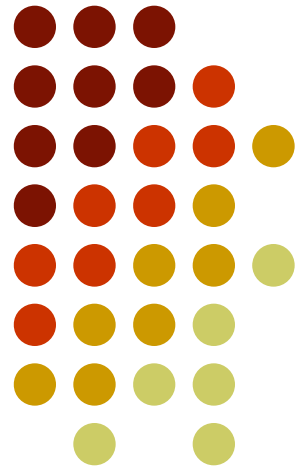


第五章 数组和字符串

- 5.1 数组的基本用法
- 5.2 多维数组
- 5.3 Java中的字符串
- 5.4 字符串操作
- 5.5 数组和字符串应用





5.1数组

- 在java中完全作为对象来处理
- 由**类型相同的**元素组成的有顺序的数据集合
- 数组是**固定的**不能动态扩展
- 可以存储基本数据类型或对象
- 通过数组名和下标可以访问各元素

5.1 数组的基本用法

- 数组在其他语言中是相同数据类型元素的集合；而在Java中，**数组是独立的类**，有自身的方法，**不是变量的集合**
- 一个数组是数组类的一个对象。
- 数组是通过**new**运算符来动态创建，并分配与数组下标序号和维数、数组数据类型相匹配的内存空间。数组一经被分配了适当的空间，则其所有元素都初始化为默认值。
- Java中，数组声明允许方括号跟在数组类型声明之后。
- 数组的声明

int intArray []; ——（[]不是指长度可变）
或者：Int [] intArray;



5.1数组的基本用法

- 数组的长度不是在声明时指定，而是在创建时由所开辟的内存单元数目确定；
 - 数组的创建（3步）
 1. 数组的声明
 2. 创建数组空间
 3. 创建数组元素并初始化。
- * 若数组元素为基本数据类型，第三步可以省略；当数组元素为对象时，则必须对数组元素进行创建和初始化。



5.1.1 数组声明

可以声明任何类型的数组——原始类型或类类型：

```
int x[ ];
```

```
char [ ] cc;
```

```
String ss[ ];
```

```
Myclass p [ ]; // here MyClass is a class
```

二维数组

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

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

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

在Java编程语言中，即使数组是由原始类型构成，或带有其它类类型，数组也是一个对象。声明不能创建对象本身，而创建的是一个引用，该引用可被用来引用数组。数组元素使用的实际存储器可由new语句或数组初始化软件动态分配。

5.1.2 创建数组空间

- 方法一：先声明后创建

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

- 方法二：在声明的同时创建

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

- 方法三：直接赋值

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

```
public static void main(String[] args)  
{  
    int i, intArray[];  
    intArray = new int[5];  
    // int intArray[] = new int[5];  
    //int intArray[] =  
        { 10, 20, 30, 40, 50 } ;  
    for (i=0; i<5;i++) {  
        intArray[i]=10+i*10;  
        System.out.println("A["+i+  
            "]="+intArray[i]);  
    }  
}
```

数组元素为类的对象

➤ 定义数组

```
Integer [] a;
```

```
Integer [] b = new Integer[5];
```

➤ 创建数组元素的对象实例

```
for(int i = 0; i < b.length; i++)  
    b[i] = new Integer(i*10);
```

➤ 创建数组时初始化

```
Integer [] d = {new Integer(1), new Integer(2), new  
Integer(3)};
```

➤ 例: Array42.java

✓ 数组元素为对象的创建

✓ 数组元素为基本数据类型的创建

```
public static void main(String[] args) {  
    int i;  
    Integer [] b = new Integer[10];  
  
    for( i = 0; i < b.length; i++)  
    { b[i] = new Integer(i*10);  
      System.out.println(i+": "+b[i]);  
    }  
}
```

数组

```
int[ ] intArray = { 11, 47, 93, 26, 38 };
```

- ✓ 分配空间+赋初值
- ✓ 静态初始化必须与数组定义放在一个语句中。

•intArray

•XXXX



•11

•47

•93

•26

•38

- `intArray = new int[10];` //原数组丢失
- 数组定义后还不能立即被访问，因为还没有为其分配内存空间。数组必须经定义后再创建这一步才能使用。



字符型数组:

```
String faces[] = { "A-Ace", "2-Deuce", "3-Three", "4-Four",  
                  "5-Five", "6-Six", "7-Seven", "8-Eight",  
                  "9-Nine", "10-Ten", "J-Jack", "Q-Queen",  
                  "K-King" };
```

```
String suits[] = { "Hearts红心", "Diamonds方块",  
                  "Clubs梅花", "Spades黑桃" };
```

数组的传递

➤ 如何把数组传递给方法

//数组b各元素 加倍

```
void doubleArray(Integer c[]){  
    for( int i = 0; i < c.length; i++)  
        c[i]*=2;  
}
```

方法参数表中
要加[]表示数组

调用时不加 “[]”

```
doubleArray(b); // 调用
```



Example: Bubble Sort(ArraySort1.java)

```
class BubbleSort {  
    // sort the elements of an array with bubble sort  
    public void bubbleSort( int b[] )  
    {  
        for ( int pass = 1; pass < b.length; pass++ ) // passes  
            for ( int i = 0; i < b.length - pass; i++ ) // one pass  
                if ( b[ i ] > b[ i + 1 ] ) // one comparison  
                    swap( b, i, i + 1 ); // one swap  
    }  
    // swap two elements of an array  
    public void swap( int c[], int first, int second )  
    {  
        int hold; // temporary holding area for swap  
        hold = c[ first ];  
        c[ first ] = c[ second ];  
        c[ second ] = hold;  
    }  
}
```



Example: Test of Bubble Sort

```
public class ArraySort1 {  
    public static void main(String[] args) {  
        int a[ ] = { 2, 78, 99, 6, 4, 8, 10, 12, 89, 68, 45, 37 };  
        String output = "Data items in original order\n";  
        for ( int i = 0; i < a.length; i++ )  
            output += "  " + a[ i ];  
  
        BubbleSort st = new BubbleSort();  
        st.bubbleSort( a );  
  
        output += "\n\nData items in ascending order\n";  
        for ( int i = 0; i < a.length; i++ )  
            output += "  " + a[ i ];  
        System.out.println( output );  
    }  
}
```

