String의 특징과 객체 생성

- String - java.lang.String
  - String 클래스는 하나의 문자열 표현

```
// 스트링 리터럴로 스트링 객체 생성
String str1 = "abcd";

// String 클래스의 생성자를 이용하여 스트링 생성
char data[] = {'a', 'b', 'c', 'd'};
String str2 = new String(data);
String str3 = new String("abcd"); // str2와 str3은 모두 "abcd" 스트링
```

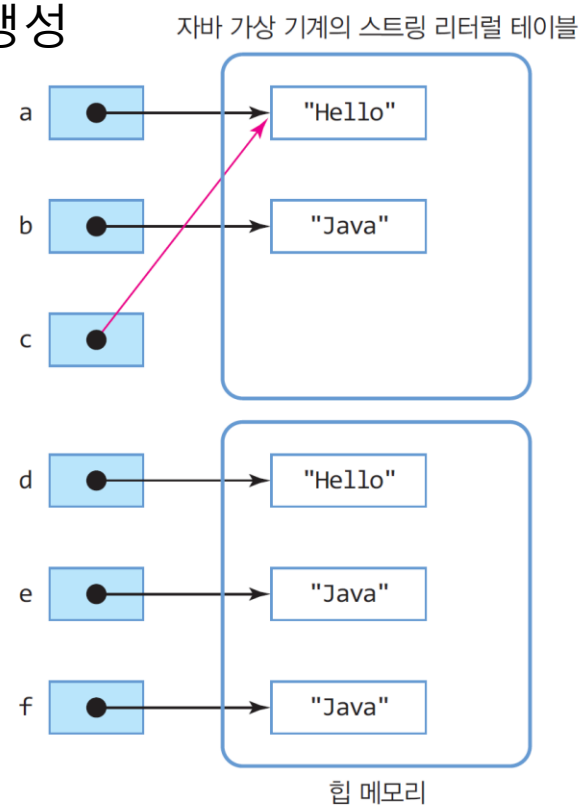
  - String 생성자

생성자	설명
String()	빈 스트링 객체 생성
String(char[] value)	char 배열에 있는 문자들을 스트링 객체로 생성
String(String original)	매개변수로 주어진 문자열과 동일한 스트링 객체 생성
String(StringBuffer buffer)	매개변수로 주어진 스트링 버퍼의 문자열을 스트링 객체로 생성

스트링 리터럴과 new String()

- 스트링 생성 방법
  - 리터럴로 생성, String s = "Hello";
    - JVM이 리터럴 관리, 응용프로그램 내에서 공유됨
  - String 객체로 생성, String t = new String("Hello");
    - 힙 메모리에 String 객체 생성

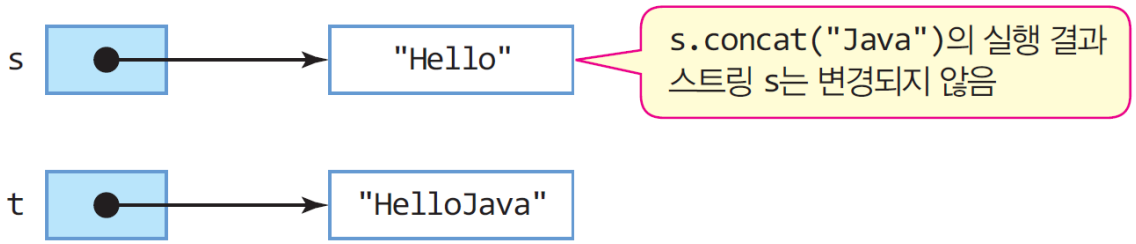
```
String a = "Hello";
String b = "Java";
String c = "Hello";
String d = new String("Hello");
String e = new String("Java");
String f = new String("Java");
```



스트링 객체의 주요 특징

- 스트링 객체는 수정 불가능

```
String s = new String("Hello");
String t = s.concat("Java"); // 스트링 s에 "Java"를 덧붙인 새로운 스트링 객체 리턴
```



