Java API

1. API文档
2. JDK API
   1. 什么是JDK API？
      * + JDK中包含大量的API类库(System、Scanner、String)，所谓API（Application Programming Interface,应用程序编程接口），已写好，可供直接调用的功能(方法，函数)，在Java中这个功能以类的形式去封装。
   2. JDK包结构
      * + 为了便于使用和维护，JDK类库按照包结构划分，不同功能的类划分在不同的包中。

|  |  |
| --- | --- |
| 包 | 功能 |
| java.lang | String,多线程(Thread、Runnable)，System等，该包中类使用频率非常高，不需要import关键字导入，可以直接使用。 |
| java.util | 常用的工具类，集合、随机数、日历、时钟 |
| java.io | 文件操作、流操作 |
| java.net | 网络操作Socket |
| java.math | 数学运算相关操作 |
| java.security | 安全相关操作 |
| java.sql | 数据库访问（JDBC技术） |
| java.text | 处理文字、日期、数字、信息的格式 |

!不知道的类，查API

* 1. 查询API的步骤
     + - 将鼠标放置需要查询的类之上，获取其包名
       - 在API中找到对应的包，按首字母顺序找到相应的类

1. 文档注释规范
   * + - 单行注释 //注释内容-----某一句语句之后
       - 多行注释 /\*注释内容\*/ /\* ---- 多条语句之前

