Java 应用与开发

Java 数组和字符串

王晓东

wangxiaodong@ouc.edu.cn

中国海洋大学

October 16, 2020

参考书目

1. 陈国君等编著, Java 程序设计基础（第 5 版）, 清华大学出版社
2. Bruce Eckel, Thinking in Java (3rd)

学习目标

1. 掌握 Java 数组的概念
2. 学会一维数组和二维数组的使用；认识 Arrays 类，掌握操作数组相关方法
3. 掌握 Java 字符串的概念，字符串与数组的关系；学会 String 类常用字符串操作方法

大纲

数组的概念

一维数组

二维数组

字符串

接下来…

数组的概念

一维数组

二维数组

字符串

数组的基本概念

数组是相同数据类型的元素按一定顺序排列的集合。Java 中，数组元素既可以为基本数据类型，也可以为对象。

❖ Java 的内存分配（基础）

栈内存 存放定义的基本类型的变量和对象的引用变量，超出作用域将自动释放。

堆内存 存放由 new 运算符创建的对象和数组，由 Java 虚拟机的自动垃圾回收器来管理。

数组的主要特点

- 数组是相同数据类型的元素的集合；
- 数组中的各元素有先后顺序，它们在内存中按照这个先后顺序连续存放；
- 数组的元素用整个数组的名字和它自己在数组中的顺序位置来表示。

例如，a[0] 表示名字为 a 的数组中的第一个元素，a[1] 表示数组 a 的第二个元素，依次类推。

接下来…

数组的概念

一维数组

二维数组

字符串

一维数组

创建 Java 数组一般需经过三个步骤：

1. 声明数组
2. 创建内存空间
3. 创建数组元素并赋值

CODE ▸ 一维数组创建声明和内存分配

```
1  int[] x; //声明名称为x的int型数组，未分配内存给数组
2  x = new int[10]; //x中包含有10个元素，并分配空间
```

```
1  int[] x = new int[10]; //声明数组并动态分配内存
```

动态内存分配说明

用 new 分配内存的同时，数组的每个元素都会自动赋默认值，整型为 0，实数为 0.0，布尔型为 false，引用型为 null。

一维数组

❖ 一维数组的初始化

若在声明数组时进行赋值即初始化称为静态内存分配。

数据类型 [] 数组名 = 初值 0，初值 1，…，初值 n;

