기본 API 클래스

컴퓨터공학전공
박요한

수업내용

- JDK에서 제공하는 기본 API 클래스
  - ✓Object 클래스
  - ✓Wrapper 클래스
  - ✓Math, Random 클래스
  - ✓String, StringBuffer, StringTokenizer 클래스
  - ✓Arrays 클래스

Object 클래스

- 주요 메소드

메소드	설명
boolean equals(Object obj)	obj가 가리키는 객체와 현재 객체를 비교하여 같으면 true 리턴
Class getClass()	현 객체의 런타임 클래스를 리턴
int hashCode()	현 객체에 대한 해시 코드 값 리턴
String toString()	현 객체에 대한 문자열 표현을 리턴
void notify()	현 객체에 대해 대기하고 있는 하나의 스레드를 깨운다.
void notifyAll()	현 객체에 대해 대기하고 있는 모든 스레드를 깨운다.
void wait()	다른 스레드가 깨울 때까지 현재 스레드를 대기하게 한다.

Wrapper 클래스

- 자바의 기본 타입을 클래스화 한 8개 클래스

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

✓ 이름이 Wrapper인 클래스는 존재하지 않음

✓ 용도

- 기본 자료형의 값을 감싸는 클래스 => 기본 타입의 값을 객체로 다룰 수 있게 함
- 자바는 객체 지향 언어, 객체만 다루는 클래스들이 존재함
- 따라서, **기본 타입의 값을 객체**로 만들 필요가 있음

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

Random 클래스

- Random 클래스
  - ✓java.util.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)

String Class

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
```

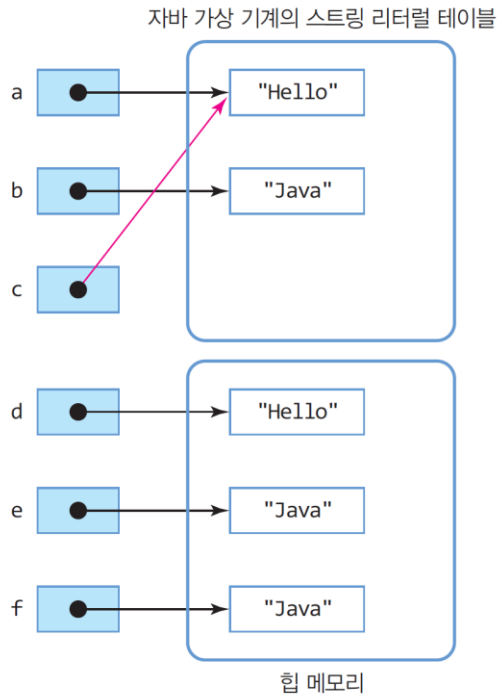
  - ✓String 생성자

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

- 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

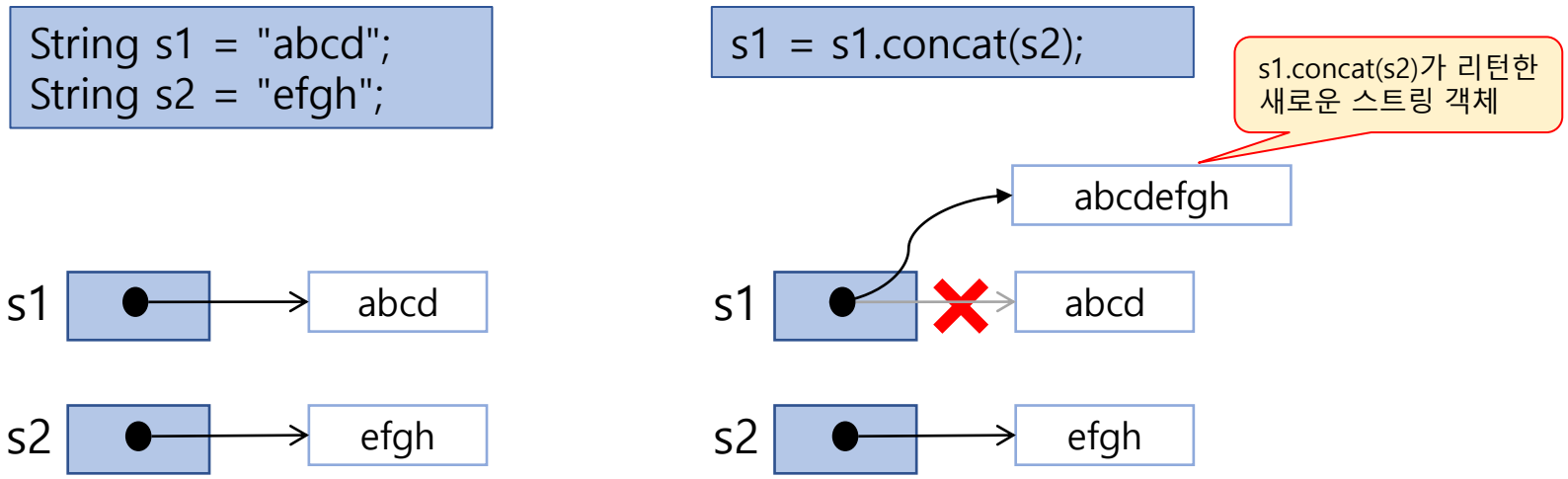
```
System.out.print("abcd" + 1 + true + 3.13e-2 + 'E'+ "fgh" );
// abcd1true0.0313Efgh   출력
```

- 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



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 = 's'
```

```
// "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
```

