笔记

[笔记 1](#_Toc26126582)

[JVM的划分 5](#_Toc26126583)

[Return后只能写一个值 5](#_Toc26126584)

[Void: 无返回值 5](#_Toc26126585)

[方法重载注意事项 5](#_Toc26126586)

[导包可以使用通配符把该目录下的类全部导入 5](#_Toc26126587)

[实参与形参 6](#_Toc26126588)

[方法参数是基本数据类型 6](#_Toc26126589)

[方法参数是引用数据类型 7](#_Toc26126590)

[方法的格式 7](#_Toc26126591)

[重载（overload） 7](#_Toc26126592)

[重载的几种测试 7](#_Toc26126593)

[引用数据类型（类，数组，接口） 10](#_Toc26126594)

[创建引用数据类型的格式 数据类型 变量名 = new 数据类型（）； 11](#_Toc26126595)

[New 实例化对象 11](#_Toc26126596)

[参数列表为引用数据类型时 11](#_Toc26126597)

[形参内往往为接口，配合下面的格式，实现向上转型 11](#_Toc26126598)

[形参里的引用类型 引用名 = 实参（）；（一般为向上转型） 11](#_Toc26126599)

[实参为 new 引用类型 形参里的引用类型 引用名 = new 引用类型 向上转型 11](#_Toc26126600)

[实参为之前已实例化的对象 形参里的引用类型 引用名 = new 之前实例化对象的类型 11](#_Toc26126601)

[注意;实参必须是可以实例化对象的类型且是形参的实现类，原因在向上转型部分 11](#_Toc26126602)

[变量名保存的是一个内存地址 11](#_Toc26126603)

[自定义类 11](#_Toc26126604)

[定义类中属性时修饰符可以不写， 11](#_Toc26126605)

[Cmd中的运行谁编译谁 11](#_Toc26126606)

[一个引用变量的内存图 12](#_Toc26126607)

[两个引用类型变量内存图 12](#_Toc26126608)

[ArrayList 13](#_Toc26126609)

[格式 13](#_Toc26126610)

[ArrayList中的方法 13](#_Toc26126611)

[注意事项 14](#_Toc26126612)

[百位十位个位的数如何求 14](#_Toc26126613)

[Char 与 int 14](#_Toc26126614)

[Char转为int类型取值0-65535 14](#_Toc26126615)

[Char和short均为两字节取值不同的原因 14](#_Toc26126616)

[冒泡排序问题汇总 15](#_Toc26126617)

[1.注意存在j+1，注意溢出问题 15](#_Toc26126618)

[2.冒泡方法中的i小于数组长度-1，所以直接输出数组【i】时会少一个元素 15](#_Toc26126619)

[面向对象的特性 继承 封装 多态 15](#_Toc26126620)

[成员变量和局部变量区别 16](#_Toc26126621)

[定义的位置不同 16](#_Toc26126622)

[在内存中的位置不同 16](#_Toc26126623)

[\*生命周期不同 16](#_Toc26126624)

[\*内存位置不同 16](#_Toc26126625)

[\*默认值不同 16](#_Toc26126626)

[局部变量生命周期 16](#_Toc26126627)

[Private 只修饰成员变量，不修饰局部变量； 17](#_Toc26126628)

[继承时 子类中有则优先自己 没有则使用父类的变量或方法 17](#_Toc26126629)

[方法“就近” 方法内有不使用则本类成员变量 17](#_Toc26126630)

[Super调用父类 17](#_Toc26126631)

[This 调用自己本类成员 17](#_Toc26126632)

[重写override 17](#_Toc26126633)

[子类方法声明应该与父类相同 17](#_Toc26126634)

[重写父类方法 子类权限应该不小于父类 17](#_Toc26126635)

[父类私有方法(private)子类不能重写 17](#_Toc26126636)

[抽象类 18](#_Toc26126637)

[抽象方法： 方法只有声明部分，没有方法体（大括号） 用abstract修饰 18](#_Toc26126638)

[抽象类： 包含抽象方法的类，一定是抽象类，使用 abstract 修饰的类 18](#_Toc26126639)

[抽象方法所在类一定是抽象类： 18](#_Toc26126640)

[抽象类与抽象方法都必须使用 abstract来修饰 18](#_Toc26126641)

[Abstract不能和 private final static一起使用 18](#_Toc26126642)

[抽象类不能直接实例化对象（不能 new 抽象类） 18](#_Toc26126643)

[抽象类中可以有抽象方法，也可以不写抽象方法 18](#_Toc26126644)

[若子类只重写了父类部分抽象方法，子类仍为抽象类 18](#_Toc26126645)

[继承（extends）弊端：耦合性高 18](#_Toc26126646)

[接口 19](#_Toc26126647)

[接口内只有抽象方法 19](#_Toc26126648)

[格式：Public abstract 返回值类型 方法名（参数列表）; 19](#_Toc26126649)

[接口定义中Public abstract可以不写或选择性书写，默认为全有，实现时必须写出该有的； 19](#_Toc26126650)

[接口中的方法均为public，接口内不写权限也默认为public 19](#_Toc26126651)

[接口内成员变量必须定义为常量，只有常量没有变量 19](#_Toc26126652)

[格式：public static final 数据类型 变量名 = 值； 19](#_Toc26126653)

[Static：可以直接被 类名. 的形式调用 19](#_Toc26126654)

[Final ：变量值不可变，不可重新赋值，即使第二次赋值与第一次一样 19](#_Toc26126655)

[接口定义中public static final可以不写或选择性书写，默认为全有 19](#_Toc26126656)

[实现 implements 19](#_Toc26126657)

[实现类可以为抽象类 19](#_Toc26126658)

[接口中多实现（public void c implements a,b） 19](#_Toc26126659)

[可以多实现的原因 20](#_Toc26126660)

[多继承的安全隐患 20](#_Toc26126661)

[多实现中两接口内可以有重名方法，但若是两方法返回值不同，子类实现方法会报错（重复定义），两种返回值都实现或只实现一种全都报错 20](#_Toc26126662)

[继承的同时多实现 （extends b implements c，d（）{}） 21](#_Toc26126663)

[接口的多继承 21](#_Toc26126664)

[接口之间有多继承，类没有 21](#_Toc26126665)

[接口的好处 21](#_Toc26126666)

[1.接口的出现扩展了功能。 21](#_Toc26126667)

[2.接口其实就是暴漏出来的规则。 21](#_Toc26126668)

[3.接口的出现降低了耦合性，即设备与设备之间实现了解耦。 21](#_Toc26126669)

[接口和抽象类的区别： 21](#_Toc26126670)

[相同点: 21](#_Toc26126671)

[区别: 21](#_Toc26126672)

[二者的选用: 22](#_Toc26126673)

[多态 22](#_Toc26126674)

[需要有继承或实现关系 22](#_Toc26126675)

[父类 变量名 = new 子类（） 22](#_Toc26126676)

[成员变量 编译运行全看父类 22](#_Toc26126677)

[成员方法 编译看父类，运行看子类 22](#_Toc26126678)

[Instanceof ：比较对象是否是某引用类型 23](#_Toc26126679)

[返回类型：boolean 23](#_Toc26126680)

[格式：Boolean b = 变量名 instanceof 引用类型； 23](#_Toc26126681)

[注意：需要变量所属类需要和instanceof后的引用类型存在继承或实现关系，否则报错 23](#_Toc26126682)

[多态中的转型 23](#_Toc26126683)

[向上转型 父类 变量名 = new 子类（） 23](#_Toc26126684)

[优点：可以调用父类子类公共方法 23](#_Toc26126685)

[缺点：无法调用子类特有方法（调用父类没有的方法报错） 23](#_Toc26126686)

[向下转型 将向上转型的对象强制类型转换成子类 23](#_Toc26126687)

[子类 新变量名 = （子类类型）原变量名 23](#_Toc26126688)

[优点：可以调用子类特有方法 23](#_Toc26126689)

[注意：应转回原本子类，转成其他类型，编译无错，运行报类型转换错误； 23](#_Toc26126690)

[注意：可使用if与instanceof避免出错 23](#_Toc26126691)

