讲义03：数组与字符串

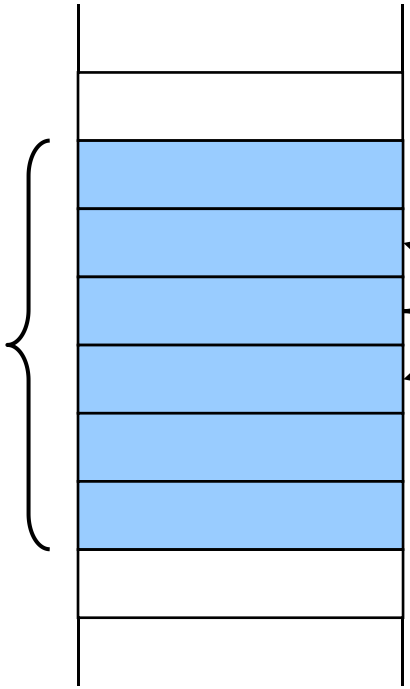
主要内容

- 讲解内容
  - Java数组的基本概念
  - 一维数组的创建与使用
  - foreach语句
  - 多维(二维)数组的创建与使用
  - 字符串String入门
- 内容来源资料
  - 教材第5章
  - API文档：
    - java.util.Arrays
    - java.lang.String

1 数组的基本概念

包括Java在内的高级语言提供称为数组(array)的数据结构。

数组是一种**线性**数据结构，它是一个**有序**集，存储**数据类型相同**、**个数固定**的多个元素。



有序是指各个元素有先后次序.

前驱

后继

数组占用连续的内存空间，且元素个数在分配内存空间时确定.

Java的数组具有基本的数组特征，同时，Java的数组是对象方式提供，其使用方法与非OO语言(例如C语言)有所不同。

2 一维数组

Java数组的使用通常有3个步骤：

1. 声明数组(声明数组变量)
2. 分配数组的内存空间(创建数组对象)
3. 使用数组元素

2.1 一维数组的使用

第1步：声明数组变量

语法：(1)数据类型[]　数组变量;

(2)数据类型　数组变量[];

不推荐使用

示例：
```
double[]  myList;
double  myList[];
```

说明：

声明一个数组变量并没有在内存中为数组分配空间.

只是为数组变量分配了空间,用以存储引用数组的地址.

可以理解只有“数组名”.

2.1 一维数组的使用

第2步：创建数组对象

创建数组对象，即为数组分配内存空间。语法如下:

```
数组变量 = new 数组元素类型[数组大小];
```

上面语句完成了2项工作:

