String 클래스

메소드 / 설명	예제	결과
String(String s) 주어진 문자열(s)를 갖는 String 객체를 생성한다.	String s = new String("Hello");	s = "Hello"
String(char[] value) 주어진 문자열(value)을 갖는 String 객체를 생성한다.	char[] c = {'H', 'e', 'l', 'l', 'o'}; String s = new String(c);	s = "Hello"
String(StringBuffer buf) StringBuffer객체가 가지고 있는 문자열과 같은 내용의 String 객체를 생성한다.	StringBuffer sb = new StringBuffer("Hello"); String s = new String(sb);	s = "Hello"
charAt(int index) : char 지정된 위치(index)에 있는 문자를 알려준다.	String s = “Hello"; String n = "0123456" char c = s.charAt(1); char c2 = n.charAt(2);	c = 'e' c2 = '1'
compareTo(String str) : int 문자열(str)과 사전 순서로 비교한다. 같으면 0을, 사전순으로 이전이면 음수를, 이후면 양수를 반환한다.	int i1 = "aaa".compareTo("aaa"); int i2 = "aaa".compareTo("bbb"); int i3 = "bbb".compareTo("aaa");	i1 = 0 i2 = -1 i3 = 1
concat(String str) : String 문자열(str)을 뒤에 덧붙인다.	String s = "Hello"; String s2 = s.concat("World");	s2 = "HelloWorld"
contains(CharSequence s) : boolean 지정된 문자열(s)이 포함되었는지 검사한다.	String s = "abcdefg"; boolean b = s.contains("bc");	b = true
endsWith(String suffix) : boolean 지정된 문자열(suffix)로 끝나는지 검사한다.	String file = "Hello.txt"; boolean b = file.endsWith("txt");	b = true
startsWith(String prefix) : boolean 지정된 문자열(prefix)로 시작하는지 검사한다.	String s = "java.lang.Object"; boolean b = s.startsWith("java"); boolean b2 = s.stratsWith("lang");	b = true b2 = false
equals(Object obj) : boolean 매개변수로 받은 문자열(obj)과 String객체의 문자열을 비교한다. obj가 String이 아니거나 문자열이 다르면 false를 반환한다.	String s = "Hello"; boolean b = s.equals("Hello"); boolean b2 = s.equals("hellp");	b = true b2 = false
equalsIgnoreCase(String str) : boolean 문자열과 String 객체의 문자열을 대소문자 구분 없이 비교한다.	String s = "Hello"; boolean b = s.equalsIgnoreCase("Hello"); boolean b2 = s.equalsIgnoreCase("hello");	b = true b2 = true
indexOf(char ch) : int 주어진 문자(ch)가 문자열에 존재하는지 확인하여 위치(index)를 알려준다. 못 찾으면 -1을 반환한다.	String s = "Hello"; int idx1 = s.indexOf('o'); int idx2 = s.indexOf('k');	idx1 = 4 idx2 = -1
indexOf(char ch, int pos) : int	String s = "Hello";	idx1 = 1

