字符串

`char` 是一个基本数据类型它在只可以保存一个2字节的`Unicode`字符，而字符串是一长串的字符，但是`java`中没有字符串这个基本数据类型，只可以通过类来定义，而且字符串中的字符一旦确定，就无法进行修改，只能重新创建

String类

也是一个类，每个使用`""`创建的字符串都是它的一个实例对象，一下两种创建字符串的方法一样

```
public class Main {

    public static void main(String[] args)  {
        String str1 = "Szy is pig";
        String str2 = new String("Szy is pig");
    }
}
```

⭐但是这两种创建的方法也有不同，使用`""`号无论怎么创建他们都是同一个对象

```
    public static void main(String[] args)  {
        String str1 = "Szy is pig";
        String str2 = "Szy is pig";
        System.out.println(str1 == str2);
    }

//输出: true
```

而使用`new`创建的就是创建两个不同的对象

```
    public static void main(String[] args)  {
        String str1 = new String("Tc is superman");
        String str2 = new String("Tc is superman");
        System.out.println(str1 == str2);
    }

//输出: false
```

⭐ 如果仅仅只比较内容是否相等需要使用`equals`

```
    public static void main(String[] args)  {
        String str1 = new String("Tc is superman");
        String str2 = new String("Tc is superman");
        System.out.println(str1.equals(str2));
    }

//输出: true
```

⭐ String 既然是个类，就表示它也可以许多可以使用的方法比如：

1. 求字符串的长度

```
    public static void main(String[] args)  {
        String str1 = new String("Tc is superman");
        System.out.println(str1.length());
    }

 //输出: 14

 //也可以这样使用

System.out.println("bx is good good".length());
```

2. 裁剪`substring`和分割`splits`，这里都是返回一个新的`String`而不是在原有的`String`基础上修改

```
    public static void main(String[] args)  {
        String str1 = new String("Tc is superman");
        String sub = str1.substring(0,3);
        System.out.println(str1.substring(0,3)); //只保留0到3的字符串，每个字符串和数组一样下标都是从0开始
    }

//输出: Tc
```

```
    public static void main(String[] args)  {
        String str1 = new String("Tc is superman");
        String[] sub = str1.split(" "); //依照" "来分割，返回的是个字符串数组
        for (String s : sub) {
            System.out.println(s);
        }
    }

//输出:
Tc
is
superman
```

3. 字符数组和字符串之间是可以互相装换的，使用`toCharArray`将字符串转换为字符数组

```
    public static void main(String[] args)  {
        String str1 = "Hello World!";
        char[] char1 = str1.toCharArray(); //转换为字符数组
        System.out.println(char1);
    }

//输出: Hello World!
```

```
    public static void main(String[] args)  {
        char[] char1 = {'加','油'};
        String str = new String(char1); //转换为字符串
        System.out.println(str);
    }

//输出: 加油
```

4. 其他一些字符串类方法

以下是Java中String类的一些常用方法：

方法名称	功能描述
`length()`	返回字符串的长度（字符数）
`charAt(int index)`	返回指定索引位置的字符
`indexOf(int ch)`	返回指定字符在字符串中首次出现的位置，如果没有找到则返回-1
`lastIndexOf(int ch)`	返回指定字符在字符串中最后一次出现的位置，如果没有找到则返回-1
`substring(int beginIndex)`	返回从指定索引开始到字符串末尾的子字符串
`substring(int beginIndex, int endIndex)`	返回从beginIndex到endIndex-1之间的子字符串
`concat(String str)`	将指定的字符串连接到此字符串的末尾，并返回新的字符串
`startsWith(String prefix)`	检查此字符串是否以指定的前缀开头
`endsWith(String suffix)`	检查此字符串是否以指定的后缀结尾
`equals(Object obj)`	比较两个字符串是否相等，忽略大小写
`equalsIgnoreCase(String anotherString)`	比较两个字符串是否相等，不考虑大小写
`toLowerCase()`	将此字符串转换为小写字母
`toUpperCase()`	将此字符串转换为大写字母
`trim()`	去除字符串首尾的空白字符
`replace(char oldChar, char newChar)`	替换字符串中的所有指定字符
`split(String regex)`	根据给定的正则表达式将字符串分割成子字符串数组

方法名称	功能描述
`contains(CharSequence sequence)`	检查此字符串是否包含指定的字符序列
`matches(String regex)`	告诉此字符串是否匹配给定的正则表达式
`replaceAll(String regex, String replacement)`	使用给定的replacement替换此字符串中所有匹配给定的正则表达式的子字符串
`replaceFirst(String regex, String replacement)`	使用给定的replacement替换此字符串中第一个匹配给定的正则表达式的子字符串
`format(String format, Object... args)`	根据给定的格式字符串和参数返回一个格式化的字符串

