

11장 기본 API 클래스

11.1 자바 API 도큐먼트

- 쉽게 API를 찾아 이용할 수 있도록 문서화한 것을 말한다.
- <http://docs.oracle.com/javase/8/docs/api/index.html>

11.2 java.lang과 java.util 패키지

11.2.1. java.lang 패키지

- java.lang 패키지에 있는 클래스와 인터페이스는 import 없이 사용할 수 있다.

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	- 기본 타입의 데이터를 갖는 객체를 만들 때 사용
	Integer, Float, Double	- 문자열을 기본 타입으로 변환할 때 사용
	Boolean	- 입력값 검사에 사용

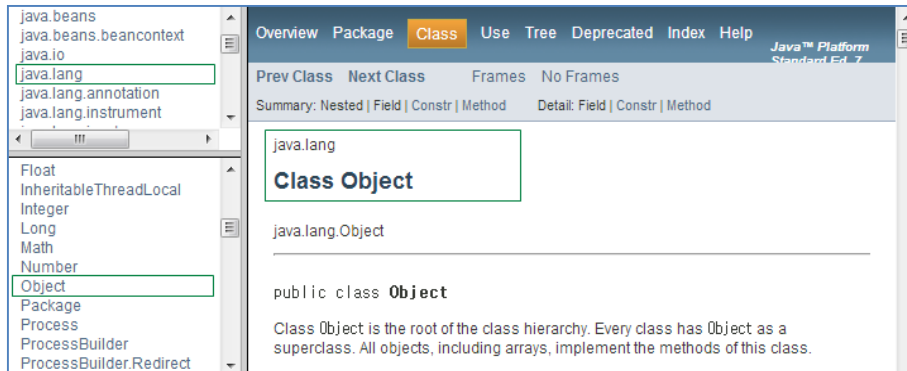
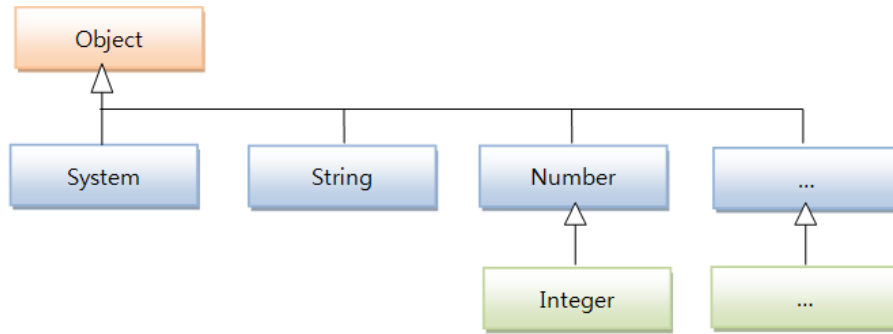
11.2.2. java.util 패키지

- 자바 프로그램 개발에 재미로 같은 역할을 하는 클래스를 담고 있다. 컬렉션 클래스들이 대부분을 차지하고 있다.

클래스	용도
Arrays	배열을 조작할 때 사용
Calendar	운영체제의 날짜와 시간을 얻을 때 사용
Date	날짜와 시간 정보를 저장하는 클래스
Objects	객체 비교, 널(null) 여부 등을 조사할 때 사용
StringTokenizer	특정 문자로 구분된 문자열을 뽑아낼 때 사용
Random	난수를 얻을 때 사용

11.3 Object 클래스

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



11.3.1. 객체 비교(equals())

- 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 재정의(오버라이딩) 필요
 - 논리적으로 동등하다는 것은 같은 객체이건 다른 객체이건 상관없이 객체가 저장하고 있는 데이터가 동일함을 뜻한다.
 - String 클래스는 Object의 equals() 메소드를 재정의해서 번지 비교가 아닌 문자열 비교로 변경되어 있다.

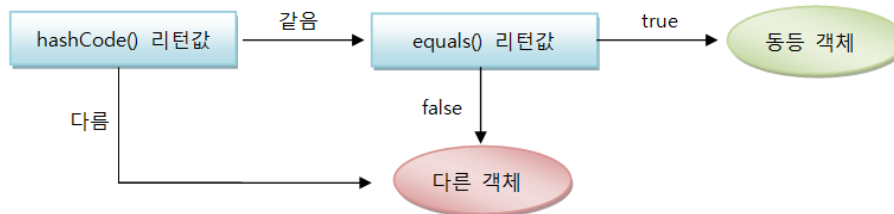
11.3.2. 객체 해시코드(hashCode())

- 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



11.3.3. 객체 문자 정보(toString())



- Object의 toString() 메소드의 리턴값은 별 값어치가 없는 정보 -> Object 하위 클래스는 toString() 메소드를 재정의하여 사용한다.
 - Data 클래스 : 현재 시스템의 날짜와 시간 정보를 리턴한다.
 - String 클래스 : 문자열을 리턴한다.
- System.out.println() 메소드의 매개값으로 객체를 주면 객체의 toString() 메소드를 호출해서 리턴값을 받아 출력한다.
 - 예) System.out.println(obj) = System.out.println(obj.toString())

```

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

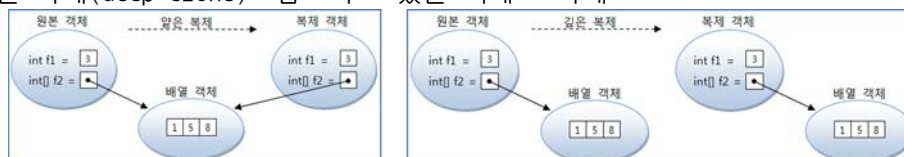
[실행 결과]
java.lang.Object@de6ced
    
```

11.3.4. 객체 복제(clone())

- 원본 객체의 필드 값과 동일한 값을 가지는 새로운 객체 생성하는 것을 말한다.

■ 복제 종류

- 얕은 복제(thin clone): 필드 값만 복제 (참조 타입 필드는 번지 공유)
- 깊은 복제(deep clone): 참조하고 있는 객체도 복제



- 메소드로 객체를 복제하려면 원본 객체는 반드시 java.lang.Cloneable 인터페이스를 구현하고 있어야 한다.

