复习：

1. 面向对象和面向过程的比较

A. 面向过程着重于每一步的执行细节；面向对象是寻找对象，只要有了对象就能拥有对象的一切功能

B. 简单场景下适合于面向过程，复杂场景下适合于用面向对象

C. 面向对象是基于面向过程的

2. 类和对象的关系

A. 类是对象的概括，对象是类的具体表现

B. 属性是一种特征，方法是一种行为

3. 对象的内存存储

A. 对象是存在堆内存中

B. 对象中的属性在堆内存中会自动赋予默认值

C. 如果一个对象被赋值为null，那么意味着这个对象没有对应的地址指向，所以这个利用这个对象去操作一切的方法和属性都会出现空指针异常

4. 成员变量和局部变量的区别：定义位置、作用范围、内存位置、生命周期

5. 构造方法

A. 定义形式：与类同名，没有返回值类型

B. 任何一个类中都有构造方法。如果没有手动定义，在编译的时候会自动添加一个public的无参构造。

C. 构造方法用于创建对象

D. 构造方法中可以写return语句，作用是用于避免非法数据

E. 构造方法可以私有，但是此时无法在类外再进行继承

6. this

A. 表示本类的当前在活动的对象的引用，可以认为是一个虚拟对象

B. 用于在本类中调用非静态方法和非静态属性的

C. this语句用于在本类的构造方法中调用本类其他形式的构造方法的，必须放在首行

7. 代码块

A. 构造代码块：在创建对象的时候先于构造方法执行

B. 局部代码块：缩小变量的使用范围，减少变量的生命周期，提高内存的利用率

8. 面向对象的特征：封装、继承、多态

9. 封装

A. 形式：方法、属性的私有化、内部类

B. 优势：复用性、合法性

10. 权限修饰符

A. public>protected>默认>private

B. protected在子类中使用的时候，必须是在对应的子类中使用

11. 继承

A. 使用extends关键字来完成继承

B. 在java中，类和类之间是单继承

C. 单继承和多继承的比较：单继承能够明确方法的调用，多继承是能够更好的提高代码的复用性

D. 优势：复用性、明确性

E. 子类通过继承可以使用父类中的一部分方法和属性

12. super

A. 代表父类对象的引用，可以认为是一个虚拟对象

B. 通过super调用父类中的方法和属性

C. super语句---在子类的构造方法中调用父类对应形式的构造方法。子类对象在创建的时候必然先创建父类对象

13. 方法的重写

A. 在父子类中存在方法签名一致的非静态方法

B. 创建的子类对象在调用方法的时候是调用重写之后的方法

C. 遵循两等两小一大

### 多态

编译时多态：方法的重载

运行时多态：向上造型、方法的重写 --- 基于继承

注意：利用向上造型创建对象的时候，父类规定可以做什么，子类决定怎么做

#### 重写的理解

1. 子类重写的方法的权限修饰符的范围要大于等于父类的权限修饰符的范围

class A {

public void m(){}

}

class B extends A {

protected void m(){}

}

A a = new B(); // a对象的声明类是A，所以A类就会告诉a对象，有一个m方法可以使用，而且m方法是public修饰的，可以在任何一个地方使用

a.m(); // 方法的执行要看实现类，实现类是B，B却告诉a这个m方法不能在其他类中使用

2. 如果父类的方法的返回值类型是引用类型，那么子类方法的返回值类型要是父类方法返回值类型的子类

class A{}

class B extends A {

public void mb(){}

}

class C {

public B m(){return new B();}

}

class D extends C {

public A m(){return new A();}

}

C c = new D(); // c对象的声明类型是C类型，C类型就告诉c对象有一个m方法可以使用，这个m方法的返回值类型是B

c.m().mb(); // 调用方法的时候要看实现类，实现类是D，D却告诉m方法返回了一个A对象，A对象身上一定有有mb方法吗？---不行

注意：基本类型之间是没有继承关系的

B -> A

A a = new A();

B b = (B)a;

