



## 第二章 Java语言基础

---

———结构化编程基础



# 涉及到课本中章节:

---

- **第2章 Java语言基础**



# 第2章 Java语言基础

---

- 2.1 标识符与关键字
- 2.2 数据类型
- 2.3 变量与常量
- 2.4 运算符与表达式
- 2.5 流程控制
- 2.6 数组和字符串



# 第2章 Java语言基础

---

- 2.1 标识符与关键字
- 2.2 变量与常量
- 2.3 数据类型
- 2.4 运算符与表达式
- **2.5 流程控制**
- 2.6 数组和字符串



# 流程分类

---

- 顺序 —— 仅有一条路一直走到底
- 选择 —— 多条路挑个一直走到底
- 循环 —— 重复走两点间的路(可不同)
- 转移 —— 直达起点或终点



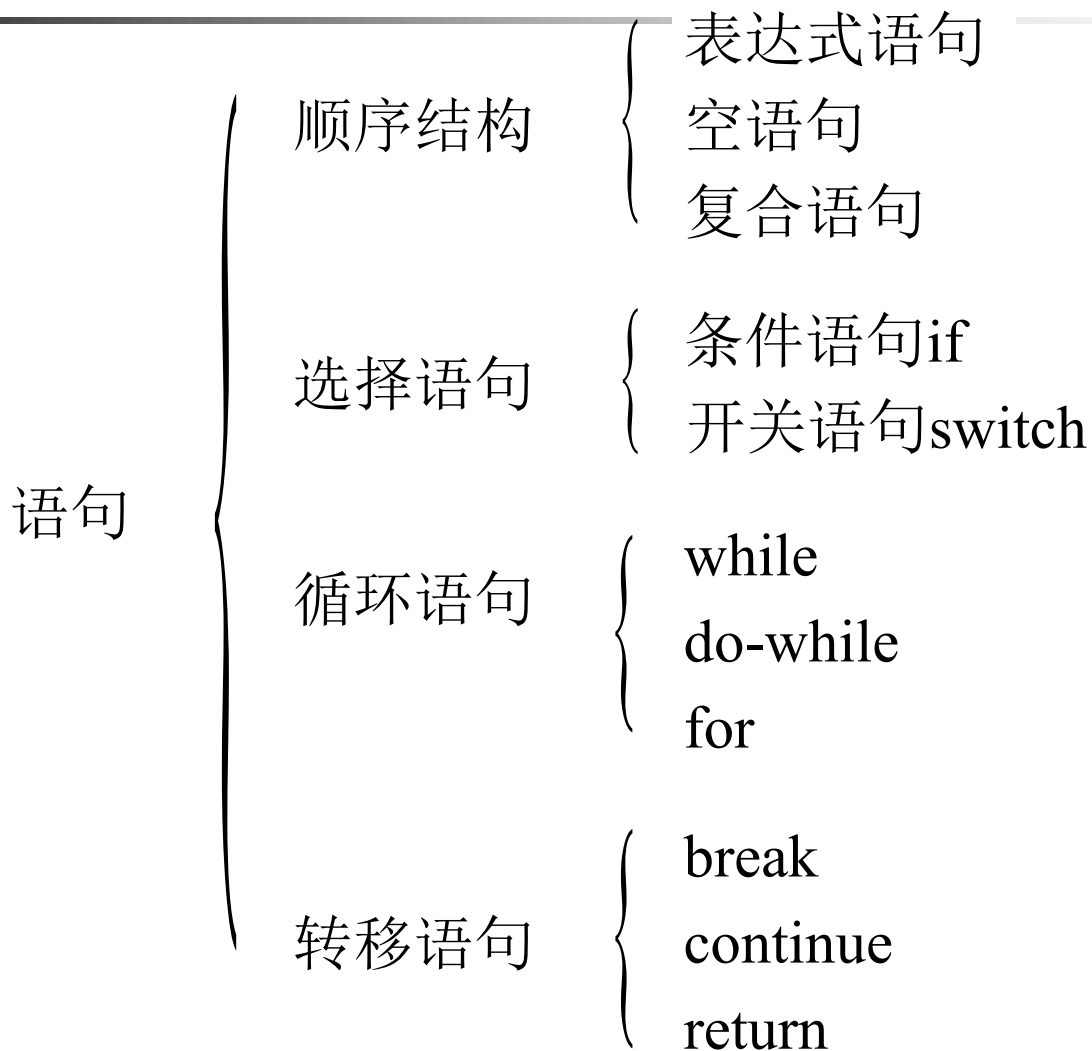
# 语句(Statement)

