

- 数组的声明、创建、引用
 - 一维数组
 - 二维数组
 - 规则二维数组
 - 不规则二维数组
- 不定长参数与数组
- for each循环
- Arrays类

- 数组：一组相同数据类型的元素按一定顺序线性排列。
- 数组的特点
 - (1) 数组是相同数据类型的元素的集合。
 - (2) 数组中的各元素是有先后顺序的。它们在内存中按照这个顺序连续存放在一起。
 - (3) 每个数组元素用整个数组的名字和它自己在数组中的位置表达（此位置被叫做下标）。

- Java中的数组是对象，因此属于引用类型，数组对象需要使用new关键字来创建。
- 数组
 - 一维数组
 - 多维数组

1. 声明数组格式

数组元素类型[] 数组名;

或

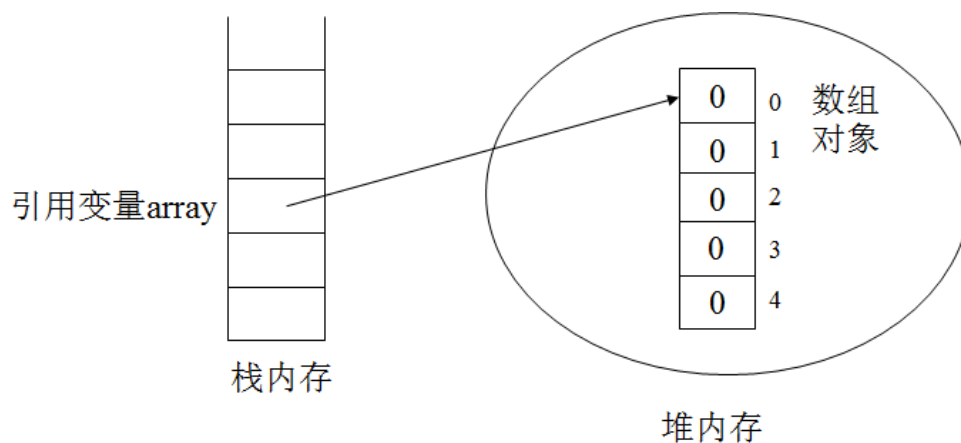
数组元素类型 数组名[];

例如: `int[] intArray;`

1、new关键字

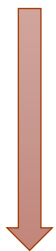
数组引用名 = new 数组元素类型[数组元素个数];

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



3.2 创建数组对象

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



合二为一

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

```
int[] Array = new int [5]; // 初值为0
```

■ 用new关键字为一个数组分配内存空间后，系统将为每个数组元素都赋予一个初值，这个初值取决于数组的类型。

❖ 所有数值型数组元素的初值为0

❖ 字符型数组元素的初值为一个Unicode编码为0的不可见的控制符 ('\u0000')

❖ 布尔型数组元素的初值为false

❖ 注意：Java中的数组对象一旦创建之后，在程序整个执行期间，就不能再改变数组元素的个数。

2、匿名数组：不需要引用名

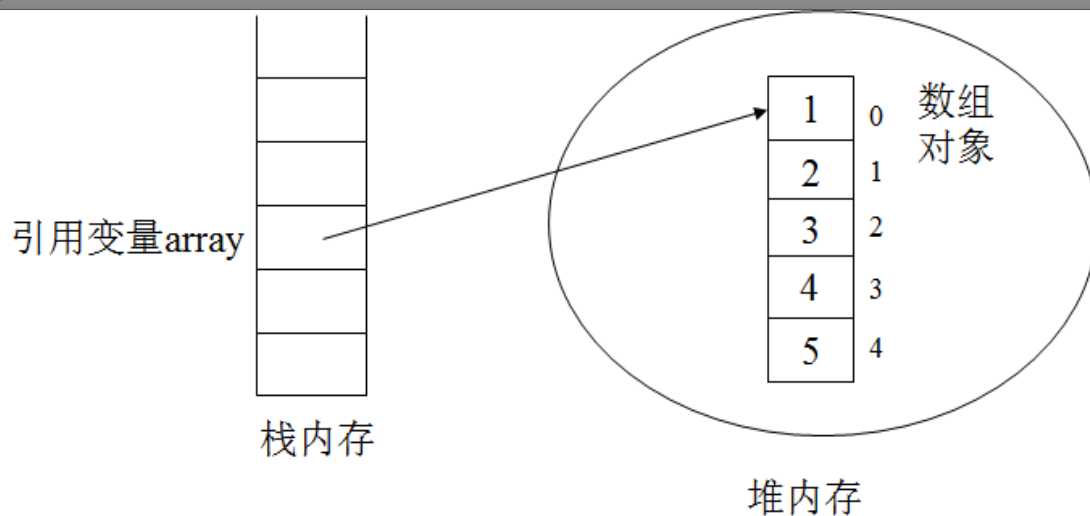
```
new int[]{1,2,3,4,5}
```

```
public static void main(String[] args) {  
    //匿名数组做参数  
    int sum = getSum(new int[]{1,2,3,4,5});  
    .....  
}  
public static int getSum(int[] a){ //形参接收实参数组  
    .....  
}
```


3. 数组的显式初始化

- 在声明数组时，进行初始化。数组**元素的个数**由初始化列表中**数据个数**决定。
- 编译器不允许指定数组的大小。现在你拥有的只是对数组的一个**引用**（已经为该引用分配了足够的存储空间），并且没有为数组对象本身分配任何空间。为了给数组创建相应的存储空间，必须对数组进行初始化。

