第二阶段 Java源码测试

## 一：课程内容

1. **Java基础知识。（javaSE）**
   1. **Java基础**
2. **JavaEE企业级技术**
   1. **Servlet,jsp，jdbc……**
3. **测试技术，测试我们后面的项目：电商。**
   1. **白盒测试技术**

## 计算机基础知识

软件开发：

* 1. 软件是什么：一系列按照特定顺序组织的计算机数据和指令的集合
     1. 软件的组成
        1. 程序
        2. 数据：比如我们在输入框输入用户名和密码（张三、root123）
        3. 文档：用户使用说明书。
     2. 软件的分类：
        1. 系统软件：
           1. 操作系统：win 7,win10,linux,unix,android,ios
           2. 直接安装在硬件上面。
        2. 支撑软件：
           1. 可以在此平台开发其他软件的软件。
           2. 比如、eclipse,myeclipse,tomcat,oracle
        3. 应用软件：
           1. 比如QQ，微信，迅雷……
     3. 跟计算机交互的几种方式
        1. 命令行交互：
           1. Linux命名行，DOS(disk operating system)
        2. 界面交互
           1. 最主流的交互方式
        3. 多媒体交互方式：
           1. 用声音或者声波，脑电波来交互。
     4. Dos命令的使用
        1. 打开方式：win + r ,然后输入cmd，回车。
        2. dir : 列出当前目录下的文件以及文件夹
        3. ▪ md : 创建目录
        4. ▪ rd : 删除目录
        5. ▪ cd : 进入指定目录
        6. ▪ cd .. : 退回到上一级目录
        7. ▪ cd\ : 退回到根目录
        8. ▪ del : 删除文件
        9. ▪ exit : 推出dos命令行
        10. ▪ cls：清屏命令
     5. 常用键和组合键的使用
        1. Tab制表符：就相当于4个空格。（在Linux，dos可以将文件或者目录的名称自动补全。）
        2. Shift上档键：切换编辑环境。
           1. 全角与半角的转换：

Shift + 空格

* + - 1. Window 键与其他键的组合使用：
         1. Win + d:显示桌面。
         2. Win + r:打开运行窗口。
         3. Win + l:锁屏。
         4. Win + e:打开我的电脑
      2. Ctrl键与其他键的使用
         1. Ctrl +a:全选。
         2. Ctrl +c :复制
         3. Ctrl + v:粘贴:
         4. Ctrl + x:剪切
         5. Ctrl + z:撤销操作。
         6. Ctrl + s;保存。

## java概述

### java发展历史

1. Java的创始人詹姆斯高斯林1955年出生于加拿大。毕业后在第一家公司IBM公司上班，不久就跑到sun公司工作，在sun公司詹姆斯高斯林领导小组就开发一个项目green project。但是最后项目完成后产生了一个副产品-新的编程语言oak之后不久。2009年sun公司被Oracle收购了。所以詹姆斯高林就到Oracle公司去工作了，但是也不久，就又跳槽到google公司工作。也没多久，现在在一家智能机器人研究公司，担任首席架构师。

### Java的发展：

❖Java V1.0 1996-01-23发行 (代号Oak)

❖…

❖…

❖…

❖Java V1.5 2004-09-30发行 (代号Tiger)

❖Java V1.6 2006-12-11发行 (代号Mustang)

❖Java V1.7 2011-07-28发行 (代号Dolphin)

❖Java V1.8 2014-03-18发行(代号Spider)

❖Java V1.9 2017-09-21

Java刚开始属于sun公司，2009年Oracle花了74亿美元将sun公司收购。2009年后java的版权是属于Oracle。

### java语言的特点

A）javaSE（java standard edition java标准版本）

是学javaEE和javaME的基础

B）javaEE （java enterprise edition java企业版）

是专门为企业级开发所提供的解决方案

C）javaME（java micro edition java微小型版）

是专门为手机移动端定制的，可以开发一些手机端应用程序

1. java语言的特点
2. 面向对象：就是将问题分配给对象去做。
3. 开源：免费，源代码对外开放，
4. 跨平台：可以在不同的平台进行使用。
5. 首先要安装java的解释执行程序：jvm：java虚拟机。
6. Write once ,run anywhere，对于写完后的源代码，可以实现一次书写，处处运行。
7. Jre和jdk
8. JRE(Java Runtime Environment Java运行环境) 包括Java虚拟机(JVM Java Virtual Machine)和Java程序 所需的核心类库等，如果想要运行一个开发好的Java程序， 计算机中只需要安装JRE即可。
9. JDK(Java Development Kit Java开发工具包) JDK是提供给Java开发人员使用的，其中包含了java的开发 工具，也包括了JRE。所以安装了JDK，就不用在单独安装 JRE了。其中的开发工具：编译工具(javac.exe) 打包工具 (jar.exe)等。

### Jdk的下载和安装

1. 下载：访问官网：java.sun.com或者www.oracle.com
2. 安装：
3. 傻瓜式安装：一直点击下一步。
4. 注意：安装时路径中不能有空格和中文字符。
5. Jre可以选择不安装。
6. 配置环境变量
   * 1. 作用：在dos的任意位置都能执行某一个程序。
7. 验证：
   1. 打开dos窗口，输入java 或者javac ,查看版本：java -version.

### 编程工具：

* 1. Notepad :记事本（windows系统自带的）
  2. Notepad++:
  3. Edit plus：editplus是收费的所以还要用keygen来激活。
  4. UE:

#### 一些重量级编程工具：

* + 1. Eclipse：免费的。
    2. Myeclipse:收费的。
    3. IntelliJ IDEA：收费的。
    4. JBuilder：收费的。

1. 第一个hello world 程序：
   1. 新建一个.java格式文件。
   2. 在.java文件里输入以下内容：
      1. public class Student {
      2. public static void main(String[] args){
      3. System.out.println("hello world");
      4. }
      5. }

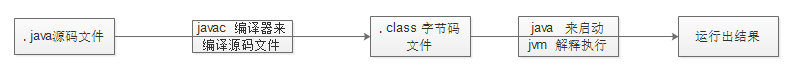
验证：在源代码那个窗口打开dos，javac [文件].java

如果编译成功，用java [文件].回车。

### 编译错误的一些原因

1. 后缀引起的错误。
2. 关键字书写错误。
3. 书写规范。
4. Jdk:
   1. 只需要该盘符和路径：
      1. 不要有空格和中文字符。

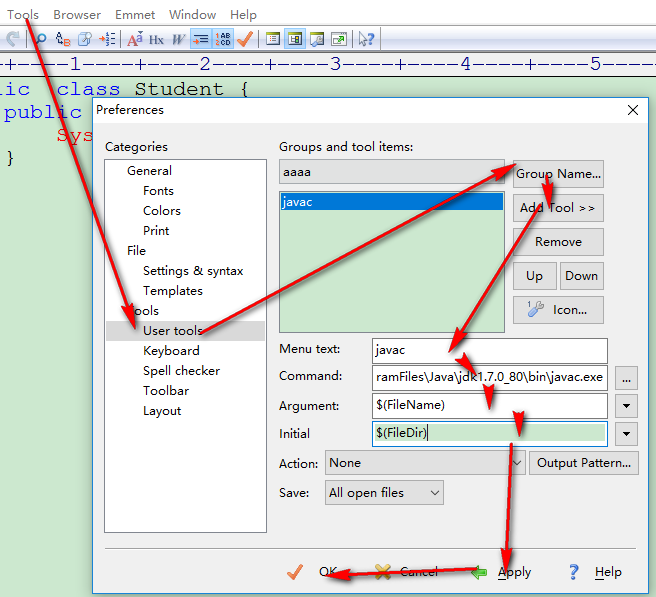
### 编译以及运行原理讲解：

* 1. 
  2. 先通过javac编译器编译源码文件🡪.class的字节码文件🡪通过java来启动jvm来解释执行🡪运行出结果。

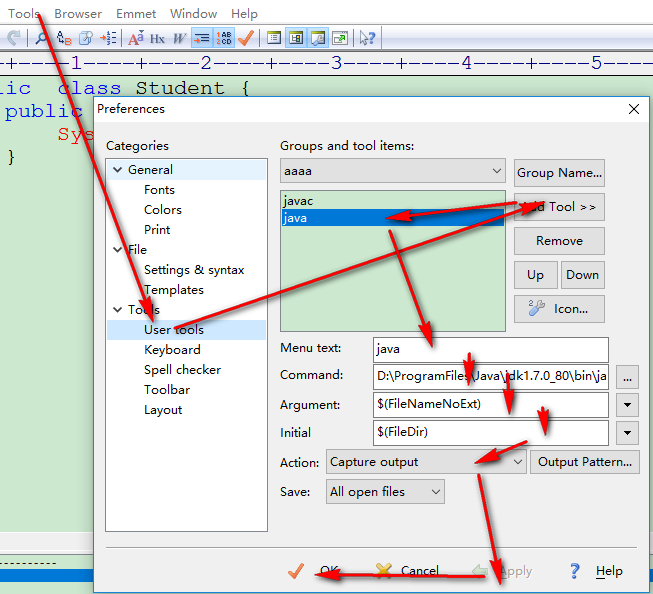
1. Classpath :
   1. 作用：JKD1.5之后自动把class文件生成到当前目录
      1. 配置classpath之后， 会去指定目录下去寻找 class文件

### Editplus 的配置：

#### Javac配置：



#### Java配置：



## Java基础语法1：

内容：

1. 关键字
2. 注释
3. 标识符
4. 常量/变量
5. 数据类型/类型转换
6. 进制的表现形式以及转换

### 关键字

在java里，被java赋予了特殊的含义。并且所有的字母都是小写的。

1. 有多少？50个。
2. 在editplus或者ue或者eclipse这些关键字都是高亮显示。
3. 
4. 

