Java中的static用法

1. Static 主要是三个用处：
2. 修饰成员变量：静态成员变量；
3. 修饰方法：静态方法；
4. 修饰代码块：静态代码块；
5. 静态成员变量和静态成员方法共同点：都可以使用类名直接调用，当然也可利用对象来调用，但是作用是相同的。
6. 关键字介绍：static既可以修饰属性也可以修饰方法也可以修饰类。
7. 修饰属性：（静态属性）无论一个类生成了多少个对象，这些对象都共用同一个唯一的静态变量；任何一个对象对此进行修改，则其他对象的静态变量的值也会随之修改。既然静态变量是共用的，则可以直接利用类名.静态变量名直接调用（这种方法是系统建议使用的），当然某个对象采用对象名.静态变量名也可以进行调用，但是不建议这种操作。
8. （考点，掌握不太好）修饰方法：利用static修饰的方法叫做静态方法。同静态属性一样，可以利用类名.静态方法名来访问此方法。父类和子类中的重写（override），若方法都是静态的，则子类的静态方法无法把父类中的同样方法覆盖掉。重写方法时，把父类的方法隐藏掉，调用的方法是引用所指向的类（父类或者子类）。如果子类和父类中的同一方法都是静态的，则调用的方法要看此引用是什么类型的（父类的或者子类的）。总结：非静态方法要看引用所指向的类（对象），对于静态方法则要看引用本身的类型。

子类可以继承父类的静态方法，但是子类无法重写父类的静态方法，只是隐藏了父类的静态方法。（静态方法只能继承不能重写（Override）。）

1. @Override 声明要重写父类的方法。
2. **静态代码块**。（使用的范围不广）作用：给类进行初始化。而构造函数是给对象进行初始化的。静态代码块在类加载的时候就执行，而构造方法在生成对象的时候执行；注意：要想调用某个类生成某个对象时，首先将类加载到java虚拟机上，然后由java虚拟机{JVM}加载这个类生成对象。先加载类再生成对象。由于类只会加载一次，所以静态代码块只会执行一次。注意某些类是不需要生成对象的（这些类一般内部都是静态变量和静态方法，不存在非静态的），所以初始化工作全都有静态代码块来完成。
3. **构造代码块（构造对象时首先执行的代码块）**：是与对象有关系的。作用是：**给所有对象进行一样的初始化**，而构造方法是对对象进行针对性的初始化。直接在类中定义的，与构造方法并列存在，区别于局部代码块。构造代码块的执行一定在构造函数之前，创建一个对象时，会先执行构造代码块，然后执行相应的构造函数。
4. **局部代码块：**在方法体中直接利用大括号括起来，用来限制变量的作用域，执行完毕就会释放内存空间。局部代码块是定义在方法体中的，注意区别于构造代码块。方法体中，局部代码块的执行顺序就是按照代码的排列顺序执行的。
5. Static 的应用：

对象的特有数据，最好是定义为static方法，这样就可以利用类名进行调用。

1. 如果一个类中的方法全部是静态的，此时再创建对象就是浪费空间。为了做到禁止其他程序创建对象，应该从构造方法上入手，如果没有可以调用的构造方法，也就是相当于无法创建对象，具体做法：将构造函数私有化即可。

例如 private Person(){}//这样把构造方法私有化，就不会创建出对象了。

1. static 定义方法是为了调用方便，可以节省内存空间，但是使变量的生命周期变长。
2. 静态方法（static）：是可以通过类名直接调用的（建议使用类名.静态方法），利用具体的对象或实例去调用当然还是可以的。所以见到static应该很高兴，直接用类名引用即可，很简单。

例如：jdk中给的一些类中的很多方法都定义为静态方法，如JDK给定了类Arrays，用来处理数组的，直接调用其静态方法：Arrays.equals（）；注意需要在开头把Arrays类导入，import java.util.Arrays; （导入类的时候一定要加上分号，否则出错）

