

# Java常用工具\_API

## 课程概要

[API简介](#)

[Object类](#)

[Scanner类](#)

[String类](#)

[StringBuilder和StringBuffer类](#)

[Date和Calendar类](#)

[基本类型的包装类](#)

## 学习目标

理解API的概念和作用

能够根据需要找到API中对应的类和方法并使用

能够根据API文档中的描述正确使用Object类中的toString()和equals()方法

能够通过查阅API使用Scanner类的常用方法

能够通过查阅API使用String类的常用方法

能够通过查阅API使用StringBuilder/StringBuffer类的常用方法

能够通过查阅API使用Date/SimpleDateFormat类的常用方法

## API简介

### 什么是API?

Application Programming Interface，应用程序编程接口，这里指的是API文档，通常叫“Java文档”，是Java中提供的类的使用说明书。Java中的类和方法几乎全部都添加了文档注释（/\*\* 文档注释内容 \*/），这些文档注释就被Java的命令（javadoc.exe，在JDK安装的bin目录下）编译成了Java文档，即API。

OVERVIEW

MODULE

PACKAGE

CLASS

USE

TREE

DEPRECATED

INDEX

HELP

Java SE 11 & JDK 11

ALL CLASSES

SEARCH:

## Java® Platform, Standard Edition & Java Development Kit Version 11 API Specification

This document is divided into two sections:

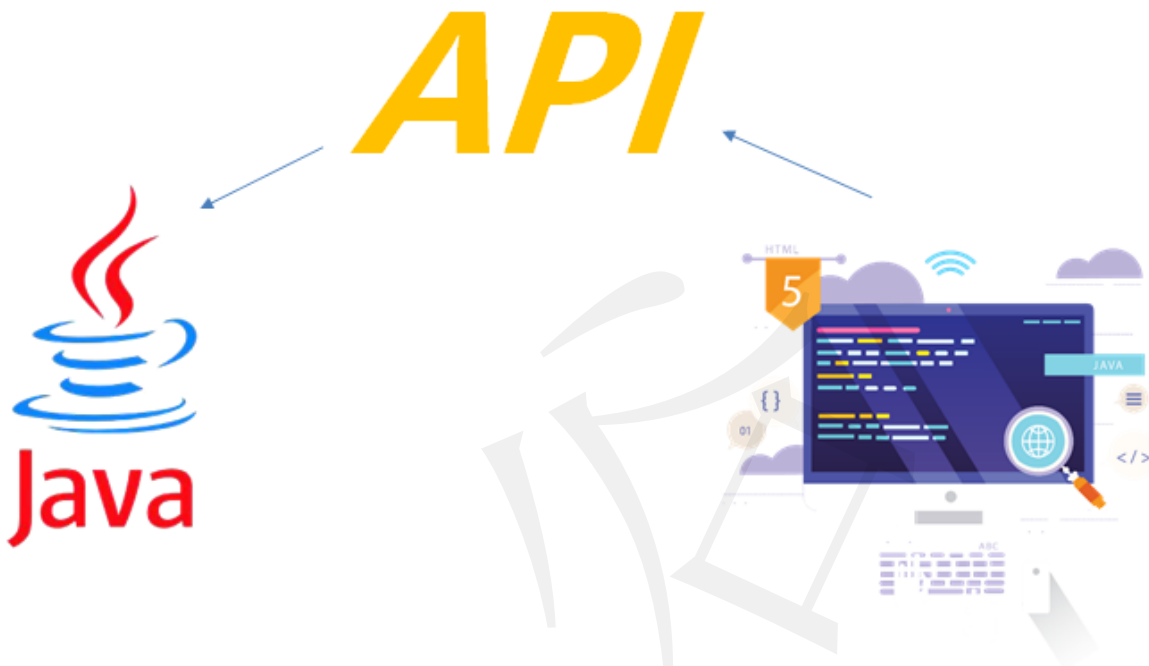
**Java SE**  
The Java Platform, Standard Edition (Java SE) APIs define the core Java platform for general-purpose computing. These APIs are in modules whose names start with `java`.

**JDK**  
The Java Development Kit (JDK) APIs are specific to the JDK and will not necessarily be available in all implementations of the Java SE Platform. These APIs are in modules whose names start with `jdk`.

All Modules	Java SE	JDK	Other Modules
Module	Description		
<code>java.base</code>	Defines the foundational APIs of the Java SE Platform.		
<code>java.compiler</code>	Defines the Language Model, Annotation Processing, and Java Compiler APIs.		
<code>java.datatransfer</code>	Defines the API for transferring data between and within applications.		
<code>java.desktop</code>	Defines the AWT and Swing user interface toolkits, plus APIs for accessibility, audio, imaging, printing, and JavaBeans.		
<code>java.instrument</code>	Defines services that allow agents to instrument programs running on the JVM.		
<code>java.logging</code>	Defines the Java Logging API.		
<code>java.management</code>	Defines the Java Management Extensions (JMX) API.		
<code>java.management.rmi</code>	Defines the RMI connector for the Java Management Extensions (JMX) Remote API.		
<code>java.naming</code>	Defines the Java Naming and Directory Interface (JNDI) API.		
<code>java.net.http</code>	Defines the HTTP Client and WebSocket APIs.		
<code>java.prefs</code>	Defines the Preferences API.		
<code>java.rmi</code>	Defines the Remote Method Invocation (RMI) API.		
<code>java.scripting</code>	Defines the Scripting API.		