---

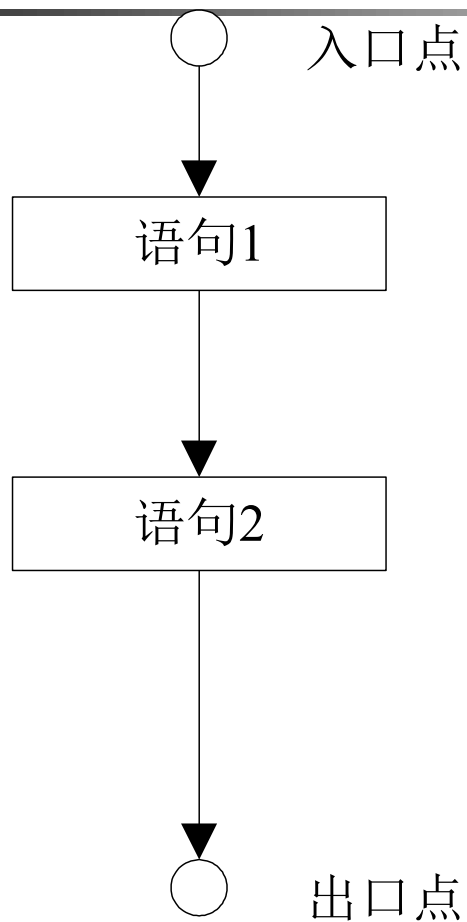
- 语句是描述对数据的操作的代码
- **Java** 中以`;`结束的为一个**Java**语句



# Java 语句分类



# 1) 顺序:



(a) 顺序结构



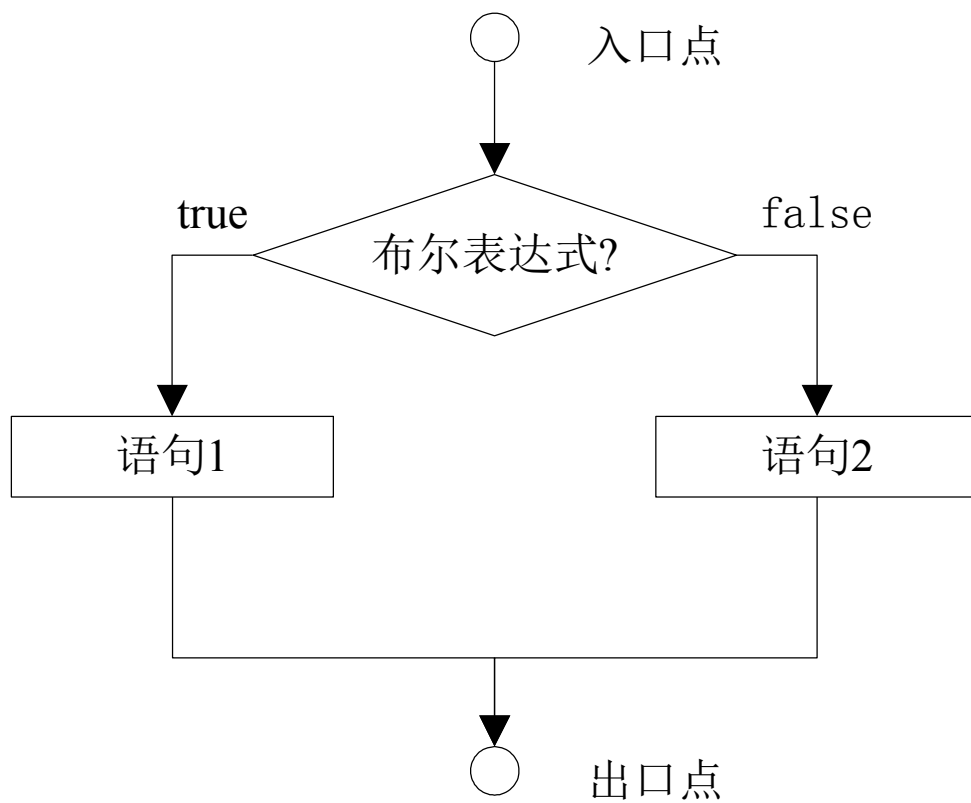


# 顺序结构语句

---

- 赋值语句                      变量定义，变量值变更
- 空语句                        不执行任何操作，为防；输重
- 复合语句(块语句)            { } 组成的语句序列

