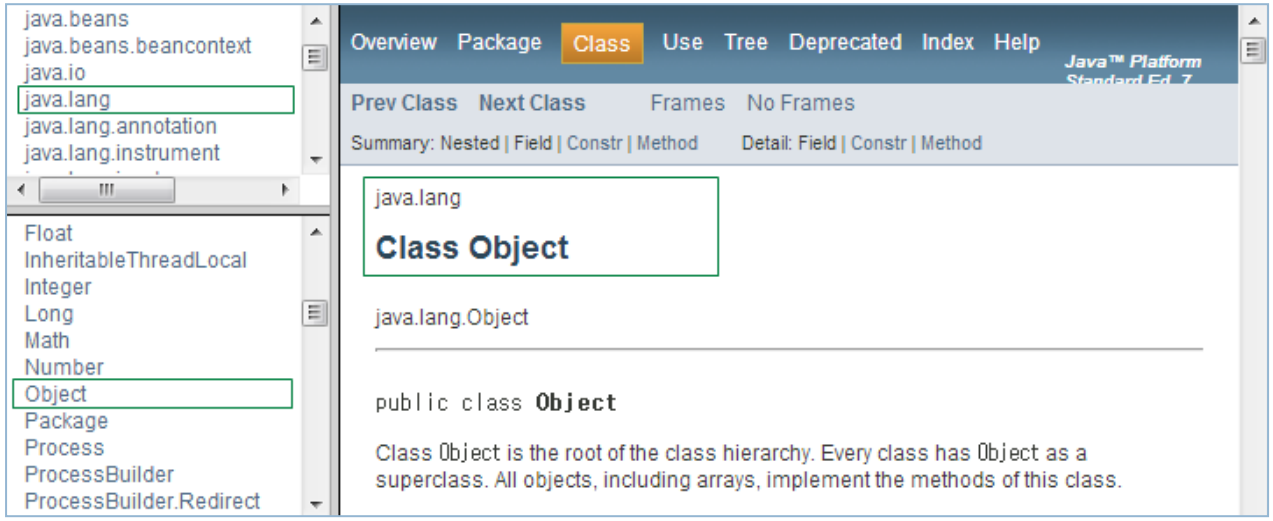
클래스		용도
Object		- 자바 클래스의 최상위 클래스로 사용
System		- 표준 입력장치(키보드)로부터 데이터를 입력 받을 때 사용 - 표준 출력장치(모니터)로 출력하기 위해 사용 - 자바 가상 머신을 종료시킬 때 사용 - 쓰레기 수집기를 실행 요청할 때 사용
Class		- 클래스를 메모리로 로딩할 때 사용
String		- 문자열을 저장하고 여러가지 정보를 얻을 때 사용
StringBuffer, StringBuilder		- 문자열을 저장하고 내부 문자열을 조작할 때 사용
Math		- 수학 함수를 이용할 때 사용
Wrapper	Byte, Short, Character Integer, Float, Double Boolean	- 기본 타입의 데이터를 갖는 객체를 만들 때 사용 - 문자열을 기본 타입으로 변환할 때 사용 - 입력값 검사에 사용

3절. Object 클래스

❖ 자바의 최상위 부모 클래스

- 다른 클래스 상속하지 않으면 java.lang.Object 클래스 상속 암시
- Object의 메소드는 모든 클래스에서 사용 가능





3절. Object 클래스

❖ 객체 비교(equals() 메소드)

```
public boolean equals(Object obj) { ... }
```

- 기본적으로 == 연산자와 동일한 결과 리턴 (번지 비교)

```
Object obj1 = new Object();
Object obj2 = new Object();

boolean result = obj1.equals(obj2);
                 기준 객체   비교 객체

boolean result = (obj1 == obj2)
```

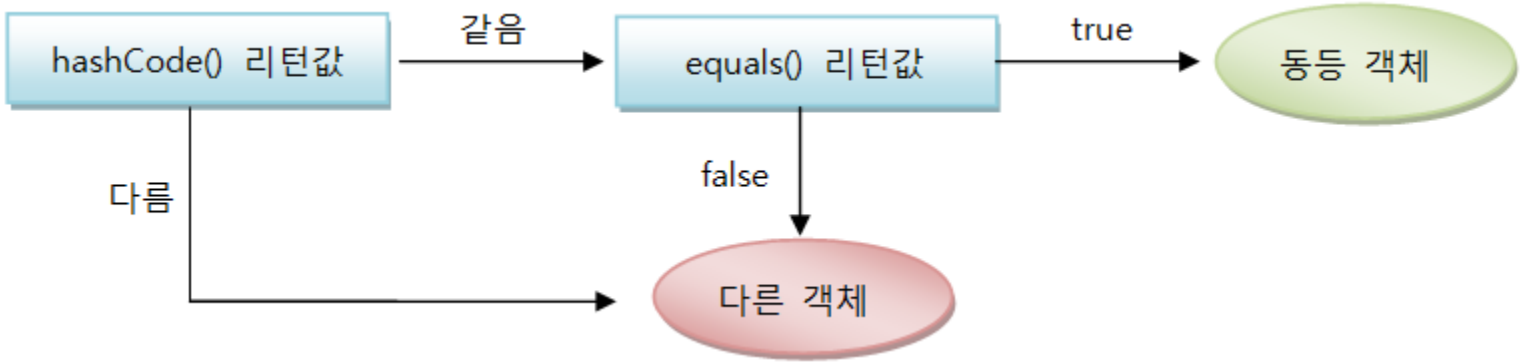
결과가 동일

- 논리적 동등 위해 오버라이딩 필요
  - 논리적 동등이란?
    - 같은 객체이건 다른 객체이건 상관없이 객체 저장 데이터 동일
  - Object의 equals() 메소드
    - 재정의하여 논리적 동등 비교할 때 이용

3절. Object 클래스

❖ 객체 해시코드(hashCode()) (p.461~463)

- 객체 해시코드란?
  - 객체 식별할 하나의 정수값
  - 객체의 메모리 번지 이용해 해시코드 만들어 리턴
    - 개별 객체는 해시코드가 모두 다름
- 논리적 동등 비교 시 hashCode() 오버라이딩의 필요성
  - 컬렉션 프레임워크의 HashSet, HashMap, Hashtable 과 같은 클래스는 두 객체가 동등한 객체인지 판단할 때 아래와 같은 과정을 거침



3절. Object 클래스

❖ 객체 문자정보(toString())

- 객체를 문자열로 표현한 값
- Object 클래스의toString() 메소드는 객체의 문자 정보 리턴

```
Object obj = new Object();
System.out.println( obj.toString() );

[실행 결과]
java.lang.Object@de6ced
```

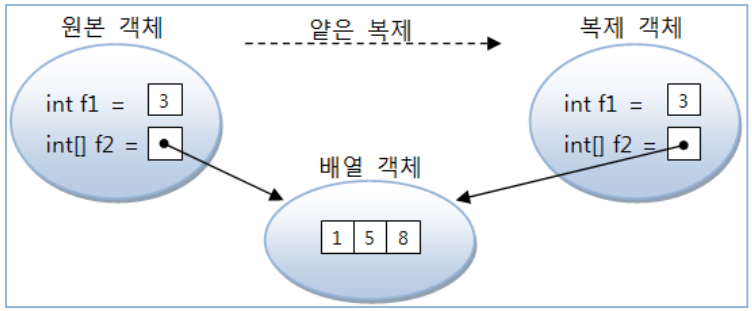
- 일반적으로 의미 있는 문자정보가 나오도록 재정의
  - Date 클래스- 현재 시스템의 날짜와 시간 정보 리턴
  - String 클래스 - 저장하고 있는 문자열 리턴