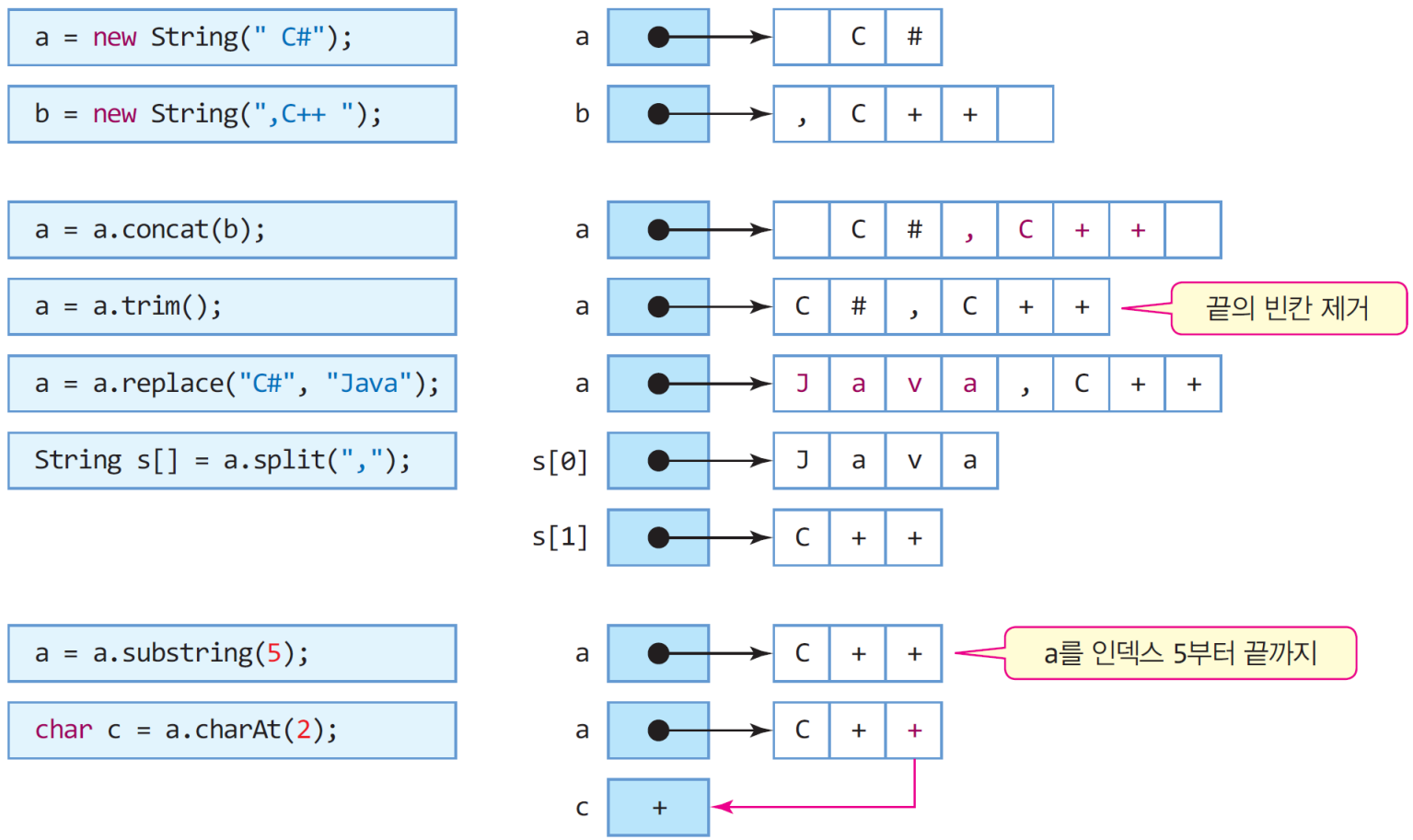
예제 6-7 : String 클래스 메소드 활용

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
s[0]
s[1]
c
끝의 빈칸 제거
a를 인덱스 5부터 끝까지

StringBuffer Class (StringBuilder)

