JAVA 课堂笔记

1 java 发展

Java1.0
Java1.1

Java1.2 分成三个块。J2se,j2me,j2ee 互联网方向

J2se 基础。

J2me 移动平台版本。Ios.

J2ee 企业级平台。网络。www.163.com.www.baidu.com

...

Java1.5 10 年 java5 变化很大。新特性。(javase,javame,javaee)

Java6,7,8

现在主流版本，java 5,6,7,

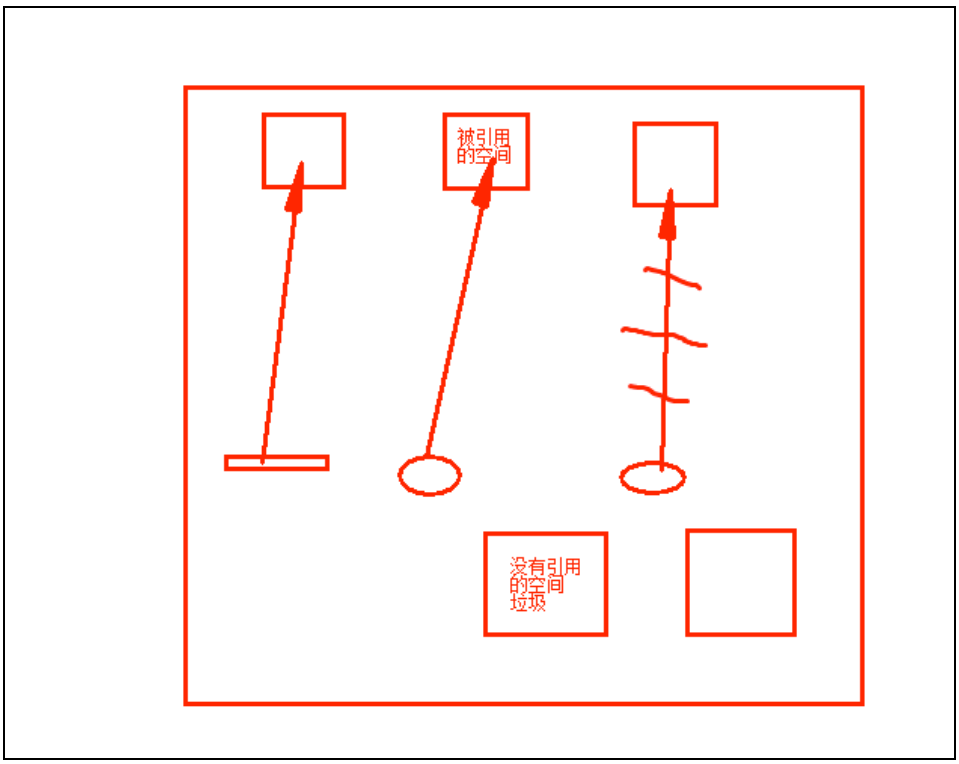
最新的 java8

2 垃圾回收

JAVA 程序运行过程中，有一个线程叫垃圾回收线程。程序运行了一时间之后很多垃圾信息。回收。

自动调用。程序员不能控制。1 内存不够。2 CPU 空闲。

垃圾：没有被引用的内存空间。



3 java 运行机制

计算机认识的只有什么？0，1

计算机认识的语言—机器语言。（人不认识）

人认识，---高级语言。C （if）

高级语言—转换-→机器语言。

因为转换的方式的不同所以，两种转换方式：

1 编译（不跨平台，编译一次之后就可以不用再编译了。）

编译器

将整个源文件（使用高级语言编写的文件.cpp）一次性全部转换成计算机可认识的指令（.exe）。

.exe 文件可不可以在非 windows 的操作系统下运行？不行。

一经编译完成可执行文件就一定要依赖与操作系统。

2 解释（跨平台，源文件必须带）

解释器。
文件的每一行一个一个的解释成机器指令。

HTML。

每次运行时都要带着源文件。
跨平台。

Java 先编译后解释型语言。

源文件(.java)----编译-→类文件(.class)（字节码文件）---解释执行（JVM：java 虚拟机）---》机器指令

4 JDK 与 JRE

JDK(Java Development Kit) Java 开发工具包

JDK = JRE + 工具（编译器、调试器、其他工具……） + 类库

JRE (Java Runtime Environment) Java 运行时环境

JRE = JVM(Java Virtual Machine) Java 虚拟机+解释器

Jdk8 安装 http://yunpan.cn/cVt4IBFkAiFBK （提取码：6309）

5 配置环境变量

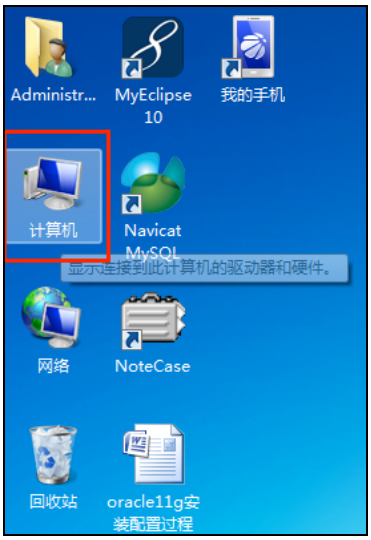
在 java 安装目录下有一个叫 bin 文件夹，这里面有两个很重要的命令。

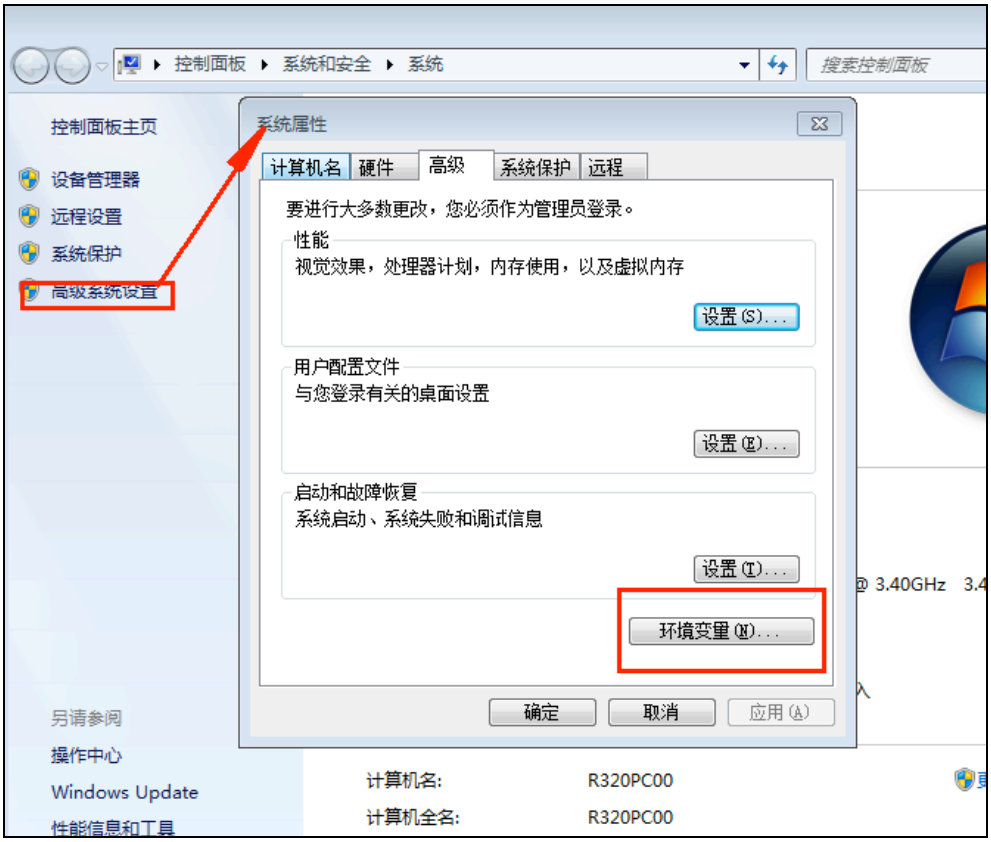
1 javac.exe 编译命令。

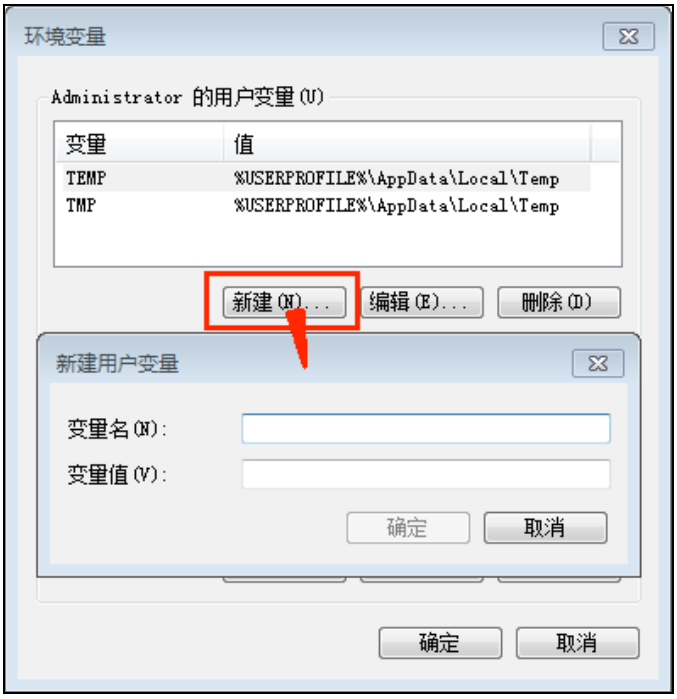
2 java.exe 解释执行命令。

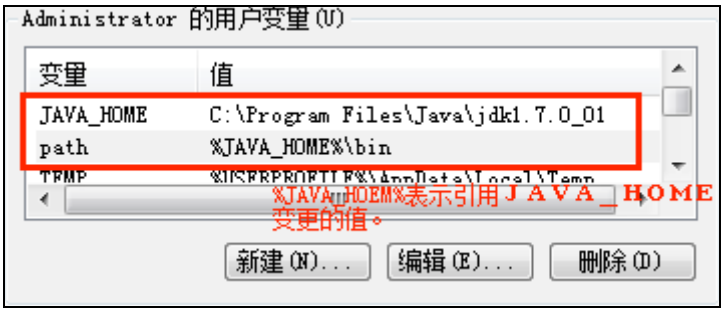
为这两个命令配置环境变量。目的：让这两个命令在 DOS 环境下的任意目录位置都可以执行。

Path=java 安装目录/bin　　-----指定到 javac 和 java 命令的目录。









6 第一个 Java 应用程序（手写）

6.1 创建一个文本文件。

小心扩展名。修改成.java 后缀。

6.2 编写 Java 源文件。

文件名：Hello.java
public class Hello{ public static void main(String[] args){

```
        System.out.println("我叫 XXXX！");
    }
}
```

Class 是类的关键字。

表示我们创建一个叫 Hello 的类。

public static void main(String[] args){这一行就是 main 方法的固定写法。

System.out.println("我叫 XXXX！");输出文本。

String[] 是 Java 中字符串数组。

注意：

1 文件名称必须与 public class 类名 一致。

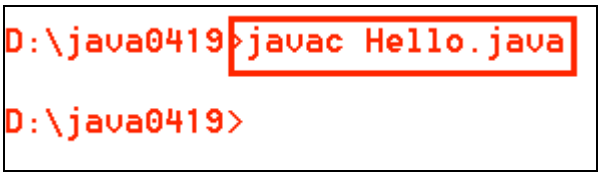
2 Java 是严格区分大小写。

3 System 与 system 是两个内容。

4 规范：类名都是首字母大写。

6.3 编译成生类文件（.class）

编译器命令：javac.exe 文件



编译成功之后 ， 在目录下会成生类文件。类文件格式：<类名.class>

注意：

1 一个源文件中可以创建多个类(class)。但只能有一个使用 public 修饰符，同时使用 public 修饰符的类名必须与文件名一致。

推荐：

1 最好一个源文件只写一个类。

6.4 解释执行

执行的命令：java.exe

格式： java 类名

7 注释

7.1 单行注释

```
//单行注释的文本内容
```

7.2 多行注释

```
/*
*多行
*/
```

7.3 文档注释 - javadoc 注释（给使用你编写的类的人看的）

```
/**
文本注释
*/
```

编写规范：

1 只能在类上使用。

@version
@author

当在源文件中类名的前面修饰@version 和@author，在执行 javadoc 命令时加-version 和-author 参数。

javadoc -d api -version -author Hello.java

2 只能在方法上使用。

@param 参数：描述方法的参数。

@return 参数：描述方法的返回值。

JavaDoc 标记	描　述
@version	指定版本信息
@since	指定最早出现在哪个版本
@author	指定作者
@see	生成参考其他 JavaDoc 文档的链接
@link	生成参考其他 JavaDoc 文档的链接，它和@see 标记的区别在于，@link 标记能够嵌入到注释语句中，为注释语句中的特定词汇生成链接
@deprecated	用来标明被注释的类、变量或方法已经不提倡使用，在将来的版本中有可能被废弃
@param	描述方法的参数
@return	描述方法的返回值
@throws	描述方法抛出的异常，指明抛出异常的条件

