

字符串、数组和文件IO流的基本操作

王海洲



CMD 使用方法

•切换路径

- cd 相对路径, cd 111
- cd 绝对路径, cd C:\LAB2\111
- 切换盘符, 比如切换到D盘, 不需要加cd, 直接输入 D:

•使用TAB来自动补全

- 比如我们想输入LAB2, 我们输出L, 按下TAB, 它会自动补全该目录下剩下的能于L匹配的字符。

•执行历史命令

- 使用上下键, 即可选择之前输入的命令。



内容

- 字符串

- 数组

- 输入输出流

字符串

Java的字符串类型为String，它不属于基本数据类型。(Java定义了8个数据类型:字节型(byte)，短整型(short)，整型(int)，长整型(long)，字符型(char)，浮点型(float)，双精度型(double)，布尔型(boolean).)

字符串在Java中被定义为对象，它的默认初始值为null。
通过双引号来表示。

字符串 ≠ 字符



字符串

1. 声明字符串

```
String str;
```

2. 创建字符串

```
str = new String("Hello World");
```

另一种方式:

```
str = "Hello World";
```

3. 声明和创建可以一步完成

```
String str = "####"; or String str = new String("####");
```

字符串

直接赋值创建和new String创建的区别:

如果是编译期已经创建好(直接用双引号定义的)的就存储在常量池中。

如果是运行期（new出来的）才能确定的就存储在堆中。

对于内容相同的字符串，在常量池中永远只有一份，在堆中有多份。



```
String str = "hello world"; (直接赋值)
```

如果hello world这个字符串在String池中已经存在，那么会直接把它的引用赋予str，不会再另外创建对象。java虚拟机会保证一个String常量在String池中只有一份copy。

```
String str = new String("hello world");
```

如果hello world这个字符串在String池里不存在，会在String池创建这个String对象，然后在堆中创建一个此对象的拷贝对象。否则，只会在heap创建一个 str 的String 对象。

一般建议使用直接赋值的方法来创建字符串。

字符串

例子:

```
String s1 = "Runoob";
```

```
String s2 = "Runoob";
```

```
String s3 = s1;
```

s1 == s2的返回值为?

s3 == s2的返回值为?

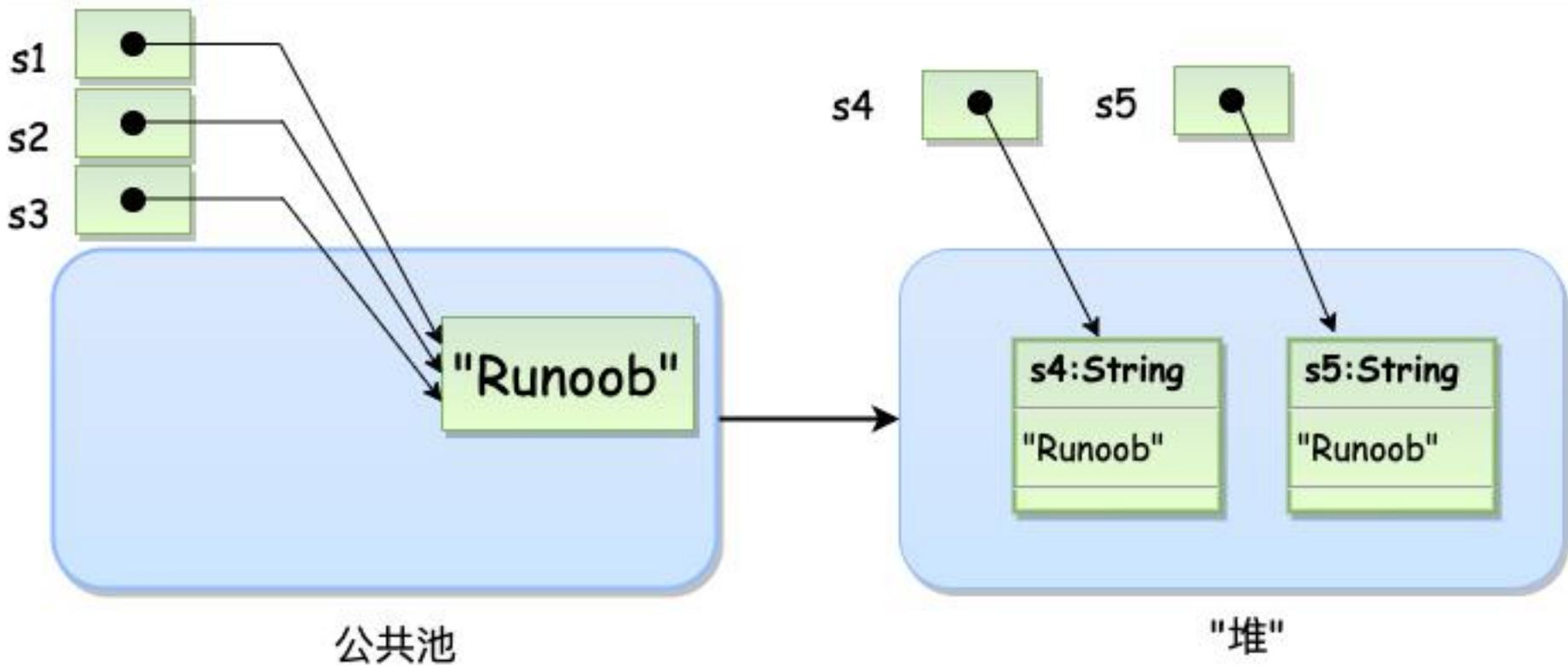
```
String s4 = new String("Runoob");
```

```
String s5 = new String("Runoob");
```