\* 注释内容

\*/

* 1. 文档注释
     + - 以/\*\*注释内容\*/ /\*\* ----某个类或某个方法之前

\* 注释内容

\*/

* 1. 文档注释规范

- 加在类或方法的开头，用于说明作者、时间、版本、要实现的功能等详细描述的信息。

- 通过Javadoc工具，可以轻松的将此注释转化为HTML文档说明。

- 文档注释和单行及多行的区别在于可以在程序员之间进行代码维护和交流，可以生成API供类的使用者使用。

* 1. Java doc命令生成文档



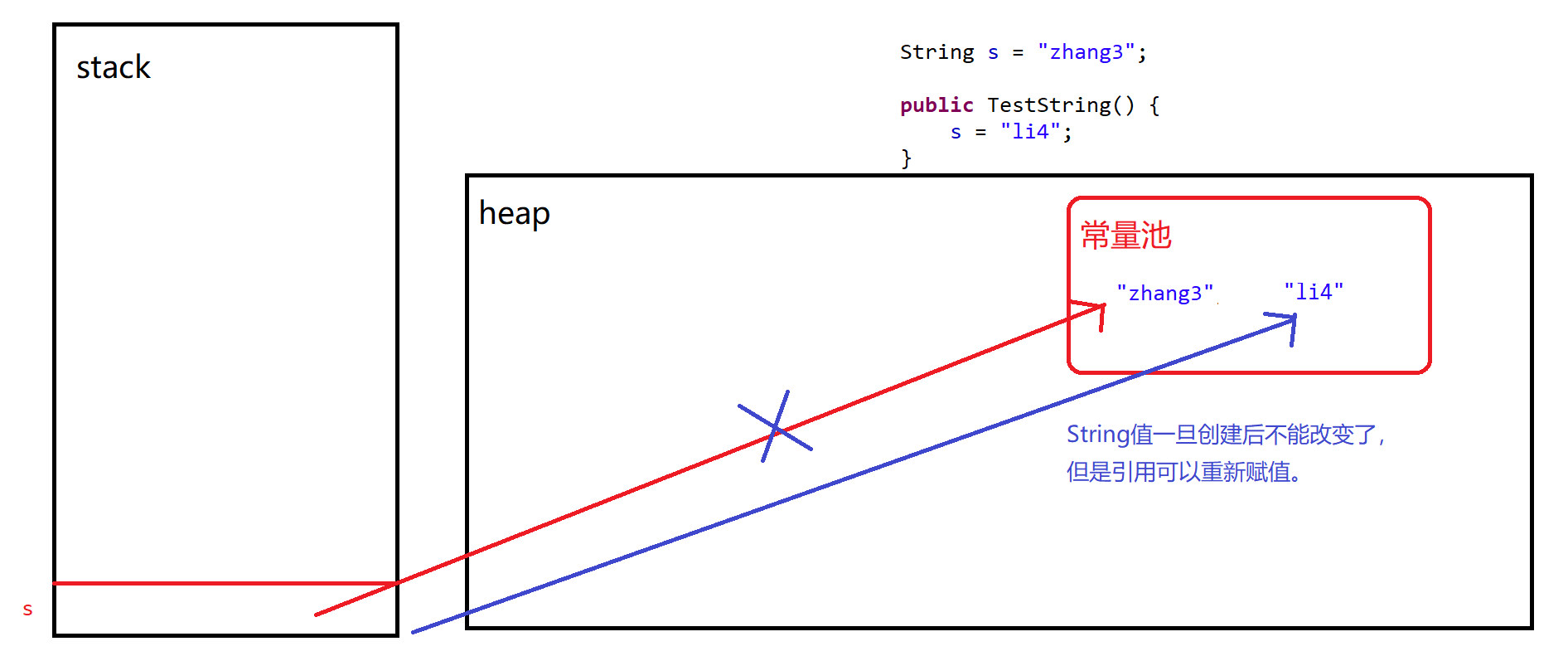
先进入到当前文件目录下执行javadoc -d docDir \*.java

1. 字符串及其基本操作
2. String及其常用API
   * + - String 是不可变对象

>. java.lang.String使用final修饰，不能被继承。

>. 字符串底层封装了字符数组及针对字符数组操作的算法。

>. 字符串一旦创建，对象就永远无法改变，但字符串的引用可以重新赋值。



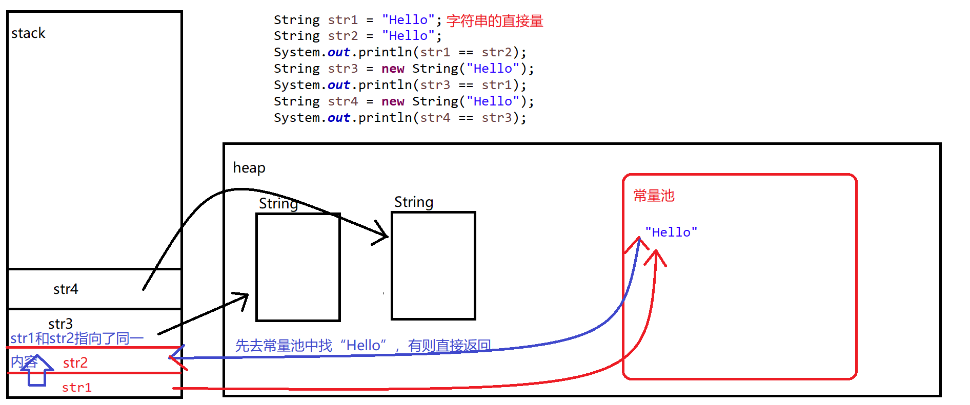
>. Java字符串在内存中采用Unicode编码方式，任意一个字符对应两个字节的定长编码。

* + - * String 常量池(重点、面试)

