字符串

字符串是不可变对象

```
public final class String
    implements java.io.Serializable, Comparable<String>, CharSequence {
    /** The value is used for character storage. */
    private final char value[];
}
```

- `public final class String`: 字符串类型被final修饰，不可以被继承
- `private final char value[];`字符串内部封装了一个不可变的字符数组

String常量池

- Java 为了提高性能，静态字符串（**字面量、常量、常量拼接的结果**）都在**常量池中创建**，并尽量使用同一个对象，**重用静态字符串。**
- 使用字符串字面量，创建字符串对象时，JVM会**首先在常量池中查找，如果存在就从常量池中返回对象。**不存在才会创建一个新的字符串对象。

示例代码：

```
public class StringDemo {
    public static void main(String[] args) {
        String s1 = "123abc";
        String s2 = "123abc";
        System.out.println(s1 == s2); // true

        String s3 = "123abc";
        System.out.println(s1 == s3); // true

        s1 = s1 + "!";                // 123abc!
        System.out.println(s1);       // 123abc
        System.out.println(s2);       // 123abc!

        System.out.println(s1 == s2); // false
        System.out.println(s3 == s2); // true
    }
}
```

内存编码及长度

- String在内存中采用Unicode编码，每个字符占两个字节。
- 字符串中的char存储的是二进制Unicode编码，长度是16位，即两个字节。

String APIs

int length()

获取字符串长度。

示例代码：

```
public class LengthDemo {
    public static void main(String[] args) {
        String string = "我爱Java!";
        int len = string.length();
        System.out.println("len：" + len);
    }
}
```

控制台输出：

```
len：7
```

boolean isEmpty()

判断是否为空串

示例代码：

```
public class IsEmptyDemo {

    public static void main(String[] args) {
        String str1 = "";
        System.out.println(str1.isEmpty());

        String str2 = "达内科技";
        System.out.println(str2.isEmpty());

    }
}
```

控制台输出：

```
true
false
```

int indexOf(char c)

- **检索字符或字符串。**
- 重载的方法：

方法	作用
`int indexOf(String str)`	在字符串中检索`str`，返回第一次出现的位置，如果找不到则返回`-1`
`int indexOf(String str, int fromIndex)`	从指定位置开始检索`str`，返回第一次出现的位置，如果找不到则返回`-1`

示例代码：

```
public class IndexOfDemo {
    public static void main(String[] args) {
        String str = "Thinking in java!";
                    // 01234567890123456
        int i = str.indexOf("i");   // 2
        System.out.println(i);
        i = str.indexOf("X");       // -1
        System.out.println(i);
        /**
         * 从指定位置开始查找字符
         */
        i = str.indexOf('i', 3);    // 5
        System.out.println(i);
        i = str.indexOf('i', 6);    // 9
        System.out.println(i);
        i = str.indexOf('i', 10);   // -1
        System.out.println(i);

        /**
         * 查找字符串串
         */
        i = str.indexOf("in");      // 9
        System.out.println(i);

        /**
         * 从指定位置查找字符串
         */
        i = str.indexOf("in", 3);   // 5
        System.out.println(i);

        /**
         * 找不到，返回-1
         */
        i = str.indexOf("OK");      // -1
        System.out.println(i);
    }
}
```

int lastIndexOf(char c)

检索字符或字符串，返回最后一次出现的位置。

方法	作用
`int lastIndexOf(Char c)`	返回字符`c`最后一次出现的位置
`int lastIndexOf(String str)`	返回字符串`str`最后一次出现的位置

示例代码：

```
package string.api;

public class LastIndexOfDemo {
    public static void main(String[] args) {
        String str = "Thinking in Java!";
                    //01234567890123456
        int i = str.lastIndexOf("in");  // -9
        System.out.println(i);
        i = str.lastIndexOf("In");      // -1
        System.out.println(i);

        int a = str.lastIndexOf('a');
        System.out.println(a);          // 15
    }
}
```

char charAt(int index)

获取指定下标处的字符

示例代码：

```
package string.api;

public class CharAtDemo {
    public static void main(String[] args) {
        String str = "Thinking in Java";
                    //01234567890123456
        char c = str.charAt(12);    // J
        System.out.println(c);
        /**
         * 如果指定的下标过大，会出现数组下标越界异常
         */
//      c = str.charAt(99);
        /**
         * 案例：统计一下str中有几个i字符？
         */
        int count = 0;
        for(int i = 0; i < str.length(); i++) {
            if (str.charAt(i)=='i') {
                count ++;
            }
        }
        // Thinking in Java中有3个i字符
        System.out.println(str+"中有"+count+"个i字符");
```

```
    }
}
```

boolean contains(String str)

判断是否包含指定字符串

示例代码：

```
package string.api;

public class ContainsDemo {
    public static void main(String[] args) {
        String line = "I love you!";
        boolean containsLove = line.contains("love");
        System.out.println("contains love: " + containsLove);
    }
}
```

控制台输出:

```
contains love: true
```

boolean equals(Object anObject)