s4 == s5的返回值为?

可使用equals () 方法来比较字符串内容是否相同

字符串



字符串

方法	说明
<code>int length()</code>	返回字符串的长度
<code>String toUpperCase()</code>	将串中字符变成大写
<code>String toLowerCase()</code>	将串中字符变成小写
<code>char charAt(int i)</code>	返回位置i处的字符
<code>String substring(int s,int e)</code>	返回从位置s到e的字符子串[s,e)
<code>String substring(int s)</code>	返回从位置s到末尾的字符子串
<code>int indexOf(String s)</code>	返回首次出现字符串s的位置
<code>int indexOf(String s,int i)</code>	返回在位置i之后首次出现s的位置
<code>String trim()</code>	返回一个新串，去除前后空白字符
<code>String replace(String a,String b)</code>	返回一个新串，将a替换为b



数组

数组：一组有序数据的集合。每个元素具有相同的数据类型，可以用一个统一的数组名和下标来唯一的确定数组中的元素。

数组类型：基本类型、引用类型。

数组可以按照使用分为一维数组、二维数组和 multidimensional 数组。

数组

数组的定义形式:

数组类型 数组名 [] ;

或是

数组类型 [] 数组名 ;

如 `int i[]; double[] d; String s[];`

数组的创建:

方法一: 采用 `new` 操作符

`i = new int[10];` //int类型默认值为0

方法二:静态的初始化器

`i = {1,2,3,4,5};`

数组

可以在声明数组时直接初始化，如：

```
int[] i = new int[5];
```

或是

```
int[] i = {1,2,3,4,5};
```

一旦数组被创建了，那么数组对象的长度就不能再被改变。数组元素数组名和下标组成，下标的下界为0，上界为数组元素个数减一。

多维数组

二维数组的定义形式：

数组类型 多维数组名 [] [] ;

或是

数组类型 [] [] 多维数组名

例如： `int[][] array; int array2[][];`

二维数组的创建（分配内存）：

方法一：直接为每一维分配空间，如：

```
int a[][] = new int[2][3];
```

多维数组

方法二：从最高维开始，分别为每一维分配空间，如：

```
int a[][] = new int[2][];
```

```
a[0] = new int[3];
```

```
a[1] = new int[2];
```

二维的长度可不一样

多维数组初始化方法：

(1) 直接对每个元素进行赋值

(2) 在定义数组的同时进行初始化，如：

```
int a[][] = {{1,2},{3,4},{5,6}};
```



文件处理

文件是信息的一种组织形式，是存储在外部存储介质上具有标识名的一组相关信息集合。

文件可分为文本文件和二进制文件。

文件输入输出流

流：一组有顺序的、有起点和终点的字节集合，是对数据传输的总称。也就是说，数据在两个对象之间的传输称为流。

按照流的传输方法，可以分为输入流和输出流。

程序 -> 文件 输出

文件 -> 程序 输入

流的基本操作有读操作和写操作，从流中取得数据的操作称为读操作，向流中添加数据的操作称为写操作。对输入流只能进行读操作，对输出流只能进行写操作

标准输入和标准输出

执行一个shell命令行时通常会自动打开三个标准文件，即标准输入文件（stdin），通常对应终端的键盘；标准输出文件（stdout）和标准错误输出文件（stderr），这两个文件都对应终端的屏幕。进程将从标准输入文件中得到输入数据，将正常输出数据输出到标准输出文件，而将错误信息送到标准错误文件中。



命令行输入

- Java 读取输入参数，
`public static void main(String[] args)`
- 保存到变量名为args的String类型的数组，我们的程序可以在运行期间使用到它。