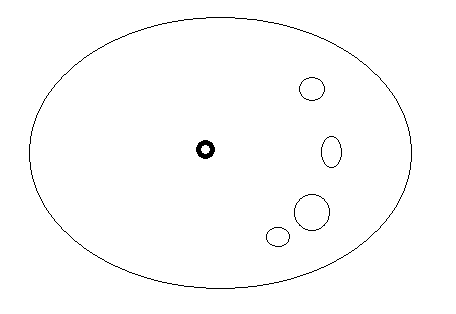
### 注释

1. 单行注释：// 注释部分
2. 多行注释：/\* 注释部分 \*/
3. 文档注释：/\*\* 注释部分\*/：java独有的注释。其他语言没有。
   1. 使用方法：
      1. 写在方法的上面或者类的上面。
   2. 一个作用：可以用javadoc来生成帮助手册。
      1. 操作：先进行编译。
         1. 然后用Javadoc来生成网页形式的帮助手册。
   3. 注释是不被jvm所解释的。是给开发或者其他人看的。
4. 作用：
   1. 提高代码的可阅读性。减少阅读的时间。

### 标识符

1. 什么是标识符？
   1. 标识类名、方法名、变量名等的符号。
   2. 就是给Java程序中的接口、类、方法、 变量等命名的字符序列
   3. 命名规则：
      1. 26个英文字母大小写。
         1. a~z
         2. A~Z
      2. 0~9:10数字
      3. $和\_
   4. 命名的注意事项：
      1. 数字不能开头。
      2. Java已定义的关键字不能进行使用。
      3. 严格区分大小写。
   5. 应用：
      1. 变量名：如果是多个单词组成，第一单词全部小写，第二个单词以后的首字母大写。
         1. Int num
         2. Int studentNum
      2. 包名：多单词组成时所有字母都小写。
         1. com.sina.user 类名接口名:多单词组成时，所有单词的首字母大写
      3. 类名接口名:多单词组成时，所有单词的首字母大写
         1. UserAgentInterface

### 变量与常量

1. 变量
   1. 什么是变量？
      1. 可以变化的来存储值。
      2. 是一个基本的存储单元，其存储的数据是可以改 变的，程序通过一个变量名来代表该变量存储的数据 。
      3. 变量的作用域：
         1. 在一对大括号之间有效。
            1. 就相当于太阳发射的光芒，在地球都可以接受到。那么相反地球的小灯泡站在太阳上是接受不到的。
            2. 
            3. 对于变量的定义不能进行重复定义。
            4. 一条语句可以同时定义多个变量。
         2. 变量的定义格式：
            1. 变量声明格式

数据类型 标识符 [= 值]

* + 1. Int a =5;
    2. Int a =9;
  1. 常量：
     1. 什么是常量？
        1. 是常规不变的量。
     2. 分类：
        1. 整数常量：所有整形值.
        2. 浮点类型：所有的小数。
           1. Float:单精度
           2. Double:双精度。
           3. //浮点类型
           4. float a = 3.14f;
           5. double b = 3.14;
        3. 布尔类型：
           1. 永远只有两个

True:真

False:假

* + - 1. 字符常量：
         1. 比如：厕所贴的‘男’或者‘女’
         2. 用单引号引起来的字叫字符。
         3. System.out.println('女');
         5. System.out.println('1');
         7. System.out.println('a');
      2. 字符串常量：
         1. 将字符串起来的量
         2. 比如：“abcd”
    1. 转义字符
       1. 将原本的意思转换为其他意思的字符。
       2. 比如System.out.println('n’);加一个斜杠System.out.println('\n');该句的意思就是换行。
          1. 换行：\n
          2. 水平制表符：\t
          3. 双引号转义：\”

可以打印双引号。

### 数据类型

1. Java数据类型：
   1. 11种数据类型。
      1. 基本数据类型
         1. 数值类型
            1. 整数类型

Byte:8位=8bit = 1字节

Short = 16位 = 16bit = 2字节

Int = 32位 = 32bit = 4字节

Long = 64位 = 64bit = 8字节

* + - * 1. 浮点类型

Float

double

* + - 1. 字符类型
         1. char
      2. 布尔类型
         1. boolean
    1. 引用数据类型
       1. 类：class
       2. 接口：interface
       3. 数组：[ ]

### 数据类型转换

1. Byte、short、int、long

#### 隐式转换：

* 1. 编译器帮我们做的数据类型，没有报错的。
  2. 比如byte，int两种数值进行相加。
     1. 首先编译器将byte类型的数值转换为int类型。然后再和其他int类型数值进行相加。

#### 强制转换：

* 1. 为什么要进行强制转换？
     1. 比如int的数值类型想往byte的类型的变量里装，这个时候编译器报错：可能会损失精度。
     2. 所以我们要在数值的前面加上要转换的类型：
        1. Int num1 = 56;
        2. Byte num2 = 12;
        3. Byte num3 = num1 + num2;//报可能损失精度错误。
        4. 但是如果非要想存放在num3变量里，要在结果的前面加个类型关键字。
           1. Byte num3 = **(byte)**(num1 + num2);

注意一点：

Byte num ;

Num += 5；

与num = num +5；的区别。

### 进制的表现方式：

1. 二进制：
   1. 只有01，没有2,逢2进一。
   2. 表现形式：0b.
      1. 比如：
      2. 0b0110
2. 八进制：
   1. 只有：0,1,2,3,4,5,6,7，没有8，逢8进一
   2. 表现形式：0开头
      1. 比如：01235
3. 十六进制：
   1. 只有：0~9和A~F
   2. 表现形式：0x
      1. 比如：0xd2
4. 进制之间的转换：
   1. 二进制🡪十进制：
      1. 把系数×基数的权次幂相加即可
      2. 0b0010:
         1. 系数：1
         2. 基数：2
         3. 0\*2^0+1\*2^1+0\*2^2+0\*2^3=2;
      3. 口诀：
         1. 8421码：
      4. 0b011101:
   2. 其他进制的转换：
      1. 需要转换到中间数：先将进制转换到10进制。然后再转到另一个进制。
      2. 073🡪十六进制：
         1. 先转到十进制：3\*8^0+7\*8^1=59
         2. 在转到十六进制：用辗转相除法。
            1. 结果是商是3余11
            2. 即0x3b，那么这就八进制073的十六进制。

### 原码、反码、补码（了解）

1. 原码：
   1. 对于正数：原、反、补都是一样的。
   2. 对于负数：原码---符号位不变，其他为取反---🡪反码-------加1------补码.
   3. 练习：
      1. 对25求原、反、补。
      2. 对-36求原、反、补。