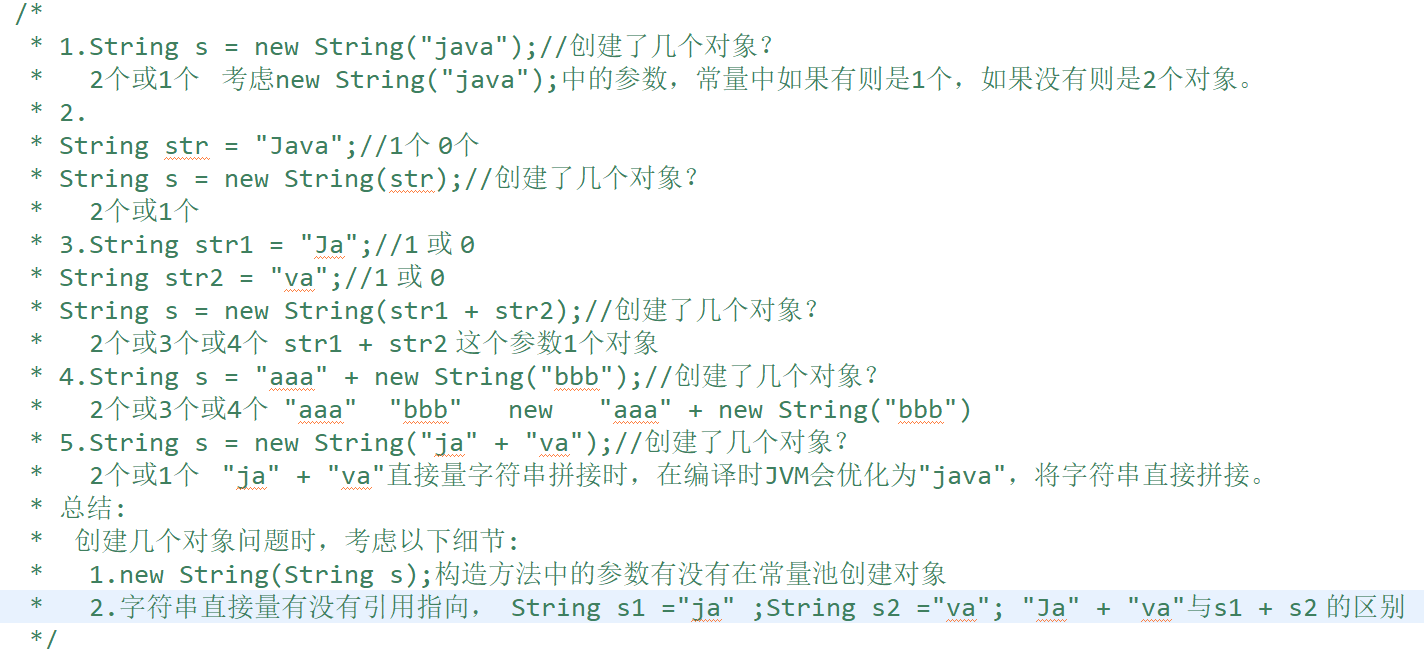
>. JVM中一块特殊的内存空间

>. Java语言为了提高性能，静态字符串(字面量，常量，常量连接结果)在常量池中创建，并尽量使用同一个对象，重用静态字符串。

>. 对于重复出现的字符串的直接量(用双引号括起来的内容)，JVM会首先从常量中去查找，如果有则返回，如果没有，则创建该对象。



常量池创建了几个对象问题(面试):

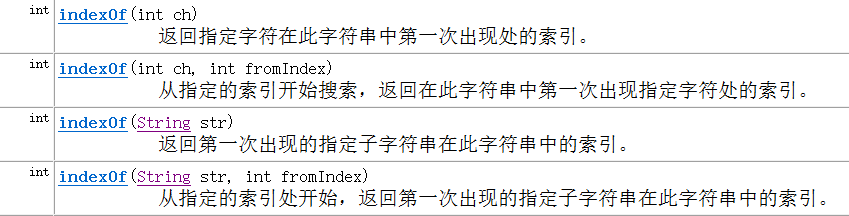


* + - * 内存编码及长度

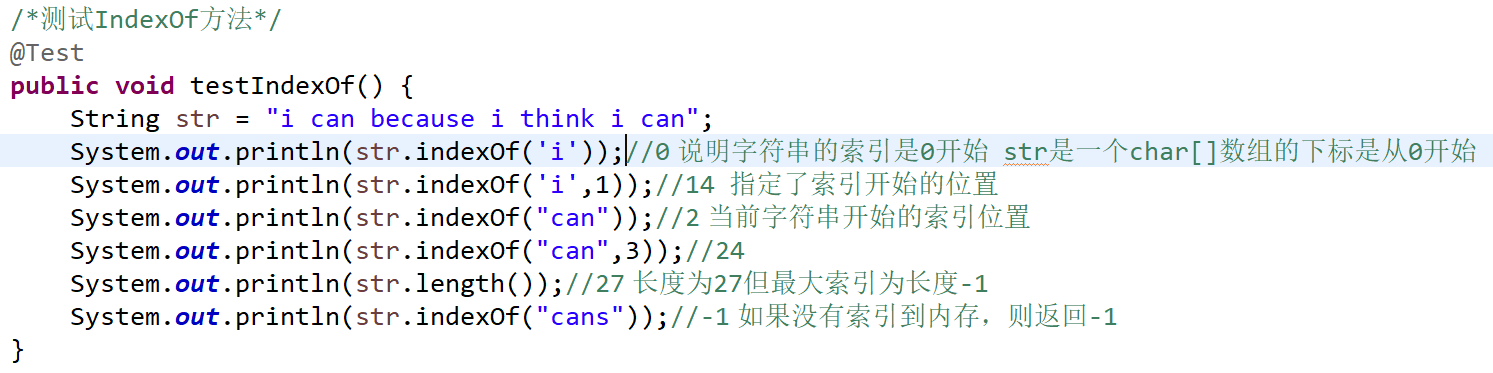
>. String在内存中采用Unicode编码，每个字符占用两个字节；任何一个字符(无论是中文还是英文)都算1个字符长度，两个字节。

>.String的length()方法在底层实际返回了封装在String中字符数组的长度。return value.length;

