Юнит-тестирование Lua-кода при помощи TAP

Илья Чесноков

DevConf

Все началось...

С маленькой программки на Lua

...несколько юнит-тестов...

lua-TestMore

Test Anything Protocol (TAP)

Test Anything Protocol (TAP)

- Текстовый интерфейс между тестовыми модулями и средой
- Во многих ЯП есть библиотеки для генерации
- Потребляется большинством CI-систем

TAP output

```
1..4
ok 1 - Input file opened
not ok 2 - First line of the input valid
ok 3 - Read the rest of the file
not ok 4 - Summarized correctly # TODO Not written yet
```

TAP output: test plan

```
1..4
ok 1 - Input file opened
not ok 2 - First line of the input valid
ok 3 - Read the rest of the file
not ok 4 - Summarized correctly # TODO Not written yet
```

TAP output: test result

```
1..4
ok 1 - Input file opened
not ok 2 - First line of the input valid
ok 3 - Read the rest of the file
not ok 4 - Summarized correctly # TODO Not written yet
```

TAP output: test number

```
1..4
ok 1 - Input file opened
not ok 2 - First line of the input valid
ok 3 - Read the rest of the file
not ok 4 - Summarized correctly # TODO Not written yet
```

TAP output: description

```
1..4
ok 1 - Input file opened
not ok 2 - First line of the input valid
ok 3 - Read the rest of the file
not ok 4 - Summarized correctly # TODO Not written yet
```

TAP output: directives

```
1..4
ok 1 - Input file opened
not ok 2 - First line of the input valid
ok 3 - Read the rest of the file
not ok 4 - Summarized correctly # TODO Not written yet
```

To plan
or not to plan?

t/plan2.lua

```
#!/usr/bin/env lua

require 'Test.More'

plan(2)
pass('First test')
pass('Second test')
pass('Third test')
```

t/plan2.lua

```
#!/usr/bin/env lua

require 'Test.More'

plan(2)
pass('First test')
pass('Second test')
pass('Third test')
```

t/plan2.lua

```
#!/usr/bin/env lua

require 'Test.More'

plan(2)
pass('First test')
pass('Second test')
pass('Third test')
```

lua t/plan2.lua

```
1..2
ok 1 - First test
ok 2 - Second test
ok 3 - Third test
```

prove -vm t/plan2.lua

```
t/plan2.lua ..
1..2
ok 1 - First test
ok 2 - Second test
ok 3 - Third test
All 2 subtests passed

Test Summary Report
-------------------
t/plan2.lua (Wstat: 0 Tests: 3 Failed: 1)
  Failed test:  3
  Parse errors: Bad plan.  You planned 2 tests but ran 3.
Files=1, Tests=3,  0 wallclock secs ( 0.02 usr +  0.00 sys =  0.02 CPU)
Result: FAIL
```

prove -vm t/plan2.lua

```
t/plan2.lua ..
1..2
ok 1 - First test
ok 2 - Second test
ok 3 - Third test
All 2 subtests passed

Test Summary Report
-------------------
t/plan2.lua (Wstat: 0 Tests: 3 Failed: 1)
  Failed test:  3
  Parse errors: Bad plan.  You planned 2 tests but ran 3.
Files=1, Tests=3,  0 wallclock secs ( 0.02 usr +  0.00 sys =  0.02 CPU)
Result: FAIL
```

t/done_testing.lua

```
require 'Test.More'

pass('First test')
pass('Second test')
pass('Third test')

done_testing()
```

prove -vm t/done_testing.lua

```
t/plan2.lua ..
ok 1 - First test
ok 2 - Second test
ok 3 - Third test
1..3
ok
All tests successful.
Files=1, Tests=3,  0 wallclock secs ( 0.02 usr +  0.00 sys =  0.02 CPU)
Result: PASS
```

