# 第9章 Effective Java

45 将局部变量的作用域最小化：要使局部变量的作用域最小化，最有力的方法就是在第一次使用它的地方声明；每一个局部变量的声明都应该包含一个初始化表达式；

46 for-each循环优先与传统的for循环：简洁，避免BUG，非查询情况下不要使用for-each；

47 了解与使用类库：例子就是random方法的缺点，使用标准类库有很多好处；每个程序员都应该熟悉java.lang、java.util、java.io中的内容；

48 如果需要精确的答案，避免使用float与double：因为float与double在计算机中不能完美表示数；使用BigDecimal或者int/long；

49 基本类型优于装箱基本类型：java的类型系统由2部分组成基本类型与引用类型，基本类型还有装箱基本类型；大多数的情况都要使用基本类型，当作为集合中的元素的时候使用装箱基本类型，范型中，在反射使用中；

50 如果其他类型更合适，尽量避免使用字符串：字符串很通用，字符串不适合代替其他的值类型，ThreadLocal；

51 当心字符串连接的性能，为连接n个字符串而重复发的使用字符串连接操作符，需要n的平方级的时间，使用StringBuilder代替拼接；

52 通过接口引用对象：如果有合适的接口类型存在，对于参数、返回值、变量域，都应该使用接口类型进行声明，方便更换实现；

53 接口优先于反射机制：；

57.只针对异常的情况才使用异常：异常是用于不正常的情况，JVM不会对异常代码进行优化；异常容易Bug；异常只能用于异常的情况，不能用于正常的控制流；设计良好的API不应该强迫它的客户端为了正常的控制流而使用异常；

58 对可恢复的情况使用受检异常，对编程错误使用运行时异常，java提供的3种可抛出的异常：受检异常、运行时异常与错误；如果希望调用者能够适当的恢复，使用受检异常；运行时异常表明编程错误（前提违例）；

59 避免不必要的使用受检的异常：不知道怎么处理受检异常，那就抛出未受检的异常；

60 优先使用标准的异常：；

61 抛出与抽象相对应的异常：更高层的实现应该捕获底层的异常，同时抛出可以按照高层抽象进行解释的异常，称为异常转译；异常链，底层异常传入到高层异常的cause属性，包装起来；最好的方法是上层方法做一些判断来避开底层异常；

62 每个方法抛出的异常都要有文档：始终要单独的声明受检的异常，并用javadoc标记，准确记录下抛出每个异常的条件，如果抛出多个异常，依次列出，不要throw Exception或者throw Throwable；

63 在细节消息中包含能捕获失败的信息：为了捕获失败，异常的细节信息应该包含所有对该异常有贡献的参数和域的值；

64 努力使失败保持原子性：一般而言，失败的方法调用应该使对象保持在被调用之前的状态，具有这种属性的方法具有失败原子性；

65 不要忽略异常：至少说明下错误的原因，不要什么都不做；