# 1.字符与字符串

|  |  |  |  |
| --- | --- | --- | --- |
| No | 方法名称 | 类型 | 说明 |
| 1 | Public String(char[] value) | 构造 | 将字符数组变为String |
| 2 | public String(char[] value,int offset,int count) | 构造 | 将部分数组变为String |
| 3 | public char charAt(int index) | 普通 | 返回索引对象的字符 |
| 4 | Public char[] toCharArray() | 普通 | 将字符串以转字符数组的形式返回 |

# 2.字节与字符串

IO 操作会用到字节

|  |  |  |  |
| --- | --- | --- | --- |
| No | 方法名称 | 类型 | 说明 |
| 1 | public String(byte[] bytes) | 构造 | 将全部的字节数组变为String |
| 2 | public String(byte[] bytes,int offset,int length) | 构造 | 将部分字节数组变为String |
| 3 | Public byte[] getBytes() | 普通 | 将字符串变为字节数组 |
| 4 | public String(byte[] bytes,String charsetName)  throws UnsupportedEncodingException | 普通 | 进行编码转化 |

范例:小写转大写

public class test{  
 public static void main(String args[]){  
 String str = "helloworld";  
 byte [] data = str.getBytes();  
 for(int x = 0;x<data.length;x++){  
 data[x] -=32;  
 }  
 System.*out*.print(new String(data));  
 }  
}

# 3.字符串比较

|  |  |  |  |
| --- | --- | --- | --- |
| No | 方法名称 | 类型 | 说明 |
| 1 | public boolean equals(Object anObject) | 普通 | 内容比较是否相等 |
| 2 | public boolean equalsIgnoreCase(String anotherString) | 普通 | 字符串内容比较是否相等,不区分大小写 |
| 3 | public int compareTo(String anotherString) | 普通 | 比较两个字符串内容的大小(按照字符编码比较)，返回差值  差值等于0:表示内容相等  差值大于0:表示大于后者  差值小于0:表示小于后者 |

# 4.字符串查找

|  |  |  |  |
| --- | --- | --- | --- |
| No | 方法名称 | 类型 | 说明 |
| 1 | public boolean contains(CharSequence s) | 普通 | 判断指定的子串是否存在 |
| 2 | public int indexOf(String str) | 普通 | 查找指定字符串的位置,从前往后,返回位置的索引,查找不到返回-1 |
| 3 | public int indexOf(String str,int fromIndex) | 普通 | 有指定位置（包括指定位置）从前向后查找指定字符串,返回位置的索引,找不到返回-1 |
| 4 | public int lastIndexOf(String str) | 普通 | 由后向前查找指定字符串的位置,找不到返回-1 |
| 5 | public int lastIndexOf(String str, int fromIndex) | 普通 | 从指定位置由后向前查找字符串的位置,找不到返回-1 |
| 6 | public boolean startsWith(String prefix) | 普通 | 判断是否以指定的字符串开头 |
| 7 | public boolean startsWith(String prefix,int toffset) | 普通 | 从指定的位置开始判断是否不以指定的字符串开头 |
| 8 | public boolean endsWith(String suffix) | 普通 | 判断是否以指定的字符串结尾 |

# 5.字符串替换

|  |  |  |  |
| --- | --- | --- | --- |
| No | 方法名称 | 类型 | 说明 |
| 1 | public String replaceAll(String regex,String replacement) | 普通 | 用新的内容替换全部旧的内容 |
| 2 | public String replaceFirst(String regex,String replacement) | 普通 | 替换首个满足条件的内容 |

范例:

public class test{  
 public static void main(String args[]){  
 String str = "hello,world";  
 String result1 = str.replaceAll("l","\_");  
 System.*out*.println(str);  
 System.*out*.println(result1);  
 String result2 = str.replaceFirst("l","\_");  
 System.*out*.println(result2);  
 }  
}  
//结果  
//hello,world  
//he\_\_o,wor\_d  
//he\_lo,world

# 6.字符串截取

|  |  |  |  |
| --- | --- | --- | --- |
| No | 方法名称 | 类型 | 说明 |
| 1 | public String substring(int beginIndex) | 普通 | 从指定索引截取到结尾 |
| 2 | public String substring(int beginIndex,int endIndex) | 普通 | 截取部分字符串的数据,不包括endIndex所在的字符 |

# 7.字符串拆分

|  |  |  |  |
| --- | --- | --- | --- |
| No | 方法名称 | 类型 | 说明 |
| 1 | public String[] split(String regex) | 普通 | 按照指定的字符串进行全部拆分 |
| 2 | public String[] split(String regex,int limit) | 普通 | 按照指定的字符串进行部分拆分返回的字符串数组的长度由limit决定,前面拆,后面不拆 |

# 8.字符串连接

Public String concat(String str)

public class test{  
 public static void main(String args[]){  
 String s1 = "hello";  
 String s2 = "helloworld";  
 String s3 = "hello" + "world";  
 String s4 = s1.concat("world");  
 System.*out*.println(s3 == s2); //true  
 System.*out*.println(s4 == s2); //false  
 }  
}

concat和”+”类似,但是concat是返回一个new String 字符串

# 9.字符串转大小写

|  |  |  |  |
| --- | --- | --- | --- |
| No | 方法名称 | 类型 | 说明 |
| 1 | Public String toLowerCase() | 普通 | 将字符串转为小写 |
| 2 | Public toUpperCase() | 普通 | 将字符串转为大写 |

# 10.去除字符串左右空白trim()

Public String trim()

# 11.字符串是否为空isEmpty()

Public boolean isEmpty() 空字符串指的是””而不是null

通过””.equals(str)也一样

# 12.字符串首字母大写,其余字母小写

这个功能要自己写

public class test{  
 public static void main(String args[]){  
 String str = "hello,world";  
 System.*out*.print(*initcap*(str));  
 }  
 public static String initcap(String temp){  
 return temp.substring(0,1).toUpperCase() + temp.substring(1).toLowerCase();  
 }  
}