复习：

1. 构造方法：

特点：与类同名而没有返回值类型

作用：用于创建对象

当类中没有手动指定构造方法的时候在编译的时候自动添加一个无参构造

2. this关键字：代表本类的在活动的对象，用于在本类中调用本类的方法和属性 --- this语句，表示调用本类中其他的对应形式的构造方法，必须放在首行

3. 代码块：

构造代码块：在创建对象的时候先于构造方法执行一次 --- 用于完成一些初始化操作

局部代码块：提高栈内存的利用率

4. 封装：

体现形式：方法、类（属性的私有化 -> 将属性设置为private，对外提供对应的get/set的方法，在方法中可以进行限定使数据更加符合场景要求）

优势：提高复用性，保证数据的合法性

5. 继承 ：用extends关键字来完成继承关系 --- 支持的类与类之间的单继承 --- 单继承和多继承的优劣性 --- 提高复用性，避免方法调用产生歧义 --- 子类可以继承父类全部的数据域，但是只有一部分可见

6. 多态：

编译时多态：方法的重载

运行时多态：

向上造型：用父类来声明用子类来创建 --- 对象能干什么看的是父类，对象如何执行方法看的是子类。

方法的重写(Override)：父子类中存在了方法签名完全相同的非静态方法。--- 两等两小一大

7. 权限修饰符：public protected 默认 private --- protected在子类中使用的时候指的是在对应的子类中使用，不能跨子类使用

8. super关键字：在子类中表示父类对象的引用，用于调用父类中的方法和属性 --- super语句，表示在子类的构造方法中调用父类对应形式的构造方法。子类构造方法中如果没有手动指定super语句，那么默认调用父类无参构造（super()）;如果父类只提供了含参构造，那么子类的构造方法中必须手动提供对应形式的super语句

## static - 静态

修饰符 - 用于修饰数据、方法、代码块以及内部类

### 静态变量

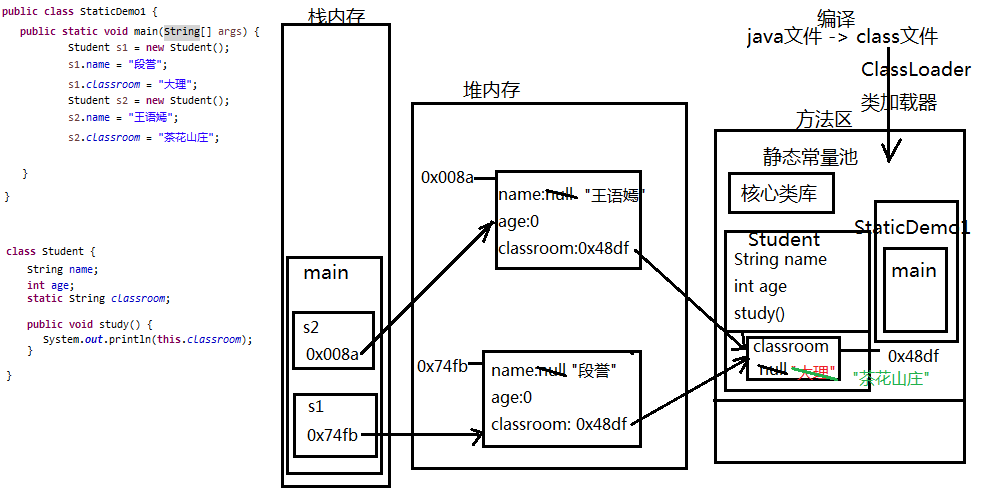
用static修饰变量，称之为静态变量，也叫类变量。在类加载的时候加载到了方法区，并且在方法区中被赋予了默认值。静态变量是先于对象出现的，所以习惯上是通过类名来调用静态变量。每一个对象存储的是这个静态变量在方法区中的地址，所以静态变量是被这个类的所有对象所共享的

