

String Class

String 생성자

자바의 문자열은 java.lang 패키지의 String 클래스의 인스턴스로 관리됩니다. 소스상에서 문자열 리터럴은 String 객체로 자동 생성되지만, String 클래스의 다양한 생성자를 이용해서 직접 String 객체를 생성할 수도 있습니다.

다음은 빈도수가 높은 String 생성자들입니다. 파일의 내용을 읽거나, 네트워크를 통해 받은 데이터는 보통 byte[] 배열이므로 이것을 문자열로 변환하기 위해 사용됩니다.

```
1 // 배열 전체를 String 객체로 생성
2 String str = new String(byte[] bytes);
3 // 배열을 지정한 문자셋으로 디코딩
4 String str = new String(byte[] bytes, String charsetName);
5 // 배열의 offset 인덱스 위치로부터 length만큼 String 객체로 생성
6 String str = new String(byte[] bytes, int offset, int length);
7 // 지정한 문자셋으로 디코딩
8 String str = new String(byte[] bytes, int offset, int length, String charsetName);
9
```

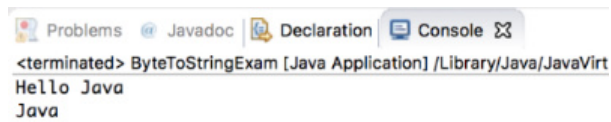
Colored by Color Scripter cs

다음은 바이트 배열을 문자열로 변환하는 예제입니다.

* ByteToStringExam.java

```
1 package api;
2
3
4 public class ByteToStringExam {
5     public static void main(String[] args) {
6         byte[] bytes = { 72, 101, 108, 108, 111, 32, 74, 97, 118, 97 };
7
8         String str1 = new String(bytes);
9         System.out.println(str1);
10
11         String str2 = new String(bytes, 6, 4);
12         System.out.println(str2);
13     }
14 }
15
16
17
```

Colored by Color Scripter cs



<http://palpit.tistory.com>

다음 예제는 키보드로부터 읽은 바이트 배열을 문자열로 변환하는 방법을 보여줍니다. `System.in.read()` 메소드는 키보드에서 입력한 내용을 파라미터로 주어진 바이트 배열에 저장하고 읽은 바이트 수를 리턴합니다.

* KeyboardToStringExam.java

```
1 package api;
2
3
4 import java.io.IOException;
5
6 public class KeyboardToStringExam {
7
8     public static void main(String[] args) throws IOException {
9         byte[] bytes = new byte[100];
10
11         System.out.print("입력: ");
12         int readByteNo = System.in.read(bytes);
13
14         String str = new String(bytes, 0, readByteNo - 1);
15         System.out.println(str);
16     }
17 }
18
19
20
21
```

Colored by Color Scripter CS

```
Problems Javadoc Declaration Console
<terminated> KeyboardToStringExam [Java Application] /Library/Java
입력: Palpit
Palpit
```

<http://palpit.tistory.com>

String 메소드

String은 문자열의 추출, 비교, 찾기, 분리, 변환 등과 같은 다양한 메소드를 가지고 있습니다. 그 중에서 사용 빈도수가 높은 메소드를 다음과 같이 정리해 봤습니다.

리턴 타입	메소드 명(매개 변수)	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[] 리턴
int	indexOf(String str)	문자열 내에서 주어진 문자열의 위치를 리턴
int	length()	문자의 수를 리턴
String	replace(CharSequence target, CharSequence replacement)	target부분을 replacement부분으로 대치한 새로운 문자열 리턴
String	substring(int beginIndex)	beginIndex 위치에서 끝까지 잘라낸 새로운 문자열 리턴
String	substring(int beginIndex, int endIndex)	beginIndex 위치에서 endIndex 전까지 잘라낸 문자열 리턴
String	trim()	앞뒤 공백을 제거한 새로운 문자열 리턴
String	valueOf(int i) valueOf(double d)	기본 타입값을 문자열로 리턴

<http://palpit.tistory.com>

문자 추출(charAt())

charAt() 메소드는 파라미터로 주어진 인덱스의 문자를 리턴합니다. 여기서 인덱스란 0부터 "문자열 길이 - 1" 까지의 번호를 말합니다. 다음 코드를 보면서 이해해 봅시다.

```
1 String subject = "자바머신";  
2 char charValue = subject.charAt(3); cs
```

charAt(3)은 3인덱스 위치에 있는 문자를 말합니다. 위에서는 결국 charValue 변수에 '신' 문자가 들어가게 됩니다.