# JVM的划分

# Return后只能写一个值

# Void: 无返回值

调用方法的时候,返回值是void, 不能写在输出语句中。

错误: 此处不允许使用 '空' 类型

# 抽象方法没有方法主体（大括号）

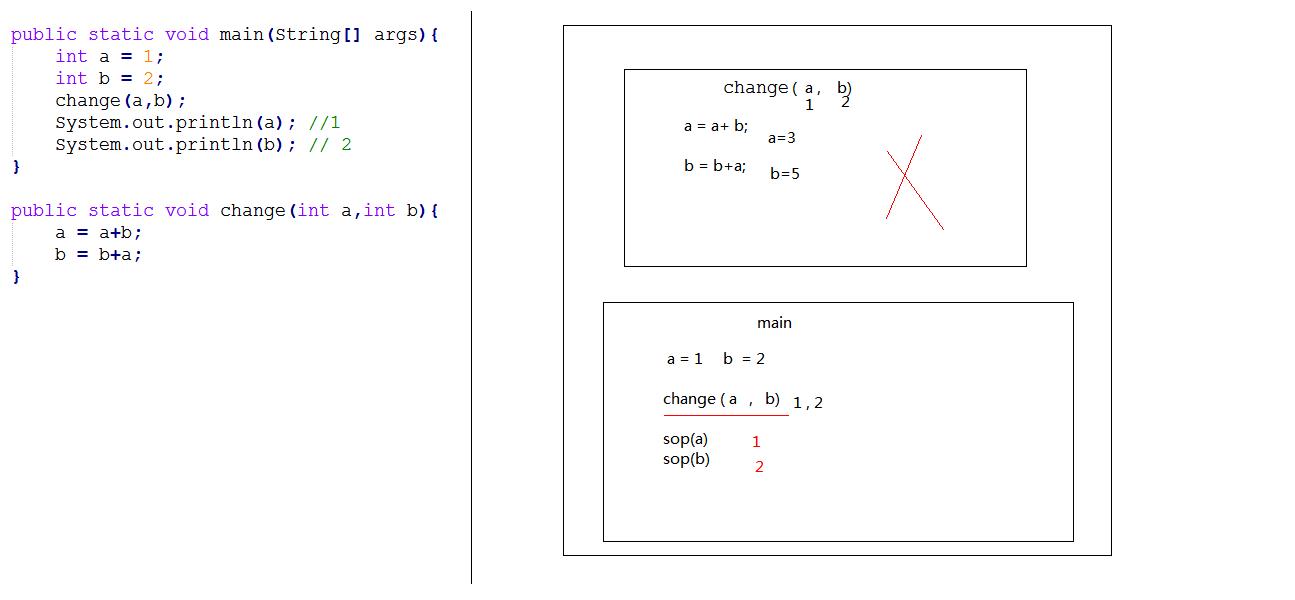
# 导包可以使用通配符把该目录下的类全部导入

如java.util.\*

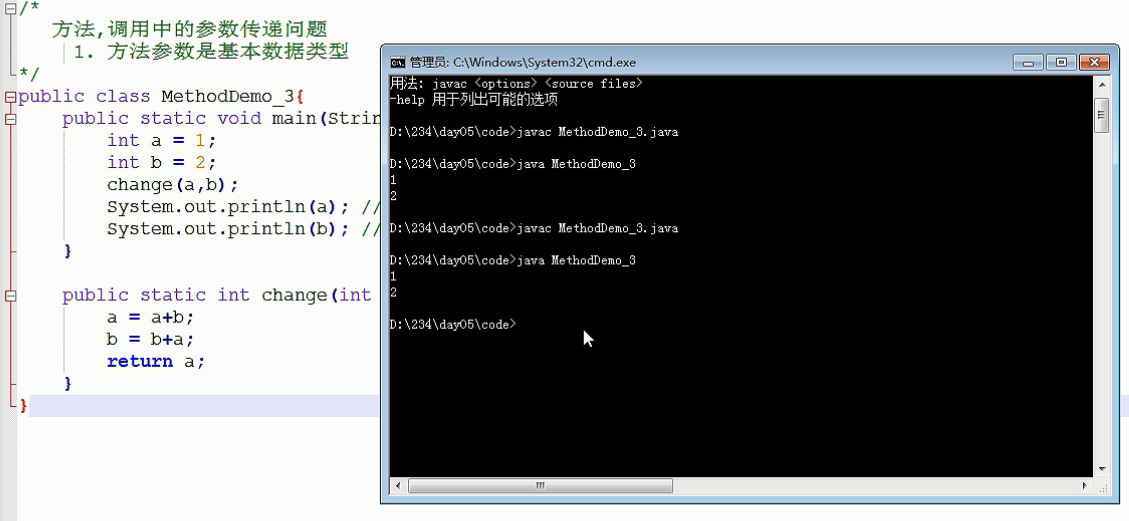
# 实参与形参

## 方法参数是基本数据类型

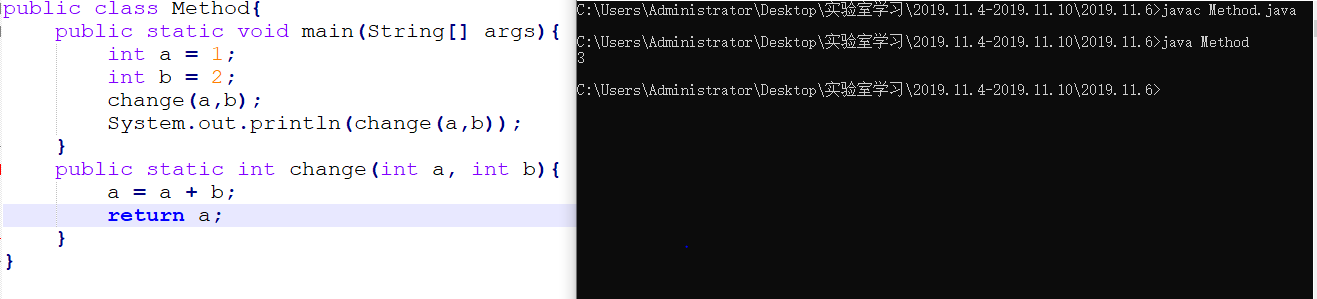
方法参数是基本类型时，传递的是值。/\*返回后不改变主方法变量的值，即使变量名一样。除非将方法返回值 赋值给主方法变量 且下图也无返回值 \*/



Tip 只传递给change方法数值，方法内的变量ab运算后随着方法从内存一起释放，故完全不影响main方法ab变量



Tip 虽然有返回a 但是返回的也是一个数值，不是变量。即change（a，b）== 3，输出change（a，b），值为3. 如果将返回值赋值给变量，即写成a = change（a，b），输出a为3



