static重要规则：类成员不能访问实例成员

1. static 变量

成员变量：定义在类里面、方法外面的变量, 分两种：

a. 实例变量;

b. 静态变量；形式和实例变量类似，在实例变量前面加static关键字；

static变量和实例变量的区别：

. static变量对于每个类而言在内存中只有一个，能被类的所有实例所共享；实例变量对于每个类的每个实例都有一份，它们之间互不影响；

. Java虚拟机在加载类的过程中为static变量分配内存，实例变量在加载完类后创建对象时分配内存；

. static变量可以直接通过类名访问，实例变量通过引用类型变量访问；

2. static 方法

成员方法分为静态方法和实例方法。用static修饰的方法叫静态方法，或类方法。静态方法也和静态变量一样，不需要创建类的实例，可以直接通过类名来访问。

a. static方法可以直接访问所属类的实例变量和实例方法，直接访问所属类的静态变量和静态方法

b. 父类的静态方法不能被子类覆为非静态方法。以下代码编译出错。

c. 父类的非静态方法不能被子类覆盖为静态方法；

3. static 代码块

类中可以包含静态代码块，它不存于任何方法中。在Java虚拟机中加载类时会执行这些静态代码块。如果类中包含多个静态代码块，那么Java虚拟机将按照它们在类中出现的顺序依次执行它们，每个静态代码块只会被执行一次。

类的构造方法用于初始化类的实例，而类的静态代码块则可用于初始化类，给类的静态变量赋初始值。它只执行一次，并在main函数之前执行。

静态代码块与静态方法一样，只能访问类的静态成员，而不允许访问实例成员。

静态代码块中不能使用this和super关键字。

4.类执行的内存机制

a:虚拟机栈(栈内存):八大基本数据类型+对象引用+returnAddress

特点:

1,方法执行需要在栈里执行,就是方法要执行需要先入栈。栈中的方法叫 栈帧。-->每一个方法被执行都会在当前栈中创建一个栈帧。

2,每个线程都对应有一个栈。

3,默认分配内存1M。

栈中主要存放一些基本类型的变量（int, short, long, byte,float, double, boolean, char）和对象句柄。

int a = 3; 这些称为自动变量，自动变量存的是字面值

b:堆:对象实例(所有的对象实例以及数组都要在堆上分配)

c:方法区

1:方法区:存储已被虚拟机加载的类信息,字段信息,方法信息,静态变量,类classloader的引用,类Class的引用,即时编译器编译后的代码等数据。

2:运行时常量池:常量,编译期生成的各种字符面量和符号引用，着部分内容将在类加载后存放在常量池中。

注：在方法区中，每个类型都对应一个常量池，存放该类型所用到的所有常量。

这里暂且只认为 常量池 = final修饰的变量 + ""String的值。

d:本地方法栈: 为虚拟机使用到的native方法服务.

e:程序计数器：当前线程所执行字节码的行号指示器。

JAVA类成员：变量、方法、构造器、初始化块（静态代码块、匿名代码块）

匿名代码块只在创建JAVA对象时隐式执行，而且在执行构造器之前执行。

静态初始化块在系统类初始化阶段执行，比匿名代码块先进行。

基本用法：如果有一段初始化处理代码对所有对象完全相同，且无须接收任何参数，就可以把这段初始化处理代码提取到初始化块中

5.singleton:

一个类始终只能创建一个实例，否则会影响系统性能下降（频繁地创建对象、回收对象带来的系统开销问题），例如：系统可能只有一个窗口管理器、一个假脱打印设备、一个数据库引擎访问点

该类唯一，没有子类，负责创建自己的对象，同时确保只有单个对象被创建。这个类提供了一种访问其唯一的对象的方式，可以直接访问，不需要实例化该类的对象。

饿汉模式，懒汉模式（非线程安全、线程安全）

二. final修改符

final具有"不可改变的"含义，它可以修饰非抽象类、非抽象成员方法和变量（包括类变量和实例变量），也可以修饰局部变量、形参。

. 用final修饰的类不能被继承，没有子类；

. 用final修饰的方法不能被子类的方法覆盖；

. 用final修饰的变量表示常量，只能被赋一次值；

final不能用来修饰构造方法，因为"方法覆盖"这一概念仅适用于类的成员方法，而不适用于类的构造方法，父类的构造方法和子类的构造方法之间不存在覆盖关系. 因此用final修饰构造方法是无意义的。父类中用private修饰的方法不能被子类的方法覆盖，因此private类型的方法默认是final类型的。

