

第一章 String类

1.1 String类概述

1.1.1 概述

`java.lang.String` 类代表字符串，Java程序中所有的字符串文字都可以看作是此类实例

类 `String` 中包括用于检查各个字符串的方法，比如用于**比较字符串**，**搜索字符串**，**提取子字符串**以及创建具有翻译为**大写**或**小写**的所有字符的字符串副本

1.1.2 特点

1. 字符串不变：字符串的值在创建后不能被更改

```
String s1 = "abc";
s1 += "d";
System.out.println(s1); //"abcd"
//内存中有"abc", "abcd"两个对象，s1从指向"abc"，改为指向"abcd"
```

2. 因为 `String` 对象是不可变的，所以他们可以被共享

```
String s1 = "abc";
String s2 = "abc"
//内存中只有一个"abc"对象，同时被s1和s2共享
```

3. `"abc"` 等效于 `char[] data={'a','b','c'}`

```
String str = "abc";
//等价于
char data[] = {'a','b','c'};
String str = new String(data);
```

1.2 使用步骤

- 查看类
 - `java.lang.String`：此类不需要导入
- 查看构造方法
 - `public String()`：初始化新创建的 `String` 对象，以使其表示空字符序列
 - `public String(char[] value)`：通过当前参数中的字符数组来构造新的 `String`
 - `public String(byte[] bytes)`：通过使用平台的默认字符集解码当前参数中的字节数组来构造新的 `String`
 - 例：

```
//无参构造
String str1 = new String();

//通过字符数组构造
char chars[] = {'a','b','c'};
String str2 = new String(chars);

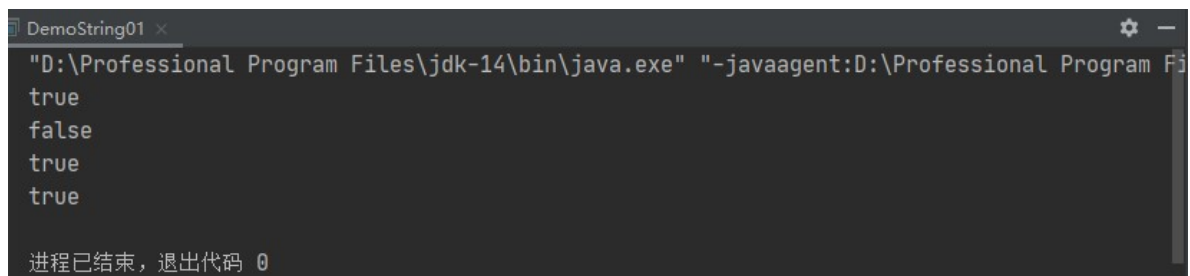
//通过字节数组构造
byte bytes[] = {87,98,99};
String str3 = new String(bytes);
```

1.3 常用方法

1.3.1 判断功能的方法

- `public boolean equals(Object anObject)`: 将此字符串与指定对象进行比较
 - `Object` 是“对象”的意思，也是一种引用类型。作为参数类型，表示任意对象都可以传递到方法中
- `public boolean equalsIgnoreCase(String anotherString)`: 将此字符串与指定对象进行比较，忽略大小写

```
public class DemoString01 {
    public static void main(String[] args) {
        String s1 = "hello";
        String s2 = "hello";
        String s3 = "HELLO";
        System.out.println(s1.equals(s2));
        System.out.println(s1.equals(s3));
        System.out.println(s1.equalsIgnoreCase(s2));
        System.out.println(s1.equalsIgnoreCase(s3));
    }
}
```



```
DemoString01 x
"D:\Professional Program Files\jdk-14\bin\java.exe" "-javaagent:D:\Professional Program F
true
false
true
true
进程已结束，退出代码 0
```

1.3.2 获取功能的方法

- `public int length()`: 返回此字符串的长度
- `public String concat(String str)`: 将指定的字符串连接到该字符串的末尾
- `public char charAt(int index)`: 返回指定索引处的char值
- `public int indexOf(String str)`: 返回指定子字符串第一次出现在该字符串内的索引
- `public String substring(int beginIndex)`: 返回一个子字符串，从 `beginIndex` 开始截取字符串到字符串结尾
- `public String substring(int beginIndex, int endIndex)`: 返回一个子字符串，从 `beginIndex` 到 `endIndex` 截取字符串，含 `beginIndex`，不含 `endIndex`

```

public static void main(String[] args) {
    String s1 = "HelloWorld";
    //public int length()
    System.out.println("字符串长度: " + s1.length());
    //public String concat(String str)
    String s2 = s1.concat("!!!");
    System.out.println(s2);
    //public char charAt(int index)
    System.out.println(s1.charAt(1));
    //public int indexOf(String str)
    System.out.println(s1.indexOf("l"));
    //public String substring(int beginIndex)
    System.out.println(s1.substring(3));
    //public String substring(int beginIndex, int endIndex)
    System.out.println(s1.substring(1,4));
}