## 方法参数是引用数据类型

方法参数是引用类型时，传递的是内存地址值。/\*数组名为地址，调用的方法凭借地址对数组进行更改 （合租）\*/

# 方法的格式

## 修饰符 返回值类型 方法名（参数列表 /\*逗号隔开\*/）{}

方法名首字母小写其后首字母大写

未调用时 方法的形参没有具体值

# 重载（overload）

## 方法重载注意事项

### a: 参数列表必须不同

### b: 重载和参数变量名无关

### c: 重载和返回值类型无关

### d: 重载和修饰符无关

在同一个类中，允许方法名相同，只需参数列表不同（包括参数个数，数据类型，顺序）只看参数列表

## 构造器也有重载

## 重载的几种测试

结果摘要

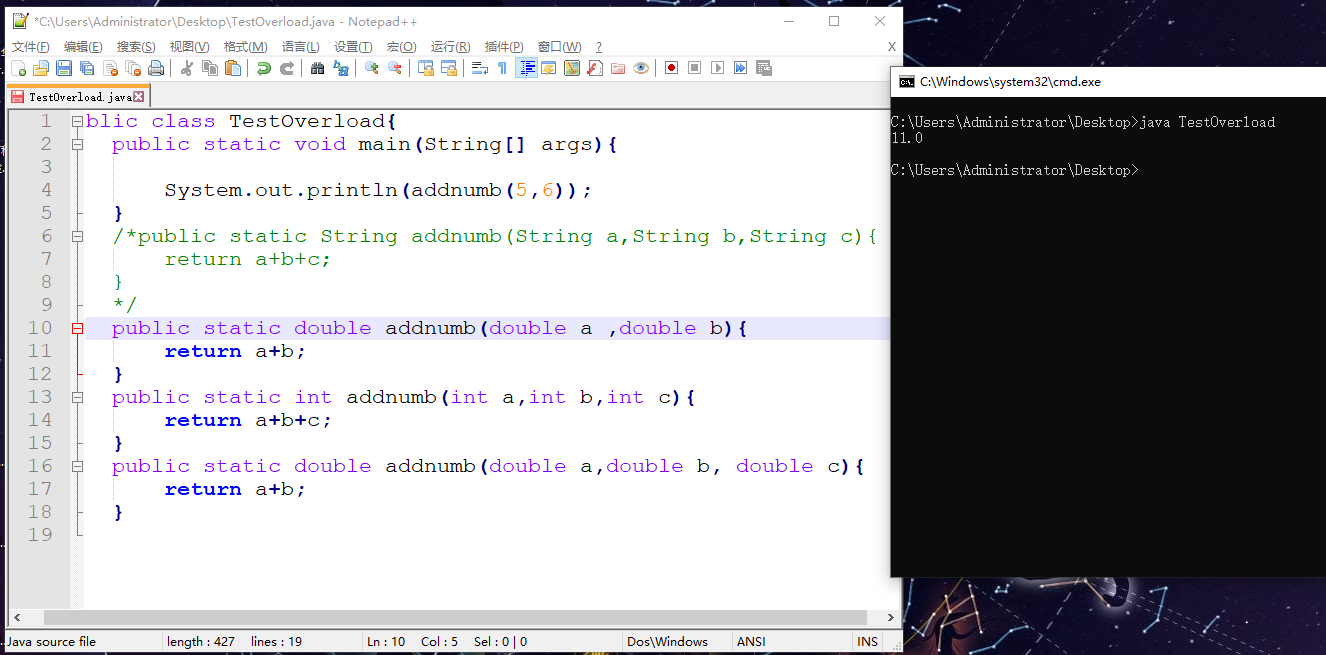
1.优先以参数个数为基准匹配方法

2.整型默认为int， int型实参可以自动转为double传给形参

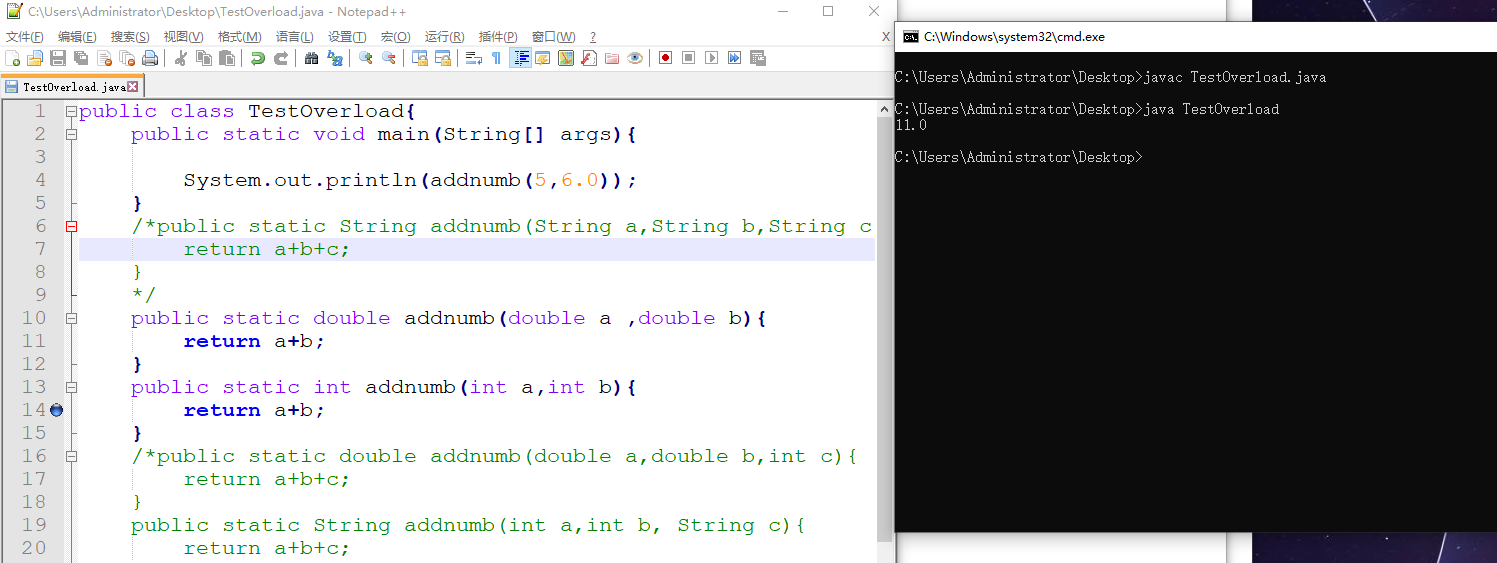
3. int型无法转为String，不兼容

4. char型可以取表值看作int计算

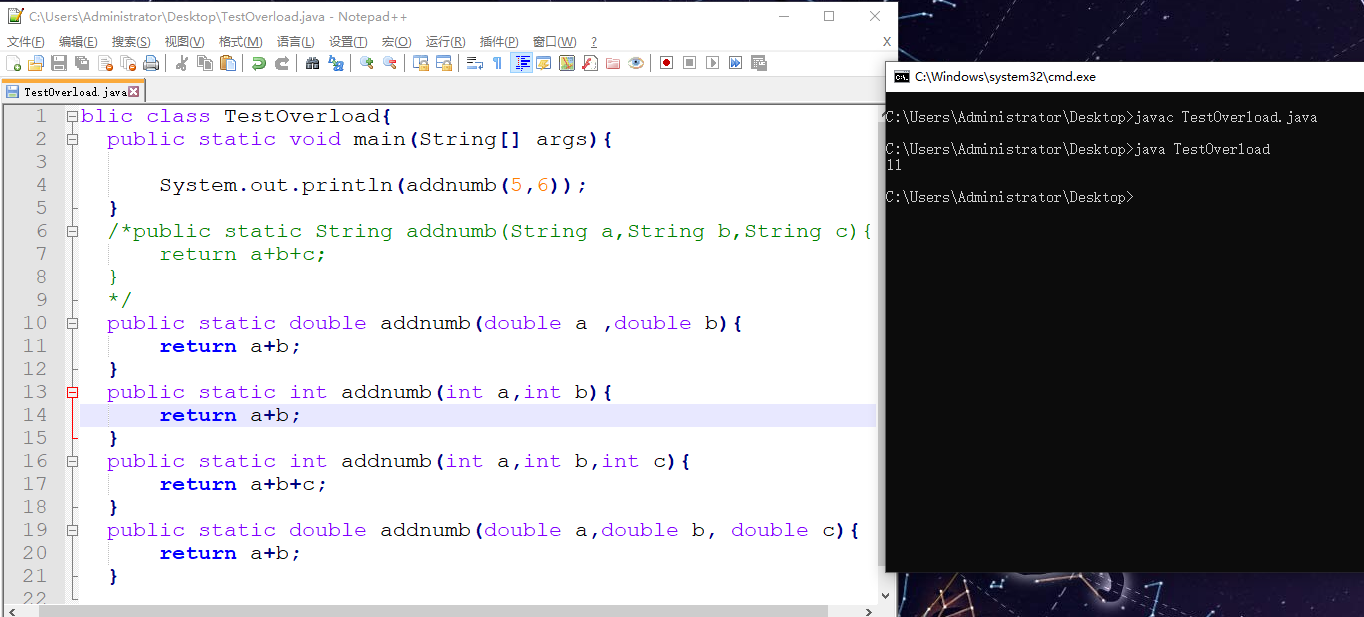
1.以参数个数为基准



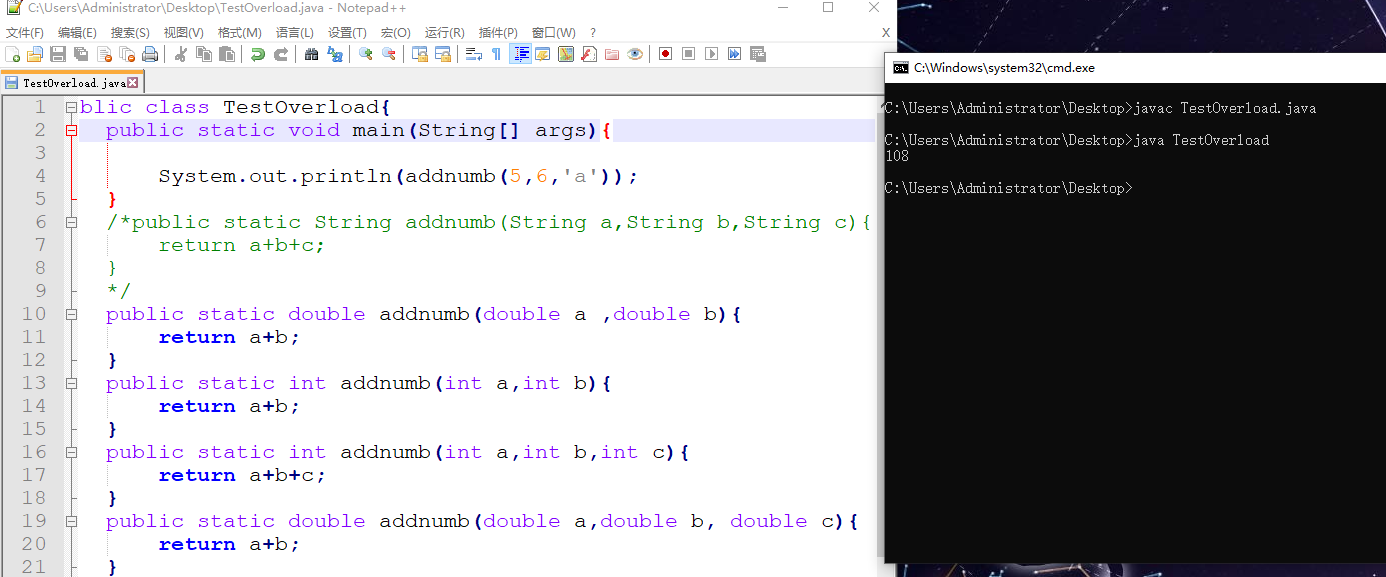
2.int型实参可以自动转为double传给形参



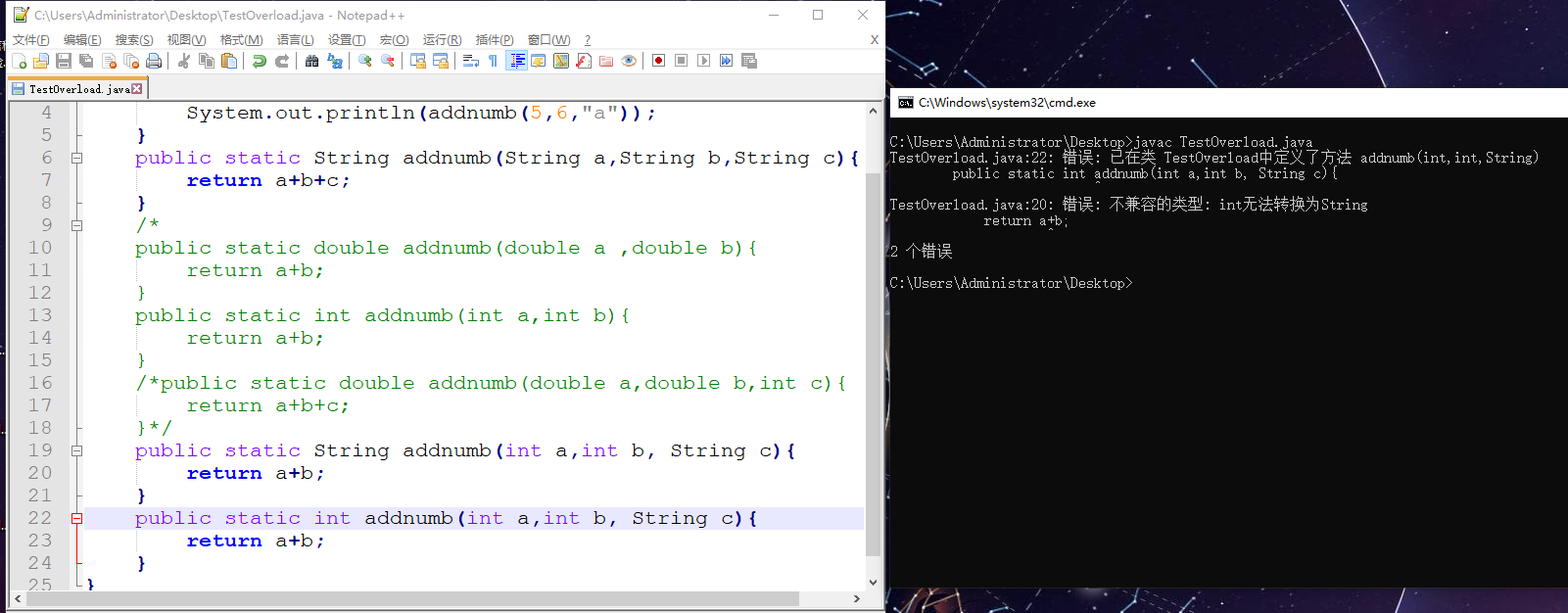
3.实参为整型时，默认为int



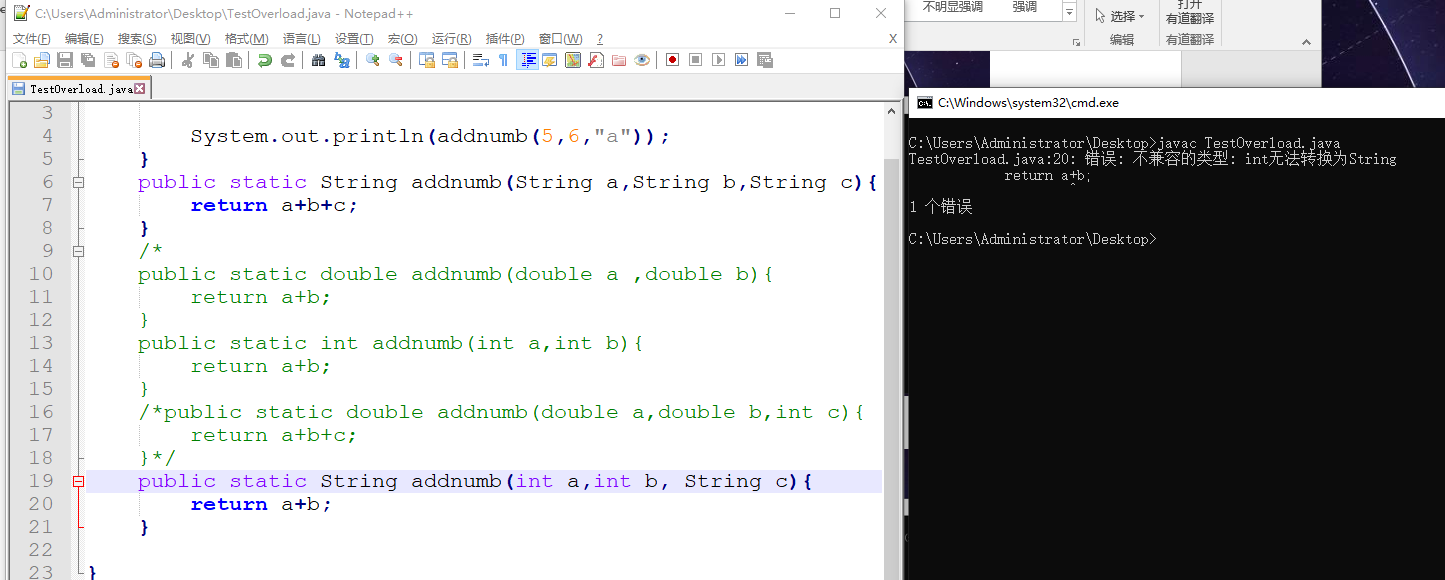
4.char型可以取表值看作int计算



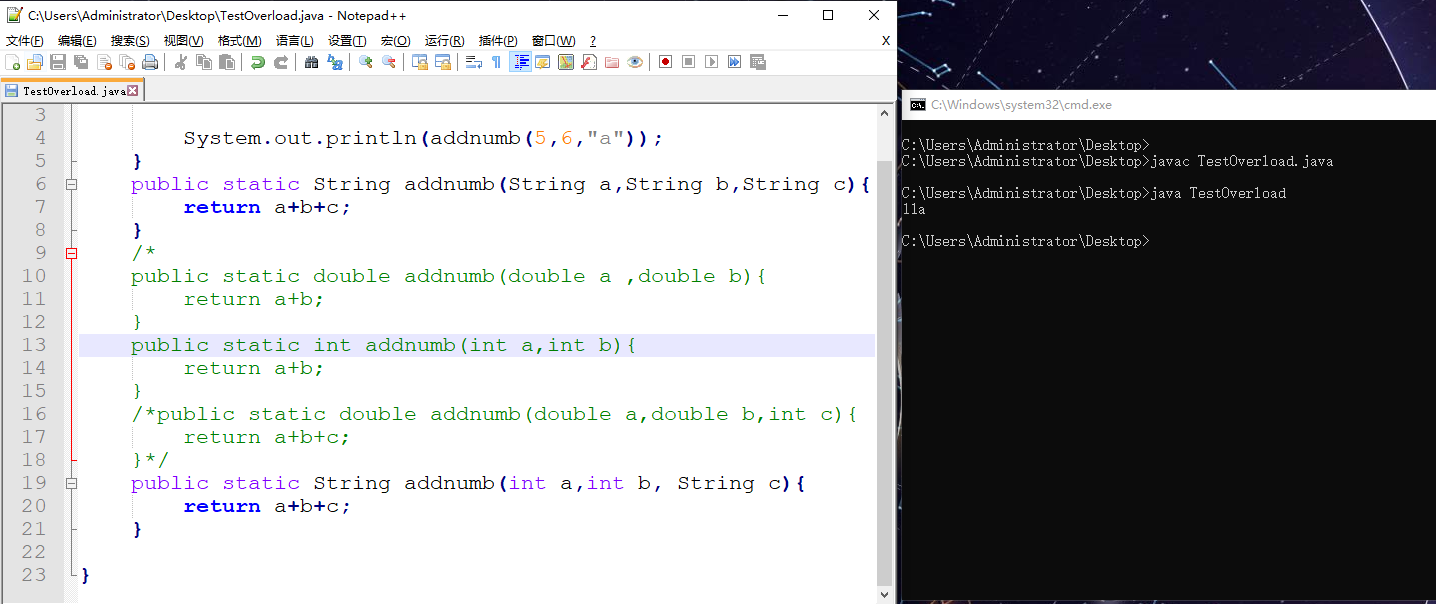
5.重载只和参数列表有关，与返回值类型无关



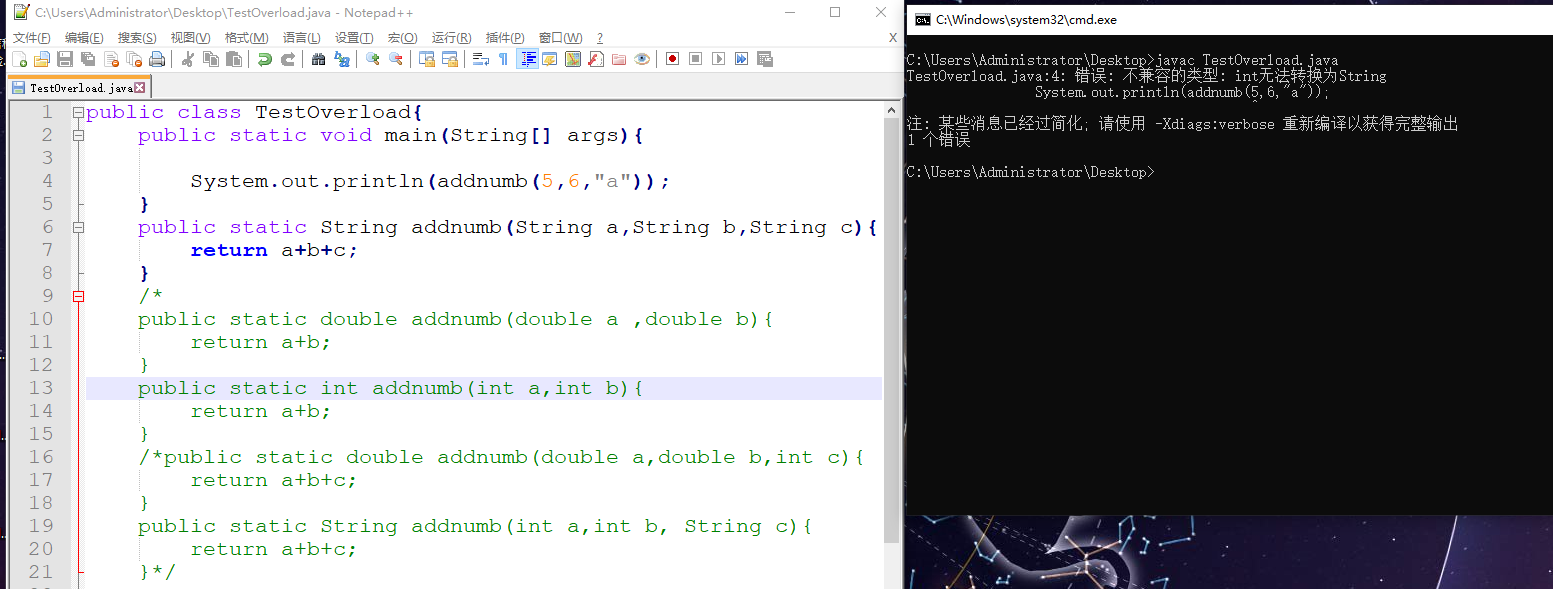
6.int型无法转为String，不兼容



7.返回值为String的情况



8.实参为int不能传给为String的形参