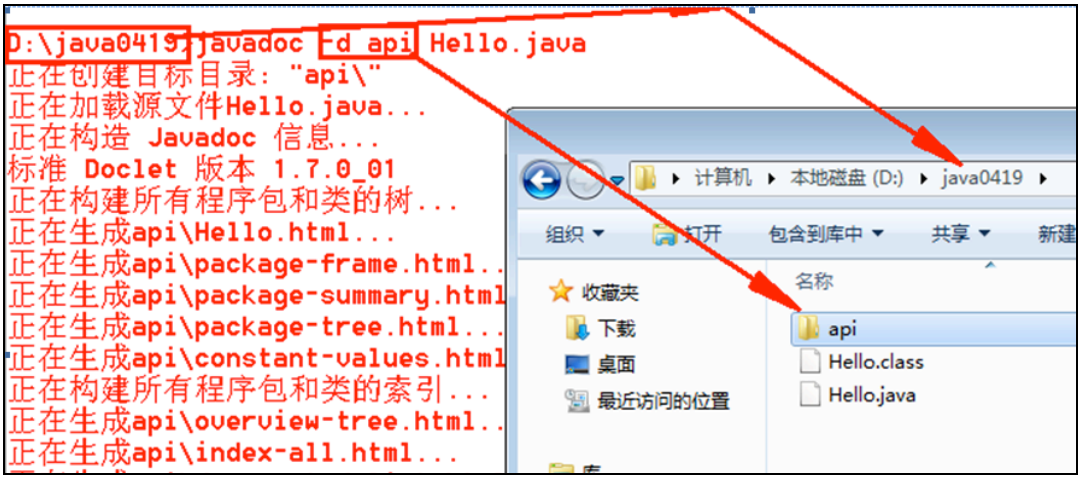
7.3.1 通过 javadoc 的命令提取源文件中的文档注释内容

基本格式：javadoc -d . 源文件名称

-d 表示输出的目录位置。

-d . 表示输出到当前的路径位置。

-d api 表示输出到当前位置的 api 目录中



8 包

Java 类中就是一个逻辑的概念。

在操作系统中是以物理文件夹的方式存在的。

作用：组织类的。（理解 windows 操作系统的文件夹）

8.1 使用包时要使用两个关键字。

8.1.1 package 打包。将类打到包里去。

语法格式 ： package 包名；

注意：

1 打包的命名每个源文件只有一个，并只能出现在第一行。

2 带包类的编译，使用-d 参数指定包的输出路径

Javac -d . 带包类的源文件名称

3 子包：

package 包 1.包 2.包 3;

8.1.2 当一个类使用了 package 语句时：

1 类名的变化：全限命名

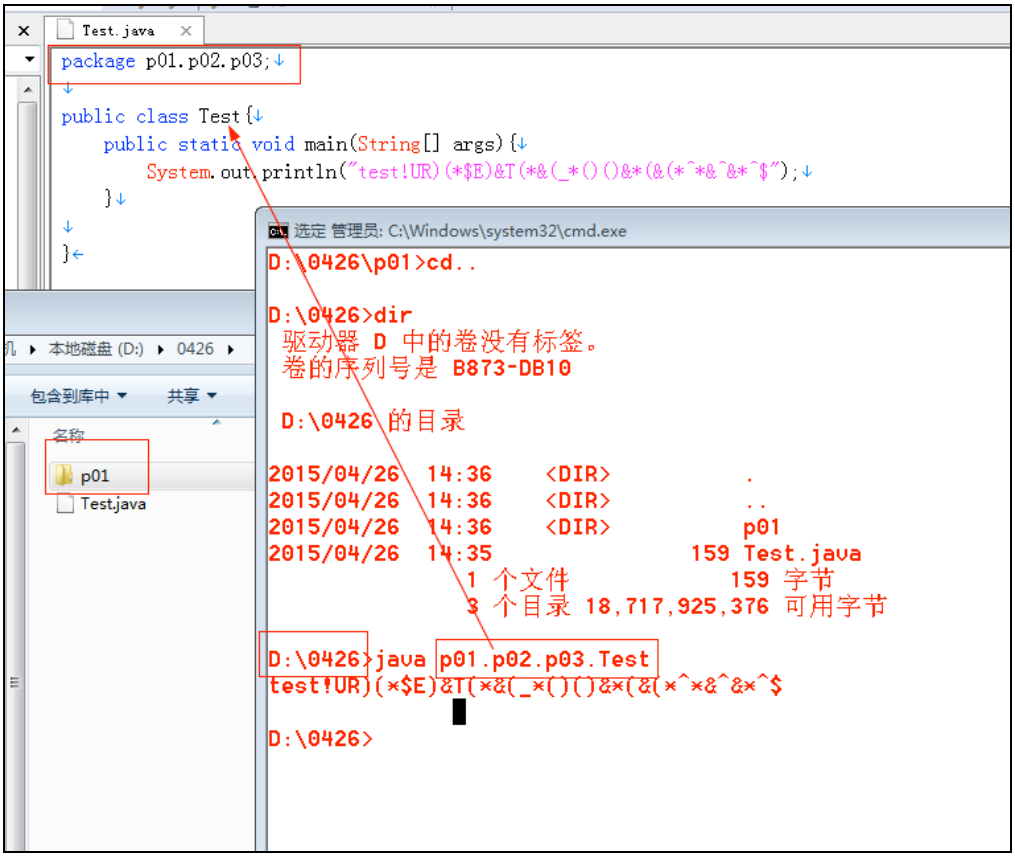
```
package p01.p02;
public class Test{
}
```

当前类的名称叫：p01.p02.Test

8.1.3 执行一个带包的类

java 类的全称（包名.类名）

在 dos 环境下执行时不要进入包的目录。



8.1.4 执行一个要调用其他包中类的类

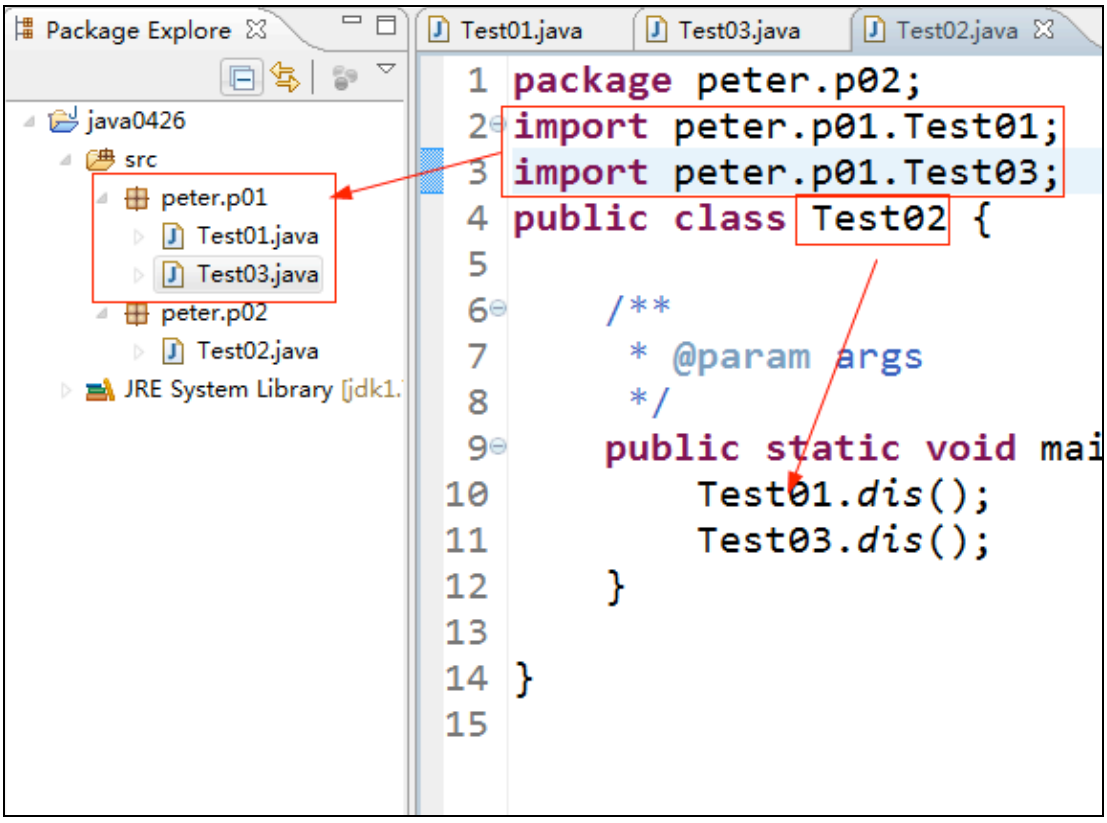
执行 Test01.class 文件时，在 p01.Test01 这个类中使用了 p02.Test02 这个类。所以在执行 Test01.class 文件时一定要能找到 p02.Test02 这个类才能正确的执行 p01.Test01 这个类。

Java 命令中有一个 -classpath 的参数（-cp）。用来指明执行一个类时所有需要用到的 class 文件的位置。

```
java -cp .;d:\  p01.Test01
```

8.1.5 import 关键字。导入包。

格式：import 类的全称；



注意与 package 语句的前后顺序。

提示：懒。(不推荐使用)

import peter.p01.*; 表示导入 peter.p01 下面的所有类。

9 标识符

有意义。

变量-首字母小写，后面每个单词首字母大写。String stuName;（驼峰）

类-首字母大写，后面每个单词首字母大写。

常量-全大写。

方法-首字母小写，后面每个单词首字母大写。

10 变量

程序运行过程中可以改变的量。
变量实际就是一块内存空间。
变量就是一个容器。就是一个存放数据的容器。

int a = 10;