Директива TODO

t/todo.lua

```
require 'Test.More'

pass('Passing test')
todo('Will fix later', 1)
fail('Failing test')

done_testing()
```

prove -vm t/todo.lua

```
t/todo.lua ..
ok 1 - Passing test
not ok 2 - Failing test # TODO Will fix later
#      Failed (TODO) test (t/todo.lua at line 7)
1..2
ok
All tests successful.
Files=1, Tests=2,  0 wallclock secs ( 0.04 usr +  0.01 sys =  0.05 CPU)
Result: PASS
```

prove -vm t/todo.lua

```
t/todo.lua ..
ok 1 - Passing test
not ok 2 - Failing test # TODO Will fix later
#     Failed (TODO) test (t/todo.lua at line 7)
1..2
ok
All tests successful.
Files=1, Tests=2,  0 wallclock secs ( 0.04 usr +  0.01 sys =  0.05 CPU)
Result: PASS
```

prove -vm t/todo.lua

```
t/todo.lua ..
ok 1 - Passing test
not ok 2 - Failing test # TODO Will fix later
#     Failed (TODO) test (t/todo.lua at line 7)
1..2
ok
All tests successful.
Files=1, Tests=2,  0 wallclock secs ( 0.04 usr +  0.01 sys =  0.05 CPU)
Result: PASS
```

Директива SKIP

t/skip_all.lua

```
require 'Test.More'

local Redis = require 'redis'
if not pcall(Redis.connect) then
  skip_all('Redis is not available')
end

-- ...

done_testing()
```

t/skip_all.lua

```
require 'Test.More'

local Redis = require 'redis'
if not pcall(Redis.connect) then
  skip_all('Redis is not available')
end

-- ...

done_testing()
```

prove -vm t/skip_all.lua

```
t/skip_all.lua .. skipped: Redis is not available
Files=1, Tests=0,  0 wallclock secs ( 0.01 usr +  0.00 sys =  0.01 CPU)
Result: NOTESTS
```

prove -vm t/skip_all.lua

```
t/skip_all.lua .. skipped: Redis is not available
Files=1, Tests=0,  0 wallclock secs ( 0.01 usr +  0.00 sys =  0.01 CPU)
Result: NOTESTS
```

testanything.org

Test.More

Test.More asserts

- ok(got, title) - проверка на true
- is(got, expected, title) - проверка got == expected
- like(got, pattern, title) - проверка на regex
- cmp_ok(this, op, that, title) - сравнение оператором **op**
- и др.

Test.More asserts

- ok(got, **title**) - проверка на true
- is(got, expected, **title**) - проверка got == expected
- like(got, pattern, **title**) - проверка на regex
- cmp_ok(this, op, that, **title**) - сравнение оператором **op**
- и др.

Пример: lib/adder.lua

```
return {

  -- add() - sums up 2 numbers and returns result
  add = function(x, y)
    return x + y
  end

}
```

t/adder.lua

```lua
-- Export all the Test.More functions as globals
require('Test.More')
local adder = require('adder')

plan(2) -- Plan 2 tests
is(adder.add(2.0001, 1.9999), 4, '2.0001 + 1.9999 == 4')
is(adder.add(1e100, 2e100), 3e100, '1e100 + 2e100 == 3e100')
```

t/adder.lua

```
-- Export all the Test.More functions as globals
require('Test.More')
local adder = require('adder')

plan(2) -- Plan 2 tests
is(adder.add(2.0001, 1.9999), 4, '2.0001 + 1.9999 == 4')
is(adder.add(1e100, 2e100), 3e100, '1e100 + 2e100 == 3e100')
```

t/adder.lua

```lua
-- Export all the Test.More functions as globals
require('Test.More')
local adder = require('adder')

plan(2) -- Plan 2 tests
is(adder.add(2.0001, 1.9999), 4, '2.0001 + 1.9999 == 4')
is(adder.add(1e100, 2e100), 3e100, '1e100 + 2e100 == 3e100')
```

t/adder.lua

```
-- Export all the Test.More functions as globals
require('Test.More')
local adder = require('adder')

plan(2) -- Plan 2 tests
is(adder.add(2.0001, 1.9999), 4, '2.0001 + 1.9999 == 4')
is(adder.add(1e100, 2e100), 3e100, '1e100 + 2e100 == 3e100')
```