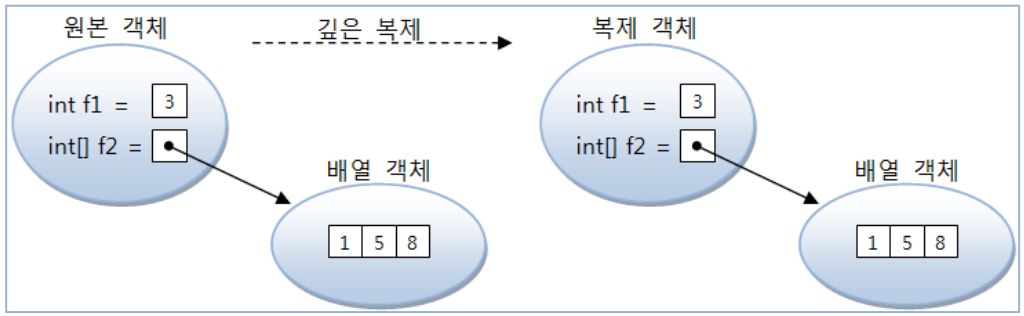
- System.out.pritnln(Object) 메소드
  - Object의 toString()의 리턴값 출력

3절. Object 클래스

❖ 객체 복제(clone())

- 원본 객체의 필드 값과 동일한 값을 가지는 새로운 객체 생성하는 것
- 복제 종류
  - 얕은 복제(thin clone): 필드 값만 복제 (참조 타입 필드는 번지 공유)
  - 깊은 복제(deep clone): 참조하고 있는 객체도 복제





- Object의 clone() 메소드
  - 동일한 필드 값 을 가진 얕은 복제된 객체 리턴
  - java.lang.Cloneable 인터페이스 구현한 객체만 복제 가능

- 깊은 복제 - clone() 메소드 재정의하고 참조 객체도 복제해야

3절. Object 클래스

❖ 객체 소멸자(finalize())

- GC는 객체를 소멸하기 직전 객체 소멸자(finalize()) 실행
- Object의 finalize() 는 기본적으로 실행 내용이 없음
- 객체가 소멸되기 전에 실행할 코드가 있다면?
  - Object의 finalize() 재정의

```
@Override
protected void finalize() throws Throwable {
   System.out.println(no + "번 객체의 finalize()가 실행됨");
}
```

- 될 수 있으면 소멸자는 사용하지 말 것
  - GC는 메모리의 모든 쓰레기 객체를 소멸하지 않음
  - GC의 구동 시점이 일정하지 않음

4절. Objects 클래스

❖ Objects 클래스

- Object의 유틸리티 클래스

리턴타입	메소드(매개변수)	설명
int	compare(T a, T b, Comparator<T> c)	두 객체 a 와 b 를 Comparator 를 사용해서 비교
boolean	deepEquals(Object a, Object b)	두 객체의 깊은 비교(필드도 비교)
boolean	equals(Object a, Object b)	두 객체의 얕은 비교(번지만 비교)
int	hash(Object... values)	매개값이 저장된 배열의 해시코드 생성
int	hashCode(Object o)	객체의 해시코드 생성
boolean	isNull(Object obj)	객체가 널 인지 조사
boolean	nonNull(Object obj)	객체가 널이 아닌지 조사
T	requireNonNull(T obj)	객체가 널인 경우 예외 발생
T	requireNonNull(T obj, String message)	객체가 널인 경우 예외 발생(주어진 예외 메시지 포함)
T	requireNonNull(T obj, Supplier<String> messageSupplier)	객체가 널인 경우 예외 발생(람다식이 만든 예외 메시지 포함)
String	toString(Object o)	객체의 toString() 리턴값 리턴
String	toString(Object o, String nullDefault)	객체의 toString() 리턴값 리턴, 첫번째 매개값이 null 일 경우 두번째 매개값 리턴

4절. Objects 클래스

- **객체 비교 (Objects.compare(T a, T b, Comparator<T> c))**
  - a, b 두 객체를 비교자(c)로 비교해 int값 리턴
  - Comparator<T> 인터페이스
    - 제너릭 인터페이스 타입
    - T 타입의 객체를 비교하는 compare(T a, T b) 메소드 가짐

```
public interface Comparator<T> {
   int compare(T  a, T b)
}
```

    - 비교 예제 p.474~476

4절. Objects 클래스

❖ 동등 비교(equals()와 deepEquals())

- 두 객체의 동등 비교
- Objects.equals(Object a, Object b)

a	b	Objects.equals(a, b)
not null	not null	a.equals(b)
null	not null	false
not null	null	false
null	null	true

- deepEquals(Object a, Object b)
  - 비교할 객체가 배열일 경우 항목 값까지도 비교

a	b	Objects.deepEquals(a, b)
not null (not array)	not null (not array)	a.equals(b)
not null (array)	not null (array)	Arrays.deepEquals(a, b)
not null	null	false
null	not null	false
null	null	true

4절. Objects 클래스

❖ 해시코드 생성(hash(), hashCode())

- Objects.hash(Object... values)
  - 매개값으로 주어진 값들 이용해 해시 코드 생성하는 역할
  - Arrays.hashCode(Object[]) 호출해 해시코드 얻어 리턴
  - 클래스의 hashCode()의 리턴값 생성할 때 유용하게 사용

```
@Override
public int hashCode() {
    return Objects.hash(field1, field2, field3);
}
```

- Objects.hashCode(Object o)
  - o.hashCode() 호출하고 받은 값 리턴
  - 매개값이 null 이면 0 리턴

4절. Objects 클래스

❖ 널 여부 조사(isNull(), nonNull(), requireNonNull())

- Objects.isNull(Object obj)
  - obj가 null일 경우 true
- Objects.nonNull(Object obj)
  - obj가 not null일 경우 true
- requireNonNull()

리턴타입	메소드(매개변수)	설명
T	requireNonNull(T obj)	not null → obj null → NullPointerException
T	requireNonNull(T obj, String message)	not null → obj null → NullPointerException(message)
T	requireNonNull(T obj, Supplier<String> msgSupplier)	not null → obj null → NullPointerException(msgSupplier.get())

4절. Objects 클래스

❖ 객체 문자정보(toString())

- 객체의 문자정보 리턴

- 첫 번째 매개값이 not null - toString ( )으로 얻은 값을 리턴
- null이면 “null” 또는 두 번째 매개값인 nullDefault 리턴

5절. System 클래스

❖ System 클래스 용도

- 운영체제의 기능 일부 이용 가능
  - 프로그램 종료, 키보드로부터 입력, 모니터 출력, 메모리 정리, 현재 시간 읽기
  - 시스템 프로퍼티 읽기, 환경 변수 읽기

❖ 프로그램 종료(exit())