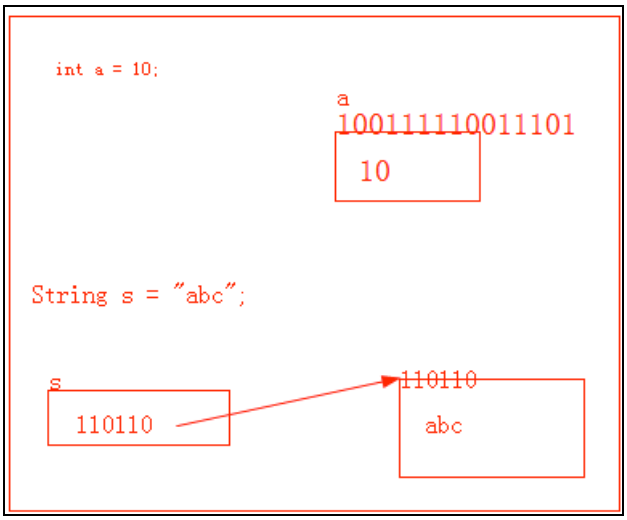
10 就是 字面量。

字面量就是一个变量可以取值的一个范围。
字面量也是有类型。

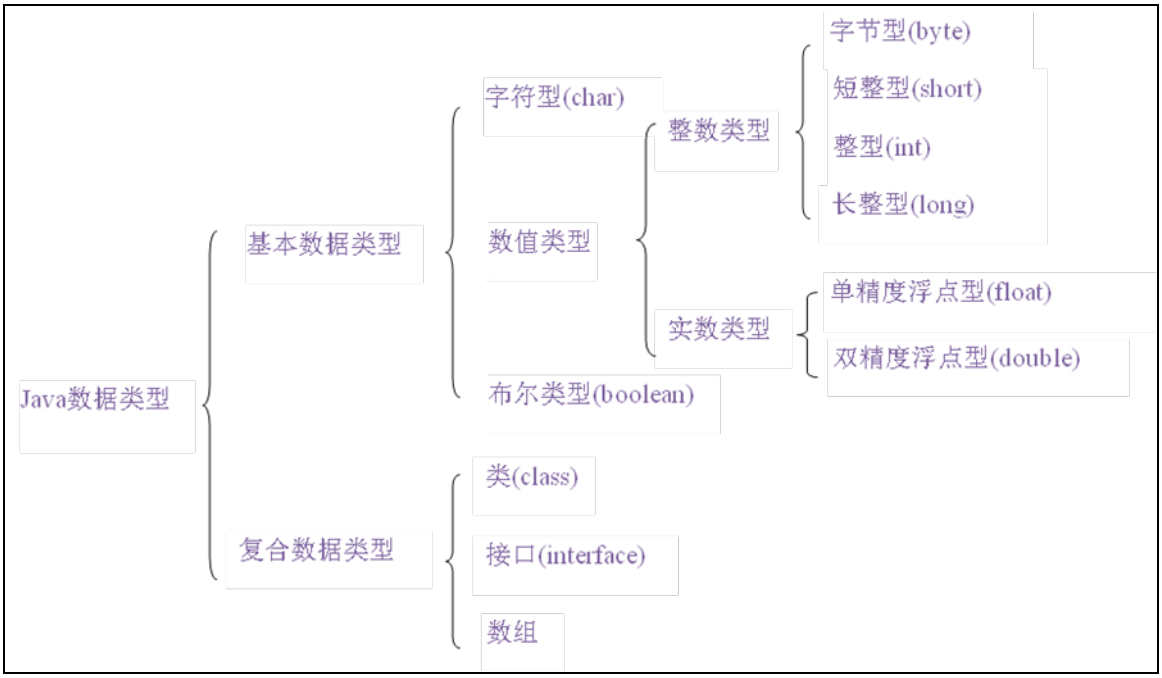
12.3 是什么类型的数值？double

10.1 变量的数据类型

两大分类：原始数据类型（基本数据类型）和引用数据类型（对象数据类型）



其中我们说 a 是原始数据类型，s 是引用数据类型。



原始数据类型就只有 8 个：byte,short,int,long　float,double　char　boolean (false/true)

long(L 后缀 123L)

float 和 double ： 如果一个小数没有特别说明，小数的类型是 double.

后缀 f 指明小数是 float 类型。

支持 8 进制和 16 进制表示方式

8： 用 0 前缀　　012

16：用 0x 前缀　　0x12

char　使用'单引号。支持汉字。支持 unicode 编码'\u 四位 16 进制'

boolean　的字面量 只有 true 和 false。

10.2 变量的定义

格式： 变量类型 变量名称

10.3 变量的赋值

使用"="

10.4 变量的作用域和生命周期

作用域：起作用的范围 ， 在哪能用。

- 在定义这个变量的{ }内起作用。

生命周期：什么时候创建，什么时候销毁。

11 数据类型的转换

两种：自动类型转换，强制类型转换。

11.1 自动类型转换

当小的数据类型向大的数据类型变化时，自动类型转换。

byte->short->int->long

float->double

int -> float

注意：表达式中的自动类型转换。

1 经过计算的数据会自动向 int 类型转换。

2 计算数据中有比 int 类型大的数据类型时自动向大的数据类型转换。

3 同类型之间的计算结果一定是相同的类型（比 int 大。）。

11.2 强制类型转换

当我们将一个大的数值类型存放到一个小的数据类型中时，系统默认是不允许。
所以我们就必须使用强制类型转换。
强转格式： 在类型前面加（），里面写要转换的类型。

```
float f = 12.3f;
int a = (int)f;
```

12 局部变量

在方法中定义的变量叫局部变量。
局部变量的使用要求：先声明，再赋值，最后使用

```
int i ;
i = 10;
System.out.println(i);
```

13 表达式+运算符

使用运算符连接起来的数值。

13.1 算数运算符

+ - * / %(mod) ++ --

除法小心：数值类型。

如果两边的运算数是同一个数据类型，结果也必然是同一数据类型。(byte/byte 之后的自动类型转换)

13.2 比较运算符

> < ==(恒等) >= <= !=

使用比较运算符组成的表达叫关系表达式，结果一定是一个boolean。true,false

13.3 位移运算符

>> 右移

<< 左移

>>> 带符号右移

将数值先转换成二进制，之后按二进制进行位移操作。

```
public static void main(String[] a
    int i = 23; //10111
    int j = i << 2;//1011100
    System.out.println(j);
```

```
/**
 * @param args
 */
public static void main(String[] args) {
    int i = 23; //10111
    int j = i << 2;//1011100
    System.out.println(j);

    int i1 = -8;//10000000 00000000 00000000 00001000
    //int j1 = i1 >>> 1;//010000000 00000000 00000000 0000100
    int j1 = i1 >> 1;//100000000 00000000 00000000 0000100
    System.out.println(j1);

    int k = 2 >>33;
    System.out.println(k);
```

```
92
-4
1
```

```java
int i = 20;//1.....10100
//向左移2位，，，1010000
System.out.println(i << 2);
//向右移2位，，，101
System.out.println(i >> 2);
//带符号右移    001.......101
System.out.println(i >>> 2);

//可爱
System.out.println(i >> 33 );
```

1 右移33会被处理成33%32=1 右移1位

13.4 赋值运算符

= 赋值。

+= 先加后赋值 .累加运算符

int a = 10;
int b = 10;

b += a;//在 b 数值的基础上累加上 a 的值。

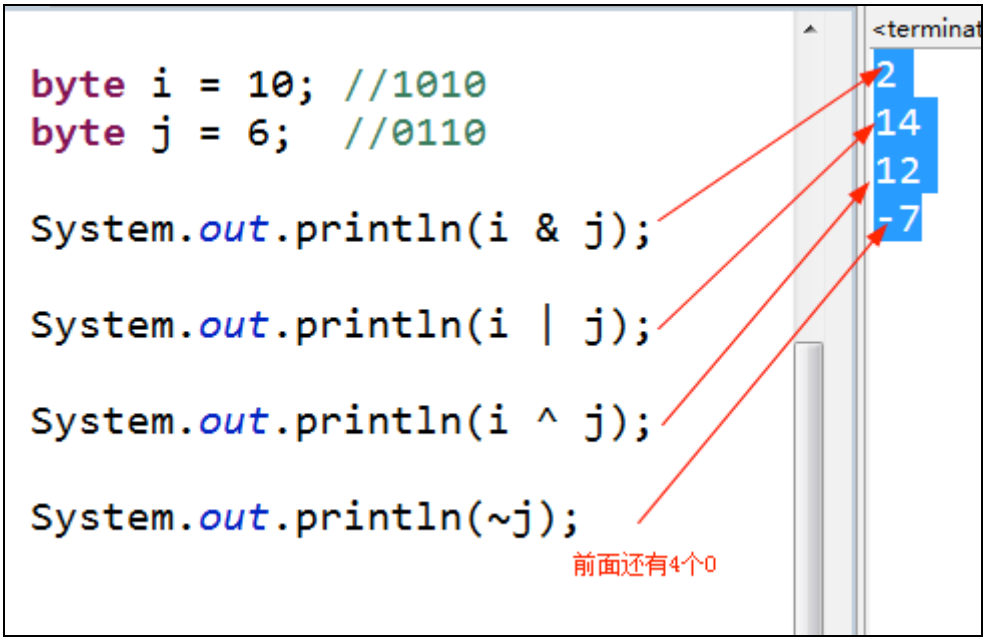
```java
int i = 1;
// 在i变量的基础上累加10;i = i + 10;
i += 10;
```

13.4 位运算符

&与，|或，^异或，~取反(一元运算符)

将数值先转换成二进制，之后每一位上的二进制进行位运算。

& 只要有 0 结果就为 0，两个必须都为 1 结果为 1.



13.5 逻辑运算符

& 逻辑与　当两边是 boolean 类型的数据时，是逻辑运算符。

| 逻辑或

&& 短路与 这个是我们后期开发过程经常使用的

|| 短路或

! 取反

短路：逻辑运算符是用来连接两个 boolean 表达式的，如果我们做“与运算”时，发现第一个 boolean 表达式的值为 false 时，这个整个表达式的值就一定为 false 了。

如果是使用“逻辑与&”，第二个表达式也要运行。

如果是使用“短路号&&”，则第二个表达式就不会运行了。

```java
String name = null ;

if(name != null && name.equals("peter")){
    System.out.println("哈哈");
}

System.out.println("end");
```

13.6 运算符的优先级

优先级什么都没有。

只有一个完美解决----()

14 String 字符串数据类型

String 实际上它是一个类。

String 只不过在使用的用法上感觉很像原始数据类型。

创建一个 String 类的实例时的语法与创建一个原始数据类型的变量很像。

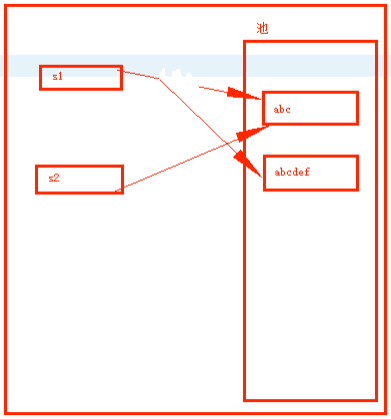
int i = 10;
String s = "peter";

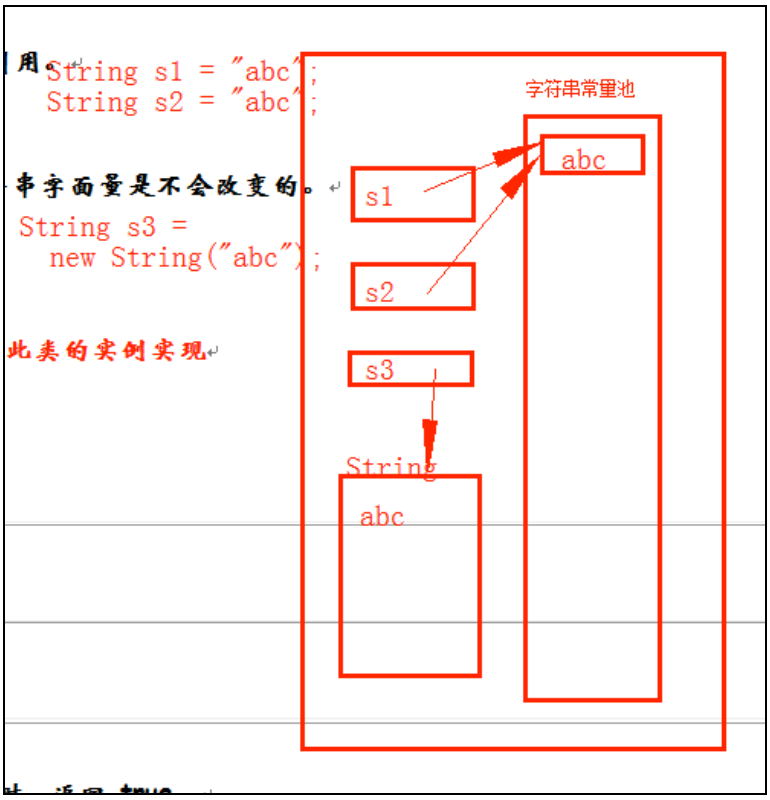
引用数据类型都有一个叫"null"的值,null 表示没有引用。

类中是有很多方法的。

在 String 类中一个特性：字符串不变性：表示字符串字面量是不会改变的。

```
ublic class StringTest {
   public static void main(String[] args) {
        String s1 = "abc";
        String s2 = "abc";
        String s3 = new String("abc");
        System.out.println(s1==s2);
/       System.out.println(s1.equals(s2));
/       System.out.println("--------------");
/       System.out.println(s1==s3);
/       System.out.println(s1.equals(s3));

        s1 = s1 + "def";

        System.out.println(s1);
        System.out.println(s2);

        System.out.println(s1==s2);

   }
```





String 类代表字符串。

Java 程序中的所有字符串字面值（如 "abc" ）都作为此类的实例实现

14.1 String 类常用的方法

char	charAt(int index) 返回指定索引处的 char 值。
String	concat(String str) 将指定字符串连接到此字符串的结尾。
boolean	contains(String s) 当且仅当此字符串包含指定的 char 值序列时，返回 true。 是否包含

