Object Oriented Analysis & Design
面向对象分析与设计

Lecture_06 从分析到设计

主讲: 姜宁康 博士

3、契约式设计 Design by Contract

- 可信软件设计的基础思想
- 谚语: When ideas fail, words come in very handy !

  他人译文“殚思竭虑之时，文字将成为利器”
  本人认为“当想法失败时，总会出来许多理由辩解”

3.1 问题的引入

For a given number, How to find its reciprocal 求一个数的倒数

Example,

```
void  _test {
        ...
        Y = _f(X);
        ....
}
float _f (float x) {
   return  1/x;
}
```

由谁负责系统的可靠性?

3.2 Contract (契约) History

- Tony Hoare，把“操作契约 Operation contract”引入了计算机科学的“形式化规范说明formal specification”研究
  - In the mid-1960s to develop an ALGOL 60 compiler
  - Read Bertrand Russell's （伯特兰·罗素） Introduction to Mathematical Philosophy, which introduced him to the idea of axiomatic theory（公理）and assertions（断言）
    - Assertions: (pre- and post-conditions)
- Peter Lucas, in 1974, VDM(Vienna Definition Method), for PL/1 compiler, at IBM Lab
  - VDM, A method applied operation contract formal specifications and rigorous proof theory

3.2 Contract (契约) History

- In the 1980s, Bertrand Meyer, compiler writer
  - the OO language: Eiffel
  - started to promote the use of pre- and post-condition assertions as first-class elements within his Eiffel language
  - 契约式设计 Design by Contract (DBC)
    - 细粒度软件类的操作需要强调“契约”，而不是针对整个系统的操作（大粒度软件类）
    - 推动了“不变量 invariant”的概念
      - 在操作执行前、执行后都不变的概念
- 1990s, Grady Booch
  - 把“契约 contracts”引入对象操作
  - 详细的用例描述中，重要的操作
    - Pre- / Post- condition

3.3 契约式设计DBC (Design By Contract)

- 名词解释
  - 客户端、客户 Client：需要另一方提供服务
  - 服务端、服务器 Server：为其它方提供服务
- 一份契约承载了相互间（client/server) 的义务与利益
  - 客户端只有在能够满足服务端的“前置条件”的情况下，才能调用服务端的服务 The client should only call a routine（server） when the routine’s pre-condition is respected
  - 服务端在结束服务后，必须保证满足其后置条件 The routine ensures that after completion its post-condition is respected
- 比喻
  - 顾客到商店买食品。必须是真钱，不是假币。食品必须卫生、安全、符合质量要求

3.3 契约式设计DBC

- 断言 assertion
  - 在类的代码中，加入一些断言，则定义了契约 Each class defines a contract, by placing assertions inside the code

```
assert    0 < value;
assert    0 < value:"value="+value;
assert    ref != null:"ref doesn't equal null";
assert    isBalanced();
```

  - 断言仅仅是一些逻辑表达式 Assertions are just Boolean expressions
  - 断言不影响程序的执行 Assertions have no effect on execution
  - 断言可以被评估，或者忽略 Assertions can be checked or ignored

