第 8 章 常用实用类

8.1 String 类

1. String 类无子类。

2. 创建字符串对象

（1）字符串常量是对象，住在常量池。

（2）声明字符串对象名，（通过使用 new）引入字符串对象。这种字符串对象住在动态区。

String 字符串对象名;

字符串对象名 = new String(字符串常量或另一个字符串对象名（该字符串对象名需已指向字符串对象）或字符型数组名（字符型数组的元素需已赋值）);

（3）可以将字符串常量的引用赋给字符串对象名。

字符串对象名 = 字符串常量;

（4）字符串对象通常简称为字符串。

3. 字符串的并置（连接）

（1）并置运算符“+”

（2）若参与并置的都是字符串常量（无字符串对象名），则结果字符串住在常量池，否则住在动态区。

pp. 177-178 代码 4

```
public class Example8_1 {
  public static void main(String args[]) {
      String hello = "你好";
      String testOne = "你"+"好";                  //【代码 1】
      System.out.println(hello == testOne);      //输出结果是 true
      System.out.println("你好" == testOne);      //输出结果是 true
      System.out.println("你好" == hello);        //输出结果是 true
```

```
      String you = "你";
      String hi = "好";
      String testTwo = you+hi;                   //【代码 2】
      System.out.println(hello == testTwo);      //输出结果是 false
      String testThree = you+hi;
      System.out.println(testTwo == testThree); //输出结果是 false
   }
}
```

4. 常用方法

（1）length()

返回当前字符串的长度。

注意数组有一个属性叫 length。

（2）equals(String s)

判断当前字符串是否与 s 所指字符串相等。

equalsIgnoreCase(String s)

判断当前字符串是否与 s 所指字符串相等（大小写等价）。

p. 179 代码 2

```
public class Example8_2 {
  public static void main(String args[]) {
      String s1,s2;
      s1 = new String("天道酬勤");
      s2 = new String("天道酬勤");
      System.out.println(s1.equals(s2));    //输出结果是: true
      System.out.println(s1==s2);           //输出结果是: false
      String s3,s4;
      s3 = "we are students";
      s4 = new String("we are students");
      System.out.println(s3.equals(s4));    //输出结果是: true
      System.out.println(s3==s4);           //输出结果是: false
      String s5,s6;
      s5 = "勇者无敌";
      s6 = "勇者无敌";
      System.out.println(s5.equals(s6));    //输出结果是: true
      System.out.println(s5==s6);           //输出结果是: true
    }
}
```

（3）startsWith(String s)

判断当前字符串的前缀是否与 s 所指字符串相等。

endsWith(String s)

判断当前字符串的后缀是否与 s 所指字符串相等。

（4）compareTo(String s)

比较当前字符串与 s 所指字符串的大小。

compareToIgnoreCase(String s)

比较当前字符串与 s 所指字符串的大小（大小写等价）。

p.180 代码 2

```
public class SortString {
  public static void sort(String a[]) {
    int count = 0;
    for(int i=0;i<a.length-1;i++) {
       for(int j=i+1;j<a.length;j++) {
          if(a[j].compareTo(a[i])<0) {
             String temp = a[i];
             a[i] = a[j];
             a[j] = temp;
          }
       }
    }
  }
}
```

pp.180-181 代码 3

```
import java.util.*;
public class Example8_3 {
   public static void main(String args[]) {
      String [] a = {"melon","apple","pear","banana"};
      String [] b = {"西瓜","苹果","梨","香蕉"};
      System.out.println("使用 SortString 类的方法按字典序排列数组 a:");
      SortString.sort(a);
```

```
        for(int i=0;i<a.length;i++) {
           System.out.print("  "+a[i]);
        }
        System.out.println("");
        System.out.println("使用类库中的 Arrays 类，按字典序排列数组 b:");
        Arrays.sort(b);
        for(int i=0;i<b.length;i++) {
           System.out.print("  "+b[i]);
        }
     }
  }
```

（5）contains(String s)

判断 s 所指字符串是否为当前字符串的子串，即模式匹配。

（6）indexOf(String s)

判断 s 所指字符串是否为当前字符串的子串，若是，则返回第一次出现的索引位置，否则返回-1。

lastIndexOf(String s)

判断 s 所指字符串是否为当前字符串的子串，若是，则返回最后一次出现的索引位置，否则返回-1。

indexOf(String s, int startpoint)

判断 s 所指字符串是否为“当前字符串从索引位置 startpoint 开始一直到最后的子串”的子串，若是，则返回第一次出现的索引位置，否则返回-1。

（7）substring(int startpoint)

返回当前字符串从索引位置 startpoint 开始一直到最后的子串。

substring(int start, int end)

返回当前字符串从索引位置 start 开始一直到索引位置(end-1)为止的子串。

（8）trim()

返回当前字符串的去除了前后空格的子串。

5. 字符串与基本数据的相互转化

（1）可以使用 parseInt(String s)，parseByte(String s)，parseShort(String s)，parseLong(String s)，parseFloat(String s)和 parseDouble(String s)方法将看上去是数值的字符串转化为数值。注意这些方法都不是 String 类的。

（2）可以使用 valueOf(int n)，valueOf(byte n)，valueOf(short n)，valueOf(long n)，valueOf(float n)和 valueOf(double n)方法将数值转化为字符串。

（3）利用字符串与基本数据的相互转化，有时可以编出极其精彩的程序。

pp.182-183 代码 4

```
public class Example8_4 {
   public static void main(String args[]) {
      double sum=0,item=0;
      boolean computable=true;
      for(String s:args) {
        try{ item=Double.parseDouble(s);
             sum=sum+item;
        }
        catch(NumberFormatException e) {
             System.out.println("您输入了非数字字符:"+e);
```

```
                computable=false;
            }
        }
        if(computable)
            System.out.println("sum="+sum);
    }
}
```

6. 对象的字符串表示

（1）toString()方法返回当前对象的字符串表示，即“当前对象所属类的名字@当前对象的引用”。

（2）可以重写 toString()方法，如 Date 类就重写了该方法。

pp.183-184 代码 2

```
public class TV {
    double price ;
    public void setPrice(double m) {
        price = m;
    }
    public String toString() {
        String oldStr = super.toString();
        return oldStr+"\n 这是电视机，价格是:"+price;
    }
}
```

```
public class Example8_5 {
    public static void main(String args[]) {
        Date date = new Date();
        System.out.println(date.toString());
        TV tv = new TV();
        tv.setPrice(5897.98);
        System.out.println(tv.toString());
    }
}
```

7. 字符串与字符型数组、字节型数组

8. 正则表达式及字符串的替换与分解

（1）正则表达式

①包含元语言字符和限定修饰符的字符串称为正则表达式。

②一个正则表达式可以表达一类字符串。

③matches(String regex)方法返回当前字符串是否与正则表达式 regex 匹配。

（2）字符串的替换

replaceAll(String regex, String replacement)

（3）字符串的分解

split(String regex)

8.2 StringTokenizer 类

1. StringTokenizer 类有两个构造方法，一个使用默认分隔标记（空格等），另一个使用指定分隔标记（注意不是正则表达式）。

2. StringTokenizer 类的对象称为字符串分析器。

8.3 Scanner 类

既可以通过默认分隔标记（空格等）分解字符串，也可以通过指定分隔标记（正则表达式）分解字符串。

8.5 Date 类与 Calendar 类

8.7 Math 类、BigInteger 类和 Random 类