## 为什么学习API文档？

发挥面向对象思想，找到Java提供的对象来实现功能，学习API文档就是学习Java中的类的使用方法。开发Java程序的过程中，想要使用Java提供的类的功能，就需要通过API文档来查找并使用。API是程序员和Java语言之间沟通的桥梁。



## API文档的使用方法

学习API其实就是学习如何使用JDK提供的类。本课时介绍的是最常用的几个，在后面的学习和工作中还要用到更多、更复杂的类。API文档的学习并不要求你把所有的类和方法都记住，也不关心这些类的具体实现，而是学习**如何使用**它们。随着你对编程的理解越来越深入，使用JDK中的类越来越多，翻阅API文档的次数和时间也会越来越多。你不需要记住Java提供的所有的类，但你一定要学会通过API文档熟练的查找并使用这些类。

### 使用步骤

#### 1. 搜索类

在搜索框里按类名模糊查询，或者通过文档顶部提供的层级结构查找。

#### 2. 查看类的包名

很多类在使用之前需要导包，所以要知道该类所属的包。

每一个类都默认导入 `java.lang` 包，所以如果你使用的是该包下的类，则无须手动导包。

#### 3. 查看类的注释

使用一个类前，要知道该类的大致功能和它的定位，要知道一个类的能力边界，以便快速确定该类是不是你真正需要的。

#### 4. 查看构造方法（如有）

使用类之前要创建类的对象。如果是静态类或者一些工具类如 `java.lang.Math` 类，则不需要这么做。

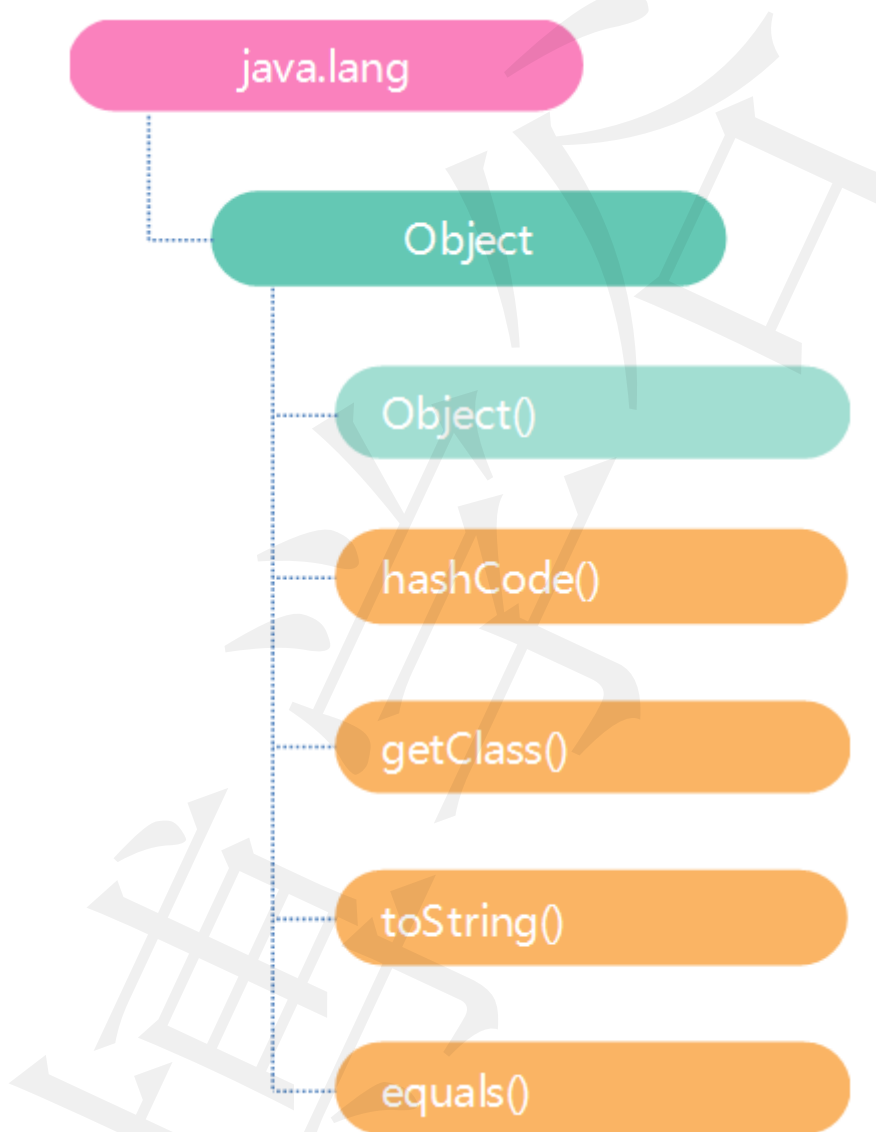
#### 5. 查找需要的方法并使用

根据标识符的命名规则，可以快速定位方法的位置。仔细阅读方法的注释，注意区分重载方法的使用有何不同。如果找不到想要的方法，可能该方法在父类，或者更高的父类中。

# Object类

## 简介

类层次结构最顶层的基类，所有类都直接或间接的继承自Object类，所以，所有的类都是一个Object（对象）。Java是严格的面向对象的语言，所以，所有的函数和数据都封装到类中（并非所有的编程语言都这样），通过类的对象来调用这些功能或实现某个算法。例外的情况是：Java的八种基本数据类型数据类型，它们由