* + - * 使用indexOf实现检索

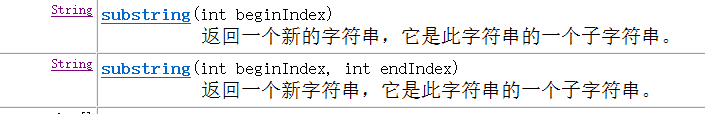


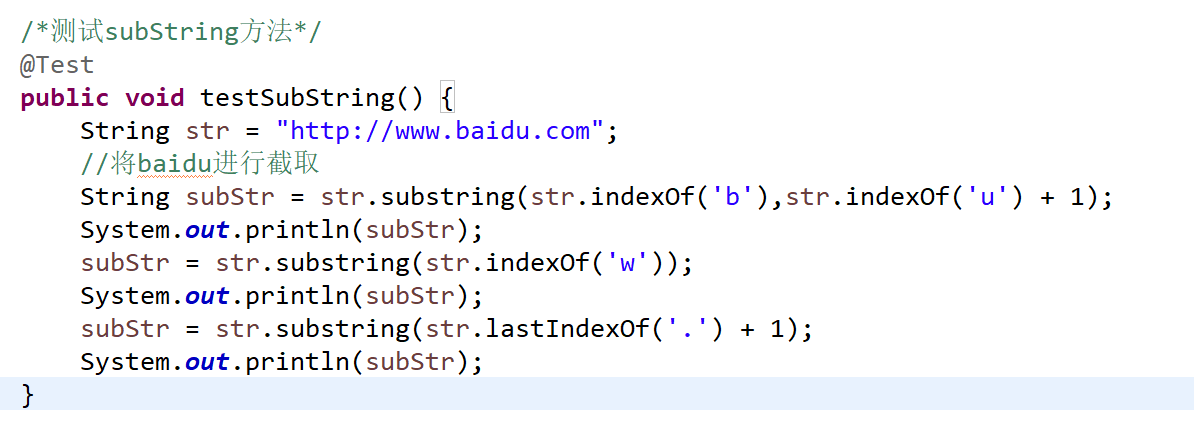
>.IndexOf方法用于实现在字符串中检索另外一个字符串



* + - * 使用subString获取字串

>.subString方法用于返回一个字符串的子字符串

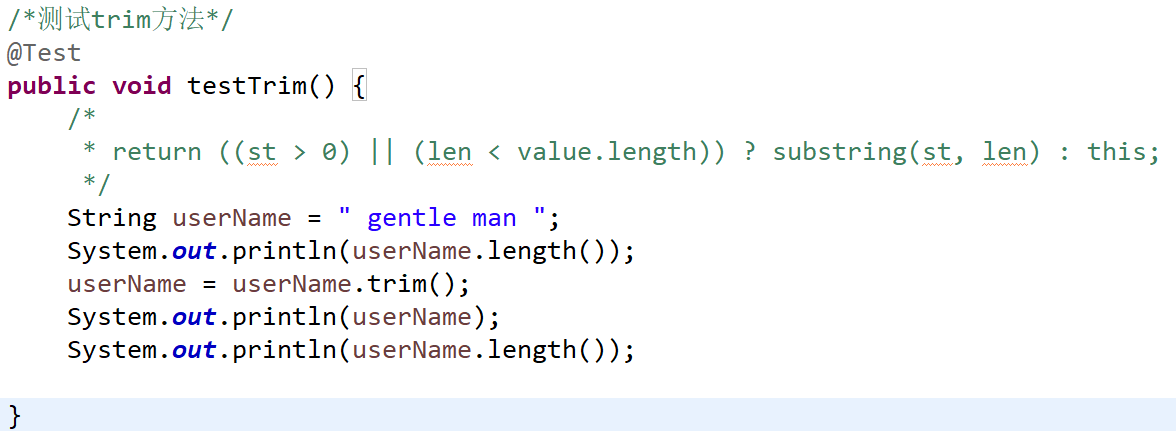




* + - * trim

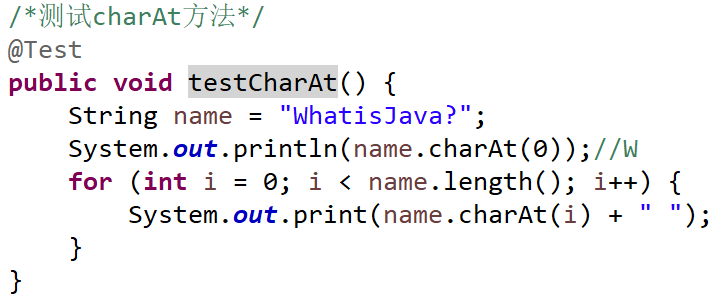


>.去除两端空格