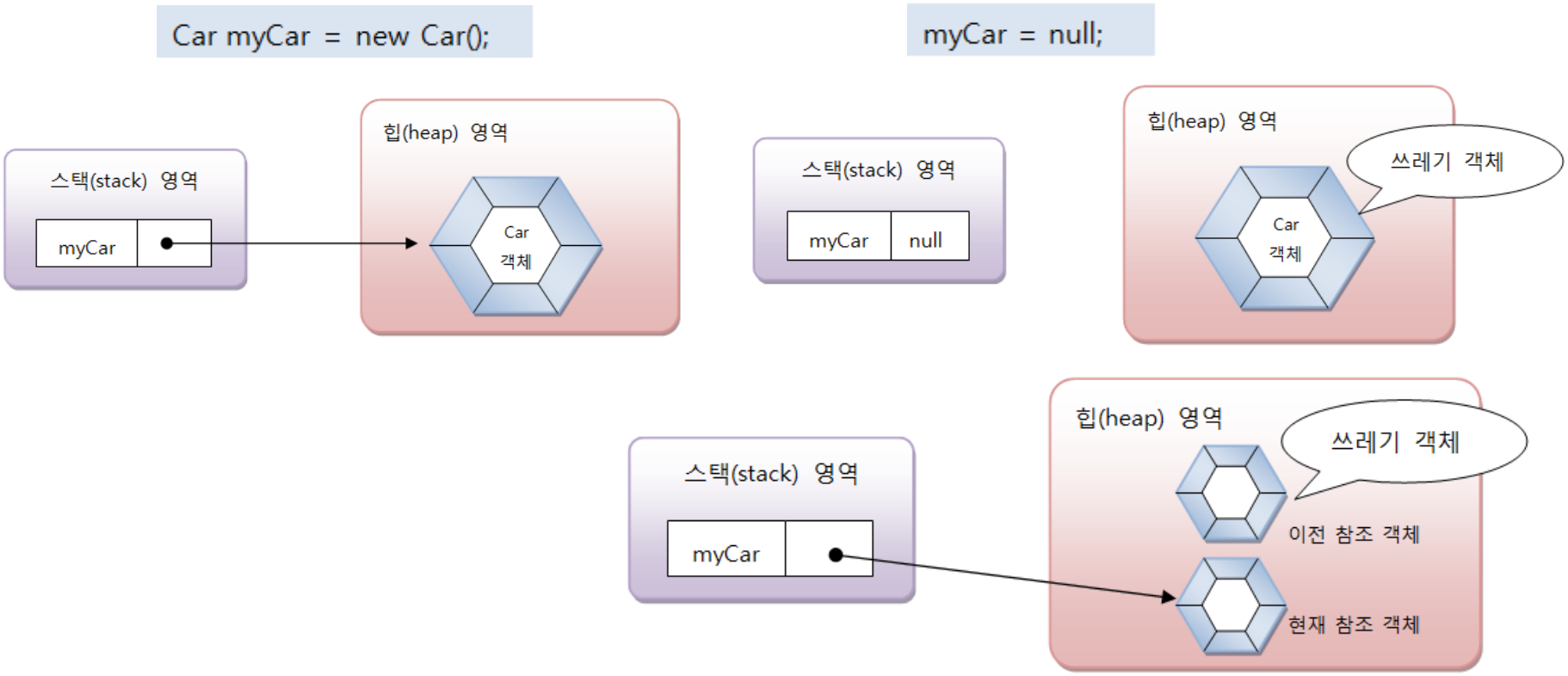
- 기능 - 강제적으로 JVM 종료

```
System.exit(0);
```

  - int 매개값을 지정하도록 - 종료 상태 값
    - 정상 종료일 경우 0, 비정상 종료일 경우 0 이외 다른 값
    - 어떤 값 주더라도 종료
  - 만약 특정 상태 값이 입력되었을 경우에만 종료하고 싶다면?
    - 자바의 보안 관리자 설정

5절. System 클래스

❖ 쓰레기 수집기 실행 요청(gc()) (p.482~484)



5절. System 클래스

❖ 현재 시각 읽기

- 현재 시간을 읽어 밀리 세컨드(currentTimeMillis() -> 1/1000초) 와 나노세컨드(nanoTime()->1/10^9초) 단위의 long값 리턴

```
long time = System.currentTimeMillis();
long time = System.nanoTime();
```

- 주로 프로그램 실행 소요 시간 구할 때 이용

5절. System 클래스

❖ 시스템 프로퍼티 읽기(getProperty())

- 시스템 프로퍼티란?
  - JVM이 시작할 때 자동 설정되는 시스템의 속성값

- 대표적인 키와 값

키(key)	설명	값(value)
java.version	자바의 버전	1.7.0_25
java.home	사용하는 JRE 의 파일 경로	<jdk 설치경로>₩jre
os.name	Operating system name	Windows 7
file.separator	File separator ("/" on UNIX)	₩
user.name	사용자의 이름	사용자계정
user.home	사용자의 홈 디렉토리	C:₩Users₩사용자계정
user.dir	사용자가 현재 작업 중인 디렉토리 경로	다양

- 시스템 프로퍼티 읽어오는 법

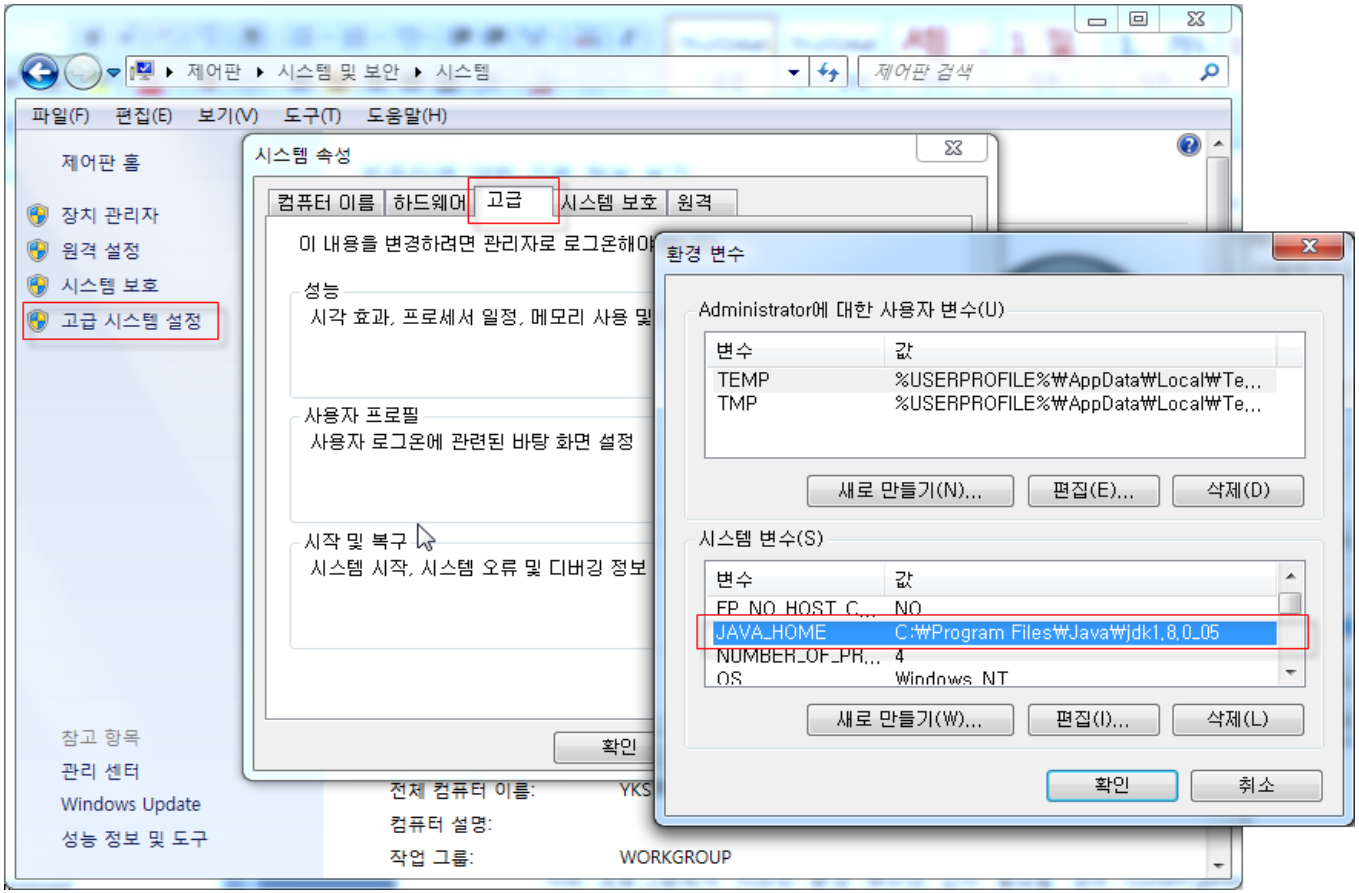
```
String value = System.getProperty(String key);
```