1. 如果是需要直接利用类名.方法名调用某个方法，不再建立一个对象，可以把这样的方法声明为static。这就是static 的最明显的功能之一。这样就会方便了很多，如果以后见到直接利用类名,方法名调用某个方法时，就可以推出此方法一定为static的。
2. 通常情况下，this代表的就是调用该方法的对象，或者是调用该变量的对象。因此在静态方法中不能出现this，因为静态方法利用对象或类名调用时一样的，如果出现this代表某个对象，就会出现矛盾。
3. 静态代码块：在装载的时候就会执行，常用于对静态成员变量赋予初始值，使用的比较少。
4. 把某些属性或方法定义为static，可以节省内存空间，因为所有对象都共享这一个属性或方法。Static修饰的数据单独存在于，各个对象实体中不再存在此数据。但是静态的生命周期变得很长，随着类的消失而消失，这是缺点。
5. Static修饰的成员变量或方法虽类的加载而存在，所以可以利用类名直接调用。
6. Static修饰的数据是共享数据，而对象中的数据是特有数据。
7. 实例变量：一般的成员变量，实例中的变量；

静态变量（类变量）：所属类的变量，类可以直接访问的变量。

注意：类类型变量：。

1. 实例变量与静态变量的区别：

**答：（1）生命周期不同：**

**实例变量随着对象的创建而存在，随着对象的被释放而回收；**

**静态变量随着类的加载而存在，随着类的消失而消失，生命周期变长了（这是缺点）。**

**思考：类什么时候消失呢？虚拟机结束了，类就消失。Java具有垃圾回收机制，会对类进行回收的。**

**（2）调用方式不同：实例变量只能利用对象进行调用，而静态变量可以利用类名（建议利用类名调用）和对象调用。**

**（3）别名不同：实例变量就是成员变量；静态变量就是类变量，不要把静态变量也称为成员变量了。**

**（4）存储位置不同：成员变量数据存储在堆内存中，所以也叫作对象的特有数据；静态变量数据存储在：方法区中的静态区（也称为共享数据区），所以也叫作共享数据。**

1. 静态方法只能访问静态变量或者静态方法；非静态方法即可以访问静态的有可以访问非静态的。因为静态方法虽类加载而存在，但是若没有对象生成时，非静态变量或非静态方法没有产生呢。
2. 静态方法或静态变量不能使用this或super关键字。

主函数是静态的。为什么？静态方法中只能调用静态变量和静态方法，所以在主函数中只能调用静态的，但是大多数不是静态的，所以就强迫通过建立对象来进行访问。**主函数之所以声明为static的，在于不让变量和方法定义在主函数中，而是引导建立各自的类，从而对变量和属性进行封装，在主函数里面进行的只是生成类的对象，通过对象调用已经封装好的方法和属性，主函数做的只是指挥工作。**如果主函数所在类还有其他非静态变量或方法，需要对自己所在类创建对象，从而进行访问它们。例如：

**public** **class** PolymorphismDemo

{

**public** **static** **void** main(String[] args)

{

PolymorphismDemo pd = **new** PolymorphismDemo();

pd.name = 10;

pd.run();

}

**int** name;

**public** **void** run() {

}

}

1. 什么时候需要把成员变量定义成静态变量，什么时候需要把成员方法定义为静态方法？
2. 如果某个类生成的所有对象的同一个变量的值都是相同的，则就需要定义为static静态变量，这样可以节省内存空间；如果是不同的，就不能定义为static，因为此时此变量应该属于对象的特有变量。
3. 成员方法是否需要定义为静态的，只需要参照一点，该方法内部是否访问到对象的特有数据，也就是说，是否需要访问非静态的成员变量，如果不需要访问，则就可以定义为静态方法，当然也可以定义为非静态的，但是如果需要访问非静态成员变量（对象的特有数据），则就不能定义为静态方法。
4. 创建对象是为了封装**特有数据**（非静态的数据）的，而对象对静态变量或者静态方法（共享的）并没有封装，他们是封装在类中的，直接利用类名就可以访问。对象是用来调用自己的特有数据的（非静态的数据），如果利用对象调用了非静态方法，但是非静态方法中并没有调用非静态数据，此时创建的对象是没有任何意义的，因为不创建对象，直接利用类名调用也可实现此目的。
5. 静态代码块：

static表示“全局”或者“静态”的意思，用来修饰成员变量和成员方法，也可以形成静态static代码块，但是Java语言中没有全局变量的概念。

被static修饰的成员变量和成员方法独立于该类的任何对象。也就是说，它不依赖类特定的实例，被类的所有实例共享。

只要这个类被加载，Java虚拟机就能根据类名在运行时数据区的方法区内定找到他们。因此，static对象可以在它的任何对象创建之前访问，无需引用任何对象。

用public修饰的static成员变量和成员方法本质是全局变量和全局方法，当声明它类的对象市，不生成static变量的副本，而是类的所有实例共享同一个static变量。