## 构造方法

Object():

构造一个对象。所有子类对象初始化时都会优先调用该方法

## 成员方法

**int hashCode():**

返回对象的哈希码值，该方法通过对象的地址值进行计算，不同对象的返回值一般不同

## **Class<?> getClass():**

返回调用此方法对象的运行时类对象（调用者的字节码文件对象）

## **String toString():**

返回该对象的字符串表示

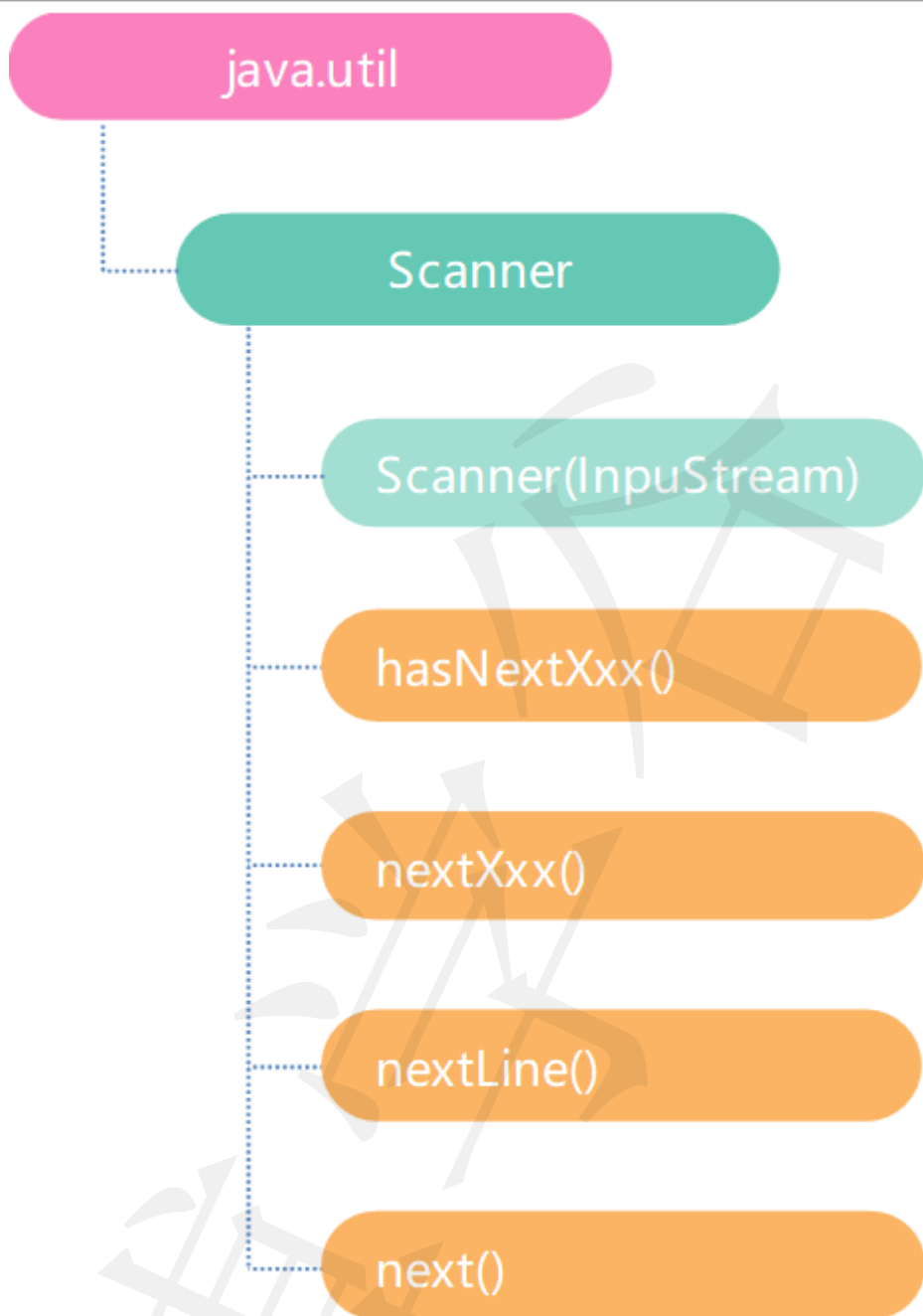
## **boolean equals()**

返回其它某个对象是否与此对象“相等”。默认情况下比较两个对象的引用（地址值），建议重写