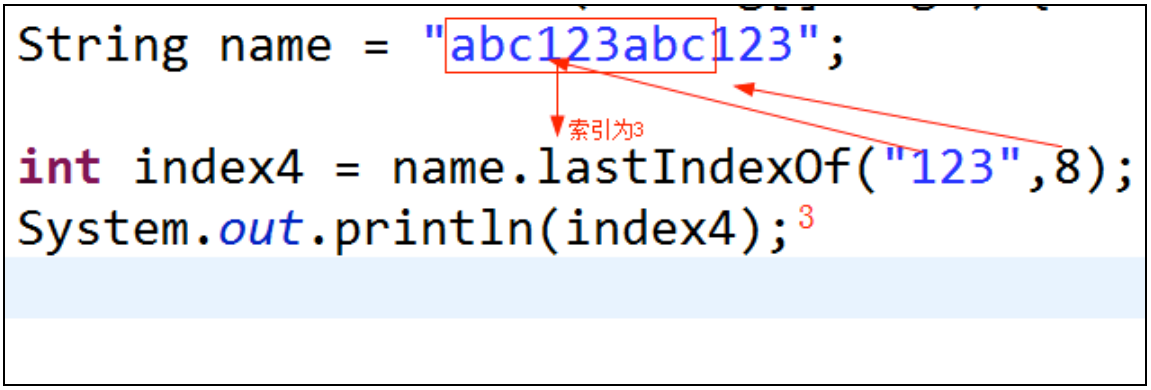
```
String name = "abcdefg";

char c = name.charAt(1);
//索引从0开始，第一个字母的索引为0;
System.out.println(c); b
//字符串拼接
String s = name.concat("123456");
System.out.println(s); abcdefg123456


boolean bool = name.contains("bcda");
System.out.println(bool); false
```

```
String name = "abc123456" ;
char c = name.charAt(2);
System.out.println(c); c

String s1 = name.concat("!@#$%^");
System.out.println(s1); abc123456!@#$%^
//在java中“+”号，也能起到字符串拼接的作用
name = name + "987";
System.out.println(name); abc123456987

boolean bool = name.contains("1223");
System.out.println(bool); false，因为1223 在name值中没找到
```

int	indexOf(String str) 返回指定子字符串在此字符串中第一次出现处的索引。
int	indexOf(String str, int fromIndex) 返回指定子字符串在此字符串中第一次出现处的索引，从指定的索引开始。
int	lastIndexOf(String str)

	返回指定子字符串在此字符串中最右边出现处的索引。
int	lastIndexOf(String str, int fromIndex) 返回指定子字符串在此字符串中最后一次出现处的索引，从指定的索引开始反向搜索。

```
public static void main(String[] args) {
    String name = "abc123abc123";
    //查找"123"在 name变量是第一次出现的位置。
    int index = name.indexOf("123");
    System.out.println(index);3
    //如果在name中没有找到则返回-1
    int index2 = name.indexOf("124");
    System.out.println(index2);-1 没找到
    //返回从第6个位置开始找到的第一个“123”的索引
    int index3 = name.indexOf("123",5);
    System.out.println(index3);9，从索引5开始找
```



int	length() 返回此字符串的长度。

14.2 字符串比较

我们在使用 String 类型时如要比较两个字符串的字面量是否相等。

boolean	equals(Object anObject) 比较两个字符串对象的字面量是否相等。比较两个字符串的值是否相等。

```
String name1 = "abcdefg";
String name2 = new String("abcdefg");
//两个字符串的值是否相等。
boolean bool = name1.equals(name2);
```

注意：推荐的格式。当一个字符串变量与一个字符串常量（“abc”）进行比较时，我们要求大家把常量写前面。

```
//两个字符串的值是否相等。
boolean bool1 = name.equals("abc");
boolean bool2 = "abc".equals(name);
System.out.println(bool1);
System.out.println(bool2);
```

boolean	equalsIgnoreCase(String anotherString) 忽略大小的比较方法。

```
String s1 = "abc";
String s2 = "aBC";

System.out.println(s1.equals(s2)); false
System.out.println(s1.equalsIgnoreCase(s2));
                   true,国为这是忽略大小写的比较
```

14.3 去空格

String	trim() 返回字符串的副本，忽略前导空白和尾部空白。

只把字符串前面和后面的空格去掉。中间的空格没影响。

14.4 取子字符串

String	substring(int beginIndex) 返回一个新的字符串，它是此字符串的一个子字符串。
String	substring(int beginIndex, int endIndex) 返回一个新字符串，它是此字符串的一个子字符串。

```
String s1 = "123456789";
System.out.println(s1.substring(1));23456789.
System.out.println(s1.substring(1,3));23
```

从索引1开始，到索引3结束，但不包含索引3的字母

```
String s1 = "0123456789";

String s2 = s1.substring(5);
System.out.println(s2);56789
String s3 = s1.substring(2,5);
System.out.println(s3);234 ，从第2个开始，到第5个停
```

但不包含第5个

14.5 字符串替换

String	replace(CharSequence target, CharSequence replacement) 使用指定的字面值替换序列替换此字符串所有匹配字面值目标序列的子字符串。

使用 replacement 这个字符串的值，替换 target 这个字符串的值。

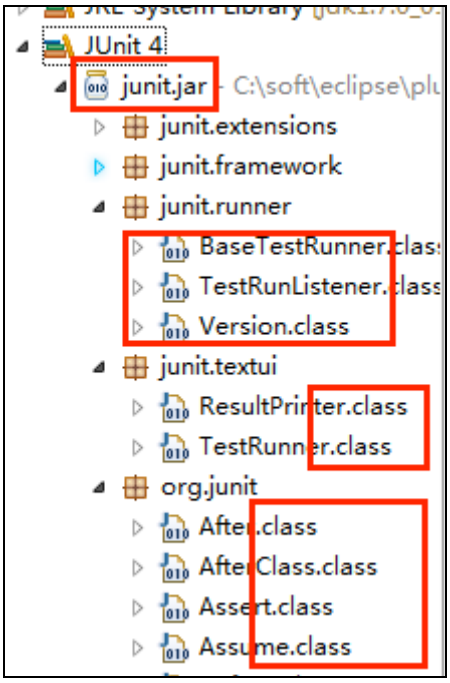
```
public static void main(String[] args) {
    String s1 = "sun1xu@163.com";
    System.out.println(s1);
    System.out.println(s1.replaceAll("163", "126"));
}
```

sun1xu@126.com 实现将163规制成126

15 jar 文件

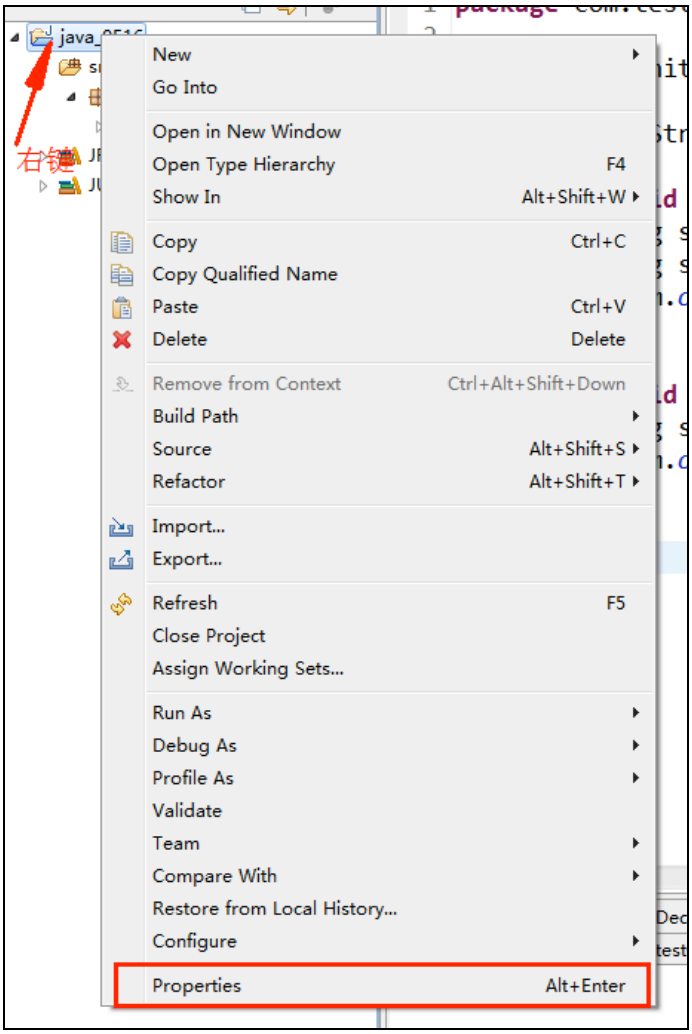
压缩包，是一堆 class 文件的压缩包。里面是一堆类文件，一般是其他人写好的类，我们开发中直接使用。

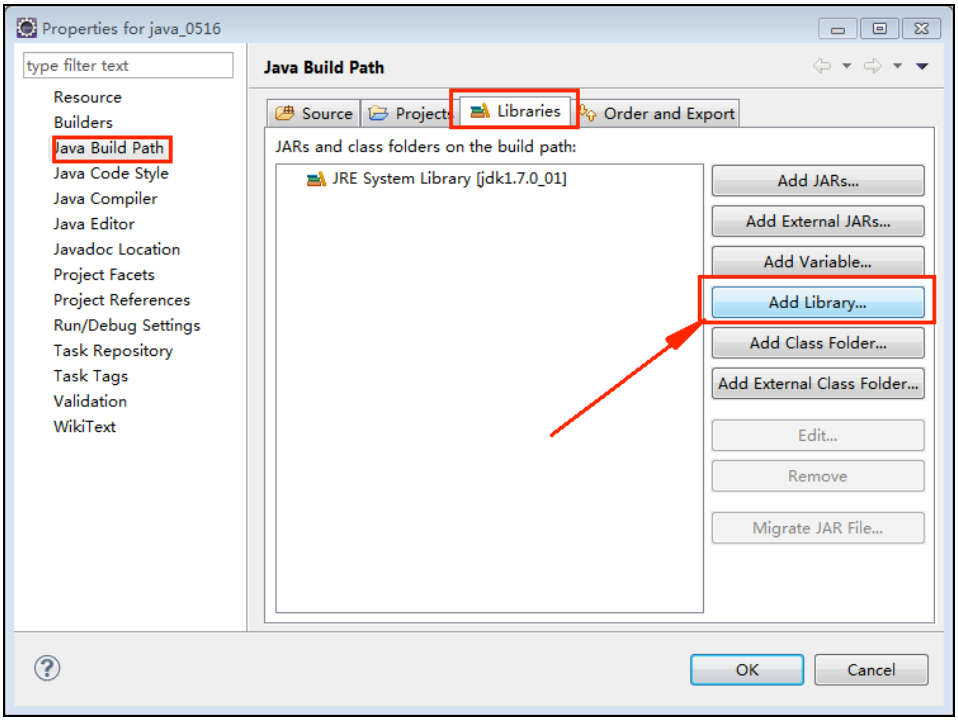
每一个 jar 文件其实就是一堆类文件。


JUnit 4
junit.jar - C:\soft\eclipse\plu
junit.extensions
junit.framework
junit.runner
BaseTestRunner.class
TestRunListener.class
Version.class
junit.textui
ResultPrinter.class
TestRunner.class
org.junit
After.class
AfterClass.class
Assert.class
Assume.class

16 Junit 单元测试

16.1 导入 Junit 的 Jar 包。





16.2 使用@Test在方法前面加

```
import org.junit.Test;

public class StringTest {
    @Test
    public void stringtest01(){
        String s1 = "abc";
        String s2 = new String("abc");
        System.out.println(s1.equals(s2));
    }

}
```