5절. System 클래스

❖ 환경 변수 읽기(getenv())

- 운영체제가 제공하는 환경 변수 값 (문자열) 을 읽음

```
String value = System.getenv (String name);
```



6절. Class 클래스

Class 클래스

- 클래스와 인터페이스의 메타 데이터 관리
  - 메타데이터: 클래스의 이름, 생성자 정보, 필드 정보, 메소드 정보

Class 객체 얻기(getClass(), forName())

- 객체로부터 얻는 방법

```
Class clazz = obj.getClass();
```

- 문자열로부터 얻는 방법

```
try {
   Class clazz = Class.forName(String  className);
} catch (ClassNotFoundException  e) {
}
```

리플렉션 (p.491~493)

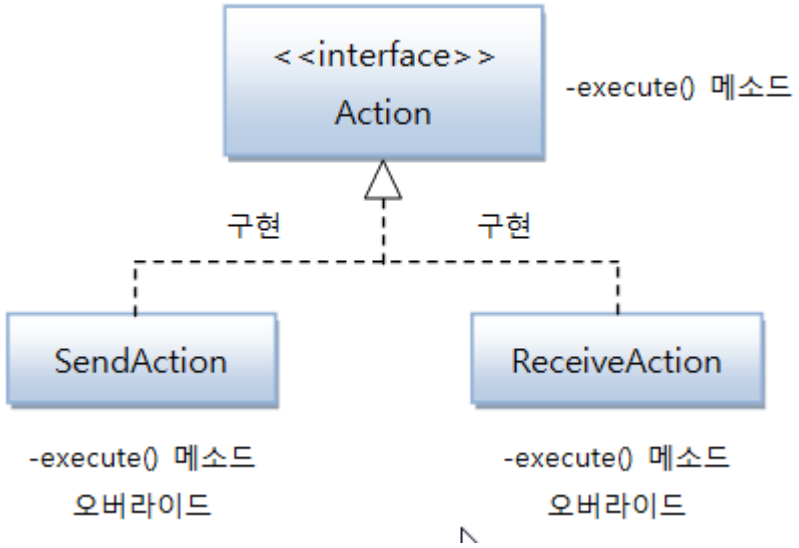
- 클래스의 생성자, 필드, 메소드 정보를 알아내는 것

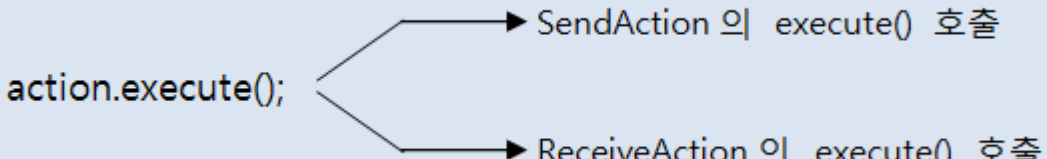
6절. Class 클래스

❖ 동적 객체 생성(newInstance()) (p.493~496)

- 실행 도중 클래스 이름이 결정될 경우 동적 객체 생성 가능

```
Class clazz = Class.forName("SendAction" 또는 "ReceiveAction");
Action action = (Action) clazz.newInstance();
action.execute();
```





7절. String 클래스

❖ 생성자

- byte[] 배열을 문자열로 변환하는 생성자

```
//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)
```

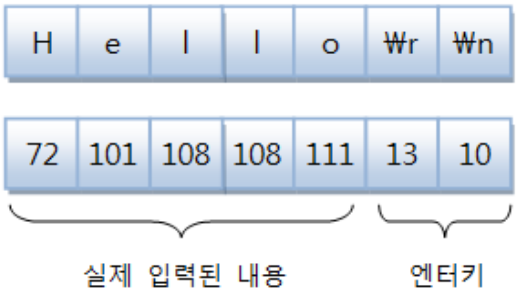
파일의 내용을 읽거나,
네트워크를 통해 받은 데이터는
보통 byte[] 배열이므로
이것을 문자열로 변환하기 위해 사용

- 키보드로부터 읽은 바이트 배열을 문자열로 변환

```
byte[] bytes = new byte[100];
int readByteNo = System.in.read(bytes);
String str = new String(bytes, 0, readByteNo-2);
```

입력내용:

바이트 배열 내용 :



7절. String 클래스

❖ String 메소드

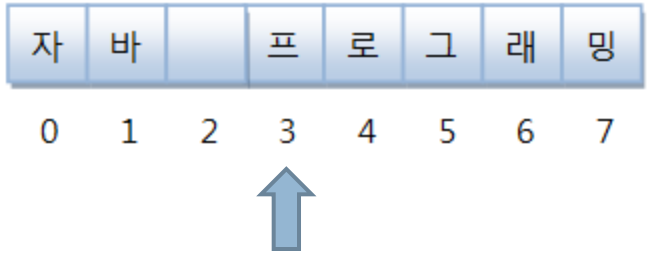
- 문자열의 추출, 비교, 찾기, 분리, 변환등과 같은 다양한 메소드 가짐
- 사용 빈도 높은 메소드

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 짤라낸 새로운 문자열을 리턴
String	substring(int beginIndex, int endIndex)	beginIndex 위치에서 endIndex 전까지 짤라낸 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

7절. String 클래스

- 문자 추출(charAt())
  - 매개 값으로 주어진 인덱스의 문자 리턴

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

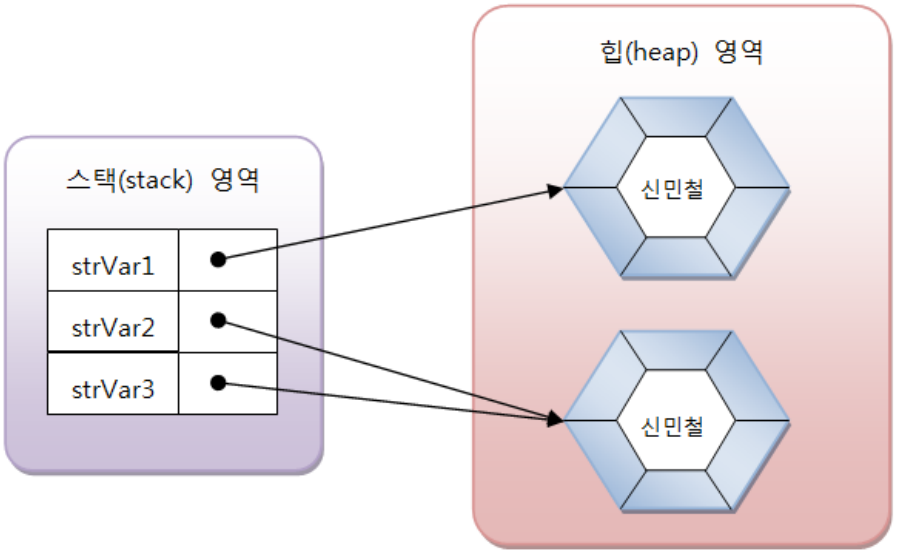


- 문자열 비교(equals())
  - 문자열 비교할 때 == 연산자 사용하면 원하지 않는 결과 발생!

```
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
```