## 基础2

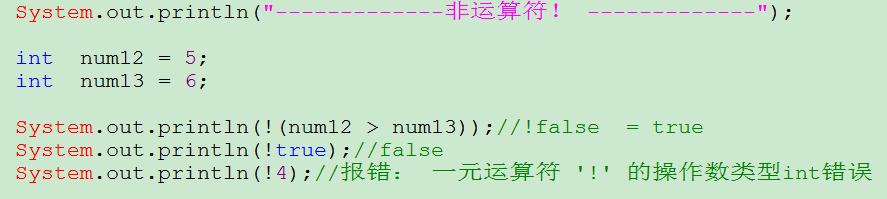
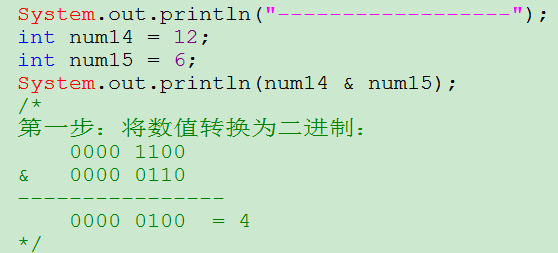
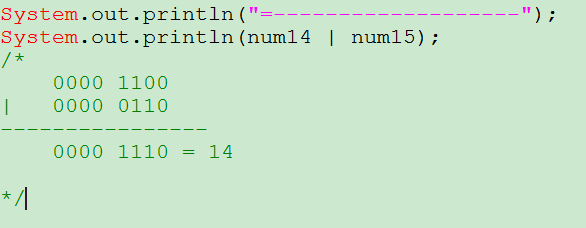
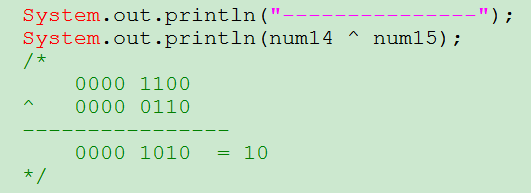
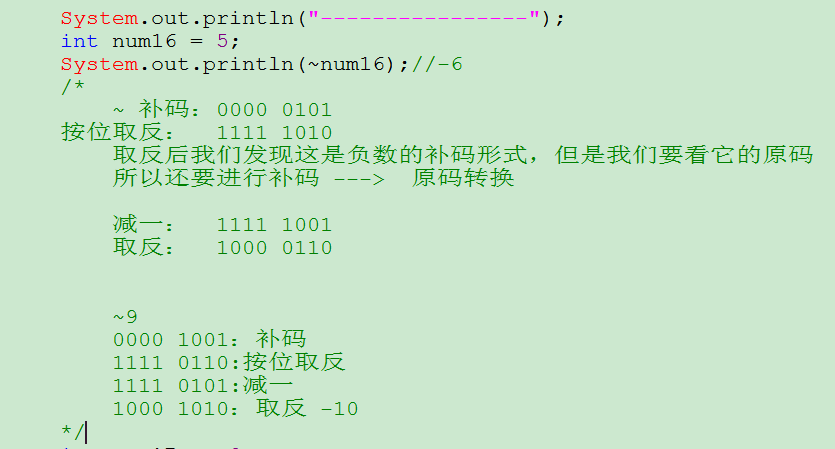
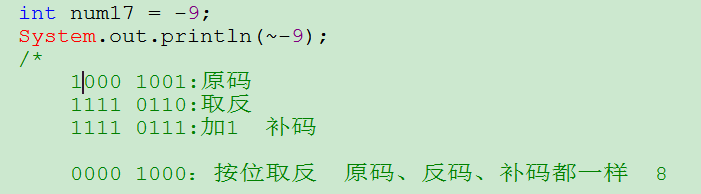
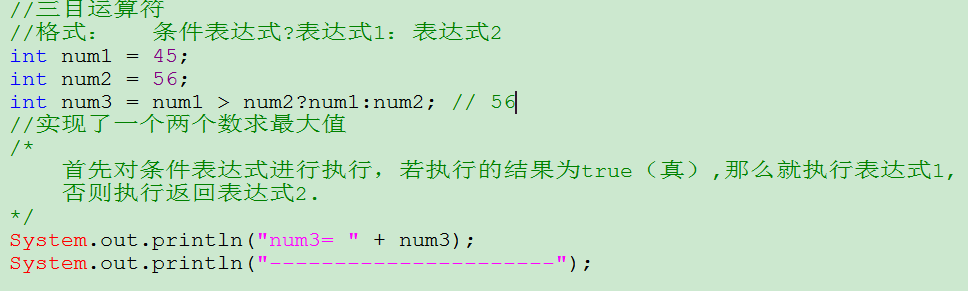
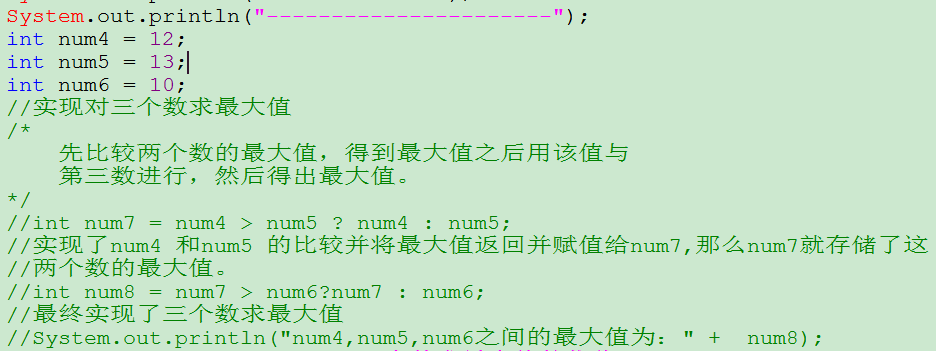
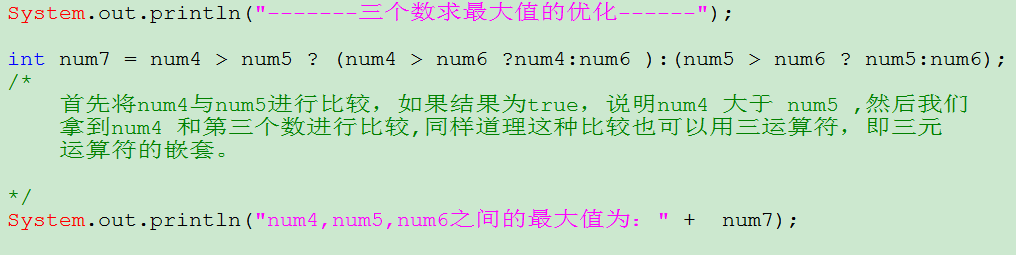
课程内容：

1. 运算符
2. 数据输入
3. 流程控制语句

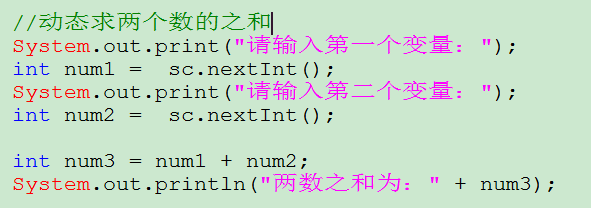
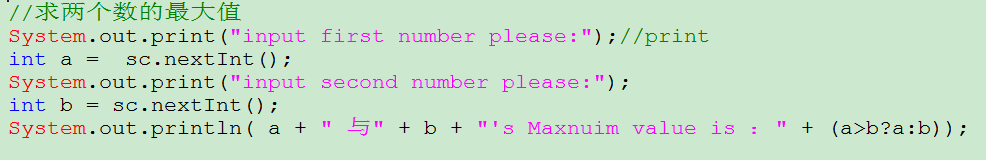
### 运算符：

* 1. 算术运算符：
     1. + - \* / % ++ -- += -= %=
     2. 1：+
        + 1. 用作连接
          2. 用作数值的相加
        1. -
           1. 数值相减
        2. \*
           1. 数值的相乘
        3. / 除法取得商值
           1. 数值的相除

2):对于除数或者被除数中其中有一个为小数，那么结果也是小数。

* + - 1. % 取商后的余数
         1. 取模运算
         2. 如果被除数大于除数，结果是取模运算
         3. 如果被除数小于除数，结果是被除数
         4. 对于取模运算：运算结果的正负只与被除数有关
      2. ++：自增
         1. 1):++放在前面
         2. 2):++放在后面
         3. 3):对于++放在前面，表达是先将值进行自增，然后进行打印，或者其他操作，对于++放在后面，表达的先把值拿出去进行操作，然后在进行自增。
      3. --：
         1. 1):--放在前面
         2. 2):--放在后面
         3. 3):对于--放在前面，表达是先将值进行自减，然后进行打印，或者其他操作,对于--放在后面，表达的先把值拿出去进行操作，然后在进行自减。
      4. +=:
         1. 比如:int num = 5;
         2. Num += 2就是 num = num + 2
         3. 所以结果就是7
      5. -=：
         1. 比如：int num1 = 4;
         2. Num1 -= 2;就是 num1 = num1 – 2;
         3. 所以结果为2；
  1. 逻辑运算符
     1. && 与 &区别：
        1. &&运算符在运算的时候，若前面的表示为false ,那么后面的条件就不执行了。
        2. 相反&运算符在运算的时候，若前面的表达式为false，那么后面的表示依然会执行。
        3. 一般都用&&运算符，因为若程序的非常庞大的时候，&运算符前面条件为false,那么如果在执行后面的程序，就显的效率非常低（因为后面的条件执行就没有意义了）。
     2. || 与 | 的区别：
        1. || 运算符在运算的时候后，若前面的表达式为true,那么后面的表达式就不执行
        2. | 运算符在运算的时候，不管前面的表达式是否为true，false，后面表达式依然会执行一遍。
        3. 所以我们一般用||运算符来运算，因为效率问题。
     3. ！非运算符：
        1. 表示把运算结果取反。
        2. 只对boolean变量有效。对数值类型无效，会报错。
        3. 
  2. 关系运算符
     1. 就是比较两者的关系。
     2. > < == >= <=
        1. 注意一点：一个等于号表示的是赋值运算。如果要比较两个值是否相等，则用==来比较。
  3. 位运算符
     1. &:
        1. 运算规则：同1为1，不同为0.
        2. 
     2. |
        1. 运算规则：见1为1，其他为0.
        2. 
     3. ^
        1. 运算规则：相同为0，不同为1。
        2. 
     4. ~
        1. 按位取反是数值的补码进行操作的。也就是说要按位取反首先要找数值的补码然后进行按位取反。（取反操作是包括符号位。）
        2. 
        3. 
     5. 插曲：位移。
        1. 右移：>>
           1. 对于正数向右位移空缺位补0
           2. 对于负数向右位移空缺位补1.
        2. 无符号右移：>>>
           1. 不管你是正数还是负数，右移空缺位一律补0；
        3. 左移：<<
           1. 向左移动
           2. 空缺位补0
  4. 三元运算符
     1. 在表达式里有三个表达式：
     2. 格式：条件表达式？表达式1：表达式2；
     3. 工作原理：
        1. 首先对条件表达式进行执行，若执行的结果为true（真）,那么就执行表达式1,否则执行返回表达式2.
        2. 
        3. 三元运算符求三个数的最大值：
           1. 
        4. 三个数求最大值的优化：
           1. 

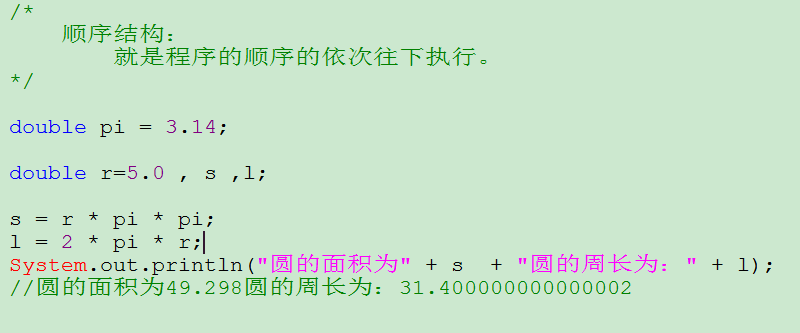
### 数据的录入

1. 从键盘输入数据。
   1. 三步：
   2. 导入包，因为我们要使用某一个类
      1. import java.util.Scanner;
   3. 创建一个对象。
      1. Scanner zhangSan = new Scanner(System.in);
   4. 让这个对象（人）来帮我们做事情，
2. nextInt():表示在键盘里你只能输入int类型数据。
3. nextLine();表示能在键盘里输入字符串类型的数据。
4. 求两个数的和
   1. 
5. 练习:求两个数的最大值。
   1. 