주어진 문자(ch)가 문자열에 존재하는지 지정된 위치(pos)부터 찾으며 확인하여 위치를 알려준다. 못 찾으면 -1을 반환한다.	int idx1 = s.indexOf('e', 0); int idx1 = s.indexOf('e', 2);	idx2 = -1
indexOf(String str) : int 주어진 문자열이 존재하는지 확인하여 그 위치(index)를 알려준다. 없으면 -1을 반환한다.	String s = "ABCDEFG"; int idx = s.indexOf("CD");	idx = 2
intern() : String 문자열을 상수풀에 등록한다. 이미 상수풀에 같은 내용의 문자열이 있을 경우 그 문자열의 주소값을 반환한다.	String s = new String("abc"); String s2 = new String("abc"); boolean b = (s == s2); boolean b2 = s.equals(s2); boolean b3 = (s.intern() == s2.intern());	b = false b2 = true b3 = true
lastIndexOf(char ch) : int 지정된 문자 또는 문자코드를 문자열의 오른쪽 끝에서부터 찾아서 위치를 알려준다. 못 찾으면 -1을 반환한다.	String s = "java.lang.java"; int idx1 = s.lastIndexOf('.'); int idx2 = s.indexOf('.');	idx1 = 9 idx2 = 4
lastIndexOf(String str) : int 지정된 문자열을 인스턴스 문자열 끝에서부터 찾아서 위치(index)를 알려준다. 못찾으면 -1을 반환한다.	String s = "java.lang.java"; int idx1 = s.lastIndexOf("java"); int idx2 = s.indexOf("java");	idx1 = 10 idx2 = 0
length() : int 문자열의 길이를 알려준다.	String s = "Hello"; int length = s.length();	length = 5
replace(char old, char new) : String 문자열 중의 문자(old)을 새로운 문자(new)로 모두 바꾼 문자열을 반환한다.	String s = "Hello"; String s2 = s.replace('H', 'C');	s2 = "Cello"
replace(CharSequence old, CharSeqence new) : String 문자열 중의 문자열(old)을 새로운 문자열(new)로 모두 바꾼 문자열을 반환한다.	String s = "Hello" String s2 = s.replace("ll", "LL");	s2 = "HeLLo"
replaceAll(String regex, String replacement) : String 문자열 중에서 지정된 문자열(regex)과 일치하는 것을 새로운 문자열(replacement)로 모두 변경한다.	String s = "AABBAABB"; String s2 = s.replaceAll("BB", "bb");	s2 = "AAbbAAbb"
replaceFirst(String regex, String replacement) : String 문자열 중에서 지정된 문자열(regex)과 일치하는 것 중, 첫 번째 것만 새로운 문자열(replacement)로 변경한다.	String s = "AABBAABB"; String s2 = s.replaceFirst("BB", "bb");	s2 = "AAbbAABB"

split(String regex) : String[] 문자열을 지정된 분리자(regex)fh 나누어 문자열 배열에 담아 반환한다.	String animals = "dog,cat,bear"; String[] arr = animals.split(","); 	arr[0] = "dog" arr[1] = "cat" arr[2] = "bear"
split(String regex, int limit) : String[] 문자열을 지정된 분리자(regex)로 나누어 문자열 배열에 담아 반환한다. 단, 문자열 전체를 지정된 수(limit)로 자른다.	String animals = "dog,cat,bear"; String[] arr = animals.split(",", 2); 	arr[0] = "dog" arr[1] = "cat,bear"
substring(int begin) : String substring(int begin, int end) : String 주어진 시작 위치(begin)부터 끝 위치(end)범위에 포함된 문자열을 얻는다. 이 때, 시작 위치의 문자는 범위에 포함되지만, 끝 위치의 문자는 포함되지 않는다. (begin <= x < end)	String s = "java.lang.Object"; String s1 = s.substring(10); String s2 = s.substring(5, 9); 	s1 = "Object" s2 = "lang"
toLowerCase() : String String 객체에 저장되어 있는 모든 문자열을 소문자로 변환하여 반환한다.	String s = "Hello"; String s1 = s.toLowerCase(s); 	s1 = "hello"
toUpperCase() : String String 객체에 저장되어 있는 모든 문자열을 대문자로 변환하여 반환한다.	String s = "Hello"; String s1 = s.toUpperCase(s); 	s1 = "HELLO"
toString() : String String 객체에 저장되어 있는 문자열을 반환한다.	String s = "Hello"; String s1 = s.toString(); 	s1 = "Hello"
trim() : String 문자열의 왼쪽 끝과 오른쪽 끝에 있는 공백을 없앤 결과를 반환한다. 이 때 문자열 중간에 있는 공백은 제거되지 않는다.	String s = " Hello World "; String s1 = s.trim(); 	s1 = "Hello World"
valueOf(boolean b) : String *valueOf(char c) : String* *valueOf(int i) : String* *valueOf(long l) : String* *valueOf(float f) : String* *valueOf(double b) : String* *valueOf(Object o) : String* 지정된 값을 문자열로 변환하여 반환한다. toString()을 호출한 결과를 반환한다.	String b = String.valueOf(true); String c = String.valueOf('a'); String i = String.valueOf(100); String l = String.valueOf(100L); String f = String.valueOf(10f); String d = String.valueOf(10.0); java.util.Date dd = new java.util.Date(); String Date = String.valueOf(dd); 	b = "true" c = "a" i = "100" l = "100" f = "10.0" d = "10.0" date = Tue Feb 6 08:59:59 KST 2018"