# 引用数据类型（类，数组，接口）

默认值为内存地址，即创建后直接输出显示内存地址。

## 创建引用数据类型的格式 数据类型 变量名 = new 数据类型（）；

# New 实例化对象

# 参数列表为引用数据类型时

## 形参内往往为接口，配合下面的格式，实现向上转型

## 形参里的引用类型 引用名 = 实参（）；（一般为向上转型）

### 实参为 new 引用类型 形参里的引用类型 引用名 = new 引用类型 向上转型

### 实参为之前已实例化的对象 形参里的引用类型 引用名 = new 之前实例化对象的类型

### 注意;实参必须是可以实例化对象的类型且是形参的实现类，原因在向上转型部分

# 变量名保存的是一个内存地址

# 自定义类

类中的属性是变量——>变量可以赋值取值

属性随类进入堆内存，赋值为默认值

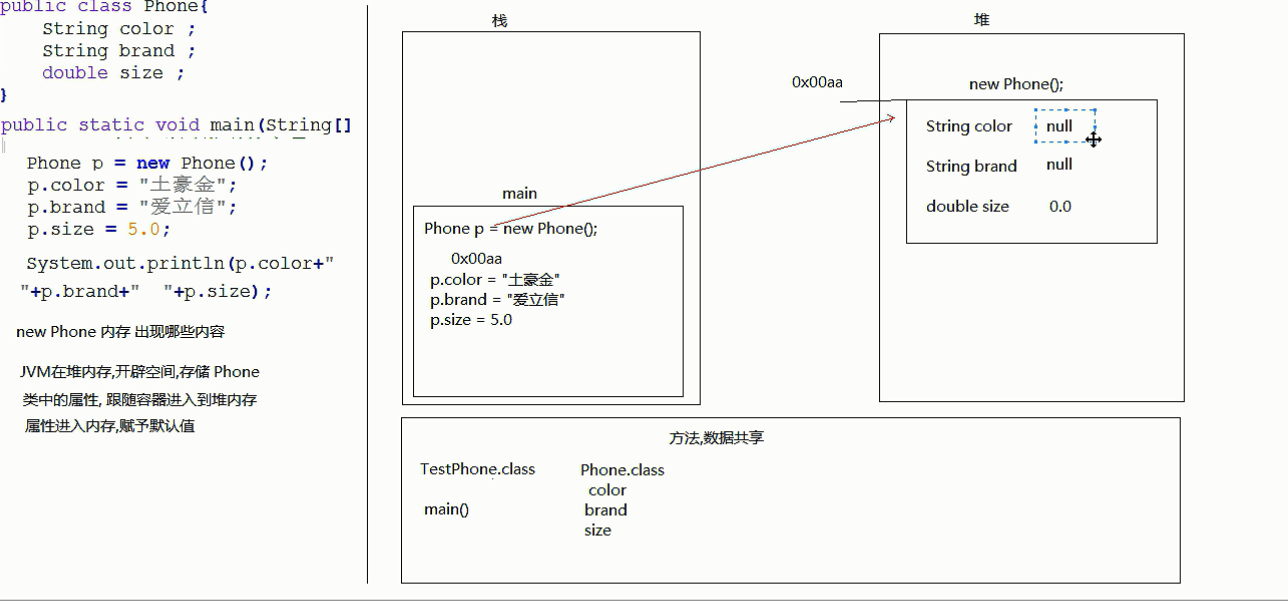
## **定义类中属性时修饰符可以不写**

取决于想要给予的权限，

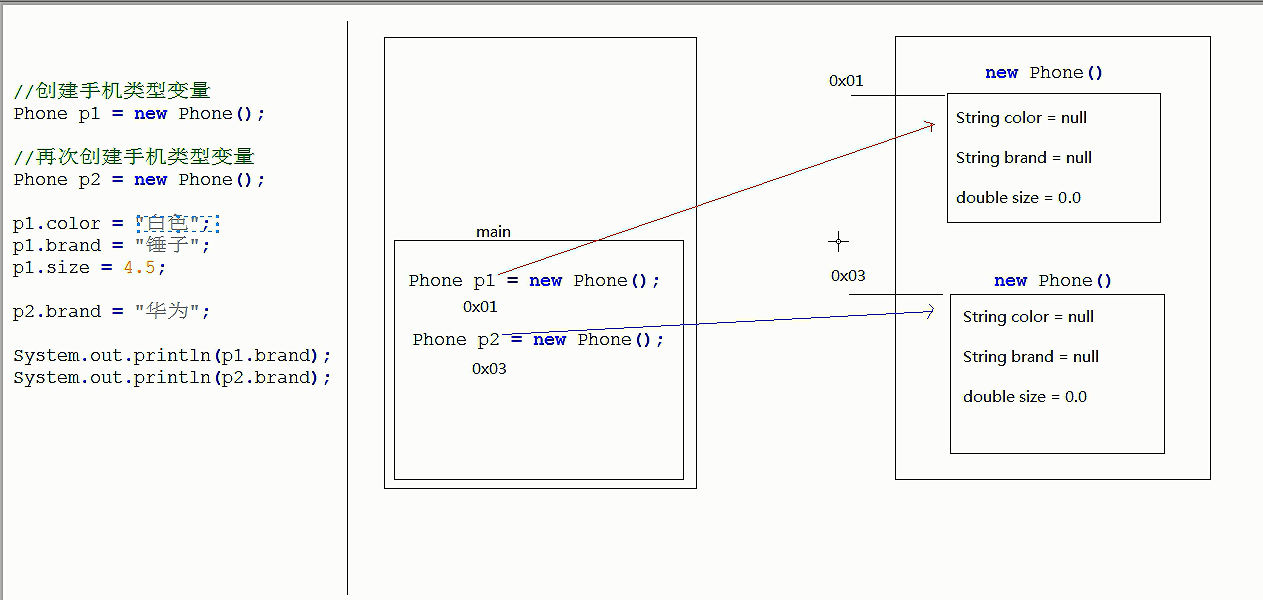
# Cmd中的运行谁编译谁

Cmd中运行谁编译谁即可，因为被调用的类也会被自动编译。

## 一个引用变量的内存图

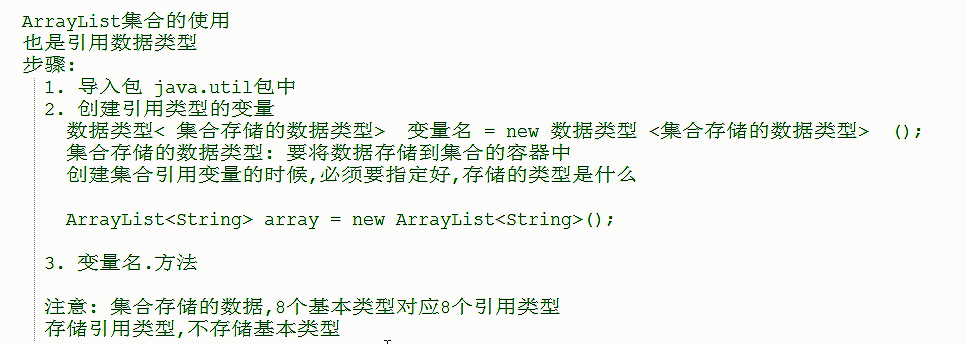


## 两个引用类型变量内存图



# ArrayList

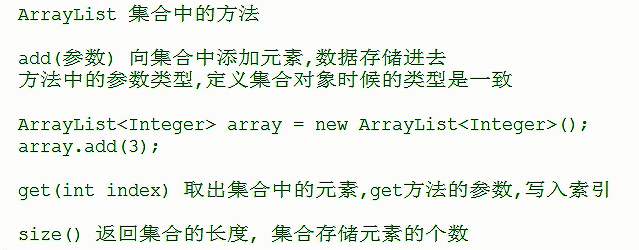
## 格式





ArrayList<> 变量名 = new ArrayList<>中的<>放引用类型均可，不只以上八种，包括自定义类。

## ArrayList中的方法



## 注意事项

### Add存入的是什么类型的数据，get取出的就是什么类型的数据

### Add方法存入自定义类类型数据，直接取出打印会显示地址，应·用该自定义类的变量接收，再打印该变量的属性。