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

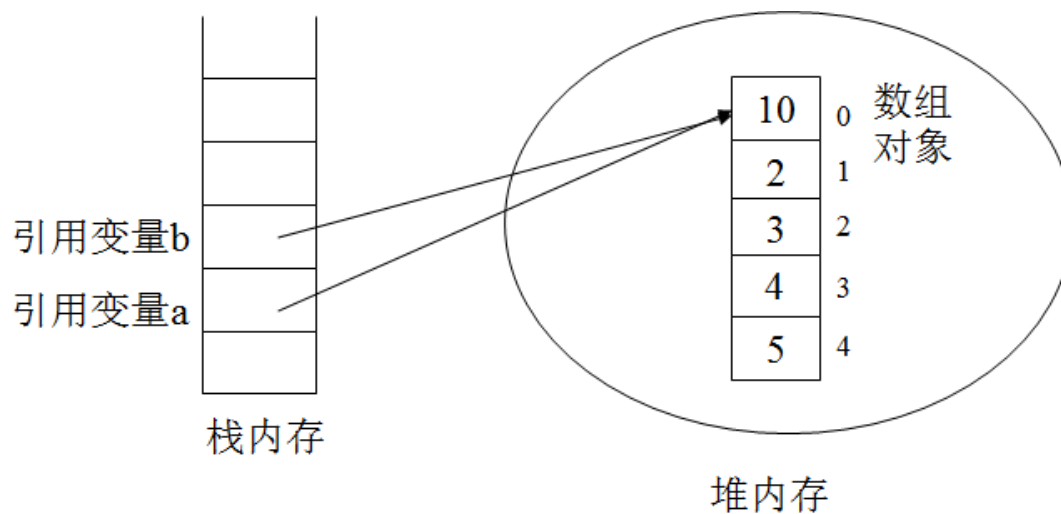


【例3-1】写出下面代码的运行结果。

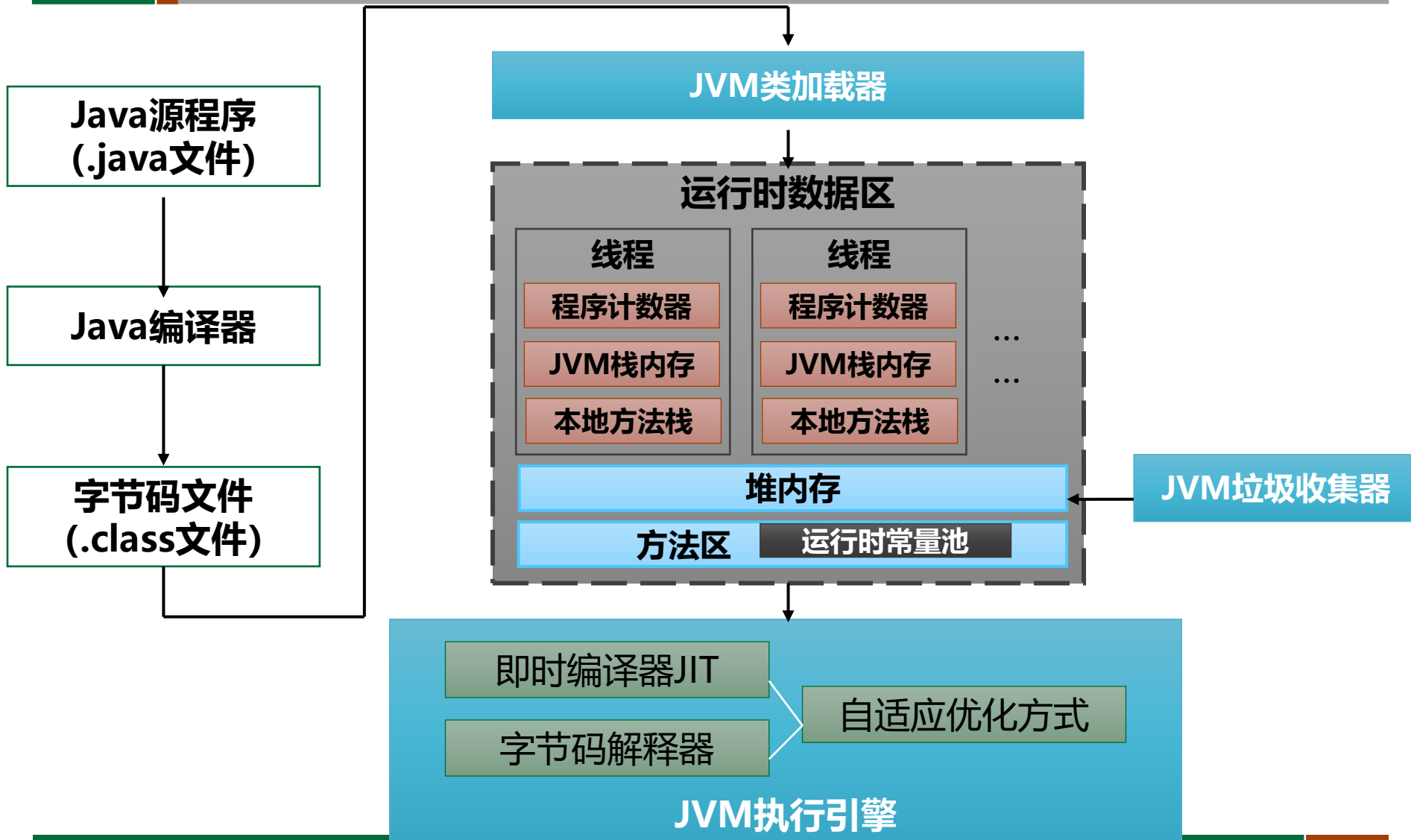
```
public static void main(String[] args) {  
    int[] a={1,2,3,4,5},b;  
  
    b=a;  
    b[0]=10;  
    System.out.println("a[0]="+a[0]);  
}
```

3.2 创建数组对象

```
public static void main(String[] args) {  
    int[] a={1,2,3,4,5},b;  
  
    b=a;  
    b[0]=10;  
    System.out.println("a[0]="+a[0]);  
}
```



JVM栈内存(stack)和堆内存(heap)



主要是用来执行程序用的，程序中的每个方法被执行时都会创建一个栈帧用来存储该方法的局部变量等信息，包括各种基本类型的变量和数组/对象的引用变量。栈内存可以称为一级缓存，由垃圾回收器自动回收



```
public class Ex215 {  
    public static void main(String[] args) {  
        printPyramid('g');  
    }  
    public static void printPyramid(char c){  
        for(char row='a'; row<=c; row++){  
            for(int i=1; i<=c-row; i++){  
                System.out.print(" ");  
            }  
            for(char ch='a'; ch<=row; ch++){  
                System.out.print(ch);  
            }  
            for(char ch=(char)(row-1); ch>='a'; ch--){  
                System.out.print(ch);  
            }  
            System.out.println();  
        }  
    }  
}
```

用于存储Java中的对象和数组，当我们new一个对象或者创建一个数组的时候，就会在堆内存中开辟一段空间给它，用于存放【对于堆内存一般开发人员会自动回收它】

堆内存

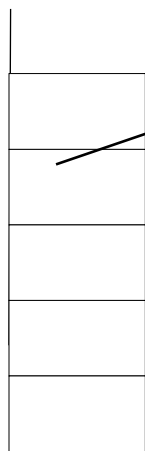
对象/数组

Java虚拟机
自动垃圾回收机制

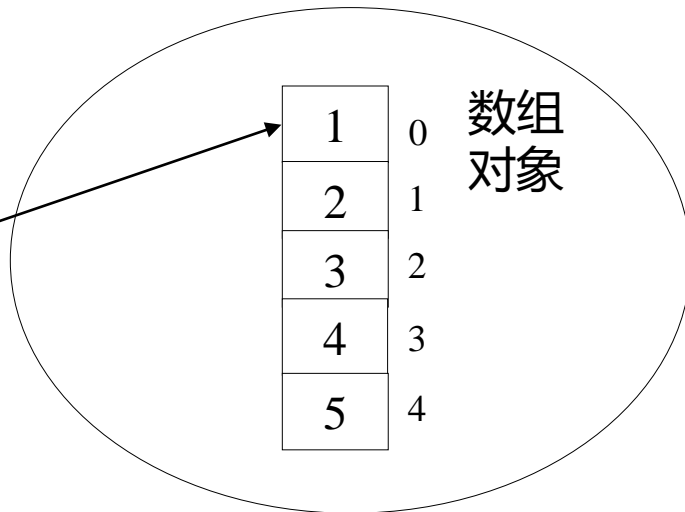
new

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

array



栈内存



堆内存

3.3 使用数组

3.3.1 数组元素的引用



1. 数组元素的使用

- 一维数组元素的使用方式

数组名[下标]