11.3.5. 객체 소멸자(finalize())

- 쓰레기 수집기는 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");
}
```

11.4 Objects 클래스

- Object의 유틸리티 클래스이다.

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

11.4.1 객체 비교(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)
}
```

11.4.2 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

11.4.3 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)

- hashCode() 호출하고 받은 값 리턴
- 매개값이 null 이면 0 리턴

11.4.4 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())

11.4.5 객체 문자 정보(toString())


- 객체의 문자정보 리턴
- 첫 번째 매개값이 not null - toString ()으로 얻은 값을 리턴
- null이면 "null" 또는 두 번째 매개값인 nullDefault 리턴

11.5 System 클래스

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

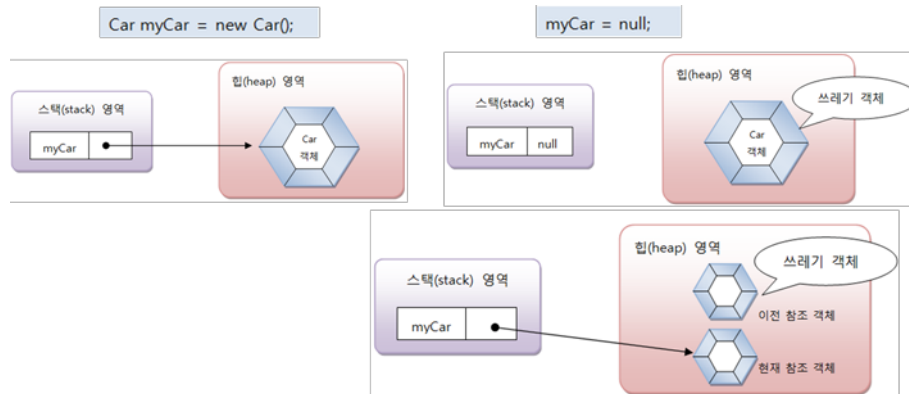
11.5.1 프로그램 종료(exit())

- 기능 - 강제로 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

11.5.2 쓰레기 수집기 실행(gc())



11.5.3 현재 시각 읽기(`currentTimeMillis()`, `nanotime()`)

- 현재 시간을 읽어 밀리 세컨드(`currentTimeMillis()` -> 1/1000초)와 나노세컨드(`nanotime()` -> 1/10⁹초) 단위의 long값 리턴
- 주로 프로그램 실행 소요 시간 구할 때 이용

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

11.5.4 시스템 프로퍼티 읽기(`getProperty()`)

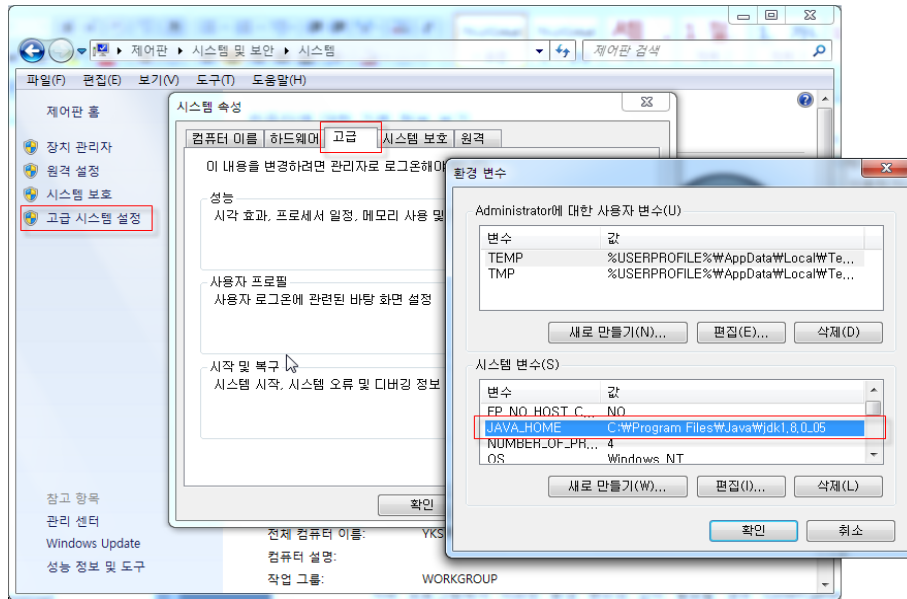
- 시스템 프로퍼티란?
 - JVM이 시작할 때 자동 설정되는 시스템의 속성값
- 대표적인 키와 값

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

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

11.5.5 환경 변수 읽기(`getenv()`)

- 운영체제가 제공하는 환경 변수 값(문자열)을 읽음



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

11.6 Class 클래스

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

11.6.1 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) {
}
```

11.6.2 리플렉션(getDeclaredConstructors(), getDeclaredFields(), getDeclaredMethods())

- Class 객체를 이용하여 클래스의 생성자, 필드, 메소드 정보를 알아내는 것

```
Constructor[] constructors = clazz.getDeclaredConstructors();
Field[] fields = clazz.getDeclaredFields();
Method[] methods = clazz.getDeclaredMethods();
```


[ReflectionExample.java]

```
package sec06.exam02_reflection;

import java.lang.reflect.*;

public class ReflectionExample {
    public static void main(String[] args) throws Exception {
        Class clazz = Class.forName("sec06.exam02_reflection.Car");

        System.out.println("[클래스 이름]");
        System.out.println(clazz.getName());
        System.out.println();

        System.out.println("[생성자 정보]");
        Constructor[] constructors = clazz.getDeclaredConstructors();
        for(Constructor constructor : constructors) {
            System.out.print(constructor.getName() + "(");
            Class[] parameters = constructor.getParameterTypes();
            printParameters(parameters);
            System.out.println(")");
        }
        System.out.println();

        System.out.println("[필드 정보]");
        Field[] fields = clazz.getDeclaredFields();
        for(Field field : fields) {
            System.out.println(field.getType().getSimpleName() + " " + field.getName());
        }
        System.out.println();

        System.out.println("[메소드 정보]");
        Method[] methods = clazz.getDeclaredMethods();
        for(Method method : methods) {
            System.out.print(method.getName() + "(");
            Class[] parameters = method.getParameterTypes();
            printParameters(parameters);
            System.out.println(")");
        }
    }