## static

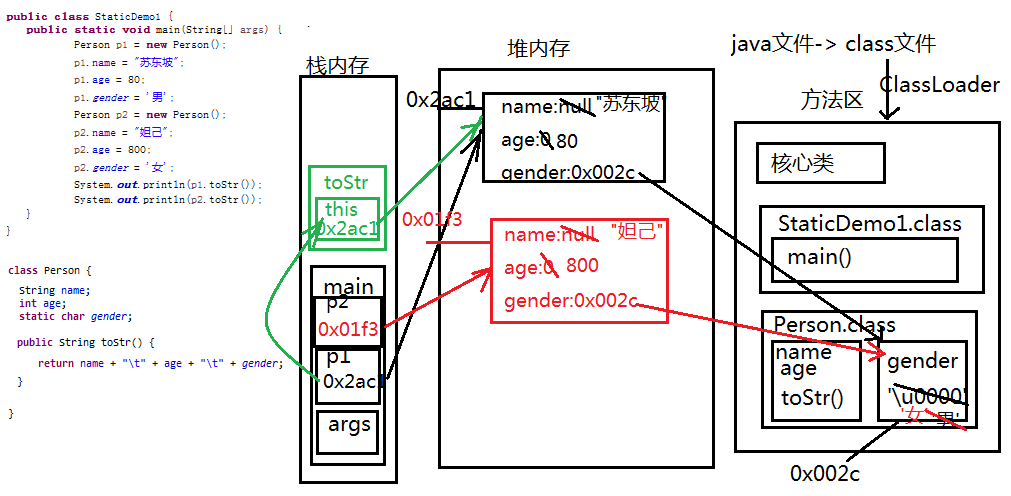
可以修饰变量、方法、代码块、内部类

### 静态变量

static修饰变量的时候称之为静态变量，也叫类变量。静态变量是在类加载的时候加载到了方法区，并且在方法区中初始化。静态变量先于对象而存在。可以不通过对象而是通过类名来调用，也可以通过对象来调用，但是习惯上是通过类名来使用。由于静态变量是存在方法区中的，所以这个类产生的所有的对象在堆中存储的是静态变量的地址。也因此静态变量是被所有对象所共享的。

System.out

System.in



注意：

1. 类是加载到方法区中的。

2. 类是在第一次被使用的时候才去加载

3. 类加载到方法区中之后不会移除，除非程序结束所有的内存全部释放---类只加载一次

4. 方法的执行是在栈内存中执行

静态变量可以放在构造方法中定义吗？---不行---静态变量是在类加载的时候出现，比对象先出来。构造方法是在创建对象的时候才会调用执行

### 静态方法

static修饰方法称之为静态方法，也叫类方法。静态方法也是随着类的加载而加载到方法区，只是存储在方法区并不执行，在方法被调用的时候到栈内存中执行。静态方法不依赖于对象的，而是依赖于类的---静态方法是先于对象出现的，所以习惯上都是通过类名来调用的静态方法。

Arrays.sort

System.arraycopy

Arrays.toString

Math.sqrt

可以在静态方法中使用this吗？---不可以

可以在静态方法中直接调用本类中的非静态方法吗？---不可以---所有的非静态方法和非静态属性都是通过对象来调用的，而静态方法不需要创建对象调用

可以在静态方法中定义静态变量吗？---不可以

静态方法可以重载吗？---可以

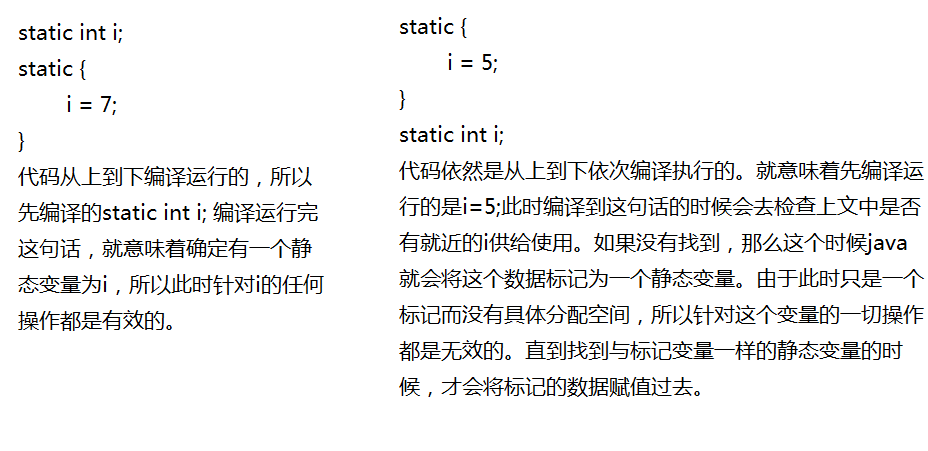
静态方法可以重写(override)吗？-- 不能 --- 但是在父子类中可以存在方法签名一致的静态方法---隐藏（hide）---如果父子类中存在了方法签名一致的方法要么都是静态（隐藏），要么都是非静态（重写） ---方法构成重写，执行的时候看的是实现类；如果方法构成的隐藏，执行的时候看的是声明类。

静态方法可以被继承吗？--可以

### 静态代码块

静态代码块在类第一次被真正使用（创建对象或者调用方法）的时候执行

执行顺序：父类静态->子类静态->父类构造->子类构造



练习：统计这个类一共创建了多少个对象 --- 静态变量