- 스트링 비교 시 반드시 equals()를 사용
  - equals()는 내용을 비교하기 때문

주요 메소드

메소드	설명
char charAt(int index)	index 인덱스에 있는 문자 값 리턴
int codePointAt(int index)	index 인덱스에 있는 유니코드 값 리턴
int compareTo(String anotherString)	두 스트링을 사전 순으로 비교하여 두 스트링이 같으면 0, 현 스트링이 anotherString보다 먼저 나오면 음수, 아니면 양수 리턴
String concat(String str)	현재 스트링 뒤에 str 스트링을 덧붙인 새로운 스트링 리턴
boolean contains(CharSequence s)	s에 지정된 문자들을 포함하고 있으면 true 리턴
int length()	스트링의 길이(문자 개수) 리턴
String replace(Charsequence target, Charsequence replacement)	target이 지정하는 일련의 문자들을 replacement가 지정하는 문자들로 변경한 스트링 리턴
String[] split(String regex)	정규식 regex에 일치하는 부분을 중심으로 스트링을 분리하고, 분리된 스트링들을 배열로 저장하여 리턴
String subString(int beginIndex)	beginIndex 인덱스부터 시작하는 서브 스트링 리턴
String toLowerCase()	소문자로 변경한 스트링 리턴
String toUpperCase()	대문자로 변경한 스트링 리턴
String trim()	스트링 앞뒤의 공백 문자들을 제거한 스트링 리턴

문자열 비교

- int compareTo(String anotherString)
  - 문자열이 같으면 0 리턴
  - 이 문자열이 anotherString 보다 사전에 먼저 나오면 음수 리턴
  - 이 문자열이 anotherString 보다 사전에 나중에 나오면 양수 리턴

```
String java= "Java";
String cpp = "C++";
int res = java.compareTo(cpp);
if(res == 0)
   System.out.println("the same");
else if(res <0)
   System.out.println(java + " < " + cpp);
else
   System.out.println(java + " > " + cpp);
```

"Java" 가 "C++" 보다 사전에 나중에 나오기 때문에 양수 리턴

```
Java > C++
```

- ==는 문자열 비교에는 사용하면 안됨

문자열 연결

- + 연산자로 문자열 연결
  - 피연산자에 문자열이나 객체가 포함되어 있는 경우
    - 객체는 객체.toString()을 호출하여 문자열로 변환하여 연결
    - 기본 타입 값은 문자열로 변환하여 연결

```
System.out.print("abcd" + 1 + true + 3.13e-2 + 'E'+ "fgh" );
// abcd1true0.0313Efgh  출력
```

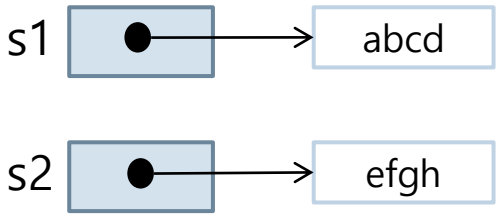
- String concat(String str)를 이용한 문자열 연결

```
"I love ".concat("Java.") 는  "I Love Java." 리턴
```

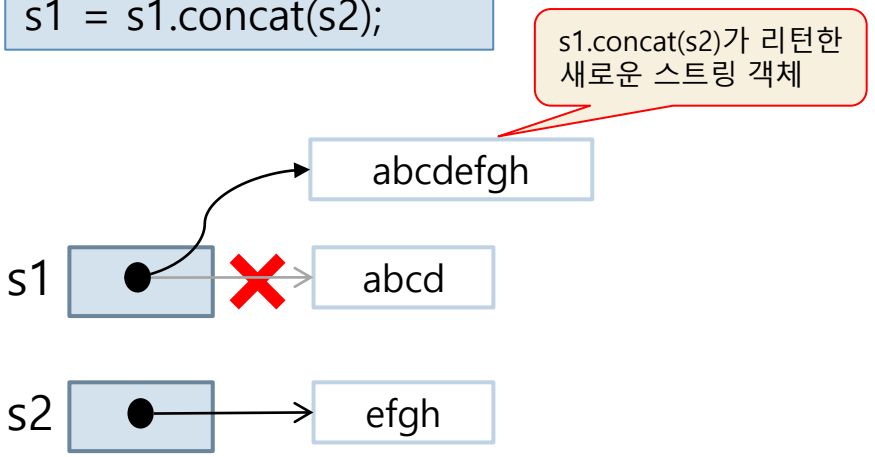
  - 기존 String 객체에 연결되지 않고 새로운 스트링 객체 리턴
    - 다음 슬라이드에서 설명

concat()은 새로운 문자열을 생성


```
String s1 = "abcd";
String s2 = "efgh";
```




```
s1 = s1.concat(s2);
```



