        public void f1(final int i) {   
                //i++;    //i是final类型的,值不允许改变的.   
                System.out.print(i);   
        }   
}

**二、static**

        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，没创建一个实例，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）。

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。

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。

        特别要注意一个问题：

        对于被static和final修饰过的实例常量，实例本身不能再改变了，但对于一些容器类型（比如，ArrayList、HashMap）的实例变量，不可以改变容器变量本身，但可以修改容器中存放的对象，这一点在编程中用到很多。

        也许说了这么多，反倒把你搞晕了，还是看个例子吧：

public class TestStaticFinal {   
        private static final String strStaticFinalVar = "aaa";   
        private static String strStaticVar = null;   
        private final String strFinalVar = null;   
        private static final int intStaticFinalVar = 0;   
        private static final Integer integerStaticFinalVar = new Integer(8);   
        private static final ArrayList<String> alStaticFinalVar = new ArrayList<String>();   
  
        private void test() {   
                System.out.println("-------------值处理前----------\r\n");   
                System.out.println("strStaticFinalVar=" + strStaticFinalVar + "\r\n");   
                System.out.println("strStaticVar=" + strStaticVar + "\r\n");   
                System.out.println("strFinalVar=" + strFinalVar + "\r\n");   
                System.out.println("intStaticFinalVar=" + intStaticFinalVar + "\r\n");   
                System.out.println("integerStaticFinalVar=" + integerStaticFinalVar + "\r\n");   
                System.out.println("alStaticFinalVar=" + alStaticFinalVar + "\r\n");   
  
  
                //strStaticFinalVar="哈哈哈哈";        //错误，final表示终态,不可以改变变量本身.   
                strStaticVar = "哈哈哈哈";                //正确，static表示类变量,值可以改变.   
                //strFinalVar="呵呵呵呵";                    //错误, final表示终态，在定义的时候就要初值（哪怕给个null），一旦给定后就不可再更改。   
                //intStaticFinalVar=2;                        //错误, final表示终态，在定义的时候就要初值（哪怕给个null），一旦给定后就不可再更改。   
                //integerStaticFinalVar=new Integer(8);            //错误, final表示终态，在定义的时候就要初值（哪怕给个null），一旦给定后就不可再更改。   
                alStaticFinalVar.add("aaa");        //正确，容器变量本身没有变化，但存放内容发生了变化。这个规则是非常常用的，有很多用途。   
                alStaticFinalVar.add("bbb");        //正确，容器变量本身没有变化，但存放内容发生了变化。这个规则是非常常用的，有很多用途。   
  
                System.out.println("-------------值处理后----------\r\n");   
                System.out.println("strStaticFinalVar=" + strStaticFinalVar + "\r\n");   
                System.out.println("strStaticVar=" + strStaticVar + "\r\n");   
                System.out.println("strFinalVar=" + strFinalVar + "\r\n");   
                System.out.println("intStaticFinalVar=" + intStaticFinalVar + "\r\n");   
                System.out.println("integerStaticFinalVar=" + integerStaticFinalVar + "\r\n");   
                System.out.println("alStaticFinalVar=" + alStaticFinalVar + "\r\n");   
        }   
  
        public static void main(String args[]) {   
                new TestStaticFinal().test();   
        }   
}

运行结果如下：

-------------值处理前----------

strStaticFinalVar=aaa

strStaticVar=null

strFinalVar=null

intStaticFinalVar=0

integerStaticFinalVar=8

alStaticFinalVar=[]

-------------值处理后----------

strStaticFinalVar=aaa

strStaticVar=哈哈哈哈

strFinalVar=null

intStaticFinalVar=0

integerStaticFinalVar=8

alStaticFinalVar=[aaa, bbb]

Process finished with exit code 0

        看了上面这个例子，就清楚很多了，但必须明白：通过static final修饰的容器类型变量中所“装”的对象是可改变的。这是和一般基本类型和类类型变量差别很大的地方。