这只是String类中的一部分常用方法，实际上String类还提供了许多其他的方法。

```
    public static void main(String[] args)  {
        String str1 = "Hello World";
        String str2 = "Hello World";
        System.out.println(str1.contains("Hello"));//判断字符串中是否有这个字符或字符串
        System.out.println(str1.equals(str2)); //判断字符串内容是否相等
        System.out.println(str1.indexOf("H")); //反回指定字符第一次出现的地址
        System.out.println(str1.lastIndexOf("o")); //返回指定字符最后一次出现的地址
        System.out.println(str1.toLowerCase()); //将字符都转换成小写字母
        System.out.println(str1.toUpperCase()); //转换成大写字母
        System.out.println(str1.endsWith("d")); //检测字符串是否以指定字符结尾
        System.out.println(str1.startsWith("h")); //检测字符串是否以指定字符开始
        System.out.println(str1.trim()); //去除首尾的空白字符
    }
```

StringBuilder类

字符串支持`+,+=`进行拼接操作，但是拼接字符串实际上做了很多操作，但是编译器在编译的时候会将这些拼接优化成一个字符串

```
    public static void main(String[] args)  {
        String str1 = "Hello" + "World";
        System.out.println(str1);
    }
```

编译器优化后

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

但是如果是多个对象编译器就不敢动了

```
    public static void main(String[] args)  {
        String str1 = "Hello";
        String str2 = "Hello";
        String str3 = "Hello";
        String str4 = "Hello";
        String str5 = str1 + str2 +str3+str4;
        System.out.println(str5);
    }
```

⭐ 编译后，看似是将4个字符串连接到一个字符串了，但是确实是`str1`先`str2`先连接成一个字符串，然后在和`str3`连接，这样就连接了好多此，非常麻烦

```
    public static void main(String[] args) {
        String str1 = "Hello";
        String str2 = "Hello";
        String str3 = "Hello";
        String str4 = "Hello";
        String str5 = str1 + str2 + str3 + str4;
        System.out.println(str5);
    }
```

优化后可以写成这样，这个类型是主要是构建字符串的，可以使用它来对字符串进行拼接，裁剪等操作，它弥补了字符串不能修改的不足

```
    public static void main(String[] args)  {
        String str1 = "Hello";
        String str2 = "Hello";
        String str3 = "Hello";
        String str4 = "Hello";
        StringBuilder str5 = new StringBuilder();
        str5.append(str1).append(str2).append(str3).append(str4);
        System.out.println(str5.toString());
```

⭐ 使用方法

```
    public static void main(String[] args)  {
        StringBuilder str = new StringBuilder();
        str.append("AAA").append("BBB");
        System.out.println(str);

    }
```

⭐ 使用`delete`裁剪

```java
    public static void main(String[] args)  {
        StringBuilder str = new StringBuilder();
        str.append("AAA").append("BBB");
        str.delete(2,4);
        System.out.println(str);

    }
```

以下是Java中`StringBuilder`类的常用方法列表，以及它们的基本功能描述：

方法名称	功能描述
`append(data)`	追加数据到`StringBuilder`对象末尾。`data`可以是不同的类型（如String, char, int等）。
`insert(index, obj)`	在指定的索引位置插入数据。`obj`是要插入的对象，可以是不同类型。
`delete(start, end)`	删除从`start`索引开始到`end`索引之间的字符（不包含`end`索引的字符）。
`deleteCharAt(index)`	删除指定索引位置的字符。
`reverse()`	将`StringBuilder`中的字符串翻转。
`replace(start, end, str)`	用新的字符串`str`替换此序列中指定的字符序列。
`toString()`	返回当前`StringBuilder`对象的字符串表示形式。
`length()`	返回`StringBuilder`对象的长度（字符数）。
`capacity()`	返回`StringBuilder`对象的当前容量（可容纳的总字符数）。
`ensureCapacity(int capacity)`	确保`StringBuilder`至少具有指定的容量。如果需要，增加容量。
`setLength(int newLength)`	设置`StringBuilder`对象的新长度。如果新长度小于此序列当前的长度，则截断此序列。如果新长度大于此序列当前的长度，则用null字符('\u0000')填充此序列。
`substring(int start)`	返回一个新字符串，它是此字符串的一个子字符串。
`substring(int start, int end)`	返回一个新字符串，它是此字符串的一个子字符串。起始索引是`start`，结束索引是`end-1`。
`charAt(int index)`	返回指定索引处的字符。
`indexOf(String str)`	返回在此序列中第一次出现指定子字符串的索引；如果此序列中没有这样的子字符串，则返回 -1。

方法名称	功能描述
`lastIndexOf(String str)`	返回在此序列中最右边出现的指定子字符串的索引；如果在此序列中未出现该子字符串，则返回 -1。
`getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)`	将此序列中的字符从此序列中的指定开始索引复制到目标字符数组。
`reverse()`	将字符以相反的顺序放入此序列中。
`trimToSize()`	将此序列的容量调整为此序列的当前大小。
`setCharAt(int index, char ch)`	将指定索引位置的字符设置为 `ch`。
`indexOf(String str, int fromIndex)`	返回在此序列中第一次出现指定子字符串的索引，从指定的索引开始搜索；如果此序列中没有这样的子字符串，则返回 -1。
`lastIndexOf(String str, int fromIndex)`	返回在此序列中最右边出现的指定子字符串的索引，从指定的索引开始搜索；如果在此序列中未出现该子字符串，则返回 -1。
`replace(int start, int end, String str)`	用另一个指定的字符串替换此序列的子序列。