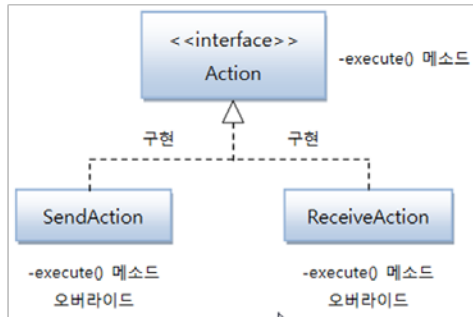
    private static void printParameters(Class[] parameters) {
        for(int i=0; i<parameters.length; i++) {
            System.out.print(parameters[i].getName());
            if(i<(parameters.length-1)) {
                System.out.print(", ");
            }
        }
    }
}
```

11.6.3 동적 객체 생성(newInstance())

- 실행 도중 클래스 이름이 결정될 경우 동적 객체 생성 가능

```
try {
    Class clazz = Class.forName("런타임 시 결정되는 클래스 이름");
    Object obj = clazz.newInstance();
}
```

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



action.execute();

- SendAction 의 execute() 호출
- ReceiveAction 의 execute() 호출



11.7 String 클래스

- 어떤 프로그램이건 문자열은 데이터로서 아주 많이 사용된다. 그렇기 때문에 문자열을 생성하는 방법과 추출, 비교, 찾기, 분리, 변환 등을 제공하는 메소드를 잘 익혀두어야 한다.

11.7.1 String 생성자

- byte[] 배열을 문자열로 변환하는 생성자 - 사용 빈도수가 높은 생성자들로 파일의 내용을 읽거나, 네트워크를 통해 받은 데이터는 보통 byte[] 배열이므로 이것을 문자열로 변환하기 위해 사용된다.
- 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:

H e l l o W r W n

바이트 배열 내용 :

72 101 108 108 111 13 10

실제 입력된 내용 엔터키



```
//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);
```

[KeyboardToStringExample.java] 바이트 배열을 문자열로 변환

```
package sec07.exam01_constructor;

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); //-2=캐리지리턴(\r)+라인피드(\n)
        System.out.println(str);
    }
}
```

11.7.2 String 메소드

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

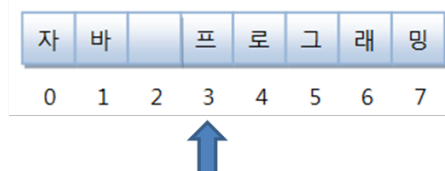
- 사용 빈도 높은 메소드

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 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

(1) 문자 추출(charAt())

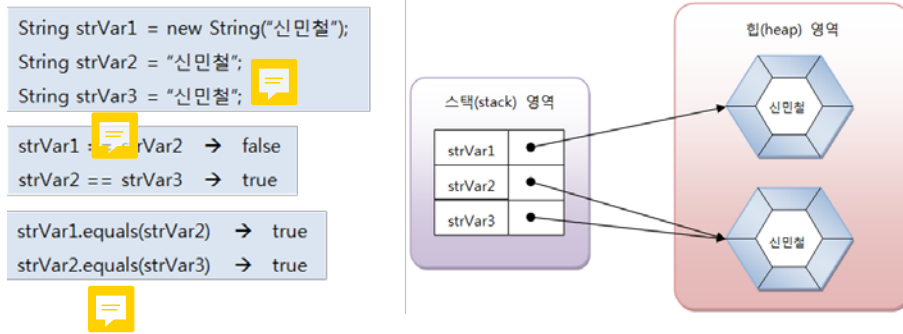
- 매개 값으로 주어진 인덱스의 문자 리턴

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



(2) 문자열 비교>equals())

- 문자열 비교할 때 == 연산자 사용하면 원하지 않는 결과 발생!



(3) 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) {
}
```

[StringGetBytesExample.java] 바이트 배열로 변환

```
package sec07.exam02_method;

import java.io.UnsupportedEncodingException;

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

        byte[] bytes1 = str.getBytes();
        System.out.println("bytes1.length: " + bytes1.length);
        String str1 = new String(bytes1);
        System.out.println("bytes1->String: " + str1);

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

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

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



(4) 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



(5) 문자열 길이(length())

- 공백도 문자에 포함

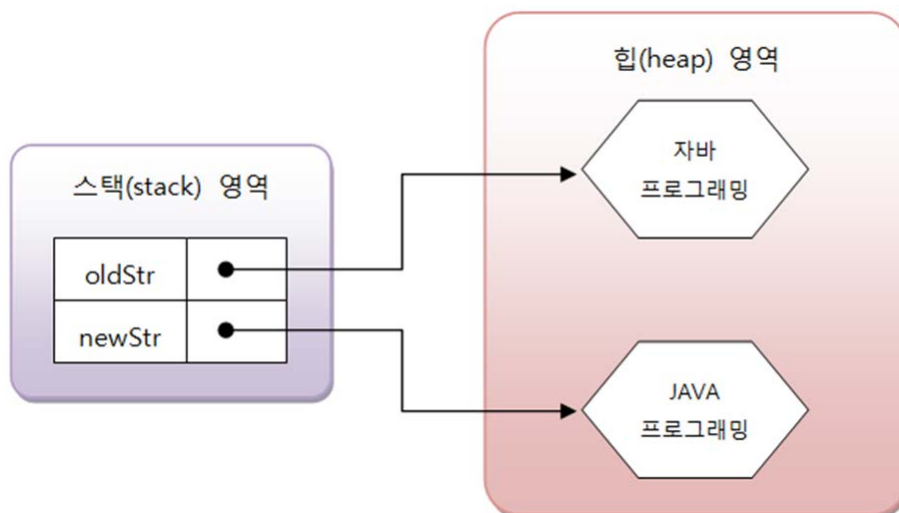
```
String subject = "자바 프로그래밍";
int length = subject.length();
```

총 8 문자							
자	바		프	로	그	래	밍
0	1	2	3	4	5	6	7

(6) 문자열 대체(replace())

- 첫 번째 매개값인 문자열 찾을 -> 두 번째 매개값인 문자열로 대체 -> 새로운 문자열 리턴

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



(7) 문자열 잘라내기(substring())

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

8	8	0	8	1	5	-	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7	8	9	10	11	12	13

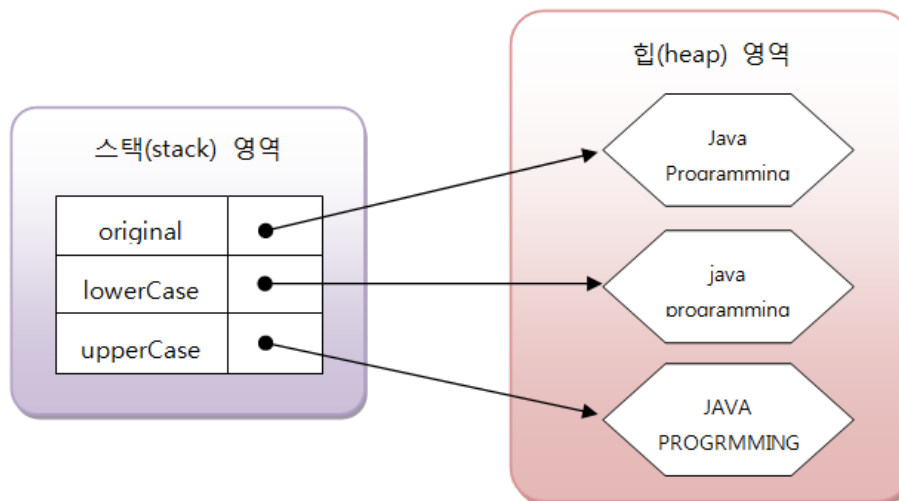
String ssn = "880815-1234567";

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

String secondNum = ssn.substring(7);

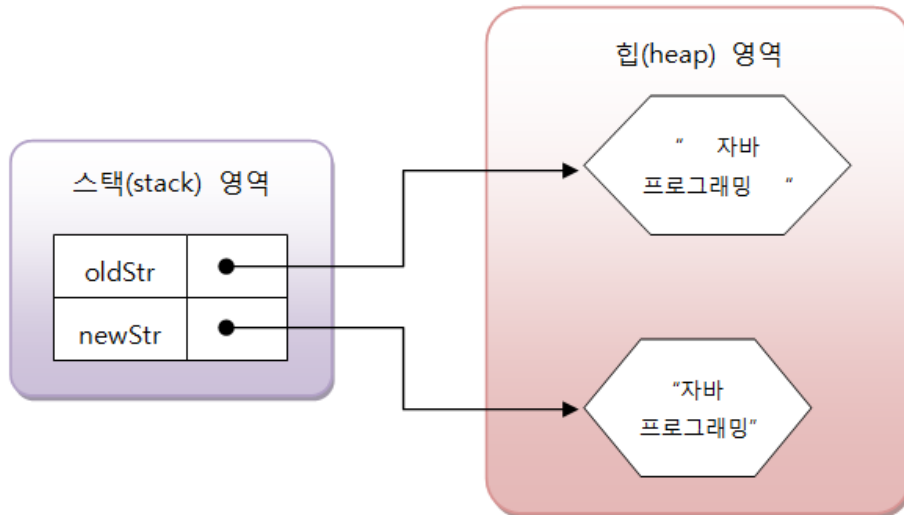
(8) 알파벳 소.대문자 변경(`toLowerCase()`, `toUpperCase()`)

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



(9) 문자열 앞뒤 공백 잘라내기(`trim()`)

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



(10) 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)
```