- 下标必须是整型或者可以转化成整型的量。下标的取值范围：从0开始到数组的长度减1。

例如, `int intArray=new int[5];`

`intArray[0], intArray[1],..., intArray[4]`

- 所有的数组都有一个属性`length`，这个属性存储了数组元素的个数。

`intArray.length` 的值为5。

`intArray[length-1]`

- Java系统能自动检查是否有数组下标越界的情况
 - 如果在程序中使用`intArray[10]`，就会发生数组下标越界，此时Java系统会自动终止当前的流程，并产生一个名为`ArrayIndexOutOfBoundsException`的异常，通知使用者出现了数组下标越界。
 - 避免越界发生的有效方法是利用`length`属性作为数组下标的上界。

【例3-2】输入n个学生的成绩，并打印成绩高于平均分的学生。

【例3-3】冒泡法排序。

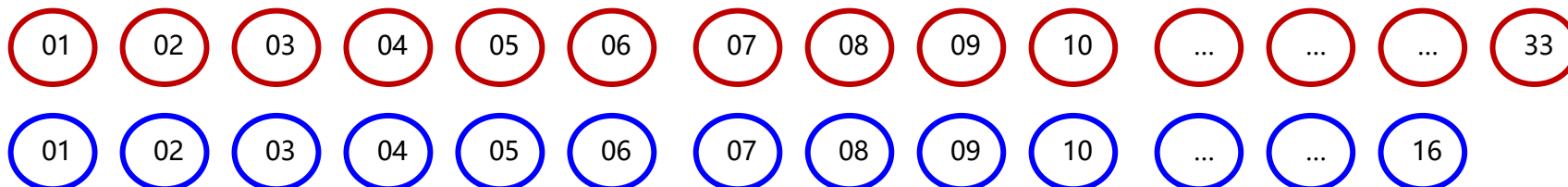
不断地比较数组中相邻的元素，较小者向上浮，较大者向下沉

- 第一步，从第一个元素开始，将相邻的两个元素进行比较，直到最后两个元素完成比较。如果前面的元素比后面的元素大，则交换它们的位置。整个过程完成后，数组中最后一个元素自然就是最大值，这样也就完成了第一轮的比较。
- 第二步，除了最后一个元素，将剩余的元素按照第一步的方法进行两两比较，这样就可以将数组中第二大的元素放到倒数第二个位置上。
- 第三步，以此类推，持续对越来越少的元素重复上面的步骤，直到没有任何一对元素需要比较为止。

【例3-4】为中国福利彩票编写一个双色球的抽奖程序。

- 中国福利彩票的双色球开奖规则是，从编号是 01~33 的红色球中选取6个，从编号是01~16的蓝色球中选取一个

。



准备开奖.....

红色球编号为：04 23 24 28 32 33

蓝色球编号为：01

中国福利彩票双色球抽奖

- 模拟这个抽奖过程，将红色球和蓝色球各自保存在一个 boolean 数组中
 - 数组元素下标：代表球号（从下标1开始使用）
 - 数组元素取值：true/false代表该球是否被选中（初始均为false）。



	false	false	false	false	false		false	false
	red[1]	red[2]	red[3]	red[4]	red[5]	red[32]	red[33]

中国福利彩票双色球抽奖

- 抽奖过程中生成随机数代表开奖球在数组中的编号，如果该球尚未被选出，则将其选中标记置为 true。

随机数选球

01	02	03	04	05	06	07	08	09	10	33
	false	false	false	false	true						false		false
	red[1]	red[2]	red[3]	red[4]	red[5]					red[32]		red[33]

再次选球时，依据数组元素判断是否已选

中国福利彩票双色球抽奖



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY



	false	false	false	false	true		false	false
	red[1]	red[2]	red[3]	red[4]	red[5]	red[32]	red[33]

```
int count=0;
while(count<6){ //不足6个时继续选择红球
    int selectedPos = (int)(Math.random()*33)+1; //生成随机数[1,33]
    if(red[selectedPos]==false){ //未选过的红球可以被选
        red[selectedPos]=true; //置已被选标记
        count++;
    }
}
```

中国福利彩票双色球抽奖



	true	false	false	false	true		false	false
	red[1]	red[2]	red[3]	red[4]	red[5]	red[32]	red[33]

```
System.out.print("红色球编号为: ");  
for(int i=1; i<red.length; i++){  
    if(red[i]==true){  
        System.out.print((i<10?"0"+i:i)+" ");  
    }  
}
```

准备开奖.....

红色球编号为: 04 23 24 28 32 33

蓝色球编号为: 01



```
public static void main(String[] args) {  
    //红色球，使用下标1-33的元素，默认值为false  
    boolean[] red = new boolean[34];  
  
    System.out.println("准备开奖.....");  
    //选择6个红球  
    int count=0;  
    while(count<6) {    //不足6个时继续选择红球  
        //生成随机数[1,33]  
        int selectedPos = (int)(Math.random()*33)+1;  
        if(red[selectedPos]==false) {    //未选过的红球可以被选  
            red[selectedPos]=true;    //置已被选标记  
            count++;  
        }  
    }  
    //输出开奖结果  
    System.out.print("红色球编号为:");  
    for(int i=1; i<red.length; i++){  
        if(red[i]==true) {  
            System.out.print((i<10?"0"+i:i)+" ");  
        }  
    }  
}
```


3.3.2 Java方法中的不定长参数与数组



- 在调用某个方法时，有时会出现方法的参数个数事先无法确定的情况，比如printf方法：

```
System.out.printf("%d",a);
```

```
System.out.printf("%d %d",a, b);
```

```
System.out.printf("%d %d %d",a, b, c);
```

定义时无法事先决定参数的个数。



- Java SE 5.0之后开始支持不定长参数用以解决这个问题。
- 不定长度的形参实为一个数组参数，不定长参数定义的语法格式为：

数据类型... 参数名



不定长的形参只能处于形参列表的最后。一个方法中最多只能包含一个不定长参数。调用包含不定长参数的方法时，既可以向其传入多个参数，也可以传入一个数组。



【例3-5】定义一个对不定个数的一组数进行求和的方法。

- 分析：因为不确定被求和的数字的个数，所以使用不定长形参。

- 带有两个以上下标的数组称为多维数组。
- 在Java语言中，多维数组被看做是**数组的数组**。

3.4.1 二维数组的声明和创建

- 二维数组声明

数据类型 数组引用名[][];

`int[][] array;`

- 二维数组的创建

- **new**

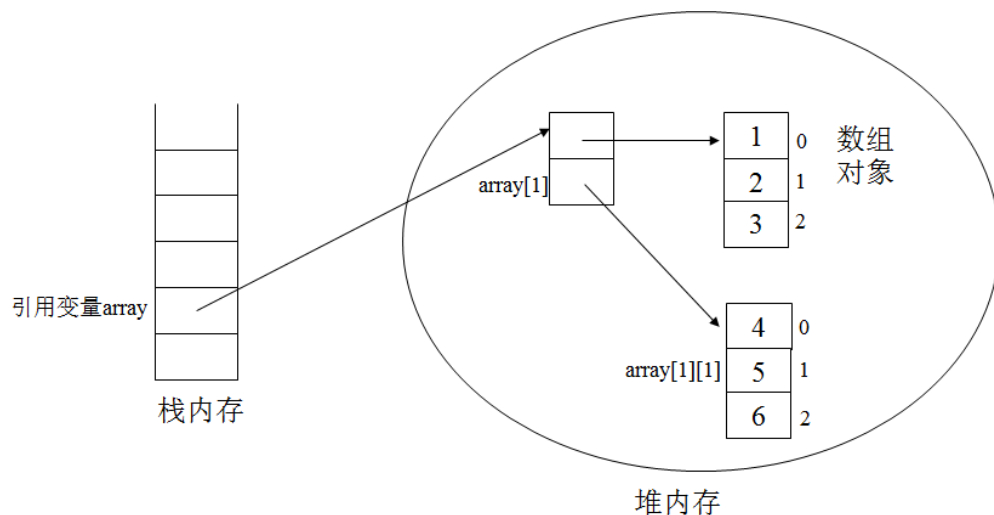
`array = new int[2][3];`

或者:

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

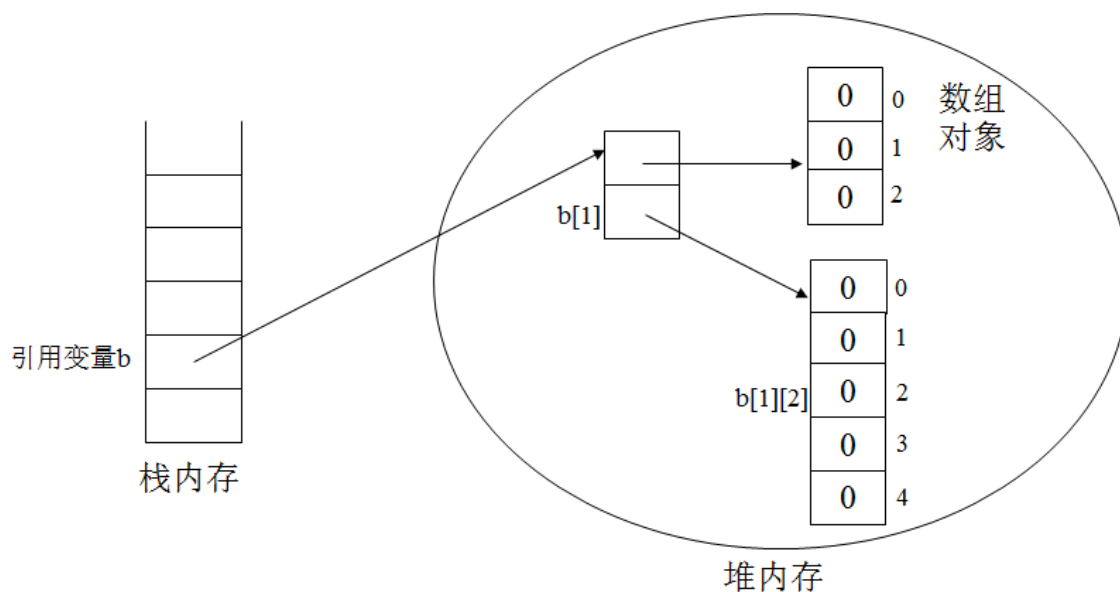
- 通过赋初值的形式创建

`array = {{1,2,3},{4,5,6}};`



- 创建二维数组对象可以进行动态分配

```
int[][][] b= new int[2][];  
b[0]= new int[3];  
b[1] = new int[5];
```



【例3-6】存储并打印杨辉三角形的前n行。

杨辉三角的规律是每行数字的第一列和最后一列的数字都是1，从第三行开始，除去第一列和最后一列都为数字1以外，其余每列的数字都等于它上方两个数字之和。

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
```

Java中的数组对象可以在程序运行的过程中根据需求动态创建，且可以是不规则的。

存储杨辉三角形的前n行

输入要打印的杨辉三角形的行数: 10

0	1									
1	1	1								
2	1	2	1							
3	1	3	3	1						
4	1	4	6	4	1					
5	1	5	10	10	5	1				
6	1	6	15	20	15	6	1			
7	1	7	21	35	35	21	7	1		
8	1	8	28	56	70	56	28	8	1	
9	1	9	36	84	126	126	84	36	9	1

i 设行数为n，则杨辉三角形一共有n行

```
int[][] tri = new int[n][];
```

第i行有i+1个元素

```
tri[i] = new int[i+1];
```


存储杨辉三角形的前n行

输入要打印的杨辉三角形的行数: 10

0	1									
1	1	1								
2	1	2	1							
3	1	3	3	1						
4	1	4	6	4	1					
5	1	5	10	10	5	1				
6	1	6	15	20	15	6	1			
7	1	7	21	35	35	21	7	1		
8	1	8	28	56	70	56	28	8	1	
9	1	9	36	84	126	126	84	36	9	1

$$[i][0] = 1$$

$$[i][i] = 1$$

$$[i][j] = [i-1][j] + [i-1][j-1]$$

i 第i行有i+1个元素

j 第j行有j+1个元素

```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
```

```
    System.out.print("输入要打印的杨辉三角形的行数:");
    int n = scn.nextInt();
    int[][] tri = new int[n][];    //创建n行
```

```
    //第1行第1个元素为0
    tri[0] = new int[1];
    tri[0][0]=1;
```

```
    for(int i=1; i<tri.length; i++){    //一行一行地处理
        tri[i] = new int[i+1];    //行号i从0开始, 第i行有i+1个元素
```

```
        tri[i][0]=tri[i][i]=1;    //第一个和最后一个元素是1
```

```
        //中间每个元素=上一行两个元素之和
```

```
        for(int j=1; j<i; j++){
            tri[i][j]=tri[i-1][j-1]+tri[i-1][j];
        }
```

```
    }
```

```
}
```

输入要打印的杨辉三角形的行数:10

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
```

打印杨辉三角形的前n行



输入要打印的杨辉三角形的行数: 10

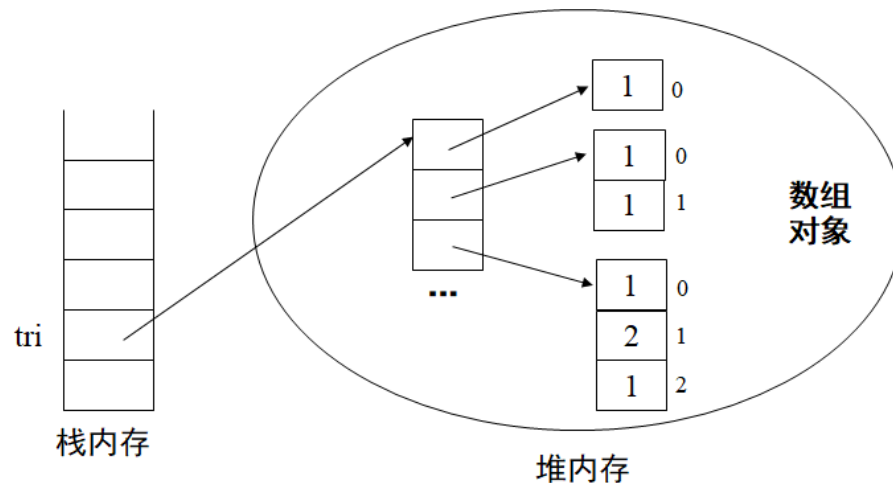
```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
```

```
for(int i=0; i<tri.length; i++){
    for (int j=0; j<tri[i].length; j++){
        System.out.printf("%4d", tri[i][j]);
    }
    System.out.println();
}
```