다음 예제는 주민등록번호에서 인덱스 7번 문자를 읽어 남자와 여자를 구분하는 예제입니다.

*** StringCharAtExam.java**

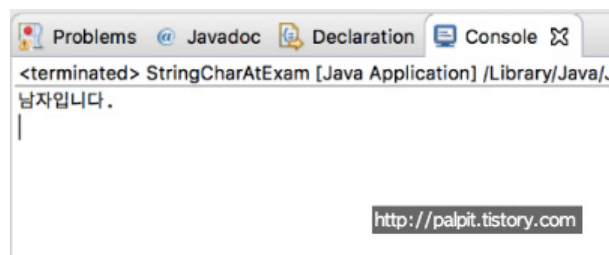
```
1 package api; cs  
2  
3  
4 public class StringCharAtExam {  
5  
6     public static void main(String[] args) {  
7         String ssn = "160329-3232121";  
8     }  
9 }
```

```

8      char sex = ssn.charAt(7);
9      switch (sex) {
10     case '1':
11     case '3':
12         System.out.println("남자입니다.");
13         break;
14
15     case '2':
16     case '4':
17         System.out.println("여자입니다.");
18         break;
19     }
20 }
21
22 }
23
24
25

```

Colored by Color Scripter



문자열 비교(equals())

기본 타입(byte, char, short, int, long, float, double, boolean) 변수의 값을 비교할 때에는 == 연산자를 사용합니다. 그러나 문자열을 비교할 때는 == 연산자를 사용하면 다른 결과를 초래할 수도 있습니다. 아래 예제를 보면서 이해해 봅시다.

* **StringEqualsExam.java**

```

1 package api;
2
3

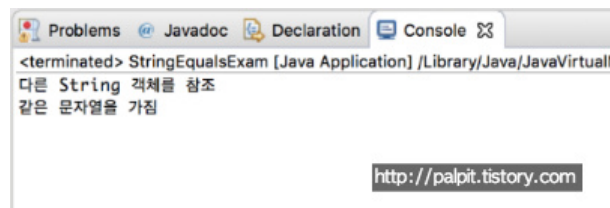
```

```

4 public class StringEqualsExam {
5
6     public static void main(String[] args) {
7         String strVar1 = new String("Jolie");
8         String strVar2 = "Jolie";
9
10        if (strVar1 == strVar2) {
11            System.out.println("같은 String 객체를 참조");
12        } else {
13            System.out.println("다른 String 객체를 참조");
14        }
15
16        if (strVar1.equals(strVar2)) {
17            System.out.println("같은 문자열을 가짐");
18        } else {
19            System.out.println("다른 문자열을 가짐");
20        }
21    }
22 }
23
24
25
26

```

Colored by Color Scripter



위에서 보는 것과 같이 == 연산자와 equals의 차이가 다르다는 것을 알 수 있습니다.

문자열 찾기(indexOf())

indexOf() 메소드는 파라미터로 주어진 문자열이 시작되는 인덱스를 리턴합니다. 만약 주어진 문자열이 포함되어 있지 않으면 -1을 리턴합니다. 다음 코드를 보면서 이해해봅시다.

```

1 String subject = "자바 프로그래밍";
2 int index = subject.indexOf("프로그래밍");

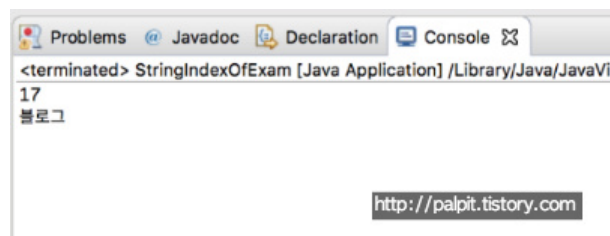
```

index 변수에는 3이 저장되는데, "자바 프로그래밍"에서 "프로그래밍" 문자열의 인덱스 위치가 3이기 때문입니다. indexOf() 메소드는 if문의 조건식에서 특정 문자열이 포함되어 있는지 여부에 따라 실행 코드를 달리할 때 자주 사용됩니다.