static变量前可以有private修饰，表示这个变量可以在类的静态代码块中，或者类的其他静态成员方法中使用（当然也可以在非静态成员方法中使用--废话），但是不能在其他类中通过类名来直接引用，这一点很重要。实际上你需要搞明白，private是访问权限限定，static表示不要实例化就可以使用，这样就容易理解多了。static前面加上其它访问权限关键字的效果也以此类推。

static修饰的成员变量和成员方法习惯上称为静态变量和静态方法，可以直接通过类名来访问，访问语法为：

类名.静态方法名(参数列表...)

类名.静态变量名

用static修饰的代码块表示静态代码块，当Java虚拟机（JVM）加载类时，就会执行该代码块（用处非常大，呵呵）。

1、static变量

　按照是否静态的对类成员变量进行分类可分两种：一种是被static修饰的变量，叫静态变量或类变量；另一种是没有被static修饰的变量，叫实例变量。

两者的区别是：

　对于静态变量在内存中只有一个拷贝（节省内存），JVM只为静态分配一次内存，在加载类的过程中完成静态变量的内存分配，可用类名直接访问（方便），当然也可以通过对象来访问（但是这是不推荐的）。

　对于实例变量，每创建一个实例，就会为实例变量分配一次内存，实例变量可以在内存中有多个拷贝，互不影响（灵活）。

所以一般在需要实现以下两个功能时使用静态变量：

 在对象之间共享值时

 方便访问变量时

2、静态方法

静态方法可以直接通过类名调用，任何的实例也都可以调用，

因此静态方法中不能用this和super关键字，不能直接访问所属类的实例变量和实例方法(就是不带static的成员变量和成员成员方法)，只能访问所属类的静态成员变量和成员方法。

因为实例成员与特定的对象关联！这个需要去理解，想明白其中的道理，不是记忆！！！

因为static方法独立于任何实例，因此static方法必须被实现，而不能是抽象的abstract。

例如为了方便方法的调用，Java API中的Math类中所有的方法都是静态的，而一般类内部的static方法也是方便其它类对该方法的调用。

静态方法是类内部的一类特殊方法，只有在需要时才将对应的方法声明成静态的，一个类内部的方法一般都是非静态的

3、static代码块

　static代码块也叫静态代码块，是在类中独立于类成员的static语句块，可以有多个，位置可以随便放，它不在任何的方法体内，JVM加载类时会执行这些静态的代码块，如果static代码块有多个，JVM将按照它们在类中出现的先后顺序依次执行它们，每个代码块只会被执行一次。例如：

public class Test5 {

private static int a;

private int b;

static{

Test5.a=3;

System.out.println(a);

Test5 t=new Test5();

t.f();

t.b=1000;

System.out.println(t.b);

}

static{

Test5.a=4;

System.out.println(a);

}

public static void main(String[] args) {

// TODO 自动生成方法存根

}

static{

Test5.a=5;

System.out.println(a);

}

public void f(){

System.out.println("hhahhahah");

}

}

运行结果：

3

hhahhahah

1000

4

5

　利用静态代码块可以对一些static变量进行赋值，最后再看一眼这些例子，都一个static的main方法，这样JVM在运行main方法的时候可以直接调用而不用创建实例。

4、static和final一块用表示什么

static final用来修饰成员变量和成员方法，可简单理解为“全局常量”！

对于变量，表示一旦给值就不可修改，并且通过类名可以访问。

对于方法，表示不可覆盖，并且可以通过类名直接访问。

有时你希望定义一个类成员，使它的使用完全独立于该类的任何对象。通常情况下，类成员必须通过它的类的对象访问，但是可以创建这样一个成员，它能够被它自己使用，而不必引用特定的实例。在成员的声明前面加上关键字static(静态的)就能创建这样的成员。如果一个成员被声明为static，它就能够在它的类的任何对象创建之前被访问，而不必引用任何对象。你可以将方法和变量都声明为static。static 成员的最常见的例子是main( ) 。因为在程序开始执行时必须调用main() ，所以它被声明为static。

声明为static的变量实质上就是全局变量。当声明一个对象时，并不产生static变量的拷贝，而是该类所有的实例变量共用同一个static变量。声明为static的方法有以下几条限制：