# 百位十位个位的数如何求

## 百位 i/100

## 十位 i/10%10

## 个位 i%10

# Char 与 int

## Char两字节 int四字节，

## 两者计算 char自动转换为int型（根据ascii表）

## 想要把数字转为编码表对应的char字符 需要强制类型准备转换；char c= (char)i

## Char中保存汉字 查询unicode （万国表）

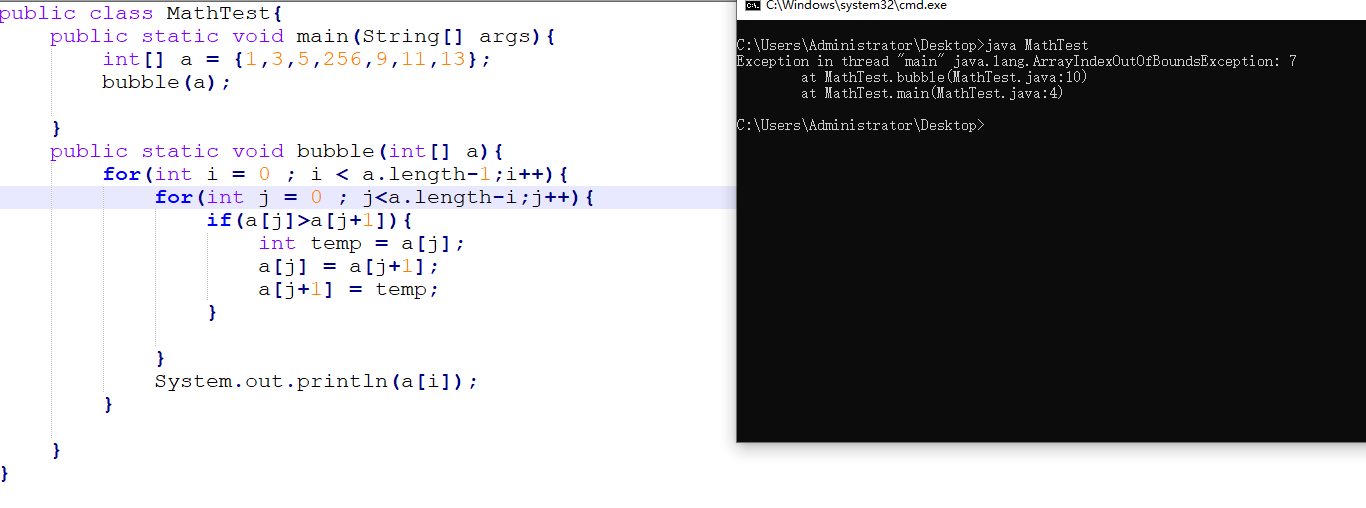
## Char转为int类型取值0-65535

## Char和short均为两字节取值不同的原因

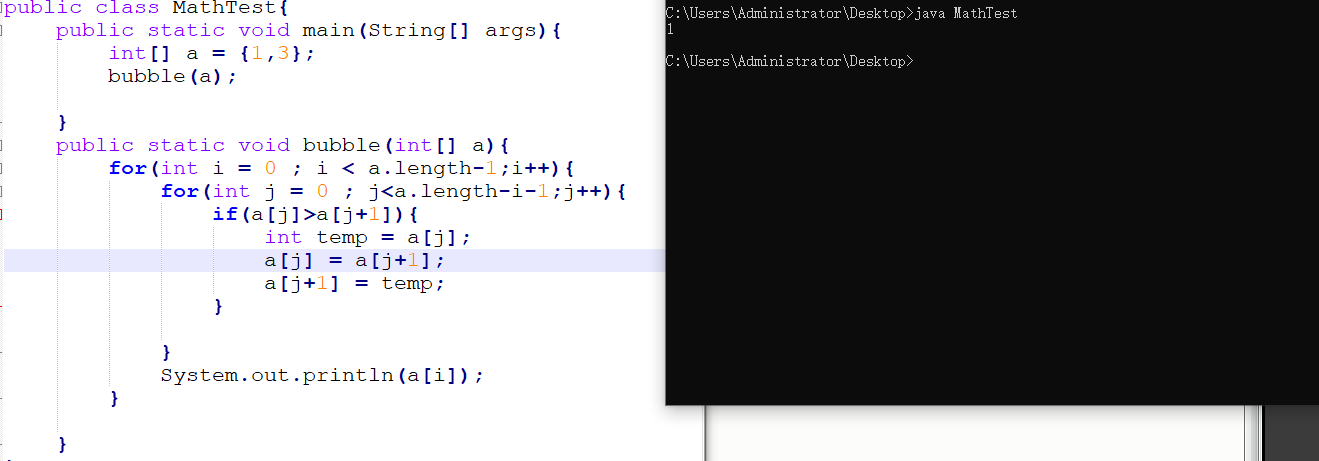


# 冒泡排序问题汇总

## 1.注意存在j+1，注意溢出问题



## 2.冒泡方法中的i小于数组长度-1，所以直接输出数组【i】时会少一个元素



# 面向对象的特性 继承 封装 多态

# 成员变量和局部变量区别

## 定义的位置不同

\* 定义在类中的变量是成员变量

\* 定义在方法中或者{}语句里面的变量是局部变量

## 在内存中的位置不同

\* 成员变量存储在对内存的对象中

\* 局部变量存储在栈内存的方法中

## 生命周期不同

\* 成员变量随着对象的出现而出现在堆中，随着对象的消失而从堆中消失

\* 局部变量随着方法的运行而出现在栈中，随着方法的弹栈而消失

## 内存位置不同

\* 成员变量因为在堆内存中

\* 局部变量没有默认的初始化值，。

## 默认值不同

\*成员变量有默认的初始化值

\*局部变量无默认值，不赋值不能使用，必须手动的给其赋值才可以使用

# 局部变量生命周期

只在定义时所在的方法或代码段内；

如for（int i= 0 ；；） i的生命周期只在for中

# **Private 只修饰成员变量，不修饰局部变量**；

# 继承时 子类中有则优先自己 没有则使用父类的变量或方法

# 方法“就近” 方法内有不使用则本类成员变量

方法内的局部变量并非真正的“覆盖“成员变量 因为一个在栈中 一个进入堆

# Super调用父类

# This 调用自己本类成员

带this.的是成员变量

# Get和set方法

在父类成员变量权限为private时提供调用修改成员变量的方法

