Java语言基础

【Day12】

String类的常用方法

String类的构造方法

- String类的构造方法如下：

String()	使用无参方式构造对象得到空字符序列
String(byte[] bytes, int offset, int length)	使用bytes数组中下标从offset位置开始的length个字节来构造对象
String(byte[] bytes)	使用bytes数组中的所有内容构造对象

String类的构造方法

- String类的构造方法如下：

String(char[] value, int offset, int count)	使用value数组中下标从offset位置开始的count个字符来构造对象
String(char[] value)	使用value数组中的所有内容构造对象
String(String original)	根据参数指定的字符串内容来构造对象，新创建对象为参数对象的副本

方法charAt和length的详解

- String定义有charAt方法：

char charAt(int index)	方法charAt用于返回字符串指定位置的字符。参数index表示指定的位置
int length()	返回字符串字符序列的长度

方法charAt和length的详解

- String类的基本方法如下：

```
String str = “上海自来水来自海上”;
for (int i = 0; i < str.length() / 2; i++) {
   if (str.charAt(i) != str.charAt(str.length() - 1 - i)) {
      System.out.println("不是回文");
      return;
    }
}
System.out.println("是回文");
```

代码用于检测字符串是否为“回文”；所谓回文是指一个字符序列无论从左向右读还是从右向左读都是相同的句子。

String类的基本方法

- String类的基本方法如下：

boolean contains(CharSequence s)	用于判断当前字符串是否包含参数指定的内容
String toLowerCase()	返回字符串的小写形式
String toUpperCase()	返回字符串的大写形式
String trim()	返回去掉前导和后继空白的字符串
boolean startsWith(String prefix)	判断字符串是否以参数字符串开头
boolean endsWith(String suffix)	判断字符串是否以参数字符串结尾

String类的基本方法

- String类的基本方法如下：

boolean equals(Object anObject)	用于比较字符串内容是否相等并返回
boolean equalsIgnoreCase(String anotherString)	用于比较字符串内容是否相等并返回，不考虑大小写，如：'A'和'a'是相等

String类中的equals方法

- String类重写了继承自Object的equals方法，其逻辑为：字符序列相同的String对象equals返回true。
- 通常使用equals方法判断两个字符串的字符序列是否相同。
- String还提供equalsIgnoreCase方法，该方法在判断时将忽略大小写。

应该调用直接量的equals而不是input的equals，因为如果调用input的equals，当input为null时将抛出异常。

```
Scanner scanner = new Scanner(System.in);
String input = scanner.next();
if("exit".equals(input)) {
    .... … …
}
```

如果需要忽略大小写判断，可以使用：“exit".equalsIgnoreCase(input)。

方法indexOf的详解

- indexOf方法用于实现在字符串中检索另外一个字符串。
- String提供几个重载的indexOf方法：

int indexOf(String str)	在字符串中检索str，返回其第一出现的位置，如果找不到则返回-1
int indexOf(String str, int fromIndex)	类似indexOf(String)，但是是从字符串的fromIndex位置开始检索

```
String str = "Thinking in Java";
int index = str.indexOf("Java");
System.out.println("index=" + index);
```

Thinking in Java
↑
12

index值为12，对应字符串中“J”的下标

方法substring的详解

- substring方法用于返回一个字符串的子字符串。
- substring常用重载方法定义如下：

String substring(int beginIndex, int endIndex)	返回字符串中从下标beginIndex（包括）开始到endIndex（不包括）结束的子字符串
String substring(int beginIndex)	返回字符串中从下标beginIndex（包括）开始到字符串结尾的子字符串

```
String str = "Thinking in Java";
String subStr = str.substring(9, 11);
System.out.println(subStr);
```

“前包括后不包括”的设计目的是使得后一个参数减去前一个参数的是值正好是截取子字符串的长度。

StringBuilder/StringBuffer

StringBuilder类和StringBuffer类的概述

- 和String对象不同， StringBuilder封装可变的字符串，对象创建后可以通过调用方法改变其封装的字符序列。
- StringBuilder和StringBuffer在手册上基本一样。
- 区别就是StringBuilder是非线程安全的，效率较高。而StringBuffer是线程安全的，效率较低