prove -vm t/adder.lua

```
t/adder.lua ..
1..2
ok 1 - 2.0001 + 1.9999 == 4
not ok 2 - 1e100 + 2e100 == 3e100
#     Failed test (t/adder.lua at line 10)
#          got: 3e+100
#     expected: 3e+100
Failed 1/2 subtests

Test Summary Report
-------------------
t/adder.lua (Wstat: 0 Tests: 2 Failed: 1)
  Failed test:  2
Files=1, Tests=2,  0 wallclock secs ( 0.02 usr +  0.00 sys =  0.02 CPU)
Result: FAIL
```

Программа выросла: больше юнит-тестов

Пожелания для юнит-тестирования

- Структуризация
  - Один класс / модуль — один тестовый файл
  - Один метод класса — один тестовый метод
- Повторное использование кода в ООП-стиле
- Возможность подготовки / очистки тестовых данных
- Запуск тестов для отдельного класса / метода

	Лаконичные assert-ы	Структури-зация	Наследование	Подготовка/ очистка данных	Выборочный запуск
Test.More					

Lua-фреймворки

- Busted
- LuaUnit
- Telescope, lunit, ...
- http://lua-users.org/wiki/UnitTesting

Busted

Busted asserts: luassert

- assert.is_true(got), assert.is_not.True(got), ...
- assert.are.equal(expected, got), ...
- assert.has.errors(function() error("this should fail") end)
- ...

Busted asserts: luassert

- Можно добавлять свои assert-ы
- Поддерживаются моки, асинхронные тесты и т.д.

busted/adder_spec.lua

```lua
require('busted.runner')()
local adder  = require('adder')
local assert = require('luassert')

describe("Function add()", function()

  it('can add 2.0001 and 1.9999', function()
    assert.are.equal(4, adder.add(2.0001, 1.9999))
  end)

  it('can add 1e100 and 2e100', function()
    assert.are.equal(3e100, adder.add(1e100, 2e100))
  end)
end)
```

busted/adder_spec.lua

```lua
require('busted.runner')()
local adder  = require('adder')
local assert = require('luassert')

describe("Function add()", function()

  it('can add 2.0001 and 1.9999', function()
    assert.are.equal(4, adder.add(2.0001, 1.9999))
  end)

  it('can add 1e100 and 2e100', function()
    assert.are.equal(3e100, adder.add(1e100, 2e100))
  end)
end)
```

busted/adder_spec.lua

```lua
require('busted.runner')()
local adder  = require('adder')
local assert = require('luassert')

describe("Function add()", function()

  it('can add 2.0001 and 1.9999', function()
    assert.are.equal(4, adder.add(2.0001, 1.9999))
  end)

  it('can add 1e100 and 2e100', function()
    assert.are.equal(3e100, adder.add(1e100, 2e100))
  end)
end)
```

.busted

```
return {
   _all = {
     output = 'TAP',
   },
}
```

```
prove -vm busted/adder_spec.lua
```

```
busted/adder_spec.lua ..
ok 1 - Function add() can add 2.0001 and 1.9999
not ok 2 - Function add() can add 1e100 and 2e100
# busted/adder_spec.lua @ 13
# Failure message: busted/adder_spec.lua:14: Expected objects to be equal.
# Passed in:
# (number) 3.0000000000000242e+100
# Expected:
# (number) 2.9999999999999998534e+100
1..2
Dubious, test returned 1 (wstat 256, 0x100)
Failed 1/2 subtests

Test Summary Report
-------------------
busted/adder_spec.lua (Wstat: 256 Tests: 2 Failed: 1)
  Failed test:  2
  Non-zero exit status: 1
Files=1, Tests=2,  0 wallclock secs ( 0.02 usr  0.00 sys +  0.02 cusr  0.01 csys =
CPU)
Result: FAIL
```

```
busted/adder_spec.lua ..
ok 1 - Function add() can add 2.0001 and 1.9999
not ok 2 - Function add() can add 1e100 and 2e100
# busted/adder_spec.lua @ 13
# Failure message: busted/adder_spec.lua:14: Expected objects to be equal.
# Passed in:
# (number) 3.0000000000000242e+100
# Expected:
# (number) 2.9999999999999534e+100
1..2
Dubious, test returned 1 (wstat 256, 0x100)
Failed 1/2 subtests

Test Summary Report
-------------------
busted/adder_spec.lua (Wstat: 256 Tests: 2 Failed: 1)
  Failed test:  2
  Non-zero exit status: 1
Files=1, Tests=2,  0 wallclock secs ( 0.02 usr  0.00 sys +  0.02 cusr  0.01 csys =
CPU)
Result: FAIL
```