1. final类

继承关系的弱点是打破封装，子类能够访问父类的方法，而且能以方法覆盖的方式修改实现细节。在以下情况下,可以考虑把类定义为final类型，使得这个类不能被继承。

. 子类有可能会错误地修改父类的实现细节；

. 出于安全，类的实现细节不允许有任何改动；

. 在创建对象模型时，确信这个类不会再被扩展；

2. final方法；

某些情况下，出于安全原因，父类不允许子类覆盖某个方法， 此时可以把这个方法声明为final类型。例如在

java.lang.Object类中，getClass()方法为final类型，调用任何类对象的getClass()方法就可以看到该对象的原始面目。

final修饰的方法只是不能被重写，完全可以被重载。

3. final变量：

a. final可以修饰静态变量、实例变量、局部变量；

b. final变量都必须显示初始化，否则会导致编译错误；

1) 静态变量，定义变量时进行初始化或者static代码块中赋值；

2) 实例变量，可以在定义变量时，或者匿名代码块，或者在构造方法中进行初始化；

c. final变量只能赋一次值。

d. 当使用final修饰基本类型变量时，不能对基本类型变量重新赋值，因此基本类型变量不能被改变。

final只保证这个引用类型变量所引用的地址不会改变，即一直引用同一个对象，但这个对象完全可以发生改变。

e. ”宏变量“：当定义final变量时，就为该变量指定初始值，而且该初始值可以在编译时就确定下来。

编译器会把程序中所有用到该变量的地方直接替换成该变量的值。

一. abstract修饰符

可用来修饰类和成员方法。

. 用abstract修饰的类表示抽象类，抽象类不能实例化，即不允许创建抽象类本身的实例。没有用abstract修饰的类称为具体类，具体类可以被实例化。

. 用abstract修饰的方法表示抽象方法，抽象方法没有方法体。抽象方法用来描述系统具有什么功能，但不提供具体的实现。

没有abstract修饰的方法称为具体方法，具体方法具有方法体。

. 抽象类可以包含的内容：成员变量、方法（抽象方法、具体方法）、构造器、初始化块、内部类（接口、枚举）。

抽象类的构造器不能用于创建实例，主要是用于被其子类调用。

. 利用抽象类和抽象方法的优势，可以更好地发挥多态的优势，使得程序更加灵活。

. 抽象的作用：模板模式

语法规则；

1) 抽象类中可以没有抽象方法，但包含了抽象方法的类必须被定义为抽象类；

2) 没有抽象构造方法，也没有抽象静态方法；

3) 抽象类中可以有非抽象的构造方法；

4) 抽象类及抽象方法不能被final修饰符修饰。

二. 接口

接口使用的目的：解决多重继承问题；例如Fish类继承Animal类，表明Fish是一种动物，但鱼同样也是一种食物，如何表示这种关系呢？ 由于Java语言不支持一个类有多个直接的父类，因此无法用继承关系来描述鱼既是一种食物，又是一种动物，为了解决这一问题，Java语言引入接口类型，简称接口。一个类只能有一个直接的父类，但是可以实现多个接口。 采用这种方式，Java语言对多继承提供了有力的支持。

抽象类和接口比较：

1. 相同点：

a. 都不能被实例化；

b. 都能包含抽象方法；

2. 不同点；

a. 抽象类中可以为部分方法提供默认的实现，从而避免子类中重复实现它们，提高代码的可重用性，而接口中只能包含抽象方法；

b. 一个类只能继承一个直接的父类，这个父类有可能是抽象类；但一个类可以实现多个接口，这是接口的优势所在。

三.几种内部类的区别：

1. 创建

a. 声明的位置:

静态内部类：类的内部，方法的外部，用static关键字修饰；

实例内部类：类的内部，方法的外部，不用static关键字修饰；

局部内部类：方法的内部；

匿名内部类：既可以在类的内部，方法的外部，也可以在方法的内部；

b. 实例化方式:

静态内部类：new Outer.Inner(); //在外部类外创建；

new Inner(); //在外部类内内部类外创建

实例内部类：new Outer().new Inner(); //在外部类外创建；

this.new Inner(); //在外部类内内部类外创建

局部内部类：new Inner(); //只能在方法内部创建；

匿名内部类：new 类名() {};

2. 访问

a. 外部类访问内部类：

静态内部类：通过完整的类名直接访问静态内部类的静态成员;

（成员）实例内部类：通过内部类的实例去访问内部类的成员;

局部内部类：不能访问；

匿名内部类：不能访问；

b. 内部类访问外部类：

静态内部类：直接访问外部类的静态成员；

实例内部类：可以直接访问外部类的所有成员;

如果实例内部类B与外部类A包含同名的成员，那么在类B中， this.v表示类B的成员，

A.this.v表示类A的成员。

局部内部类：可以直接访问外部类的所有成员, 访问所在方法中的final类型的参数和变量；

匿名内部类：可以直接访问外部类的所有成员, 访问所在方法中的final类型的参数和变量；

四.final

final具有"不可改变的"含义，它可以修饰非抽象类、非抽象成员方法和变量（包括类变量和实例变量），也可以修饰局部变量、形参。

. 用final修饰的类不能被继承，没有子类；

. 用final修饰的方法不能被子类的方法覆盖；

. 用final修饰的变量表示常量，只能被赋一次值；

final不能用来修饰构造方法，因为"方法覆盖"这一概念仅适用于类的成员方法，而不适用于类的构造方法，父类的构造方法和子类的构造方法之间不存在覆盖关系. 因此用final修饰构造方法是无意义的。父类中用private修饰的方法不能被子类的方法覆盖，因此private类型的方法默认是final类型的。