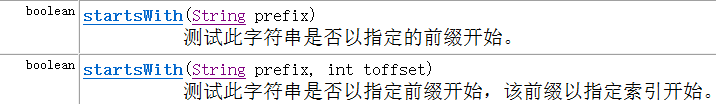
* + - * charAt



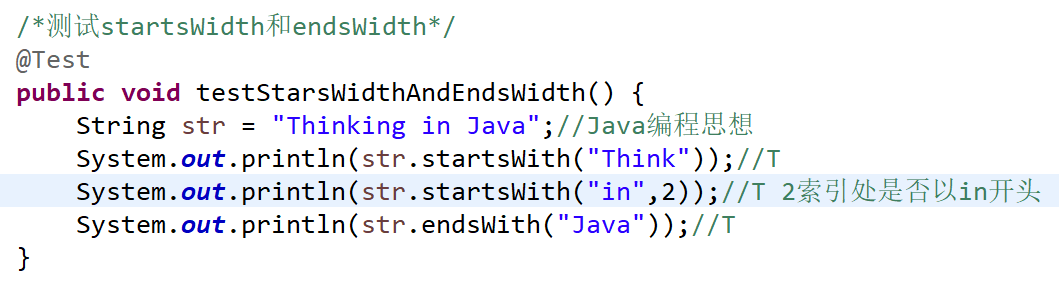


>. 底层实现原理:返回字符数组的指定索引元素return value[index];

* + - * startsWith和endsWith







* + - * 大小写变换

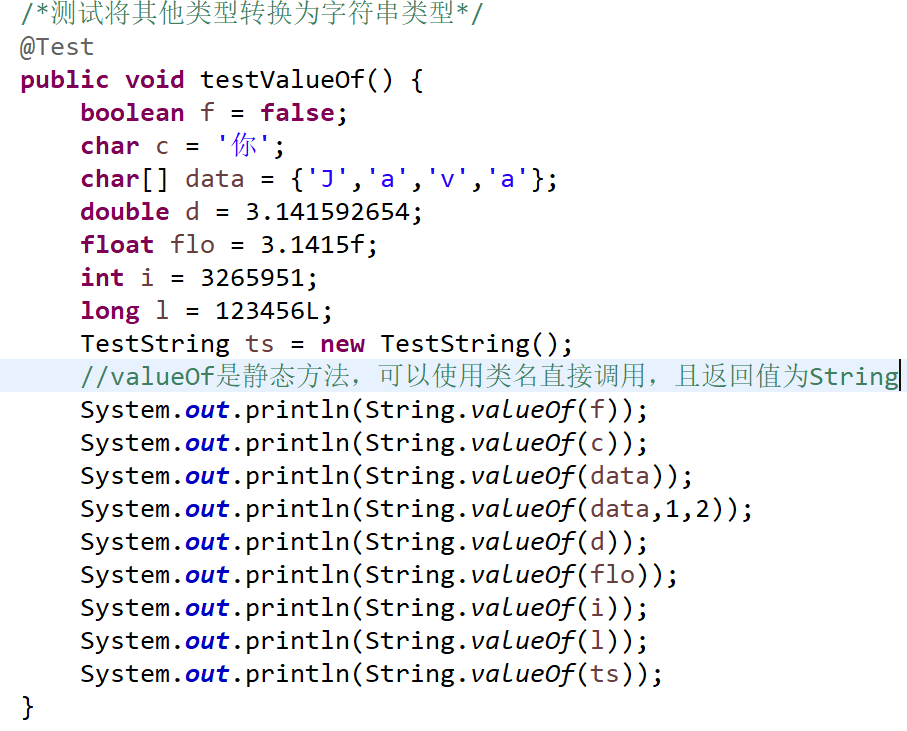




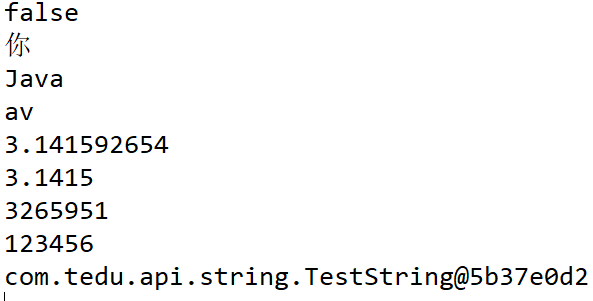
* + - * ValueOf



>.将其他类型转化为字符串类型，(常用)



