Recap for The Art of Naming

Recap for The Art of Naming “命名”课回顾

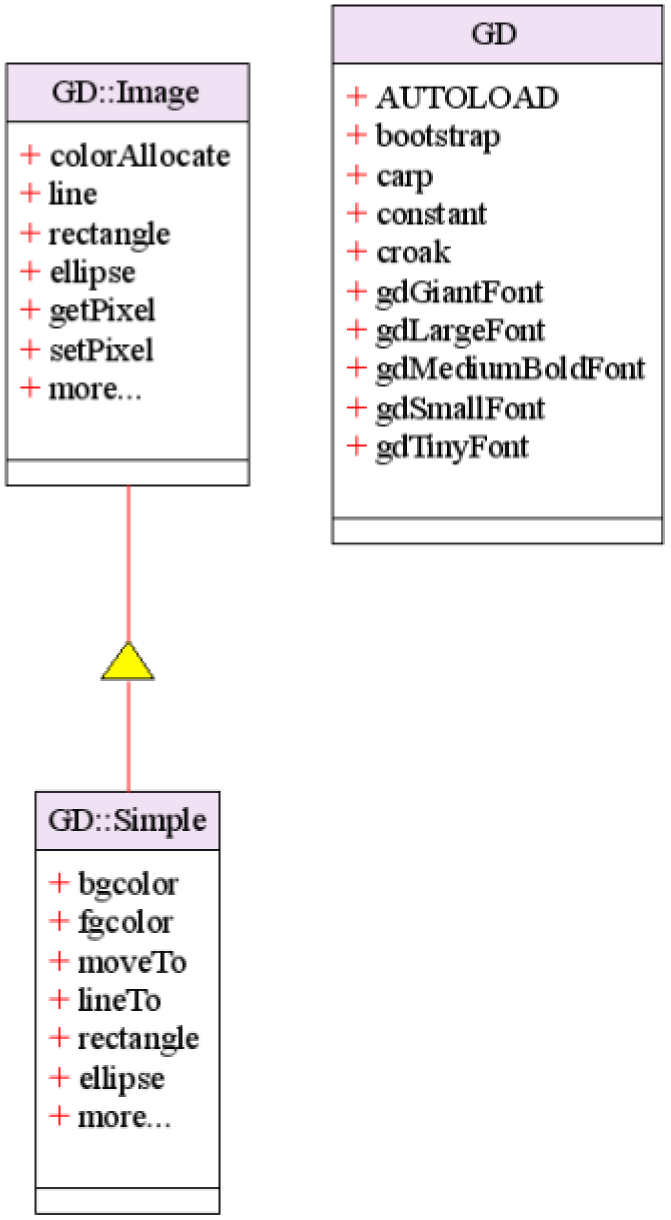
☺ *Agent Zhang（章亦春）* ☺

2006.10

To *simplify* a class interface,
use of inheritance is *deprecated*.

若想简化类的接口，
则不应使用继承。

A *bad* example in perl 5 ⇒


GD::Image
+ colorAllocate
+ line
+ rectangle
+ ellipse
+ getPixel
+ setPixel
+ more...
GD
+ AUTOLOAD
+ bootstrap
+ carp
+ constant
+ croak
+ gdGiantFont
+ gdLargeFont
+ gdMediumBoldFont
+ gdSmallFont
+ gdTinyFont
GD::Simple
+ bgcolor
+ fgcolor
+ moveTo
+ lineTo
+ rectangle
+ ellipse
+ more...

```perl
# This works:
use GD::Simple;

my $img = new GD::Simple(40, 50);
$img->bgcolor('white');
$img->fgcolor('red');
$img->rectangle(10, 10, 50, 50);
```

```perl
# But this doesn't work, since
# setPixel is a method derived
# directly from GD::Image
use GD::Simple;

my $img = new GD::Simple(40, 50);
$img->bgcolor('white');
$img->fgcolor('red');
$img->setPixel(10, 10, 'red');
```

```perl
# We have to degrade to the harder way:
use  GD::Simple;

my  $img  =  new  GD::Simple(40, 50);
my  $red  =  $img ->colorAllocate(255, 0, 0);
$img ->setPixel(10, 10, $red );
```

```perl
# The Perl 5 way:
print "hello, world!\n";
```

```
# The Perl 5 way:
print "hello, world!\n";
```