16.3 测试。

Run -> junit test

17 流程控制

三种流程控制方式：顺序，条件（选择，分支），循环

条件结构：

if, if-else

switch-case

循环结构：

While

Do-while

For,foreach

17.1 条件结构：

17.1.1 if 语句。

格式 1：if 格式

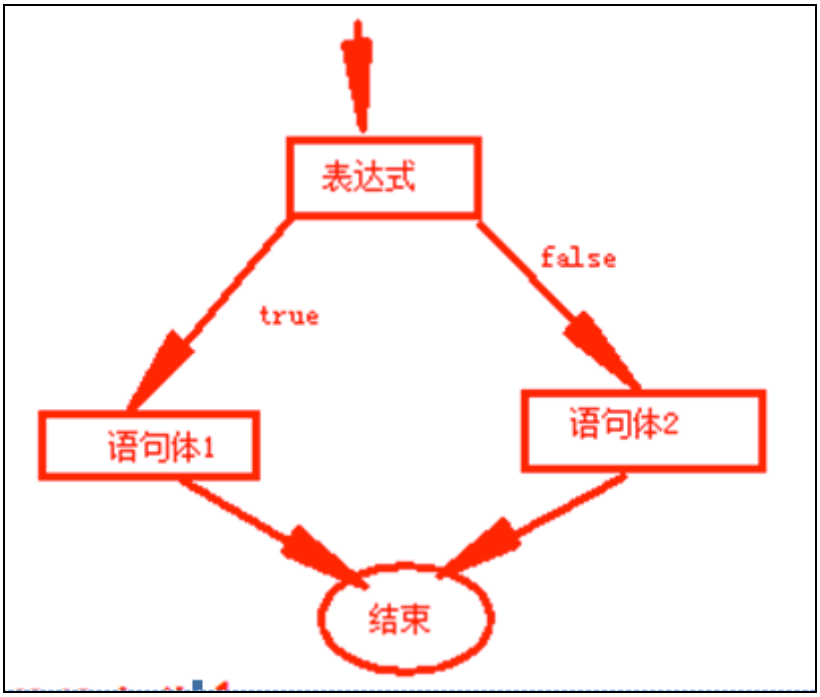
```
if(boolean 表达式){
    语句体；
}
```

执行方式：

只在当表达式为 true 时，语句体才执行。

格式 2：if-else 格式

```
if(boolean 表达式){
    语句体 1；
}else{
    语句体 2；
}
```



执行方式：

当表达式的值为 true 时执行语句体 1，

当表达式的值为 false 时执行语句体 2。

1、判断一个数字是奇数还是偶数
int i = new Scanner(System.*in*).nextInt();从控制台接收一个 int 类型的数据

int i = 10;
if(i % 2 == 0){
System.*out*.println(i+"是一个偶数！ ");
}else{
System.*out*.println(i+"是一个奇数！ ");
}

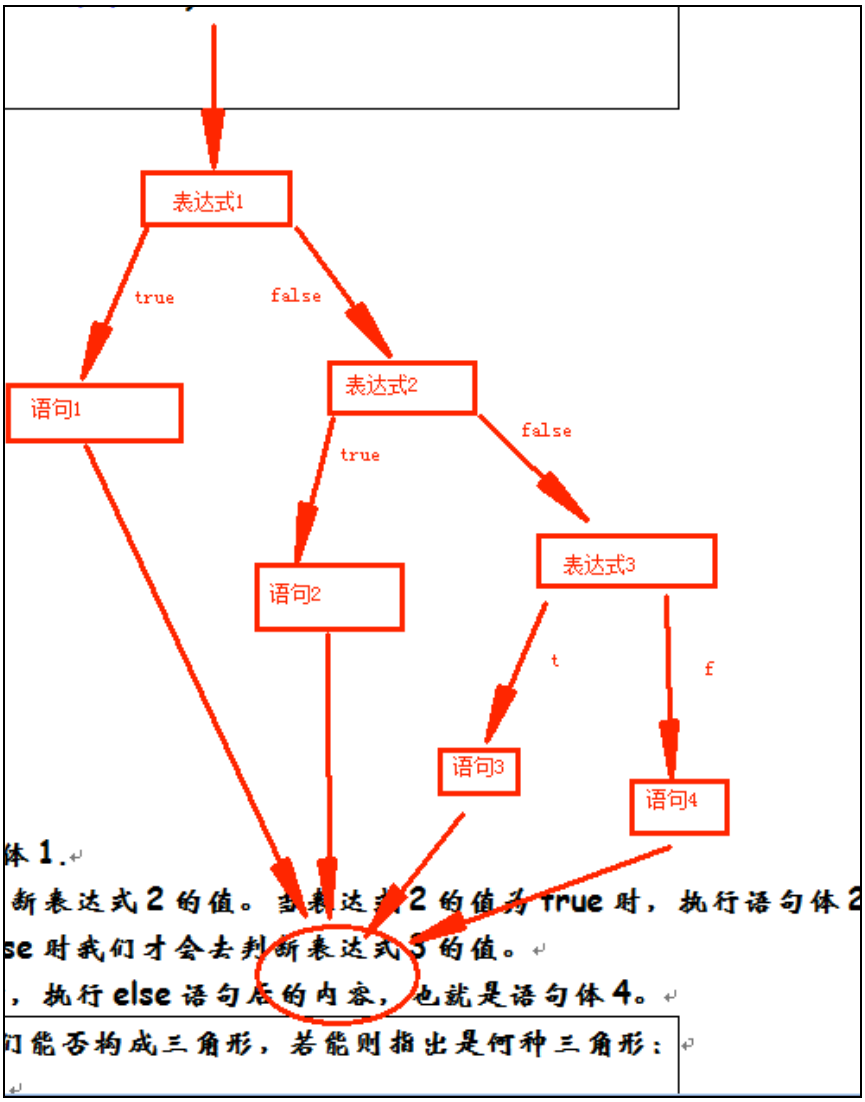
2、输入年，显示是否是闰年的 能被 4 整除但不能被 100 整除，或能被 400 整除 优先级要使用“()”
(year % 4 ==0 && year % 100 != 0) || year % 400 ==0

public void lx02(){
System.*out*.println("请输入年份：");
int year = new Scanner(System.*in*).nextInt();
if((year % 4 ==0 && year % 100 != 0) || year % 400 ==0){
System.*out*.println(year+"是一个闰年！ ");

```
        }else{
            System.out.println(year+"是一个平年！ ");
        }
    }
```

格式 3：if-elseif 格式

```
    if(boolean 表达式 1){
        语句体 1；
    }else if (表达式 2){
        语句体 2；
    }else if (表达式 3){
        语句体 3；
    }else{
        语句体 4；
    }
```



体 1.

断表达式 2 的值。当表达式 2 的值为 true 时，执行语句体 2

se 时我们才会去判断表达式 3 的值。

，执行 else 语句后的内容，也就是语句体 4。

们能否构成三角形，若能则指出是何种三角形：

执行方式：

当表达式 1 为 true 时，执行语句体 1.

当表达式 1 为 false 时，才会去判断表达式 2 的值。当表达式 2 的值为 true 时，执行语句体 2

只有当表达式 1 和表达式都为 false 时我们才会去判断表达式 3 的值。

当上面所有的表达式都为 false 时，执行 else 语句后的内容，也就是语句体 4。

4、输入三角形的三条边 a,b,c ，判断它们能否构成三角形，若能则指出是何种三角形：等腰三角形、直角三角形、一般三角形。
分别编写判断构成三角形，判断是否是等边三角型，是否是等腰三角型，是否是直角三角型的函数。

If(是不是三角形){
 	
If(等边){
等边
}else if(等腰){
 	
If(直角三角形){
等腰直角三角形
}else{
等腰
}
}else if(直角三角形){
直角三角形
}else{
一般三角形
}
}else{
不是三角形
}

是不是三角形 a+b>c && a+c>b && b+c>a
等边 a==b && b==c && c==a
等腰 a==b || b==c || c==a
直角三角形 a*a+b*b==c*c || c*c+b*b==a*a || a*a+ c*c == b*b

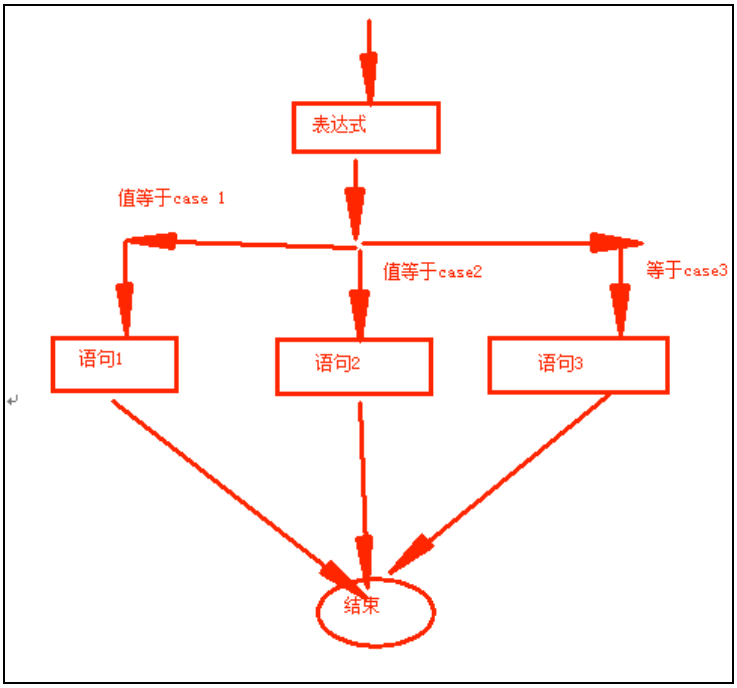
17.1.2 switch 语句

多条件选择语句。

格式：

```
switch(表达式){
    case 常量值 1：
```

```
        语句体1;
        break;
    case 常量值2:
        语句体2;
        break;
    case …
    default:
        语句体default;
        break;
}
```



注意事项：

1 switch 只能比较“==”等于。

2 switch 只能使用可以自动转成 int 类型的数据（jdk7 之前）和 String 类型的数据（jdk7）

3 switch 语句中当有一个 case 的值相等之后，后面的每一个 case 就不判断了。所以一般情况下每一个 case 子句最后都加一个 break。

4 break 语句的作用是“跳出 switch 结构”

5 default 语句没有位置的要求。

执行过程：

当表达式的值与第一个 case 关键字后的值相等时，执行相应 case 后的语句体。

当没有一个 case 的值与表达式相等时执行 default 语句。

5、在屏幕上显示一张如下所示的时间表： *****Time***** 1　morning 2　afternoon 3　night Please enter your choice: 操作人员根据提示进行选择，程序根据输入的时间序号显示相应的问候信息， 选择 1 时显示"Good morning"， 选择 2 时显示"Good afternoon"， 选择 3 时显示"Good night"， 对于其它选择显示"Selection error!"， 用 switch 语句编程实现。
从控制台接收一个整数的语句 `int i = new Scanner(System.in).nextInt();`

6、输入年月日，计算日期是今年的第几天 比如：1 月 1 日，第 1 天。 比如：2 月 1 日，第 32 天。 比如：3 月 1 日，第？天，闰年
@Test 　　public void lx06() {

```
        int year = 2100;
        int month = 4;
        int day = 10;
        int days = 0;
        switch (month) {
        case 12:
            days += 30;// 加 11 月份天数
        case 11:
            days += 31;// 加 10 月份的天数
        case 10:
            days += 30;// 加 9 月份的天数
        case 1:
            days += day;
        }
        if(month > 2){
            //闰年
            if((year % 4 == 0 && year % 100 != 0) || year % 400 == 0){
                days++;
            }
        }
        System.out.println(days);
    }
```

17.2 循环结构：

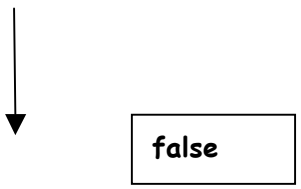
有规律的重复执行某些操作时。可以使用循环的语法结构。

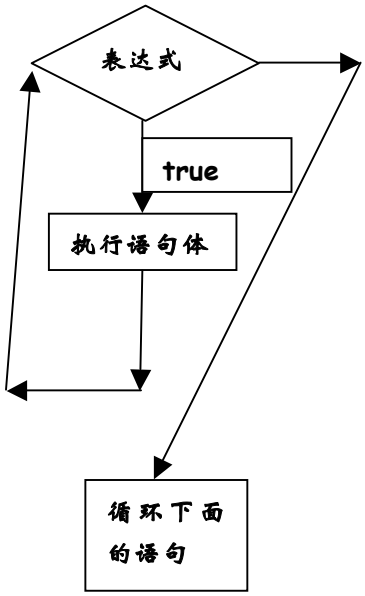
17.2.1 while 循环

格式：

```
while(boolean 表达式){

    语句体;

}
```

执行过程：





8、计算 1 到 100 的累加。

重复的行为---加法。

有规律　1 + 2 + 3+ 4 +5　规律是每次增加 1 个

```
@Test
public void lx08(){
    int sum = 0;
    int i = 1;
    while( i < 101){
        sum += i;
        i++;
    }
    System.out.println(sum);
}
```