## Get返回值 为成员变量数据类型

## Set返回值 为void

# 重写override

## 子类方法声明应该与父类相同

## 重写父类方法 子类权限应该不小于父类

### 权限由大到小 Public protected default private

其中default是默认权限，不写即为default 写default报错。

## 父类私有方法(private)子类不能重写

# 抽象类

## 抽象方法

### 方法只有声明部分

### 没有方法体（大括号）

### 用abstract修饰

## 抽象类： 包含抽象方法的类，一定是抽象类，使用 abstract 修饰的类

## 抽象方法所在类一定是抽象类：

## 抽象类与抽象方法都必须使用 abstract来修饰

## Abstract不能和 private， final， static一起使用

## 抽象类不能直接实例化对象（不能 new 抽象类）

## 抽象类中可以有抽象方法，也可以不写抽象方法

## 若子类只重写了父类部分抽象方法，子类仍为抽象类

# 继承（extends）

## 获得父类所有的成员：成员变量与方法

## 弊端：耦合性高

## 耦合性：一个地方修改，得修改很多地方

# 接口

## 接口内只有抽象方法

## 方法格式：Public abstract 返回值类型 方法名（参数列表）;

## 接口定义中Public abstract可以不写或选择性书写，默认为全有，实现时必须写出该有的；

## 接口中的方法均为public，接口内不写权限也默认为public

## 接口内成员变量必须定义为常量，只有常量没有变量

## 格式：public static final 数据类型 变量名 = 值；

## Static：可以直接被 类名. 的形式调用（其他为 对象.）

## Final ：变量值不可变，不可重新赋值，即使第二次赋值与第一次一样

## 接口定义中public static final可以不写或选择性书写，默认为全有

# 实现 implements

## 实现类可以为抽象类

## 接口中多实现（public void c implements a,b）

类不可以多继承，因为存在安全隐患，接口可以多实现，无安全隐患