5 characters

```
# The Perl 6 way:
say "hello, world!";
```

```
# The Perl 6 way:
say "hello, world!";
```

3 characters

☺ That's the Huffman coding *principle*

这正是哈夫曼编码原理。

✓ Rant on the software vendors and show them where the technology *really* wants to go!

向那些软件商怒吼，
并向他们指出技术真正想去的地方！

RANT
ON!

Broad background knowledge is *very* important to *good* programmers.

宽广的背景知识对于好的程序员来说是非常重要的。

AN2000

```
<agentzh>  yeah
<agentzh>  audreyt++ # you seem to know everything.
<audreyt>  nah, not really :)
<agentzh>  hehe
<audreyt>  that's what you get from spending far too
               much time on wikipedia...
<agentzh>  ah, wikipedia++
```

```
<章亦春>    是
<章亦春>    唐凤++ # 你似乎知道所有的事情
<唐凤>      才不是呢 :)
<章亦春>    呵呵
<唐凤>      这是在 wikipedia 网站上花费了很多时间的结果……
<章亦春>    啊，wikipedia++
```

☺ Understanding the *culture* behind the technology is very important .

理解技术背后的文化是很重要的。

The *culture* of Windows feels like...

♨

Windows 的文化就感觉像……

While the *culture* of UNIX feels like...

♨

而 UNIX 的文化就感觉像……

THE AQUARIUS COLLECTION
WIZARD
MYLES PINKNEY

Java gives me the *feeling* like...

♨

Java 给我的感觉就像是……

Perl gives me the *feeling* like...

♨

Perl 给我的感觉就像是……

The *top* 3 jumps in my programming learning *curve* $\Rightarrow$

我的编程学习曲线中的三次飞跃 $\Rightarrow$

☆ *OOP* (Object-Oriented Programming)

<u>*2001.2*</u> *C++, Java, C#, encapsulation*

☆ *OOP* (Object-Oriented Programming)

2001.2 *C++, Java, C#, encapsulation*

☆ *Dynamic* Programming

2002.9 *Perl, Awk, regexes, the UNIX culture*

☆ *OOP* (Object-Oriented Programming)

2001.2 *C++, Java, C#, encapsulation*

☆ *Dynamic* Programming

2002.9 *Perl, Awk, regexes, the UNIX culture*

☆ *TDD* (Test-Driven Development)

2004.4 *C# NUnit, Perl's Test::More, Pugs*

The *potential* 4th jump at present:

The *potential* 4th jump at present:

☆ *Functional* Programming

<u>*2006.?*</u> *Haskell, CPS, Perl 6*

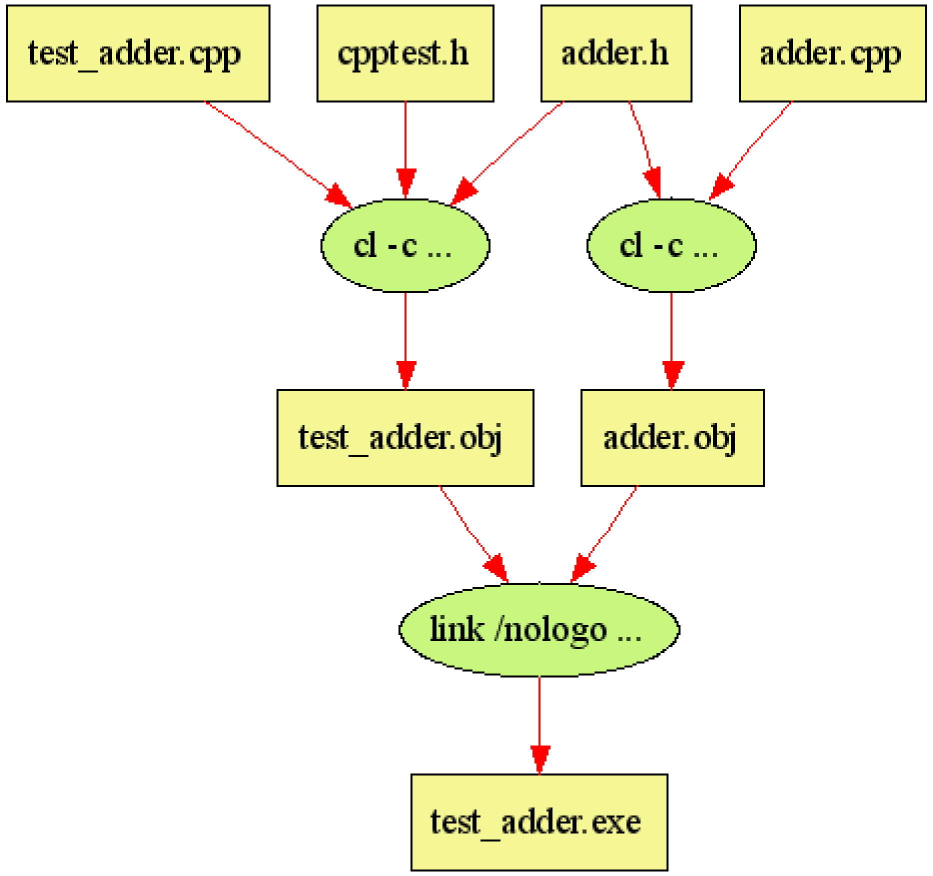
☺ A small *perlish* TDD example in C/C++ ⇒