9、计算 N!

```
@Test
public void lx09(){
    int n = new Scanner(System.in).nextInt();
    int sum = 1;//累乘用 1 初始化
    int i = 1;
    while(i <= n){
        sum *= i;
        i++;
    }
```

```
        System.out.println(sum);
    }
```

17.2.2 do-while 循环

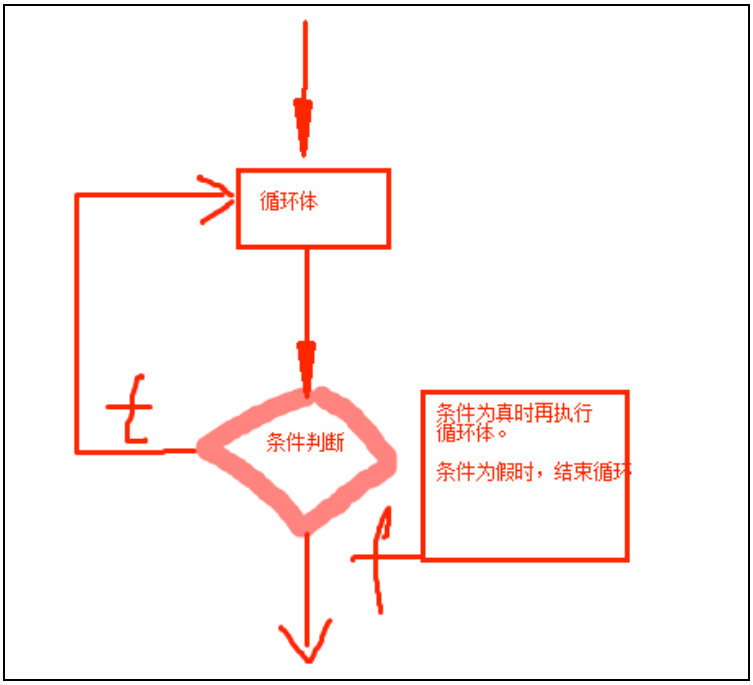
语法格式：

```
do{
    循环体;
}while(条件表达式);
```

执行方式：

先执行一次循环体，之后再判断，如果条件为 true，再执行一次循环体之后再判断，重复。

条件为 false 时结束循环。



7、重复执行"5、在屏幕上显示一张如下所示的时间表"，直到用户输入 1-3 之外的数字时结束。

```
@Test
public void lx07() {
    int i ;
    do{
```

```
            System.out.println("*****Time*****");
            System.out.println("1   morning");
            System.out.println("2   afternoon");
            System.out.println("3   night");
            System.out.println("Please enter your choice:");
            i = new Scanner(System.in).nextInt();
            switch(i){
                case 1:
                    System.out.println("morning");
                    break;
                case 2:
                    System.out.println("afternoon");
                    break;
                case 3:
                    System.out.println("night");
                    break;
                default:
                    System.out.println("error");
                    break;
            }
        }while(i>=1 && i<=3);
    }
```

17.2.3 for 循环

在 jdk5.0 中新特性，foreach 循环。放到后面数组之后再看。

常见的循环：循环的三个要素->循环变量（在循环中控制循环次数的变量）

1 循环变量的初始化

2 循环条件

3 循环变量的变化规律

语法格式：

```
for( 初始化(1) ; 循环条件(2) ; 循环变量的变化(3) ){

    循环体(4);

}
```

(5)

执行过程是：

(1)->(2)-true->(4)->(3)->(2) -true->(4)->(3)->(2) -true->(4)->(3)->(2)-false->(5,循环结束)

17.2.4 嵌套循环。

一个循环的循环体是另一个循环。

```
for(){ 外层循环

    for(){ } 内层循环

}
```

17.3 break 和 continue

17.3.1 break;退出整个循环

作用：退出。跳过。
范围：整个。
如果使用在循环中，退出整个循环。

```
@Test
//我们不知道循环次数，我们只有一个退出循环的条件。
/**
 * 累加，每次+1，当累加的值第一次大于 100 时停止
 */
public void test01(){
    int sum = 0;
    int i = 1;
    while(true){
        sum += i;
        if(sum > 100){
            break;
        }
        i++;
    }
    System.out.println(sum);
    System.out.println(i);
}
```

17.3.2 continue;退出一次循环。

作用：退出，跳过

范围：一次循环

如果使用在循环中，跳过本次的循环，进入下一次循环。

```
@Test
/**
 * 1 到 10 的累加
 * 能被 7 整除的不加。
 */
public void test01(){
    int sum = 0;
    for(int i = 1; i<= 10 ; i++){
        if(i % 7 == 0){
            continue;
        }
        sum += i;
    }
    System.out.println(sum);
}
```

18 数组

18.1 数组是什么？

数组就是保存同一类型的一大堆的变量。

当我们有一大堆的同类型的数据要使用时，应该定义一大堆同类型变量，或定义一个数组。

在内存中连续存储的空间

数组大小不能改变

数组元素都相同的数据类型

18.2 使用数组

18.2.1 声明一个数组。

int[] a; 推荐使用。

int a[];

18.2.2 实例化（new）数组。

开辟空间，确定数据大小 。

使用 new 的关键字开辟空间。

格式：

数组名 = new 类型[大小]

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

18.2.3 ?一堆的变量名都叫什么？数组名+索引

数组定义的变量的名称的格式： 数组名[索引]

索引从 0 开始到（数组大小-1）的 5 个数值。

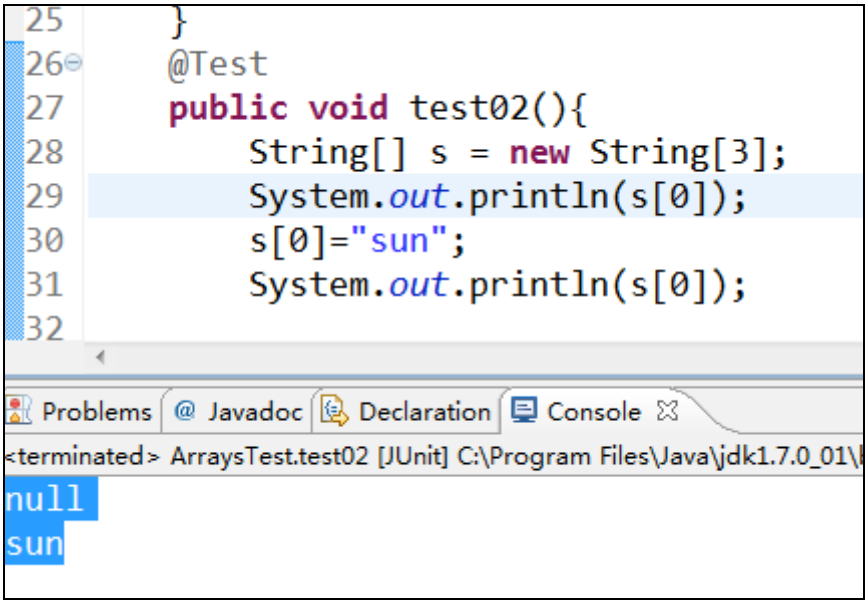
```
int[] a = new int[5];
5 个变量的名字都叫：a[0],a[1],a[2],a[3],a[4]

这 5 个变量我们又统称为：数组元素。
```

18.3 数组的默认值

数组一经使用 new 实例化之后每个数组元素都自己的有一个默认值。

原始数据类型都是 0 值。只能 boolean 是 false;

引用类型类型，默认值都为 null;



18.4 数组的遍历。迭代

将数组中的每一个元素都操作一次，输出，赋值。

数组的遍历都要配合循环一起使用。

数组的规律：每个元素的格式：数组名[索引]，索引又是从 0 开始，每次+1、到数组大小-1

```
@Test
public void test03(){
    int [] a = new int[10];
    a[1] = 1;
    a[4] = 4;
    a[7] = 7;
    a[9] = 9;
    //通过循环变量的控制让循环变量能表示数组元素的每一个索引;
    for(int i = 0 ; i < 10 ; i ++){
        System.out.println(a[i]);
    }
}
```

18.5 数组的异常- ArrayIndexOutOfBoundsException（索引超出范围）

java.lang.ArrayIndexOutOfBoundsException: 8

18.6 获取一个数组的大小

length 属性

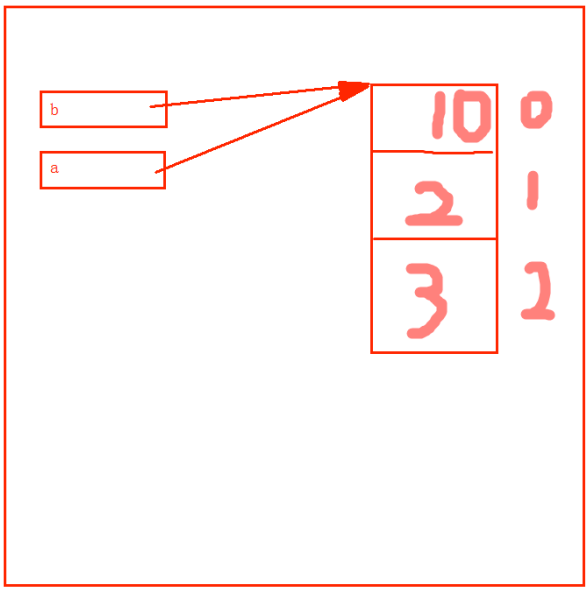
数组名.length 返回数组的大小。

```
@Test
public void test05(){
    int [] a = new int[8];
    a[1] = 1;    a[4] = 4;    a[7] = 7;
    for(int i = 0 ; i < a.length ; i ++){
        System.out.println(a[i]);
    }
}
```

18.7 数组是引用数据类型

当一个数组给另一个数组，使用数组名称赋值时，实际上是将数组的引用交给另一个数组。
两个数组使用的是同一块内存，两个数组引用是相同的。
所以一个数组对数组元素的修改等于两个数组一起改变。

```
public class TestArrayQuestion {
    public static void main(String args[]){
        byte[] b = {1,2,3};
        byte[] a = b;
        a[0] = 10;
        System.out.println(b[0]);
    }
}
```



18.8 java.util.Arrays 类。

封装了针对数组的操作方法。

18.8.1 sort 方法。排序方法。

static void	sort(byte[] a) 对指定的 byte 型数组按数字升序进行排序。

```
@Test
public void test08() {
    int[] a = { 10, 20, 15, 13, 18, 90, 3, 6 };

    Arrays.sort(a);

    for (int i = 0; i < a.length; i++) {
        System.out.print(a[i]+"  ");
    }
    System.out.println();
    for(int i = a.length-1;i>=0;i--){
        System.out.print(a[i]+"  ");
    }
}
```

18.8.2 binarySearch 方法，查找方法、

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

使用二分搜索法来搜索指定的 int 型数组，以获得指定的值。必须在进行此调用之前对数组进行排序（通过 sort(int[]) 方法）。如果没有对数组进行排序，则结果是不确定的。如果数组包含多个带有指定值的元素，则无法保证找到的是哪一个。