CODE ▸ 一维数组静态初始化

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

☛ 注意

在 Java 程序中声明数组时，无论用何种方式定义数组，都不能指定其长度。

课程配套代码 ▸ sample.array.ArraySample.java

接下来…

数组的概念

一维数组

二维数组

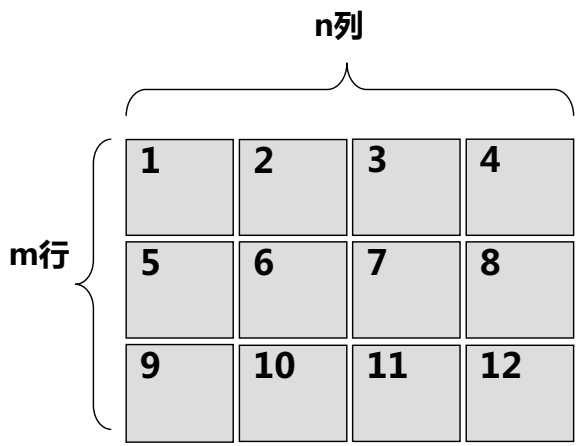
字符串

二维数组

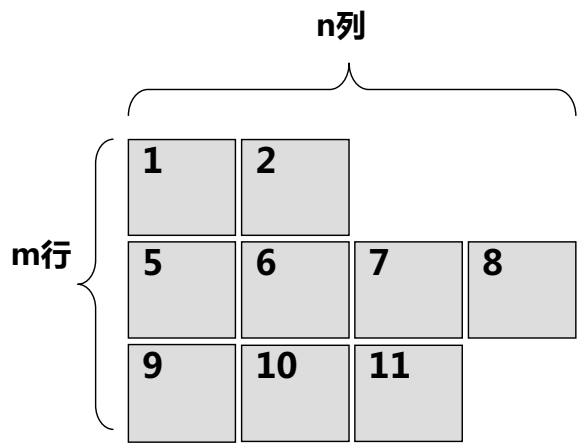
Java 中无真正的多维数组，只是数组的数组。

❖ 二维数组的声明和内存分配

数据类型 [] [] 数组名;
数组名 = new 数据类型 [行数] [列数];
数据类型 [] [] 数组名 = new 数据类型 [行数][列数];



C语言的二维数组是矩形



Java语言的二维数组不一定是矩形

二维数组定义的含义

- Java 中的二维数组看作是由多个一维数组构成
- 二维数组申请内存必须指定高层维数

```
int[][] myArray1 = new int[10][];
```

```
int[][] myArray2 = new int[10][3];
```

- ```
  int[][] x;
  ```

  表示定义了一个数组引用变量 x，第一个元素为 x[0]，最后一个为 x[n-1]，其长度不确定

- ```
  x = new int[3][];
  ```

  表示数组 x 有三个元素，每个元素都是 int[] 类型的一维数组，分别为 int x[0][]、int x[1][]、int x[2][]

```
x[0] = new int[3]; x[1] = new int[2];
```

给 x[0]、x[1]、x[2] 赋值（长度可以不一样）

二维数组定义的含义

- Java 中的二维数组看作是由多个一维数组构成
- 二维数组申请内存必须指定高层维数

```
int[][] myArray1 = new int[10][];
```

```
int[][] myArray2 = new int[10][3];
```

- `int[][] x;`

表示定义了一个数组引用变量 x，第一个元素为 x[0]，最后一个为 x[n-1]，其长度不确定

- `x = new int[3][];`

表示数组 x 有三个元素，每个元素都是 int[] 类型的一维数组，分别为 int x[0][]、int x[1][]、int x[2][]

```
x[0] = new int[3]; x[1] = new int[2];
```

给 x[0]、x[1]、x[2] 赋值（长度可以不一样）

二维数组定义的含义

- Java 中的二维数组看作是由多个一维数组构成
- 二维数组申请内存必须指定高层维数

```
int[][] myArray1 = new int[10][];
```

```
int[][] myArray2 = new int[10][3];
```

- `int[][] x;`

  表示定义了一个数组引用变量 x，第一个元素为 x[0]，最后一个为 x[n-1]，其长度不确定

- `x = new int[3][];`

  表示数组 x 有三个元素，每个元素都是 int[] 类型的一维数组，分别为 int x[0][]、int x[1][]、int x[2][]

  ```
  x[0] = new int[3]; x[1] = new int[2];
  ```

  给 x[0]、x[1]、x[2] 赋值（长度可以不一样）

二维数组定义的含义

- Java 中的二维数组看作是由多个一维数组构成
- 二维数组申请内存必须指定高层维数

```
int[][] myArray1 = new int[10][];
```

```
int[][] myArray2 = new int[10][3];
```

- ```
  int[][] x;
  ```

  表示定义了一个数组引用变量 x，第一个元素为 x[0]，最后一个为 x[n-1]，其长度不确定

- ```
  x = new int[3][];
  ```

  表示数组 x 有三个元素，每个元素都是 int[] 类型的一维数组，分别为 int x[0][]、int x[1][]、int x[2][]

  ```
  x[0] = new int[3]; x[1] = new int[2];
  ```