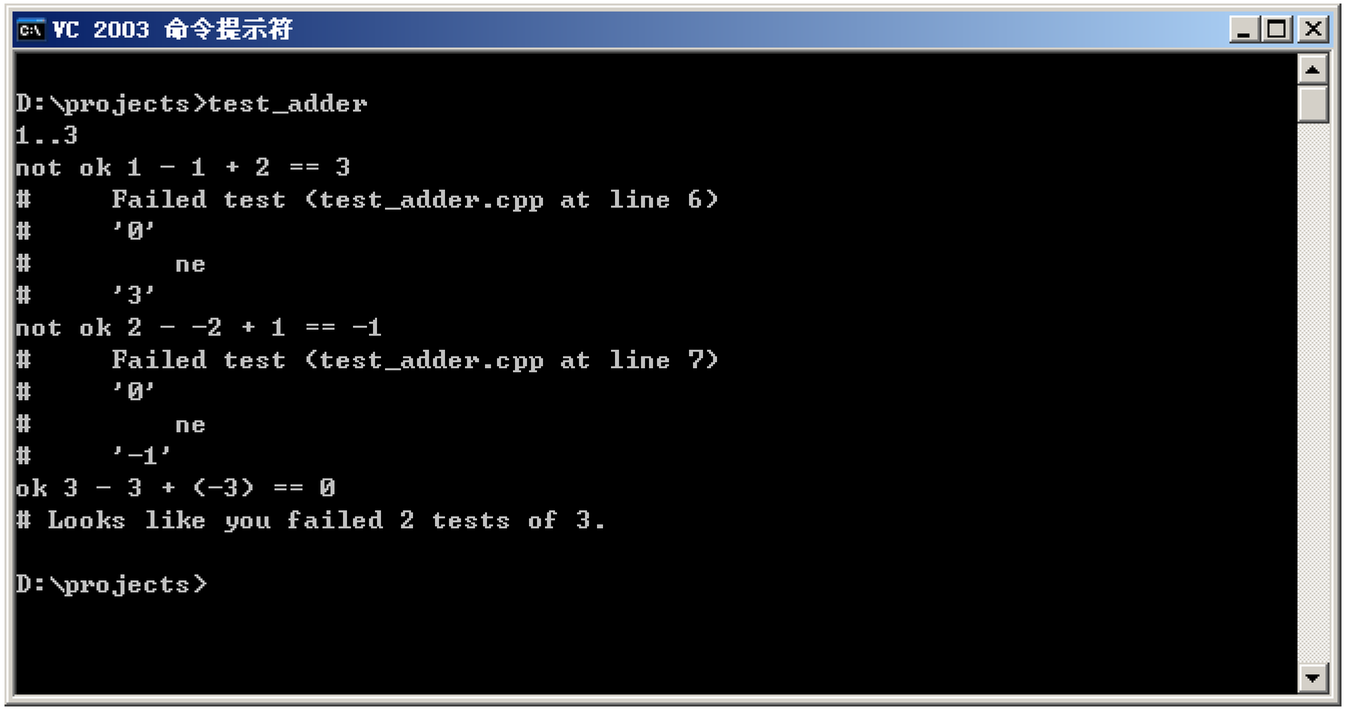
C/C++ 中的一个 perl 风味的测试驱动的示例 ⇒

```c
/* adder.h */
#ifndef _ADDER_H_
#define _ADDER_H_

int add( int a, int b);

#endif
```

```
/* adder.cpp */
#include "adder.h"

int add( int a, int b) {
    // doesn't do anything useful right now:
    return 0;
}
```

```
/* test_adder.cpp #/
#include "adder.h"
#include "cpptest.h"

int main() {
     plan(3);
     is_(add(1, 2),  3, "1 + 2 == 3");
     is_(add(-2,1), -1, "-2 + 1 == -1");
     is_(add(3,-3),  0, "3 + (-3) == 0");
     summary();
     return 0;
}
```


test_adder.cpp
cpptest.h
adder.h
adder.cpp
cl -c ...
cl -c ...
test_adder.obj
adder.obj
link /nologo ...
test_adder.exe



```
D:\projects>test_adder
1..3
not ok 1 - 1 + 2 == 3
#     Failed test (test_adder.cpp at line 6)
#     '0'
#         ne
#     '3'
not ok 2 - -2 + 1 == -1
#     Failed test (test_adder.cpp at line 7)
#     '0'
#         ne
#     '-1'
ok 3 - 3 + (-3) == 0
# Looks like you failed 2 tests of 3.

D:\projects>
```

```
D:\projects> test_adder
1..3
not ok 1 - 1 + 2 == 3
#     Failed test (test_adder.cpp at line 6)
#     '0'
#         ne
#     '3'
not ok 2 - -2 + 1 == -1
#     Failed test (test_adder.cpp at line 7)
#     '0'
#         ne
#     '-1'
ok 3 - 3 + (-3) == 0
# Looks like you failed 2 tests of 3.
```