[StringValueOfExample.java] 기본 타입 값을 문자열로 변환

```
package sec07.exam02_method;

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);
    }
}
```

11.8 StringTokenizer 클래스

11.8.1 split() 메소드

- 정규표현식을 구분자로 해서 부분 문자열 분리
- 배열에 저장하고 리턴

```
// 기본형
String[] result = "문자열".split("정규표현식");

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

[StringSplitExample.java]

```
package sec08.exam01_split_stringtokenizer;

public class StringSplitExample {
    public static void main(String[] args) {
        String text = "홍길동&이수홍,박연수,김자바~최명호";
        String[] names = text.split("&|,|-");

        for (String name : names) {
            System.out.println(name);
        }
    }
}
```

11.8.2 StringTokenizer 클래스

- 문자열이 한 종류의 구분자로 연결되어 있을 경우, StringTokenizer 클래스를 사용하면 손쉽게 문자열을 분리해 낼 수 있다.
- StringTokenizer의 메소드

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

```
//기본형
StringTokenizer st = new StringTokenizer("문자열", "구분자");

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

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

[StringTokenizerExample.java]

```
package sec08.exam01_split_stringtokenizer;

import java.util.StringTokenizer;

public class StringTokenizerExample {
    public static void main(String[] args) {
        String text = "홍길동/이수홍/박연수";
```



```
// how1: 전체 토큰 수를 얻어 for문으로 루핑
StringTokenizer st = new StringTokenizer(text, "/");
int countTokens = st.countTokens();
for (int i = 0; i < countTokens; i++) {
    String token = st.nextToken();
    System.out.println(token);
}

System.out.println();

// how2: 남아 있는 토큰을 확인하고 while문으로 루핑
st = new StringTokenizer(text, "/");
while (st.hasMoreTokens()) {
    String token = st.nextToken();
    System.out.println(token);
}

}
```

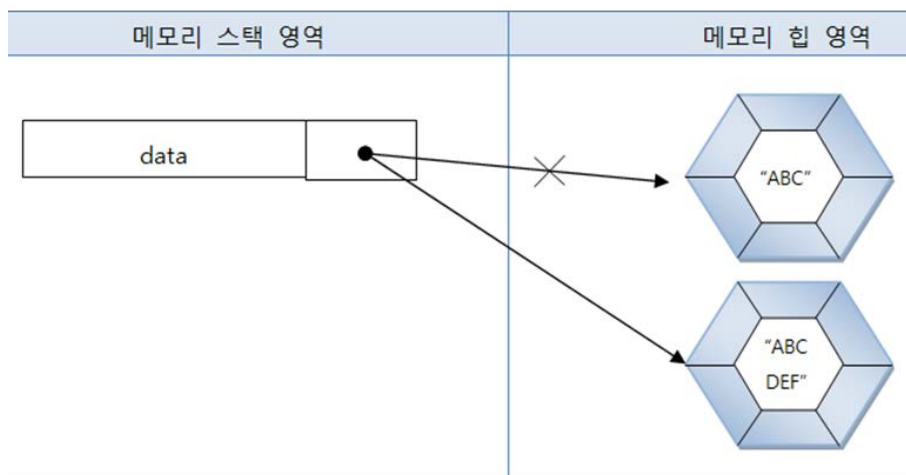


11.9 StringBuffer, StringBuilder 클래스

■ 문자열 결합 연산자 +

- String 은 내부의 문자열 수정 불가 -> 대치된 새로운 문자열 리턴
- String 객체의 수가 늘어나기 때문에 프로그램 성능을 느리게 하는 요인이 된다.

