蓝桥 LAN QIAO

Java 程序设计基础

String 字符串

String 字符串主要内容

- 5.1 Java API 及 API 文档
- 5.2String 类（重点）
- 5.3StringBuffer 类（重点）
- 5.4其他工具类

教学目标

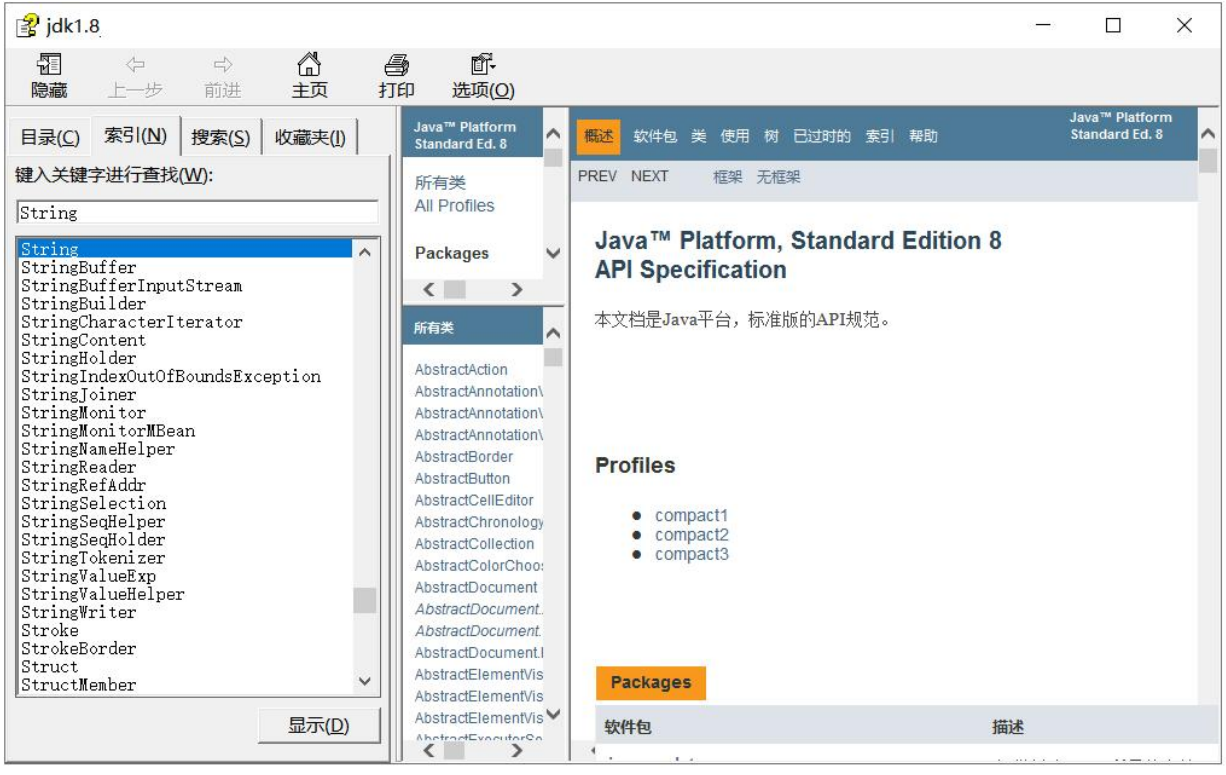
- 能通过查阅 API 文档学习类及方法的基本作用
- **能准确使用 String 类中的常用方法及其作用**
- **能理解 String 的不可变性**
- **能准确区分 String 的两种“相等”比较**
- 能准确区分 StringBuffer 类和 String 类的特点
- 能正确的使用构造方法创建 String、StringBuffer 及 StringBuilder 对象
- 能理解 StringBuffer 的内存模型
- 能正确使用 Date、SimpleDateFormat 等常用工具类

Java API

- Java 给程序员提供了 Java API（Application Programming Interface，应用程序编程接口）文档，供 Java 程序员随时查阅。
- API 文档描述了类库中的类及其方法的输入、输出和功能，程序员依据文档直接调用，无须关注实现细节。
- 在使用 Java API 文档时，需要注意 API 文档的版本号要和 JDK 的版本号一致，否则可能出现文档与实际使用的类库不一致的情况。