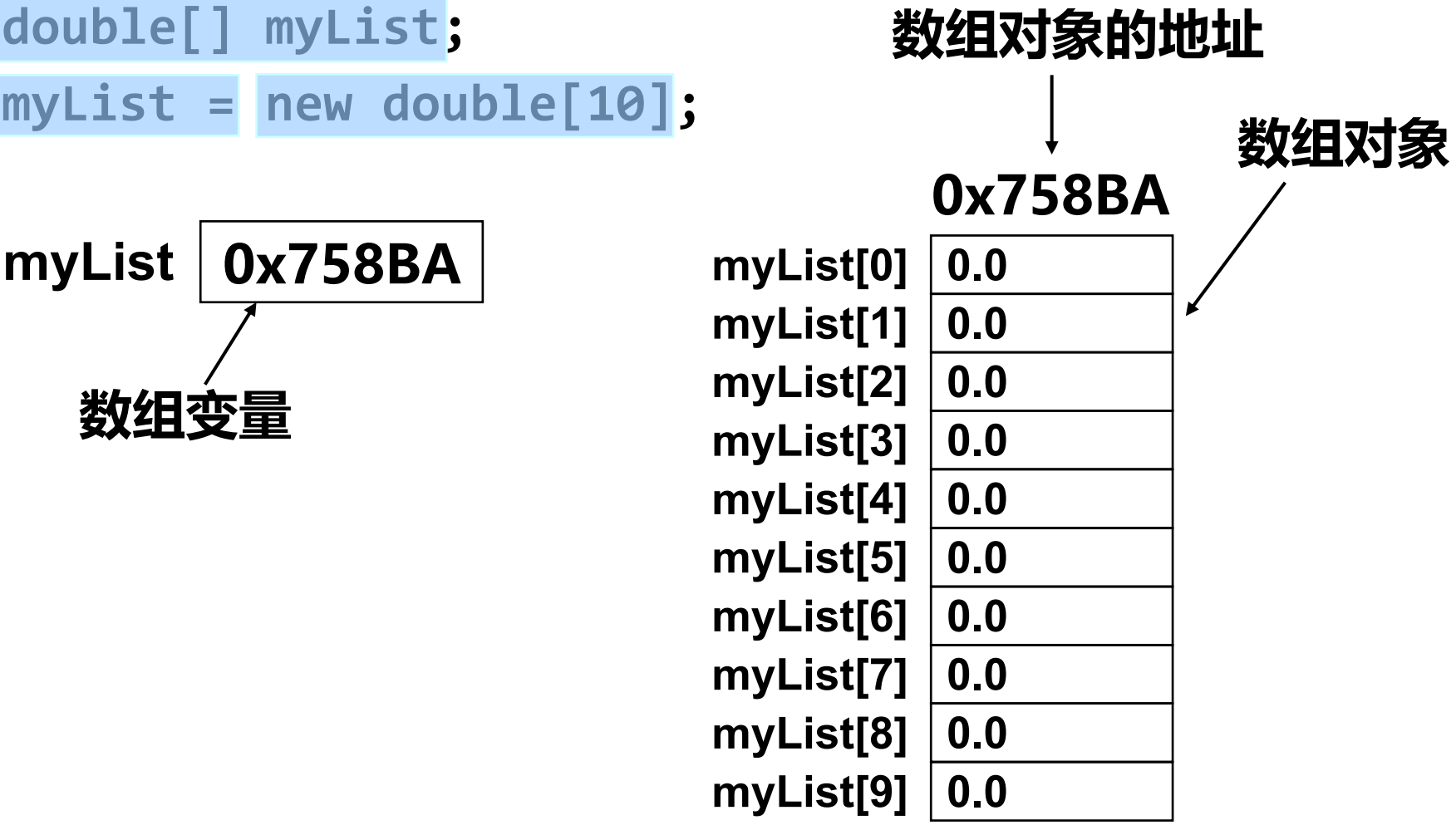
(1)“new 数组元素类型[数组大小]” 完成为数组分配内存空间.

(2)“=”把创建的数组对象的地址赋值给数组变量.

2.1 一维数组的使用

数组对象创建过程演示

```
double[] myList;
myList = new double[10];
```



2.1 一维数组的使用

第3步：使用数组元素

通过数组变量和下标访问数组元素：数组名[下标]

说明：

Java的数组下标是基于0的.

Java的数组下标必须是整数或整数表达式.

示例：

```
double d = myList[5] * 10.0;
myList[3] = 12.34;
```

2.2 Java数组的基本特点

- 数组分配内存空间时确定数组的大小，不能改变。
- 使用数组时，数组名.length 表示数组的大小。
- 数组创建后，其元素有默认值
  - 数值型默认为：0
  - char型默认为：‘\u0000’
  - boolean型默认为：false

2.3 Java数组的初始化

数组元素初始化的语法:

(1) 数组类型[] 数组变量 = {直接量0, 直接量1, …, 直接量k};

(2) 数组变量 = new 元素类型[] {直接量0, 直接量1, …, 直接量k};