5.2 多维数组定义

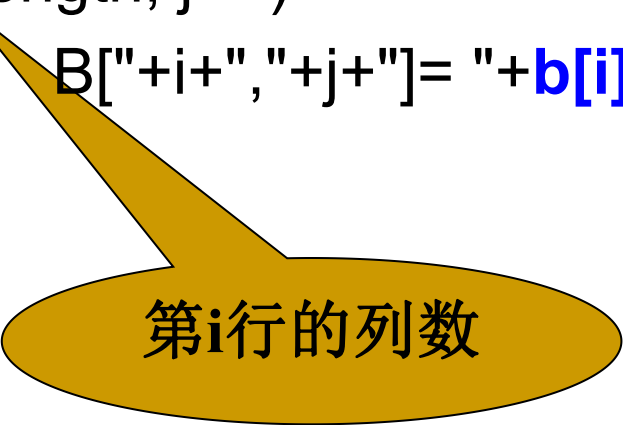
- 在Java中，对多维数组的定义是通过对一维数组的嵌套式定义来实现，即数组的数组
 `type arryName[][];`
- 定义之后，须用关键字new建立数组，即为其分配内存空间，可采用两种方法：
 - ✓ 一步完成二维分配
 如: `int [][] intArray = new int [2][8];`
 - ✓ 分别为每一维分配存储空间，但须从高维开始
 如: `int [][] intArray=new int[2][];` //最高维含2个元素
 `intArry[0] = new int[8];` //第二维第一个元素为8个整型数
 `intArry[1] = new int[8];` //第二维第二个元素为8个整型数
- `int [][] intArray = { {1,3}, {2,4}, {1,2,3,4} };` //每行元素可以不等

数组类

- 数组使用之前要确定大小，可以在程序中根据需要指定其大小（如用表达式计算出大小）。
- 一旦生成数组，**大小不可改变。java不支持变长数组。**
- 数组的属性— **length** 代表数组元素的个数
数组是一个对象, **length**为数组的成员变量, 调用它可查看数组长度
- **Java**数组作为一种对象类型，可以作为方法的参数，传递的是指向数组的引用。
- 越界例外
用下标变量访问数组元素(0~N-1)，越界时系统会产生例外：
ArrayIndexOutOfBoundsException 例外

二维数组举例

```
public class Array44 {  
    public static void main(String[] args) {  
        int [ ][ ] b = { {1,3},{2,4},{1,2,3,4} }; //每行元素可以不等  
        for( int i = 0; i < b.length; i++){  
            for( int j = 0; j < b[i].length; j++){  
                System.out.print("    B["+i+", "+j+"]= "+b[i][j]);  
                System.out.println();  
            }  
        }  
    }  
}
```



例5.7 存储并输出九九乘法表

- 例5.7 存储并输出九九乘法表。存储并输出九九乘法表程序，主要说明不同长度二维数组的应用 程序的名字： **Multiplication_table**

```
int mulTable[][] = new int[9][]; // 定义二维数组有9行，列未定义
for (int i = 1; i <= 9; i++) {
    mulTable[i - 1] = new int[i]; // 定义各行的大小（列数）
    for (int j = 1; j <= i; j++)
        mulTable[i - 1][j - 1] = i * j; // 计算乘法表
}
```




5.3 Arrays类和数组应用实例

- 在java.util.Arrays类包中提供了Arrays类，用于对数组进行诸如排序、比较、转换、搜索等运算操作。它提供的所有方法都是静态的。详细使用的请参考API文档。
- Arrays类的方法：

1、static void fill(array, [int from, int to,] val);

功能： 设置array数组从from到to位置的元素的值为val，如果没有from和to参数，就给各个元素的值赋为val。

2、static void sort(数据类型 [] array [, int from,int to]);

功能： 用于对数组array进行排序（升序）

3、static boolean equals(数据类型[] d1,数据类型[] d2)

功能： 判断d1和d2两数组是否相等。

4、static int binarySeach(数据类型[] a,数据类型 key);

功能： 利用二进制搜索已经排序的数组内的元素值为key的所在位置。



5.3 Arrays类和数组应用实例

5.数组遍历

(1) Arrays工具类中toString静态方法遍历

利用Arrays工具类中的toString静态方法可以将一维数组转化为字符串形式并输出。

```
String s=Arrays.toString(arrA);
```

(2) 字符型和字符串的迭代器遍历

```
Arrays.asList(arrA).stream().forEach(x ->  
System.out.println(x));
```

也可以这样写：

```
Arrays.asList(test).stream().forEach(System.out::println);
```

具体用法见程序【EX5-7: ArrayTraver.java】。



实验7 数组

2020-11-4 #7

复习实验5

- 4-2: 用数组随机生成50个2位整数，去掉数组中的重复元素；
- 4-3: 统计一段英文中每个字母出现的次数。
- 4-4 编写一个使用数组实现折半查找程序 (BiSearch.java);

(如果没有完成，请做完)

2022-04-08 #6



实验7 数组

练习5

1. 给Ex0503.java加上调用语句（或方法），并将数组反转。
2. 求两个矩阵乘积的程序。(ProductOfMatrix.java)
3. 数组遍历（ArrayTraver.java）

（1）Arrays工具类中toString静态方法遍历

利用Arrays工具类中的toString静态方法可以将一维数组转化为字符串形式并输出。

```
String s=Arrays.toString(arrA);
```

（2）字符型和字符串的迭代器遍历

```
Arrays.asList(arrA).stream().forEach(x -> System.out.println(x));
```

也可以这样写：

```
Arrays.asList(test).stream().forEach(System.out::println);
```

5号机房 ftp://10.20.34.34 stu

5.3 JAVA中的字符串

- Java中的字符串也是类。
- 由于使用频繁，有时也被视为基本类型
 - ✓ 遇到双引号自动创建String类的对象
 - ✓ 提供字符串运算符，字符串的连接：+
- 定长字符串：String类（效率较高）
 - ✓ 不能更改
- 可变字符串：StringBuffer类(使用灵活)
 - ✓ 可追加、插入、修改，但内存管理复杂

引用类型—字符串类

- Java语言将字符串作为对象来处理，每一个字符串常量是字符串类String的一个实例。

- 构造方法

String()、String(String)、String(char chars[])

- 创建字符串

1. String s = new String(); //生成一个空字符串
2. String s = new String("abc");
3. char[] data= {'a', 'b', 'c'};
String s = new String(data);
4. String s = "abc"; //采用字符串直接定义获得内存空间