문자열 내의 공백 제거, 문자열의 각 문자 접근

- 공백 제거
  - String trim()
    - 문자열 앞 뒤 공백 문자(tab, enter, space) 제거한 문자열 리턴

```
String a = "       abcd def       ";
String b = "       xyz₩t";
String c = a.trim(); // c = "abcd def". 문자열 중간에 있는 공백은 제거되지 않음
String d = b.trim(); // d = "xyz". 스페이스와 '₩t' 제거됨"
```

- 문자열의 문자
  - char charAt(int index)
    - 문자열 내의 문자 접근

```
String a = "class";
char c = a.charAt(2); // c = 'a'
```

```
// "class"에 포함된 's'의 개수를 세는 코드
int count = 0;
String a = "class";
for(int i=0; i<a.length(); i++) { // a.length()는 5
   if(a.charAt(i) == 's')
      count++;
}
System.out.println(count); // 2 출력
```

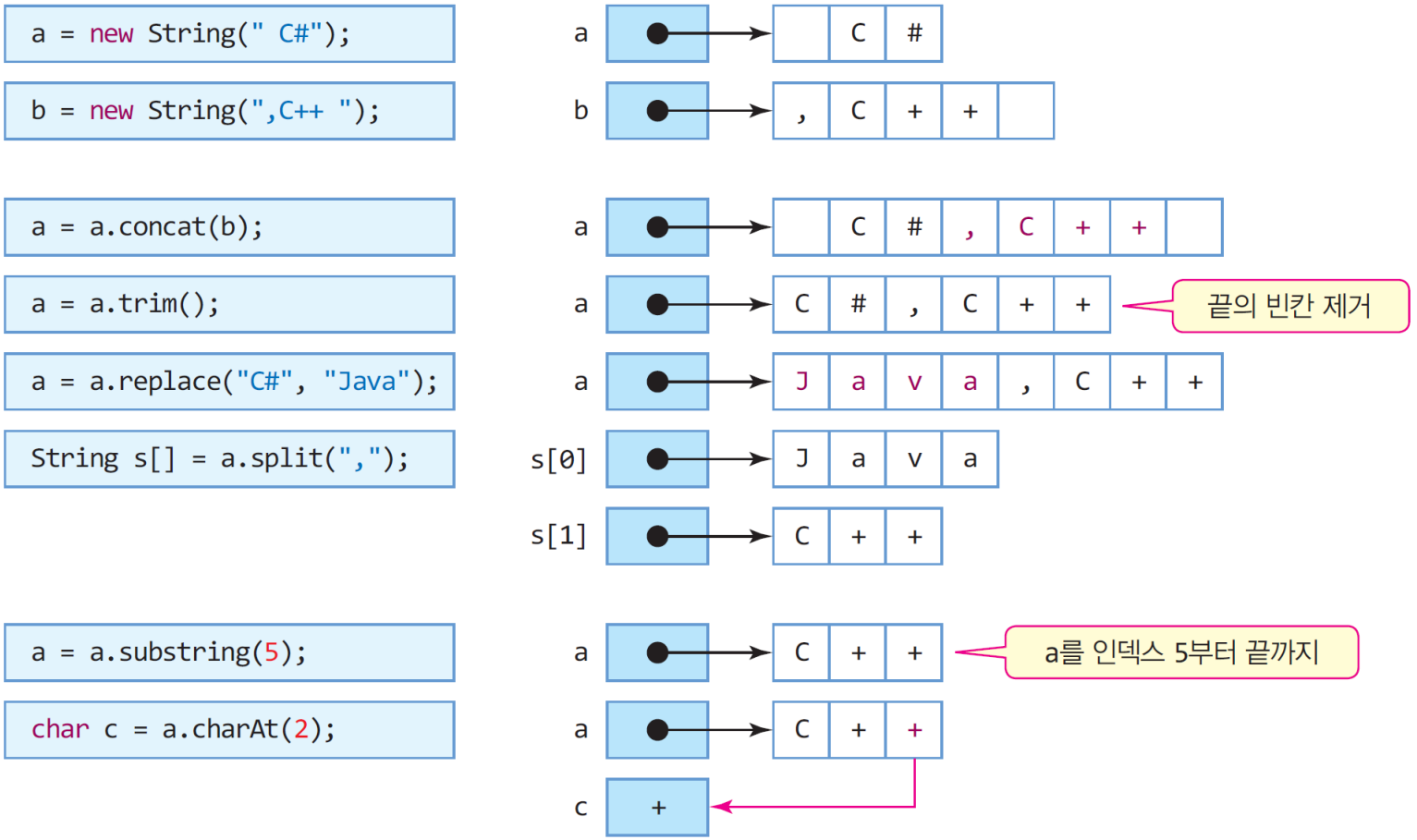
예제 6-7 : String 클래스 메소드 활용

String 클래스의 다양한 메소드를 활용하는 예를 보여라.

```
public class StringEx {
    public static void main(String[] args) {
        String a = new String(" C#");
        String b = new String(",C++ ");

        System.out.println(a + "의 길이는 " + a.length()); // 문자열의 길이(문자 개수)
        System.out.println(a.contains("#")); // 문자열의 포함 관계

        a = a.concat(b); // 문자열 연결
        System.out.println(a);

        a = a.trim(); // 문자열 앞 뒤의 공백 제거
        System.out.println(a);

        a = a.replace("C#","Java"); // 문자열 대치