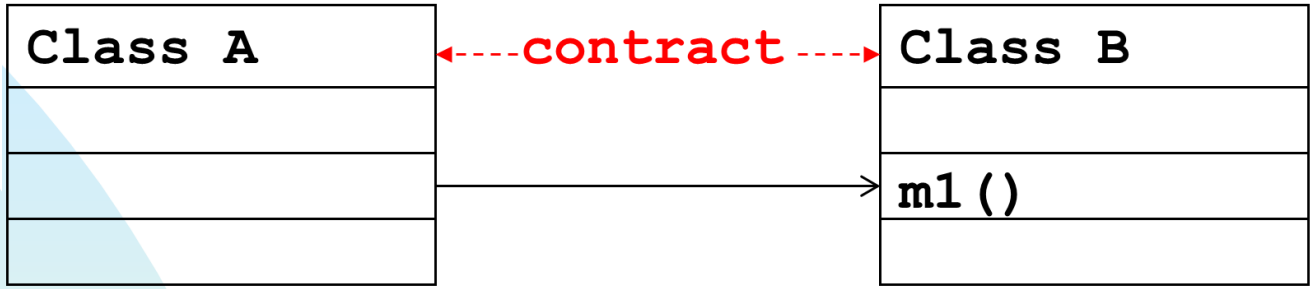
3.3 契约式设计DBC

- 每个功能定义了一个前置条件、一个后置条件 Each feature is equipped with a precondition and a postcondition

```
double sqrt (double x) {

        require
                x >= 0;
        do
                ...
                ...

        ensure
                result * result == x;
        end

}
```

3.3 契约式设计DBC

Consider the relationship between a class and its clients:



DbC views this as a formal agreement, or contract, expressing each party's rights and obligations

claims

responsibilities

Without such a precise definition we cannot have a significant degree of trust in large software systems

Contract violations lead to run-time errors, i.e. exceptions.

So we will also consider exception handling

3.4 Software Correctness

假定有一个人拿着一个程序到你面前问:
“这个程序正确吗? Is this program correct”

这个问题有意义的前提是:程序应该完成什么功能有一个精确的描述 This question is meaningful only if there is a precise description of what the program is supposed to do

不管这个程序的大小 regardless of the size of the program

这样的一个描述,就是规格说明 specification, 规格说明必须精确,否则不可能推理出是否正确

例1,女孩给男朋友打电话:如果你到了,我还没有到,你就等着吧;如果我到了,你还没有到,你就等着吧!

例2,单身的原因: 原来是喜欢一个人,现在是喜欢一个人

不精确的表达、描述!

3.5 规格说明Specification的形式化表示

Let A be some operation

e.g. a single instruction or a whole method

A correctness formula is an expression of the form:
{P} A {Q}

P and Q are assertions (logical predicates):
P is the pre-condition, Q the post-condition

It means:

"Any execution of A, starting in a state where P holds, will terminate in a state where Q holds" 每次要执行A，满足P才能开始，结束的时候，Q必须得到满足

E.g. {x >= 9} x := x + 5 {x >=13}

3.6 前置-条件、 后置- 条件

pre-condition — { P } A { Q } — **post-condition**

– expresses the **constraints** under which a method will function properly

– applies to **all** calls of the method

both from within the class and from clients

A **correct** system will **never** execute a call in a state that does not satisfy the pre-condition of the called method.

一个正确的系统永远不会在被调用操作的前置条件不满足的情况下开始执行

– expresses properties of the state resulting from a method's execution

– expresses a **guarantee** that the method will yield a state satisfying certain properties, **assuming** it has been called with the pre-condition satisfied

A **correct** system will **always** deliver a state that satisfies the post-condition of the called method

一个正确的系统永远在调用结束时处于满足被调用操作的后置条件的状态

3.7 类不变量 Class Invariants

前置条件与后置条件描述了单个操作的特性 Preconditions and post-conditions describe the properties of individual methods

类不变量是类实例的全局特性，在所有的操作中应得到遵守 A class invariant is a global property of the instances of a class, which must be preserved by all methods

类不变量是类定义的一个断言 A class invariant is an assertion in the class definition

E.g. 一个堆栈类有如下的类不变量

count >= 0 and
count <= capacity and
stack_is_empty = (count = 0)

Class Stack
Attrs …
Methods …
Inv:…

An invariant may link attributes (e.g. count and capacity), or functions (e.g. stack_is_empty), or attributes and functions

小结 契约式设计

- 软件可靠性需要服务的提供方与客户方都有精确的规格说明
  Software reliability requires precise specifications which are honoured by both the supplier and the client
- 契约式设计DbC使用断言（前置条件、后置条件、不变量等）作为供/需双方之间的契约 uses assertions (pre and postconditions, invariants) as a contract between supplier and client
- 注：更细致的内容，属于“形式化”建模、设计、验证等，超出本课程范围

- 本讲结束