## 2) 分支:



(b) if-else二路分支结构



## 2).分支结构

## 之if-else语句

---

➤If语句(二路分支): 格式

if(布尔表达式)

<语句1>

//或者是{语句序列;}

[else <语句2> ]

//或者是{语句序列;}

**注:** if(布尔表达式)可以由多个条件经过逻辑运算而得到, 称为复合条件。复合条件的布尔运算符是&,|,!,^,&&,||.

**if 语句可以嵌套**



# 关于if嵌套

---

- `if(i%3==0 && i%5==0)`

可以写成嵌套if

```
if(i%3==0)
```

```
    if(i%5==0)
```

```
        //语句
```

[else ?与谁配对?]

注：&&运算的四种情况都需单独处理时只有嵌套才能满足。



# 常见错误形式

---

- 请查看代码错误,并改正之

```
if i=3 || 5>=j  
else (i<4)  
    { m=m+3;}
```

## 举例：（分支语句）

```
public class Example2_4 //if语句举例
{
    public static void main(String args[])
    {
        int math=65;
        int english=85;
        if(math>60)
        {
            System.out.println("数学及格了");
        }
        else
        {
            System.out.println("数学不及格");
        }
        if(english>90)
        {
            System.out.println("英语是优");
        }
        else
        {
            System.out.println("英语不是优");
        }
        if(math>60&&english>90)
        {
            System.out.println("英语是优,数学也及格了");
        }
        System.out.println("我在学习控制语句");
    }
}
```



## 多路分支示例——分支过多

---

- A (优)

- B (良)

- C (差)

- `if(s=='A'){...优...}`

- `else if(s=='B'){...良...}`

- `else{...差...}`

- A (优)

- B (良)

- C (中)

- D (及格)

- E (不及格)

- `if(s=='A'){...优...}`

- `else if(s=='B'){...良...}`

- `else if(s=='C'){...中...}`

- `else if(s=='D'){...及格...}`

- `else{...不及格...}`



## 2).多路分支

提高代码条理性和运行效率

➤ Switch语句(多路分支): 格式

**switch**(表达式)

{ **case**<常量1>:<语句1>;

**break**;

**case**<常量2>:<语句2>;

**break**;


.....

[**default**:<语句>;]

}

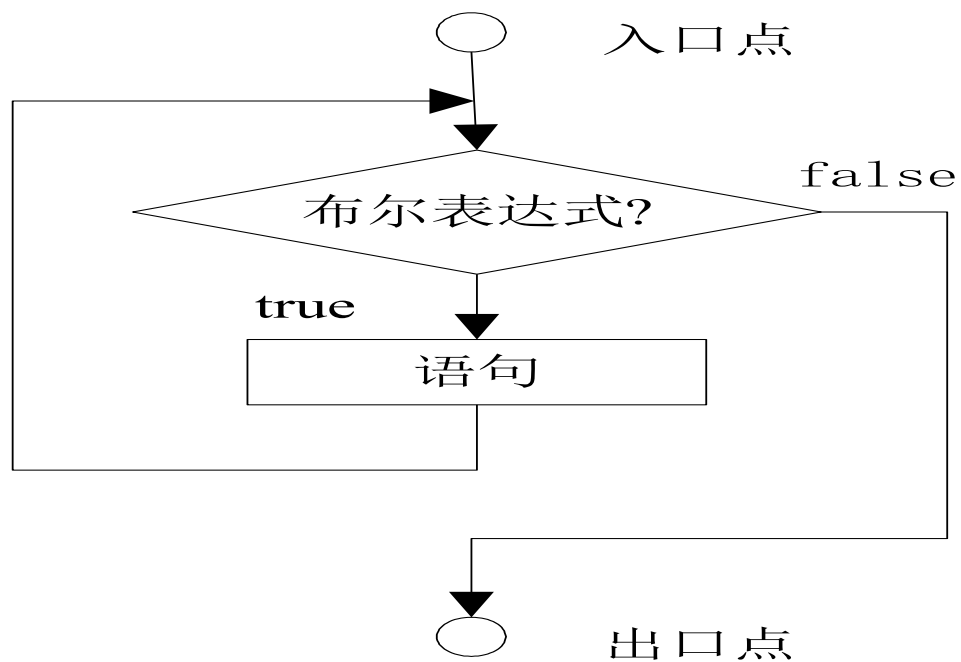
**注:** **switch**和**case**常量表达式中的类型可以是**byte**、**short**、**int**、**long**和**char**。而不能为**boolean**,且类型一致。





```
import java.applet.*;
import java.awt.*;
public class Example2_5 extends Applet //switch语句
{ public void paint(Graphics g)
    { int x=2;
      int y=1;
      switch(x+y)
      {case 1 :
        g.setColor(Color.red);g.drawString("i am 1",5,10);
        break;
        case 2:
        g.setColor(Color.blue); g.drawString("i am 2",5,10);
        break;
        case 3:
        g.setColor(Color.green); g.drawString("i am 3",5,10);
        break;
        default: g.drawString("没有般配的",5,10);
      }
    }
}
```

