附录 A

经验原则总结

第 2 章　类和对象：面向对象范型的建材

经验原则 2.1　所有数据都应当隐藏在它所在的类内部。

经验原则 2.2　类的使用者必须依赖类的公有接口，但类不能依赖它的使用者。

经验原则 2.3　尽量减少类的协议中的消息。

经验原则 2.4　实现所有类都理解的最基本公有接口［例如：拷贝操作（深拷贝与浅拷贝）、相等性判断、正确输出内容、从 ASCII 描述解析等等］。

经验原则 2.5　不要把实现细节（例如放置共用代码的私有函数）放到类的公有接口中。

经验原则 2.6　不要以用户无法使用或不感兴趣的东西污染类的公有接口。

经验原则 2.7　类之间应该零耦合，或者只有导出耦合关系。也即，一个类要么同另一个类毫无关系，要么只使用另一个类的公有接口中的操作。

经验原则 2.8　类应当只表示一个关键抽象。

经验原则 2.9　把相关的数据和行为集中放置。

经验原则 2.10　把不相关的信息放在另一个类中（也即：互不沟通的行为）。

经验原则 2.11　确保你为之建模的抽象概念是类，而不只是对象扮演的角色。

第 3 章 应用程序布局：面向动作与面向对象

经验原则 3.1 在水平方向上尽可能统一地分布系统功能，也即：按照设计，顶层类应当统一地共享工作。

经验原则 3.2 在你的系统中不要创建全能类/对象。对名字包含 Driver、Manager、System、Subsystem 的类要特别多加小心。

经验原则 3.3 对公共接口中定义了大量访问方法的类多加小心。大量访问方法意味着相关数据和行为没有集中存放。

经验原则 3.4 对包含太多互不沟通的行为的类多加小心。互不沟通的行为是指在类的数据成员的一个真子集上进行操作的方法。全能类经常有很多互不沟通的行为。

经验原则 3.5 在由同用户界面交互的面向对象模型构成的应用程序中，模型不应该依赖于界面，界面则应当依赖于模型。

经验原则 3.6 尽可能地按照现实世界建模（我们常常为了遵守系统功能分布原则、避免全能类原则以及集中放置相关数据和行为的原则而违背这条原则）。

经验原则 3.7 从你的设计中去除不需要的类。

经验原则 3.8 去除系统外的类。

经验原则 3.9 不要把操作变成类。质疑任何名字是动词或者派生自动词的类，特别是只有一个有意义行为（即：不考虑存取和打印成员的行为）的类。考虑一下那个有意义的行为是否应当迁移到已经存在或者尚未发现的某个类中。