- Here's an example of creating a string from a character array:

```
char[] helloArray = { 'h', 'e', 'l', 'l', 'o' };  
String helloString = new String(helloArray);  
System.out.println(helloString);
```

引用类型—字符串类

```
int x=123;
```

```
int y=x ;
```

```
String s1="abc" ;
```

```
String s2 = s1 ;
```

•x

•123

•y

•123

•s1

•0x01234567

•s2

•0x01234567

•s2="def"

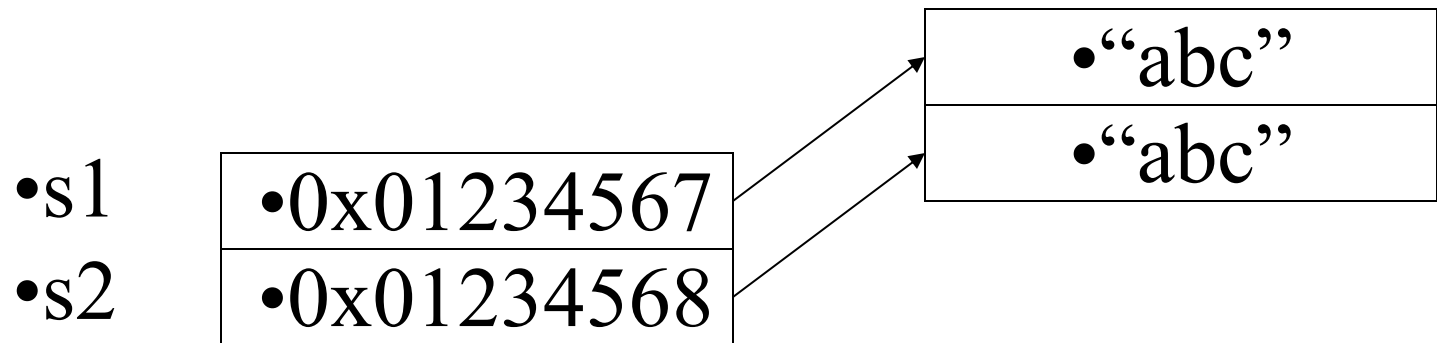
•“abc”

•“def”

引用类型—字符串

```
String s1 = new String("abc");
```

```
String s2 = new String("abc");
```



•s1 == s2

false ?

•s1.equals(s2)

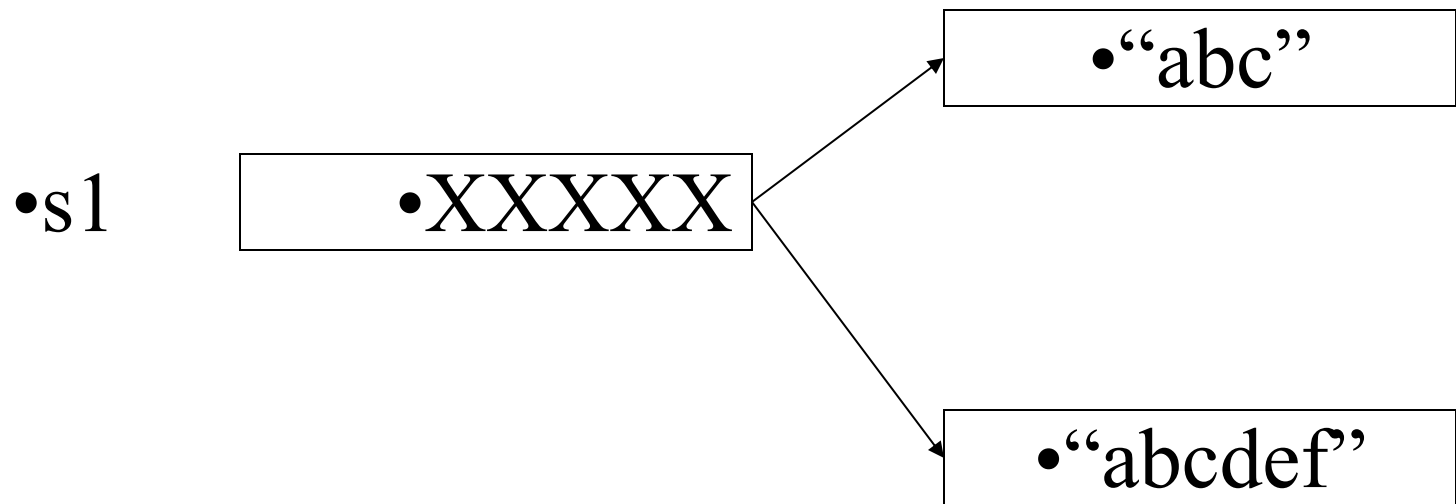
true



```
public static void main(String[] args) {  
    //String myName = "abc";  
    //String yourName = "abc";           //true  
    String myName = new String("abc");  
    String yourName = new String("abc"); //false  
    boolean ourName = myName.equals(yourName);  
    boolean as= myName == yourName;  
    System.out.println(ourName);  
    System.out.println(as);// myName == yourName);  
}
```

引用类型—字符串

```
String s1 = "abc" ;  
s1=s1 + "def" ;
```



字符串类-String

➤ 常用方法

- `int length()` // 返回字符串长度
例: “Java与面向对象程序设计”.length() 的值为13
- `char charAt(int index)` //返回指定位置的字符
例: "Java与面向对象程序设计".charAt(2) v
- `int compareTo(String s2)` //按字母序进行字符串比较
- `boolean equals(Object obj)`//判断字符串相等(区分大小写)
- `boolean equalsIgnoreCase(String s2)` //(不区分大小写)
- `String toLowerCase()`//将字符串所有大写字母转换为小写
- `String toUpperCase()`//将字符串所有小写字母转换为大写



字符串类-String

- String substring(int beginIndex) //取子串
String substring(int beginIndex, int endIndex)
- int indexOf(String str) //返回str在当前串中开始位置
int indexOf(String str, int fromIndex)
- boolean startsWith(String prefix) //判断该字符串是否以prefix为前缀。
- boolean endsWith(String suffix) //判断该字符串是否以suffix为后缀。
- char[] toCharArray() //将字符串转为字符数组
char[] letters=str1.toCharArray();