运行结果：



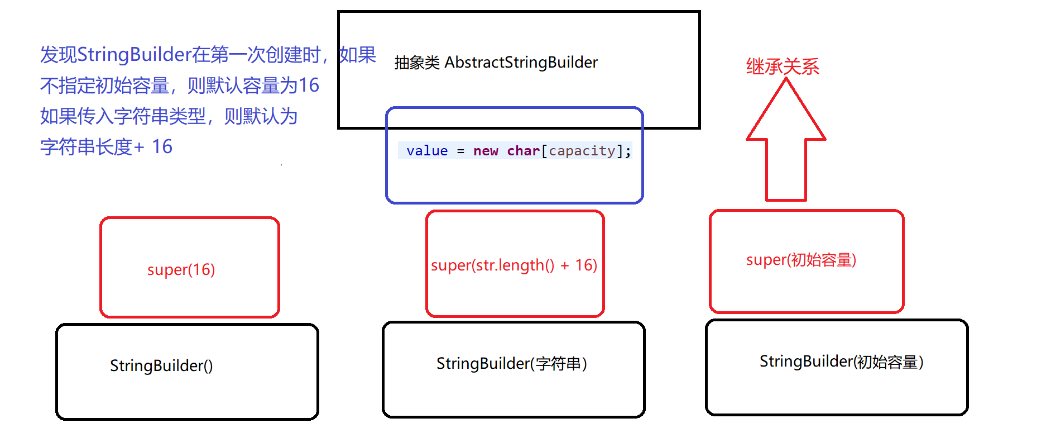
1. StringBuilder
   * + - StringBuilder封装可变字符串

>. StringBuilder封装可变字符串，对象创建后可以根据调用方法改变其封装的字符序列

>. 常见构造方法

public StringBuilder()

public StringBuilder(String str)

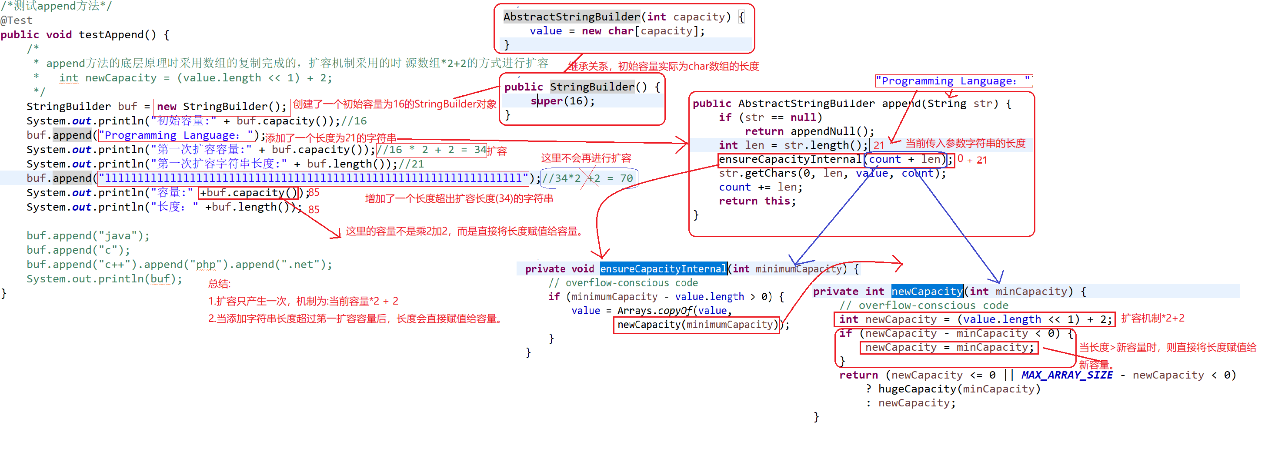


* + - * 使用append实现追加

|  |  |
| --- | --- |
| StringBuilder类常用方法 | 功能 |
| StringBuilder append（String str） | 追加 |
| StringBuilder insert(int dstoffset,String s) | 插入 |
| StringBuilder delete(int start,int end) | 删除 |
| StringBuilder replace(int start,int end,String str) | 替换 |
| StringBuilder reverse() | 字符串反转 |

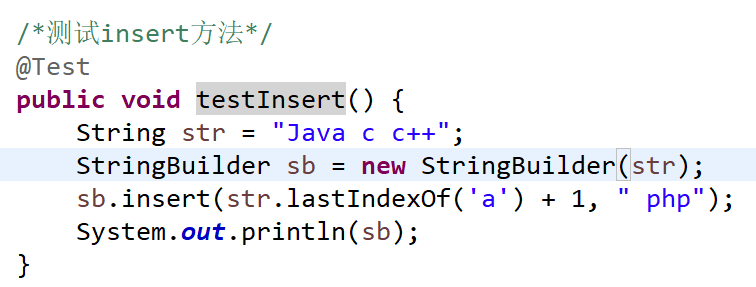
>. StringBuilder的很多方法的返回值都为StringBuilder类型。

这些方法的返回值均为:return this;