StringBuffer 클래스

메소드 / 설명	예제	결과
StringBuffer() 16문자를 담을 수 있는 버퍼를 가진 StringBuffer객체를 생성한다.	StringBuffer sb = new StringBuffer();	sb = ""
StringBuffer(int length) 지정된 개수의 문자를 담을 수 있는 버퍼를 가진 StringBuffer객체를 생성한다.	StringBuffer sb = new StringBuffer(10);	sb = ""
StringBuffer(String str) 지정된 문자열 값(str)을 갖는 StringBuffer객체를 생성한다.	StringBuffer sb = new StringBuffer("Hi");	sb = "Hi"
append(boolean b) : StringBuffer **append(char c) : StringBuffer** **append(char[] arr) : StringBuffer** **append(double d) : StringBuffer** **append(float f) : StringBuffer** **append(int i) : StringBuffer** **append(long l) : StringBuffer** **append(Object obj) : StringBuffer** **append(String str) : StringBuffer** 매개변수로 입력된 값을 문자열로 변환하여 StringBuffer객체가 저장하고 있는 문자열 뒤에 덧붙인다.	StringBuffer sb = new StringBuffer("ABC"); StringBuffer sb2 = sb.append(true).append('d').append(10.0f); StringBuffer sb3 = sb.append("ABC").append(123);	sb = "ABC" sb2 = "ABCtrued10.0" sb3 = "ABCtrued10.0ABC123"
capacity() : int StringBuffer객체의 버퍼 크기를 알려준다. length()는 버퍼에 담긴 문자열의 길이를 알려준다.	StringBuffer sb = new StringBuffer(100); sb.append("abcd"); int bufferSize = sb.capacity(); int stringSize = sb.length();	bufferSize = 100 stringSize = 4
length() : int StringBuffer객체에 저장되어 있는 문자열의 길이를 반환한다.	StringBuffer sb = new StringBuffer("0123456"); int length = sb.length();	length = 7
charAt(int index) : char 지정된 위치(index)에 있는 문자를 반환한다.	StringBuffer sb = new StringBuffer("abc"); char c = sb.charAt(2);	c = 'c'
delete(int start, int end) : StringBuffer 시작위치(start)부터 끝 위치(end)사이에 있는 문자를 제거한다. 단, 끝 위치의 문자는 제외	StringBuffer sb = new StringBuffer("0123456"); StringBuffer sb2 = sb.delete(3, 6);	sb = "0123456" sb2 = "0126"
deleteCharAt(int index) : StringBuffer 지정된 위치(index)의 문자를 제거한다.	StringBuffer sb = new StringBuffer("0123456"); sb.deleteCharAt(3);	sb = "012456"
replace(int start, int end, String str) : StringBuffer 지정된 범위(start~end)의 문자들을 주어진 문자열로 바꾼다.	StringBuffer sb = new StringBuffer("0123456"); sb.replace(3, 6, "AB");	sb = "012AB8"