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



■ 버퍼(buffer:데이터를 임시로 저장하는 메모리)에 문자열 저장

- 버퍼 내부에서 추가, 수정, 삭제 작업 가능 -> String 처럼 새로운 객체를 만들지 않고도 문자열을 조작할 수 있다.
- 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)

11.10 정규 표현식과 Pattern 클래스

11.10.1 정규 표현식 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		

11.10.2 Pattern 클래스



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

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

[PatternExample.java]

```
package sec10_exam01_pattern_match;

import java.util.regex.Pattern;

public class PatternExample {
    public static void main(String[] args) {
        String regExp = "(02|010)-\\d{3,4}-\\d{4}";
        String data = "010-123-4567";
        boolean result = Pattern.matches(regExp, data);
        if (result) {
            System.out.println("정규식과 일치합니다.");
        } else {
            System.out.println("정규식과 일치하지 않습니다.");
        }

        regExp = "\\w+@\\w+\\.\\w+(\\.\\w+)?";
        data = "angel@navercom";
        result = Pattern.matches(regExp, data);
        if (result) {
            System.out.println("정규식과 일치합니다.");
        } else {
            System.out.println("정규식과 일치하지 않습니다.");
        }
    }
}
```

```
}
```

[과제] 주민번호 유효성 검사

1. 키보드를 통해서 주민번호를 입력 했을때 타당한 주민번호 인지 아닌지를 판별하는 프로그램을 작성 하세요. (단, 타당하지 않은 주민번호의 경우에는 메시지를 출력)

1. 주민번호 앞자리 6자리 아니면 메시지 출력
2. 주민번호 뒷자리 7자리 아니면 메시지 출력
3. 유효한 주민번호 아니면 메시지 출력

■ 주민 등록 번호 타당성 검사법

1. 주민 번호 각 자리에 해당 숫자를 곱한다.

예) 8 0 0 1 0 1 - 1 2 3 4 5 6 7

체크용 번호

X) 2 3 4 5 6 7 8 9 2 3 4 5

$$(8*2)+(0*3)+(0*4)+(1*5)+(0*6)+(1*7)+(1*8)+(2*9)+(3*2)+(4*3)+(5*4)+(6*5) = 122$$

2. 1번의 연산 결과를 11로 나누어서 나머지를 구한다.

$$122 \ / \ 11 = 11 \ \dots\dots\dots 1$$

(몫) (나머지)

3. 11에서 나머지 값을 뺀후 “ 체크용 번호와 ” 와 비교해서 같으면 올바른 주민번호, 그렇지 않으면 틀린 주민 번호

$$11 - 1 = \quad 10 \quad \neq \quad 7$$

(연산결과) (체크용 번호)

만약, 연산결과가 10이상이면, 다시 10으로 나누어서 그 나머지를 체크용 번호와 비교한다.

$$10 \ / \ 10 = 1 \ \dots\dots\dots 0 \quad \neq \quad 7$$

(몫) (나머지) (체크용 번호)

∴ 위 주민번호는 틀린 주민 번호임.

[JuminCheck01.java]

```
package verify;

import java.io.InputStream;
import java.util.Scanner;

public class JuminCheck01 {
```

```
// 주민번호 타당성 검사하는 메소드
public boolean juminCheck(String jumin) {
    ...
}

public static void main(String[] args) {
    // TODO Auto-generated method stub
    JuminCheck01 jc = new JuminCheck01();

    InputStream is = System.in;
    Scanner sc = new Scanner(is);

    System.out.println("주민번호 앞자리를 입력 하세요?");
    String jumin1 = sc.next(); // 111111
    System.out.println("주민번호 뒷자리를 입력 하세요?");
    String jumin2 = sc.next(); // 1111011

    if (jumin1.length() != 6) {
        System.out.println("주민번호 앞자리는 6자리 입력");
    } else if (jumin2.length() != 7) {
        System.out.println("주민번호 뒷자리는 7자리 입력");
    } else if (!jc.juminCheck(jumin1 + jumin2)) {
        System.out.println("잘못된 주민번호 입니다.");
    } else {
        System.out.println("올바른 주민번호 입니다");
    }
}
}
```



11.11 Arrays 클래스



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

11.11.1 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
```

11.11.2 배열 항목 비교

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



11.11.3 배열 항목 정렬

- `Arrays.sort(배열)`- 항목 오름차 순으로 정렬
 - 기본 타입이거나 String 배열 자동 정렬
- 사용자 정의 클래스 배열은 `Comparable` 인터페이스를 구현해야만 정렬

11.11.4 배열 항목 검색

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

11.12 Wrapper(포장) 클래스

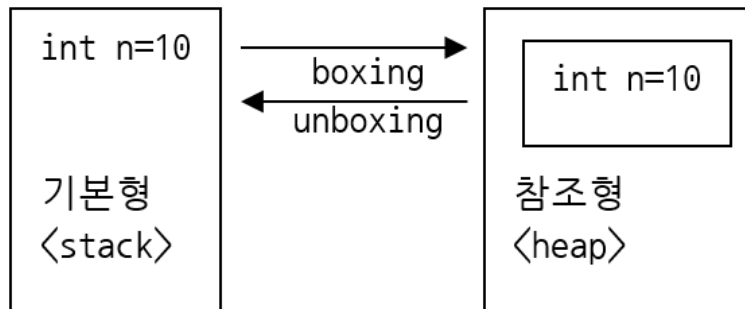


- 기본형의 값을 갖는 객체를 생성할 수 있는데 이런 객체를 포장(Wrapper) 객체라고 한다. 그 이유는 기본형의 값을 내부에 두고 포장하기 때문이다. 포장된 기본형 값을 외부에서 변경하려면 새로운 포장 객체를 만들어야 한다.

기본형	Wrapper 클래스
boolean	Boolean
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character

■ 스택과 힙 영역 사용

```
int n=10;
Integer n1 = new Integer(n); // 박싱, boxing
```



11.12.1 박싱(Boxing)과 언박싱(Unboxing)

- Boxing: Heap메모리를 한 개의 박스로 생각하고 Stack메모리에 있는 데이터를 Heap메모리에 복사하는 것
- Unboxing: Heap메모리의 데이터를 Stack메모리로 복사하는 것.

[WrapperTestA.java] 박싱과 언박싱