```

```

D:\Professional Program Files\jdk-14\bin\java.exe" "-javaagent:D:\Professional Program Files\JetBrai
字符串长度: 10
HelloWorld!!!
e
2
loWorld
ell

进程已结束，退出代码 0

```

1.3.3 转换功能的方法

- `public char[] toCharArray()`: 将此字符串转换成新的字符数组
- `public byte[] getBytes()`: 使用平台的默认字符集将该 `String` 编码转换为新的字节数组
- `public String replace(CharSequence target, CharSequence replacement)`: 将与 `target` 匹配的字符串使用 `replacement` 字符串替换

```

public static void main(String[] args) {
    String s = "HelloWorld";
    char[] c = s.toCharArray();
    for (int i = 0; i < c.length; i++)
        System.out.print(c[i] + "\t");
    System.out.println();
    byte[] b = s.getBytes();
    for (int i = 0; i < b.length; i++)
        System.out.print(b[i] + "\t");
    System.out.println();
    String s2 = s.replace("World", "world");
    System.out.println(s2);
}

```

```

D:\Professional Program Files\jdk-14\bin\java.exe" "-javaagent:D:\Professional Program Files\JetBrai
H e l l o W o r l d
72 101 108 108 111 87 111 114 108 100
Helloworld

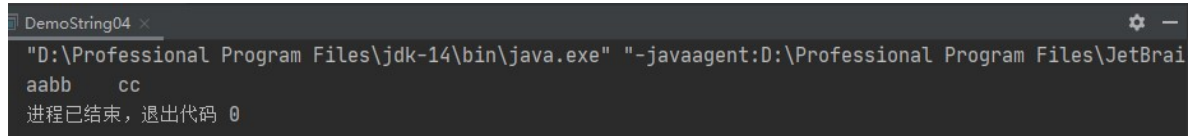
进程已结束，退出代码 0

```

1.3.4 分割功能的方法

- `public String[] split(String regex)`: 将此字符串按照给定的 `regex` (规则) 拆分为字符串数组

```
public static void main(String[] args) {
    String s = "aabb,cc";
    String[] strArray = s.split(",");
    for (int i = 0; i < strArray.length; i++)
        System.out.print(strArray[i]+"\t");
}
```

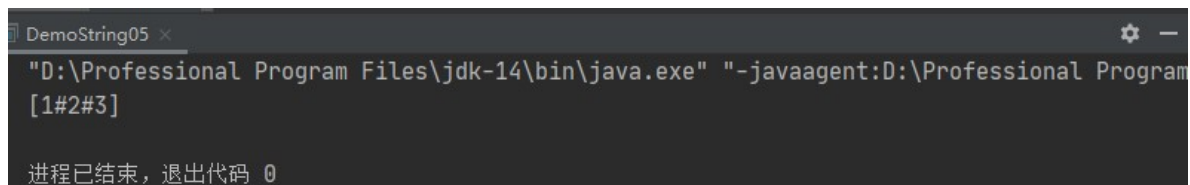


1.4 练习

1.4.1 拼接字符串

定义一个方法，把数组 (1,2,3) 按照指定格式拼接成一个字符串，格式如下: [word1#word2#word3]

```
public class DemoString05 {
    public static void main(String[] args) {
        int[] arr = {1,2,3};
        System.out.println(arrayToString(arr));
    }
    public static String arrayToString(int[] arr) {
        String s = new String("");
        for (int i = 0; i < arr.length - 1; i++)
            s = s.concat(arr[i] + "#");
        s = s.concat(arr[arr.length-1] + "");
        return s;
    }
}
```



1.4.2 统计字符个数

键盘录入一个字符，统计字符串中大小写字母及数字字符个数

```
public class DemoString06 {
    public static void main(String[] args) {
        System.out.print("请输入一个字符串:");
        String s = new String(new Scanner(System.in).next());
        System.out.println("字符串: " + s);
        int bigcount = 0;
        int smallcount = 0;
        int numbercount = 0;
        for (int i = 0; i < s.length(); i++) {
            char c = s.charAt(i);
```

```

        if(c >= 'A' && c <= 'Z')
            bigcount++;
        else if(c >= 'a' && c <= 'z')
            smallcount++;
        else if(c >= '0' && c <= '9')
            numbercount++;
        else {
            System.out.println("该字符" + c + "非法");
            return;
        }
    }
    System.out.println("大写字母: " + bigcount);
    System.out.println("小写字母: " + smallcount);
    System.out.println("数字: " + numbercount);
}
}