end위치의 문자는 범위에 포함되지 않음 (start <= x < end)		
reverse() : StringBuffer StringBuffer객체에 저장되어 있는 문자열의 순서를 거꾸로 나열한다.	StringBuffer sb = new StringBuffer("0123456"); sb.reverse();	sb = "6543210"
setCharAt(int index, char ch) : void 지정된 위치의 문자를 주어진 문자(ch)로 바꾼다.	StringBuffer sb = new StringBuffer("0123456"); sb.setCharAt(5, 'o');	sb = "01234o6"
setLength() : void 지정된 길이로 문자열의 길이를 변경한다. 길이를 늘리는 경우에 나머지 빈 공간은 공백문자로 채운다.	StringBuffer sb = new StringBuffer("0123456"); sb.setLength(5); StringBuffer sb2 = new StringBuffer("0123456"); sb.setLength(10);	sb = "01234" sb2 = "0123456 "
toString() : String StringBuffer객체의 문자열을 String으로 반환한다.	StringBuffer sb = new StringBuffer("0123456"); String str = sb.toString();	str = "0123456"
substring(int start) : String **substring(int start, int end) : String** 지정된 범위 내의 문자열을 String으로 뽑아서 반환한다. 시작위치(start)만 지정하면 시작위치부터 문자열 끝까지를 반환한다.	StringBuffer sb = new StringBuffer("0123456"); String str = sb.substring(3); String str2 = sb.substring(3, 5);	str = "3456" str2 = "34"
insert(int pos, boolean b) : StringBuffer **insert(int pos, char c) : StringBuffer** **insert(int pos, char[] arr) : StringBuffer** **insert(int pos, double d) : StringBuffer** **insert(int pos, float f) : StringBuffer** **insert(int pos, int i) : StringBuffer** **insert(int pos, Object obj) : StringBuffer** **insert(int pos, String str) : StringBuffer** 지정된 위치(pos)의 다음에 두 번째 매개변수로 받은 값을 문자열로 변환하여 추가한다.	StringBuffer sb = new StringBuffer(4, '.');	sb = "0123.456"

Math 클래스

메소드 / 설명	예제	결과
abs(double d) : double *abs(float f) : float* *abs(int i) : int* *abs(long l) : long* 주어진 값의 절대값을 반환한다.	int i = Math.abs(-10); double d = Math.abs(-10.0);	i = 10 d = 10.0
ceil(double d) : double 주어진 값을 올림하여 반환한다.	double d = Mate.ceil(10.1); double d2 = Math.ceil(-10.1); double d3 = Math.ceil(10.0000015);	d = 11.0 d2 = -10.0 d3 = 11.0
floor(double d) : double 주어진 값을 버림하여 반환한다.	double d = Math.floor(10.8); double d2 = Math.floor(-10.8);	d = 10.0 d2 = 11.0
round(double d) : long *round(float f) : long* 소수점 첫째자리에서 반올림한 정수값을 반환한다.	long l = Math.round(5.5); long l2 = Math.round(5.11); long l3 = Math.round(-5.5); long l4 = Math.round(-5.1);	l = 6 l2 = 5 l3 = -5 l4 = -5
rint(double d) : double 주어진 double값과 가장 가까운 정수 값을 double형으로 반환한다.	double d = Math.rint(5.5); double d2 = Math.rint(5.11); double d3 = Math.rint(-5.5); double d4 = Math.rint(-5.1);	d = 6.0 d2 = 5.0 d3 = -6.0 d4 = -5.0
random() : double 0.0 <= x < 1.0 범위의 임의의 double값을 반환한다.	double d = Math.random();	0.0 <= d < 1.0
max(double a, double b) : double *max(float a, float b) : float* *max(int a, int b) : int* *max(long a, long b) : long* 주어진 두 값을 비교하여 큰 쪽을 반환한다.	double d = Math.max(9.5, 9.51); int i = Math.max(0, -1);	d = 9.51 i = 0
min(double a, double b) : double *min(float a, float b) : float* *min(int a, int b) : int* *min(long a, long b) : long* 주어진 두 값을 비교하여 작은 쪽을 반환한다.	double d = Math.max(9.5, 9.51); int i = Math.max(0, -1);	d = 9.5 i = -1