静态变量能否定义到构造方法中？---不能。静态变量是在类加载的时候出现，先于对象出现。构造方法在创建对象的时候执行。

注意：

1. 类是加载到方法区中的

2. 类是在第一次使用的时候才加载，加载之后就不会移除



练习：定义一个类，统计这个类创建对象的个数

### 静态方法

用static修饰的方法，称之为静态方法。静态方法随着类的加载而加载到方法区中，但是在方法区中不执行只存储，在方法被调用的时候到栈内存执行。静态方法先于对象存在，所以习惯上是通过类名来调用静态方法。

main Arrays.sort() System.arraycopy()

静态方法中可以定义静态变量吗？--- 不能 --- 静态方法在调用的时候执行，静态方法执行的时候里面的变量才能初始化；静态变量是在类加载的时候初始化

静态方法中能否使用this/super？--- 不行 --- this代表当前在活动的对象，静态方法先于对象存在

能否在静态方法中直接使用本类中的非静态方法/非静态属性？--- 不行

public class A {

int i = 5;

public static void m(){

int j = (this.)i;

}

}

静态方法可以重载吗？---可以

静态方法可以被继承吗？---可以

静态方法可以重写吗？---不可以

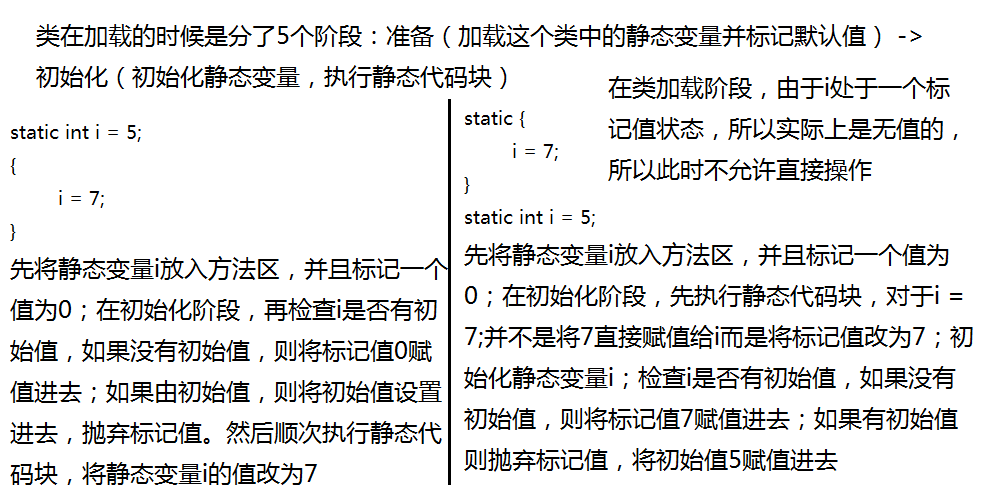
静态方法虽然不能被重写，但是父子类中可以存在方法签名一致的静态方法 --- 静态方法的隐藏（hide）

注意：父子类中可以存在方法签名一致的方法，要么都是非静态（重写）要么都是静态（隐藏）

### 静态代码块

用static{}包起来的代码 --- 在类加载的时候执行一次

执行顺序：父类静态 -> 子类静态 -> 父类非静态 -> 父类的构造方法 -> 子类非静态 -> 子类的构造方法



## final

修饰符 --- 修饰数据、方法以及类

final修饰数据的时候 --- 常量 -> 定义好之后不可改变。如果final修饰的是基本类型的数据，那么指的是实际值不可变；如果final修饰的引用类型的数据，那么指的是地址不可变，但是对象中的元素或者属性值可以改变 --- 对于成员常量要求在对象创建完成之前给值；对于静态常量而言要求在类加载完成之前给值

arr.length System.in System.out

注意：常量的存储和方法区中的运行时常量池有关

public class A {

private final int i;

/\*

{

i = 4;

}

\*/

public A(){

// this(5);

this.i = 0;

} // 无参构造中未初始化常量i

public A(int i){

this.i = i; // 常量的二次赋值

}

}

final修饰方法 --- 最终方法，能被继承但是不可被重写/隐藏，能被重载。

final修饰类 --- 最终类 --- 不能被继承