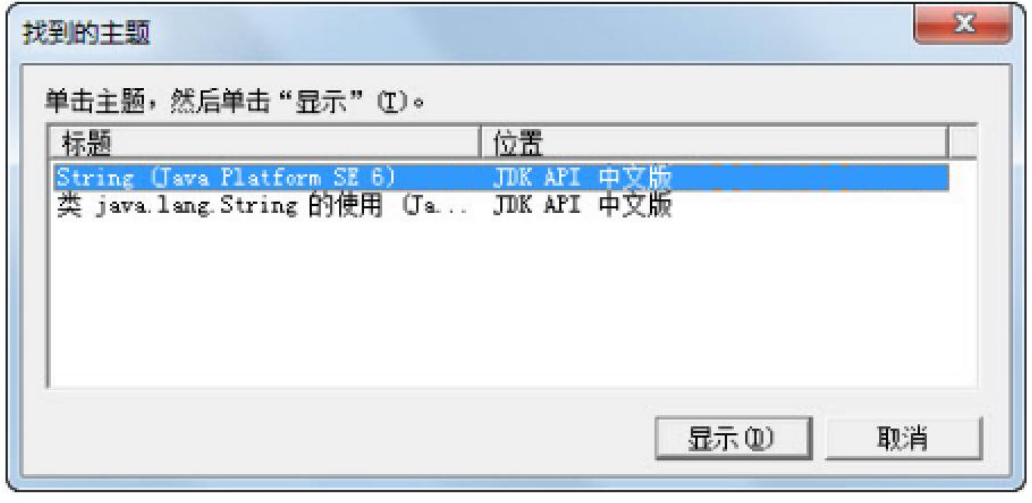
API 文档

- 以 JDK8 为例，Java API 文档的文档结构



API 文档

- 如果要查找 String 字符串类的其他方法，可以在“索引”处输入 String，在弹出的对话框（见下图）中选择相关主题，单击“显示”按钮，就会显示 String 类的相关内容



String 简介

- String 类表示字符串，Java 程序中的所有字符串（例如“蓝桥”）都作为此类的对象
- String 类不是基本数据类型，它是一个类。因为对象的初始化默认值是 null，所以 String 类对象的初始化默认值也是 null
- **String 是一种特殊的对象**

  String 字符串是常量，字符串的值在创建之后不能更改

  String 类是 final 修饰的最终类，因此不能被继承

创建 String 对象

- String(String s)：创建一个新的 String 对象，使其内容为参数 s 中的字符序列
- String(char[] value)：创建一个新的 String 对象，使其内容为参数 value 中的字符序列
- String(char[] value, int offset, int count)：创建一个新的 String 对象，其内容为取自字符数组参数 value 的一个子序列。offset 参数是子数组第一个字符的索引（从 0 开始建立索引），count 参数指定子数组的长度

创建 String 对象

- 示例

String stuName1 = new String("王云");

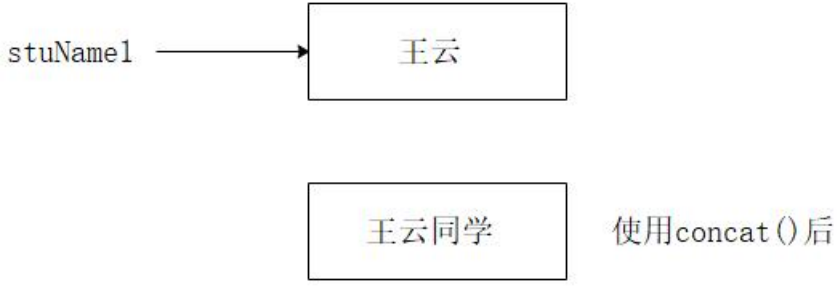
char[] charArray = {'刘','静','涛'};

String stuName2 = new String(charArray);

String stuName3 = new String(charArray,1,2);//从'静'字开始，截取 2 个字符，结果是“静涛”

不可变特性

- **String 字符串是常量，字符串的值在创建之后不能更改。**concat(String str)方法实际创建了一个新 String 字符串，用来存放 stuName1 字符串加上“同学”的结果，而不是在原来 stuName1 字符串的后面增加内容，对于 stuName1 而言，它是常量，内容并没有变化



比较相等

- 比较字符串常用的两个方法是运算符 == 和 String 类的 equals 方法

  使用“==”比较两个字符串，是比较两个对象在内存中的地址是否一致，本质上就是判断两个变量是否指向同一个对象，如果是则返回 true，否则返回 false

  String 类的 equals 方法则是比较两个 String 字符串的内容是否一致，返回值也是一个布尔类型。

常用方法

- public char charAt(int index)