String.valueOf()方法

// Fig. 10.10: StringValueOf.java
// 各种类型数据转换为String

```
import javax.swing.*;  
public class StringValueOf {  
    public static void main( String args[] )  
    {  
        char charArray[] = { 'a', 'b', 'c', 'd', 'e', 'f' };  
        boolean b = true;  
        char c = 'Z';  
        int i = 7;  
        long l = 10000000;  
        float f = 12345678.12345f;  
        double d = 333333333.333333;  
        Object o = "hello"; // Assign to an Object reference  
        String output;
```

String.valueOf()方法

```
output = "char array = " + String.valueOf( charArray ) +  
        "\npart of char array = " +  
        String.valueOf( charArray, 3, 3 ) +  
        "\nboolean = " + String.valueOf( b ) +  
        "\nchar = " + String.valueOf( c ) +  
        "\nint = " + String.valueOf( i ) +  
        "\nlong = " + String.valueOf( l ) +  
        "\nfloat = " + String.valueOf( f ) +  
        "\ndouble = " + String.valueOf( d ) +  
        "\nObject = " + String.valueOf( o );
```



```
JOptionPane.showMessageDialog(null, output, "String Class valueOf  
Methods",
```

```
JOptionPane.INFORMATION_MESSAGE );  
System.exit( 0 );
```

```
}
```

```
}
```



```
int a=10;  
int b=5;  
System.out.println("a+b="+a+b);  
System.out.println(a+b);  
System.out.println("a+b="+a+b);
```

a+b=105

15

a+b=15

String类转换为数值

Converting Strings to Numbers

```
String piStr = "3.14159";  
String xx="12345";  
Float pi = Float.valueOf(piStr);  
Int ix=Integer.valueOf(xx);  
Int x = Integer.parseInt(xx)
```



严重关注

```
public class ArgAdd {  
    public static void main(String[] args) {  
        int aInt=0;  
        if(args.length<2){  
            System.out.println("Too short arguments!");  
            System.exit(0); }  
        for (int i=0;i<args.length;i++ ){  
            aInt+=Integer.parseInt(args[i]);  
            if (i==0)  
                System.out.print("Result = "+args[i]);  
            else  
                System.out.print(" + "+args[i]);  
        }  
        System.out.println(" = "+aInt);  
    } }  
}
```

Result = 1 + 2 + 3 + 4 + 5 + 6 + 7 = 28



String类



创建字符串对象

- 使用字符串字面值创建String对象。

```
String str = "Java is cool";
```

- 使用String类的构造方法。

```
String str = new String("Java is cool");
```

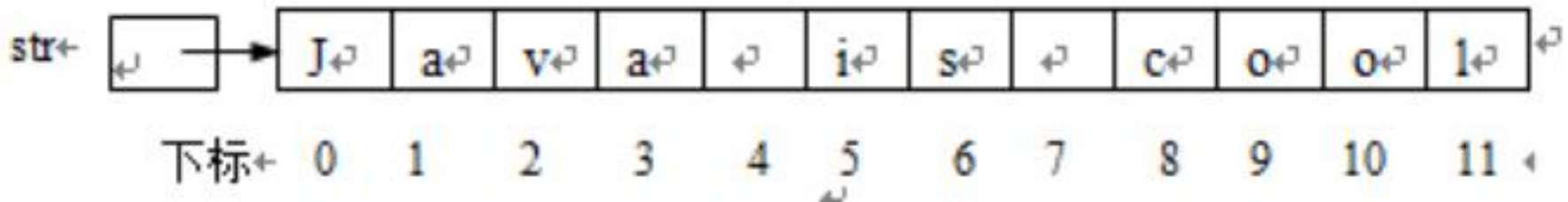
- 这两种方式有区别，后面说明。



字符串基本操作

- 字符串在内存的表示。设有下面声明：

```
String str = new String("Java is cool");
```



- 这个字符串共有12个字符，每个字符有一个下标，下标从0开始。 `str.charAt(6)`返回字母s。



字符串基本操作

问题描述

- 编写一个方法判断字符串是否是回文串。

```
public static boolean isPalindrome(String s)
```

- 思路：取出字符串的第一个和最后一个比较，若不相同，程序结束，返回false。若相等，比较第二个字符和倒数第二个字符，直到比较到字符串的中间字符为止，若都相等，则是回文，返回true。



字符串查找

- `int indexOf(int ch)`
- `int lastIndexOf(int ch)`
- 找到返回下标值，找不到返回-1。



字符串转换为数组

➤ `char[] toCharArray()`

➤ `byte[] getBytes()`

•

•

•

•

•



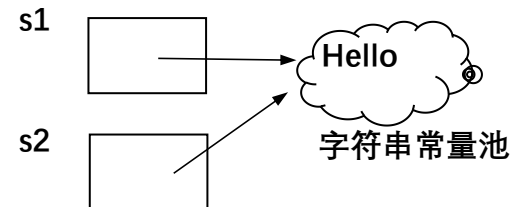
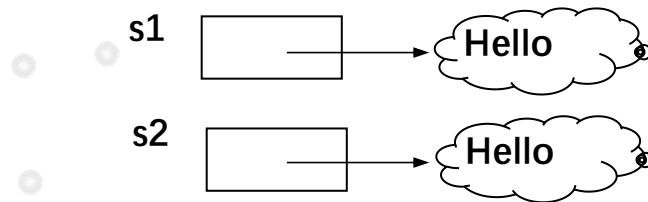
字符串比较

➤ 比较**内容**是否相等：

boolean equals(String str)

boolean equalsIgnoreCase(String str)

➤ 不能使用“==”号来比较字符串内容是否相等





字符串比较

➤ 比较大小：

int compareTo(String str)

小于，返回值大于0

等于，返回值等于0

大于，返回值小于0

```
String s1 = "ABC";  
String s2 = "ABE";  
System.out.println(  
    s1.compareTo(s2);
```

输出-2

➤ 判断前缀、后缀和包含

boolean startsWith(String prefix)

boolean endsWith(String suffix)

boolean contains(String str)



字符串的拆分和组合

- **String[] split(String regex)**
- **static String join(CharSequence delimiter, CharSequence... elements)**
- **boolean matches(String regex)**