```

```

D:\Professional Program Files\jdk-14\bin\java.exe" "-javaagent:D:\Professional Program
请输入一个字符串:132asDFAsSfs547SAs
字符串: 132asDFAsSfs547SAs
大写字母: 6
小写字母: 6
数字: 6

进程已结束, 退出代码 0

```

第二章 Static关键字

关于 `static` 关键字的使用，它可以用来修饰成员变量和成员方法，被修饰的成员是**属于类**的，而不是单单属于某个对象的，也就是说，可以不依靠创建对象来调用

2.1 定义和使用格式

2.1.1 类变量

当 `static` 修饰成员变量时，该变量称为**类变量**。该类的每个对象都**共享**同一个类变量的值。任何对象都可以更改该类变量的值，但也可以在不创建该类的对象的情况下对类变量进行操作

- **类变量**：使用 `static` 关键字修饰的成员变量

定义格式：

```
static 数据类型 变量名;
```

例：

```

public class Student {
    private String name;
    private int age;
    private int sid; //学生id
    //类变量，记录学生数量，分配学号
    public static int numberOfStudent = 0;
    //有参构造方法
}

```

```

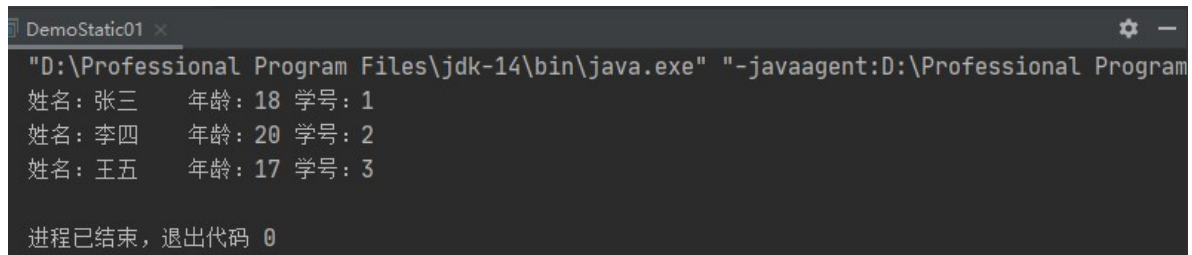
public Student(String name, int age) {
    this.name = name;
    this.age = age;
    this.sid = ++numberOfStudent;
}
//打印属性值
public void show() {
    System.out.println("姓名: " + name + "\t年龄: " + age + "\t学号: " + sid);
}
}

```

```

public class DemoStatic01 {
    public static void main(String[] args) {
        Student s1 = new Student("张三",18);
        Student s2 = new Student("李四",20);
        Student s3 = new Student("王五",17);
        s1.show();
        s2.show();
        s3.show();
    }
}

```



```

DemoStatic01 x
"D:\Professional Program Files\jdk-14\bin\java.exe" "-javaagent:D:\Professional Program
姓名: 张三    年龄: 18 学号: 1
姓名: 李四    年龄: 20 学号: 2
姓名: 王五    年龄: 17 学号: 3

进程已结束, 退出代码 0

```

2.1.2 静态方法

当 `static` 修饰成员方法时，该方法称为**类方法**。静态方法在声明中有 `static`，建议使用类名来调用，而不需要创建类的对象

- **类方法**：使用 `static` 关键字修饰的成员方法，习惯称为**静态方法**

定义格式：

```

修饰符 static 返回值类型 方法名 (参数列表){
    //执行语句
}

```

- 静态方法调用的注意事项
 - 静态方法可以直接访问类变量和静态方法
 - 静态方法**不能直接访问**普通成员变量或成员方法，反之，成员方法可以直接访问类变量或静态方法
 - 静态方法中不能使用 `this` 关键字

2.1.3 调用格式

被 `static` 修饰的成员可以且建议通过**类名直接访问**。虽然也可以通过对象名访问静态成员，原因即多个对象均属一个类，共享使用同一个静态成员，但是不建议，会出现警告信息

