Ch5

理想的设计特征

最小的复杂度

易于维护

松散耦合

可扩展性

可重用性

高扇入

低扇出

可移植性

精简性

层次性

标准技术

设计中的挑战

设计是一个险恶的问题

设计是个了无章法的过程

设计就是确定取舍和调整顺序的过程

设计受到诸多限制

设计是不确定的

设计是一个启发式过程

设计是自然而然形成的

设计实践

迭代

分而治之

自上而下和自下而上的设计方法

建立试验性原型

合作设计

记录设计成果

设计的层次

软件系统

分解为子系统或包

分解为类

分解成子程序

子程序内部的设计

Ch6

ADT是指一些数据以及对这些数据所进行的操作的集合。

模块化设计的目标是使每个子程序都成为一个“黑盒子”：

* 你知道进入盒子和从盒子里出来的是什么，却不知道里边发生什么
* 通过接口的定义明确其功能

类=ADT+继承+多态

使用ADT的益处：

隐藏实现细节

改动不会影响到整个程序

让接口能提供更多信息

更容易提高性能

让程序的正确性更显而易见

程序更具自我说明性

无须在程序内到处传递数据

操作实体而不是底层

良好的类接口

——之良好的抽象

理解所实现的抽象

提供成对的服务

把不相关的信息转移到其他类

让接口可编程而不是表达语义

谨防在修改时破坏接口的抽象

不要添加与接口抽象不一致的公用成员

同时考虑抽象性和内聚性

良好的类接口

——之良好的封装

尽可能地限制类和成员的可访问性

不要公开暴露成员数据

避免把私用的实现细节放入类的接口中

不要对类的使用者作出任何假设

避免使用友元类

不把子程序归入公开接口

个要警惕从语义上破坏封装性

留意过于紧密的耦合关系

ch8

创建类的理由

对现实世界中的对象建模

对抽象对象建模

降低复杂度

隔离复杂度

隐藏实现细节

限制变化所影响的范围

隐藏全局数据

让参数传递更顺畅

创建中心控制点

让代码更易于重用

为程序族做计划

把相关的操作放到一起

实现特定的重构

防御式编程

检查所有来源于外部的数据的值

检查子程序所有输入参数的值

决定如何处理错误的输入数据

断言：让程序在运行时进行自检的代码

用错误处理代码来处理预期会发生的状况，用断言来处理绝不应该发生的状况。

避免把需要执行的代码放到断言中

用断言来注解并验证前条件和后条件

对于高健壮性的代码应先使用断言再处理错误

异常：把代码中的错误或异常事件传递给调用方代码的一种特殊手段

用异常通知程序的其他部分发生了不可忽略的错误

只在真正里外的情况下才抛出异常

不能用异常来推卸责任

避免在构造函数和析构函数中抛出异常

在恰当的抽象层次抛出异常

……

Ch7

创建子程序的理由

降低复杂度

引入中间、易懂的抽象

避免代码重复

支持子类化

隐藏顺序

隐藏指针操作

提高可移植性

简化复杂的布尔判断

改善性能

隔离复杂度

隐藏实现细节

限制变化所影响的范围

隐藏全局数据

创建中心控制点

让代码更易于重用

实现特定的重构

内聚性

功能内聚（一项操作）

顺序内聚（一系列共享数据的步骤且）

通信内聚（共同的数据）

临时内聚（同时执行）

过程内聚（一系列步骤）

逻辑内聚（传入操作符选择执行若干操作）

巧合的内聚（无关）