String对象的不变性

➤ 一旦创建一个String对象，就不能对其内容进行改变。

```
String s = "Hello,world;
```



```
s.replace('o','A');
```



```
s = s.substring(0,6).concat("Java");
```

```
s.toUpperCase();
```



```
System.out.println(s);
```





命令行参数

➤main()方法参数String[] args称为命令行参数。

```
public static void main(String []args){}
```

```
public static void main(String ... args){}
```



课堂讨论及训练

- 使用下面的方法签名编写一个方法，统计一个字符串中包含字母的个数。

```
public static int countLetters(String s)
```

- 编写测试程序调用countLetters("Beijing 2022")方法并显示它的返回值7。





String对象是不变的

```
public class StringDemo {  
    public static void main(String[] args) {  
        String ss = new String("Initial: Text");  
        ss.replace('i', 'I');  
        ss.concat("123"); //  
        System.out.println(ss); //Initial Text  
        System.out.println(ss.replace('i', 'I')); //InItIal Text  
        String s2=ss.replace('i', 'I');  
        System.out.println(s2); //InItIal Text  
        System.out.println(ss.concat("123")); //Initial Text123  
    }  
}
```

运行结果:

Initial Text

In**I**t**I**al Text

In**I**t**I**al Text

Initial Text123



5.4.3 StringBuilder类和StringBuffer类

//可改变

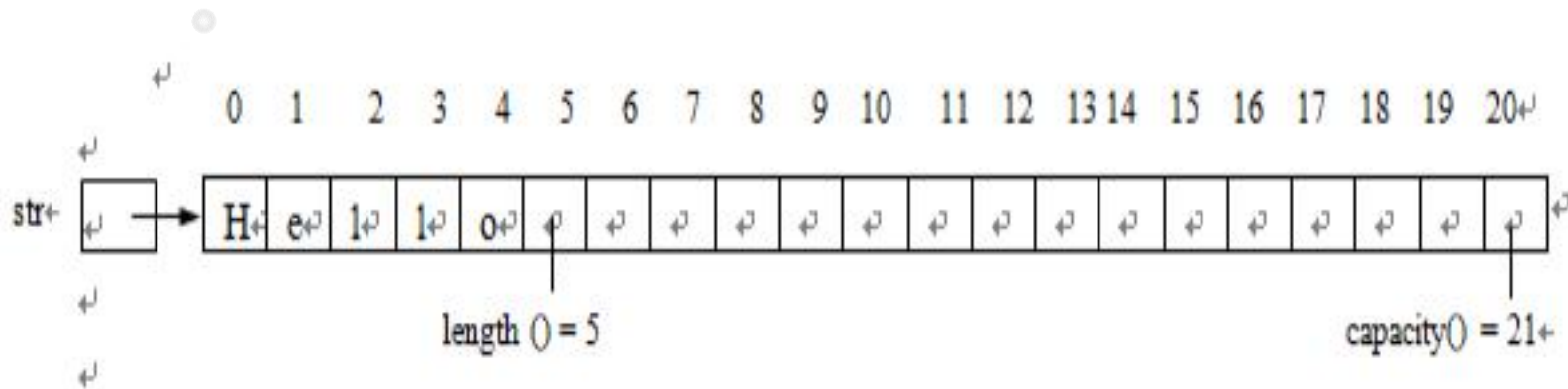
```
StringBuilder sb= new StringBuilder("Initial: Text");  
sb.insert(0, "12345");  
System.out.println(sb); // 12345Initial: Text  
sb.append("XYZ"); //  
System.out.println(sb); // 12345Initial: TextXYZ
```



创建StringBuilder对象

➤ `public StringBuilder(String str)`

`StringBuilder str = new StringBuilder("Hello");`





StringBuilder基本操

length()、charAt()、indexOf()、substring()

int capacity():

void setCharAt(int index, char ch)

StringBuilder append(int n)

StringBuilder append(String str)

StringBuilder insert(int offset, int n)

StringBuilder insert(int offset, String str)



StringBuilder基本操作

StringBuilder deleteCharAt(int index)

StringBuilder delete(int start, int end)

StringBuilder **replace**(int start, int end, String str)

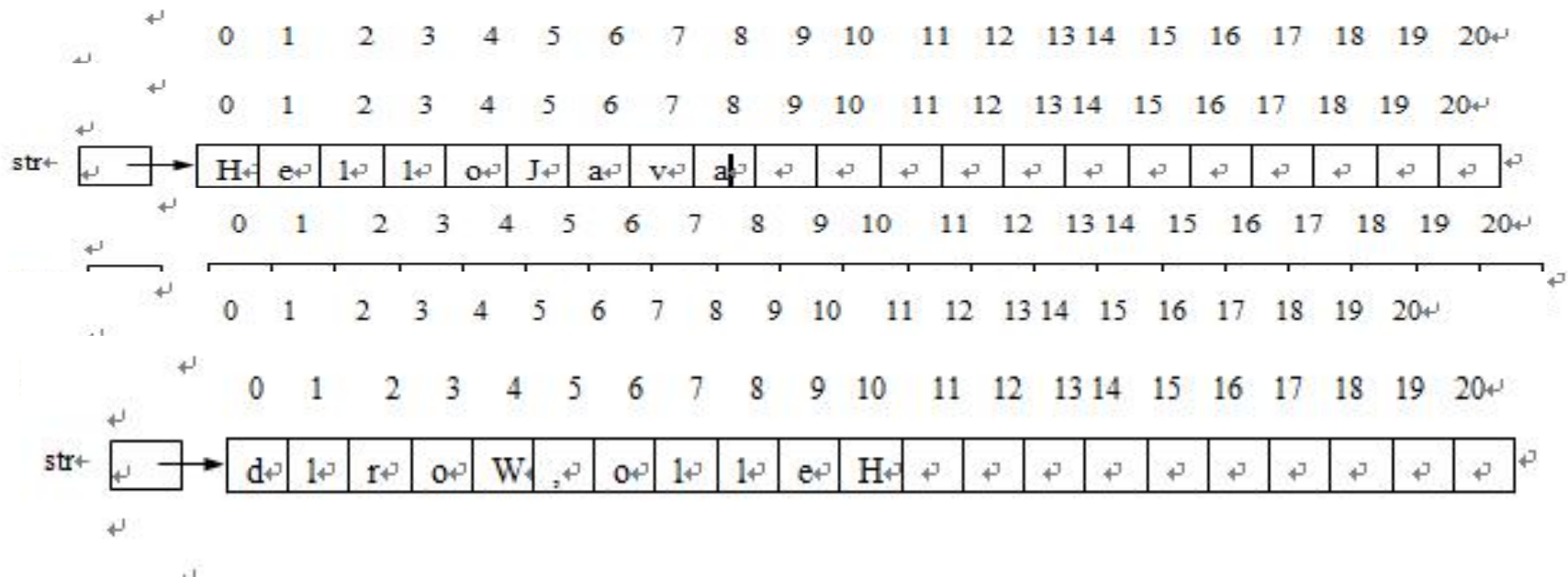
StringBuilder **reverse**()

void setLength(int newLength)