```
class WrapperTestA {
    public static void main(String[] args) {
        int n01 = 10;
        int n02;
        Integer num01; // Integer 객체 생성
        Integer num02 = new Integer(20);

        num01 = n01; // 오토 박싱
        // num01 = new Integer(n01);

        n02 = num02; // 오토 언 박싱
        // n02 = num02.intValue();

        System.out.println(n01 + ", " + num01);
        System.out.println(n02 + ", " + num02);
    }
}
```

11.12.2 자동 박싱(Auto-boxing)과 자동 언박싱(Auto-unboxing)

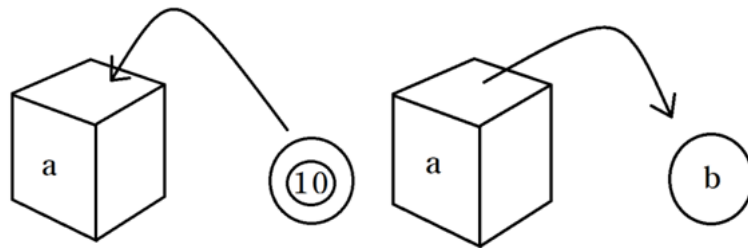
- 자동 박싱은 포장 클래스 타입에 기본값이 대입될 경우에 발생한다.
- 자동 언박싱은 기본 타입에 포장 객체가 대입될 경우에 발생한다.

```
// 자동 박싱
Integer obj = 10;
// Integer obj = new Integer(10);
List<Integer> list = new ArrayList<Integer>();
list.add(10);

// 자동 언박싱
Integer obj = new Integer(10);
int value1 = obj;
```

11.12.3 문자열을 기본 타입 값으로 변환

- **int -> String** : 10 -> "10"
 - 방법1: String a = String.valueOf(10);
 - 방법2: String a = Integer.toString(10);
 - 방법3: String a = 10+"";
- **String -> int** : "10" -> 10
 - 방법1: int b = Integer.parseInt("10");
 - 방법2: Integer num = new Integer("10"); int b = num.intValue();



11.12.4 포장 값 비교

- 내부의 값을 비교하려면 언박싱한 값을 얻어 비교하거나, equals() 메소드로 내부 값을 비교하는 것이 좋다.
- 다만, 박싱된 값이 다음 표에 나와 있는 값이라면 ==와 != 연산자로 내부의 값을 바로 비교할 수 있다.

타입	값의 범위
boolean	true, false
char	\u0000 ~ \u007f
byte, short, int	-128 ~ 127

[ValueCompareExample.java] 포장 객체 비교

```

01 package sec12.exam01_wrapper;
02
03 public class ValueCompareExample {
04     public static void main(String[] args) {
05         System.out.println("[-128~127 초과값일 경우]");
06         Integer obj1 = 300;
07         Integer obj2 = 300;
08         System.out.println("==결과: " + (obj1 == obj2));
09         System.out.println("언박싱후 ==결과: " + (obj1.intValue() == obj2.intValue()));
10         System.out.println("equals() 결과: " + obj1.equals(obj2));
11         System.out.println();
12
13         System.out.println("[-128~127 범위값일 경우]");
14         Integer obj3 = 10;
15         Integer obj4 = 10;
16         System.out.println("==결과: " + (obj3 == obj4));
17         System.out.println("언박싱후 ==결과: " + (obj3.intValue() == obj4.intValue()));
18         System.out.println("equals() 결과: " + obj3.equals(obj4));
19     }
20 }
    
```


11.13 Math, Random 클래스

11.13.1 Math 클래스

- 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

11.13.2 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)

11.14 Date, Calendar 클래스

11.14.1 Date 클래스

- Date는 날짜를 표현하는 클래스이다. 여러 개의 생성자가 선언되어 있지만 대부분 Deprecated(비권장)되어 현재는 Date() 생성자만 주로 사용된다.

[DateExample.java] 현재 날짜를 출력하기

```
package sec14.exam01_date;

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

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);
    }
}
```

11.4.2 Calendar 클래스

- Calendar 클래스는 달력을 표현한 클래스이다. Calendar 클래스는 추상 클래스이므로 new 연산자를 사용해서 인스턴스를 생성할 수 없다.

[DateTime01.java] java.util.Calendar 활용

```
import java.util.Calendar;

public class DateTime01 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        // 3. Calendar
        // Calendar c = new Calendar();
        Calendar c = Calendar.getInstance();

        // GregorianCalendar c1 = new GregorianCalendar();
        // Calendar c2 = new GregorianCalendar();// 업캐스팅

        int y = c.get(Calendar.YEAR);
        int m = c.get(Calendar.MONTH) + 1; // 0~11
        int d = c.get(Calendar.DATE);

        int ap = c.get(Calendar.AM_PM); // 오전(0), 오후(1)
        int h1 = c.get(Calendar.HOUR); // 12시간제
        int h2 = c.get(Calendar.HOUR_OF_DAY); // 24시간제
        int mm = c.get(Calendar.MINUTE);
        int s = c.get(Calendar.SECOND);

        System.out.println(y + "년" + m + "월" + d + "일");
        if (ap == 0) {
            System.out.print("오전");
        } else {
            System.out.print("오후");
        }
        System.out.println(h1 + "시");
        System.out.println(h2 + ":" + mm + ":" + s);
    }
}
```

[꿀팁] java.sql.Timestamp

[DateTime.java] java.sql.Timestamp 활용

```
import java.sql.Timestamp;
import java.text.SimpleDateFormat;
import java.util.Date;

public class DateTime {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        // 1. Date

        SimpleDateFormat sd = new SimpleDateFormat("yyyy년MM월dd일 HH:mm:ss EEE요일");

        Date d = new Date();
        System.out.println("Date0=" + d);
        System.out.println("Date1=" + sd.format(d));

        // 2. Timestamp
        SimpleDateFormat sd1 = new SimpleDateFormat("yyyy년 MM월 dd일");
        Timestamp ts = new Timestamp(System.currentTimeMillis());
        System.out.println("Time0=" + ts);
        System.out.println("Time1=" + sd.format(ts));
        System.out.println("Time2=" + sd1.format(ts));
    }
}
```

[과제] GregorianCalendar 클래스 활용

키보드를 통해서 연도를 입력 받아 입력 받은 연도가 윤년인지 평년인지를 판별하는 프로그램을 작성하라.

[YearCheck.java] 윤년 혹은 평년인지 판별