# **Scanner类**

## **简介**

扫描器。能够解析字符串（String）和基本数据类型的数据



## 构造方法

`Scanner(InputStream)`: 构造一个扫描器对象，从指定输入流中获取数据参数`System.in`，对应键盘录入

## 成员方法

### `hasNextXxx()`:

判断是否还有下一个输入项，其中Xxx可能是任意基本数据类型，返回结果为布尔类型

### `nextXxx()`:

获取下一个输入项，其中Xxx可能是任意基本数据类型，返回对应类型的数据

### `String nextLine()`:

获取下一行数据。以换行符作为分隔符。

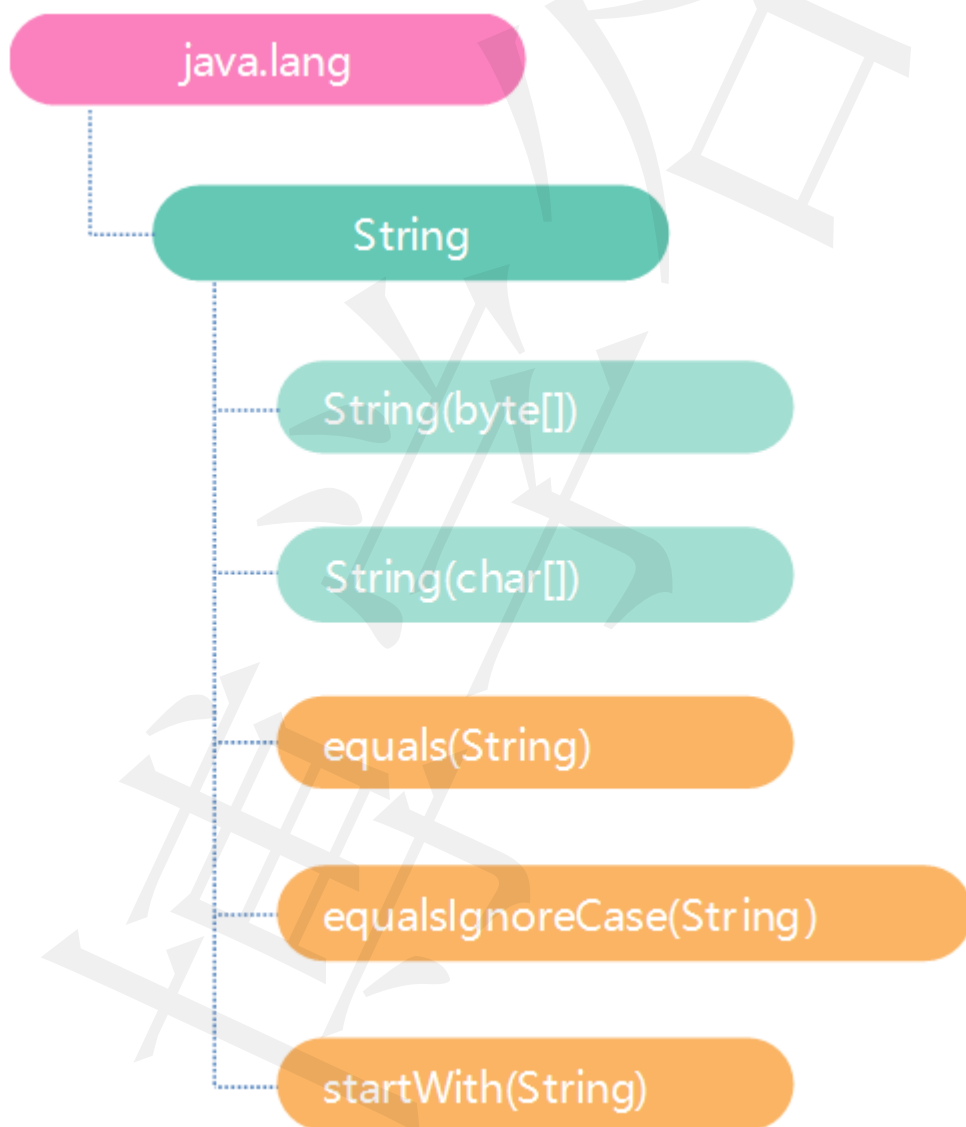
## String next()