	Лаконичные assert-ы	Структури-зация	Наследование	Подготовка/ очистка данных	Выборочный запуск
Test.More	✓	✓	✗	✗	✗
Busted	✗	✓	✗	✓	✗

LuaUnit

LuaUnit asserts

- assert_true(value) / assert_false(value)
- assert_nil(value) / assert_not_nil(value)
- assert_equals(actual, expected) / assert_not_equals(actual, expected)
- assert_str_contains(str, sub, [, useRe])
- ...

luaunit/tests/TestAdder.lua

```lua
-- Global table!
TestAdder = {}

local adder = require('adder')

function TestAdder:test_add()
  assert_equals(adder.add(2.0001, 1.9999), 4)
  assert_equals(adder.add(1e100, 2e100), 3e100)
end
```

luaunit/run_tests.lua

```
EXPORT_ASSERT_TO_GLOBALS = true
local luaunit = require('luaunit')

-- Load test library
require('TestAdder')

lu = luaunit.LuaUnit.new()
lu:setOutputType('tap')
lu:runSuite()
```

prove -vm luaunit/run_tests.lua

```
luaunit/run_tests.lua ..
1..1
# Started on Sun May  6 19:15:46 2018
# Starting class: TestAdder
not ok 1 TestAdder.test_add
#   luaunit/tests/TestAdder.lua:8: expected: 3e+100, actual: 3e+100
# Ran 1 tests in 0.001 seconds, 0 successes, 1 failure
Failed 1/1 subtests

Test Summary Report
-------------------
luaunit/run_tests.lua (Wstat: 0 Tests: 1 Failed: 1)
  Failed test:  1
Files=1, Tests=1,  0 wallclock secs ( 0.02 usr +  0.00 sys =  0.02 CPU)
Result: FAIL
```

```
luaunit/run_tests.lua ..
1..1
# Started on Sun May  6 19:15:46 2018
# Starting class: TestAdder
not ok 1 TestAdder.test_add
#   luaunit/tests/TestAdder.lua:8: expected: 3e+100, actual: 3e+100
# Ran 1 tests in 0.001 seconds, 0 successes, 1 failure
Failed 1/1 subtests

Test Summary Report
-------------------
luaunit/run_tests.lua (Wstat: 0 Tests: 1 Failed: 1)
  Failed test:  1
Files=1, Tests=1,  0 wallclock secs ( 0.02 usr +  0.00 sys =  0.02 CPU)
Result: FAIL
```

	Лаконичные assert-ы	Структури-зация	Наследование	Подготовка/ очистка данных	Выборочный запуск
Test.More	✓	✓	✗	✗	✗
Busted	✗	✓	✗	✓	✗
LuaUnit	✗	✓	✗	✓	✗

LuaUnit: особенности

- Каждый тест:
  - в отдельной **глобальной** таблице (модуле) "by design"
- Тестовые сообщения задавать нельзя вообще
- Фреймворк "перегружен" лишними проверками и функциями

Что можно сделать?

Perl: Test::Class...

...основан на jUnit...

...использует Test::More!

Lua: ...?