例如:

```
double[] myList = {1.9, 2.9, 3.4, 3.5};

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

3 多维数组(二维数组为例)

Java语言没有真正的多维数组。
所谓多维数组，是数组元素也是数组的数组。

3.1 声明多维数组变量

声明多维数组变量的语法:

(1)数据类型[ ]…[ ]　数组引用变量;

(2)数据类型　数组引用变量[ ]…[ ];

(3)数据类型[ ]…[ ]　数组引用变量[ ]…[ ];

声明中[]的个数决定数组的维度。不推荐(2)(3)两种语法

例如，3种声明二维整型数组的方式:

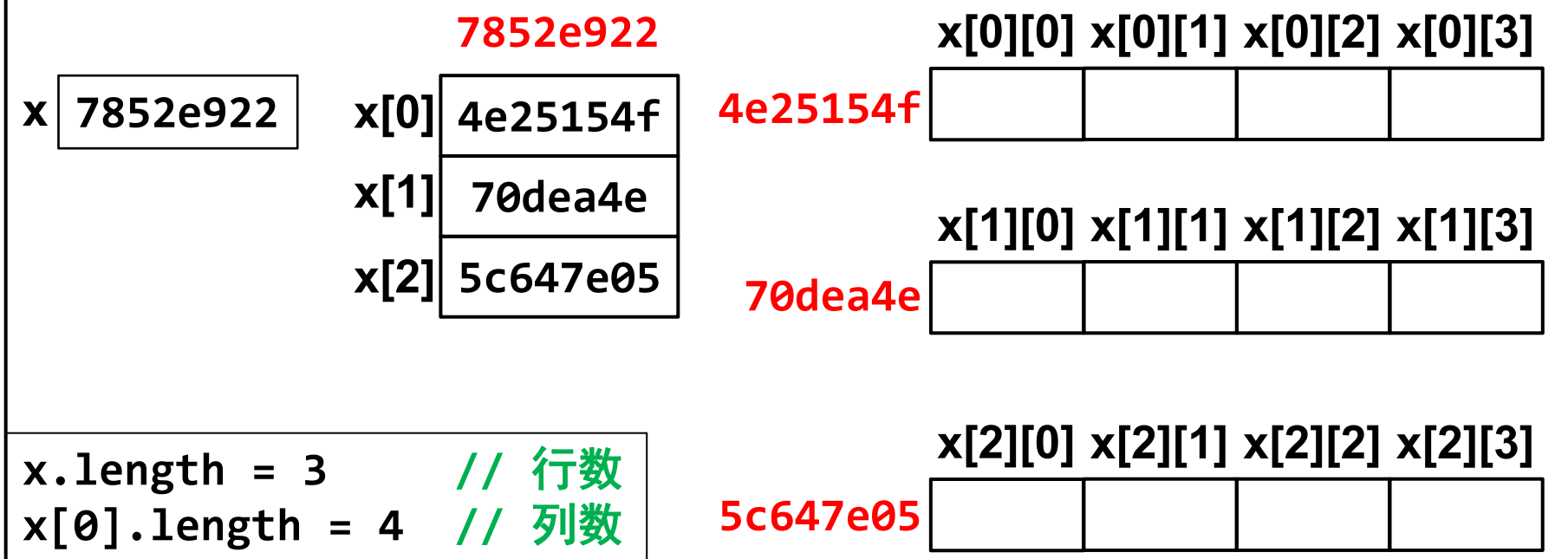
```
int[][] matrix;
int matrix[][]; //不推荐使用
int[] matrix[]; //不推荐使用
```

3.2 创建多维数组对象

方式1：同时创建整个多维数组对象

```
int[][] x = new int[3][4]; //同时指定行、列数
```

内存示意图

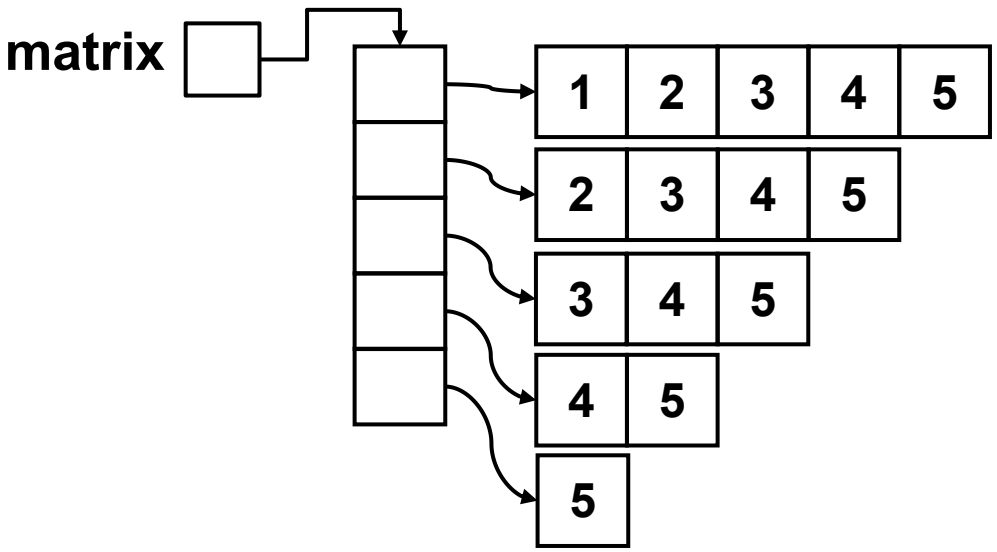



```
x.length = 3        // 行数
x[0].length = 4     // 列数
x[1].length = 4
x[2].length = 4
```

3.2 创建多维数组对象

方式2：分别创建“行对象”和“列对象”