7절. String 클래스

- 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) {
}
```

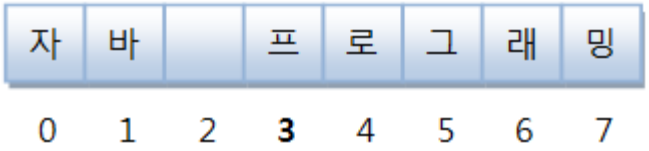
[참고] 디코딩

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

7절. String 클래스

- 문자열 찾기(indexOf())
  - 매개값으로 주어진 문자열이 시작되는 인덱스 리턴
  - 주어진 문자열이 포함되어 있지 않으면 -1 리턴

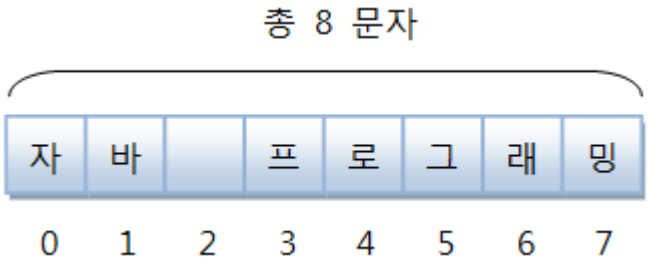
```
String subject = "자바 프로그래밍";
int index = subject.indexOf("프로그래밍");
```



  - 특정 문자열이 포함되어 있는지 여부 따라 실행 코드 달리할 때 사용

- 문자열 길이(length()) – 공백도 문자에 포함

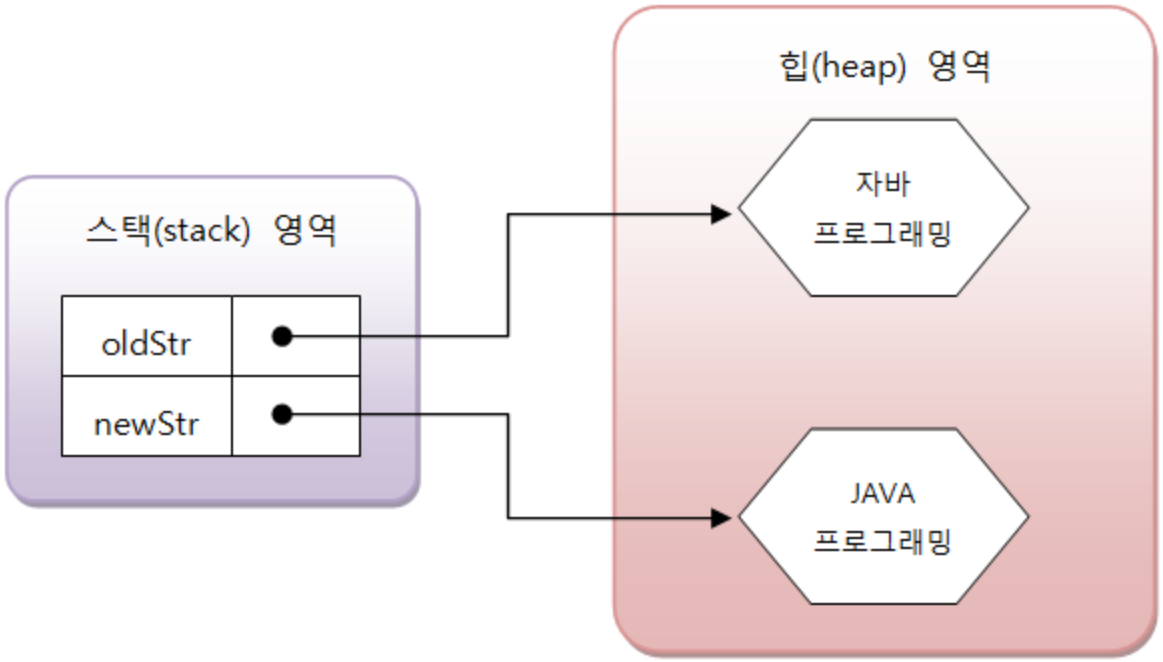
```
String subject = "자바 프로그래밍";
int length = subject.length();
```



7절. String 클래스

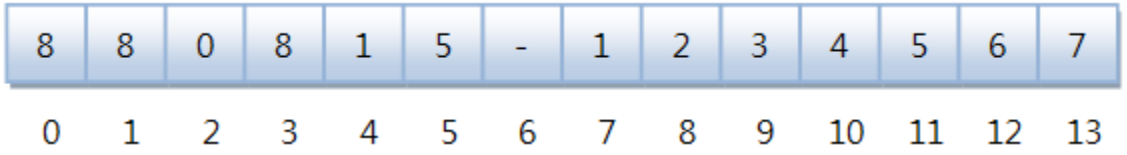
- 문자열 대치(replace())
  - 첫 번째 매개값인 문자열 찾음
  - 두 번째 매개값인 문자열로 대치
  - 새로운 문자열 리턴

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



7절. String 클래스

- 문자열 잘라내기(substring())
  - substring(int beginIndex, int endIndex)
    - 주어진 시작과 끝 인덱스 사이의 문자열 추출
  - substring(int beginIndex)
    - 주어진 인덱스 이후부터 끝까지 문자열 추출

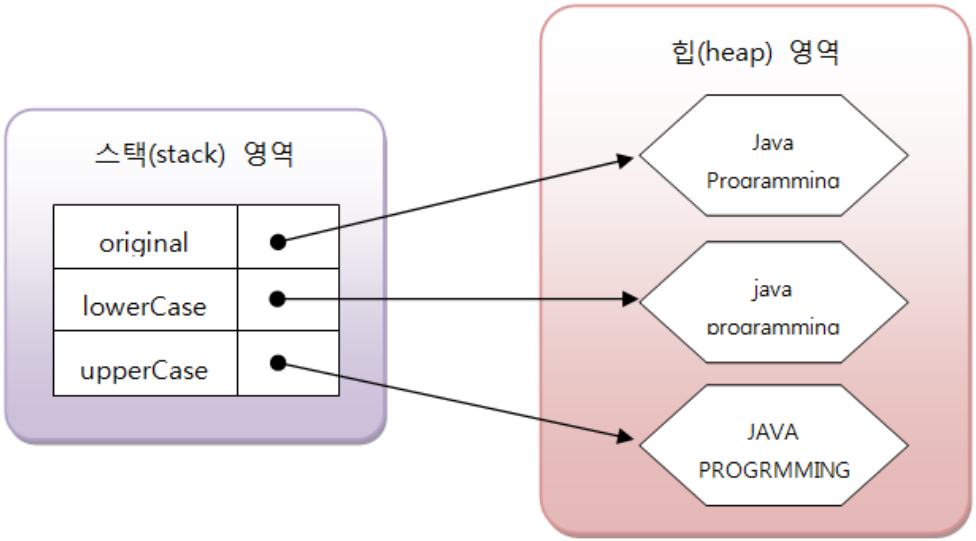



```
String ssn = "880815-1234567";
String firstNum = ssn.substring(0, 6);
String secondNum = ssn.substring(7);
```

7절. String 클래스

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

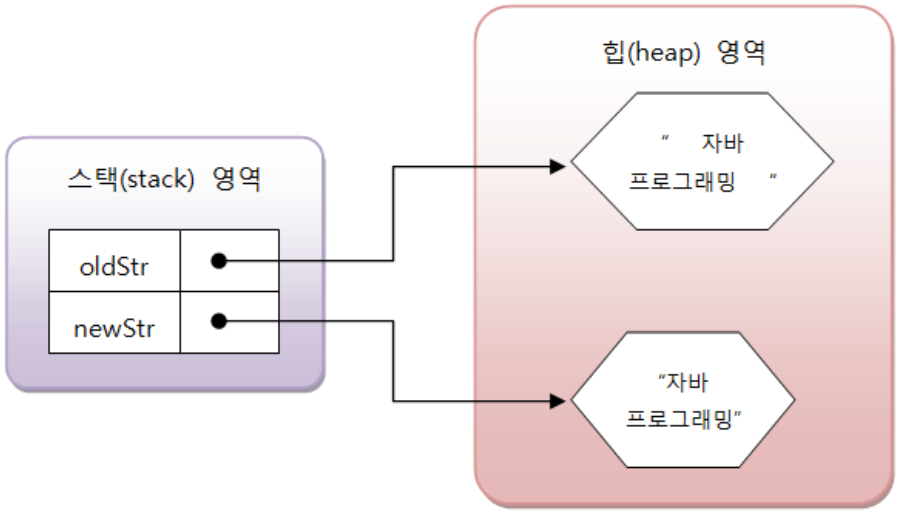
```
String original = "Java Programming";
String lowerCase = original.toLowerCase();
String upperCase = original.toUpperCase();
```



7절. String 클래스

- 문자열 앞뒤 공백 잘라내기(trim())

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



7절. String 클래스

- 문자열 변환(valueOf())
  - 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)
```