StringBuilder基本操作

```
StringBuilder str = new StringBuilder("Hello");  
System.out.println(str.length());  
System.out.println(str.capacity());  
str.append("Java");  
System.out.println(str);  
System.out.println(str.insert(5, ", "));  
System.out.println(str.replace(6, 10, "World"));  
System.out.println(str.reverse());
```





StringBuffer类

- StringBuffer类与StringBuilder类的主要区别是：
StringBuffer类的实例是**线程安全的**
StringBuilder类的实例不是线程安全的
- 如果不需要线程同步，建议使用StringBuilder类。

问题描述：

- 在Java API文档中查找LocalDate类，根据API文档说明写一个程序输出当前日期。



Java API文档

- Java**应用编程接口**（Application Program Interface，API）也称为库，包括为开发Java程序而预定义的和接口。
- 用Java编程时，需要经常从Java API文档中查看有关类库。



Java API文档

➤ 在线API文档:

<https://docs.oracle.com/javase/9/index.html>

➤ Java API文档下载地址:

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>





Java® Platform, Standard Edition & Java Development Kit Version 9 API Specification

This document is divided into three sections:

Java SE

The Java Platform, Standard Edition (Java SE) APIs define the core Java platform for general-purpose computing. These APIs are in modules whose names start with `java`.

JDK

The Java Development Kit (JDK) APIs are specific to the JDK and will not necessarily be available in all implementations of the Java SE Platform. These APIs are in modules whose names start with `jdk`.

JavaFX

The JavaFX APIs define a set of user-interface controls, graphics, media, and web packages for developing rich client applications. These APIs are in modules whose names start with `javafx`.

Java SE

Module	Description
<code>java.activation</code>	Defines the JavaBeans Activation Framework (JAF) API.
<code>java.base</code>	Defines the foundational APIs of the Java SE Platform.



课堂讨论及训练

➤ 1. 编写一段代码实现从控制台读取一个字符串和一个字符。

➤ 2. 执行下列语句后输出的结果是 () 。

```
String s = "\"Hello,World!\"";  
System.out.println(s.length());
```

A. 12

B. 14

C. 16

D. 18

➤ 3. 如果要求下列代码输出“Hello.java”，请给出程序代码 。

```
String s = "D:\\study\\Hello.java";
```

```
( _____ );
```

```
System.out.println(s);
```



课堂讨论及训练

➤ 3. 执行下列语句后输出的结果是（ ）。

```
String foo = "ABCDE";  
foo.substring(3);  
foo.concat("XYZ");  
System.out.println(foo);
```

A. ABCDE B. DEXYZ C. XYZ D. D

- `String foo = "ABCDE";`
 `foo=foo.substring(3); //DE`
- `foo=foo.concat("XYZ"); //DE+XYZ`
 `System.out.println(foo); //DEXYZ`

编程作业

5.1 使用下面的方法签名编写一个方法，统计一个字符串中包含字母的个数。

```
public static int countLetters(String s)
```

编写测试程序调用countLetters("Beijing 2008")方法并显示它的返回值7。

5.2 编写一个方法，将十进制数转换为二进制数的字符串，方法签名如下：

```
public static String toBinary(int value)
```



字符串类—StringBuffer

- String类对象是不可变的字符串
- StringBuffer类对象是可变的字符串，有改变字符串的若干方法。

构造方法

- StringBuffer() // 创建一个空的StringBuffer对象
- StringBuffer(int length) //设置初始容量
- StringBuffer(String s) //利用已有字符串String对象来初始化

方法

- String toString() 将可变串变为不可变字符串
System.out.println()不接受可变字符串



字符串类—StringBuffer

- 修改可变字符串(StringBuffer类的方法)
 - ✓ `append(char c)`
在字符串的最后追加一个字符
 - ✓ `insert(int index , substring)`
在字符串某位置插入子串
 - ✓ `void setCharAt(int index, char c)`
将字符串指定位置处的字符替换成指定字符
 - ✓ `replace(int start, int end, String str)`
 - ✓ `reverse()`



例

```
StringBuffer buf = new StringBuffer("Initial Text");  
int index = 1;  
buf.insert(index, "123");      // I123nitial Text  
buf.append("456");             // I123nitial Text456  
buf.delete(1, 4);              // Initial Text456  
buf.replace(12, 15, "XY");     // Initial Text XY  
// Convert to string  
String s = buf.toString();     // Initial Text XY
```

Java中常用的特殊字符

注意每个特殊字符的前面都有一个反斜线（\）。

特殊字符	显 示
\'	单引号
\"	双引号
\\	反斜线
\t	制表符
\b	回退符
\n	换行



String in Java or C++

```
//Java programming language code  
String s1 = "hello";  
String s2 = s1;  
s1 += " world";  
System.out.println(s1 + "\n" + s2);  
//s1 = "hello world" and s2 = "hello"
```

CS 2020-11-03 T
补P17 5.3 Arrays
类和数组应用实例

```
//C++ code  
string* s1 = new string("hello");  
string* s2 = s1;  
(*s1) += " world";  
cout<<*s1<<endl<<*s2<<endl;  
return 0;  
//s1 = s2 = "hello world"
```




5.4 字符串操作

- Java的字符串连接运算符 +
- Object 的方法 toString() 实现其他对象向字符串的转换
- main方法的参数

```
class a{  
    public static void main(String[] args) {  
        int x = Integer.parseInt(args[0]);  
        int y = Integer.parseInt(args[1]);  
        System.out.println("x+y=" + (x+y) );  
    } //利用命令行参数读入两个数，把它们相加后输出  
}
```