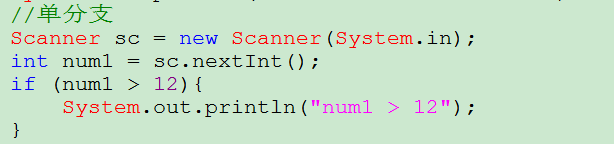
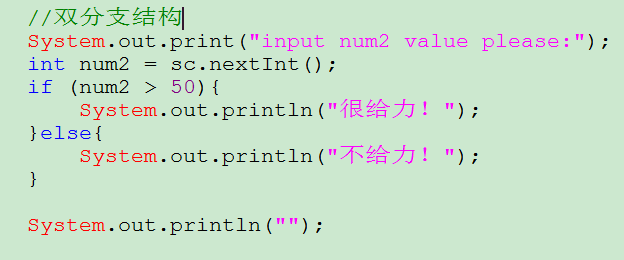
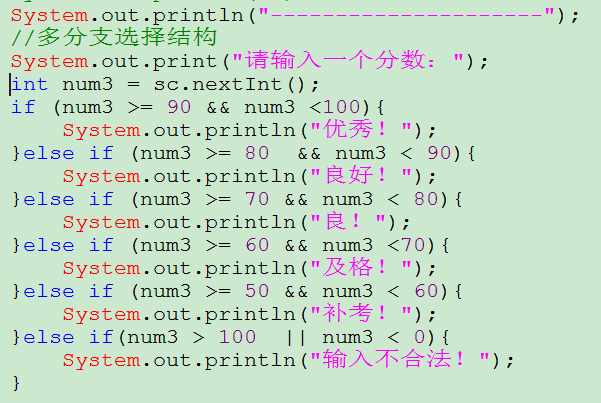
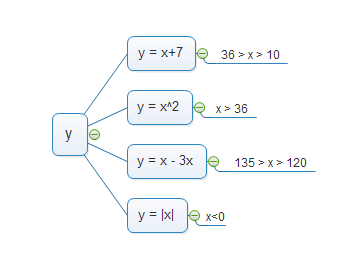
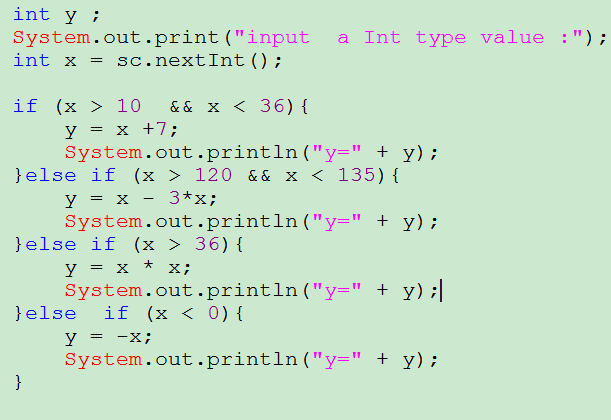
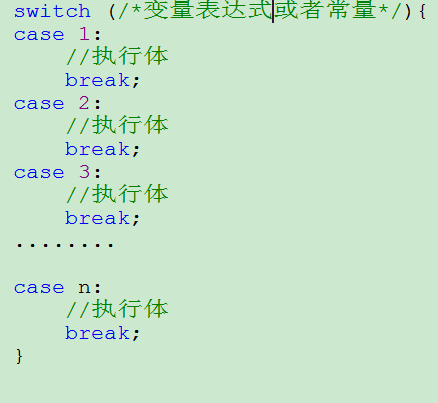
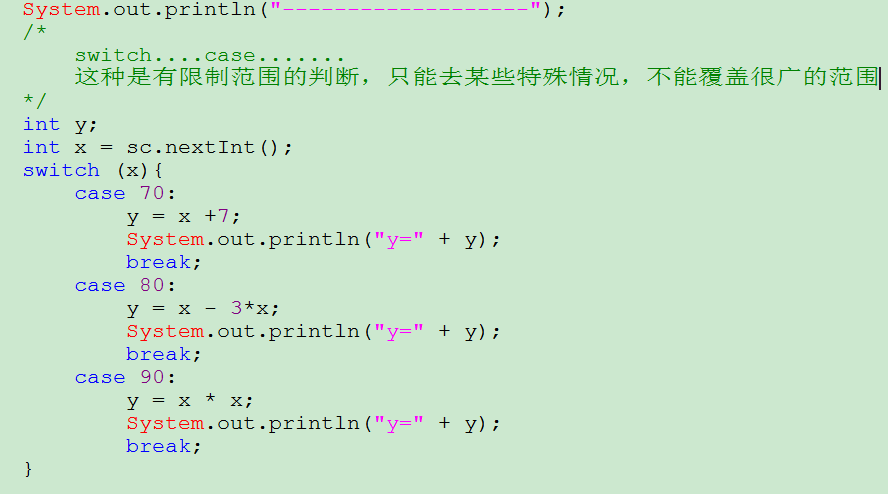
### 程序控制流程

1. 顺序结构
2. 选择结构
3. 循环结构

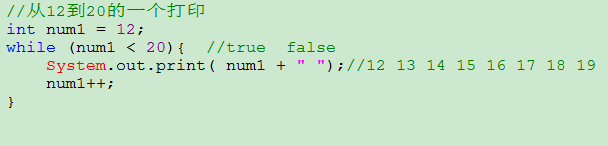
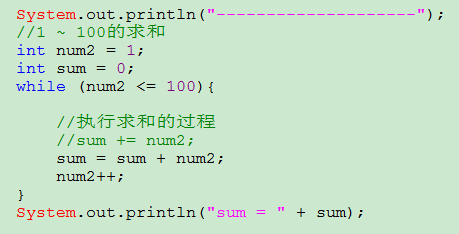
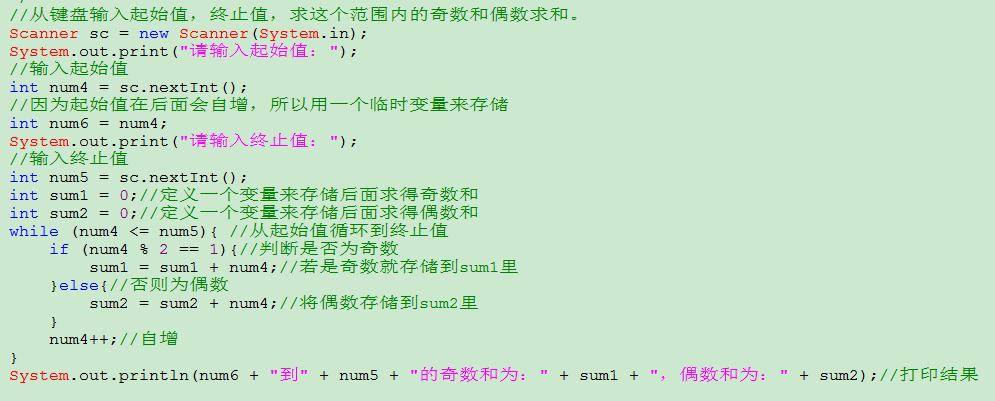
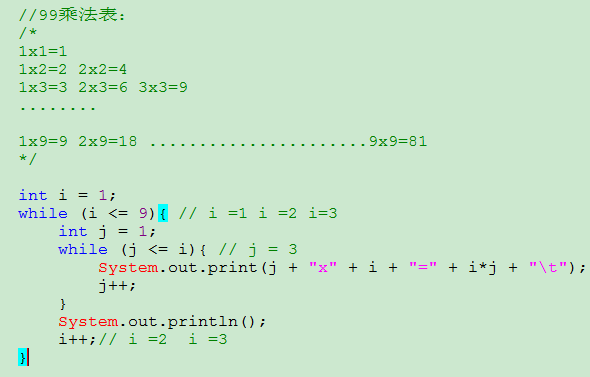
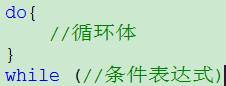
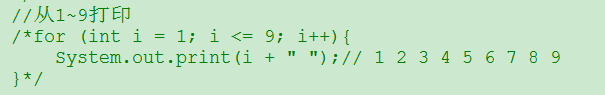
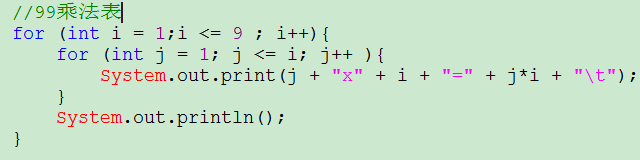
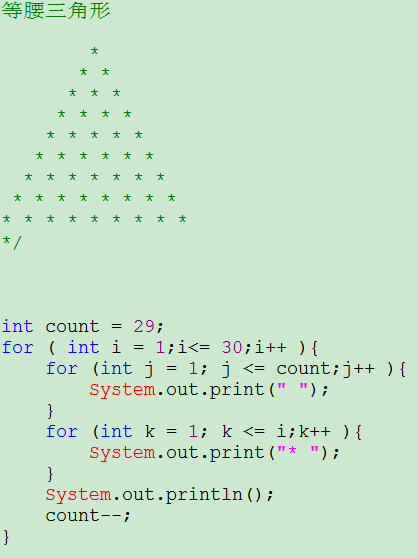
#### 顺序结构：

1. 程序按照顺序的方式往下执行。
2. 