  返回 index 指定的索引处的字符

- public int length()

  返回此字符串的长度。这里需要和获取数组长度区别开，获取数组长度是通过数组名.length获取的

- public int indexOf(String str)

  返回指定子字符串 str 在此字符串中第一次出现处的索引。

常用方法

- public int indexOf(String str,int fromIndex)

  返回指定子字符串 str 在此字符串中第一次出现处的索引，从指定的索引 fromIndex 处开始搜索

- public boolean equalsIgnoreCase(String another)

  将此 String 与另一个字符串 another 比较，比较时不区分大小写

- public String replace(char oldChar,char newChar)

  返回一个新的字符串，它是通过用 newChar 替换此字符串中出现的所有 oldChar 得到的。这里再重申一下，String 类方法中的索引都是从 0 开始编号的。

常用方法

- public boolean startsWith(String prefix)

  判断此字符串是否以 prefix 指定的前缀开始

- public boolean endsWith(String suffix)

  判断此字符串是否以 suffix 指定的后缀结束

- public String toUpperCase()

  将此 String 中的所有字符都转换为大写

常用方法

- public String toLowerCase()

  将此 String 中的所有字符都转换为小写

- public String substring(int beginIndex)

  返回一个从 beginIndex 开始到结尾的新的子字符串

- public String substring(int beginIndex,int endIndex)

  返回一个从 beginIndex 开始到 endIndex 结尾（不含 endIndex 所指字符）的新的子字符串

常用方法

- public String trim()

  返回字符串的副本，忽略原字符串前后的空格

- public String[] split(String regex)

  通过 regex 指定的分隔符分隔字符串，返回分隔后的字符串数组。

常用方法

- public static String valueOf(基本数据类型参数)

  返回基本数据类型参数的字符串表示形式，例如：

  public static String valueOf(int i)

  public static String valueOf(double d)

常用方法

- 前两个方法是 String 类的静态方法，关于“静态”的内容将会在后面的课程详细介绍，这里需要大家注意的是，静态方法是通过类名.方法名直接调用的，例如：

```
String result = String.valueOf(100);//将int型100转换为字符串"100"
```

StringBuffer 简介

- StringBuffer 字符串代表的是可变的字符序列，可以对字符串对象的内容进行修改
- StringBuffer 类最常用的构造方法。

  StringBuffer()：构造一个空白的字符串缓冲区，其初始容量为 16 个字符。

  StringBuffer(String str)：构造一个字符串缓冲区，并将其内容初始化为指定的字符串内容。

StringBuffer 简介

- 通过 StringBuffer 类的构造方法创建 StringBuffer 字符串

```
StringBuffer strB1 = new StringBuffer();
System.out.println(strB1.length());
```

- 以上通过 strB1.length()返回的字符串长度是 0。但实际上，strB1 的底层创建了一个长度为 16 的字符数组，为接收字符串内容做准备。

```
StringBuffer strB2 = new StringBuffer("柳海龙");
System.out.println(strB2.length());
```

通过strB2.length()返回长度是 3，实际在底层创建了一个长度为 3+16 的字符数组

追加和插入

- public StringBuffer append(String str)

  将 str 指定的字符串追加到此字符序列的末尾

- public StringBuffer append(StringBuffer str)

  将 str 指定的 StringBuffer 字符串追加到此序列的末尾

- public StringBuffer append(char[] str)

  将 str 指定的字符数组追加到此序列的末尾。

追加和插入

- public StringBuffer append(char[] str,int offset,int len)

  自索引 offset 开始截取 str 的 len 个字符追加到此序列中。

- public StringBuffer append(double d)

  将 double 类型的变量 d 的字符串表示形式追加到此序列的末尾。

- public StringBuffer append(Object obj)

  将参数 obj 的字符串表示形式追加到此序列的末尾。

- public StringBuffer insert(int offset,String str)