//字符串比较

```
public static void main(String[] args) {  
    String myName = "abc";  
    String yourName = "abc";  
    boolean ourName = myName.equals(yourName);  
    System.out.println(ourName);  
    System.out.println(myName == yourName);  
}
```

结果: **true**
true



例5.11 给出一段英文句子，将每一个单词分解出来放入数组元素中并排序输出。

```
1. public static void main(String[] args) {
2.     String str1 = "The String class represents character strings. All string literals in Java programs, such as \"abc\", are
   implemented as instances of this class.";
3.     String[] s = new String[50]; // 定义数组含50个元素
4.     str1 = str1.replace("\\", ' '); // 将字符串中的转义字符\" 替换为空格
5.     str1 = str1.replace(',', ' '); // 将字符串中的,号字符替换为空格
6.     str1 = str1.replace('.', ' '); // 将字符串中的.字符替换为空格
7.     System.out.println(str1); //输出处理后的字符串
8.     int i = 0, j;
9.     while ((j = str1.indexOf(" ")) > 0) // 查找空格,若找到,则空格前是一单词
10.    {
11.        s[i++] = str1.substring(0, j); // 将单词取出放入数组元素中
12.        str1 = str1.substring(j + 1); // 在字符串中去掉取出的单词部分
13.        str1 = str1.trim(); // 去掉字符串的前导空格
14.    }
15.    Arrays.sort(s, 0, i); // 在上边析取了i 个单词，对它们进行排序
16.    for (j = 0; j < i; j++) {
17.        System.out.print(s[j] + " "); // 输出各单词
18.        if ((j + 1) % 5 == 0)
19.            System.out.println();
20.    }
21.    System.out.println();
22. }
```



5.5 数组和字符串的综合实例

- 统计每个字母出现的次数。
- 统计不同数据个数(剔除重复数据)



5.5.1 StringTokenizer

- StringTokenizer parses a string into tokens
- The StringTokenizer class allows an application to break a string into tokens. The tokenization method is much simpler than the one used by the StreamTokenizer class.
- The StringTokenizer methods do not distinguish among identifiers, numbers, and quoted strings, nor do they recognize and skip comments.

java.util.StringTokenizer

```
1. import java.util.StringTokenizer;
2. public class StrToken {
3.     public static void main(String[] args) {
4.         int i=0;
5.         String str1 ="The tokenization method is much simpler ";
6.         StringTokenizer tokens = new StringTokenizer( str1," " );
7.         int tokenNumbs= tokens.countTokens();
8.         String [] words =new String[tokenNumbs];
9.         while ( tokens.hasMoreTokens() ){
10.             words[i]=tokens.nextToken();
11.             System.out.println(i+" "+words [i]);
12.             i++;
13.         }
14.     }
15. }
```



Wheel.java

//创建一个字符串数组，统计每个字母出现的次数。

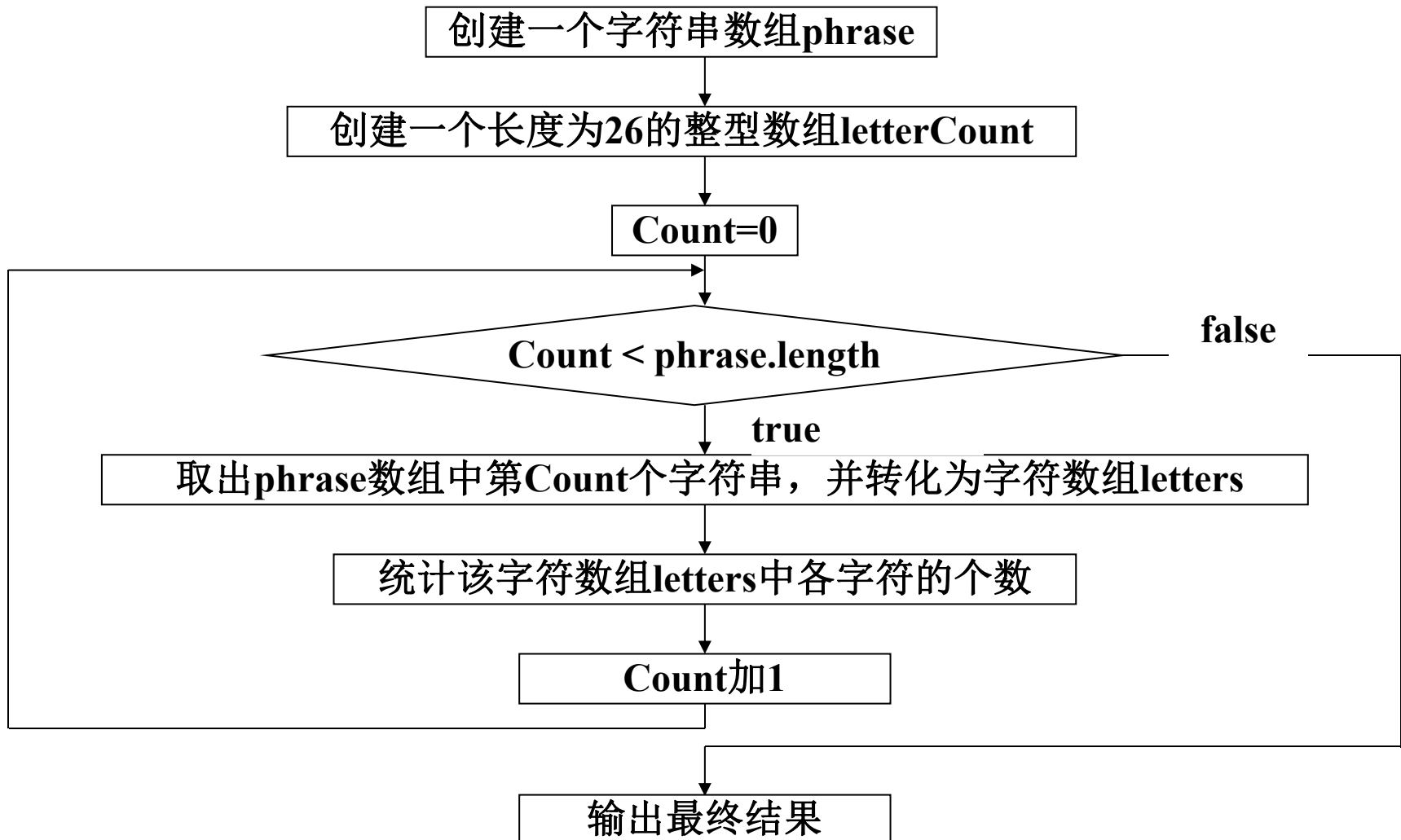
```
public class Wheel{
public static void main(String args[ ]){
String phrase[ ]={"A STITCH IN TIME SAVES NINE, DON'T EAT
                  WILL IT PLAY IN PEORIA, ....." };
int[ ] letterCount=new int[26];
for (int count=0;count<phrase.length;count++){
    String current=phrase[count];
    char[ ] letters=current.toCharArray();
    for (int count2=0;count2<letters.length;count2++){
        char lett=letters[count2];
        if((lett>='A')&(lett<='Z')){
            letterCount[lett-'A']++;
        }
    }
}
for (char count='A';count<='Z';count++)
    System.out.print(count+": "+letterCount[count-'A']+"\\t");
System.out.println();
}
```