```
int[][] matrix = new int[5][];              //创建“行对象”
matrix[0] = new int[] { 1, 2, 3, 4, 5 }; //创建“列对象”
matrix[1] = new int[] { 2, 3, 4, 5 };
matrix[2] = new int[] { 3, 4, 5 };
matrix[3] = new int[] { 4, 5 };
matrix[4] = new int[] { 5 };
```



matrix.length is 5
matrix[0].length is 5
matrix[1].length is 4
matrix[2].length is 3
matrix[3].length is 2
matrix[4].length is 1

4 数组的基本操作

数组的常见算法

- 向数组中填入若干元素的值。
- 累加求和等运算
- 数组元素的排序
- 查找最大、最小和特定的值及其下标
- 数组元素的插入、删除、移动等

4 数组的基本操作

数组的2种循环遍历

```
for(int i = 0; i < array.length; i++) {
    //处理数组元素array[i]
}
```

```
for(元素类型 元素变量: 数组变量){
    //处理元素变量
}
```

for-each循环
使用元素变量顺序、
只读的访问整个数组

```
double[ ] myList = new double[10];
……
for(double value : myList) {
   System.out.println(value);
}
```

4 数组的基本操作

复制数组, 已经定义：int[] a = {1, 2, 3, 4, 5};

- 用循环语句复制数组
  int[] b = new int[a.length];
  for(int i=0; i<a.length; i++) { b[i] = a[i]; }
- 使用System类的arraycopy方法
  int[] b = new int[a.length];
  System.arraycopy(a, 0, b, 0, a.length);
- 使用数组对象的clone方法复制数组
  int[] b = a.clone();

思考：为什么不能直接使用 int[] b = a; 实现数组的复制?

4 数组的基本操作

使用Arrays类对数组进行操作

- Arrays类中包含了对数组的常用操作方法, 例如:
  - int index = Arrays. binarySearch(数组名, 查找数据)
  - Arrays.sort(数组名)
  - boolean b = Arrays.equals(数组名1, 数组名2);
  - ……

- 使用该类时, 应在程序中导入Arrays类
  import java.util.Arrays;

- 更多内容需要自行JDK API文档

5 数组示例

示例1：一维数组的遍历、查找和排序
从键盘输入个数n。
随机产生n个0~1000之间的整数；输出这些整数；找到并输出这些整数的最大值；对这些整数进行升序排序并输出。

代码：examples/lecture03/numbers

示例2：使用二维数组存储杨辉三角
从键盘输入生成杨辉三角的行数；生成杨辉三角数据存储到二维数组；输出杨辉三角。

代码： examples/Lecture03/yanghui

6 字符串 java.lang.String

- String类用来描述和处理字符串类型
- 所有的字符串直接量, 例如"abc", 都是String类
- String类的对象是不可变对象, 即它们的值在对象创建后就不能改变了

- String类需要了解
  - 构造方法
  - 实例方法
  - 静态方法
- 注意不要使用被标注为Deprecated的方法。

6.1 java.lang.String - 常用构造方法

1. 使用字符串直接量构造, 例如:
   String message = new String(“welcome to Java”);

2. 字符串的快捷初始化方法, 例如:
   String message = “welcome to Java”;

3. 使用字符数组构造, 例如:
   char[] charArray = [‘G’, ‘o’, ‘o’, ‘d’, ‘ ’, ‘D’, ‘a’, ‘y’];
   String message = new String(charArray);

6.2 java.lang.String - 不可变及规范字符串

String对象是不可变的, 其内容不能改变.

Java把相同的字符串直接量存储为一个对象(规范字符串).

```
String s = “welcome to Java”;
String s1 = new String(“welcome to Java”);
String s2 = s1.intern();
String s3 = “welcome to Java”;

System.out.println(“s1 == s is” + (s1 == s));
System.out.println(“s2 == s is” + (s2 == s));
System.out.println(“s3 == s is” + (s3 == s));
```

输出结果:

s1 == s is false
s2 == s is true
s3 == s is true

6.3 java.lang.String - 字符串更新方法

由于String对象是不可变的，因此对String对象进行修改操作时，原字符串不变，生成一个新的String对象。

以下给出几个方法:

- `String trim()`
- `String replace(char oldChar, char newChar)`
- `String replaceFirst(String oldString, String newString)`
- `String replaceAll(String oldString, String newString)`

使用代码演示方法的使用

6.4 java.lang.String - 其他常用方法

- int length()
- boolean equals(Object anObject)
- String substring(int beginIndex)
- String substring(int beginIndex, int endIndex)
- char chatAt(int index)
- int indexOf(String str)
- int lastIndexOf(String str)
- int compareTo(String anotherString)

使用代码演示方法的使用

课后工作

- 结合教材内容，复习课堂讲授内容
- 完成作业网站：练习1、练习2