StringBuilder类的构造方法

- StringBuilder有如下常用构造方法：

  public StringBuilder()

  public StringBuilder(String str)

- 构造方法StringBuilder()将创建不含任何字符序列的StringBuilder 对象；而StringBuilder(String str) 将创建包含参数字符串str的StringBuilder对象。

StringBuilder类的常用方法

- StringBuilder类的常用方法如下：

StringBuilder insert(int offset, String str)	插入字符串
StringBuilder append(String str)	追加字符串
StringBuilder delete(int start，int end)	删除字符串
StringBuilder replace(int start，int end，String str)	替换字符串
StringBuilder reverse()	字符串反转

返回值的意义

- StringBuilder的很多方法的返回值均为StringBuilder类型。这些方法的返回语句均为：return this。
- 可见，这些方法在对StringBuilder所封装的字符序列进行改变后又返回了该对象的引用。基于这样的设计的目的在于可以按照如下简洁的方式书写代码：

```
sb.append("ibm").append("java").insert(3, "oracle").replace(9, 13, "JAVA");
System.out.println(sb.toString());
```

Date/SimpleDateFormat

Date类的概述

- java.util.Date 类用于封装日期及时间信息。
- Date类的大多数用于进行时间分量计算的方法已经被Calendar取代。

```
// 无参的构造方法，构造的Date对象封装当前的日期及时间信息。
Date date = new Date();
// Date类重新了toString方法，输出的字符串形如：
// Sun Jan 06 11:52:55 CST 2013
System.out.println(date);
long time = date.getTime();
date.setTime(time + 24 * 60 * 60 * 1000);
System.out.println(date);
```

getTime方法用于获取对应的毫秒数，即1970年1月1日距所此时间经历的毫秒。

setTime方法用于通过毫秒数设置时间。

SimpleDateFormat类的概述

- java.text.SimpleDateFormat类用于实现Date对象和字符串表示的日期信息间的转换。其构造方法如下：

SimpleDateFormat(String pattern)　参数用于说明格式的模式匹配字符串。

```
SimpleDateFormat sdf
    = new SimpleDateFormat("yyyy年MM月dd日");
Date date = new Date();
String dateStr = sdf.format(date);
System.out.println(dateStr);
```

format方法用于将日期数据按照指定的格式转换为字符串。

SimpleDateFormat类的概述

- parse方法用于按照特定格式将表示时间的字符串转换为Date对象

```
String dateStr = "2013-01-06";
String pattern = "yyyy-MM-dd";
SimpleDateFormat sdf =
        new SimpleDateFormat(pattern);
Date date = sdf.parse(dateStr);
System.out.println(date);
```

如果字符串格式与pattern不匹配，该方法将抛出异常

SimpleDateFormat类的概述

- 常用格式字符串

字符	含义	示例
y	年	yyyy年—2013年；yy—13年
M	月	MM月—01月；M月—1月
d	日	dd日—06日；d日—6日
H	小时（24小时制）	a h时—下午 12时 HH:mm:ss—12:46:33 hh(a):mm:ss—12(下午):47:48
h	小时（12小时制）	
m	分钟	
s	秒	

Calendar类

Calendar类的概述

- java.util.Calendar 类用于封装日历信息，其主要作用在于其方法可以对时间分量进行运算。
- Calendar是抽象类，其具体子类针对不同国家的日历系统，其中应用最广泛的是GregorianCalendar（格里高利历），对应世界上绝大多数国家/地区使用的标准日历系统。

```
Calendar c = Calendar.getInstance();
```

通常使用Calendar的静态方法getInstance获得Calendar对象；
getInstance方法将根据系统地域信息返回不同的Calendar类的实现

总结与答疑

技术资料

IT兄弟连

ITXDL.CN

变态严管　让学习成为一种习惯