8절. StringTokenizer 클래스

❖ 문자열 분리 방법

- String의 split() 메소드 이용
- java.util.StringTokenizer 클래스 이용

❖ String의 split()

- 정규표현식을 구분자로 해서 부분 문자열 분리
- 배열에 저장하고 리턴

```
홍길동&이수홍,박연수,김자바-최명호
```

```
String[] names = text.split("&|,|-");
```

8절. StringTokenizer 클래스

❖ StringTokenizer 클래스

```
String text = "홍길동/이수홍/박연수";
StringTokenizer st = new StringTokenizer(text, "/");
```

```
while( st.hasMoreTokens() ) {
   String token = st.nextToken();
   System.out.println(token);
}
```

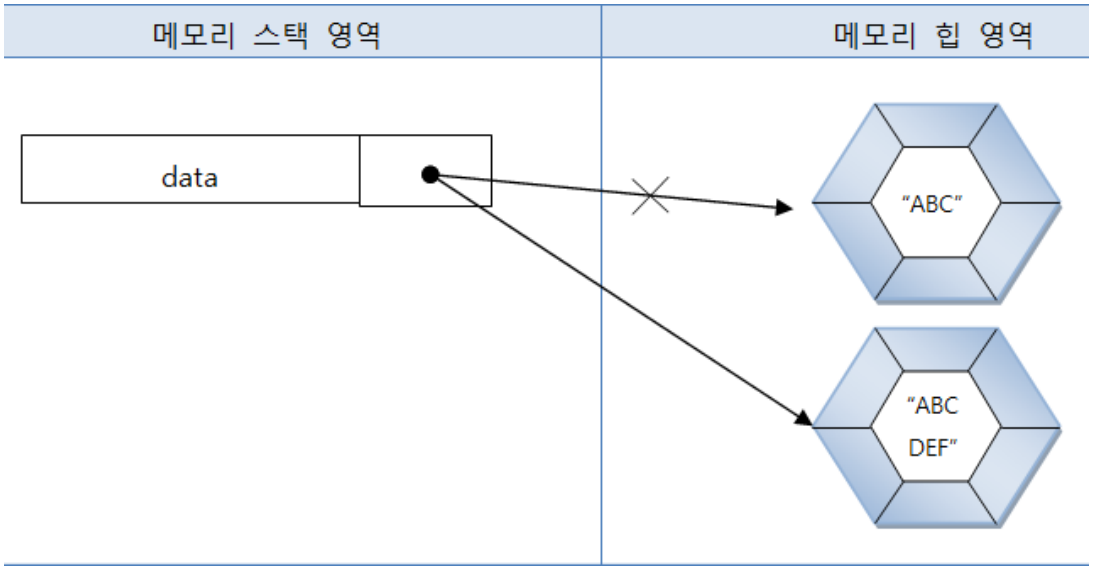
메소드		설명
int	countTokens()	꺼내지 않고 남아있는 토큰의 수
boolean	hasMoreTokens()	남아 있는 토큰이 있는지 여부
String	nextToken()	토큰을 하나씩 꺼내옴

9절. StringBuffer, StringBuilder 클래스

❖ 문자열 결합 연산자 +

- String 은 내부의 문자열 수정 불가 -> 대치된 새로운 문자열 리턴

```
String data = "ABC";
data += "DEF";
```



9절. StringBuffer, StringBuilder 클래스

❖ StringBuffer, StringBuilder (p.514~516)

- 버퍼(buffer:데이터를 임시로 저장하는 메모리)에 문자열 저장
- 버퍼 내부에서 추가, 수정, 삭제 작업 가능
- 멀티 스레드환경: StringBuffer 사용
- 단일 스레드환경: StringBuilder 사용

```
StringBuilder sb = new StringBuilder();
StringBuilder sb = new StringBuilder(16);
StringBuilder sb = new StringBuilder("Java");
```

메소드
append(...)
insert(int offset, ...)
delete(int start, int end)
deleteCharAt(int index)
replace(int start, int end, String str)
StringBuilder reverse()
setCharAt(int index, char ch)

10절. 정규 표현식과 Pattern 클래스

❖ 정규 표현식(Regular Expression) 작성 방법

- 문자열이 정해져 있는 형식으로 구성되어 있는지 검증할 때 사용
  - Ex) 이메일, 전화번호, 비밀번호 등

- 문자 또는 숫자 기호와 반복 기호가 결합된 문자열

기호	설명		
[ ]	한 개의 문자	[abc]	a, b, c 중 하나의 문자
		[^abc]	a, b, c 이외의 하나의 문자
		[a-zA-Z]	a~z, A~Z 중 하나의 문자
₩d	한 개의 숫자, [0-9]와 동일		
₩s	공백		
₩w	한 개의 알파벳 또는 한 개의 숫자, [a-zA-Z_0-9]와 동일		
?	없음 또는 한 개		
*	없음 또는 한 개 이상		
+	한 개 이상		
{n}	정확히 n 개		
{n,}	최소한 n 개		
{n, m}	n 개에서부터 m 개까지		
( )	그룹핑		

10절. 정규 표현식과 Pattern 클래스

❖ Pattern 클래스

- 정규 표현식으로 문자열을 검증하는 역할
  - 결과는 boolean 타입 !!!

```
boolean result = Pattern.matches("정규식", "입력된 문자열");
```

11절. Arrays 클래스

❖ Arrays

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

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)	시작인텍스부터 끝인덱스까지의 항목에만 동일한 값을 저장
void	sort(배열)	배열의 전체 항목을 올림차순으로 정렬

11절. Arrays 클래스

- 배열 복사
  - Arrays.copyOf(원본배열, 복사할 길이)
    - 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)
    - 시작인덱스 ~ (끝 인덱스-1)까지 항목 복사

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

  - System.arraycopy()

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

11절. Arrays 클래스

- 배열 항목 비교
  - Arrays.equals(배열, 배열) - 1차 항목의 값만 비교
  - Arrays.deepEquals(배열, 배열) - 중첩된 배열의 항목까지 비교
- 배열 항목 정렬
  - Arrays.sort(배열)- 항목 오름차 순으로 정렬
    - 기본 타입이거나 String 배열 자동 정렬
  - 사용자 정의 클래스 배열은 Comparable 인터페이스를 구현해야만 정렬
- 배열 항목 검색
  - 특정 값 위치한 인덱스 얻는 것
  - Arrays.sort(배열)로 먼저 정렬
  - Arrays.binarySearch(배열, 찾는 값) 메소드로 항목을 찾아야

12절. 포장(Wrapper) 클래스

❖ 포장(Wrapper) 객체란?

