

# 기본 API

## String 클래스

문자열 값을 수정 못하는 immutable(불변)이다.

수정시 수정된 문자열이 새로 할당되어 새 주소를 넘긴다.

## StringBuffer 클래스

문자열 값을 수정할 수 있는 mutable(가변)이다.

기존 문자열에 수정되어 적용된다.(수정, 삭제 등)

기본 16문자 크기로 지정된 버퍼를 이용하며, 크기를 증가시킬 수 있다.

쓰레드 safe 기능을 제공한다.(성능저하요인)

## StringBuilder 클래스

String Buffer와 동일함

쓰레드 safe기능을 제공하지 않음

# StringTokenizer 클래스

String 클래스에서 제공하는 split() 메소드와 같은 기능을 하는 클래스이다.  
생성시 전달받은 문자열을 구분자로 나누어 각 토큰에 저장한다.

```
import java.util.*;

public class TestStringTokenizer{
    public static void main(String[] args){
        String str = "AA|BB|CC"

        StringTokenizer st = new StringTokenizer(str, "|");

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

# Wrapper 클래스


기본 자료형을 객체화 해주는 클래스이다.


기본 자료형	Wrapper Class	기본형으로 래퍼클래스 생성
boolean	Boolean	<code>Boolean bool = new Boolean(true);</code>
byte	Byte	<code>Byte b = new Byte((byte)1);</code>
char	Character	<code>Character c = new Character('A');</code>
short	Short	<code>Short s = new Short((short)2);</code>
int	Integer	<code>Integer age = new Integer(30);</code>
long	Long	<code>Long l = new Long(100L);</code>
float	Float	<code>Float f = new Float(0.7f);</code>
double	Double	<code>Double d = new Double(0.75);</code>

# Wrapper 클래스 형변환

## - 오토박싱(AutoBoxing)


기본자료형 → Wrapper클래스 변환


 Integer num = new Integer(3);  
Integer num = 3;

 Double dnum = new Double(3.14);  
Double dnum = 3.14;

## - 오토언박싱(Auto Unboxing)

Wrapper클래스 → 기본자료형 변환

 int n = num.intValue();  
int n = num;

 Double d = dnum.doubleValue();  
Double d = dnum;

# String을 기본 자료형으로 바꾸기

```
byte b = Byte.parseByte("1");  
short s = Short.parseShort("2");  
int i = Integer.parseInt("3");  
long l = Long.parseLong("4");  
float f = Float.parseFloat("0.1");  
double d = Double.parseDouble("0.2");  
boolean b = Boolean.parseBoolean("true");  
  
char c = "abc".charAt(0);
```

# 기본자료형을 String으로 바꾸기

```
String b = Byte.valueOf((byte)1).toString();  
String s = Short.valueOf((short)2).toString();  
String i = Integer.valueOf(3).toString();  
String l = Long.valueOf(4L).toString();  
String f = Float.valueOf(0.1f).toString();  
String d = Double.valueOf(0.2).toString();  
String b = Boolean.valueOf(true).toString();  
String c = Character.valueOf('a').toString();
```

이밖에도 `String.valueOf(1234)`, `""+1234` 등 여러가지 방법이 있다.

## java.util.Calendar

Calendar 클래스는 추상클래스로써, 생성자가 protected이기 때문에 new를 통해 객체 생성을 하지 못하고, getInstance() 메소드를 통해 객체를 생성한다. GregorianCalendar는 Calendar 클래스의 자식 클래스이며, 년, 월, 일, 시, 분, 초 정보를 필드를 이용하여 다룰 수 있다.

Calendar의 필드 Month(월)은 값이 0부터 시작하므로 주의해야 한다. 월을 출력시 반드시 1을 더해줘야 한다.

0~11  
= 1월 ~ 12월  
0 = 1월



# Calendar / GregorianCalendar

```
Calendar today = Calendar.getInstance();  
//생성자가 protected 임. new 에 사용 못함.  
GregorianCalendar today = new GregorianCalendar();  
//Calendar 의 후손(파생) 클래스임.
```

\* 년, 월, 일, 시, 분, 초 정보를 필드(멤버변수)를 이용하여, 각각 다룰 수 있음.  
예)