  将字符串 str 插入到此字符序列中，offset 表示插入位置。

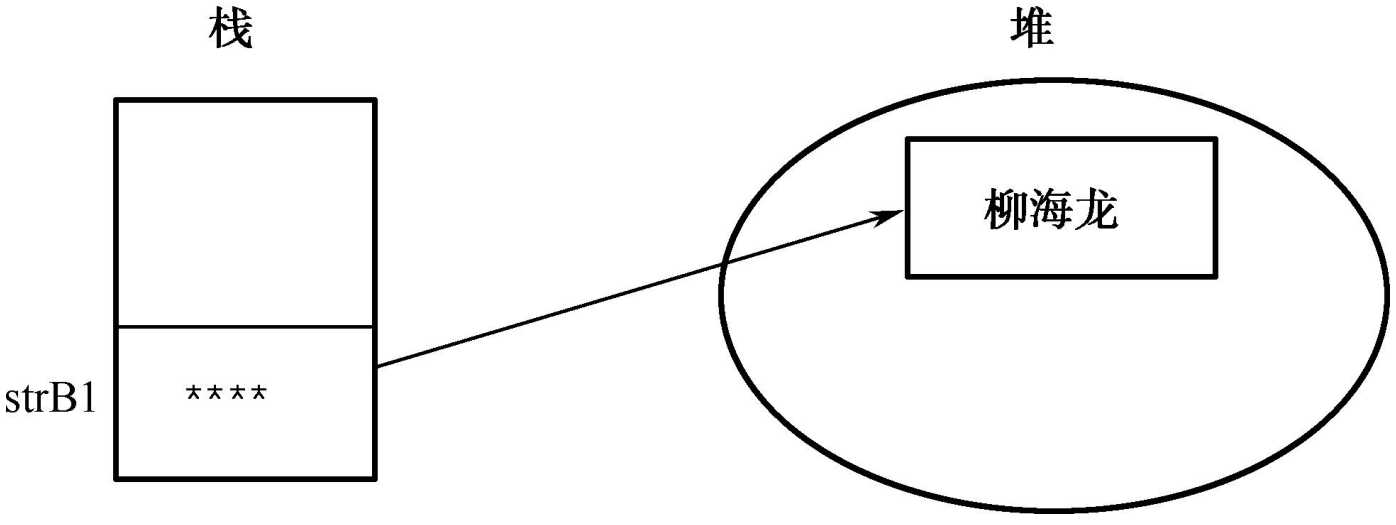
StringBuilder

- **StringBuilder 类是在 Java 5.0 中被提出的，它和 StringBuffer 之间的最大不同在于 StringBuilder 的方法是非线程安全的，而 StringBuffer 是线程安全的**

- 对于我们初学者而言，非线程安全意味着速度更快，因此在目前阶段建议使用 StringBuilder 类；等以后大家学习了高并发相关内容以后，再根据情况考虑使用 StringBuffer。

