1. 当创建对象时,针对于对象中成员变量赋值场景有3个

1) JVM虚拟机,会将所有的成员变量进行一个默认的赋初值动作

2) JVM检测,如果有显示的赋值,那么显示赋值会替代掉默认赋值

3) JVM自动的调用了构造方法,给成员变量进行赋值

**this 关键字** : 表示本类对象的引用

this关键字的使用场景:

1. this关键字,区分成员变量和局部变量, 在变量之前添加this关键字,就表示调用的变量是成员变量
2. this关键字可以在构造方法之间进行调用,使用方式: this(构造方法参数列表);

说明: this() 必须写在构造方法的第一行

1. static关键字可以做到,static关键字具有共享性, 静态属于类,不属于任何一个对象
2. 静态的加载时机：当类的.class文件加载进内存时随着类的加载一起加载。
3. 静态优先于对象存在的
4. 静态修饰的成员可以被所有对象共享使用
5. 静态的调用方式:

静态属于类 : 1) 可以使用类名.直接调动(推荐) 2)可以创建对象,通过对象名调用(不推荐)

静态中不能使用非静态

1…静态方法中不能使用非静态的变量

2. 静态方法中不能使用非静态的方法

3. 静态方法中,不能使用this关键字

this关键字 : 表示本类对象的引用 , 静态优先于对象存在

1. 所属不同
2. 静态变量 : 属于类
3. 非静态变量 : 对象
4. 在内存中存储空间不同
5. 静态变量 : 方法区或数据共享,与类.class文件在同一块区域
6. 非静态变量 : 堆内存中
7. 生命周期不同
8. 静态变量 : 跟随着类进内存而进内存,随着类的死亡才结束
9. 非静态变量 : 随着对象的创建进内存,随着对象不再使用死亡
10. 静态的生命周期长于非静态
11. 访问方式不同
12. 静态变量 : 1) 类名.直接调用(推荐) 2) 创建对象,通过对象名.调用 (不推荐)
13. 非静态变量 : 创建对象,通过对象名.调用

注意: 1) 定义在局部代码块之外的变量,那么在代码块中对变量做的操作,出了局部代码块,仍然生效

1. 定义在局部代码块中的变量,一个是作用范围有限,如果局部代码块执行完毕,那么局部代码块中定义的变量就会变成垃圾,就会被回收掉,释放内存,节省内存空间.

总结代码块的执行顺序:

静态代码块(只执行一次,与创建几次对象无关)----->构造代码块(每次创建对象执行一次)---->构造方法(每次创建对象执行一次) 局部代码块(方法进栈运行执行)

**继承**：

继承的优势:

1. 提高代码的复用性
2. 提高代码的可维护性(代码维护:代码修改的难易程度),表示代码修改时,方便,简易
3. 继承为多态提供了前提

继承的弊端:

1. 提高了类与类之间的耦合性

耦合性 : 指类与类之间的关系,太紧密了

继承中的注意事项：

1. 父类中的私有成员(成员变量,方法),子类不能继承使用
2. 父类中的构造方法,不能被子类继承（但是可以被子类的构造方法调用）
3. 子类可以无条件的继承父类中的所有非私有成员变量

如果方法中定义了局部变量,优先使用局部变量

如果没有局部变量,优先使用类中定义的成员变量

如果类中没有成员变量,优先使用父类中成员变量

父类中如果也没有成员变量,那么接着继续找父类的父类...直到找到Object类,仍然没有这个变量,报错

**子类的方法权限,要大于等于父类方法权限**

**super : 表示父类对象的引用**

**super() : 表示调用父类的空参数构造方法**