        System.out.println(a);

        String s[] = a.split(","); // 문자열 분리
        for (int i=0; i<s.length; i++)
            System.out.println("분리된 문자열" + i + ": " + s[i]);

        a = a.substring(5); // 인덱스 5부터 끝까지 서브 스트링 리턴
        System.out.println(a);

        char c = a.charAt(2); // 인덱스 2의 문자 리턴
        System.out.println(c);
    }
}
```

```
 C#의 길이는 3
true
 C#,C++
C#,C++
Java,C++
분리된 문자열0: Java
분리된 문자열1: C++
C++
+
```

예제 실행 과정


a = new String(" C#");
b = new String(",C++ ");
a = a.concat(b);
a = a.trim();
a = a.replace("C#", "Java");
String s[] = a.split(",");
a = a.substring(5);
char c = a.charAt(2);
a
b
a
a
a
s[0]
s[1]
a
a
c
C #
, C + +
C # , C + +
C # , C + +
J a v a , C + +
J a v a
C + +
C + +
+
끝의 빈칸 제거
a를 인덱스 5부터 끝까지

StringBuffer 클래스

- 가변 크기의 문자열 저장 클래스
  - Java.lang.StringBuffer
  - String 클래스와 달리 문자열 변경 가능
  - StringBuffer 객체의 크기는 스트링 길이에 따라 가변적
- 생성

```
StringBuffer sb = new StringBuffer("java");
```

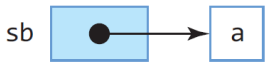
생성자	설명
StringBuffer()	초기 버퍼의 크기가 16인 스트링 버퍼 객체 생성
StringBuffer(charSequence seq)	seq가 지정하는 일련의 문자들을 포함하는 스트링 버퍼 생성
StringBuffer(int capacity)	지정된 초기 크기를 갖는 스트링 버퍼 객체 생성
StringBuffer(String str)	지정된 스트링으로 초기화된 스트링 버퍼 객체 생성