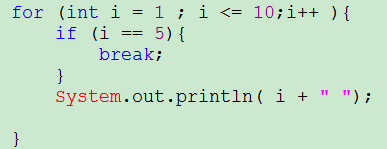
#### 选择结构：

1. If选择结构
   1. 单分支选择结构：
      1. 语句里只有一个if 没有else
      2. 格式:if………
      3. 
   2. 双分支结构：
      1. 语句里不仅有if还有else ，也就是说可能有2种判断
      2. 格式：if …………else…………
      3. 
   3. 多分支结构：
      1. 语句里可能有n种判断
      2. 格式：if ……else if ………else if ………………………………else
      3. 
      4. 练习：
         1. 
         2. 
2. Switch……case
   1. 如果说if判断能对取值范围有很大的扩展，那么switch…case…就是具体的一些判断。
   2. 格式：
   3. 
   4. 注意点：
      1. 在每一个case必须加break;否则会继续执行下去。
3. Switch……Case与if 什么区别
   1. Switch …case适用于某写固定的几种情况
   2. If 判断是适用于一些范围的判断。

#### 循环结构

1. 条件型循环：while
   1. 当型循环:当……时候才执行循环。
      1. 格式：while(//条件){ //循环体}
      2. 执行过程：当条件为true的时候，执行循环体，如果不成立就跳出循环。
      3. 
      4. 注意：死循环状态，需要一个条件来控制程序避免进入死循环装态。
      5. 练习：
      6. 
      7. 
      8. While 的嵌套：
         1. While里面可以在嵌套其他的语句。
         2. 
   2. 直到型循环：
      1. 先做一次循环，然后在进行判断。
      2. 格式：
      3. 相同：注意死循环。
      4. 练习：将while里的用do…while做一遍。
2. 次数型循环
   1. For循环：
   2. 格式：for (//初始化条件 ;//条件表达式 ; //参数更新序列 ){ //循环体}
   3. 
   4. 
3. 打印等腰三角形：
   1. 

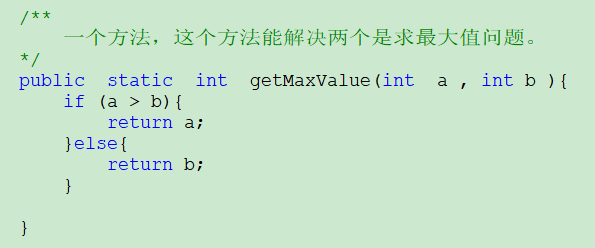
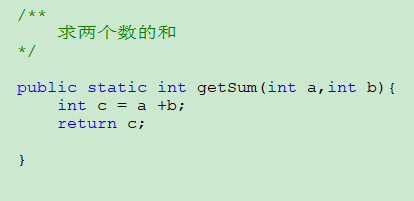
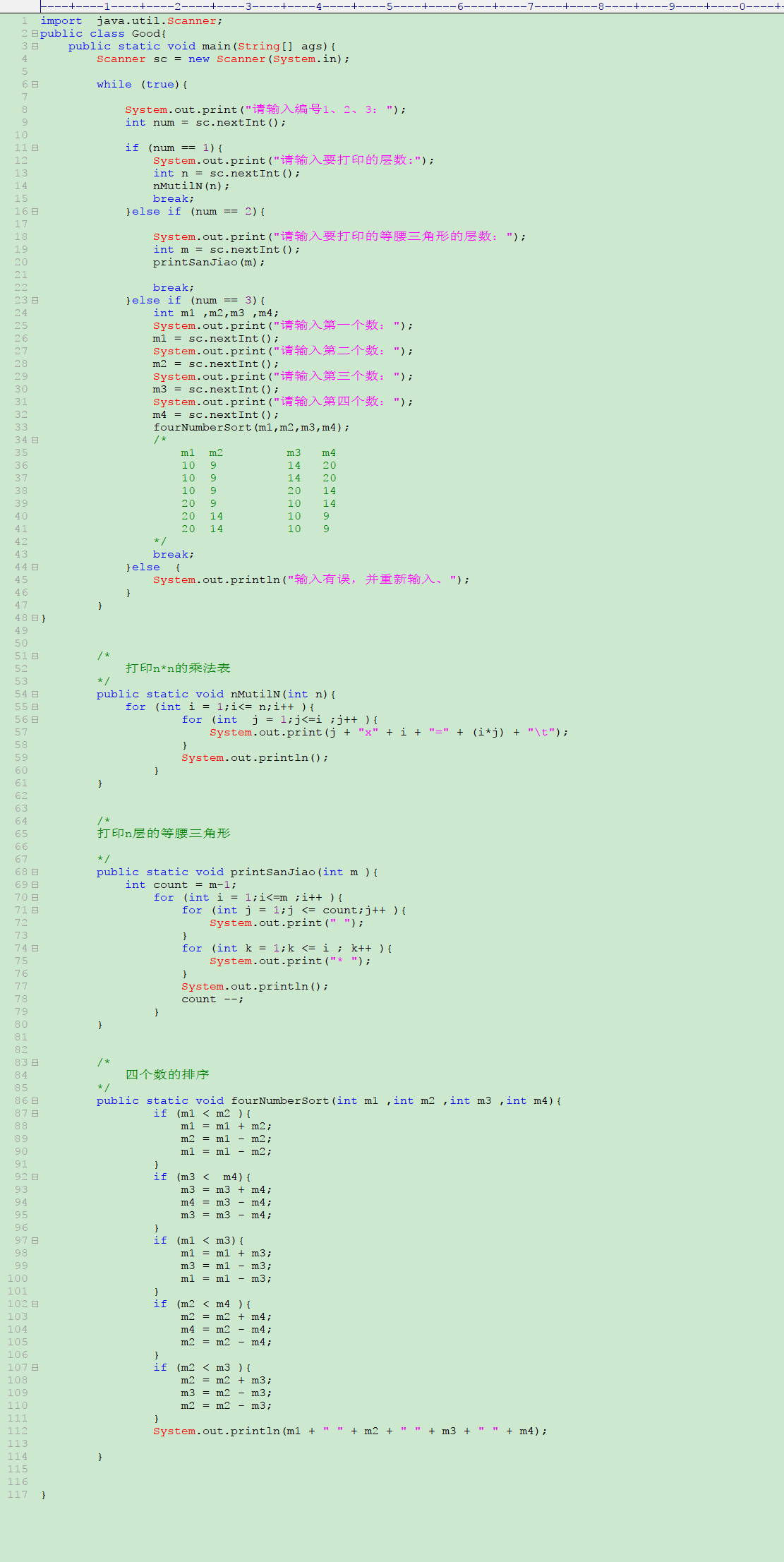
#### 小插曲：

1. 程序控制语句：
   1. break
      1. 在switch 或者loop循环中，用于跳出switch或者循环。
      2. 
   2. continue
      1. 结束本次循环，进入下一次循环。

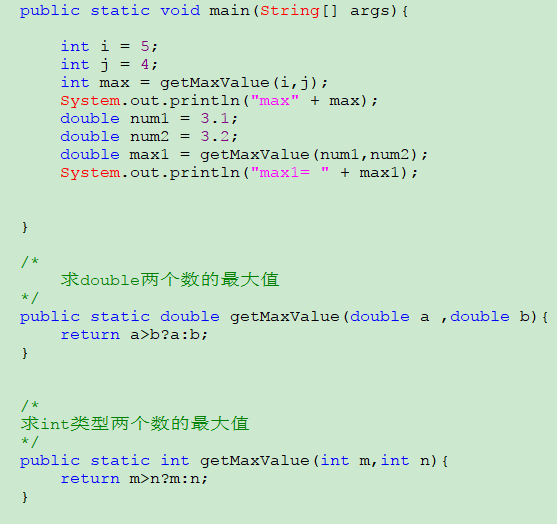
### 方法

1. 是一个代码块，有固定的组织规则，实际上描述的是 一个行为、一种功能，提高代码块的复用性，其决定了一 个对象能够接收什么样的消息做出什么样的反应
   1. 简单来讲：就是解决一类问题的代码集合。
   2. 方法的好处：提高了代码的复用性。
   3. 方法的格式：

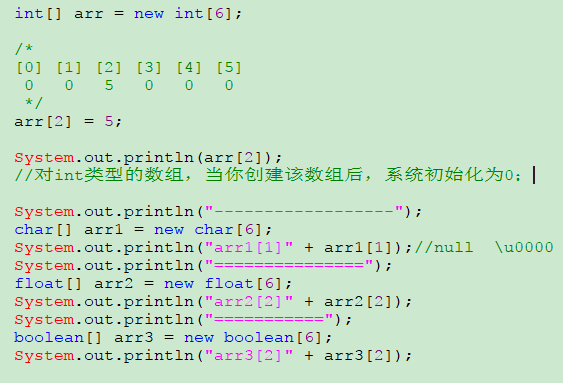
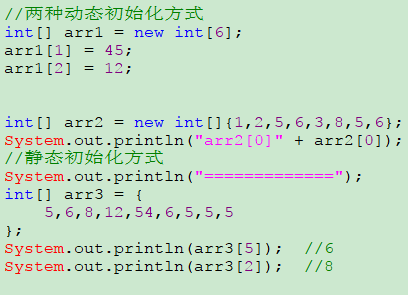
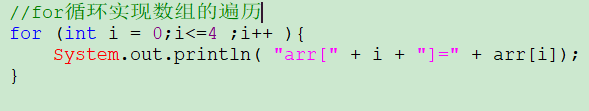
|  |
| --- |
| 方法的格式：  [修饰符] [返回值类型] [方法的名称]（传入的参数列表）{  //具体处理的代码块  return [返回值];  }  修饰符：public private protect  返回值类型：标识返回值的类型，void(如果写void，那么就不需要return 语句了。)  方法的名称：首个单词小写，后面的单词首字母大写。  传入的参数：可以不传入，也可以传入。 |

* + 1. 方法的定义和使用：
       1. 
       2. 练习：
          1. 两个数的求和：
          2. 
          3. 代码的优化：
          4. 