* StringIndexOfExam.java

```
1 package api;
2
3
4 public class StringIndexOfExam {
5
6     public static void main(String[] args) {
7         String subject = "Palpit's Tistory blog";
8
9         int loc = subject.indexOf("blog");
10        System.out.println(loc);
11
12        if (subject.indexOf("blog") != -1) {
13            System.out.println("블로그");
14        } else {
15            System.out.println("Nothing");
16        }
17    }
18 }
19
20
21
22
```

Colored by Color Scripter CS



문자열 길이(length())

length() 메소드는 문자열의 길이(문자의 수)를 리턴합니다.

```
1 String subject = "Palpit";
2 int length = subject.length();
3 // length is 6;
4
```

CS

length 변수에는 6이 저장됩니다.

문자열 대치(replace())

replace() 메소드는 첫 번째 파라미터인 문자열을 찾아 두 번째 파라미터로 대치한 새로운 문자열을 생성하고 리턴합니다.

```
1 String str = "자바 프로그래밍";
2 String newStr = str.replace("자바", "JAVA");
3
```

Colored by Color Scripter CS

String 객체의 문자열은 변경이 불가능한 특성을 갖기 때문에 replace() 메소드가 리턴하는 문자열은 원래 문자열의 수정본이 아니라 새로운 문자열입니다. 따라서 newStr 변수는 새로운 문자열을 참조하게 되는 것입니다.

문자열 잘라내기(substring())

substring() 메소드는 주어진 인덱스에서 문자열을 추출합니다. substring() 메소드는 파라미터의 수에 따라 두 가지 형태로 사용됩니다. substring(int beginIndex, int endIndex)는 주어진 시작과 끝 인덱스 사이의 문자열을 추출하고, substring(int beginIndex)는 주어진 인덱스 이후부터 끝까지 문자열을 추출합니다.


```

1 | String ssn = "160329-3123123";
2 | String birth = ssn.substring(0, 6); // 160329
3 | String identity = ssn.substring(7); // 3123123

```

문자열 앞뒤 공백 잘라내기(trim())

trim() 메소드는 문자열의 앞뒤 공백을 제거한 새로운 문자열을 생성하고 리턴합니다. 다음 코드를 보면 newStr 변수는 새로 생성된 "Palpit's blog" 문자열을 참조합니다. trim() 메소드는 앞뒤의 공백만 제거할 뿐中间的 공백은 제거하지 않습니다.

```

1 | String str = " Palpit's blog ";
2 | String newStr = str.trim();

```

문자열 변환(valueOf())

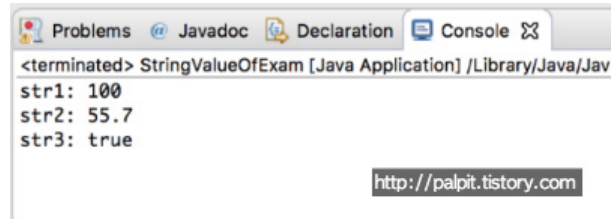
valueOf() 메소드는 기본 타입의 값을 문자열로 변환하는 기능을 가지고 있습니다. String 클래스에는 매개 변수의 타입별로 valueOf() 메소드가 오버로딩 되어 있습니다.

* StringValueOfExam.java

```

1 | package api;
2 |
3 |
4 | public class StringValueOfExam {
5 |
6 |     public static void main(String[] args) {
7 |         String str1 = String.valueOf(100);
8 |         String str2 = String.valueOf(55.7);
9 |         String str3 = String.valueOf(true);
10 |
11 |         System.out.println("str1: " + str1);
12 |         System.out.println("str2: " + str2);
13 |         System.out.println("str3: " + str3);
14 |     }
15 |
16 | }
17 |
18 |

```



The screenshot shows a console window from an IDE. The title bar includes tabs for 'Problems', 'Javadoc', 'Declaration', and 'Console'. The console text is as follows:

```
<terminated> StringValueOfExam [Java Application] /Library/Java/Jav  
str1: 100  
str2: 55.7  
str3: true
```

At the bottom right of the console area, there is a small dark box containing the URL <http://palpit.tistory.com>.