### 可以多实现的原因

接口中全为抽象方法，无方法主体，方法的实现依赖实现类，重写才可执行，接口中的功能都没有方法体，由子类来明确，所以无安全隐患

### 多继承的安全隐患

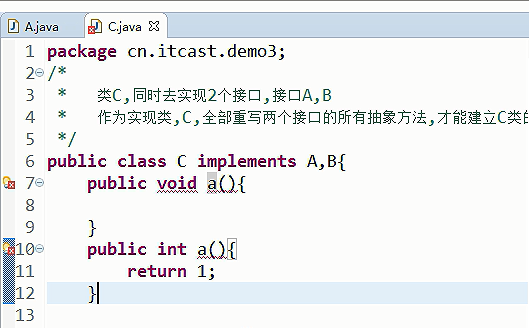
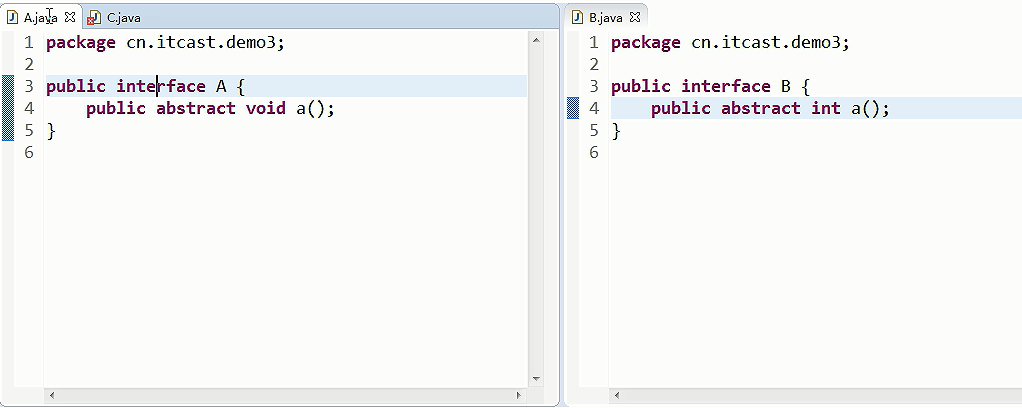
多继承时，当多个父类中有相同功能时，子类调用会产生不确定性。

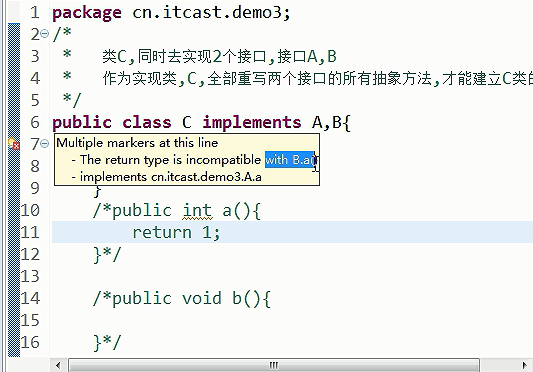
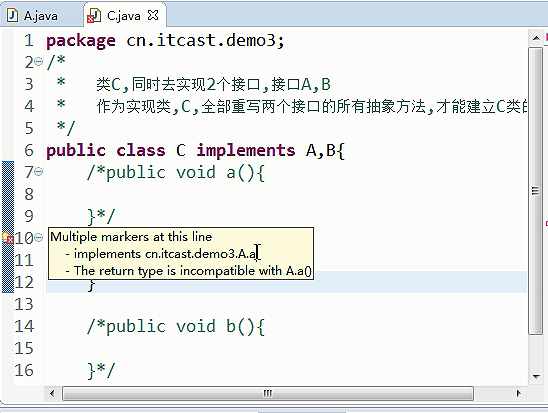
核心原因就是在于多继承父类中功能有主体，而导致调用运行时，不确定运行哪个主体内容。

为什么多实现能解决了呢？

因为接口中的功能都没有方法体，由子类来明确。

### 多实现中两接口内可以有重名方法，但若是两方法返回值不同，子类实现方法会报错（重复定义），两种返回值都实现或只实现一种全都报错



## 继承的同时多实现 （extends b implements c，d（）{}）

## 接口的多继承

### 接口之间有多继承，类没有

## 接口的好处

### **1.接口的出现扩展了功能**。

### **2.接口其实就是暴漏出来的规则**。

### 3.接口的出现降低了耦合性，即设备与设备之间实现了解耦。

# **接口和抽象类的区别**：

## 相同点:

都位于继承的顶端,用于被其他类实现或继承;

都不能直接实例化对象;

都包含抽象方法,其子类都必须覆写这些抽象方法;

## 区别:

抽象类为部分方法提供实现,避免子类重复实现这些方法,提高代码重用性;接口只能包含抽象方法;

一个类只能继承一个直接父类(可能是抽象类),却可以实现多个接口;(接口弥补了Java的单继承)

抽象类是这个事物中应该具备的你内容, 继承体系是一种 is..a关系

接口是这个事物中的额外内容,继承体系是一种 like..a关系

## 二者的选用:

优先选用接口,尽量少用抽象类;

需要定义子类的行为,又要为子类提供共性功能时才选用抽象类;