- ☺ The first 2 tests failed *as expected*~~~

  前 2 个测试如期失败~~~

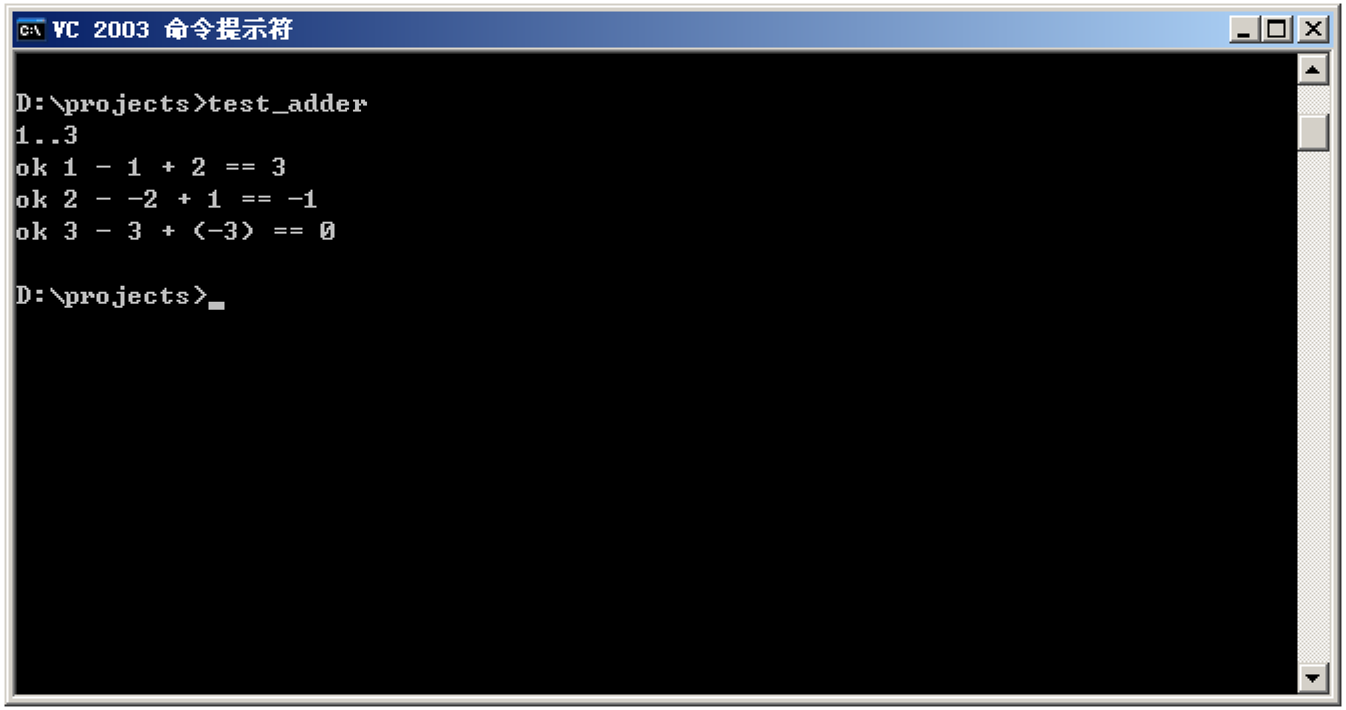
- ☺ Now it's time to *actually* implement the add function.

现在是真正给出 add 函数的实现的时候了。

```cpp
/* adder.cpp */
#include "adder.h"

int add( int a, int b) {
    // now we add the functionality:
    return a + b;
}
```

☺ Now let's rebuild the project and *rerun* the tests...

现在让我们来重新生成项目并再次运行测试……



```
D:\projects>test_adder
1..3
ok 1 - 1 + 2 == 3
ok 2 - -2 + 1 == -1
ok 3 - 3 + (-3) == 0

D:\projects>_
```

```
D:\projects>test_adder
1..3
ok 1 - 1 + 2 == 3
ok 2 - -2 + 1 == -1
ok 3 - 3 + (-3) == 0
```

Write test...

...watch test fail.

Write code...

...watch test pass.

Refactor...

...watch test pass.

Write test...

...watch test fail.

Write code...

...watch test pass.

Refactor...

...watch test pass.

编写 测试……

……观察测试 失败。

编写 代码 ……

……观察测试 通过 。

重构 ……

……观察测试 通过 。

编写 测试……

……观察测试 失败。

编写 代码 ……

……观察测试 通过 .

重构 ……

……观察测试 通过 。

Get my slides when you're in our university:

http://yxy.ujs.edu.cn/images/index.html

Contact me on the *web*!

♨

agentzh@gmail.com

These slides are *powered* by Sporx and *Takahashi*++

Thank you!

☺