주요 메소드

메소드	설명
StringBuffer append(String str)	str 스트링을 스트링 버퍼에 덧붙인다.
StringBuffer append(StringBuffer sb)	sb 스트링 버퍼를 현재의 스트링 버퍼에 덧붙인다. 이 결과 현재 스트링 버퍼의 내용이 변한다.
int capacity()	스트링 버퍼의 현재 크기 리턴
StringBuffer delete(int start, int end)	start 위치에서 end 위치 앞까지 부분 문자열 삭제
StringBuffer insert(int offset, String str)	str 스트링을 스트링 버퍼의 offset 위치에 삽입
StringBuffer replace(int start, int end, String str)	스트링 버퍼 내의 start 위치의 문자부터 end가 지정하는 문자 앞의 서브 스트링을 str로 대치
StringBuffer reverse()	스트링 버퍼 내의 문자들을 반대 순서로 변경
void setLength(int newLength)	스트링 버퍼 내 문자열 길이를 newLength로 재설정, 현재 길이보다 큰 경우 널 문자('')로 채우며 작은 경우는 기존 문자열이 잘린다.

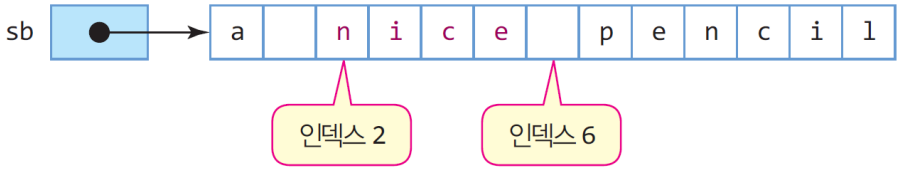
StringBuffer의 메소드 활용 예


```
StringBuffer sb = new StringBuffer("a");
```

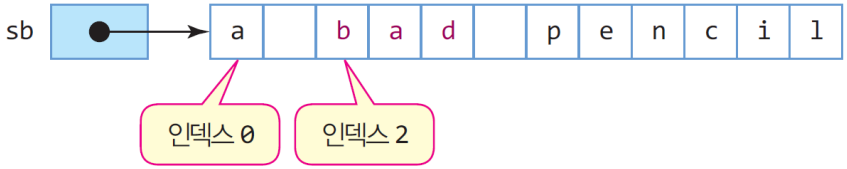



```
sb.append(" pencil");
```


```
sb.insert(2, "nice ");
```

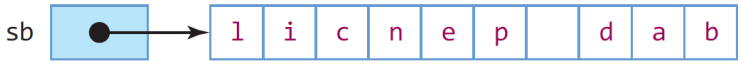



```
sb.replace(2, 6, "bad");
```

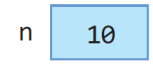



```
sb.delete(0, 2);
```

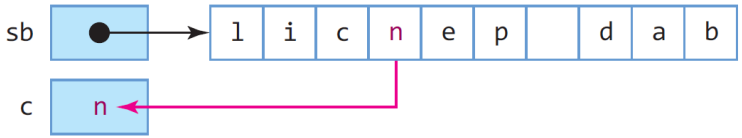

```
sb.reverse();
```




```
int n = sb.length();
```




```
char c = sb.charAt(3);
```




```
sb.setLength(3);
```

예제 6-8 : StringBuffer 클래스 메소드 활용

StringBuffer를 이용하여 문자열을 조작하는 다음 코드의 실행 결과는 무엇인가?

```
public class StringBufferEx {
   public static void main(String[] args) {
      StringBuffer sb = new StringBuffer("This");

      sb.append(" is pencil"); // 문자열 덧붙이기
      System.out.println(sb);

      sb.insert(7, " my"); // "my" 문자열 삽입
      System.out.println(sb);

      sb.replace(8, 10, "your"); // "my"를 "your"로 변경
      System.out.println(sb);

      sb.delete(8, 13); // "your " 삭제
      System.out.println(sb);

      sb.setLength(4); // 스트링 버퍼 내 문자열 길이 수정
      System.out.println(sb);
   }
}
```

sb.toString()으로 자동 바뀜

```
This is pencil
This is my pencil
This is your pencil
This is pencil
This
```