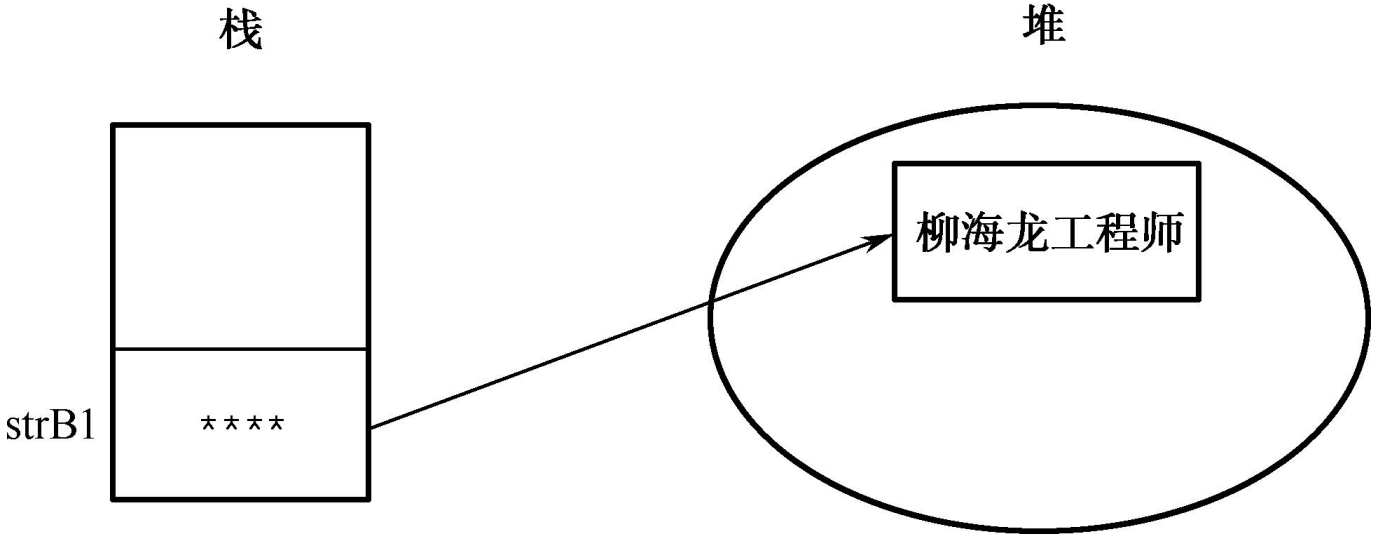
StringBuilder

- **StringBuffer 是一个内容可变的字符序列，或者说它是一个内容可变的字符串类型。**
  当使用 StringBuffer strB1 = new StringBuffer("柳海龙"); 语句创建 StringBuffer 对象时，内存结构示意图如图所示。



StringBuilder

- 当使用 strB1.append("工程师") 方法时，将之前创建的 StringBuffer 对象的内容“柳海龙”修改成“柳海龙工程师”，内存结构示意图如图所示



Date 类

- JDK8.0 以前的 Date 类用于定义当前的日期和时间,提供了以下常用方法

- **Date(long millisec)**

  带一个参数的 Date 构造方法，该参数是从 1970 年 1 月 1 日零点起的毫秒数。

- **long getTime()**

  返回自 1970 年 1 月 1 日 零点以来，此 Date 对象经过的毫秒数。

- **boolean after(Date date)**

  判断当前对象是否在参数 date 指定的日期之后。如果是返回 true,否则返回 false。

- **boolean before(Date date)**

  判断当前对象是否在参数 date 指定的日期之前。如果是返回 true,否则返回 false。

Date 类

- 在 JDK8.0 提供的新日期 API 中，所有的类都是不可变的，这就对高并发编程提供了友好支持。并且新的 API 将“时间”的概念更加精细化，提供了日期（Date）、时间（Time）、日期时间（DateTime）、时间戳（unix timestamp）以及时区等细化的时间类

- **DK8.0 新增的日期 API 都在 java.time 包下，其中常见的 API 如下表所示**

API	含义
Instant	时间戳
LocalDate	日期，如 2020-06-16
LocalTime	时刻，如 11:52:52
LocalDateTime	具体时间，如 2020-06-16 11:52:52

SimpleDateFormat 类

- 在 SimpleDateFormat 中，是通过构造方法指定 Date 的输出格式的，并且在设置这些格式时需要使用时间通配符，常见的通配符如下表所示

通配符	含义
y	年
M	月
d	日
H	时
m	分
s	秒

- 例如 yyyy-MM-dd HH:mm:ss 就代表了“4 位数年-2 位数月-2 位数日 2 位数时:2 位数分:2 位数秒”的日期显示格式

其他类

- 除了 Date 和 SimpleDateFormat 以外，JDK 还提供了非常丰富的其他工具类
- 例如，Calendar 类为特定时间与日历字段（YEAR、MONTH、DAY_OF_MONTH、HOUR 等）之间的转换，以及操作日历字段提供了一些方法。而 Math 类可以方便的进行一些数学运算。Calendar、Math 以及其它工具类的具体使用，大家可以查阅 Java API 文档。

笔试题

- 下列String字符串类的（ ）方法实现了“将一个字符串按照指定的分隔符分隔，返回分隔后的字符串数组”的功能。

A substring(...)

B split(...)

C valueOf(...)

D replace(...)

笔试题

- 给定如下Java代码，编译运行后，输出的结果将是（　）？

```
public class Test {
    public static void main(String args[]) {
        String s1 = new String("Test");
        String s2 = new String("Test");
        if (s1 == s2)
            System.out.println("Same");
        if (s1.equals(s2))
            System.out.println("Equals");
    }
}
```

A　Same　　B　Equals　　C　SameEquals　　D　什么都不输出

笔试题

- StringBuilder和StringBuffer的区别，下面说法错误的是？（）

A StringBuffer是线程安全的

B StringBuilder是非线程安全的

C StringBuffer对 String 类型进行改变的时候其实都等同于生成了一个新的 String 对象，然后将指针指向新的 String 对象

D 字符串拼接效率比较String<StringBuffer<StringBuilder

笔试题

- 编写程序，实现以下功能：

  计算某个字符串中，某个字符出现的次数