文件处理

File类(使用请import java.io.*;)

File类对象主要用来获取文件本身的一些信息，如文件所在的目录、文件长度、文件读写权限等。

File类的主要方法

方法定义	方法说明
public File(String pathname)	通过文件名创建一个file实例
public boolean canRead()	判断文件是否可读
public boolean canWrite()	判断文件是否可写
public boolean exists()	判断文件或目录是否存在

文件处理

方法定义	方法说明
<code>public long length()</code>	获取文件长度
<code>public String getName()</code>	获取文件名字，不包含路径
<code>public String getAbsolutePath()</code>	获取文件的绝对路径
<code>public boolean isFile()</code>	判断是否是一个文件
<code>public boolean isDirectory()</code>	判断是否是一个目录
<code>public Boolean mkdir()</code>	创建一个目录
<code>public boolean createNewFile()</code>	创建新文件
<code>public boolean delete()</code>	删除文件或空目录
<code>public boolean setReadOnly()</code>	设置文件属性为只读

输入输出流

字节流由InputStream和OutputStream处理

输入流	输出流	说明
FileInputStream	FileOutputStream	读写文件
BufferedInputStream	BufferedOutputStream	使用缓冲区，提高读写效率

输入输出流

字符流由Reader和Writer处理。

输入流	输出流	说明
FileReader	FileWriter	文件读写
BufferedReader	BufferedWriter	使用缓冲区，提高读写效率

以字节流为例的文件写操作：

```
public static void fileOutput() throws IOException {  
    String str = "hello world!";  
    File file = new File("d:\\test.txt"); //创建file对象  
    if(!file.exists()){  
        file.createNewFile(); //如果文件不存在，则进行创建  
    }  
    FileOutputStream fOutput = new FileOutputStream(file);  
    BufferedOutputStream bOutput = new BufferedOutputStream(fOutput);  
    byte[] buffer = str.getBytes(); //将字符串文本转换成字节数组  
    bOutput.write(buffer);  
    bOutput.close();  
    fOutput.close();  
}
```

以字节流为例的文件读操作：

```
public static void fileInput() throws IOException{
    File file = new File("d:\\test.txt"); //创建file对象
    FileInputStream flOutput = new FileInputStream(file);
    BufferedInputStream bInput = new BufferedInputStream(flOutput);
    int temp = 0;
    while((temp = bInput.read())!= -1){ //当temp为-1时，数据读取完毕
        System.out.print((char)temp);
    }

    bInput.close();
    flOutput.close();
}
```

以字符流为例的文件写操作：

```
public static void fileWriter() throws IOException {  
    String str = "hello world!";  
    File file = new File("d:\\test.txt");  
    if(!file.exists()){  
        file.createNewFile(); //如果文件不存在，则进行创建  
    }  
    FileWriter fwWriter = new FileWriter(file);  
    BufferedWriter bWriter = new BufferedWriter(fwWriter);  
    bWriter.write(str);  
    bWriter.close();  
    fwWriter.close();  
}
```

以字符流为例的文件读操作：

```
public static void fileReader() throws IOException {  
    File file = new File("d:\\test.txt");  
    FileReader fReader = new FileReader(file);  
    BufferedReader buReader = new BufferedReader(fReader);  
    String temp = null;  
    while((temp = buReader.readLine()) != null) {  
        System.out.println(temp);  
    }  
    buReader.close();  
    fReader.close();  
}
```

输入输出流

上述两种方法都能实现将文本"hello world"写入到D盘test.txt文件中，并读取该文件文本内容并打印出来的功能。

注意：

(1) 在做IO处理需要引入必要的IO包，在程序顶部加上

```
import java.io.*;
```

(2) 在java中进行输入输出流操作时必须进异常处理，可以使用throws抛出IOException。

(3) 对于流对象的操作，使用完毕后，一定要调用其close()方法将其释放掉。



阅读资料

- 数组 (<https://www.runoob.com/java/java-array.html>)
- 字符串 (<https://www.runoob.com/java/java-string.html>)
- 文件IO (<https://www.runoob.com/java/java-files-io.html>)

作业标准

- 文件夹（邮件名）命名为LabX_学号_姓名，然后打压缩包。
- 文件夹只需要一个文档（尽量使用pdf，其他格式在不同电脑上容易出现乱码），请编好题号！！！不要给N个文件夹，以及若干代码文件。
- 在文档中，每道题都请尽量添加文字描述或图片。
- 邮件正文里不要发作业内容。
- 提交作业前，请检查好自己的作业格式。

 height.java	20
 HelloWorld.java	20
 Hi.java	20
 readme.docx	20
 tenhello.java	20