# 多态

## 需要有继承或实现关系

### 父类 变量名 = new 子类（）

### 成员变量 编译运行全看父类

调用父类的，父类没有该变量会报错

### 成员方法 编译看父类，运行看子类

调用子类实现或重写的方法,父类没有该方法会报错

## Instanceof ：比较对象是否是某引用类型

### 返回类型：boolean

### 格式：Boolean b = 变量名 instanceof 引用类型；

### 注意：需要变量所属类需要和instanceof后的引用类型存在继承或实现关系，否则报错

## 多态中的转型

### 向上转型 父类 变量名 = new 子类（）

#### 优点：可以调用父类子类公共方法

#### 缺点：无法调用子类特有方法（调用父类没有的方法报错）

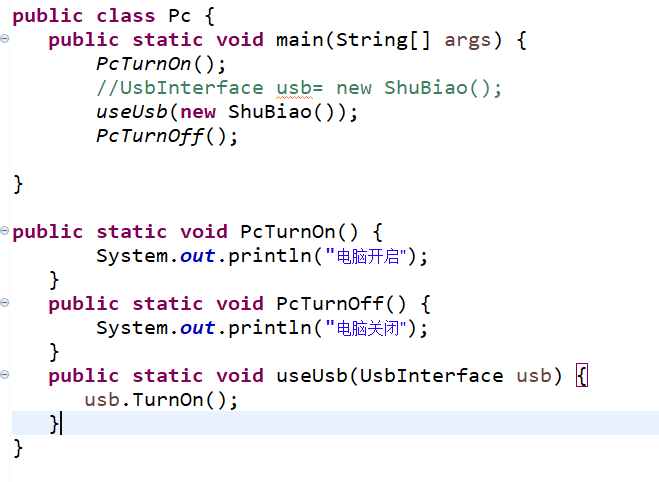
### 向下转型 将向上转型的对象强制类型转换成子类

#### 子类 新变量名 = （子类类型）原变量名

#### 优点：可以调用子类特有方法

#### 注意：应转回原本子类，转成其他类型，编译无错，运行报类型转换错误；

#### 注意：可使用if与instanceof避免出错



# 构造方法（构造器）

## 格式：权限修饰符 方法名（参数列表）{}

### 方法名必须与类名一模一样

### 无返回值，void也无，可以写return （无意义）

### 有方法主体

## 每一次new执行一次，也只执行一次

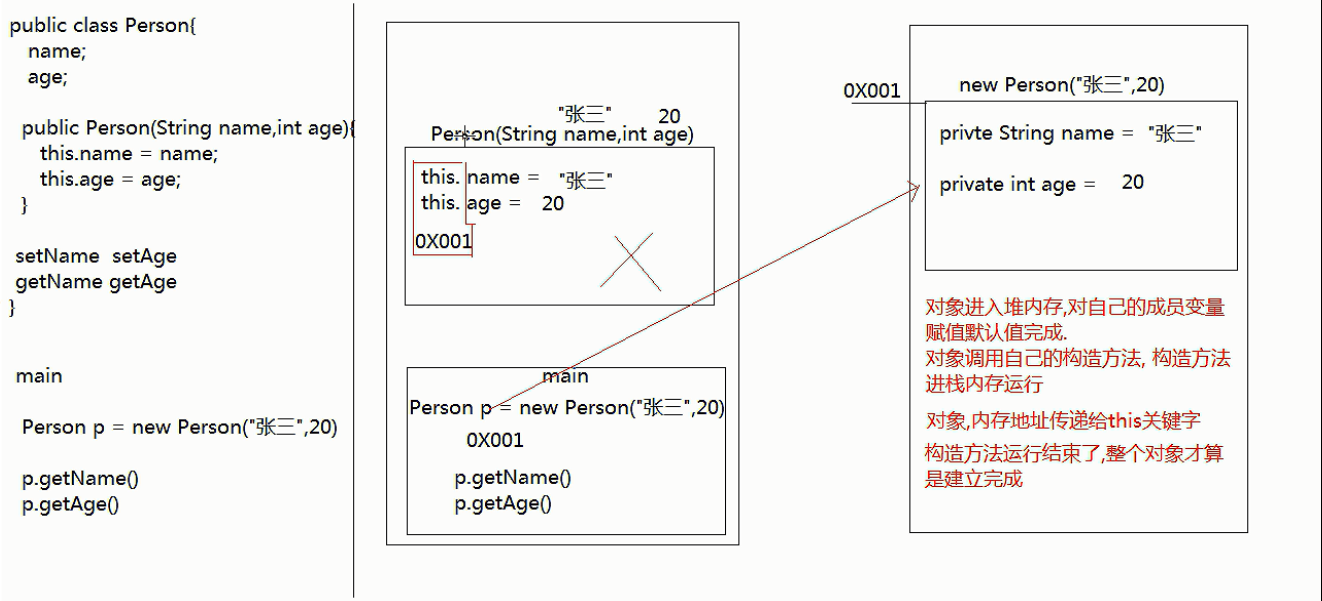
## Class中必有构造方法，不写也有

编译时，javac自动检查有无构造方法

如果没写构造方法，则自动添加

## 写了有参构造方法，new 数据类型（）要加上参数，否则报错

## 构造方法的内存图（在new对象赋默认值之后，把内存地址传给变量名之前调用）



## 构造器的重载

根据参数列表决定调用哪个

## 构造方法可以修改权限修饰符，但权限为private无法被其他类调用

## 构造方法如果赋值了成员变量，也有必要写get和set方法

## This（）与super（）

### 小括号中为空或参数列表

### This（）：用于调用本类的构造方法（根据参数列表）

### super（）：用于调用父类构造方法（根据参数列表）

### This（）与super（）都必须在构造方法第一条

### This（）与super（）两者冲突

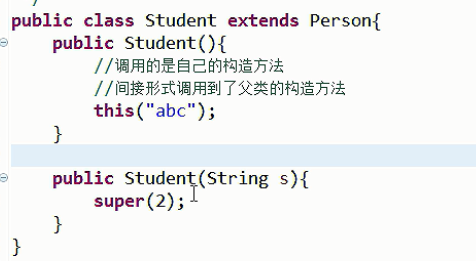
### 子类this()与super（）都不写 默认为super()；

#### 继承关系中先在父类写有参构造方法后子类报错 因为子类构造方法第一句默认的super（）为无参

#### 子类构造器无论重载多少个 都要注意默认super（）为无参的问题；

### 默认为super（）是因为子类构造方法只能选其一 但必须要保证可以调用父类的构造方法，允许间接调用

### 子类构造方法可以间接调用父类构造方法 图



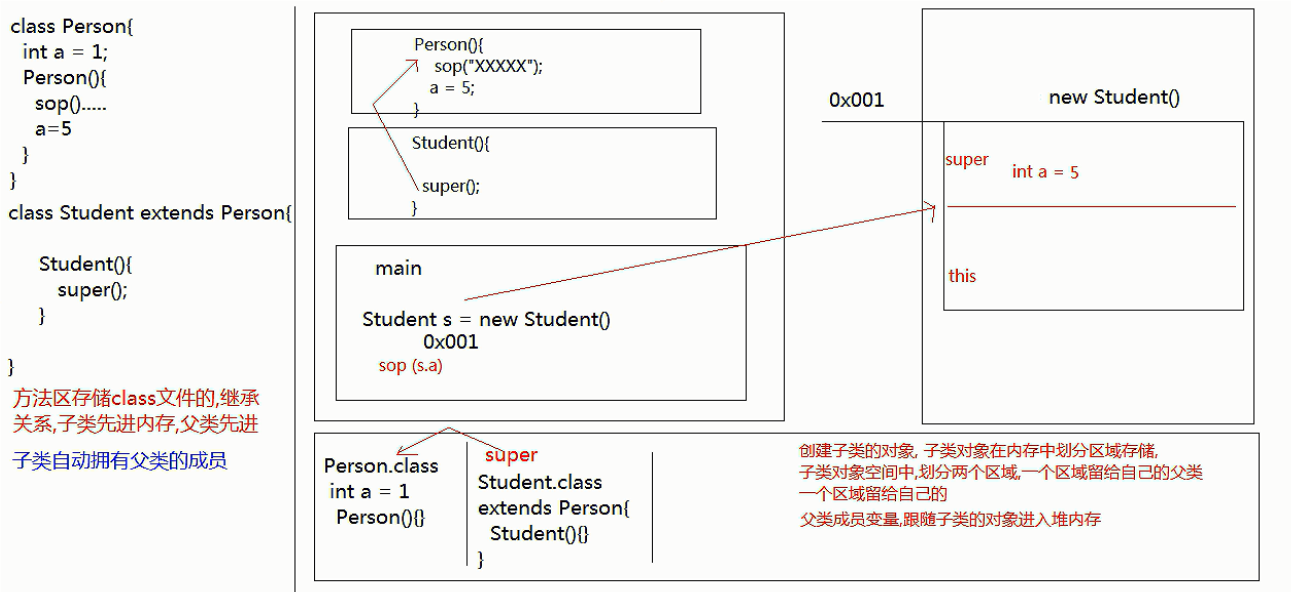
### Super（）用于了解父类成员变量的赋值。

### 父类构造方法中第一行没必要super 因为父类的父类是 object（object类的构造方法为空参）没必要写

### 子类调用父类构建方法的内存图

#### 在方法区中父类class文件先进去，子类后进入并根据super()获取父类地址

#### 创建子类对象 对象空间分为两个区域，一个留给父类，一个留个自己，故父类成员变量跟随子类进入堆内存



### 父类有多个构建方法 子类可以任选其一调用 调用不同的构建方法也可以