获取下一个输入项，以空白字符作为分隔符 空白字符：空格、tab、回车等

# String类

## 简介

字符串。每一个字符串对象都是常量。



## 构造方法

### String(byte[]):

构造一个String对象，将指定字节数组中的数据转化成字符串

### String(char[]):

构造一个String对象，将指定字符数组中的数据转化成字符串

## 成员方法

### **boolean equals(String):**

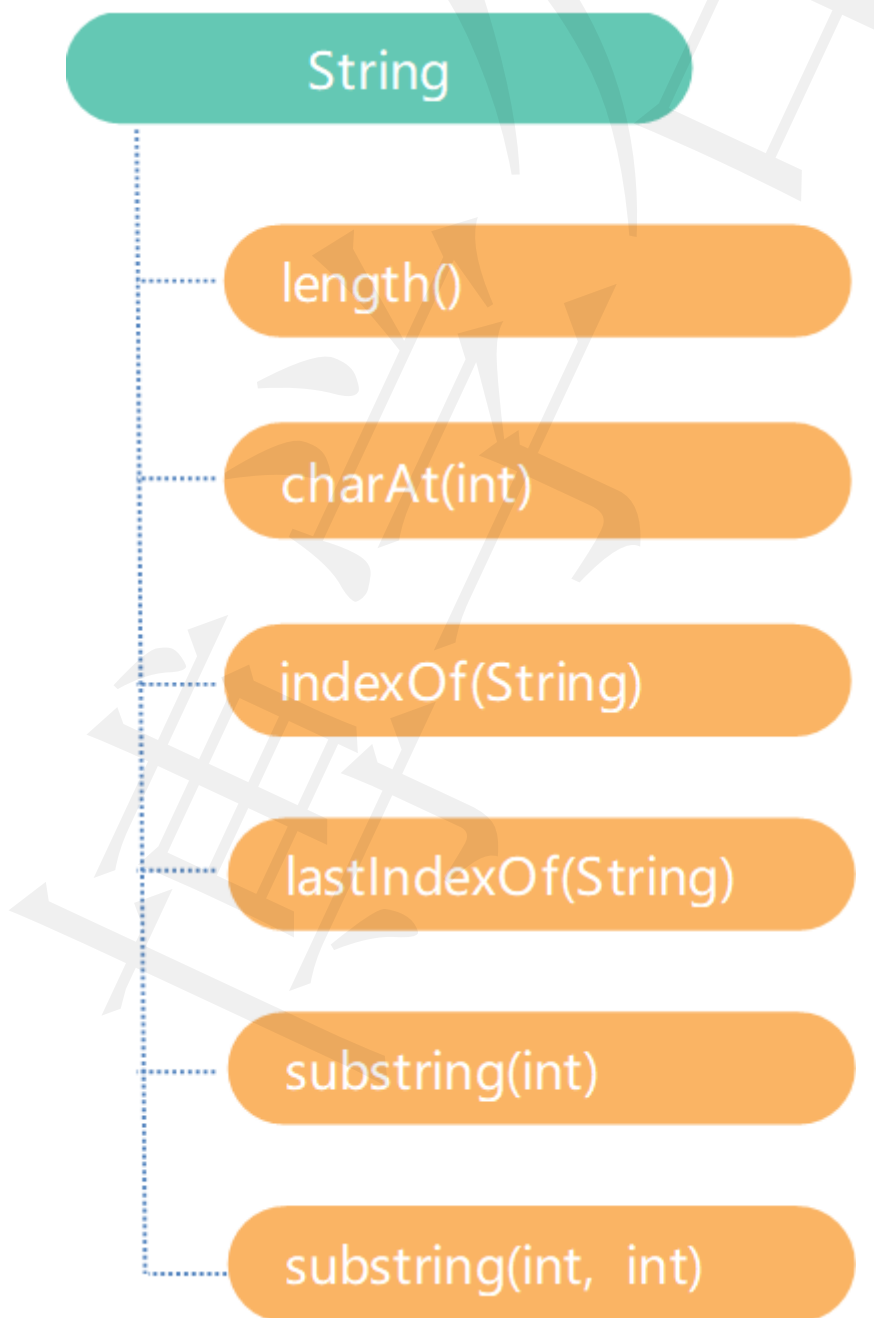
判断当前字符串与给定字符串是否相同，区分大小写

### **boolean equalsIgnoreCase(String):**

判断当前字符串与给定字符串是否相同，不区分大小写

### **boolean startsWith(String):**

判断是否以给定字符串开头



### **int length():**



获取当前字符串的长度

### **char charAt(int index):**

获取指定索引位置的字符