StringTokenizer 클래스

- java.util.StringTokenizer
  - 하나의 문자열을 여러 문자열 분리
    - 문자열을 분리할 때 사용되는 기준 문자 : 구분 문자(delimiter)
      - 다음 예에서 '&'가 구분 문자

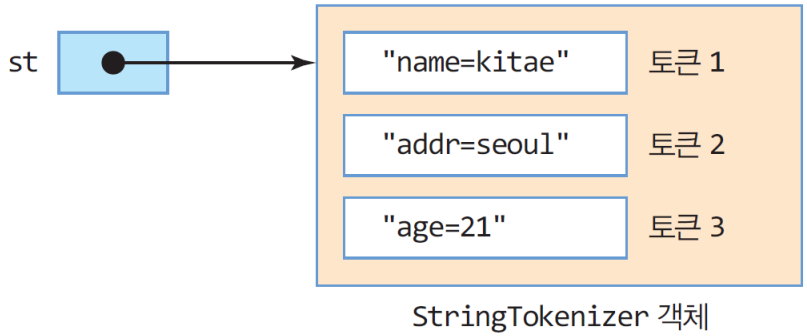
```
String query = "name=kitae&addr=seoul&age=21";
StringTokenizer st = new StringTokenizer(query, "&");
```

    - 토큰(token)
      - 구분 문자로 분리된 문자열
  - String 클래스의 split() 메소드를 이용하여 동일한 구현 가능

StringTokenizer 객체 생성과 문자열 분리

```
String query = "name=kitae&addr=seoul&age=21";
StringTokenizer st = new StringTokenizer(query, "&");
```

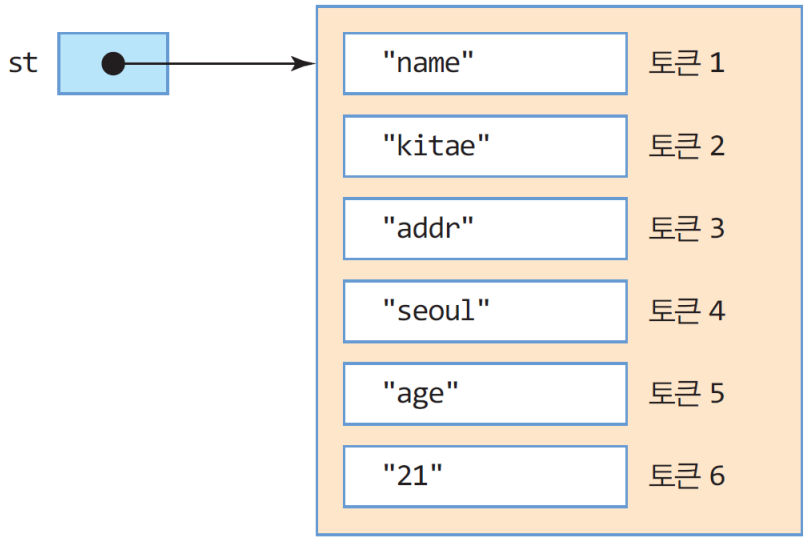
구분 문자 '&'



StringTokenizer 객체

```
StringTokenizer st = new StringTokenizer(query, "&=");
```

구분 문자 '&'와 '='



StringTokenizer 객체

생성자와 주요 메소드

생성자

생성자	설명
StringTokenizer(String str)	str 스트링의 각 문자를 구분 문자로 문자열을 분리하는 스트링 토크나이저 생성
StringTokenizer(String str, String delim)	str 스트링과 delim 구분 문자로 문자열을 분리하는 스트링 토크나이저 생성
StringTokenizer(String str, String delim, boolean returnDelims)	str 스트링과 delim 구분 문자로 문자열을 분리하는 스트링 토크나이저 생성. returnDelims가 true이면 delim이 포함된 문자도 토큰에 포함된다.