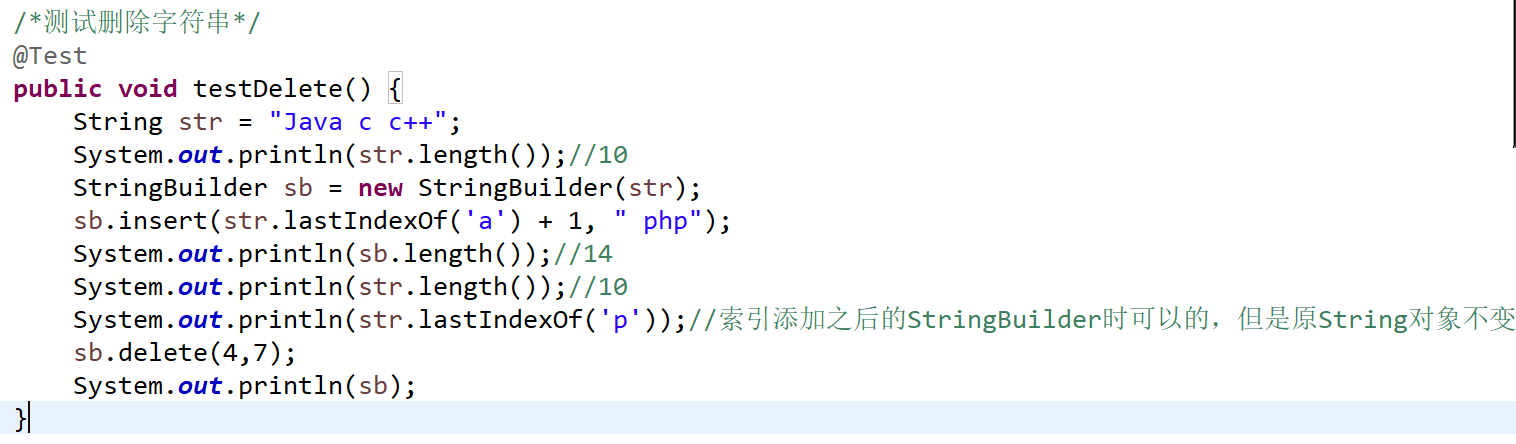
* + - * intsert

>.插入字符串数据



* + - * delete

>.删除字符串



* + - * StringBuilder和StringBuffer区别（面试）:

>. StringBuilder是可变字符串。字符串计算，建议采用StringBuilder实现，性能稍快。

>. Java的字符串连接符是采用StringBuilder实现的。

String s = “s” + “DE” + 1 ; 等价于:

String s = new StringBuilder(s).append(“s”).append(“DE”).append(1).toString();

>. 区别在于:

- StringBuilder是非线程安全的，并发处理的，性能快。

- StringBuffer是线程安全的(源码中的方法用synchronized修饰)，同步处理的性能慢。

三、正则表达式

1. 基本正则表达式
   * + - 正则表达式简介

>. 对字符串进行一些复杂的匹配、查找、替换等操作时，正则表达式可以简化上述操作。

>. 正则表达式: 一串特定的字符，组成“规则字符串”，是描述文本规则的工具，而正则表达式就是记录文本规则的代码。

|  |  |
| --- | --- |
| 正则表达式 | 功能 |
| [abc] | a、b、c中任意一个字符 |
| [^abc] | 除了a、b、c外任意一个字符 |
| [a-z] | a、b、c、…………z中任意一个字符 |
| [a-zA-Z0-9] | a~z、A~Z、0~9中的任意一个字符 |
| [a-z&&[^bc]] | a~z中除了b和c中的任意一个字符 |

>.预定义字符集

|  |  |
| --- | --- |
| 正则表达式 | 功能 |
| . | 任意一个字符 |
| \d | 任意一个数字字符，相当于[0-9] |
| \w | 单词字符，相当于[a-zA-Z0-9\_] |
| \s | 空白字符，相当于[\t\n\x0B\f\r] |
| \D | 任意一个非数字字符 |
| \W | 任意一个非单词字符 |
| \S | 任意一个非空白字符 |

>.数量词

|  |  |
| --- | --- |
| 正则表达式 | 功能 |
| X？ | 表示0个或1个X |
| X\* | 表示0个或任意多个X |
| X+ | 表示1个到任意多个X(大于等于1个X) |
| X{n} | 表示n个X |
| X{n,} | 表示n个到任意多个X(大于等于n个X) |
| X{n,m} | 表示n个到m个X |

* + - * 分组“()”

>. ()表示分组，可以将一系列正则表达式看成一个整体，分组时可以使用|表示或关系。

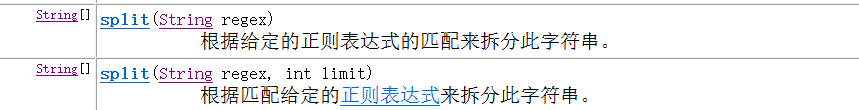
(\+86|0086)?\s?\d{11} // 有区号的电话号码的正则表达式。