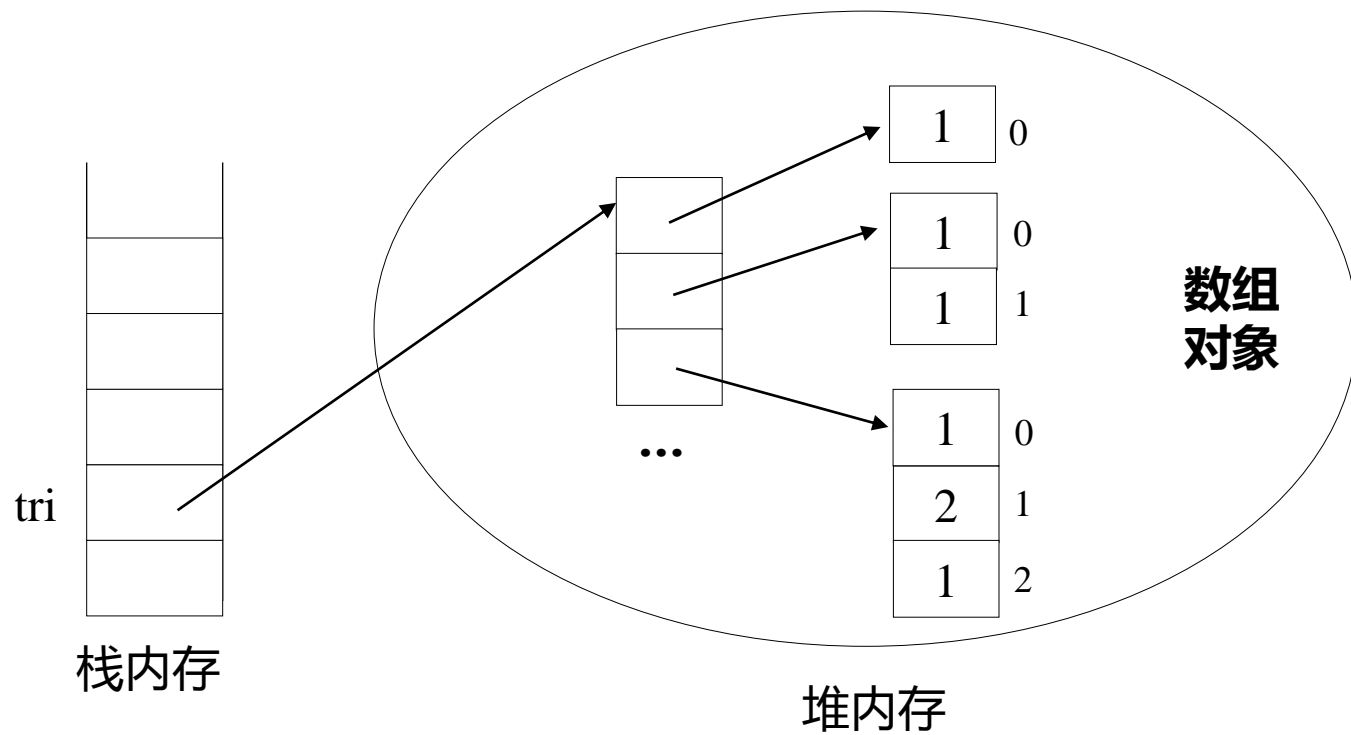
打印杨辉三角形的前n行

输入要打印的杨辉三角形的行数: 10

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
```



```
for(int[] rows: tri){ //rows: 二维数组的每一行
    for(int element :rows){ //element: 行上的每个元素
        System.out.printf("%4d",element);
    }
    System.out.println();
}
```



- Java SE 5.0中增加了一种循环结构，可以用来更便捷地遍历数组中的每个元素，程序设计者不必为指定下标值而分心，称为**for each**循环。

- 语句格式

```
for(数据类型 迭代变量: 数组){  
    //迭代变量即为依次访问的数组中的元素  
}
```

3.5 Java中的for each循环

for each+一维数组

```
int[] array = new int[5];  
for(int element:array){  
    System.out.print(element+" ");  
}
```

for each+二维数组