...lua-TestClass!

t/tests/TestsFor/Adder.lua

```
local _M = require('middleclass')('TestsFor.Adder', require('Test.Class'))
require('Test.More')

local adder = require('adder')

function _M:test_add()
  is(adder.add(2.0001, 1.9999), 4, '2.0001 plus 1.9999 equals 4')
  is(adder.add(1e100, 2e100), 3e100, '1e100 plus 2e100 equals 3e100')
end

return _M
```

t/tests/TestsFor/Adder.lua

```
local _M = require('middleclass')('TestsFor.Adder', require('Test.Class'))
require('Test.More')

local adder = require('adder')

function _M:test_add()
  is(adder.add(2.0001, 1.9999), 4, '2.0001 plus 1.9999 equals 4')
  is(adder.add(1e100, 2e100), 3e100, '1e100 plus 2e100 equals 3e100')
end

return _M
```

t/tests/TestsFor/Adder.lua

```
local _M = require('middleclass')('TestsFor.Adder', require('Test.Class'))
require('Test.More')

local adder = require('adder')

function _M:test_add()
  is(adder.add(2.0001, 1.9999), 4, '2.0001 plus 1.9999 equals 4')
  is(adder.add(1e100, 2e100), 3e100, '1e100 plus 2e100 equals 3e100')
end

return _M
```

t/tests/TestsFor/Adder.lua

```
local _M = require('middleclass')('TestsFor.Adder', require('Test.Class'))
require('Test.More')

local adder = require('adder')

function _M:test_add()
  is(adder.add(2.0001, 1.9999), 4, '2.0001 plus 1.9999 equals 4')
  is(adder.add(1e100, 2e100), 3e100, '1e100 plus 2e100 equals 3e100')
end

return _M
```

t/tests/TestsFor/Adder.lua

```
local _M = require('middleclass')('TestsFor.Adder', require('Test.Class'))
require('Test.More')

local adder = require('adder')

function _M:test_add()
  is(adder.add(2.0001, 1.9999), 4, '2.0001 plus 1.9999 equals 4')
  is(adder.add(1e100, 2e100), 3e100, '1e100 plus 2e100 equals 3e100')
end

return _M
```

t/tests/TestsFor/Adder.lua

```
local _M = require('middleclass')('TestsFor.Adder', require('Test.Class'))
require('Test.More')

local adder = require('adder')

function _M:test_add()
  is(adder.add(2.0001, 1.9999), 4, '2.0001 plus 1.9999 equals 4')
  is(adder.add(1e100, 2e100), 3e100, '1e100 plus 2e100 equals 3e100')
end

return _M
```

t/tests/TestsFor/Adder.lua

```
local _M = require('middleclass')('TestsFor.Adder', require('Test.Class'))
require('Test.More')

local adder = require('adder')

function _M:test_add()
  is(adder.add(2.0001, 1.9999), 4, '2.0001 plus 1.9999 equals 4')
  is(adder.add(1e100, 2e100), 3e100, '1e100 plus 2e100 equals 3e100')
end

return _M
```

t/tc.lua

```
require('TestsFor.Adder')

local tc = require('Test.Class')
tc.run_tests()
```