程序的执行结果为:

A:22	B:3	C:5	D:13	
E:28	F:6	G:5	H:8	I:18
J:1	K:0	L:13	M:10	
N:19	O:27	P:3	Q:0	
R:13	S:15	T:19	U:4	V:7
W:9	X:0	Y:10	Z:0	

程序流程图

AI191班 2020-11-4 W



附: java中的正则表达式

```
String s = "@Shang Hai Hong Qiao Fei Ji Chang";
```

```
String regEx = "a|F"; //表示a或F
```

```
Pattern pat = Pattern.compile(regEx);
```

```
Matcher mat = pat.matcher(s);
```

```
boolean rs = mat.find();
```

```
public class Test{
    public static void main(String args[]) {
        String str="@Shang Hai Hong Qiao Fei Ji
Chang";
        boolean rs = false;
        for(int i=0;i<str.length();i++){
            char z=str.charAt(i);
            if('a' == z || 'F' == z) {
                rs = true;
                break;
            }else{
                rs= false;
            }
        }
        System.out.println(rs);
    }
}
```



练习五

1. 练习：分别用迭代和递归方法,编写一个使用数组实现折半查找程序(**BiSearch.java**);
2. 统计一段英文中每个字母出现的次数;
3. 单词统计WordCount.java
4. 去重（针对数字或字符串）
(RemoveRedundance.java)
5. 一元钱用5角，2角，1角，5分，2分，1分对换，计算有多少种方法？并写出具体组合方式。
6. 模拟扑克(52张)<洗牌>和<发牌>



```
package yexjjava;  
import java.util.ArrayList;  
import java.util.Collections;
```

```
public class PlayCard {
```

```
    @SuppressWarnings("unchecked")  
    public static void main(String[] args) {  
        ArrayList cards = new ArrayList();
```

```
        String[] hua = { "黑桃", "梅花", "红桃", "方块" };  
        String[] dian = { "A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K" };  
        for (int j = 0; j < 4; j++) {  
            for (int i = 0; i < 13; i++) { // 定义除大王小王的整副牌，加入cards数列中  
                cards.add(hua[j] + dian[i]);  
            }  
        }  
        System.out.println("扑克准备完毕，共计" + cards.size() + "张牌:");  
        System.out.println(cards);  
        // 洗牌  
        Collections.shuffle(cards); // 随即洗牌
```

```
        System.out.println("洗牌完毕，共计" + cards.size() + "张牌:");  
        System.out.println(cards);
```

```
        ArrayList player1 = new ArrayList(); // 设定四个玩家的对象序列  
        ArrayList player2 = new ArrayList();  
        ArrayList player3 = new ArrayList();  
        ArrayList player4 = new ArrayList();
```

```
        for (int i = 0; i < 13; i++) {  
            player1.add(cards.remove(0)); // 做13次循环进行发牌，玩家手里牌由发牌除去的牌决定  
            player2.add(cards.remove(0));  
            player3.add(cards.remove(0));  
            player4.add(cards.remove(0));  
        }
```

```
        Collections.sort(player1); // 把玩家手里的牌进行排序，在CardComparator类中按数元素索引第二位比较排序  
        Collections.sort(player2);
```



```
> public class Card {
>
>     /**
>      * @param args
>      */
>
>     private String face;
>     private String suit;
>
>
>     public Card( String f, String s ) {
>         face = f;
>         suit = s;
>     }
>
>     protected String getSuit(){
>         return suit;
>     }
>
>     protected String getFace(){
>         return face;
>     }
>
>     public String toString(){
>         return face + " of " + suit;
>     }
>
>
>     public static void main(String[] args) {
>         String faces[] = { "A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K" };
>         String suits[] = { "黑桃", "红桃", "梅花", "方块" };
>         // (2) 请编写模拟洗牌的程序段，即把数组deck中的扑克牌随机打乱存放顺序。
>         Card deck[] = new Card[52];
>
>         for ( int i = 0; i < deck.length; i++ ) {
>             deck[i] = new Card( faces[ i % 13 ], suits[ i / 13 ] ); // suits[ i % 4 ];
>         }
>
>         // Test
>         System.out.println("Before Shuffle ");
>         for (int i=0; i<deck.length; i++) 叶锡君主编《Java程序设计案例教程》中国农业出版社 yexj@njau.edu.cn
>         {
>             System.out.println( deck[i].toString() );
>         }
>     }
> }
```



/*文档词频统计WordCount串行处理方式(Java):

```
import java.util.Hashtable;
import java.util.HashSet;
import java.util.StringTokenizer;
import java.util.Iterator;

public class WordCount {

    public static void main(String[] args) {
        int i;
        String[] text    = new String[]
            { "hello", "hello world", "hello every one", "hello every in the world", "say    hello to everyone in the
world" };

        Hashtable    ht = new Hashtable();
        for(i=0; i< text.length;      ++i)
        {
            StringTokenizer st          = new StringTokenizer(text[i]);
            while (st.hasMoreTokens()) {

                String word    = st.nextToken();
                if(!ht.containsKey(word)) {
                    ht.put(word, new Integer(1));
                }
                else
                {
                    int wc =      ((Integer)ht.get(word)).intValue() +1; // 计数加1
                    ht.put(word, new      Integer(wc));
                }
            }
        }
    }
}
```