  给 x[0]、x[1]、x[2] 赋值（长度可以不一样）

二维数组赋初值

```
1    int[][] a = {{11,22,33,44}, {66,77,88,99}};
```

注意

声明多维数组并初始化时不能指定其长度，否则出错。

课程配套代码 ▸ sample.array.Array2DimSample.java

Arrays 类

java.util.Arrays 工具类能方便地操作数组，它提供的所有方法都是静态的。该类具有以下功能：

给数组赋值 通过 fill 方法。

对数组排序 通过 sort 方法。

比较数组 通过 equals 方法比较数组中元素值是否相等。

查找数组元素 通过 binarySearch 方法能对排序好的数组进行二分查找法操作。

复制数组 把数组复制成一个长度为 length 的新数组。

课程配套代码 ➧ sample.array.ArrayToolsSample.java

接下来…

数组的概念

一维数组

二维数组

字符串

字符串

字符串操作是计算机程序设计中最常见的行为。

- 字符串是用一对双引号括起来的字符序列。Java 语言中，字符串常量或变量均用类实现。
- String 对象是不可变的。String 类每一个看起来会修改 String 值的方法，实际上都创建了一个新的对象，以包含修改后的字符串内容。

```
1    String s = new String("hello java");
2    s.toUpperCase();
3    System.out.println(s);
```

理解 Java 字符串

CODE ◗ String.java Part 1

```
1     public final class String
2     implements java.io.Serializable, Comparable<String>, CharSequence { //1
3     /** The value is used for character storage. */
4     private final char value[]; //2

6     /** The offset is the first index of the storage that is used. */
7     private final int offset;

9     /** The count is the number of characters in the String. */
10    private final int count;

12    /** Cache the hash code for the string */
13    private int hash; // Default to 0

15    /** use serialVersionUID from JDK 1.0.2 for interoperability */
16    private static final long serialVersionUID = -6849794470754667710L;
17    ........
18  }
```

1. String 类是 final 类，即意味着 String 类不能被继承，并且它的成员方法都默认为 final 方法。
2. 从 String 类的成员属性可以看出 String 类其实是通过 char 数组来保存字符串的。

理解 Java 字符串

CODE ◗ String.java Part 2

```
1     public String substring(int beginIndex, int endIndex) {
2       if (beginIndex < 0) {
3         throw new StringIndexOutOfBoundsException(beginIndex);
4       }
5       if (endIndex > count) {
6         throw new StringIndexOutOfBoundsException(endIndex);
7       }
8       if (beginIndex > endIndex) {
9         throw new StringIndexOutOfBoundsException(endIndex - beginIndex);
10      }
11      return ((beginIndex == 0) && (endIndex == count)) ? this :
12      new String(offset + beginIndex, endIndex - beginIndex, value);
13    }
```

1. 无论是 substring 还是 concat 操作等都不是在原有的字符串上进行的，而是重新生成了一个新的字符串对象，最原始的字符串并没有被改变。
2. String 对象一旦被创建就是固定不变的，对 String 对象的任何操作都不影响到原对象，而是会生成新的对象。

字符串变量的创建

❖ 字符串变量的创建

CODE ◗ 格式 1

```
1    String s;                  //声明字符串型引用变量s，此时s的值为null
2    s = new String("Hello"); //在堆内存中分配空间，并将s指向该字符串首地址
```

CODE ◗ 格式 2

```
1    String s = new String("Hello");
```

CODE ◗ 格式 3

```
1    String s = "Hello";
```

String 类的常用方法

CODE ◗ 求字符串长度

```
1    String str = new String("asdfzxc");
2    int strlength = str.length(); //strlength = 7
```

CODE ◗ 获取字符串某一位置字符

```
1    char ch = str.charAt(4); //ch = z
```

CODE ◗ 提取子串

```
1    String str2 = str1.substring(2); //str2 = "dfzxc"
2    String str3 = str1.substring(2,5); //str3 = "dfz"
```

CODE ◗ 字符串连接

```
1    String str = "aa".concat("bb").concat("cc");
2    String str = "aa" + "bb" + "cc"; // 相当于上一行
```

String 类的常用方法

CODE ◗ 字符串比较

```
1    String str1 = new String("abc");
2    String str2 = new String("ABC");
3    int a = str1.compareTo(str2); //a>0
4    int b = str1.compareTo(str2); //b=0
5    boolean c = str1.equals(str2); //c=false
6    boolean d = str1.equalsIgnoreCase(str2); //d=true
```