### 3) 循环



(c) while 循环结构

### 3).循环结构语句

#### ➤for语句：格式

**for**(<初始化语句>;<布尔表达式>;<更新语句>)

<语句>

//{语句序列}

#### ➤while语句：格式

**while**(<布尔表达式>)

<语句>;                      或 { 语句序列 }

--确认循环控制变量正确初始化，控制变量必须被正确更新以防止死循环。

#### ➤do ...while语句：格式

**do**    {<语句>

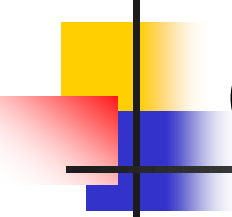
      //语句序列

**} while** (<布尔表达式>;

# 举例：（循环结构）

## (1)求10的阶乘(for 语句)

---



- **import java.applet.\*;**
- **import java.awt.\*;**
- **public class Example2\_5 extends Applet**
- **{ public void paint(Graphics g)**
- **{ long jiecheng=1;**
- **for(int i=10;i>=1;i--)**
- **{ jiecheng=jiecheng\*i;**
- **}**
- **g.drawString("10的阶乘是 "+jiecheng,10,20);**
- **}**
- **}**



# 常见错误形式

---

- 对如下源程序进行分析,找出错误原因

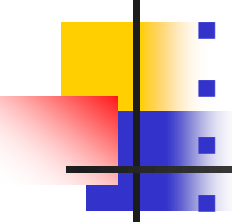
```
for(i=2;i<=100;i++);  
    i+=2;
```

考查内容: 变量作用域问题

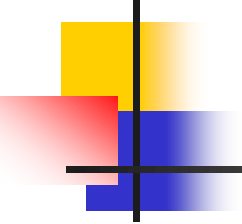
for语句结构和写法

for( ; ; )            //代表while(true),注意死循环

## (2) for语句



```
public class Example2_6
{ public static void main(String args[])
{ int sum=0;
  int I;
  int j;
  for( i=1;i<=10;i++)           //计算1+3+5+7+9。
  { if(i%2==0)
    continue;
    sum=sum+i;
  }
  System.out.println("sum="+sum);
  for( j=2;j<=50;j++)           //求50以内的素数
  { for( i=2;i<=j/2;i++)        //i<j/2的用意?
    {if(j%i==0)
      break;
    }
    //end of the inner for
    if(i>j/2)                    //i>j/2的用意?
    {System.out.println(""+j+"是素数");
    }
  }
  //end of the outer for
}
}
```