```
for(int[] rows: tri){  
    //二维数组的每行是一维数组，迭代变量是一维数组元素类型int[]  
    for(int element :rows){  
        //对每行进行迭代，迭代的对象为一维数组元素rows  
        System.out.printf("%4s",element);  
    }  
    System.out.println();  
}
```

【题目】为了防止对网站的恶意注册，用户在注册时通常被要求输入网站提供的验证码。编写一个为网站生成验证码的程序，设验证码是4位，不能重复，验证码由数字和大小写字母组成，但为了避免混淆，不能包含1, l, L, 0, o, O, 2, z, Z, 9, g。

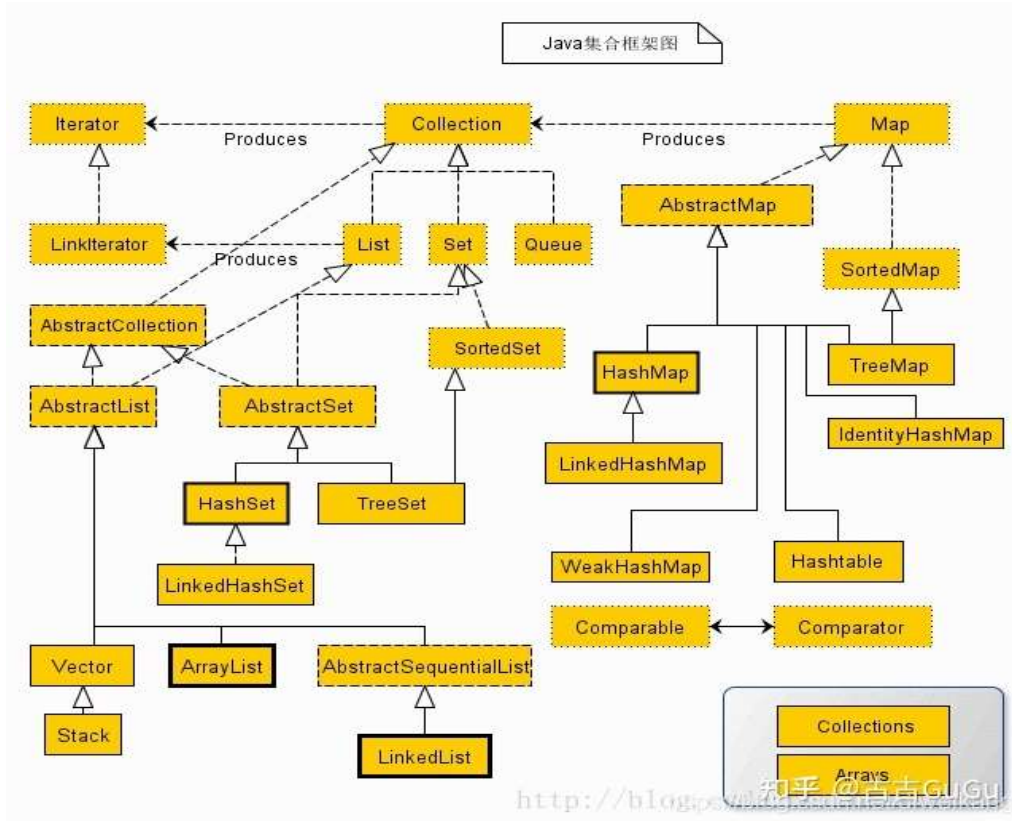
```
char[] code = new
char[]{'3','4','5','6','7','8','a','b','c','d','e','f','h','i','j','k','m','n','p','q','r','s','t',
'u','v','w','x','y','A','B','C','D','E','F','G','H','I','J','K','M','N','P','Q','R','S','T',
'U','V','W','X','Y'};
boolean[] flag = new boolean[code.length];
```


3.6 Arrays类——集合工具类

- Java.util.Arrays类使jdk提供的一个工具类，用来处理数组的各种方法，而且每个方法基本都是静态方法，可以直接通过类名调用。


Java集合类 继承关系图

细虚线边框是接口
粗虚线边框是抽象类
实线是java类



3.6 java.util.Arrays数组工具类

查询API



Java™ Platform
Standard Ed. 6

[All Classes](#)

Packages

- [java.applet](#)
- [java.awt](#)
- [java.awt.color](#)
- [java.awt.datatransfer](#)
- [java.awt.dnd](#)
- [java.awt.event](#)
- [java.awt.font](#)
- [java.awt.geom](#)
- [java.io](#)
- [java.lang](#)
- [java.math](#)
- [java.net](#)
- [java.nio](#)
- [java.rmi](#)
- [java.security](#)
- [java.sql](#)
- [java.text](#)
- [java.time](#)
- [java.util](#)
- [java.util.concurrent](#)
- [java.util.logging](#)
- [java.util.regex](#)
- [java.xml](#)
- [javax.swing](#)
- [javax.swing.text](#)
- [javax.xml](#)

All Classes

- [AbstractAction](#)
- [AbstractAnnotationValueVisitor6](#)
- [AbstractBorder](#)
- [AbstractButton](#)
- [AbstractCellEditor](#)
- [AbstractCollection](#)
- [AbstractColorChooserPanel](#)
- [AbstractDocument](#)
- [AbstractDocument.AttributeCont](#)
- [AbstractDocument.Content](#)
- [AbstractDocument.ElementEdit](#)
- [AbstractElementVisitor6](#)
- [AbstractExecutorService](#)
- [AbstractInterruptibleChannel](#)
- [AbstractLayoutCache](#)
- [AbstractLayoutCache.NodeDimc](#)
- [AbstractList](#)
- [AbstractListModel](#)

Overview Package Class Use Tree Deprecated Index Help

PREV NEXT

[FRAMES](#) [NO FRAMES](#)

Java™ Platform, Standard Edition 6
API Specification

This document is the API specification for version 6 of the Java™ Platform, Standard Edition.

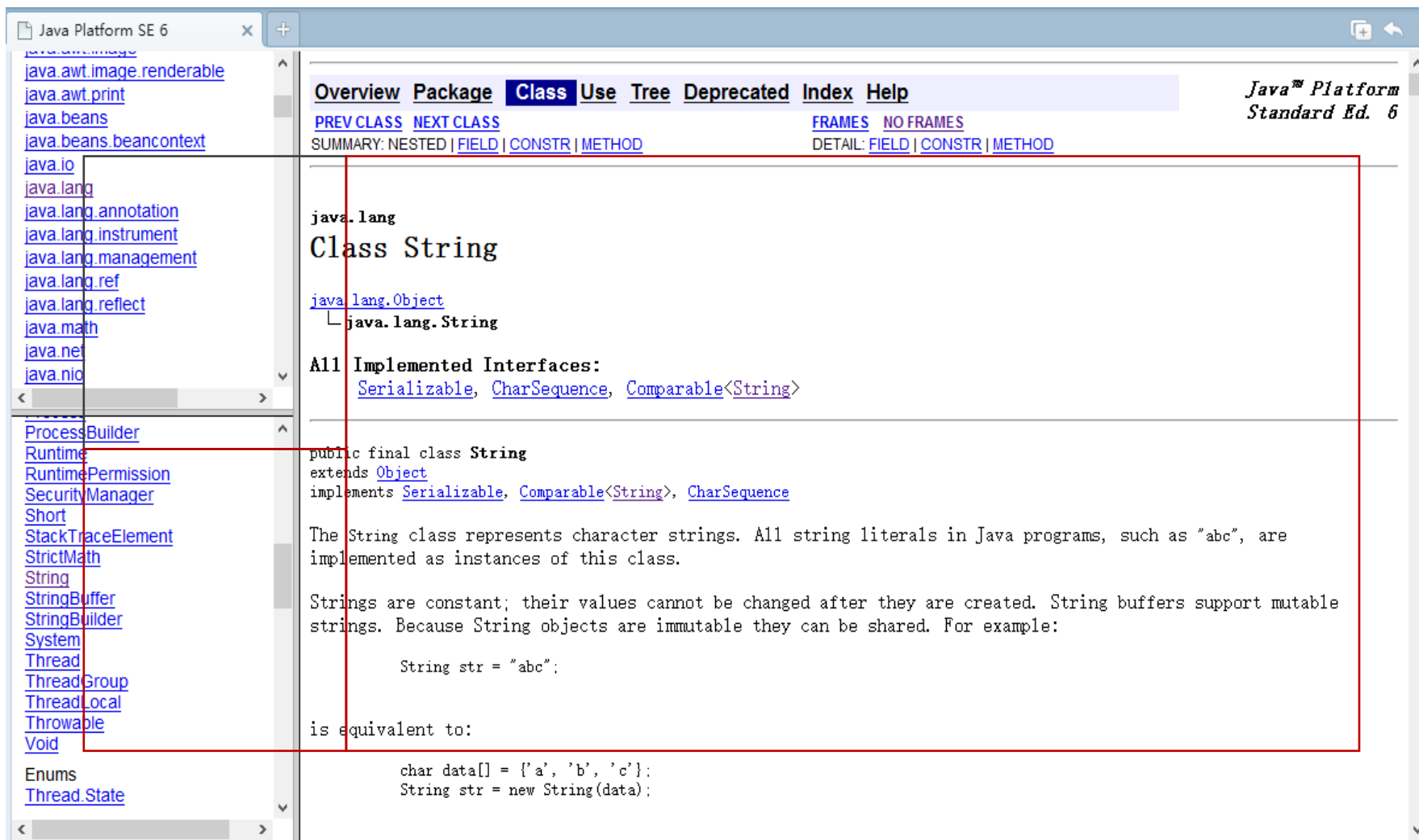
See: [Description](#)

Packages	
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.

3.6 java.util.Arrays数组工具类

- API窗口分为三个窗格
 - 左上方窗格：显示可以使用的所有包
 - 左下窗格：列出所有的类。浏览单击左上窗格链接可以控制左下窗格中显示该包下所有的接口、类等
 - 右侧窗格：单击左下窗格的某一个类的链接后，这个类的API文档就会显示在右侧的大窗格中

3.6 java.util.Arrays数组工具类



The screenshot shows the Java Platform SE 6 documentation for the `String` class. The left sidebar lists various Java packages and classes, with `String` selected. The main content area displays the class details for `String`.

Overview Package Class Use Tree Deprecated Index Help

[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#)

SUMMARY: [NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#) [DETAIL: FIELD](#) | [CONSTR](#) | [METHOD](#)

java.lang

Class String

[java.lang.Object](#)
└─ [java.lang.String](#)

All Implemented Interfaces:
[Serializable](#), [CharSequence](#), [Comparable<String>](#)

public final class **String**
extends [Object](#)
implements [Serializable](#), [Comparable<String>](#), [CharSequence](#)

The `String` class represents character strings. All string literals in Java programs, such as "abc", are implemented as instances of this class.

Strings are constant; their values cannot be changed after they are created. String buffers support mutable strings. Because `String` objects are immutable they can be shared. For example:

```
String str = "abc";
```

is equivalent to:

```
char data[] = {'a', 'b', 'c'};  
String str = new String(data);
```

3.6 java.util.Arrays数组工具类

- **查阅**java.util.Arrays类的API—sort()方法
- Arrays类中的排序方法使用的是优化的**快速排序算法**

```
public static void main(String[] args) {  
    int[] a = new int[]{32, 32, 96, 10, 29, 55};  
  
    //以"[32, 32, 96, 10, 29, 55]"形式打印输出  
    System.out.println(Arrays.toString(a));  
  
    //对数组a进行排序  
    Arrays.sort(a);  
  
    //打印输出数组  
    System.out.println(Arrays.toString(a));  
}
```

3.6 java.util.Arrays数组工具类

- **查阅**java.util.Arrays类的API—binarySearch()方法
- 折半查找要求数组有序

```
public static void main(String[] args) {  
    double[] myArray = { 2.9, 3.4, 1.9, 3.5 };  
    Arrays.sort(myArray); // {1.9, 2.9, 3.4, 3.5}  
    System.out.println(Arrays.binarySearch(myArray, 3.5));  
    System.out.println(Arrays.binarySearch(myArray, 10));  
}
```

sort () 方法--升序排序

```
public static void sort(int[] a)
```

Sorts the specified array into ascending numerical order.

Implementation note: The sorting algorithm is a Dual-Pivot Quicksort by Vladimir Yaroslavskiy, Jon Bentley, and Joshua Bloch. This algorithm offers $O(n \log(n))$ performance on many data sets that cause other quicksorts to degrade to quadratic performance, and is typically faster than traditional (one-pivot) Quicksort implementations.

Parameters:

a - the array to be sorted

```
APIDemo.java x
1 import java.util.Arrays;
2
3 public class APIDemo {
4
5     public static void main(String[] args) {
6
7         int[] a = {25,162,83,139,1};
8
9         Arrays.sort(a);
10    }
11 }
```

toString () 方法--将数组转换成String类型输出



```
public static String toString(int[] a)
```

Returns a string representation of the contents of the specified array. The string representation consists of a list of the array's elements, enclosed in square brackets ("[]"). Adjacent elements are separated by the characters ", " (a comma followed by a space). Elements are converted to strings as by `String.valueOf(int)`. Returns "null" if `a` is null.

Parameters:

`a` - the array whose string representation to return

Returns:

a string representation of `a`

Since:

1.5

APIDemo.java x

```
1 import java.util.Arrays;
2
3 public class APIDemo {
4
5     public static void main(String[] args) {
6
7         int[] a = {25,162,83,139,1};
8
9         Arrays.sort(a);
10
11         System.out.println(Arrays.toString(a));
12     }
13 }
```

Problems @ Javadoc Declaration Console x

```
<terminated> APIDemo [Java Application] C:\Program Files
[1, 25, 83, 139, 162]
```


binarySearch () 方法--折半查找

```
public static int binarySearch(int[] a,  
                               int key)
```

Searches the specified array of ints for the specified value using the binary search algorithm. The array must be sorted (as by the `sort(int[])` method) prior to making this call. If it is not sorted, the results are undefined. If the array contains multiple elements with the specified value, there is no guarantee which one will be found.

Parameters:

a - the array to be searched

key - the value to be searched for

Returns:

index of the search key, if it is contained in the array; otherwise, `-(insertion point) - 1`. The *insertion point* is defined as the point at which the key would be inserted into the array: the index of the first element greater than the key, or `a.length` if all elements in the array are less than the specified key. Note that this guarantees that the return value will be `>= 0` if and only if the key is found.

```
APIDemo.java
1 import java.util.Arrays;
2
3 public class APIDemo {
4
5     public static void main(String[] args) {
6
7         int[] a = {25,162,83,139,1};
8
9         Arrays.sort(a);
10        System.out.println(Arrays.toString(a));
11
12        int res;
13        res = Arrays.binarySearch(a, 100);
14        System.out.println(res);
15
16        res = Arrays.binarySearch(a, 162);
17        System.out.println(res);
18    }
19 }
```

```
Problems @ Javadoc Declaration Console
<terminated> APIDemo [Java Application] C:\Program I
[1, 25, 83, 139, 162]
-4
4
```



```
public static int binarySearch(int[] a,  
                               int fromIndex,  
                               int toIndex,  
                               int key)
```

Searches a range of the specified array of ints for the specified value using the binary search algorithm. The range must be sorted (as by the `sort(int[], int, int)` method) prior to making this call. If it is not sorted, the results are undefined. If the range contains multiple elements with the specified value, there is no guarantee which one will be found.

Parameters:

`a` - the array to be searched

`fromIndex` - the index of the first element (inclusive) to be searched

`toIndex` - the index of the last element (exclusive) to be searched

`key` - the value to be searched for

Returns:

index of the search key, if it is contained in the array within the specified range; otherwise, $-(\text{insertion point}) - 1$. The *insertion point* is defined as the point at which the key would be inserted into the array: the index of the first element in the range greater than the key, or `toIndex` if all elements in the range are less than the specified key. Note that this guarantees that the return value will be ≥ 0 if and only if the key is found.

[1, 25, 83, 139, 162]

APIDemo.java x

```
1 import java.util.Arrays;
2
3 public class APIDemo {
4
5     public static void main(String[] args) {
6
7         int[] a = {25,162,83,139,1};
8
9         Arrays.sort(a);
10        System.out.println(Arrays.toString(a));
11
12        int res;
13        res = Arrays.binarySearch(a, 1, 4, 83);
14        System.out.println(res);
15    }
16 }
```

[1, 25, 83, 139, 162]

Problems @ Javadoc Declaration Console x

<terminated> APIDemo [Java Application] C:\Program

[1, 25, 83, 139, 162]

2

3.6.1 sort()方法

- Arrays类中的排序方法使用的是优化的快速排序算法。
- sort方法是Arrays类中的静态方法，可以通过类直接进行调用
- sort方法有各种参数类型的重载版本，可以实现对各种数据类型数组的排序

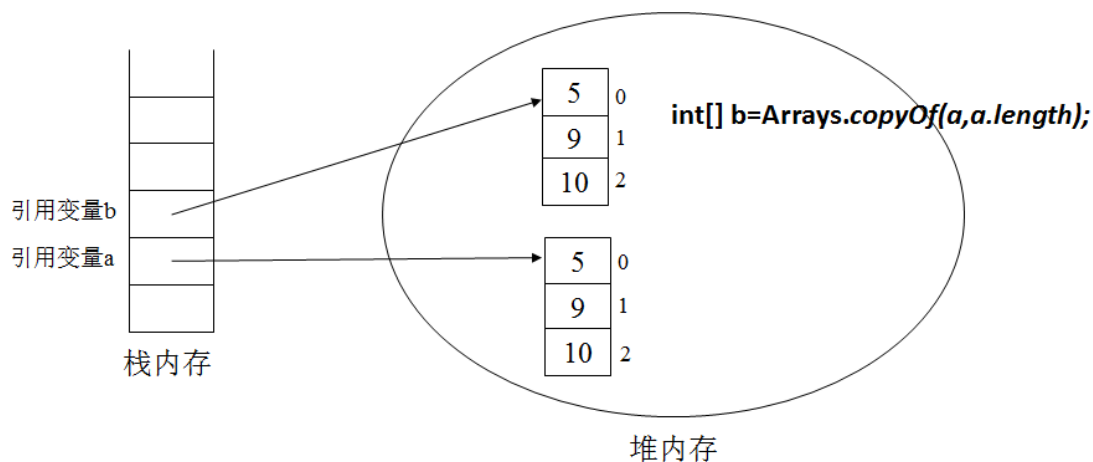
3.6.1 sort()方法

【例3-7】 利用Arrays类对int型数组进行排序。