- 기본 타입(byte, char, short, int, long, float, double, boolean) 값을 내부에 두고 포장하는 객체
- 기본 타입의 값은 외부에서 변경 불가

기본 타입	포장 클래스
byte	Byte
char	Character
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean

12절. 포장(Wrapper) 클래스

❖ 박싱(Boxing)과 언박싱(Unboxing)

- 박싱(Boxing): 기본 타입의 값을 포장 객체로 만드는 과정
- 언박싱(Unboxing): 포장 객체에서 기본 타입의 값을 얻어내는 과정
- 박싱 하는 방법
  - 생성자 이용

기본 타입의 값을 줄 경우	문자열을 줄 경우
Byte obj = new Byte(10);	Byte obj = new Byte("10");
Character obj = new Character('가');	
Short obj = new Short(100);	Short obj = new Short("100");
Integer obj = new Integer(1000);	Integer obj = new Integer("1000");
Long obj = new Long(10000);	Long obj = new Long("10000");
Float obj = new Float(2.5F);	Float obj = new Float("2.5F");
Double obj = new Double(3.5);	Double obj = new Double("3.5");
Boolean obj = new Boolean(true);	Boolean obj = new Boolean("true");

  - valueOf() 메소드 이용

```
Integer obj = Integer.valueOf(1000);
Integer obj = Integer.valueOf("1000");
```

12절. 포장(Wrapper) 클래스

- 언박싱 코드
  - 각 포장 클래스마다 가지고 있는 클래스 호출
  - 기본 타입명 + Value()

기본 타입의 값을 이용		
byte	num	= obj.byteValue();
char	ch	= obj.charValue();
short	num	= obj.shortValue();
int	num	= obj.intValue();
long	num	= obj.longValue();
float	num	= obj.floatValue();
double	num	= obj.doubleValue();
boolean	bool	= obj.booleanValue();

12절. 포장(Wrapper) 클래스

❖ 자동 박싱과 언박싱

- 자동 박싱 - 포장 클래스 타입에 기본값이 대입될 경우 발생

```
Integer obj = 100;     //자동 박싱
```

```
List<Integer> list = new ArrayList<Integer>();
list.add(200);    //자동 박싱
```

- 자동 언박싱 - 기본 타입에 포장 객체가 대입될 경우 발생

```
Integer obj = new Integer(200);
int value1 = obj;              //자동 언박싱
int value2 = obj + 100;        //자동 언박싱
```

12절. 포장(Wrapper) 클래스

❖ 문자열을 기본 타입 값으로 변환

- parse + 기본타입 명 → 정적 메소드

기본 타입의 값을 이용		
byte	num	= Byte.parseByte("10");
short	num	= Short.parseShort("100");
int	num	= Integer.parseInt("1000");
long	num	= Long.parseLong("10000");
float	num	= Float.parseFloat("2.5F");
double	num	= Double.parseDouble("3.5");
boolean	bool	= Boolean.parseBoolean("true");

❖ 포장값 비교

- 포장 객체는 내부 값을 비교하기 위해 ==와 != 연산자 사용 불가
- 값을 언박싱해 비교하거나, equals() 메소드로 내부 값 비교할 것

13절. Math, Random 클래스

❖ Math 클래스 (예제는 p.533~536 참고)

- 수학 계산에 사용할 수 있는 정적 메소드 제공

메소드	설명	예제 코드	리턴값
int abs(int a) double abs(double a)	절대값	int v1 = Math.abs(-5); double v2 = Math.abs(-3.14);	v1 = 5 v2 = 3.14
double ceil(double a)	올림값	double v3 = Math.ceil(5.3); double v4 = Math.ceil(-5.3);	v3 = 6.0 v4 = -5.0
double floor(double a)	버림값	double v5 = Math.floor(5.3); double v6 = Math.floor(-5.3);	v5 = 5.0 v6 = -6.0
int max(int a, int b) double max(double a, double b)	최대값	int v7 = Math.max(5, 9); double v8 = Math.max(5.3, 2.5);	v7 = 9 v8 = 5.3
int min(int a, int b) double min(double a, double b)	최소값	int v9 = Math.min(5, 9); double v10 = Math.min(5.3, 2.5);	v9 = 5 v10 = 2.5
double random()	랜덤값	double v11 = Math.random();	0.0<= v11<1.0
double rint(double a)	가까운 정수의 실수값	double v12 = Math.rint(5.3); double v13 = Math.rint(5.7);	v12 = 5.0 v13 = 6.0
long round(double a)	반올림값	long v14 = Math.round(5.3); long v15 = Math.round(5.7);	v14 = 5 v15 = 6

13절. Math, Random 클래스

❖ Random 클래스

- boolean, int, long, float, double 난수 입수 가능
- 난수를 만드는 알고리즘에 사용되는 종자값(seed) 설정 가능
  - 종자값이 같으면 같은 난수

- Random 클래스로 부터 Random객체 생성하는 방법

생성자	설명
Random()	호출시 마다 다른 종자값(현재시간 이용)이 자동 설정된다.
Random(long seed)	매개값으로 주어진 종자값이 설정된다.

- Random 클래스가 제공하는 메소드

리턴값	메소드(매개변수)	설명
boolean	nextBoolean()	boolean 타입의 난수를 리턴
double	nextDouble()	double 타입의 난수를 리턴(0.0<= ~ < 1.0)
int	nextInt()	int 타입의 난수를 리턴(-2^{32}<= ~ <=2^{32}-1);
int	nextInt(int n)	int 타입의 난수를 리턴(0<= ~ <n)

14절. Date, Calendar 클래스

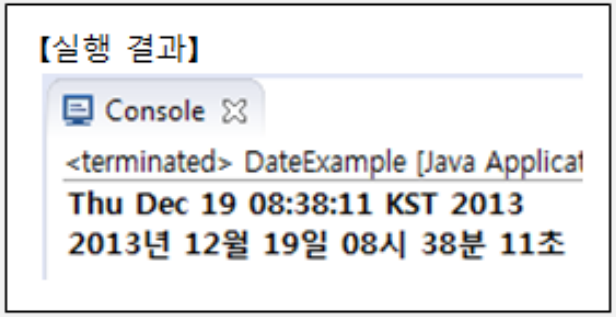
❖ Date 클래스

- 날짜를 표현하는 클래스
- 날짜 정보를 객체간에 주고 받을 때 주로 사용

```
Date now = new Date();
String strNow1 = now.toString();
System.out.println(strNow1);


SimpleDateFormat sdf =
            new SimpleDateFormat("yyyy 년 MM 월 dd 일 hh 시 mm 분 ss 초");
String strNow2 = sdf.format(now);
System.out.println(strNow2);
```

【실행 결과】



14절. Date, Calendar 클래스

❖ Calendar 클래스

- 달력을 표현한 추상 클래스
- OS에 설정된 시간대(TimeZone) 기준의 Calendar 객체 얻기

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

- 다른 시간대의 Calendar 객체 얻기

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

- 날짜 및 시간 정보 얻기

```
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
```

15절. Format 클래스

❖ 형식(Format) 클래스

- 숫자와 날짜를 원하는 형식의 문자열로 변환
- 종류
  - 숫자 형식: DecimalFormat
  - 날짜 형식: SimpleDateFormat
  - 매개변수화 된 문자열 형식: MessageFormat

❖ 숫자 형식 클래스(DecimalFormat)

- 적용할 패턴 선택해 생성자 매개값으로 지정 후 객체 생성

```
DecimalFormat df = new DecimalFormat("#,###.0");
String result =   df.format(1234567.89);
```

15절. Format 클래스

❖ 날짜 형식 클래스(SimpleDateFormat)

```
SimpleDateFormat sdf = new SimpleDateFormat("yyyy 년 MM 월 dd 일");
String strDate = sdf.format(new Date());
```

패턴 문자	의미	패턴 문자	의미
y	년	H	시(0~23)
M	월	h	시(1~12)
d	일	K	시(0~11)
D	월 구분이 없는 일(1~365)	k	시(1~24)
E	요일	m	분
a	오전/오후	s	초
w	년의 몇번째 주	S	밀리세컨드(1/1000 초)
W	월의 몇번째 주		