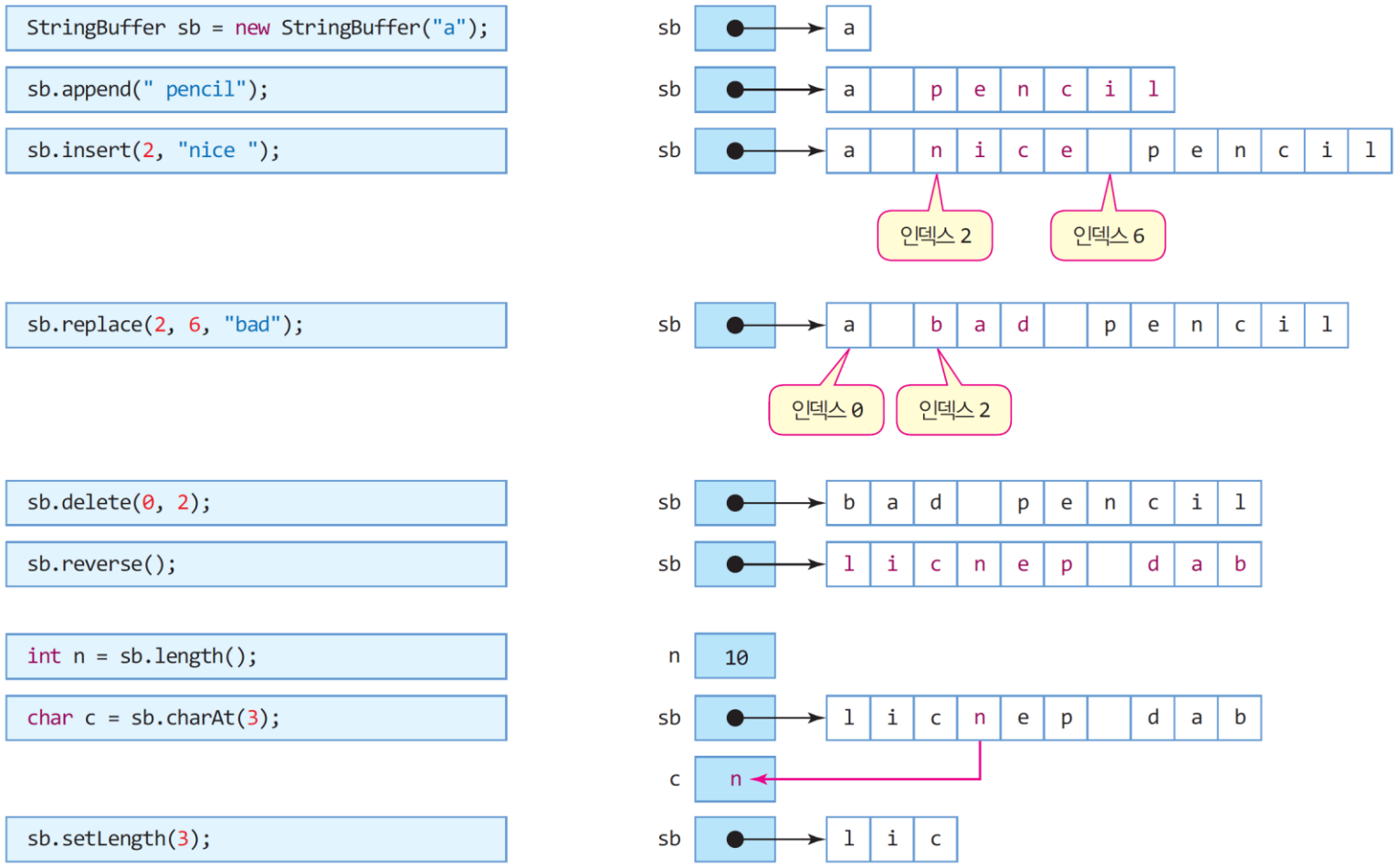
StringBuffer 클래스

- 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

StringBuffer의 메소드 활용 예



StringTokenizer Class

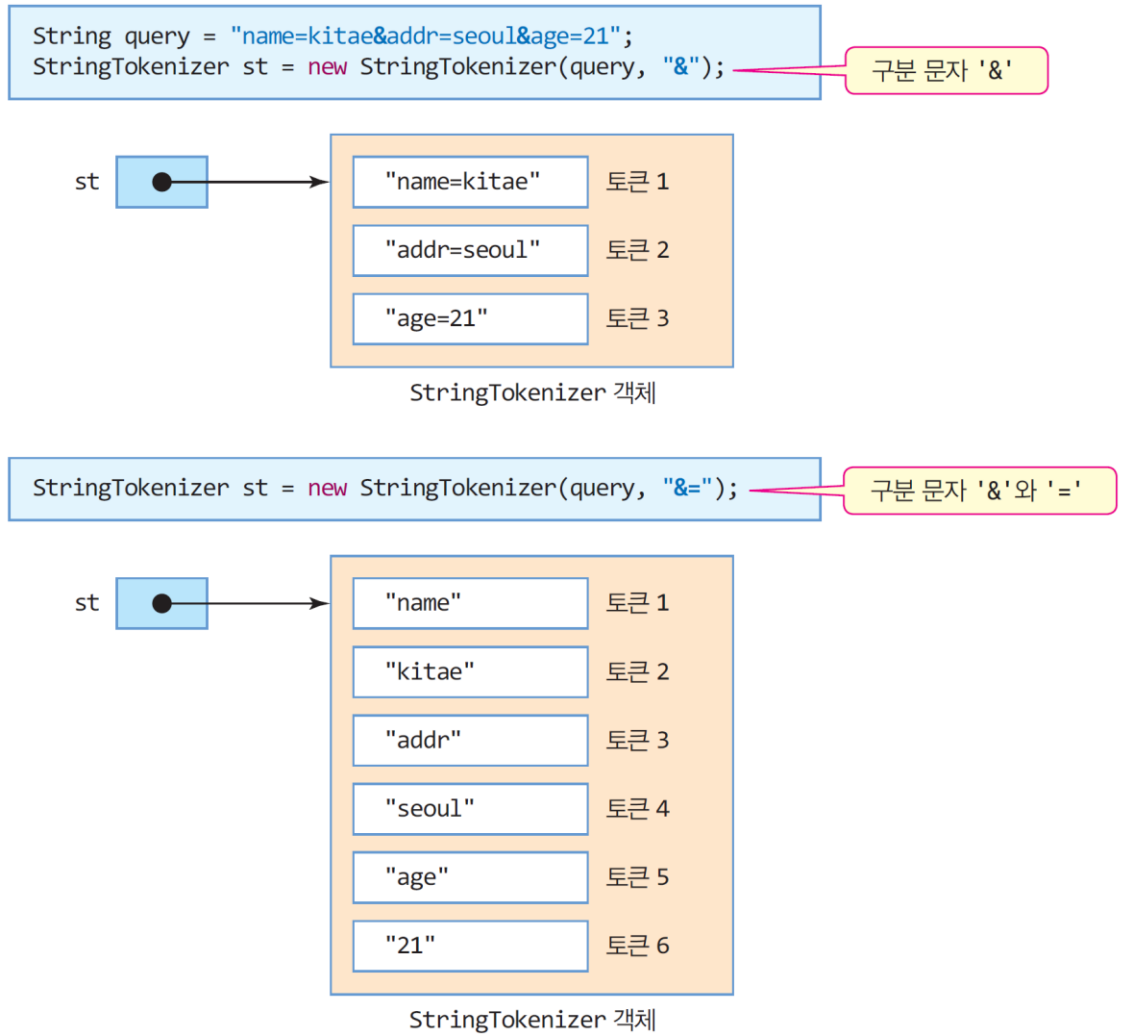
StringTokenizer 클래스

- 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



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
```

Arrays Class

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

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
```

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