```
public static void main(String[] args) {  
    int[] a = new int[]{32, 32, 96, 10, 29, 55};  
  
    System.out.println(Arrays.toString(a));  
    //以"[32, 32, 96, 10, 29, 55]"形式打印输出  
  
    Arrays.sort(a); //对数组a进行快速排序  
  
    System.out.println(Arrays.toString(a));  
    //以"[10, 29, 32, 32, 55, 96]"形式打印输出  
}
```

3.6.2 copyOf()方法

- Arrays类中的copyOf方法能够实现数组的复制，两种格式：
 - type copyOf(type[] a, int length)*
 - type copyOf(type[] a, int start, int end)*
- copyOf实现的复制，已经使目标数组脱离了源数组，即复制得到一个新的数组对象



3.7 综合实践—学生成绩查询系统



- 1、输入学生姓名以及课程名称
- 2、输入每个学生每门课程的成绩
- 3、计算每门课程学生的平均成绩
- 4、查询某个学生某门课程的成绩
- 5、按照某门课程的成绩排序
- 6、退出系统
- 7、需要判断课程或者学生是否存在

	C	Java	mySQL	Linux	HTML
zhang	26	69	46	25	5
wang	27	10	24	66	58
li	44	58	0	82	75
zhao	6	68	92	9	84
liu	75	1	51	41	74
song	23	38	65	1	55

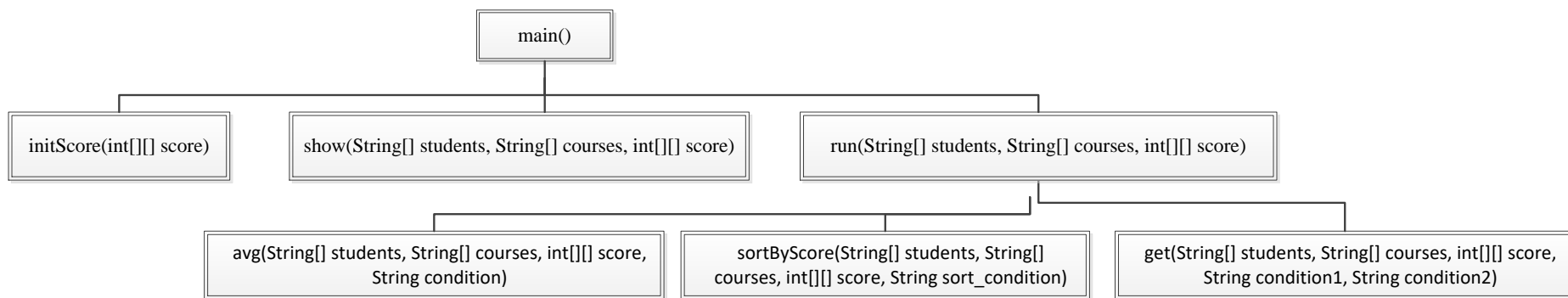
```
请输入命令:avg java
java的平均分是: 40.67
请输入命令:avg song
song的平均分是: 36.40
请输入命令:avg yan
你输入的既不是课程名,也不是学生名
请输入命令:get song java
song的java的成绩是: 38
请输入命令:get yan java
没有 yan 这个人
请输入命令:get song c++
song没有 c++ 这门课程
请输入命令:sort java
名次    姓名    Java
1       liu     1
2       wang    10
3       song    38
4       li      58
5       zhao    68
6       zhang   69
请输入命令:sort c++
没有这门课程
请输入命令:exit
退出查询系统! byebye!
```

3.7.1 查询系统的数据结构

```
final int STUDENT_NUM=6;  
final int COURSE_NUM=5;  
static String[] students=  
    {"zhang","wang","li","zhao","liu","song"};  
static String[] courses=  
    {"C","Java","mySQL","Linux","HTML"};  
static int[][] score=  
    new int[STUDENT_NUM][COURSE_NUM];
```

	C	Java	mySQL	Linux	HTML
zhang	23	27	72	91	87
wang	93	1	20	95	27
li	36	18	65	3	17
zhao	42	58	37	40	40
liu	59	40	16	54	37
song	26	59	33	83	79

3.7.2 模块化设计



1、初始化方法

```
public static void initScore(){ //用随机数初始化成绩
    for(int i=0; i<score.length; i++)
        for(int j=0; j<score[i].length; j++)
            score[i][j]=(int)(Math.random()*101);
}
```

2、显示成绩方法



```
public static void show(){ //显示成绩
    System.out.print("\t"); //留出显示姓名的位置
    for(int i=0; i< courses.length; i++){//输出课程名称
        System.out.print(courses[i] + "\t");
    }
    System.out.println();
    for(int i=0; i< score.length; i++){
        //显示学生姓名
        System.out.print(students[i] + "\t");
        //显示该学生成绩
        for(int j=0; j< score[i].length; j++){
            System.out.print(score[i][j] + "\t");
        }
        System.out.println();
    }
}
```

3.7.3 控制台命令的读取和控制run()

- get+姓名+课程名
- avg+课程名
- avg+姓名
- sort+课程名
- exit

字符串比较方法: equalsIgnoreCase()

3.7.3 控制台命令的读取和控制run()



```
public static void run(){
    Scanner scn=new Scanner(System.in);
    while(true){
        System.out.print("请输入命令:");
        String command=scn.next();
        if(command.equalsIgnoreCase("avg")){//"avg"命令需要一个参数
            String parameter=scn.next();
            avg(parameter);
        }
        if(command.equalsIgnoreCase("get")){//"get"命令需要两个参数
            String parameter1=scn.next();
            String parameter2=scn.next();
            get(parameter1,parameter2);
        }
        if(command.equalsIgnoreCase("sort")){//"sort"命令需要一个参数
            String parameter=scn.next();
            sortByScore(parameter);
        }
        if (command.equalsIgnoreCase("exit")){//退出查询系统
            System.out.println("退出查询系统! byebye! ");
            System.exit(0);
        }
    }
}
```

3.7.4 查询某人某门课成绩get()

```
get(String[] students, String[] courses,  
    int[][] score,  
    String condition1,  
    String condition2){  
  
    .....  
}
```