#### 方法的重载：

1. 什么是重载（OverLoad）:方法的名称，但是传入的参数个数或者类型不一样，那么这几个函数之间就是重载关系。在同一个类中。
2. 

## 数组(重要)

1. 什么是数组？
   1. 简单来讲就是能不能用一个集中方式来存储值，因为原来的变量来存储值，如果值比较多，那个变量就显得力不从心，。
   2. 由相同数据类型的元素组成的集合, 它是用来存 储同种数据类型的多个值,也可以看作是一个容器
   3. 存储什么类型的数据？
      1. 11种。
      2. 基本数据类型
      3. 引用数据类型。
   4. 数组的定义格式：
      1. 
      2. 
   5. 数组的特点：
      1. 数组只能存储某一个类型的数据
      2. 数组里面的数据是有顺序排列的。（通过角标，索引来体现）
      3. 数组中的元素是通过数组名称和角标来确定的。
      4. 注意：ArrayIndexOutOfBoundsException;
         1. 找不到这个元素。所需的角标已经超出了范围。
      5. 数组的默认初始化：
         1. Int的数组，系统的默认初始化为0;
         2. Char的数组，系统默认初始化为\u0000;（空字符）
         3. Boolean的数组，系统会默认初始化为false,
         4. 浮点类型的默认初始化为0.0
         5. 
         6. 对于引用数据类型的默认初始化为null;
      6. 数组的两种初始化方式：
         1. 动态：
            1. Int[] arr = new int[6];
            2. Int [] arr= new int[]{1,3,54,6,55,4,4};
         2. 静态
            1. Int[] arr = {5,5,6,6,6};
            2. 在造房间的时候就把人赛进去。
         3. 
         4. 数组的遍历：
            1. 
         5. 数组的属性：
            1. Length:获取数组的长度。
         6. 数组的内存解析：
            1. Int[] arr = new int[6];
            2. 内存解析：

栈

存储局部变量和一些方法的引用

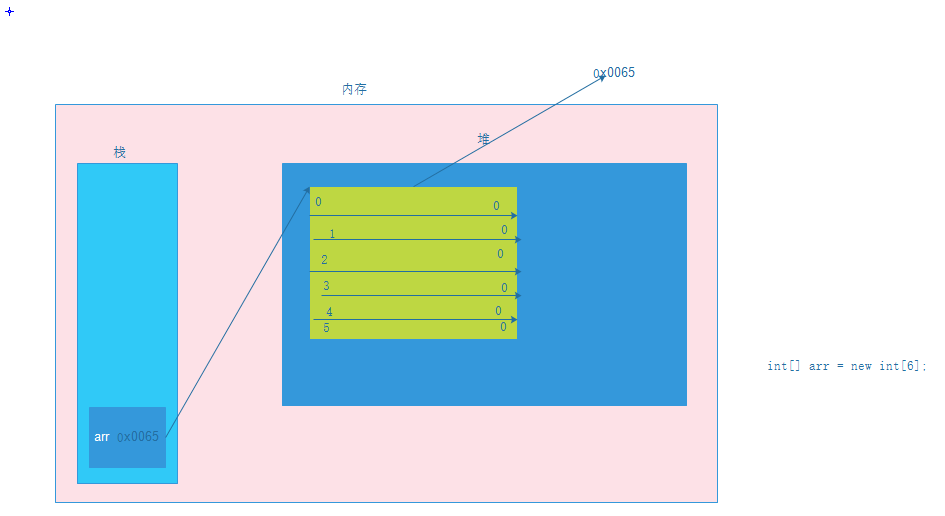
堆

存放的是一些对象的实体。

方法区

本地方法区

寄存器



空指针异常：

NullPointerException.

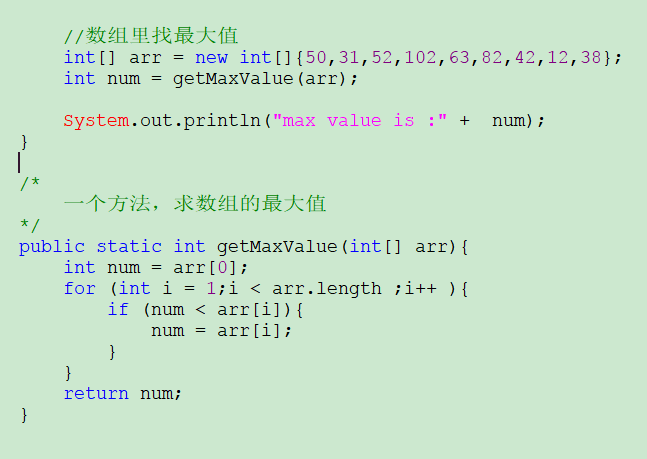
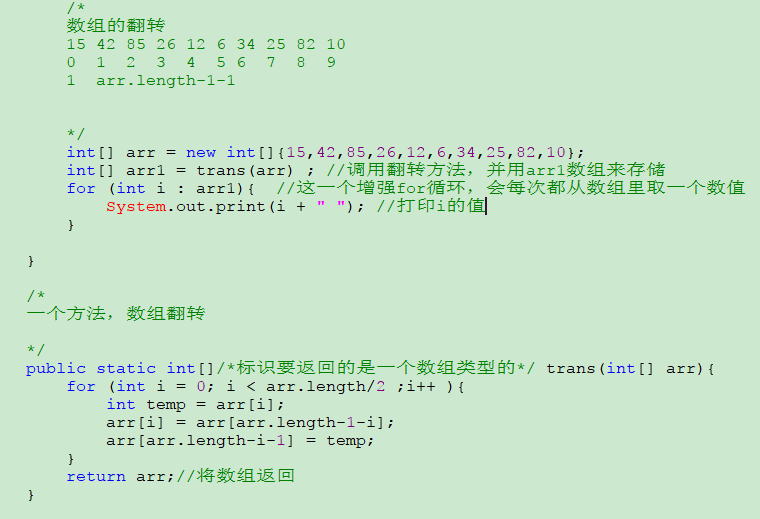
栈里面的引用不指向堆内存中任意实体对象时，强制操作会报空指针异常。

垃圾：

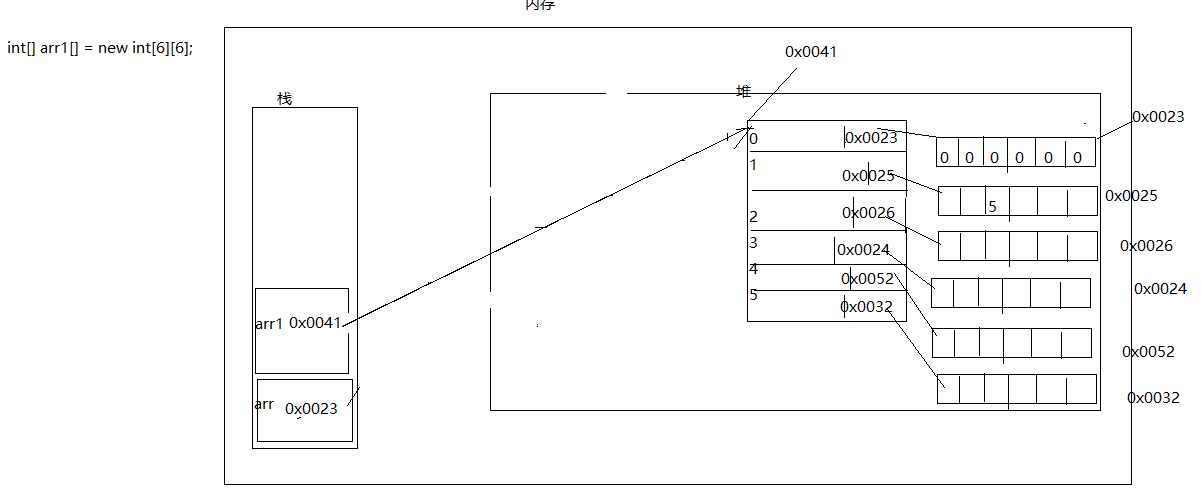
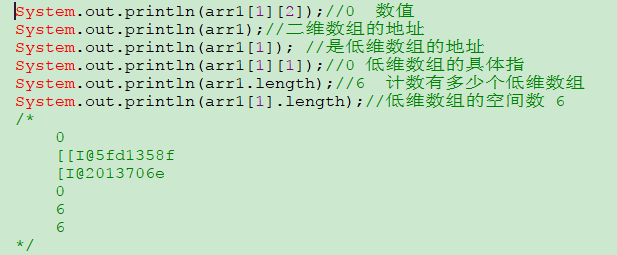
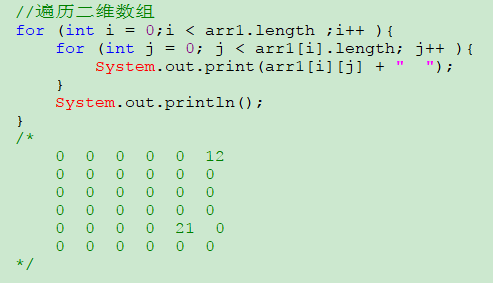
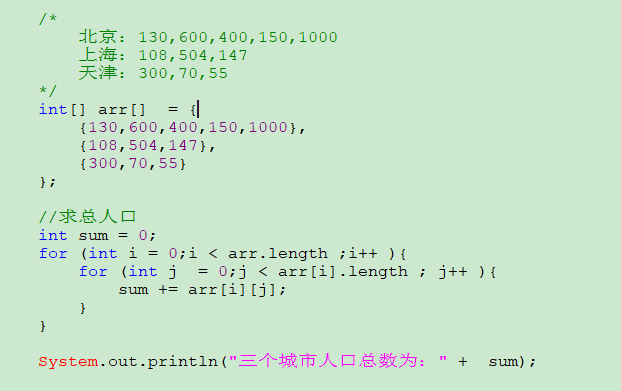
堆里面的实体对象没有引用指向他，这个时候这个对象就是一个垃圾，那么java的一个程序GC处理器，会不定时来处理堆里面的垃圾。

垃圾回收机制：

java语言对C++语言的内存进行了很好的优化，在堆里若有垃圾会有个GC处理器不定时来回收这些垃圾。避免了程序员手动去清除造成内存崩溃。

1. 数组的一道开胃小菜：
   1. 数组的求最大值：
      1. 
   2. 比较两个一维数组是否相同？
      1. 
   3. 数组的翻转：
      1. 