prove -vm t/tc.lua

```
t/tc.lua ..
# Running tests for TestsFor.Adder
#
# Subtest: TestsFor.Adder
    # TestsFor.Adder.test_add()
    # Subtest: test_add
        ok 1 - 2.0001 plus 1.9999 equals 4
        not ok 2 - 1e100 plus 2e100 equals 3e100
        #     Failed test (t/tests/TestsFor/Adder.lua at line 9)
        #          got: 3e+100
        #     expected: 3e+100
        1..2
    not ok 1 - test_add
    #     Failed test (/Users/ichesnokov/git/lua-TestClass/lib/Test/Class.lua at line 53)
    1..1
not ok 1 - TestsFor.Adder
#     Failed test (/Users/ichesnokov/git/lua-TestClass/lib/Test/Class.lua at line 18)
1..1
Failed 1/1 subtests

Test Summary Report
-------------------
t/tc.lua (Wstat: 0 Tests: 1 Failed: 1)
  Failed test:  1
Files=1, Tests=1,  0 wallclock secs ( 0.02 usr +  0.00 sys =  0.02 CPU)
Result: FAIL
```

```
t/tc.lua ..
# Running tests for TestsFor.Adder
#
# Subtest: TestsFor.Adder
    # TestsFor.Adder.test_add()
    # Subtest: test_add
        ok 1 - 2.0001 plus 1.9999 equals 4
        not ok 2 - 1e100 plus 2e100 equals 3e100
        #     Failed test (t/tests/TestsFor/Adder.lua at line 9)
        #          got: 3e+100
        #     expected: 3e+100
        1..2
    not ok 1 - test_add
    #     Failed test (/Users/ichesnokov/git/lua-TestClass/lib/Test/Class.lua at line 53)
    1..1
not ok 1 - TestsFor.Adder
#     Failed test (/Users/ichesnokov/git/lua-TestClass/lib/Test/Class.lua at line 18)
1..1
Failed 1/1 subtests

Test Summary Report
-------------------
t/tc.lua (Wstat: 0 Tests: 1 Failed: 1)
  Failed test:  1
Files=1, Tests=1,  0 wallclock secs ( 0.02 usr +  0.00 sys =  0.02 CPU)
Result: FAIL
```

```
t/tc.lua ..
# Running tests for TestsFor.Adder
#
# Subtest: TestsFor.Adder
    # TestsFor.Adder.test_add()
    # Subtest: test_add
        ok 1 - 2.0001 plus 1.9999 equals 4
        not ok 2 - 1e100 plus 2e100 equals 3e100
        #     Failed test (t/tests/TestsFor/Adder.lua at line 9)
        #          got: 3e+100
        #     expected: 3e+100
        1..2
    not ok 1 - test_add
    #     Failed test (/Users/ichesnokov/git/lua-TestClass/lib/Test/Class.lua at line 53)
    1..1
not ok 1 - TestsFor.Adder
#     Failed test (/Users/ichesnokov/git/lua-TestClass/lib/Test/Class.lua at line 18)
1..1
Failed 1/1 subtests

Test Summary Report
-------------------
t/tc.lua (Wstat: 0 Tests: 1 Failed: 1)
  Failed test:  1
Files=1, Tests=1,  0 wallclock secs ( 0.02 usr +  0.00 sys =  0.02 CPU)
Result: FAIL
```

```
t/tc.lua ..
# Running tests for TestsFor.Adder
#
# Subtest: TestsFor.Adder
    # TestsFor.Adder.test_add()
    # Subtest: test_add
        ok 1 - 2.0001 plus 1.9999 equals 4
        not ok 2 - 1e100 plus 2e100 equals 3e100
        #     Failed test (t/tests/TestsFor/Adder.lua at line 9)
        #          got: 3e+100
        #     expected: 3e+100
        1..2
    not ok 1 - test_add
    #     Failed test (/Users/ichesnokov/git/lua-TestClass/lib/Test/Class.lua at line 53)
    1..1
not ok 1 - TestsFor.Adder
#     Failed test (/Users/ichesnokov/git/lua-TestClass/lib/Test/Class.lua at line 18)
1..1
Failed 1/1 subtests

Test Summary Report
-------------------
t/tc.lua (Wstat: 0 Tests: 1 Failed: 1)
  Failed test:  1
Files=1, Tests=1,  0 wallclock secs ( 0.02 usr +  0.00 sys =  0.02 CPU)
Result: FAIL
```

```
t/tc.lua ..
# Running tests for TestsFor.Adder
#
# Subtest: TestsFor.Adder
    # TestsFor.Adder.test_add()
    # Subtest: test_add