주요 메소드

메소드	설명
int countTokens()	스트링 토크나이저가 분리한 토큰의 개수 리턴
boolean hasMoreTokens()	스트링 토크나이저에 다음 토큰이 있으면 true 리턴
String nexToken()	스트링 토크나이저에 들어 있는 다음 토큰 리턴

예제 6-9 : StringTokenizer 클래스 메소드 활용

"홍길동/장화/홍련/콩쥐/팥쥐"문자열을'/'를 구분 문자로 하여 토큰을 분리하여 각 토큰을 출력하라.

```
import java.util.StringTokenizer;

public class StringTokenizerEx {
   public static void main(String[] args) {
      StringTokenizer st = new StringTokenizer("홍길동/장화/홍련/콩쥐/팥쥐", "/");
      while (st.hasMoreTokens())
         System.out.println(st.nextToken());
   }
}
```

```
홍길동
장화
홍련
콩쥐
팥쥐
```

Math 클래스

- 산술 연산 메소드 제공, java.lang.Math
  - 모든 메소드는 static 타입 : 클래스 이름으로 바로 호출해야 함

메소드	설명
static double abs(double a)	실수 a의 절댓값 리턴
static double cos(double a)	실수 a의 cosine 값 리턴
static double sin(double a)	실수 a의 sine 값 리턴
static double tan(double a)	실수 a의 tangent 값 리턴
static double exp(double a)	e^a 값 리턴
static double ceil(double a)	올림. 실수 a보다 크거나 같은 수 중에서 가장 작은 정수를 실수 타입으로 리턴
static double floor(double a)	내림. 실수 a보다 작거나 같은 수 중에서 가장 큰 정수를 실수 타입으로 리턴
static double max(double a, double b)	두 수 a, b 중에서 큰 수 리턴
static double min(double a, double b)	두 수 a, b 중에서 작은 수 리턴
static double random()	0.0보다 크거나 같고 1.0보다 작은 임의의 실수 리턴
static long round(double a)	반올림. 실수 a를 소수 첫째 자리에서 반올림한 정수를 long 타입으로 반환
static double sqrt(double a)	실수 a의 제곱근 리턴

Math 클래스를 활용한 난수 발생

- 난수 발생
  - static double random()
    - 0.0 이상 1.0 미만의 임의의 double 값을 반환
    - 0에서 100사이의 난수 10개 발생시키는 샘플 코드

```
for(int x=0; x<10; x++) {
   int n = (int)(Math.random()*100 + 1); // n은 [1~100] 사이의 랜덤 정수
   System.out.println(n);
}
```

    - *Math.random()*100은 0.0~99.99.. 사이의 실수 리턴*
    - *Math.random()*100+1은 1.0~100.99.. 사이의 실수 값*
    - *(int)(Math.random()*100 + 1)는 소수점이하를 제거하여 1~100 사이의 정수 값*

- java.util.Random 클래스
  - 다양한 형태로 난수 발생 가능

예제 6-10 : Math 클래스 메소드 활용

Math 클래스의 다양한 메소드 활용 예를 보여라.

```
public class MathEx {
   public static void main(String[] args) {
      System.out.println(Math.PI); // 원주율 상수 출력
      System.out.println(Math.ceil(a)); // ceil(올림)
      System.out.println(Math.floor(a)); // floor(내림)
      System.out.println(Math.sqrt(9)); // 제곱근
      System.out.println(Math.exp(2)); // e의 2승
      System.out.println(Math.round(3.14)); // 반올림

      // [1, 45] 사이의 정수형 난수 5개 발생
      System.out.print("이번주 행운의 번호는 ");
      for(int i=0; i<5; i++)
         System.out.print((int)(Math.random()*45 + 1) + " ");
      }
}
```

```
3.141592653589793
4.0
3.0
3.0
7.38905609893065
3
이번주 행운의 번호는 15 31 9 7 5
```

Calendar 클래스

- Calendar 클래스의 특징
  - java.util 패키지
  - 시간과 날짜 정보 저장 관리
    - 년, 월, 일, 요일, 시간, 분, 초, 밀리초, 오전 오후 등
    - Calendar 클래스의 각 시간 요소를 설정하기나 알아내기 위한 필드들