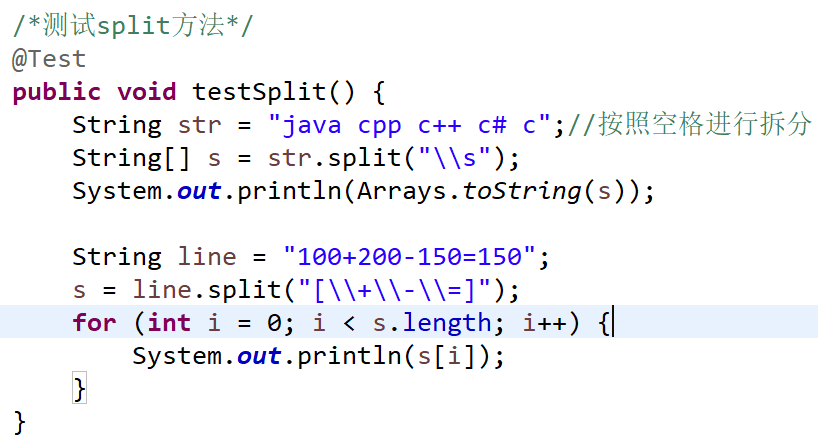
* + - * “^”和“$”

1. String及其正则API
   * + - matches方法

>. 将一个字符串与正则表达式进行匹配，返回值为boolean

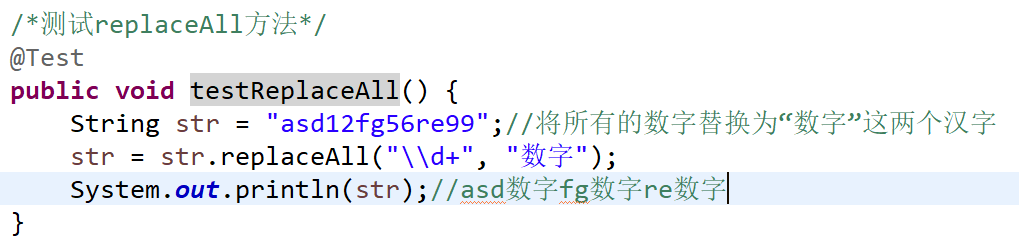
* + - * split方法





* + - * replaceAll方法

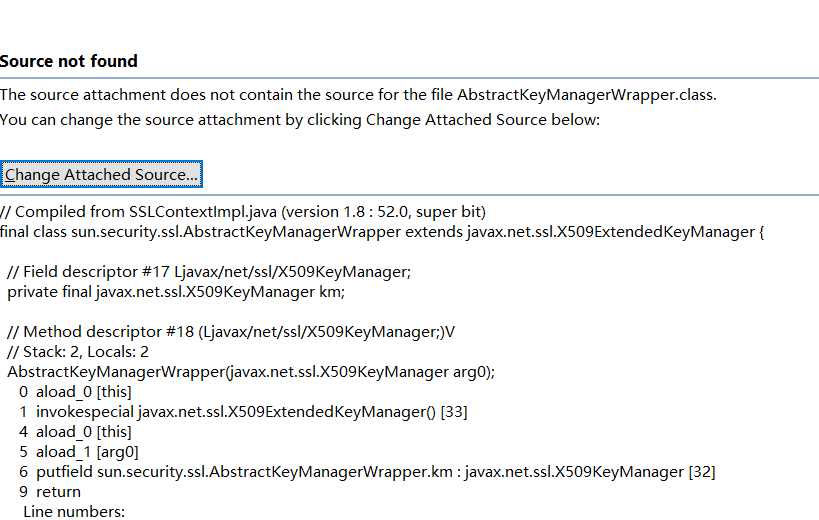




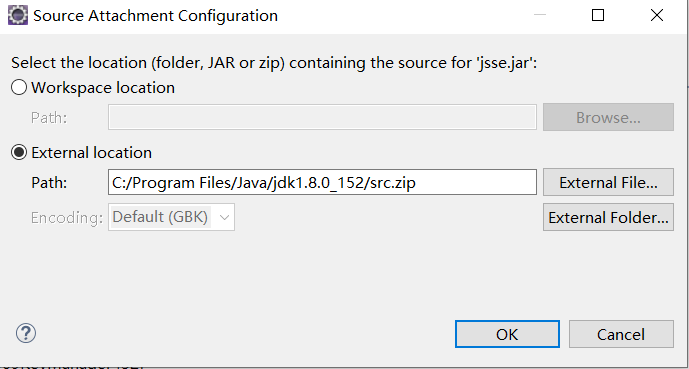
附件1:

添加源码包

1. 点击



1. 勾选第二个



1. 选择External File



1. 进入JDK安装目录下，导入src.zip，点击确定。