第8章 通用程序设计

2018年7月17日

20:44

第45条：将局部变量的作用域最小化

**作用域最小化的最有力方法是在第一次使用它的地方声明。**

**几乎每个局部变量的声明都应该包含一个初始化表达式。**

**使用for循环比使用while更能限制局部变量的作用域范围。**

**使方法小而集中也可以将局部变量作用域最小化。**

**第46条：for-each循环优于传统的for循环**

for-each循环在简介性和预防Bug方面有着传统for循环无法比拟的优势。

1. 过滤

如果需要遍历集合，并删除选定的元素，就需要使用显式的迭代器，以便可以调用它的remove方法。

1. 转换

如果需要遍历列表或者数组，并取代它部分或者全部的元素值，就需要列表迭代器或者数组索引，以便设定袁术的值

1. 平行迭代

如果需要并行地遍历多个集合，就需要显式地控制迭代器或者索引变量，一年所有迭代器或者索引变量都可以得到同步前移

**第47条：了解和使用类库**

通过使用标准库，可以充分利用这些编写标准库的专家的知识，以及在你之前的其他人的使用经验。

不要重新发明轮子。

第48条：如果需要精确的答案，请避免使用float和double。

float和doub类型尤其不适合用于货币计算，因此要让一个float或者double的精确地表示小数使不可能的。

使用BigDecimal、int或者long进行货币计算。

需要精确答案的计算任务，如果性能非常关键，可以使用int或者long。

**第49条：基本类型优先于装箱基本类型**

**基本类型只有值，而装箱基本类型则具有与它们的值不同的同一性。**

当使用==操作符比较两个装箱基本类型时，它做了个同一性的比较，而结果不是我们所期望的。

**基本类型只有功能完备的值，而每个装箱基本类型还有非功能值：null。**

未初始化的装箱基本类型拆箱时，会导致空指针异常。

**基本类型通常比装箱基本类型更节省时间和空间。**

当程序装箱了基本类型时，会导致高开销和不必要的对象创建。

**自动装箱减少了使用装箱基本类型的繁琐性，但是并没有减少它的风险。**

第50条：如果其他类型更适合，则尽量避免使用字符串

**字符串不适合代替其他的值类型。**

**字符串不适合代替枚举类型。**

**字符串也不适合代替能力表或者聚集类型。**

**如果可以使用更加合适的数据类型，或者可以编写更加合适当的数据类型，就该避免使用字符串来表示对象。**

**第51条：当心字符串连接的性能**

为连接n个字符串而重复地使用字符串连接操作符，需要n的平方级的时间。

为了获得可以接受的性能，请使用StringBuilder代替String。

第52条：通过接口引用对象

如果你养成了使用接口作为类型的习惯，你的程序将会更加灵活。

如果没有合适的接口存在,完全可以用类而不是接口来引用对象。

如果对象属于基于类的框架，就应该用相关的基类（往往是抽象类）来引用这个对象。

类实现了接口，但是它提供了接口不存在的额外方法，如LinkedhashMap、CompletableFuture。

第53条：接口优先于反射机制

反射机制的代价：

* 丧失了编译时类型检查的好处。
* 执行反射访问所需要的代码非常笨拙和冗长。
* 性能损失。

通常，普通应用程序在运行时不应该以反射方式访问对象。

如果编写编写的程序要与编译时未知的类一起工作，就应该使用反射机制来实例化对象，而访问对象时

则使用编译时已知的某个接口或者超类。

**第54条：谨慎地使用本地方法**

使用本地方法来提高性能不值得提倡。

因为本地语言是不安全的，所以，使用本地方法的应用程序也不能免受内存毁坏错误的影响。

在进入和退出本地代码时，需要相关的固定开销，所以可能降低性能。

第55条：谨慎地进行优化

要努力编写更好的程序而不是快的程序。

努力避免那些限制性能的设计决策。

要考虑API设计决策的性能后果。

第56条：遵守普遍接受的命名惯例

字面惯例：

|  |  |
| --- | --- |
| 标识符类型 | 例子 |
| 包 | com.google.inject, org.joda.time.format |
| 类或者接口 | Timer, FutureTask, LinkedHashMap, HttpServlet |
| 方法或者域 | remove, ensureCapacity, getCrc |
| 常量域 | MIN\_VALUE, NEGATIVE\_INFINITY |
| 局部变量 | I, xref, houseNumber |
| 类型参数 | T, E, K, V, X T1, T2 |

应该把标准的命名惯例当作一种内在的机制来看待，并且学着用它们作为第二特性。