```
01 package verify;
02
03 import java.util.*;
04
05 public class YearCheck {
06
07     public static void main(String[] args) {
08         System.out.print("연도를 입력 하세요?");
09         Scanner sc = new Scanner(System.in);
10         int year = sc.nextInt();
11
12         /*         if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0) {
13                     System.out.println(year + "은 윤년");
14                 } else {
15                     System.out.println(year + "은 평년");
16                 }*/
17
18         ....
19     }
20 }
```

11.15 Format 클래스

11.15.1 숫자 형식 클래스(DecimalFormat)

- 적용할 패턴 선택해 생성자 매개값으로 지정해서 객체를 생성한다.

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

[DecimalFormatExample.java] 숫자를 원하는 형식으로 출력

```
01 package sec15.exam01_format;
02
03 import java.text.DecimalFormat;
04
05 public class DecimalFormatExample {
06     public static void main(String[] args) {
07         double num = 1234567.89;
08
09         DecimalFormat df = new DecimalFormat("0");
10         System.out.println( df.format(num) );
11
12         df = new DecimalFormat("0.0");
13         System.out.println( df.format(num) );
14
15         df = new DecimalFormat("0000000000.00000");
16         System.out.println( df.format(num) );
17
18         df = new DecimalFormat("#");
19         System.out.println( df.format(num) );
20
21         df = new DecimalFormat("#.#");
22         System.out.println( df.format(num) );
23
24         df = new DecimalFormat("#####.#####");
25         System.out.println( df.format(num) );
26
27         df = new DecimalFormat("#.0");
28         System.out.println( df.format(num) );
29
30         df = new DecimalFormat("+#.0");
31         System.out.println( df.format(num) );
32
33         df = new DecimalFormat("-#.0");
34         System.out.println( df.format(num) );
35
36         df = new DecimalFormat("#,###.0");
37         System.out.println( df.format(num) );
38
39         df = new DecimalFormat("0.0E0");
40         System.out.println( df.format(num) );
41
42         df = new DecimalFormat("+#,### ; -#,###");
43         System.out.println( df.format(num) );
44
45         df = new DecimalFormat("#.# %");
46         System.out.println( df.format(num) );
47
48         df = new DecimalFormat("\u00A4 #,###");
49         System.out.println( df.format(num) );
```

```
50     }
51 }
```

11.15.2 날짜 형식 클래스(SimpleDateFormat)

- Date 클래스의 toString() 메소드는 영문으로된 날짜를 리턴하는데 만약 특정 문자열 포맷으로 얻고 싶다면 java.text.SimpleDateFormat 클래스를 이용하면 된다.

```
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		

11.15.3 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);
```

11.16 java.time 패키지

- 자바 7 이전까지는 Date와 Calendar 클래스를 이용해서 날짜와 시간 정보를 얻을 수 있었다.
 - Date 클래스의 대부분의 메소드는 Deprecated 되었고, Date의 용도는 단순히 특정 시점의 날짜 정보를 저장하는 역할만 한다.

11.16.1 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

11.16.2 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

11.16.3 날짜와 시간을 조작하기

(1) 빼기와 더하기

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

(2) 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)

11.16.4 날짜와 시간을 비교하기

클래스	리턴타입	메소드(매개변수)	설명
LocalDate LocalDateTime	boolean	isAfter(ChronoLocalDate other) isBefore(ChronoLocalDate other) isEqual(ChronoLocalDate other)	이후 날짜인지? 이전 날짜인지? 동일 날짜인지?
LocalTime LocalDateTime	boolean	isAfter(LocalTime other) isBefore(LocalTime other)	이후 시간인지? 이전 시간인지?
LocalDate	Period	until(ChronoLocalDate endDateExclusive)	날짜 차이
LocalDate LocalTime LocalDateTime	long	until(Temporal endDateExclusive, 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

11.16.5 파싱과 포매팅

(1) 파싱(Parsing) 메소드

상황에 맞는 포맷 변환 같이 사용

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

(2) 포매팅(Formatting) 메소드

날짜와 시간을 포매팅된 문자열로 변환

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

```

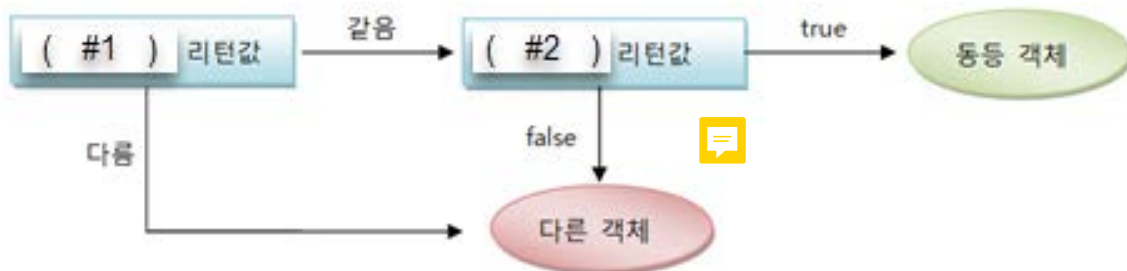
LocalDateTime now = LocalDateTime.now();
DateTimeFormatter dateTimeFormatter =
    DateTimeFormatter.ofPattern("yyyy년 M월 d일 a h시 m분");
String nowString = now.format(dateTimeFormatter);
    
```

[과제] 확인문제

1. Object 클래스에 대한 설명 중 틀린 것은 무엇입니까?

- (1) 모든 자바 클래스의 최상위 부모 클래스이다.
- (2) Object의 equals() 메소드는 == 연산자와 동일하게 번지를 비교한다.
- (3) Object의 clone() 메소드는 얇은 복사를 한다.
- (4) Object의 toString() 메소드는 객체의 필드값을 문자열로 리턴한다.

2. 여러분이 작성하는 클래스를 동등 비교하는 컬렉션 객체인 HashSet, HashMap, Hashtable을 사용하려고 합니다. Object의 equals()와 hashCode() 메소드를 오버라이딩했다고 가정할 경우, 메소드 호출 순서를 생각하고 다음 괄호() 안을 채워보세요.



3. Student 클래스를 작성하되, Object의 equals()와 hashCode()를 오버라이딩해서 Student의 학번(studentNum)이 같으면 동등 객체가 될 수 있도록 해보세요. Student 클래스의 필드는 다음과 같습니다. hashCode()의 리턴값은 studentNum 필드값의 해시코드를 리턴하도록 하세요.

[Student.java]