■ 质数是指在大于**1**的自然数中，除了**1**和它本身以外不再有其他因数的自然数。

■  **$i < j/2$ 的用意** 一个数不可能被它本身的一大半整除。

123456789 假设算9以内素数，第一层for循环到7

大于7的数不可能有7的因数，第二层for至少应该 $i < j$

$j/2 \rightarrow j$  的数  $7/2=3.5$ ,  $7/3=2.3\dots$ ,  $7/4=1.75$ ,  $7/5=1.4$

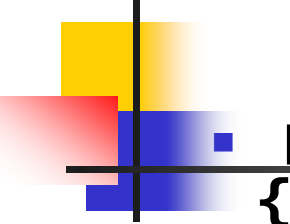
能整除的最小自然数是2， $j/2$ 之后商都小于2

■  **$i > j/2$ 的用意** 去掉if试试

排除第二层for循环，能整除的情况下**break**后的打印

# (3)求Fibonacci序列(while 语句)

第1项是0，第2项是1，第3项开始，每一项都等于前两项之和。



```
public class Example2_7
{
    public static void main(String args[ ])
    {
        final int MAX=20;
        int i=0;    //---F0
        int j=1;    //----F1
        int k=0;
        while(k<MAX)
        { System.out.println(“ “+i+” “+j);
          i=i+j;    //---F2=F0+F1
          j=i+j;    //---F3=F2+F1
          k=k+2;
        }
        System.out.println();
    }
}
```



# (4)用辗转相除法求两个整数的最大公因数

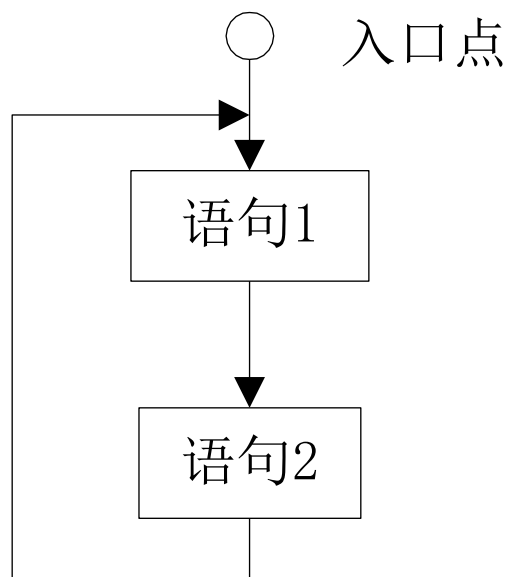
## (do.while语句)

以除数和余数反复做除法运算，当余数为 0 时，取当前算式除数为最大公约数

```
public class Example2_8
{
    public static void main(String args[ ])
    {
        int a=12;
        int b=18;
        int k=0;
        System.out.print("gcd("+a+", "+b+")=");
        do
        {
            k=a%b;
            a=b;
            b=k;
        }while(k!=0); //先做运算，判断在后
        System.out.println(a); //a是被除数还是除数?
    }
}
```

# 循环注意问题:

## ■ 死循环



○ 出口点

(2) 死循环



## 死循环举例:

---

- ```
int i=1,n=10,s=0;  
while(i<n)  
    s=s+i;
```

参与判断的条件在重复执行过程中未被改变



## 补充:

---

### ■ **For each**循环

```
for(元素类型t 元素变量x : 遍历对象obj){
```

```
    引用了x的java语句;
```

```
}
```

**局限：**无法改变某个元素值，只能一一遍历

**特点：**无需下标操作数据



## 4)转移语句

## -----特殊流程控制

---

- **break[label];**

- 用来从**switch**语句、循环语句中退出。一般与if一起使用

- **continue[label];**

- 用来结束**本次**循环。一般与if一起使用

- **return 返回值;**

- 用来使程序从方法中返回，并为方法返回一个值。

- **label: 语句;**

- 标识控制需要转换到的任何有效语句，它被用来标识循环构造的复合语句。
- **!!不提倡用标号!!**



## 举例

---

- **public static void breakAndContinue() {**
- **System.out.println("start");**
- **System.out.println("=====");**
- **for(int i=1;i<=8;i++) {**
- **if(i==2)System.out.println("不continue, 也不break");**
- **if(i==4)continue; //continue 以下不执行**
- **if(i==6)break; //break 退出for循环**
- **System.out.println(i);**
- **System.out.println("=====");**
- **}**
- **System.out.println("over");**
- **}**