判断两个字符串内容是否相等

示例代码：

```
package string.api;

public class EqualsDemo {
    public static void main(String[] args) {
        String s1 = "ABC";
        String s2 = "ABC";

        String s3 = new String(s1);

        System.out.println(s1==s2); // true
        System.out.println(s1==s3); // false
        System.out.println(s1.equals(s3)); //true

    }
}
```

char[] toCharArray()

将字符串转为字符数组。

示例代码：

```java
package string.api;

public class ToCharArrayDemo {
    public static void main(String[] args) {
        String string = "Hello, Thank you!";
        char[] charArray = string.toCharArray();
        for (char c : charArray) {
            System.out.println(c);
        }
    }
}
```

控制台输出:

```
H
e
l
l
o
,

T
h
a
n
k

y
o
u
!
```

String toLowerCase()

返回字符串的小写形式
示例代码:

```java
package string.api;

public class ToLowerCaseDemo {

    public static void main(String[] args) {
        String str = "Hello, World!";
        // hello, world!
        String lowerCaseStr = str.toLowerCase();
        System.out.println(lowerCaseStr);
    }
}
```

String toUpperCase()

返回字符串的大写形式

示例代码：

```
package string.api;

public class ToUpperCaseDemo {

    public static void main(String[] args) {
        String str = "Hello, World!";
        //            HELLO, WORLD!
        String upperCaseStr = str.toUpperCase();
        System.out.println(upperCaseStr);
    }
}
```

String substring(int begin, int end)

- **截取子字符串。**
- 重载方法：

方法	作用
`String substring(int beginIndex, int endIndex)`	返回字符串从`beginIndex`到`endIndex`的子字符串。含前不含后
`String substring(int beginIndex)`	截取子字符串，从指定位置开始，一直截取到最后。

示例代码：

```
package string.api;

public class SubstringDemo {
    public static void main(String[] args) {
        String str = "Thinking in Java!";
        //            01234567890123456

        // begin index
        String subStr = str.substring(1);   // hinking in Java!
        System.out.println(subStr);
        // begin index, end index
        subStr = str.substring(1, 2);       // h
        System.out.println(subStr);
        // begin index, begin index + n
        subStr = str.substring(0, 0+5);     // Think
        System.out.println(subStr);

        /**
         * 案例:检查一个图片是否为.png
```

```
         */
        String file = "demo.png";
        // begin index
        String suffix = file.substring(file.length()-4);
        if (!suffix.equals(".png")) {
            System.out.println("file不是图片");
        } else {
            System.out.println("file是图片");// file是图片
        }

    }
}
```

String trim()

去除字符串两端的空白字符。

示例代码：

```
package string.api;

public class TrimDemo {
    public static void main(String[] args) {
        String str = "\n\t   小猪  \n";
        System.out.println("str:" + str);
        str = str.trim();
        System.out.println("strTrimed:" + str);


        String name = "robin  ";
        name = name.trim();
        if (name.equals("robin")) {
            System.out.println("用户名存在");
        } else {
            System.out.println("用户名不存在");
        }
    }
}
```

控制台输出：

```
str:
        小猪

strTrimed:小猪
用户名存在
```

String replace(char oldChar, char newChar)

使用指定的**新字符**替换**所有**指定的**旧字符**。

```
package string.api;

public class ReplaceDemo {
    public static void main(String[] args) {
        String string = "I love you!";
        char oldChar = 'o';
        char newChar = 'e';
        string = string.replace(oldChar, newChar);
        System.out.println(string);
        // I leve yeu!
    }
}
```

String 正则相关API

String replaceFirst(String regex, String replacement)

使用正则表达式替换文本，只替换第一次匹配到的文本。

示例代码：

```
package string.api.regex;

public class ReplaceFirstDemo {
    public static void main(String[] args) {
        String line = "我去上学，我去吃饭，我去看电影";
        String regex = "我[去]";
        line = line.replaceFirst(regex, "**");
        System.out.println(line);
        // **上学，我去吃饭，我去看电影

    }
}
```

String replaceAll(String regex, String replacement)

使用正则表达式替换所有匹配到的文本，并将结果返回。

示例代码:

```
package string.api.regex;

public class ReplaceAllDemo {
    public static void main(String[] args) {
        String line = "我去上学，我去吃饭，我去看电影";
        String regex = "我[去]";
        line = line.replaceAll(regex, "**");
        System.out.println(line);
        // **上学，**吃饭，**看电影
    }
}
```

String split(String regex, int limit)

使用正则表达式，分割字符串，得到一个字符串数组。 `int limit`是分割次数。

```
package string.api.regex;

public class SplitDemo {
    public static void main(String[] args) {
        String string = "three two one ready~ go!";
        String[] strings = string.split("\\s");
        for (String str : strings) {
            System.out.println(str);
        }
        System.out.println("--------");
        String[] strings2 = string.split("\\s", 4);
        for (String str2 : strings2) {
            System.out.println(str2);
        }
    }
}
```

控制台输出:

```
three
two
one
ready~
go!
--------
three
two
one
ready~ go!
```