CODE ◗ 字符串中字符的大小写转换

```
1    String str = new String("asDF");
2    String str1 = str.toLowerCase(); //str1 = "asdf"
3    String str2 = str.toUpperCase(); //str2 = "ASDF"
```

CODE ◗ 字符串中字符的替换

```
1    String str = "asdzxcasd";
2    String str1 = str.replace('a','g'); //str1 = "gsdzxcgsd"
3    String str2 = str.replace("asd","fgh"); //str2 = "fghzxcfgh"
4    String str3 = str.replaceFirst("asd","fgh"); //str3 = "fghzxcasd"
5    String str4 = str.replaceAll("asd","fgh"); //str4 = "fghzxcfgh"
```

✍ 课后小作业 请自行总结常用的字符串操作方法。

+ 与 StringBuilder

String 不变性带来了一定的效率问题。用于 String 的“+”和“+=”是 Java 中仅有的两个重载过的运算符。“+”可以用来连接字符串。

```
1    String s = "Java";
2    String ss = "Hello" + s + ". I love you.";
3    System.out.println(ss);
```

这段代码可能是怎么工作的？

String 可能有一个 append() 方法，它会生成一个新的 String 对象，该新对象包含了“Hello”与 s 连接后的字符串；然后再与“I love you”连接，再次生成新的 String 对象。

这种工作方式会产生一大堆需要垃圾回收的中间对象！

+ 与 StringBuilder

String 不变性带来了一定的效率问题。用于 String 的“+”和“+=”是 Java 中仅有的两个重载过的运算符。“+”可以用来连接字符串。

```
1    String s = "Java";
2    String ss = "Hello" + s + ". I love you.";
3    System.out.println(ss);
```

这段代码可能是怎么工作的？
String 可能有一个 append() 方法，它会生成一个新的 String 对象，该新对象包含了“Hello”与 s 连接后的字符串；然后再与“I love you”连接，再次生成新的 String 对象。
这种工作方式会产生一大堆需要垃圾回收的中间对象！

+ 与 StringBuilder

String 不变性带来了一定的效率问题。用于 String 的“+”和“+=”是 Java 中仅有的两个重载过的运算符。“+”可以用来连接字符串。

```
1    String s = "Java";
2    String ss = "Hello" + s + ". I love you.";
3    System.out.println(ss);
```

这段代码可能是怎么工作的？
String 可能有一个 append() 方法，它会生成一个新的 String 对象，该新对象包含了“Hello”与 s 连接后的字符串；然后再与“I love you”连接，再次生成新的 String 对象。
这种工作方式会产生一大堆需要垃圾回收的中间对象！

实际情况是如何？

课程配套代码 ➧ sample.string.StringConcatSample.java
反编译 Java 类文件：

```
1    javap -c StringConcatSample
```

-c 表示生成 JVM 字节码。

- 我们并没有主动使用 StringBuilder 类，但编译器自作主张的使用了它，因为它更高效。
- StringBuilder 用于创建最终的 String，为每个字符串调用 StringBuilder 的 append() 方法。

实际情况是如何？

课程配套代码 ➧ sample.string.StringConcatSample.java
反编译 Java 类文件：

```
1    javap -c StringConcatSample
```

-c 表示生成 JVM 字节码。

- 我们并没有主动使用 StringBuilder 类，但编译器自作主张的使用了它，因为它更高效。
- StringBuilder 用于创建最终的 String，为每个字符串调用 StringBuilder 的 append() 方法。

本节习题

❖ 小编程

1. 编写程序，求一个三阶方阵的对角线上各元素之和。
2. 编写程序，从键盘上输入一个字符串和子串开始位置与长度，截取该字符串的子串并输出。
3. 编写程序，统计用户从键盘输入的字符串中包含的字母、数字和其他字符的个数。

THE END

wangxiaodong@ouc.edu.cn