•

它们仅能调用其他的static 方法。

•

它们只能访问static数据。

•

它们不能以任何方式引用this 或super（关键字super 与继承有关，在下一章中描述）。

如果你需要通过计算来初始化你的static变量，你可以声明一个static块，Static 块仅在该类被加载时执行一次。下面的例子显示的类有一个static方法，一些static变量，以及一个static 初始化块：

// Demonstrate static variables，methods，and blocks.

class UseStatic {

static int a = 3;

static int b;

static void meth(int x) {

System.out.println("x = " + x);

System.out.println("a = " + a);

System.out.println("b = " + b);

}

static {

System.out.println("Static block initialized.");

b = a \* 4;

}

public static void main(String args[]) {

meth(42);

}

}

一旦UseStatic 类被装载，所有的static语句被运行。首先，a被设置为3，接着static 块执行(打印一条消息)，最后，b被初始化为a\*4 或12。然后调用main()，main() 调用meth() ，把值42传递给x。3个println ( ) 语句引用两个static变量a和b，以及局部变量x 。

注意：在一个static 方法中引用任何实例变量都是非法的。

下面是该程序的输出：

Static block initialized.

x = 42

a = 3

b = 12

在定义它们的类的外面，static 方法和变量能独立于任何对象而被使用。这样，你只要在类的名字后面加点号运算符即可。例如，如果你希望从类外面调用一个static方法，你可以使用下面通用的格式：

classname.method( )

这里，classname 是类的名字，在该类中定义static方法。可以看到，这种格式与通过对象引用变量调用非static方法的格式类似。一个static变量可以以同样的格式来访问——类名加点号运算符。这就是Java 如何实现全局功能和全局变量的一个控制版本。

下面是一个例子。在main() 中，static方法callme() 和static 变量b在它们的类之外被访问。

class StaticDemo {

static int a = 42;

static int b = 99;

static void callme() {

System.out.println("a = " + a);

}

}

class StaticByName {

public static void main(String args[]) {

StaticDemo.callme();

System.out.println("b = " + StaticDemo.b);

}

}

下面是该程序的输出：

a = 42

b = 99

static成员是不能被其所在class创建的实例访问的。

如果不加static修饰的成员是对象成员，也就是归每个对象所有的。

加static修饰的成员是类成员，就是可以由一个类直接调用，为所有对象共有的

自我总结：只是针对成员方法，如果是abstract的，则此方法不能有函数体，即不能有大括号，之所以利用abstract来修饰，**目的就是用来被重载的**；如果是static的，则目的是为了在创建对象（实例）之前利用**类名直接调用此静态方法**；如果是final的，则目的是说明此方法**不能被重载**。显然，abstract和final的作用是相反的，所以这两个不能同时修饰一个方法。

成员方法讨论好了，其实成员变量差不多了。如果public static final 同时修饰一个成员变量，则说明此变量是**静态常量（全局常量）**。只用public static修饰，说明此变量是静态变量（全局变量）。

1. 再次学习static静态。
2. 静态成员变量：特点① 可以直接使用类名.成员变量（建议这样，也可以利用对象.成员变量名进行调用），进行调用，而非静态变量只能使用对象名.成员变量进行调用；特点② 静态成员变量是整个类的所有对象所共享的（就是共享的一个），任何一个对象对此进行修改，则其他对象或直接利用类调用时也是修改的值，对于非静态的成员变量，每个对象都有自己独有的变量。
3. 静态方法：特点① 同静态成员变量一样，直接可以使用类名.成员变量（建议这样，也可以利用对象.成员变量名进行调用），进行调用；特点② **静态方法中不能调用非静态的成员变量**。
4. 静态代码块：stack {静态代码；} 特点① 没有名字；无需调用，装载类时就会执行静态代码块，主要作用是用于类的初始化（使用不多，面试的时候常常考察）。
5. static 也是既可以修饰类、方法，也可以修饰属性；但是修饰类时比较特殊，以后讨论。Static的特点在于直接使用类名.\*\*\*直接调用方法或者属性。

（1） 修饰方法时，表示此方法不可以被重写，但是可以被隐藏，属于整个类的，此类的每个对象都可以调用此方法，建议直接类名.静态方法名，进行调用；

（2）修饰属性时 ，表示此属性属于整个类的，此类的每个对象的此方法都是一样的，建议直接利用类名.属性名，进行调用，其中任意一个对象对此属性修改了，那么其他的对象的此值也是修改了的。