❖ 매개변수화 된 문자열 형식 클래스(MessageFormat)

```
String message = "회원 ID: {0} ₩n 회원 이름: {1} ₩n 회원 전화: {2}";
String result = MessageFormat.format(message, id, name, tel);
```

```
String text = "회원 ID: {0} ₩n 회원 이름: {1} ₩n 회원 전화: {2}";
Object[] arguments = { id, name, tel };
String result = MessageFormat.format(text, arguments);
```

16절. java.time 패키지

java.time 패키지

- 자바8부터 추가된 패키지
- 날짜와 시간을 나타내는 여러 가지 API가 새롭게 추가됨
- 날짜와 시간을 조작하거나 비교하는 기능이 추가됨
  - Date와 Calendar는 날짜와 시간을 조작하거나 비교하는 기능이 불충분

패키지	설명
java.time	날짜와 시간을 나타내는 핵심 API 인 LocalDate, LocalTime, LocalDateTime, ZonedDateTime 을 포함하고 있다. 이들 클래스는 ISO-8601 에 정의된 달력 시스템에 기초한다.
java.time.chrono	ISO-8601 에 정의된 달력 시스템 이외에 다른 달력 시스템을 사용코저할때 사용할 수 있는 API 들이 포함되어 있다.
java.time.format	날짜와 시간을 파싱하고 포맷팅하는 API 들이 포함되어 있다.
java.time.temporal	날짜와 시간을 연산하기 위한 보조 API 들이 포함되어 있다.
java.time.zone	타임존을 지원하는 API 들이 포함되어 있다.

16절. java.time 패키지

❖ 날짜와 시간 객체 생성

- 날짜와 시간을 표현하는 5개의 클래스

클래스명	설명
LocalDate	로컬 날짜 클래스
LocalTime	로컬 시간 클래스
LocalDateTime	로컬 날짜 및 시간 클래스(LocalDate + LocalTime)
ZonedDateTime	특정 타임존(TimeZone)의 날짜와 시간 클래스
Instant	특정 시점의 Time-Stamp 클래스

16절. java.time 패키지

❖ 날짜와 시간에 대한 정보 얻기

클래스	리턴타입	메소드(매개변수)	설명
LocalDate	int	getYear()	년
	Month	getMonth()	Month 열거값
	int	getMonthValue()	월
	int	getDayOfYear	일년의 몇번째 일
	int	getDayOfMonth()	월의 몇번째 일
	DayOfWeek	getDayOfWeek()	요일
	boolean	isLeapYear()	윤년 여부
LocalTime	int	getHour()	시간
	int	getMinute()	분
	int	getSecond()	초
	int	getNano()	나노초 리턴

- isLeapYear()는 toLocalDate() 메소드로 LocalDate로 변환 후 사용
- ZonedDateTime에서 제공하는 추가 메소드

클래스	리턴타입	메소드(매개변수)	설명
ZonedDateTime	ZoneId	getZone()	존아이디를 리턴 (예: Asia/Seoul)
	ZoneOffset	getOffset()	존오프셋(시차)을 리턴

16절. java.time 패키지

❖ 날짜와 시간을 조작하기 (p.566~559)

- 빼기와 더하기
  - 빼기 – minus + 변수 (long) 의 형태
    - Ex) minusYears(long) → 년 빼기
  - 더하기 – plus + 변수 (long) 의 형태

16절. java.time 패키지

- 변경하기
  - with(TemporalAdjuster adjuster)
    - 현재 날짜를 기준으로 상대적 날짜 리턴
  - TemporalAdjuster 객체는 아래 표에 있는 정적 메소드로 얻음

메소드(매개변수)	설명
firstDayOfYear()	이번 해의 첫일
lastDayOfYear()	이번 해의 마지막 일
firstDayOfNextYear()	다음 해의 첫일
firstDayOfMonth()	이번 달의 첫일
lastDayOfMonth()	이번 달의 마지막 일
firstDayOfNextMonth()	다음 달의 첫일
firstInMonth(DayOfWeek dayOfWeek)	이번 달의 첫 요일
lastInMonth(DayOfWeek dayOfWeek)	이번 달의 마지막 요일
next(DayOfWeek dayOfWeek)	돌아오는 요일
nextOrSame(DayOfWeek dayOfWeek)	돌아오는 요일(오늘 포함)
previous(DayOfWeek dayOfWeek)c	지난 요일
previousOrSame(DayOfWeek dayOfWeek)	지난 요일(오늘 포함)

16절. java.time 패키지

❖ 날짜와 시간을 비교하기

- **Period:** 년, 달, 일의 양을 나타내는 날짜 기준 클래스
- **Duration:** 시, 분, 초, 나노초의 양을 나타내는 시간 기준 클래스

클래스	리턴타입	메소드(매개변수)	설명
LocalDate LocalDateTime	boolean	isAfter(ChronoLocalDate other)	이후 날짜인지?
		isBefore(ChronoLocalDate other)	이전 날짜인지?
		isEqual(ChronoLocalDate other)	동일 날짜인지?
LocalTime LocalDateTime	boolean	isAfter(LocalTime other)	이후 시간인지?
		isBefore(LocalTime other)	이전 시간인지?
LocalDate	Period	until(ChronoLocalDate endDateExclusive)	날짜 차이
LocalDate LocalTime LocalDateTime	long	until(Temporal endExclusive, TemporalUnit unit)	시간 차이
Period	Period	between(LocalDate startDateInclusive, LocalDate endDateExclusive)	날짜 차이
Duration	Duration	between(Temporal startInclusive, Temporal endExclusive)	시간 차이
ChronoUnit.YEARS	long	between(Temporal temporal1Inclusive, Temporal temporal2Exclusive)	전체 년 차이
ChronoUnit.MONTHS			전체 달 차이
ChronoUnit.WEEKS			전체 주 차이
ChronoUnit.DAYS			전체 일 차이
ChronoUnit.HOURS			전체 시간 차이
ChronoUnit.SECONDS			전체 초 차이
ChronoUnit.MILLIS			전체 밀리초 차이
ChronoUnit.NANOS			전체 나노초 차이

16절. java.time 패키지

- Period와 Duration
  - Period: 년, 달, 일의 양을 나타내는 날짜 기준 클래스
  - Duration: 시, 분, 초, 나노초의 양을 나타내는 시간 기준 클래스

클래스	리턴타입	메소드(매개변수)	설명
Period	int	getYears()	년의 차이
	int	getMonths()	달의 차이
	int	getDays()	일의 차이
Duration	int	getSeconds()	초의 차이
	int	getNano()	나노초의 차이

- between() 메소드의 차이점
  - Period와 Duration의 between()
    - 년, 달, 일, 초의 단순 차이를 리턴
  - ChronoUnit의 between()
    - 전체 시간을 기준으로 차이를 리턴

16절. java.time 패키지

❖ 파싱과 포맷팅

- 파싱: 주어진 문자열로 날짜와 시간을 생성
- 포맷팅: 날짜와 시간을 형식화된 문자열로 변환

❖ 파싱(Parsing) 메소드 (p.563~565)

- 상황에 맞는 포맷 변환 같이 사용

클래스	리턴타입	메소드(매개변수)
LocalDate LocalTime LocalDateTime ZonedDateTime	LocalDate LocalTime LocalDateTime ZonedDateTime	parse(CharSequence)
		parse(CharSequence, DateTimeFormatter)

16절. java.time 패키지

❖ 포맷팅(Formatting) 메소드

- 날짜와 시간을 포맷팅된 문자열로 변환

클래스	리턴타입	메소드(매개변수)
LocalDate LocalTime LocalDateTime ZonedDateTime	String	format(DateTimeFormatter formatter)

```
LocalDateTime now = LocalDateTime.now();
DateTimeFormatter dateTimeFormatter =
   DateTimeFormatter.ofPattern("yyyy 년 M 월 d 일 a h 시 m 분");
String nowString = now.format(dateTimeFormatter);
```