1.明确方法功能，精确（而不是近似）地实现方法设计。如果一个功能将在多处实现，即使只有两行代码，也应该编写方法实现。

说明：

虽然为仅用一两行就可完成的功能去编方法好象没有必要，但用方法可使功能明确化，增加程序可读性，亦可方便维护、测试。

2.应明确规定对接口方法参数的合法性检查应由方法的调用者负责还是由接口方法本身负责，缺省是由方法调用者负责。

说明：

对于模块间接口方法的参数的合法性检查这一问题，往往有两个极端现象，即：要么是调用者和被调用者对参数均不作合法性检查，结果就遗漏了合法性检查这一必要的处理过程，造成问题隐患；要么就是调用者和被调用者均对参数进行合法性检查，这种情况虽不会造成问题，但产生了冗余代码，降低了效率。

3.明确类的功能，精确（而不是近似）地实现类的设计。一个类仅实现一组相近的功能。说明：划分类的时候，应该尽量把逻辑处理、数据和显示分离，实现类功能的单一性。

4.所有的数据类必须重载toString() 方法，返回该类有意义的内容。说明：父类如果实现了比较合理的toString() ， 子类可以继承不必再重写。

5.数据库操作、IO操作等需要使用结束close()的对象必须在try -catch-finally 的finally中close()。

6.异常捕获后，如果不对该异常进行处理，则应该记录日志（针对后台）。

说明：若有特殊原因必须用注释加以说明。

7.自己抛出的异常必须要填写详细的描述信息。

说明：便于问题定位。

8. 在程序中使用异常处理还是使用错误返回码处理，根据是否有利于程序结构来确定，并且异常和错误码不应该混合使用，推荐使用异常。说明：一个系统或者模块应该统一规划异常类型和返回码的含义。但是不能用异常来做一般流程处理的方式，不要过多地使用异常，异常的处理效率比条件分支低，而且异常的跳转流程难以预测。

9.避免使用不易理解的数字，用有意义的标识来替代。涉及物理状态或者含有物理意义的常量，不应直接使用数字，必须用有意义的静态变量来代替。

10.数组声明的时候使用 int[] index ，而不要使用 int index[] 。说明：

11.异常捕获尽量不要直接 catch (Exception ex) ，应该把异常细分处理。

12.不要使用难懂的技巧性很高的语句，除非很有必要时。说明：高技巧语句不等于高效率的程序，实际上程序的效率关键在于算法。