```
int year = today.get(Calendar.YEAR);  
int month = today.get(Calendar.MONTH) + 1; // 0부터 시작 1더함  
int date = today.get(Calendar.DATE);  
int ampm = today.get(Calendar.AM_PM);  
int hour = today.get(Calendar.HOUR);  
int min = today.get(Calendar.MINUTE);  
int sec = today.get(Calendar.SECOND);
```

```
String sAmPm = (ampm == Calendar.AM) ? "오전" : "오후";
```

```
System.out.printf("%d년 %d월 %d일 %s %d시 %d분 %d초",  
                  year, month, date, sAmPm, hour, min, sec);
```

## java.util.Date

시스템으로부터 현재 날짜, 시간 정보를 가져와서 다룰 수 있게 만들어진 클래스로, 생성자 2개만 사용 가능하고 나머지는 deprecated이다.

Calendar 클래스 혹은 GregorianCalendar 클래스 사용 권장

## 사용예시

```
Date today = new Date();
```

//시스템으로부터 현재 날짜, 시간 정보를 가져와  
기본값으로 사용

```
Date today = new Date(123456789L);
```

//long형 정수값을 가지고 날짜 시간 계산함(밀리초)

//1970년 1월 1일 09시 00분 00초를 기준으로 함.

```
//Date today = new Date(Calendar.getTimeMillis());
```

## java.util.SimpleDateFormat

Date의 날짜, 시간 정보를 원하는 format으로 출력하도록 기능 제공을 하는 클래스이다. Java.text 패키지에 속해 있다.

### 사용예시

```
Date today = new Date();
```

```
SimpleDateFormat ft  
    = new SimpleDateFormat("yyyy-MM-dd");
```

```
String ftToday = ft.format(today);  
//today에 포맷을 적용한 결과를 문자열로 리턴
```

# SimpleDateFormat 패턴

기호	의미	보기
G	연대(BC, AD)	AD
y	년도	2009
M	월 (1~12월 또는 1월~12월)	10또는 10월, OCT
w	년의 몇 번째 주(1~53)	50
W	월의 몇 번째 주(1~5)	4
D	년의 몇 번째 일(1~366)	100
d	월의 몇 번째 일(1~31)	15
F	월의 몇 번째 요일(1~5)	1
E	요일	월
a	오전/오후(AM, PM)	PM
H	시간(0~23)	20
k	시간(1~24)	12
K	시간(0~11)	10
h	시간(1~12)	11
m	분(0~59)	35
s	초(0~59)	55
S	천분의 1초(0~999)	253
z	Time zone(General time zone)	GMT+9:00
Z	Time zone(RFC 822 time zone)	+0900
'	escape문자(특수문자를 표현하는데 사용)	없음

## java.util.Formatter(final class)

값 출력시 format을 적용해서 출력할 수 있다.

Formatter 객체를 생성시 변환된 결과를 보낼 곳의 정보를 생성자 인자로 전달한다.

## Formatter객체 생성 및 출력

```
Formatter f = new Formatter(System.out);  
f.format("%s, %d, %d \n", "String", 10, 20);
```

//유사한 결과 만들기.

```
System.out.printf("%s, %d, %d □n", "문자열", 10, 20);  
System.out.println(String.format("%s, %d, %d □n", "문자열", 10, 20));
```

# Format클래스 - 변환문자

변환문자	의미
%d	정수(십진수)
%c	유니코드문자
%b	boolean값
%s	String
%f	부동소수점(십진수)
%e	부동소수점(과학적 표기)
%x	정수(16진수)
%h	해시코드(16진수)
%%	리터럴'%'

# Format클래스 - escape문자

특수문자	문자 리터럴	비 고
tab	\t	정해진 공간만큼 띄어쓰기
new line	\n	출력하고 다음라인으로 옮김
역슬래쉬	\\	특수문자 사용시 백슬러시(\)를 넣고 특수문자를 넣어야 함
작은따옴표	'\''	
큰따옴표	'\"'	
유니코드	\u	유니코드 표시할때 사용