```

01 package verify.exam03;
02
03 public class Student {
04     private String studentNum;
05
06     public Student(String studentNum) {
07         this.studentNum = studentNum;
08     }
09
10     public String getStudentNum() {
11         return studentNum;
12     }
13
14     // 작성 위치
15
16 }
    
```

[StudentExample.java]

```

01 package verify.exam03;
02
03 import java.util.HashMap;
04
05 public class StudentExample {
06     public static void main(String[] args) {
07         //Student 키로 총점을 저장하는 HashMap 객체 생성
08         HashMap<Student, String> hashMap = new HashMap<Student, String>();
09
10         //new Student("1")의 점수 95를 저장
11         hashMap.put(new Student("1"), "95");
    
```



```

12
13          //new Student("1") 로 점수를 읽어옴
14          String score = hashMap.get(new Student("1"));
15          System.out.println("1번 학생의 총점: " + score);
16      }
17  }
18
19  // 실행 결과
20  // 1번 학생의 총점: 95

```

4. Member 클래스를 작성하되, Object 의 toString() 메소드를 오버라이딩해서 MemberExample 클래스의 실행 결과처럼 나오도록 작성해보세요.

[Member.java]

```

01  package verify.exam04;
02
03  public class Member {
04      private String id;
05      private String name;
06
07      public Member(String id, String name) {
08          this.id = id;
09          this.name = name;
10      }
11
12      // 작성 위치
13
14  }

```

[MemberExample.java]

```

01  package verify.exam04;
02
03  public class MemberExample {
04      public static void main(String[] args) {
05          Member member = new Member("blue", "이파란");
06          System.out.println(member);
07      }
08  }
09
10  // 실행 결과
11  // blue: 이파란

```

5. Class 객체에 대한 설명 중 틀린 것은 무엇입니까?

- (1) Class.forName() 메소드 또는 객체의 getClass() 메소드로 얻을 수 있다.
- (2) 클래스의 생성자, 필드, 메소드에 대한 정보를 알아낼 수 있다.
- (3) newInstance() 메소드는 기본 생성자를 이용해서 객체를 생성시킨다.
- (4) newInstance()의 리턴 타입은 생성된 객체의 클래스 타입이다.

6. 다음에 주어진 바이트 배열을 문자열로 변환시켜보세요.

```
{ 73, 32, 108, 111, 118, 101, 32, 121, 111, 117 };
```

[BytesToStringExample.java]

```
01 package verify.exam06;
02
03 public class BytesToStringExample {
04     public static void main(String[] args) {
05         byte[] bytes = { 73, 32, 108, 111, 118, 101, 32, 121, 111, 117 };
06         String str = ( #1 )
07         System.out.println( str ); // I love you
08     }
09 }
```

7. 다음 문자열에서 "자바" 문자열이 포함되어 있는지 확인하고, "자바"를 Java 로 대치한 새로운 문자열을 만들어보세요.

"모든 프로그램은 자바 언어로 개발될 수 있다.";

[FindAndReplaceExample.java]

```
01 package verify.exam07;
02
03 public class FindAndReplaceExample {
04     public static void main(String[] args) {
05         String str = "모든 프로그램은 자바 언어로 개발될 수 있다.";
06         int index = ( #1 )
07         if(index == -1) {
08             System.out.println("자바 문자열이 포함되어 있지 않습니다.");
09         } else {
10             System.out.println("자바 문자열이 포함되어 있습니다.");
11             str = ( #2 )
12             System.out.println("-->" + str);
13         }
14     }
15 }
16
17 // 실행 결과
18 // 자바 문자열이 포함되어 있습니다.
19 // -->모든 프로그램은 Java 언어로 개발될 수 있다.
```

8. 다음 문자열에서 쉼표(,)로 구분되어 있는 문자열을 String 의 split() 메소드 또는 StringTokenizer 를 이용해서 분리해보세요.

아이디,이름,패스워드

[SplitExample.java]

```
01 package verify.exam08;
02
03 import java.util.StringTokenizer;
04
```

```

05 public class SplitExample {
06     public static void main(String[] args) {
07         String str = "아이디,이름,패스워드";
08
09         // 작성 위치
10         // #1: split() 메소드 이용
11
12         System.out.println();
13
14         // #2: StringTokenizer 이용
15
16     }
17 }
18
19 // 실행 결과
20 // 아이디
21 // 이름
22 // 패스워드
23 //
24 // 아이디
25 // 이름
26 // 패스워드

```

9. 다음 코드는 1 부터 100 까지의 숫자를 통 문자열로 만들기 위해서 += 연산자를 이용해서 100 번 반복하고 있습니다. 이것은 곧 100 개 이상의 String 객체를 생성하는 결과를 만들기 때문에 좋은 코드라고 볼 수 없습니다. StringBuilder 를 사용해서 좀 더 효율적인 코드로 개선시켜 보세요.

[StringBuilderExample.java]

```

01 package verify.exam09;
02
03 public class StringBuilderExample {
04     public static void main(String[] args) {
05         String str = "";
06         for(int i=1; i<=100; i++) {
07             str += i;
08         }
09         System.out.println(str);
10
11         // 작성 위치
12
13     }
14 }
15
16 // 실행 결과
17 // 12345678910111213141516171819202122232....100
18 // 12345678910111213141516171819202122232....100

```

10. 첫 번째는 알파벳으로 시작하고 두 번째부터 숫자와 알파벳으로 구성된 8자~12 자 사이의 ID 값인지 검사하고 싶습니다. 일파벳은 대소문자를 모두 허용할 경우에 정규 표현식을 이용해서 검증하는 코드를 작성해보세요.

[SplitExample.java]

```

01 package verify.exam10;
02

```

```
03 import java.util.regex.Pattern;
04
05 public class PatternMatcherExample {
06     public static void main(String[] args) {
07         String id = "5Angel1004";
08         //String regExp = "[a-zA-Z][a-zA-Z0-9]{7,11}";
09         String regExp = (          #1          )
10         boolean isMatch = Pattern.matches(regExp, id);
11         if(isMatch) {
12             System.out.println("ID로 사용할 수 있습니다.");
13         } else {
14             System.out.println("ID로 사용할 수 없습니다.");
15         }
16     }
17 }
```