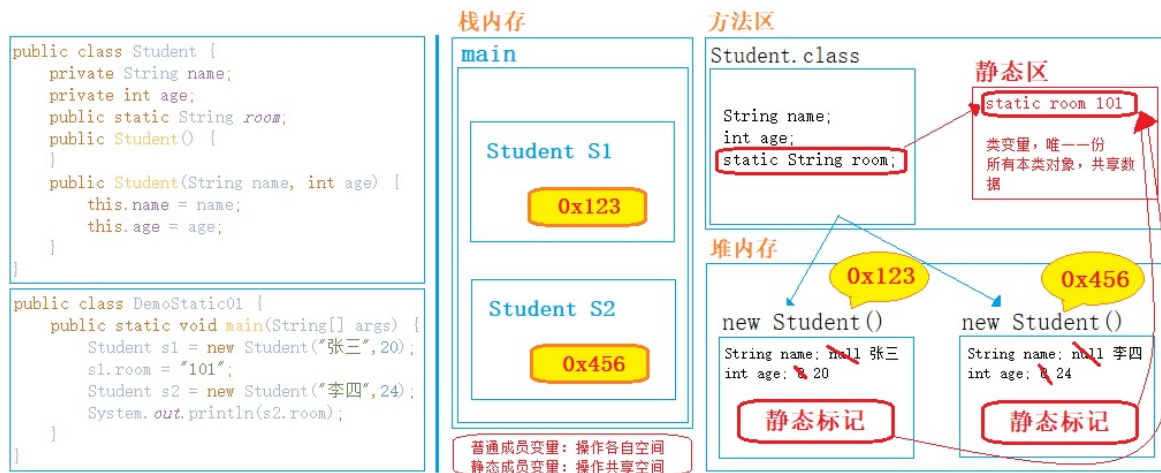
格式：

```
//访问变量
类名.类变量名;
//调用静态方法
类名.静态方法名(参数);
```

2.2 静态原理图解

static 修饰的内容:

- 是随着类的加载而加载的，且只加载一次
- 存储于一块固定的内存区域（静态区），所以，可以直接被类名调用
- 它优于对象存在，所以，可以被所有对象共享



2.3 静态代码块

- **静态代码块:** 定义在成员位置，使用 `static` 修饰的代码块
 - 位置：类中方法外
 - 执行：随着类的加载而执行且只执行一次，优先于 `main` 方法和构造方法的执行

格式:

```
public class ClassName{
    static{
        //执行语句
    }
}
```

- 作用：给类变量进行初始化赋值

例:

```

public class Game{
    public static int number;
    public static ArrayList<String> list;
    static{
        number = 2;
        list = new ArrayList<String>();
        list.add("张三");
        list.add("李四");
    }
}

```

- 注：static 关键字，可以修饰变量，方法和代码块。在使用的过程中，其主要目的还是想在不创建对象的情况下，去调用方法

第三章 Arrays 类

java.util.Arrays 此类包含用来操作数组的各种方法，比如排序和搜索等。其所有方法均为静态方法，调用起来非常简单

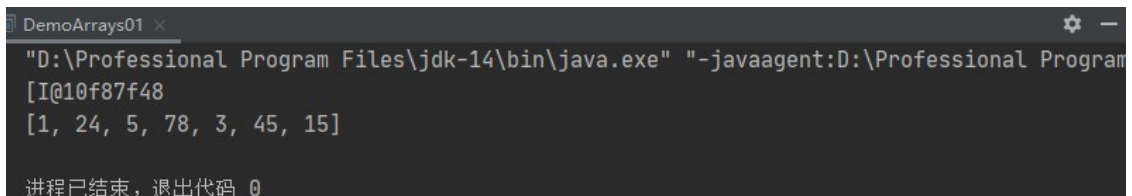
3.1 操作数组的方法

- public static String toString(int[] a)：返回指定数组内容的字符串表示形式

```

public static void main(String[] args) {
    int[] arr = {1,24,5,78,3,45,15};
    System.out.println(arr);
    String s = Arrays.toString(arr);
    System.out.println(s);
}

