1、方法局部变量final问题

在**方法**内部定义内部类时，内部类如果需要访问方法内的局部变量，则此局部变量必须要用final修饰。

原因：

当方法被调用运行完毕之后，局部变量死亡。但内部类**对象**可能还存在，直到没有被引用时才会死亡。此时就可能出现一种情况：内部类要访问一个不存在的局部变量。

解决这一问题的办法就是使用final修饰局部变量，通过将final局部变量"**复制**"一份，复制品将作为方法中内部类中的**数据成员**，这时方法的内部类访问的其实是**复制品**而不是原本的局部变量（原本的局部变量随方法调用完毕而死亡了）。final的作用：保证变量赋值后不能再修改，保证了复制品与原始变量的一致。

final局部变量的生命周期仍然是局部的，方法执行完毕即死亡。其复制品成为了内部类的数据成员而存在。即：复制保证生命周期延续，final保证引用一致。

只有static变量生命周期才是全局的。

当局部变量是final时：

若是基本数据类型，由于其值不变，访问复制品与访问原始值是一样的，语义效果相同【若不是final就无法保证复制品与原始变量保持一致，因为在方法中改变的是原始变量，在内部类中改变的是复制品】

若是引用类型，由于引用值不变（指向同一个对象），因而复制品与原始的引用变量都永远指向同一个对象，语义效果也是相同的。

Thinking in Java:

如果定义一个匿名内部类，并且希望它使用一个在其外部定的对象，那么编译器会要求其参数引用是final 的。经研究，Java虚拟机的实现方式是：编译器会探测局部内部类中是否有直接使用外部定义变量的情况，如果有访问就会定义一个同类型的变量，然后在构造方法中用外部变量给自己定义的变量赋值。final只是保证外部变量和内部定义的变量值相同。