필드	의미	필드	의미
YEAR	년도	DAY_OF_MONTH	한 달의 날짜
MONTH	달(0~11)	DAY_OF_WEEK	한 주의 요일
HOUR	시간(0~11)	AM_PM	오전인지 오후인지 구분
HOUR_OF_DAY	24시간을 기준으로 한 시간	MINUTE	분
SECOND	초	MILLISECOND	밀리초

Calendar 객체 생성 및 날짜와 시간

- Calendar 객체 생성,
  - Calendar now = Calendar.getInstance();
    - now객체는 현재 날짜와 시간 정보를 가지고 생성
    - Calendar는 추상 클래스이므로 new Calendar() 하지 않음
- 날짜와 시간 알아내기

```
int year = now.get(Calendar.YEAR);              // now에 저장된 년도
int month = now.get(Calendar.MONTH) + 1; // now에 저장된 달
```

- 날짜와 시간 설정하기
  - Calendar 객체에 저장
    - Calendar 객체에 날짜와 시간을 저장한다고 컴퓨터의 날짜와 시간을 바꾸는 것은 아님

```
// 이성 친구와 처음으로 데이트한 날짜와 시간 저장
Calendar firstDate = Calendar.getInstance();

firstDate.clear(); // 현재 날짜와 시간 정보를 모두 지운다.
firstDate.set(2016, 11, 25); // 2016년 12월 25일. 12월은 11로 설정
firstDate.set(Calendar.HOUR_OF_DAY, 20); // 저녁 8시로 설정
firstDate.set(Calendar.MINUTE, 30); // 30분으로 설정
```

예제 6-11 : Calendar를 이용하여 현재 날짜와 시간 알아내기/날짜 시간 설정하기


```
import java.util.Calendar;
public class CalendarEx {
    public static void printCalendar(String msg, Calendar cal) {
        int year = cal.get(Calendar.YEAR);

        // get()은 0~30까지의 정수 리턴.
        int month = cal.get(Calendar.MONTH) + 1;
        int day = cal.get(Calendar.DAY_OF_MONTH);
        int dayOfWeek = cal.get(Calendar.DAY_OF_WEEK);
        int hour = cal.get(Calendar.HOUR);
        int hourOfDay = cal.get(Calendar.HOUR_OF_DAY);
        int ampm = cal.get(Calendar.AM_PM);
        int minute = cal.get(Calendar.MINUTE);
        int second = cal.get(Calendar.SECOND);
        int millisecond = cal.get(Calendar.MILLISECOND);

        System.out.print(msg + year + "/" + month + "/" + day + "/");
        switch(dayOfWeek) {
            case Calendar.SUNDAY : System.out.print("일요일"); break;
            case Calendar.MONDAY : System.out.print("월요일"); break;
            case Calendar.TUESDAY : System.out.print("화요일"); break;
            case Calendar.WEDNESDAY : System.out.print("수요일"); break;
            case Calendar.THURSDAY : System.out.print("목요일"); break;
            case Calendar.FRIDAY: System.out.print("금요일"); break;
            case Calendar.SATURDAY : System.out.print("토요일"); break;
        }
        System.out.print("(" + hourOfDay + "시)");
        if(ampm == Calendar.AM) System.out.print("오전");
        else System.out.print("오후");
        System.out.println(hour + "시 " + minute + "분 " + second + "초 "
            + millisecond +"밀리초");
    }
```

```
public static void main(String[] args) {
        Calendar now = Calendar.getInstance();
        printCalendar("현재 ", now);

        Calendar firstDate = Calendar.getInstance();
        firstDate.clear();
        // 2016년 12월 25일. 12월을 표현하기 위해 month에 11로 설정
        firstDate.set(2016, 11, 25);
        firstDate.set(Calendar.HOUR_OF_DAY, 20); // 저녁 8시
        firstDate.set(Calendar.MINUTE, 30); // 30분
        printCalendar("처음 데이트한 날은 ", firstDate);
    }
}
```

```
현재 2017/3/29/수요일(19시)오후7시 59분 51초 892밀리초
처음 데이트한 날은 2016/12/25/일요일(20시)오후8시 30분 0초 0밀리초
```