经验原则 3.10 我们在创建应用程序的分析模型时常常引入代理类。在设计阶段，我们常会发现很多代理是没有用的，应当去除。

第 4 章 类和对象的关系

经验原则 4.1 尽量减少类的协作者的数量。

经验原则 4.2 尽量减少类和协作者之间传递的消息的数量。

经验原则 4.3 尽量减少类和协作者之间的协作量，也即：减少类和协作者之间传递的不同消息的数量。

经验原则 4.4　尽量减小类的扇出，也即：减少类定义的消息数和发送的消息数的乘积。

经验原则 4.5　如果类包含另一个类的对象，那么包含类应当给被包含的对象发送消息。也即：包含关系总是意味着使用关系。

经验原则 4.6　类中定义的大多数方法都应当在大多数时间里使用大多数数据成员。

经验原则 4.7　类包含的对象数目不应当超过开发者短期记忆的容量。这个数目常常是 6。

经验原则 4.8　让系统功能在窄而深的继承体系中垂直分布。

经验原则 4.9　实现语义约束时，最好根据类定义来实现。这常常会导致类泛滥成灾，在这种情况下约束应当在类的行为中实现，通常是在构造函数中实现，但不是必须如此。

经验原则 4.10　当在类的构造函数中实现语义约束时，把约束测试放在构造函数领域所允许的尽量深的包含层次中。

经验原则 4.11　约束所依赖的语义信息如果经常改变，那么最好放在一个集中式的第三方对象中。

经验原则 4.12　约束所依赖的语义信息如果很少改变，那么最好分布在约束所涉及的各个类中。

经验原则 4.13　类必须知道它包含什么，但是不能知道谁包含它。

经验原则 4.14　共享字面范围（也就是被同一个类所包含）的对象相互之间不应当有使用关系。

第 5 章　继承关系

经验原则 5.1　继承只应被用来为特化层次结构建模。

经验原则 5.2　派生类必须知道它们的基类，基类不应当知道关于它们的派生类的任何信息。

经验原则 5.3　基类中的所有数据都应当是私有的，不要使用保护数据。

经验原则 5.4　在理论上，继承层次体系应当深一点，越深约好。

经验原则 5.5　在实践中，继承层次体系的深度不应当超出一个普通人的短期记忆能力。一个广为接受的深度值是 6。

经验原则 5.6　所有的抽象类都应当是基类。

经验原则 5.7 所有的基类都应当是抽象类。

经验原则 5.8 把数据、行为和/或接口的共性尽可能地放到继承层次体系的高端。

经验原则 5.9 如果两个或更多个类共享公共数据（但没有公共行为），那么应当把公共数据放在一个类中，每个共享这些数据的类都包含这个类。

经验原则 5.10 如果两个或更多个类有共同的数据和行为（就是方法），那么这些类的每一个都应当从一个表示了这些数据和方法的公共基类继承。

经验原则 5.11 如果两个或更多个类共享公共接口（指的是消息，而不是方法），那么只有它们需要被多态地使用时，它们才应当从一个公共基类继承。

经验原则 5.12 对对象类型的显式地分情况分析一般是错误的。在大多数这样的情况下，设计者应当使用多态。

经验原则 5.13 对属性值的显式地分情况分析常常是错误的。类应当解耦合成一个继承层次结构，每个属性值都被变换成一个派生类。

经验原则 5.14 不要通过继承关系来为类的动态语义建模。试图用静态语义关系来为动态语义建模会导致在运行时切换类型。

经验原则 5.15 不要把类的对象变成派生类。对任何只有一个实例的派生类都要多加小心。

经验原则 5.16 如果你觉得需要在运行时创建新的类，那么退后一步以认清你要创建的是对象。现在，把这些对象概括成一个类。

经验原则 5.17 在派生类中用空方法（也就是什么都不做的方法）来覆写基类中的方法应当是非法的。

经验原则 5.18 不要把可选包含同对继承的需要相混淆。把可选包含建模成继承会带来泛滥成灾的类。

经验原则 5.19 在创建继承层次时，试着创建可复用的框架，而不是可复用的组件。

第 6 章 多重继承

经验原则 6.1 如果你在设计中用到了多重继承，先假设你犯了错误。如果没犯错误，你需要设法证明。

经验原则 6.2 只要在面向对象设计中用到了继承，问自己两个问题：

1．派生类是否是它继承的那个东西的一个特殊类型？

2．基类是不是派生类的一部分？

经验原则 6.3　如果你在一个面向对象设计中发现了多重继承关系，确保没有哪个基类实际上是另一个基类的派生类。

第 7 章　关联关系

经验原则 7.1　在面向对象设计中如果你需要在包含关系和关联关系间做出选择，请选择包含关系。

第 8 章　与特定类相关的数据及行为

经验原则 8.1　不要把全局数据或全局函数用于类的对象的簿记工作。应当使用类变量或者类方法。

第 9 章　面向对象物理设计

经验原则 9.1　面向对象设计者不应当让物理设计准则来破坏他们的逻辑设计。但是，在对逻辑设计做出决策的过程中我们经常用到物理设计准则。

经验原则 9.2　不要绕开公有接口去修改对象的状态。