参数：

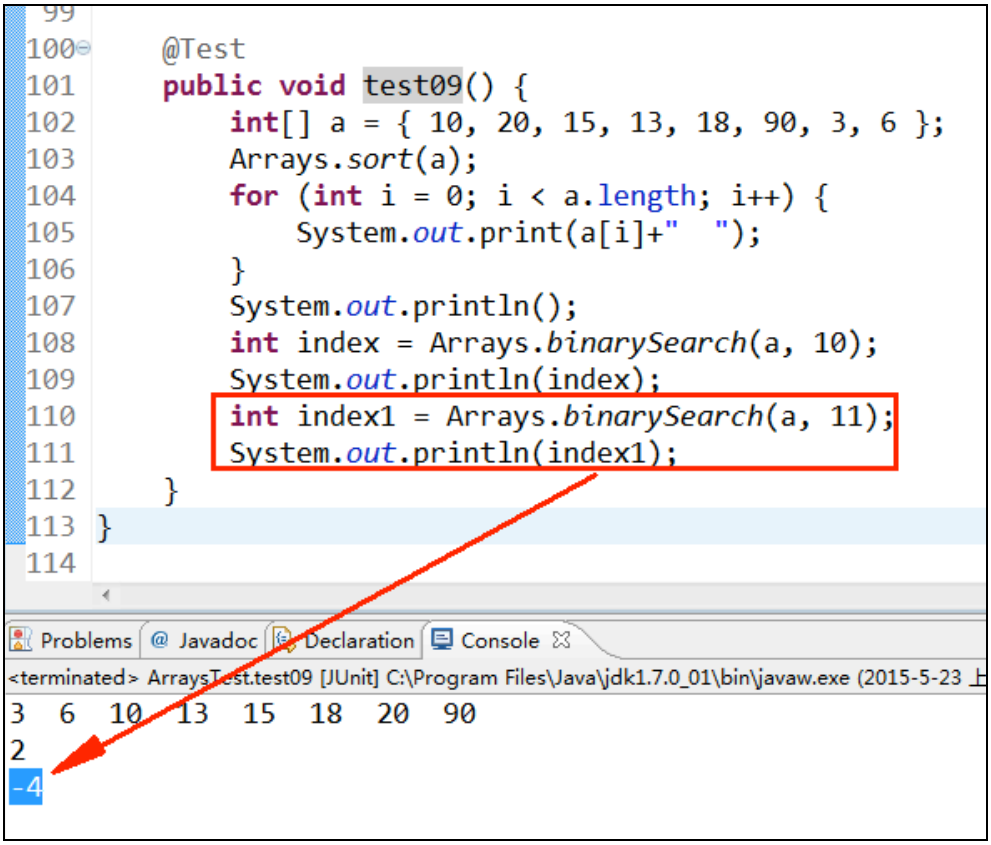
a - 要搜索的数组

key - 要搜索的值

返回：

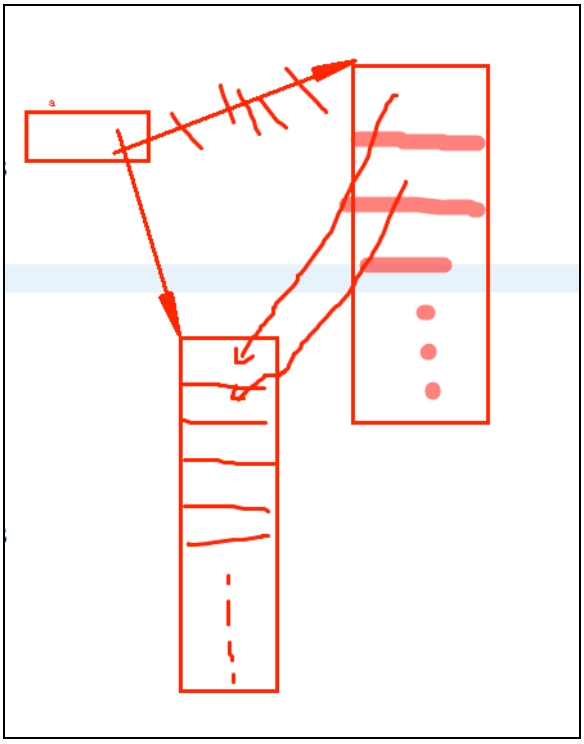
如果它包含在数组中，则返回搜索键的索引；否则返回 (-(插入点) - 1)。插入点 被定义为将键插入数组的那一点：即第一个大于此键的元素索引，如果数组中的所有元素都小

于指定的键，则为 a.length。注意，这保证了当且仅当此键被找到时，返回的值将 >= 0。



18.8.3 copyOf 使用老数组创建一个新的数组。同时拷贝数据。

```
@Test
public void test10() {
    int[] a = { 10, 20, 15, 13, 18, 90, 3, 6 };
    a = Arrays.copyOf(a, a.length*2);
    for (int i = 0; i < a.length; i++) {
        System.out.println(a[i]);
    }
}
```



18.9 练习

1、使用数组的方式实现计算

输入年月日，计算日期是今年的第几天。

```
public void lx01() {
    int year = 2000;
    int month = 5;
    int day = 1;
    int sum = 0;
    int[] days = { day, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30 };
    if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0) {
        days[2] = 29;
    }
    for (int i = 0; i < month; i++) {
        sum += days[i];
    }
    System.out.println(sum);
}
```

2、输入一个金额，输出它的人民币的组成情况（100，50，20，10，5，1）。

如：168 元，组成情况是 100 元 1 张，50 元 1 张，20 元 0 张，10 元 1 张，5 元 1 张，1 元 3 张

```
public void lx02() {
    int[] moneys = { 100, 50, 20, 10, 5, 1 };
    int money = 168;
    for (int i = 0; i < moneys.length; i++) {
        int n = money / moneys[i];
        money = money % moneys[i];
        System.out.println(n + "张" + moneys[i] + "元！");
    }
}
```

3、实现非波纳切数列。

计算第 10 位数值是多少。

```
public void lx03() {
    int [] a = new int[10];
    a[0]=1;
    a[1]=1;
    for(int i = 2; i <a.length;i++){
        a[i]=a[i-1]+a[i-2];
    }
    System.out.println(a[9]);
}
```

18.10 二维数组和多维数组

一个数组的每个元素又是一个数组。
理解经常把二维数组看成一个行列的矩阵。

18.10.1 声明一个二维数组

```
int[][] a ;
```

18.10.2 实例化

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

18.10.3 变更名叫

int [][] a = new int[2][3];

? 问 a 数组有几个元素。

两个。a[0]和 a[1].只不过 a[0]和 a[1]也是一个数组

?a[0]这个数组有几个元素。3 个。

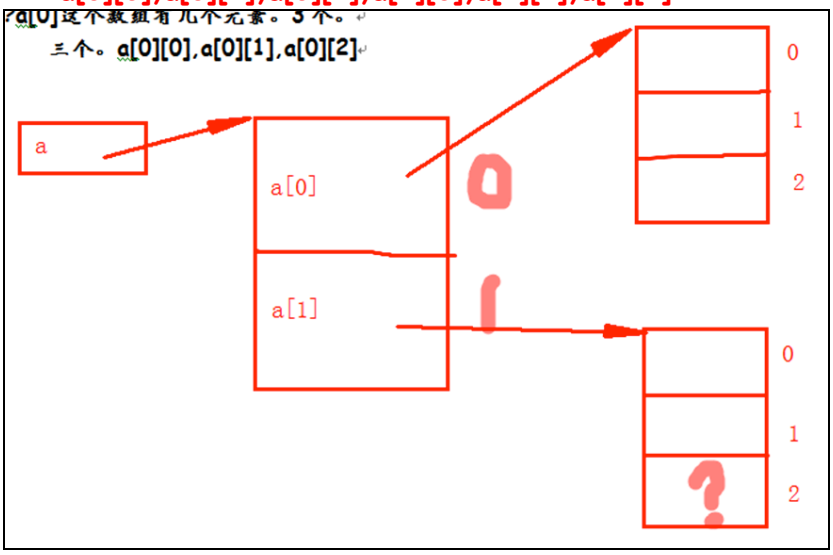
三个。a[0][0],a[0][1],a[0][2]

?a[1] 这个数组有几个元素。3 个。

三个。a[1][0],a[1][1],a[1][2]

所以最终 a 这个二维数组可以有 6 个变量存储数据。

a[0][0],a[0][1],a[0][2],a[1][0],a[1][1],a[1][2]



18.10.4 二维数组的遍历

一定要使用二层 for 循环。

```
@Test
public void test13(){
    int [][] a= { {1,2,3} , {4,5,6} };
    for (int i = 0; i < a.length; i++) {
        for (int j = 0; j < a[i].length; j++) {
            System.out.println(a[i][j]);
        }
```

```
        }
    }
```

18.10.5 不规则的二维数组。

第二维的长度上可以不一致

```
int [][] a = new int[2][];
a[0] = new int[3];
a[1] = new int[4];

int [][] a= { {1,2,3} , {4,5,6,7} };
```

18.10.6 二维数组是否可以排序？不能，会报 ClassCastException 异常。

```
    @Test
    public void test15(){
        int [][] a= { {1,2,3} , {4,5,6,7} };
        Arrays.sort(a);
        for (int i = 0; i < a.length; i++) {
            System.out.println(a[i]);
        }
    }
```

```
java.lang.ClassCastException: [I cannot be cast to java.lang.Comparable
```

18.10.7 练习

4 杨辉三角。

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

规则：

1 所有的第一列为 1

2 所有的行与列相等的位置为 1

3 其他位置= 上一个+上一个左边。

1 i=j　　X[i][j]=1

2 j=0　X[i][j]=1

3 其他　X[i][j] = X[i-1][j]+X[i-1][j-1]

```
public void lx04() {
    int n = 15;
    int[][] num = new int[n][];
    for (int i = 0; i < num.length; i++) {
        num[i] = new int[i + 1];
        for (int j = 0; j < num[i].length; j++) {
            if (i == j || j == 0) {
                num[i][j] = 1;
            } else {
                num[i][j] = num[i - 1][j] + num[i - 1][j - 1];
            }
        }
    }
    //输出
    for (int i = 0; i < num.length; i++) {
        for (int j = 0; j < num[i].length; j++) {
            System.out.print(num[i][j]+"\t");
        }
        System.out.println();
    }
}
```

19 函数

19.1 为什么要创建函数?

一个函数代表了一个功能。功能最好单一，独立，不可拆分。

可以重复使用

调用方便

结构清晰，层次更明确，修改容易。

可以利用函数将多层的循环变成多个一层的循环。

为了给我们的程序分模块。

从现在开始你发现一个基本的现象。拆。Main 方法所有内容，拆出来变成一个一个的函数。

19.2 怎么创建函数?

```
public static 返回值类型 函数名称(函数的参数列表){
    函数体;
}
```

```
public static int sum(int a, int b) {
    int s = a + b;
    return s;
}
```

public static ： 目前不要多想。

返回值类型 ： 一个函数代表一个功能。返回值是一个函数执行完成之后返回的数据，描述返回值这个数据的类型。

函数名称 ： 调用函数时使用的名字。

参数列表 ： 一个函数代表一个功能。有时调用功能时要提供数据。比如计算两个数和，调用这个功能一定要提供两个数。

函数体 ： 是实现功能的代码。

19.3 函数的参数

19.4.1 参数的作用：

调用函数时提供给函数的数据。

例如：计算两个数的和。想调用计算的功能。就必须提供两个数。

public static int sum(int a , int b)

19.4.2 参数使用：

1 在声明一个函数时，设置形参。int a , int b

设置功能实现时需要的数据类型。以及接收数据的变量名称。

public static int sum(int a , int b)

2 调用函数时，设置实参：

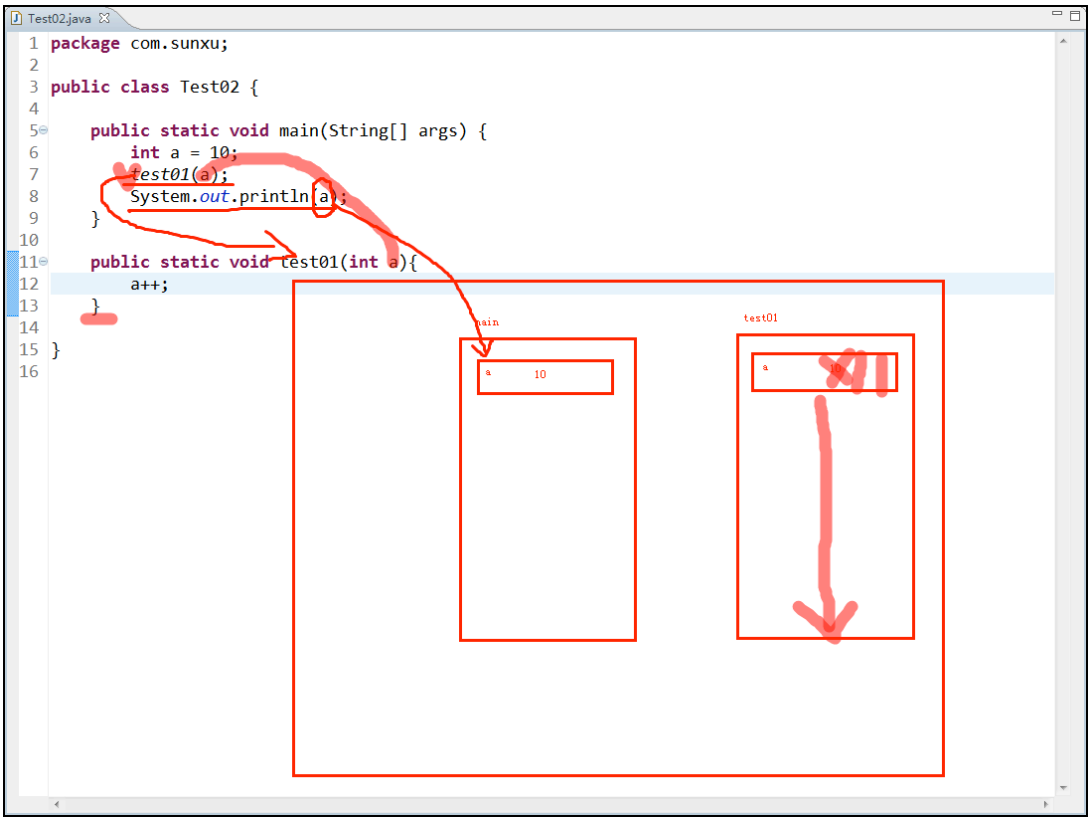
调用函数时用来真正给 a 和 b 赋值的数据。

sum(10,20)