### 二维数组

1. 其实就是高维数组里面存储低维数组的地址。
2. 所以高维数组内存储不是数值而是地址（十六进制。）
3. 格式：
4. 内存：
   1. 
5. 三种属性：
   1. System.out.println(arr1);
   2. System.out.println(arr1[1]);
   3. System.out.println(arr1[1][1]);
   4. System.out.println(arr1.length);
   5. System.out.println(arr1[1].length);
   6. 
6. 遍历二维数组：
   1. 
   2. 练习：三个地方人口，求和：
      1. 
   3. Int[] arr[] = new int[6][6];
   4. Int[] arr1[] = new int[][3];
      1. 直接从低维数组分配是不合法的。
   5. Int[] arr2[] = new int[2][];
      1. 可以。

## 面向对象：

课程内容：

1. 什么是面向对象?
2. 类与对象
3. 成员变量与与局部变量
4. 成员方法及其参数
5. this对象引用
6. 封装
7. 构造方法(构造器)
8. static关键字、静态变量与成员变量
9. 系统类的使用
10. 案例

### 什么是面向对象？

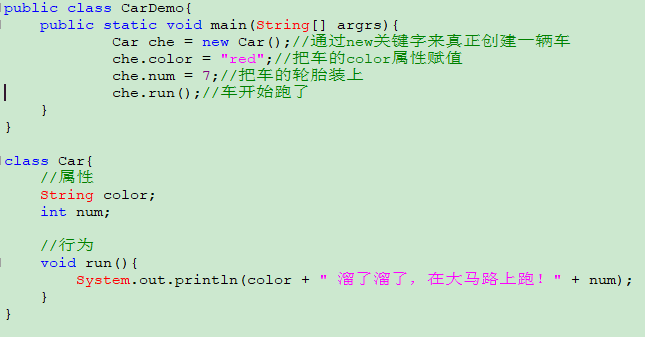
什么是面向对象？

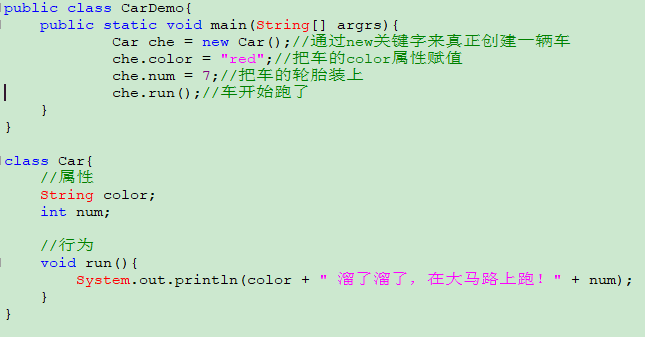
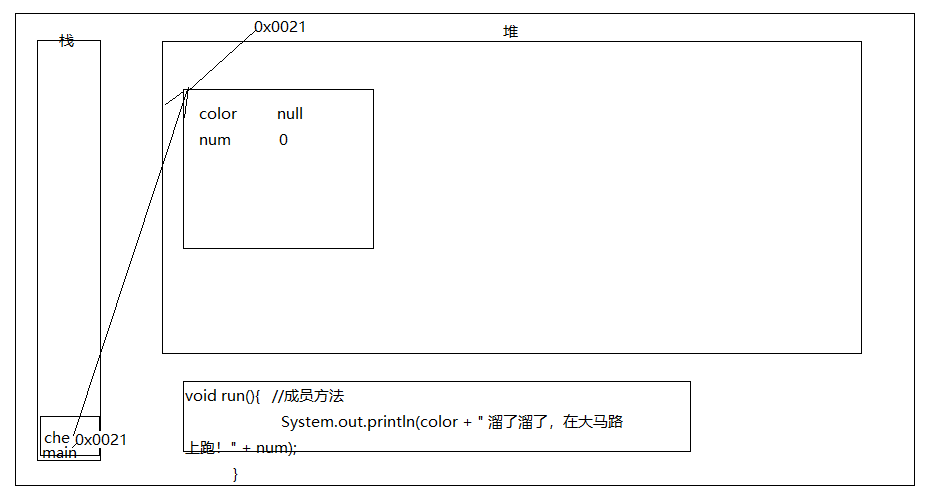
1. 是一个编程思想。
2. 比面向过程好在什么地方？
   1. 将数据和函数封装在一个单元中。
   2. 更接近与人们思考习惯。
3. 那么基于面向对象(OO)
   1. OOA(objective orentied analysis)面向对象分析
   2. OOD(objective orentied design)面向对象设计
   3. OOP(objective orentied program)面向对象编程
4. OO：OO是一种方法，一种思想，同时又是一种技术 ，用客观的自然语言描述现实世界，使分析、设计和实现系统的方法同认识客观世界的过程尽可 能一致。
5. 面向对象的特点：
   1. 抽象：
   2. 继承：提高了代码扩展性。复用性。
   3. 封装：提高代码的安全性。
   4. 多态：基于继承
6. 对象
   1. 就是协调数据存储以及作用于数据之上操作的独立实体或者实例

万物皆对象

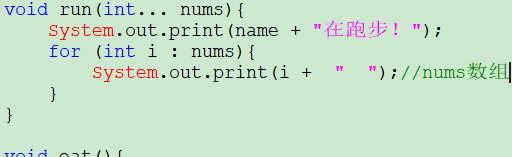
### 类与对象

1. 什么是类？
   1. 类是构造对象的模板或蓝图或者图纸，是具体事物的一个 抽象，决定了某一类对象的外观与行为.
   2. 申明格式：
      1. class 类名{
         1. //封装的代码
      2. }



1. 成员方法：也叫做行为
2. 成员变量：也叫做属性
3. 成员方法和成员变量的调用：
   1. 
   2. 对于成员变量
      1. 对象名称.成员变量名
   3. 对于成员方法
      1. 对象名称.方法名称
   4. 对象内存：
      1. 
4. 成员变量可以定义的范围：
   1. 11类型都可以。
   2. 作用域：
      1. 在类里，方法外
5. 局部变量
   1. 作用域：
      1. 在方法里。

### 变长参数：

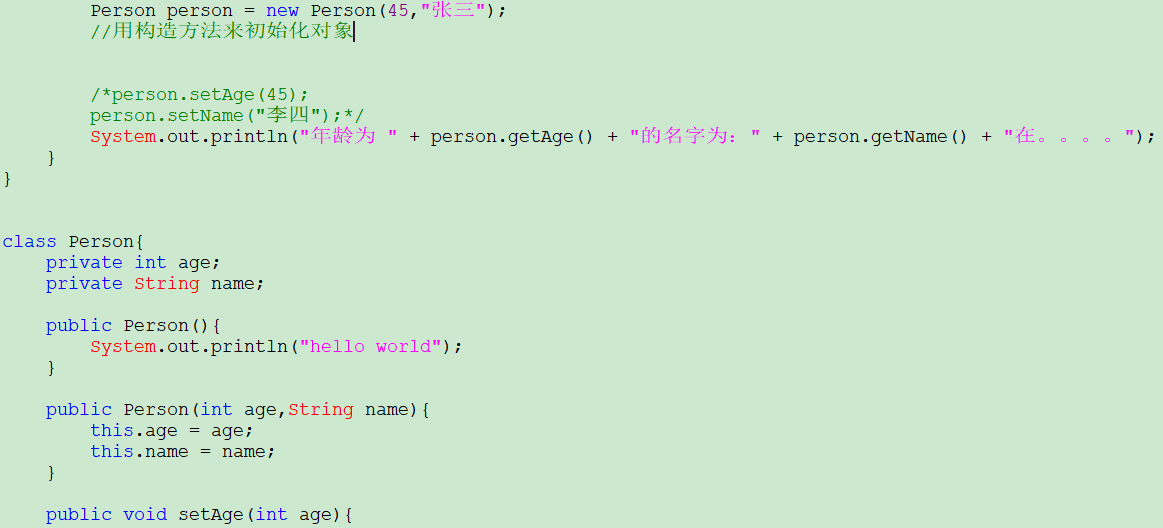
1. 可以变化的来接受你传入的参数个数
2. 格式：[参数的类型]... nums
3. 原理：他会将你传入的参数分割开来并存到**数组**里去。
4. 
5. 注意：
   1. 一个方法最多只能有一个变长参数
   2. 方法参数大于两个时，变长参数放在最后