### **int indexOf(String):**

获取指定字符（串）第一次出现的索引

### **int lastIndexOf(String):**

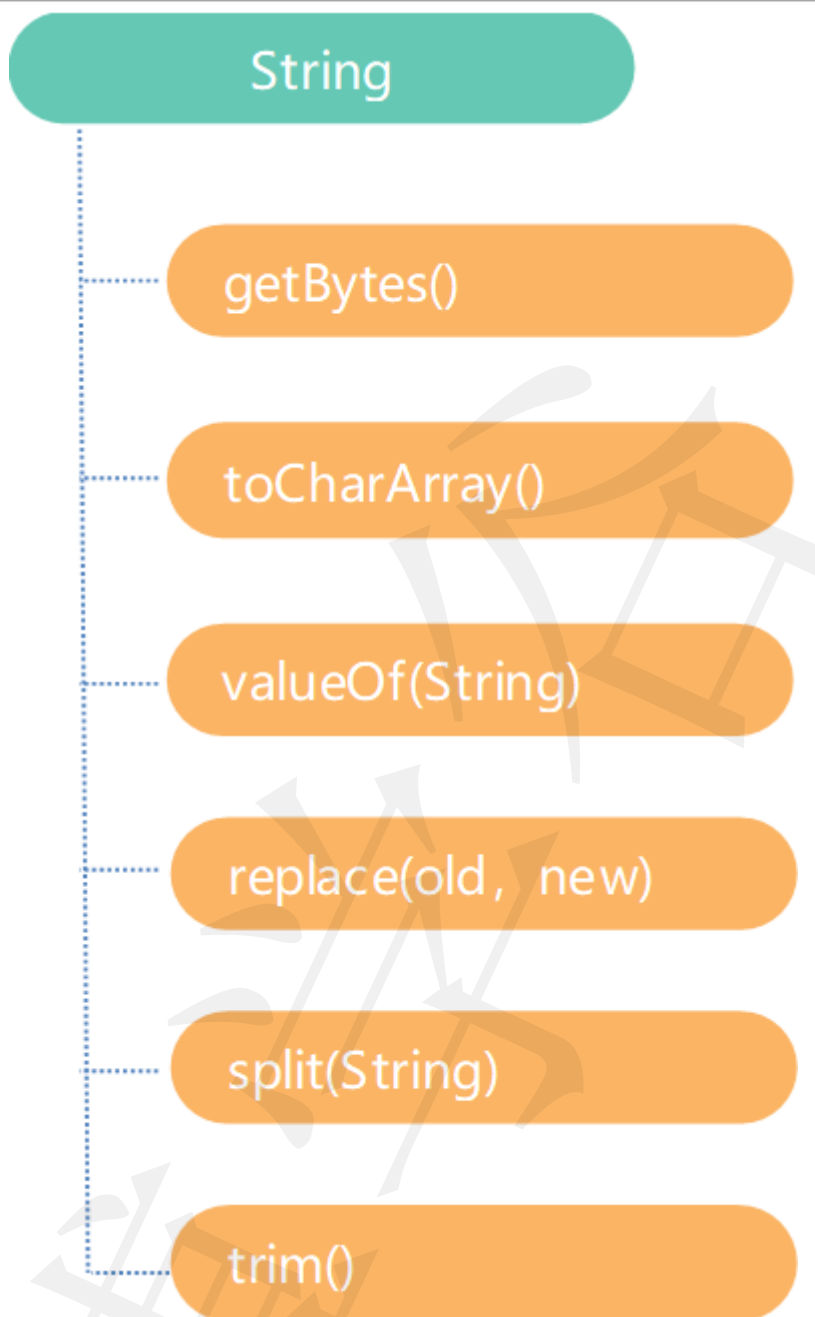
获取指定字符（串）最后一次出现的索引

### **String substring(int):**

获取指定索引位置（含）之后的字符串

### **String substring(int, int):**

获取从索引start位置（含）起至索引end位置（不含）的字符串



**byte[] getBytes():**

将字符串转换成字节数组

**char[] toCharArray():**

将字符串转换成字符数组

**static String valueOf(..) :**

将指定类型数据转换成字符串

**String replace(old, new):**

将指定字符（串）替换成新的字符（串）

**String[] split(String):**

切割字符串，返回切割后的字符串数据，原字符串不变