19.4 不同数据类型对于参数的影响（重点）

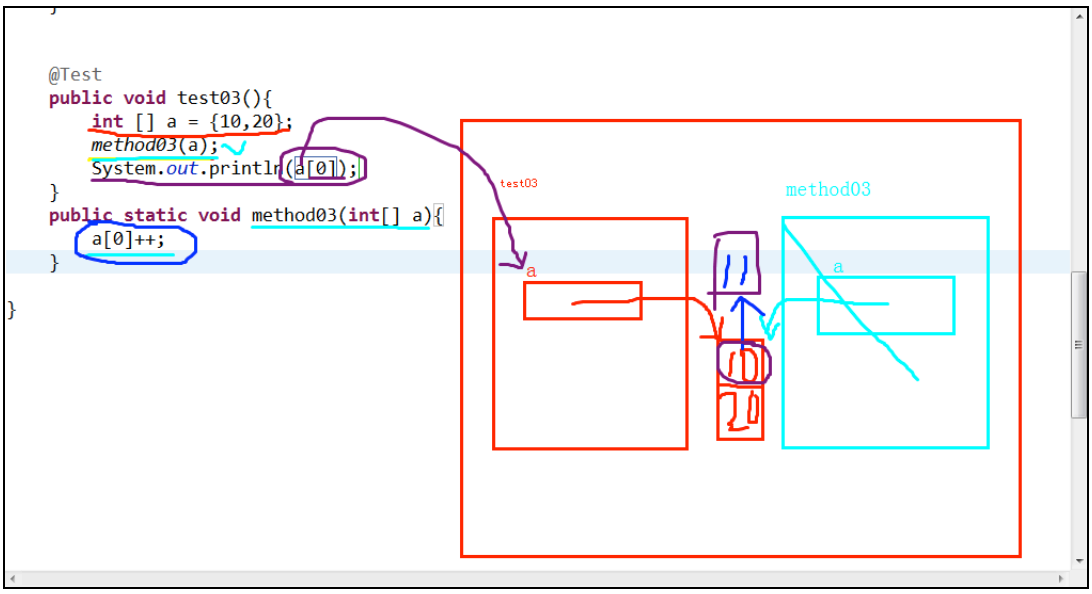
分成二大类。原始和引用

原始数据类型做为参数时：是两个变量。在函数中对参数的变量不影响实参的值。



引用数据做为参数时：当两个引用数据类型赋值时，表示两个引用指向同一个空间，两个引用可能认为是同一个。\所以一个引用修改，另一个引用也一起变化。

String（特殊，不通用），数组。



19.5 函数的返回值类型。

表示函数执行完成之后返回的结果的数据类型。

1 void ： 当函数没有返回值时，使用 void 表示。

一个无返回值，无参数的函数的调用格式：

函数名();//调用。

2 return ： 关键字。当函数有返回值时，使用 return 确认返回值是什么。

格式：return 返回值;

例如：返回两个数的和。

```
public static int sum(int a , int b){
    int s = a+b;
    return s;
}
```

return s ； 表示 s 是 sum 这个函数的返回值。这个函数的结果就是 s 变量的值。

int sum(int a , int b) sum 前面的 int 表示 sum 这个方法将返回一个 int 类型的数值。

有返回值的函数的调用 ，一般都会使用一个与返回值类型相同的变量接收。

调用格式：返回值类型 变量名=函数名();

return 关键字的使用是什么？返回。一个函数运行过程中一旦执行了 return 代

表这个函数结束。

特殊的使用方式：如果在函数直接出现 return ； 表示函数的结束。

19.6 练习：

2、编写函数要求，根据输入的参数在一行中打印“*”号。如果参数为 4 则在一行中输出“*****”

```
public static void print02(int n){
    for (int i = 1; i <= n ; i++) {
        System.out.print("*");
    }
}
```

3、使用 2 题的函数，实现图形的输出

```
*
**
***
****
*****
```

```
public static void print03(int n) {
    for (int i = 1; i <= n; i++) {
        print02(i);
        System.out.println();
    }
}
```

```
public static void main(String[] args) {
    print03(7);
}
```

4、打印九九乘法表。

```
1*1=1
1*2=2 2*2=4
1*3=3 2*3=6 3*3=9
```

```
@Test
public void lx04() {
    for (int i = 1; i <= 9; i++) {
        for(int j = 1; j <=i ; j++ ){
```

```
                System.out.print(j+"*"+i+"="+(i*j)+"   ");
            }
            System.out.println();
        }
    }
```

```
    @Test
    public void lx04() {
        for (int i = 1; i <= 9; i++) {
            printlx04(i);
            System.out.println();
        }
    }
```

```
    public static void printlx04(int i){
        for(int j = 1; j <=i ; j++ ){
            System.out.print(j+"*"+i+"="+(i*j)+"   ");
        }
    }
```

5、编写一个计算 n!的函数，使用函数计算 1!+2!+3!+...+n!

```
    @Test
    public void lx05() {
        int x = 5;
        int sum=0;
        for(int i = 1; i <=x; i++){
            sum += jc(i);
        }
        System.out.println(sum);
    }

    public static int jc(int n) {
        int jc = 1;
        for (int i = 1; i <= n; i++) {
            jc *= i;
        }
        return jc;
    }
```

6、编写一个判断一个数字是不是“水仙花数”的函数，再调用函数打印所有的“水仙花数”。

判断返回 boolean

是不是什么？ isXxxx()

有没有什么？hasXxxx()

```
@Test
public void lx06() {
    for(int i = 100; i<=999 ;i++){
        if(isSXH(i)){
            System.out.println(i);
        }
    }
}

public static boolean isSXH(int i) {
    int a = i / 100;// 百位
    int b = i / 10 % 10;// 十位
    int c = i % 10;// 个位
    return i == a * a * a + b * b * b + c * c * c;
}
```

7、编写一个函数用来判断是否是闰年。

8、打印图形

如果输入的 4

```
   *
  ***
 *****
*******
 *****
  ***
   *
```

```
@Test
public void lx08() {
    int rows = 10;
    for (int i = 1; i <= rows; i++) {
        printrow(rows,i);
    }
    for (int i = rows - 1; i >= 1; i--) {
        printrow(rows,i);
    }
}
/**
```

```
     * 打一行
     */
    public static void printrow(int rows ,int i){
        print(rows - i, " ");// 打空格
        print((2 * i) - 1, "*");// 打星号
        System.out.println();
    }
    /**
     * 将打空格与打星号合成一个函数
     */
    public static void print(int n, String s) {
        for (int i = 1; i <= n; i++) {
            System.out.print(s);
        }
    }

    /**
     * 打星号(过程版本，不使用了)
     */
    public static void prints(int n) {
        for (int i = 1; i <= n; i++) {
            System.out.print("*");
        }
    }

    /**
     * 打空格(过程版本，不使用了)
     */
    public static void printk(int n) {
        for (int i = 1; i <= n; i++) {
            System.out.print(" ");
        }
    }
```

19.7 函数的递归调用

递归调用：在一个函数当中调用自己叫递归调用。

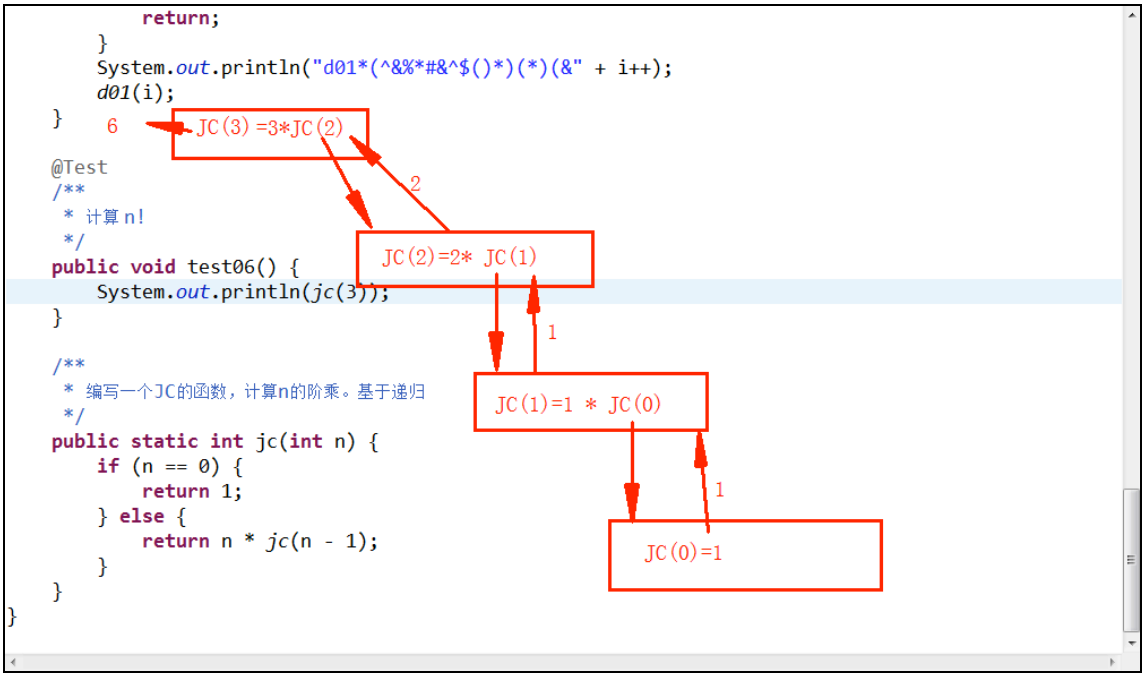
在递归调用中一定要有一个终止的条件。

n! = n * (n-1)! (n>0) 重复规律

n! = 1 (n=0) 结束条件

写一个叫 jc 的方法：

```
/**
 * 编写一个JC的函数，计算n的阶乘。基于递归
 */
public static int jc(int n) {
    if (n == 0) {
        return 1;
    } else {
        return n * jc(n - 1);
    }
}
```



19.8 递归练习

9、将十进制转换为二进制。
重复内容：打印余数之前先打印商除 2 的余数。 结束条件：当商为 0 当被除数除以 2 的商为 0 时，打印被除数除以 2 的余数 当被除数除以 2 的商不为 0 时，打印被除数除以 2 的商除以 2 的余数，再打印被除数除以

2 的余数

```
public void lx09(){
    print09(6);
}
/**
 * 打印除 2 余数的函数
 */
public static void print09(int n){
    if(n/2==0){
        System.out.print(n%2);
    }else{
        print09(n/2);
        System.out.print(n%2);
    }
}
```

10、小猴子第一天摘下若干桃子，当即吃掉一半，又多吃一个.第二天早上又将剩下的桃子吃一半，又多吃一个.以后每天早上吃前一天剩下的一半另一个.到第 10 天早上猴子想再吃时发现，只剩下一个桃子了.问第一天猴子共摘多少个桃子？

重复内容：当天数<10 时：f(天数)=(f(天数+1)+1)*2

结束条件：当天数=10 时：返回 1

```
@Test
public void lx10() {
    System.out.println(getApple(1));
}
/**
 * 返回当天有多少个苹果的函数 当天数<10 时：f(天数)=(f(天数+1)+1)*2 当天数=10 时：返回 1
 */
public static int getApple(int day) {
    if (day == 10) {
        return 1;
    }else{
        return (getApple(day+1)+1)*2;
    }
}
```