```



```

DemoArrays01 x
"D:\Professional Program Files\jdk-14\bin\java.exe" "-javaagent:D:\Professional Program
[I@10f87f48
[1, 24, 5, 78, 3, 45, 15]
进程已结束，退出代码 0

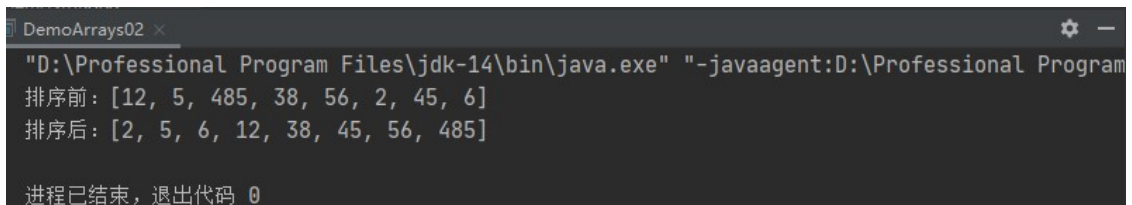
```

- public static void sort(int[] a)：对指定的 int 类型数组按数字升序进行排序

```

public static void main(String[] args) {
    int[] arr = {12,5,485,38,56,2,45,6};
    System.out.println("排序前: " + Arrays.toString(arr));
    Arrays.sort(arr);
    System.out.println("排序后: " + Arrays.toString(arr));
}

```



```

DemoArrays02 x
"D:\Professional Program Files\jdk-14\bin\java.exe" "-javaagent:D:\Professional Program
排序前: [12, 5, 485, 38, 56, 2, 45, 6]
排序后: [2, 5, 6, 12, 38, 45, 56, 485]
进程已结束，退出代码 0

```


3.2 练习

使用 Arrays 相关的API，将一个随机字符串中的所有字符升序排列，并倒序打印

```
public static void main(String[] args) {
    String line = "asljofwninJSOAnfosanfPnSON";
    char[] chars = line.toCharArray();
    Arrays.sort(chars);
    for (int i = chars.length - 1; i >= 0; i--)
        System.out.print(chars[i] + "\t");
}
```

```
DemoArrays03 x
"D:\Professional Program Files\jdk-14\bin\java.exe" "-javaagent:D:\Professional Program Files\JetBrains\Intel
w s s p o o n n n n l j i f f f a a S S 0 0 N N J A
进程已结束，退出代码 0
```

第四章 Math类

`java.lang.Math` 类包含用于执行基本数学运算的方法，如初等指数、对数、平方根和三角函数。类似这样的工具类，其所有方法均为静态方法，并且不会创建对象，调用起来非常简单

4.1 基本运算的方法

- `public static double abs(double a)`: 返回 double 值的绝对值

```
double d1 = Math.abs(-5); //d1=5
```

- `public static double ceil(double a)`: 返回大于等于参数的最小的整数

```
double d1 = Math.ceil(3.3); //d1=4
```

- `public static double floor(double a)`: 返回小于等于参数最大的整数

```
double d1 = Math.floor(3.3); //d1=3
```

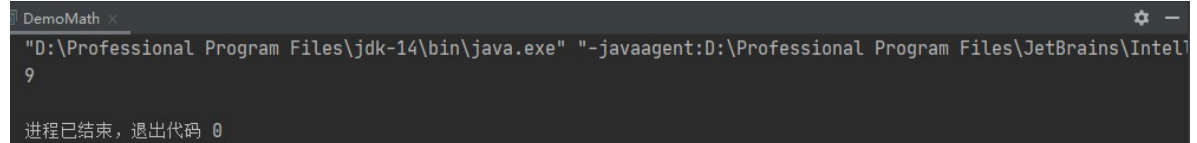
- `public static long round(double e)`: 返回最接近参数的long (相当于四舍五入方法)

```
long d1 = Math.round(5.5); //d1=6
```

4.2 练习

请使用 `Math` 相关的API, 计算在 `-10.8` 到 `5.9` 之间, 绝对值大于 `6` 或者小于 `2.1` 的整数有多少个

```
public class DemoMath {  
    public static void main(String[] args) {  
        int count = 0;  
        for (double i = Math.ceil(-10.8); i < Math.floor(5.9); i++) {  
            if(Math.abs(i)>6 || Math.abs(i)<2.1)  
                count++;  
        }  
        System.out.println(count);  
    }  
}
```



The screenshot shows a terminal window titled "DemoMath" with a dark background. The command prompt displays the full path to the Java executable and the class name: "D:\Professional Program Files\jdk-14\bin\java.exe" "-javaagent:D:\Professional Program Files\JetBrains\IntelliJ 9". The output of the program is "9". At the bottom, a status message reads "进程已结束，退出代码 0".

```
DemoMath x  
"D:\Professional Program Files\jdk-14\bin\java.exe" "-javaagent:D:\Professional Program Files\JetBrains\IntelliJ 9  
9  
进程已结束，退出代码 0
```