# continue和break的区别

---

- **continue** —— start the next loop
- 只结束本次循环，重新开始一次循环，而不是终止整个循环的执行。
  
- **break** —— exit the loop all
- 结束整个循环过程，不再判断执行循环的条件是否成立，而是执行循环体后面的语句。

# 生活中的continue与break

外卖算法与人性大数据



continue——继续接单  
break——休息、下班

路径规划示意图。

该骑手身上有5个订单，其路径规划结果为：  
取（4个）→送（2个）→取（1个）→送（3个）。





# 第2章 Java语言基础

---

- 2.1 标识符与关键字
- 2.2 变量与常量
- 2.3 数据类型
- 2.4 运算符与表达式
- 2.5 流程控制
- **2.6 数组和字符串**

- 
- 
- **A** 数组
  - **B** 字符串



# 数组

---

- 具有相同数据类型的元素的有序集合
  - 数，是指数据类型，不是数字
  - 有序，通过序号可以定位和变更



# A: 数组

## ■ Java 中创建一个一维数组步骤:

### ➤ 1. 声明+new

声明一维数组变量:

**<类型>    <数组名>[ ];**

**<类型>[ ]    <数组名>;**

使用**new**为数组分配空间:

**<数组名>=new <类型>[长度]**

### ➤ 2. 声明同时赋初值

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



## 数组注意问题:

---

- 方括号在**变量**名之后表示该变量为数组。
- 方括号在**类型**之后表示**后面的变量全部**为数组，**推荐使用这种形式**，因为便于阅读和初始化。

例： `student[] mystudent, herstudent;`  
      `//说明都是数组`



## 引申问题:

---

- SCJP:SunCertificatedJavaProgrammer

```
int iArray1[10];  
int iArray2[];  
int iArray3[]=new int[10];  
int iArray4[10]=new int[10];  
int[] iArray5=new int[10];  
int iArray6[]=new int[];  
int iArray7[]=null;  
Select all right answers:
```



## 分析与答案:

---

- 数组声明时不需要指定数组长度
- 数组的长度只能在**new**时指定.

# 举例：数组

//求一组数中的最大和最小值

```
public class Example3_3{
    public static void main(String[] args)
    {
        final int SIZE=10; // 常量
        int table[ ]=new int[SIZE];
        int i,max,min;
        for(i=0;i<table.length;i++)
            table[i]=(int)(Math.random()*100); // 产生随机数
        System.out.print("table: ");
        for(int i=0;i<table.length;i++)
            System.out.print(" "+table[i]); // 输出一维数组
        System.out.println();
        max=table[0];
        min=table[0];
        for(i=1;i<table.length;i++)
        {
            if(table[i]>max)    max=table[i];
            if(table[i]<min)    min=table[i];
        }
        System.out.println("Max="+max);
        System.out.println("Min="+min);
    }
}
```





---

- A 数组

- B 字符串



# B 字符串

---

**Java**用**String**标记字符串

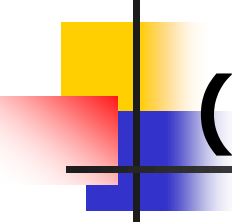
是**Java**的一个类



# B 字符串

---

- (1) 创建一个字符串
- (2) 字符串的常用方法
- (3) 与**StringBuffer**类的比较



## (1) 创建一个字符串

---

**-----String str="abc";**

**--字符串常量**

**-----String str=new String("this is a string");**

**----String str=null;**

**str="this is a string";**



## 注：String类有若干个不同的构造函数

---

- **String()**
- **String(byte[] bytes)**
- **String(byte[] bytes, int offset, int length)**
- **String(byte[] bytes, int offset, int length, String charsetName)**
- **String(byte[] bytes, String charsetName)**
- **String(char[] value)**
- **String(char[] value, int offset, int count)**
- **String(String original)**
- **String(StringBuffer buffer)**

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

思考：标红的两个函数，String类在调用的时候是如何区分的

:

## (2)String类的常用方法

| 方法                                                   | 说明                                                  |
|------------------------------------------------------|-----------------------------------------------------|
| <b>int length()</b>                                  | 返回字符串的长度                                            |
| <b>boolean equals(Object obj)</b>                    | 比较字符串是否相等                                           |
| <b>int compareTo(String str)</b>                     | 比较字符串，返回两者之间的差值                                     |
| <b>String concat(String str)</b>                     | 连接字符串                                               |
| <b>String substring(int beginIndex)</b>              | 返回字符串从 <b>beginIndex</b> 开始的字符串                     |
| <b>String substring(int beginIndex,int endIndex)</b> | 返回字符串从 <b>beginIndex</b> 开始到 <b>endIndex</b> 结束的字符串 |
| <b>char charAt(int index)</b>                        | 返回 <b>index</b> 指定位置的字符                             |
| <b>int indexOf (String str)</b>                      | 返回 <b>str</b> 在字符串中第一次出现的位置                         |
| <b>String replace(char oldc,char newc)</b>           | 以 <b>newc</b> 字符替换串中所有的字符                           |