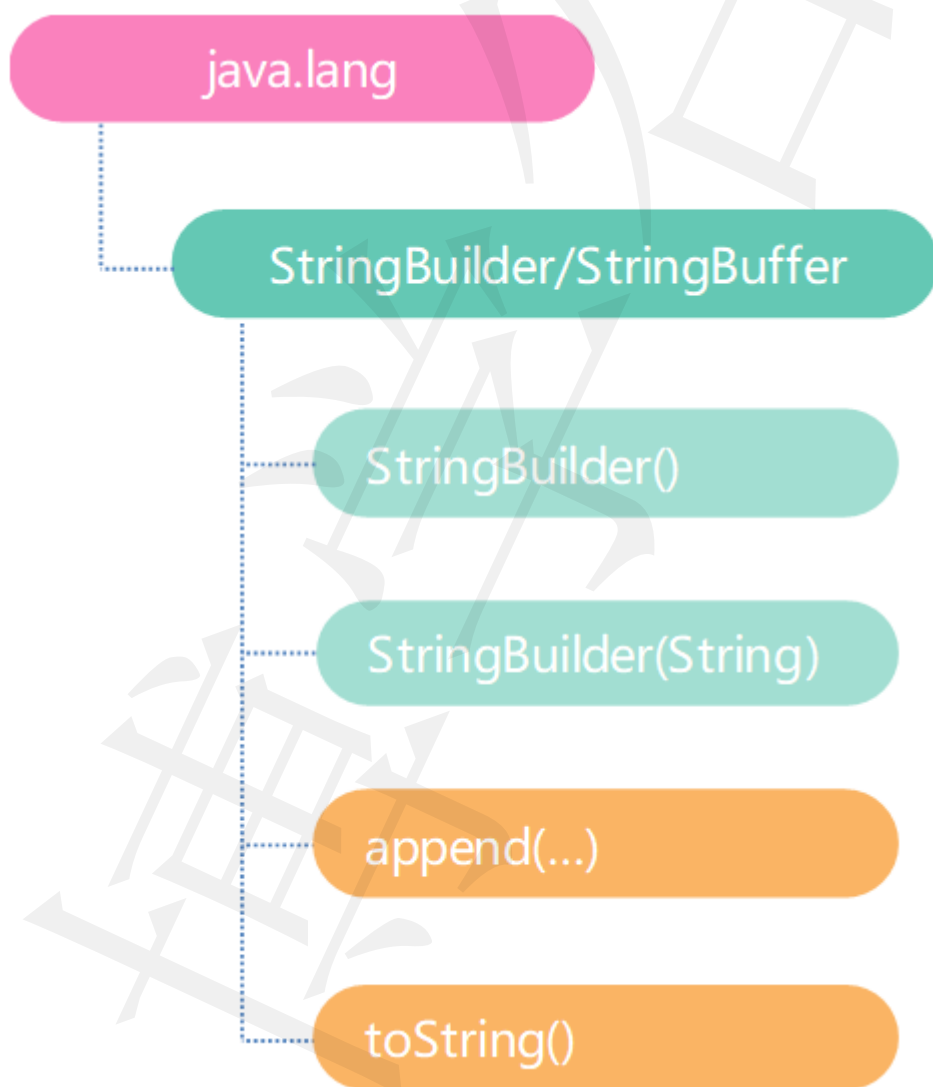
### **String trim():**

去掉字符串两端的空白字符

## **StringBuilder和StringBuffer类**

### **简介**

可变字符序列，用于构造字符串对象。内部使用自动扩容的数组操作字符串数据。StringBuilder和StringBuffer使用相同的API。



### **构造方法**

#### **StringBuilder():**

构造一个空的StringBuilder容器

#### **StringBuilder(String):**

构造一个StringBuilder容器，并添加指定字符串

## 成员方法

### StringBuilder append(...):

将任意数据添加到StringBuilder容器中

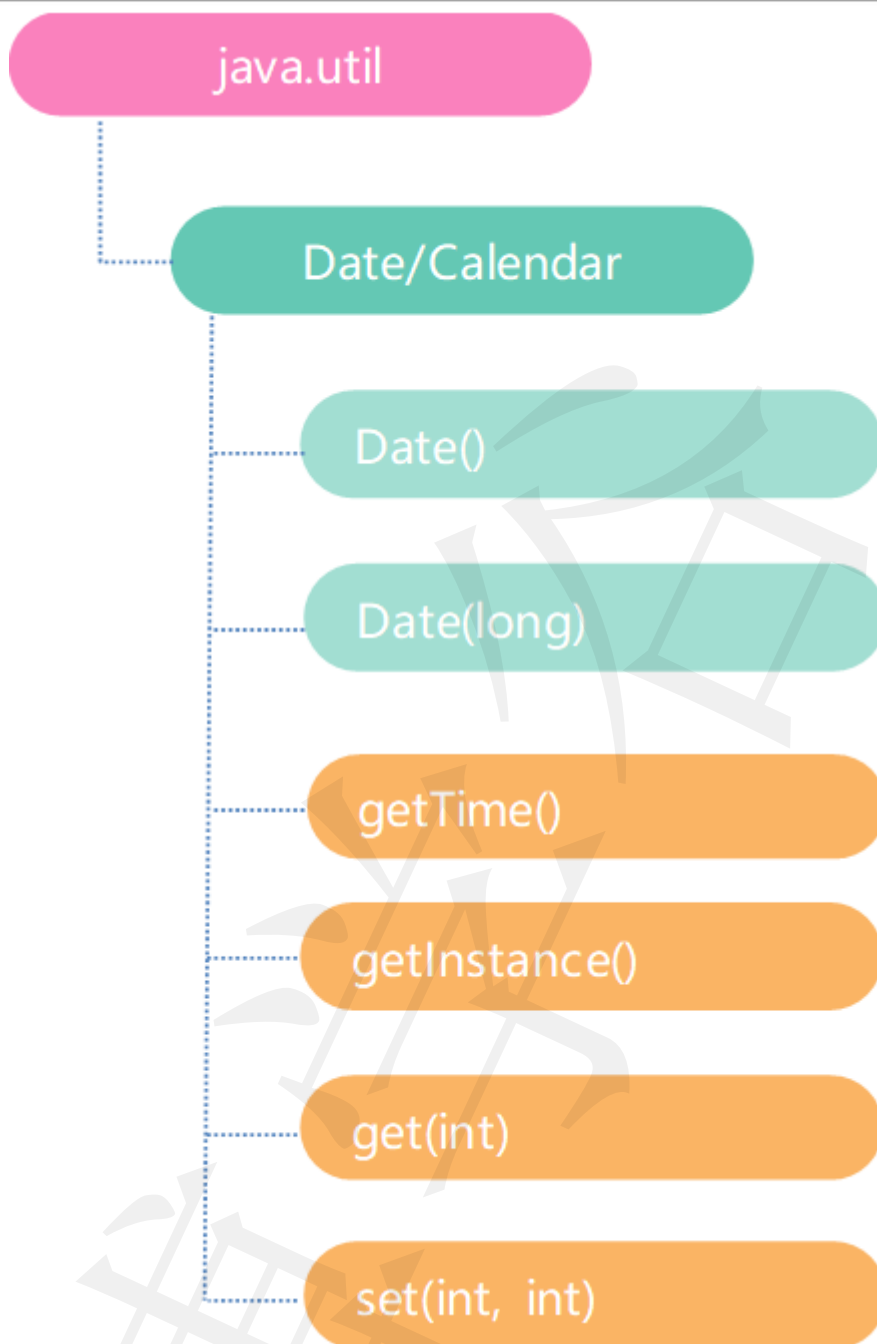
### String toString():