### 封装：

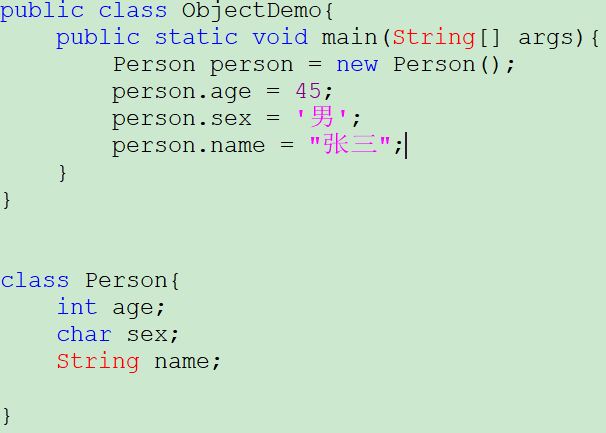
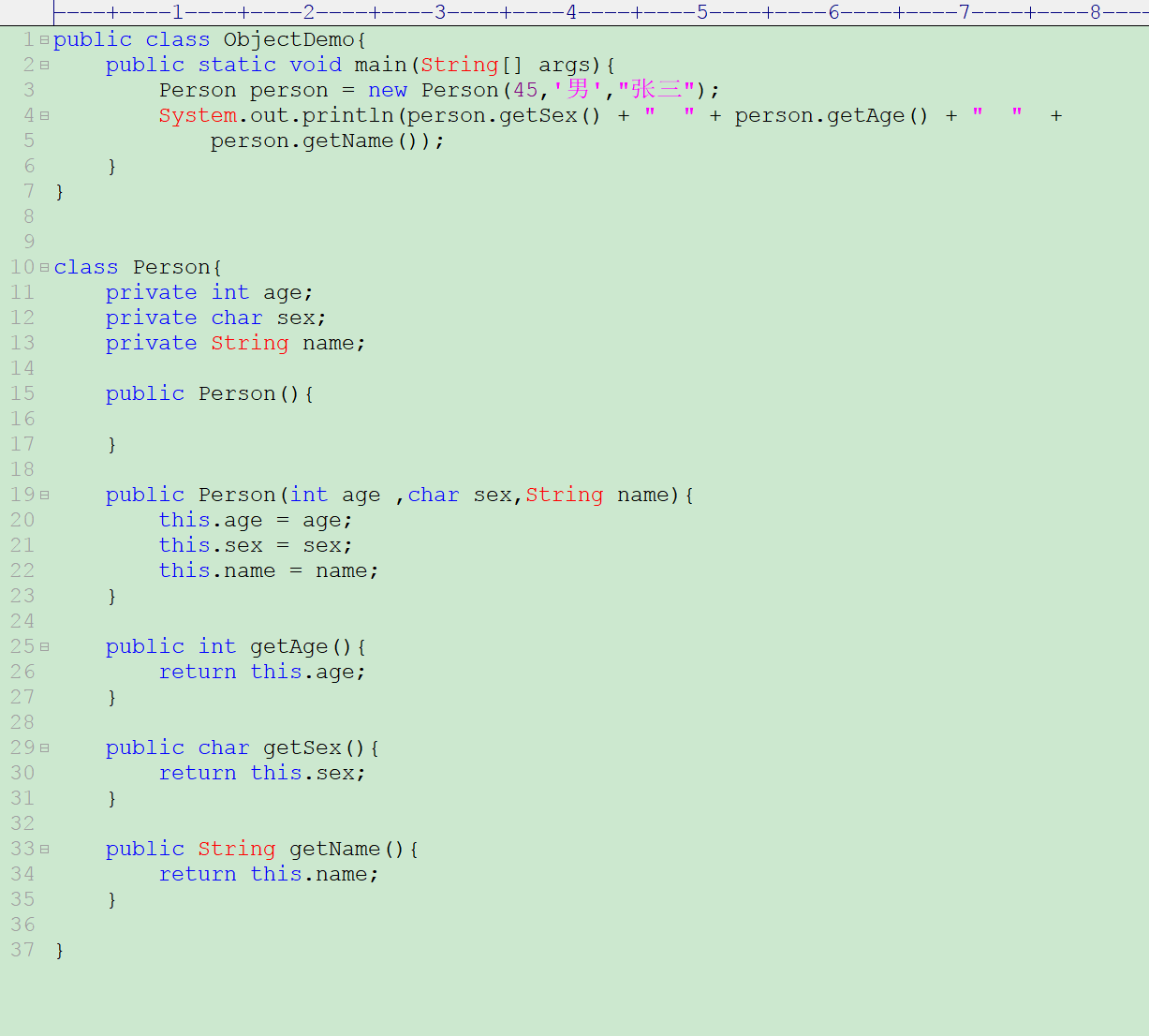
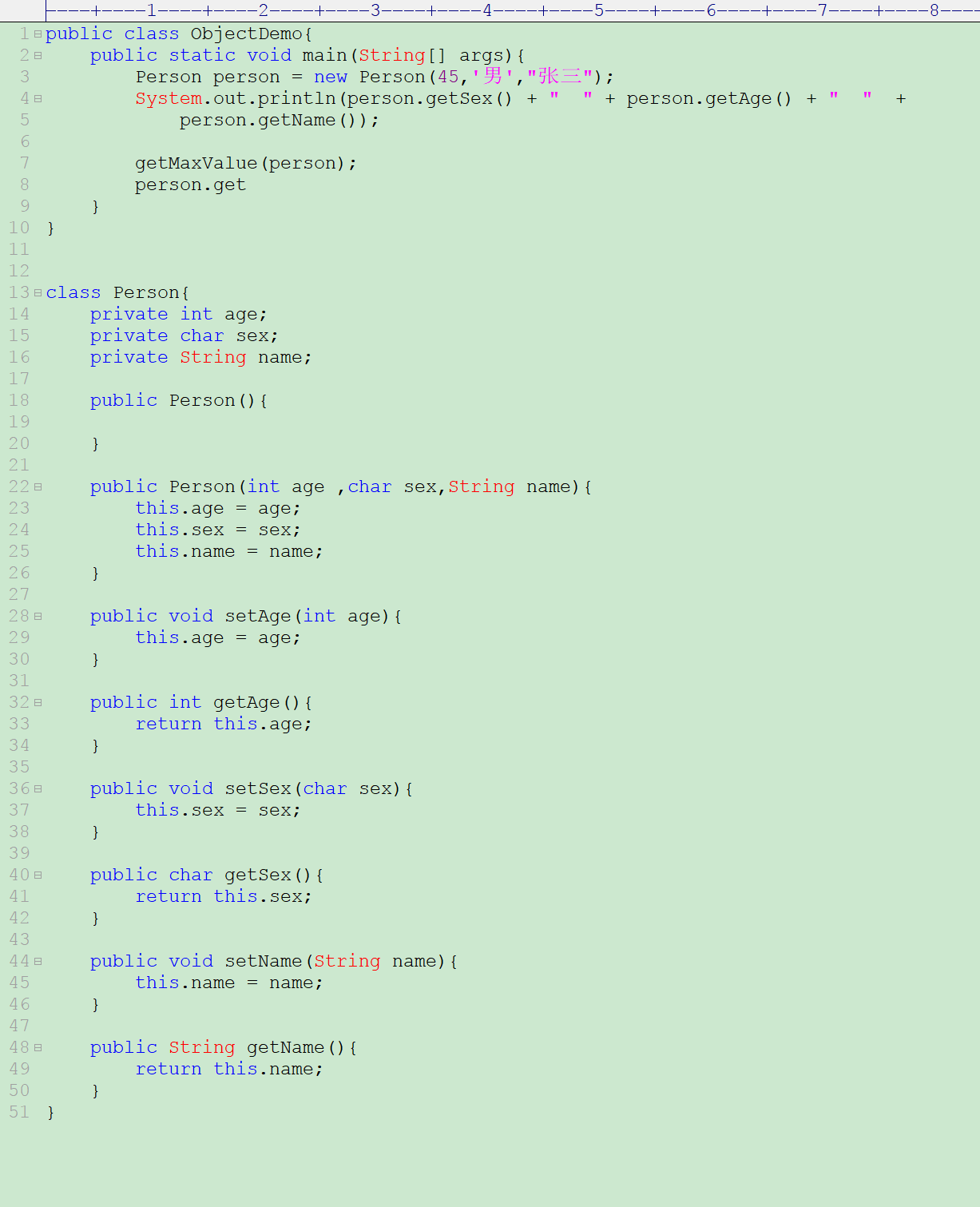
什么是封装：

1. 将类内部的实现细节或者属性用private（私有）关键字来进行封装。
2. 是面向对象语言重要的特性之一，是指隐藏类的属性和实现细节，仅仅对外公开访问接口.
3. 好处
   1. 防止误操作
   2. 安全
   3. 提高软件重用性
   4. 降低各个子系统、模块、组件间的耦合关系
   5. 方便给使用者使用
4. 封装的原则：
   1. 隐藏所需要的属性
   2. 尽可能隐藏内部实现细节，只保留简单的对外访问的方法

### 构造方法：

1. 在类里面有一个与类名一样并且没有返回值的的方法，（构造方法）
2. 作用：
   1. 用来初始化对象的。
   2. 
3. 注意点：
   1. 如果我们没有在类里显示写构造方法，那么系统在执行的时候会自己生成一个与类名一样的构造函数
   2. 如果我们显示的写了构造方法，那么系统就不会在提供默认的构造方法。

### 对于对象的初始化得三种方式：

1. 直接拿对象名称.成员变量 来进行赋值。
   1. 
2. 用private关键字将属性或者成员变量进行私有化，然后对外提供公共访问接口，来间接给参数赋值。（javabean java豆子：封装数据。）
   1. 
3. 通过构造方法来将对象进行初始化
   1. 
4. 那种用比较多：
   1. 2和3结合用比较多。
   2. 

### This：

1. 一个指针：代表当前对象。
2. 放在在进栈的时候会有一个默认的this指针，来代表当前所调用的对象。

### 匿名对象：

1. 没有名字的对象。
2. 没有在程序中指定具体名称
3. 可以将匿名对象作为实际参数进行传递
4. 当对对象的方法只调用一次时，可以用匿名对 象以简化的方式来完成