# 举例：字符串

## 1. indexOf()举例---一个查找子串的应用程序

```
public class Example3_4
{
    public Example3_4( )
    { super();
    }
    public int doSearch(String str,String substr)
    { return str.indexOf(substr);
    }
    public static void main(String[] args)
    {if(args.length!=2)
        {System.out.println("Usage:java
IndexOfExample<string><SubString>");
        System.exit(0);
        }
    Example3_4 example=new Example3_4();
    String str=args[0];
    String substr=args[1];
```

```
        System.out.println("searching
string:"+str+"for:"+substr);
        int index=example.doSearch(str,substr);
        if(index!=-1)
        {System.out.println("Found the substring
position"+index);
        }
        else
        {
            System.out.println("Did not find the
substring");
        }
    }
}
```

Java Example3\_4 “an apple a day” day

## ■ 2.字符串比较

- ✓ 调用**equals()**;
- ✓ 调用**equalsIgnoreCase()**
- ✓ **compareTo()**

----一个比较两个字符串的应用程序

```
public class Example3_5{
    public Example3_5()
    {super();}

    public int compareStrings(String
str1,String str2)
{return str1. compareTo(str2);}

    public static void main(String[] args)
    { if(args.length!=2)
        { System.out.println("Usage:java
stringCompareExample<string1><string2>");}
        System.exit(0);
    }
}
```

```
Example3_5 example=new Example_5();
String str1=args[0];
String str2=args[1];
System.out.println("compare string:"+str1+"
with:"+str2+"");

Int result=example.compareStrings(str1,str2);
If(result<0)
{ System.out.println("String 1 is less than
string 2");}

Elseif(result>0)
{System.out.println("String1 is greater than
String 2");}

Else
{System.out.println("String 1 and String 2 are
equal");}
}

Java StringCompareExample" All good
things"" All good people"
```



### (3) 与StringBuffer/StringBuilder的比较

**StringBuffer/StringBuilder**类创建的**string**可以改变  
**String**类创建的**string**不能改变（**JVM**层面）



多次对**String**创建的**string**赋值的“改变”只是**string**赋值逻辑给程序员的一种**错觉**。



# 创建一个**StringBuffer**对象

---

**StringBuffer();**

----生成空的**StringBuffer**

**StringBuffer(int length);**

----生成一个给定长度的空**StringBuffer**

**StringBuffer(String str);**

----通过一个**String**对象或**String**文字生成一个**StringBuffer**

# ---StringBuffer的操作

## ➤ (1)在StringBuffer后附加内容

例: `StringBuffer buf=new StringBuffer();`  
`buf.append("this is");`  
`buf.append("another way");`  
`buf.append(5);`

或者: `buf.append("this is ").append("Stringbuffer Object");`

## ➤ (2)转换一个StringBuffer对象为字符串

例: `StringBuffer buf=new StringBuffer();`  
`buf.append("this is");`  
`buf.append("another way");`  
`return buf.toString();`

## ➤ (3)提取字符串(charAt() 和getChars(),subString())

例: `StringBuffer buf=new StringBuffer("String Buffer");`  
`char ch=buf.charAt(5);`

或者: `StringBuffer buf=new StringBuffer("String Buffer");`  
`char ch[]=new char[20];`  
`buf.getChars(7,10,ch,0);`

## ➤ (4)使用set系列的方法对StringBuffer对象进行操作

例: `buf.setLength();buf.ensureCapacity(512);`  
`buf.setCharAt(index,char);`



# 引申问题:参数传递

---

```
public class TempTest {  
    private void test1(int a){  
        a = 5;  
        System.out.println("test1方法中的a="+a);  
    }  
    public static void main(String[] args) {  
        TempTest t = new TempTest();  
        int a = 3;  
        t.test1(a);  
        System.out.println("main方法中的a="+a);  
    }  
}
```

值传递? 引用传递?

基本数据类型 / 引用数据类型