将当前StringBuilder容器转成字符串

## Date和Calendar类

### 简介

日期和日历类，用于操作日期相关信息。



## 构造方法

### Date():

构造一个日期对象，当前系统时间，精确到毫秒

### Date(long):

构造一个日期对象，时间为自“1970年1月1日00:00:00 GMT”起，至指定参数的毫秒数

## 成员方法

### long getTime():

将日期对象转换成对应时间的毫秒值

## static Calendar getInstance():

根据当前系统时区和语言环境获取日历对象

## int get(int field):

返回给定日历字段的值

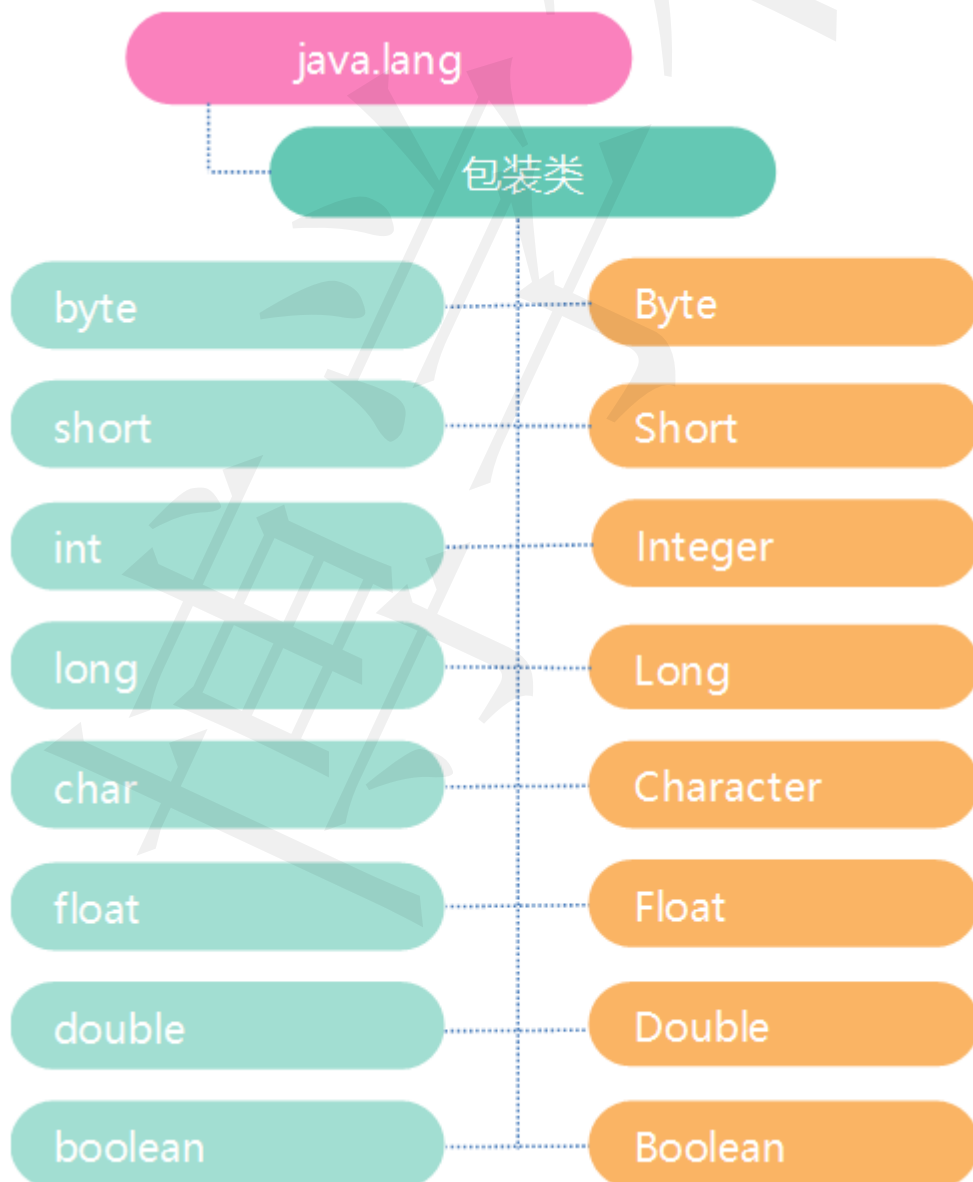
## void set(int field, int value):

将给定的日历字段设置为指定的值

## 基本类型的包装类

### 简介

基本数据类型不是对象，所以Java针对基本类型提供了对应的包装类，以对象的形式来使用。



## 装箱

基本类型转包装类型（对象类型）

## 拆箱

包装类型（对象类型）转基本类型

## 成员方法

**基本类型** `parseXxx(String)`:

将字符串类型的数据转换成对应的基本类型