        ok 1 - 2.0001 plus 1.9999 equals 4
        not ok 2 - 1e100 plus 2e100 equals 3e100
        #     Failed test (t/tests/TestsFor/Adder.lua at line 9)
        #          got: 3e+100
        #     expected: 3e+100
        1..2
    not ok 1 - test_add
    #     Failed test (/Users/ichesnokov/git/lua-TestClass/lib/Test/Class.lua at line 53)
    1..1
not ok 1 - TestsFor.Adder
#     Failed test (/Users/ichesnokov/git/lua-TestClass/lib/Test/Class.lua at line 18)
1..1
Failed 1/1 subtests

Test Summary Report
-------------------
t/tc.lua (Wstat: 0 Tests: 1 Failed: 1)
  Failed test:  1
Files=1, Tests=1,  0 wallclock secs ( 0.02 usr +  0.00 sys =  0.02 CPU)
Result: FAIL
```

```
t/tc.lua ..
# Running tests for TestsFor.Adder
#
# Subtest: TestsFor.Adder
    # TestsFor.Adder.test_add()
    # Subtest: test_add
        ok 1 - 2.0001 plus 1.9999 equals 4
        not ok 2 - 1e100 plus 2e100 equals 3e100
        #     Failed test (t/tests/TestsFor/Adder.lua at line 9)
        #          got: 3e+100
        #     expected: 3e+100
        1..2
    not ok 1 - test_add
    #     Failed test (/Users/ichesnokov/git/lua-TestClass/lib/Test/Class.lua at line 53)
    1..1
not ok 1 - TestsFor.Adder
#     Failed test (/Users/ichesnokov/git/lua-TestClass/lib/Test/Class.lua at line 18)
1..1
Failed 1/1 subtests

Test Summary Report
-------------------
t/tc.lua (Wstat: 0 Tests: 1 Failed: 1)
  Failed test:  1
Files=1, Tests=1,  0 wallclock secs ( 0.02 usr +  0.00 sys =  0.02 CPU)
Result: FAIL
```

```
t/tc.lua ..
# Running tests for TestsFor.Adder
#
# Subtest: TestsFor.Adder
    # TestsFor.Adder.test_add()
    # Subtest: test_add
        ok 1 - 2.0001 plus 1.9999 equals 4
        not ok 2 - 1e100 plus 2e100 equals 3e100
        #     Failed test (t/tests/TestsFor/Adder.lua at line 9)
        #          got: 3e+100
        #     expected: 3e+100
        1..2
    not ok 1 - test_add
    #     Failed test (/Users/ichesnokov/git/lua-TestClass/lib/Test/Class.lua at line 53)
    1..1
not ok 1 - TestsFor.Adder
#     Failed test (/Users/ichesnokov/git/lua-TestClass/lib/Test/Class.lua at line 18)
1..1
Failed 1/1 subtests

Test Summary Report
-------------------
t/tc.lua (Wstat: 0 Tests: 1 Failed: 1)
  Failed test:  1
Files=1, Tests=1,  0 wallclock secs ( 0.02 usr +  0.00 sys =  0.02 CPU)
Result: FAIL
```

```
t/tc.lua ..
# Running tests for TestsFor.Adder
#
# Subtest: TestsFor.Adder
    # TestsFor.Adder.test_add()
    # Subtest: test_add
        ok 1 - 2.0001 plus 1.9999 equals 4
        not ok 2 - 1e100 plus 2e100 equals 3e100
        #     Failed test (t/tests/TestsFor/Adder.lua at line 9)
        #          got: 3e+100
        #     expected: 3e+100
        1..2
    not ok 1 - test_add
    #     Failed test (/Users/ichesnokov/git/lua-TestClass/lib/Test/Class.lua at line 53)
    1..1
not ok 1 - TestsFor.Adder
#     Failed test (/Users/ichesnokov/git/lua-TestClass/lib/Test/Class.lua at line 18)
1..1
Failed 1/1 subtests

Test Summary Report
-------------------
t/tc.lua (Wstat: 0 Tests: 1 Failed: 1)
  Failed test:  1
Files=1, Tests=1,  0 wallclock secs ( 0.02 usr +  0.00 sys =  0.02 CPU)
Result: FAIL
```

Добавим ещё один класс

lib/Subtractor.lua

```
return {

  -- subtract() - subtracts y from x and returns result
  subtract = function(x, y)
    return x - y
  end

}
```

t/tc.lua

```
require('TestsFor.Adder')
require('TestsFor.Subtractor')

local tc = require('Test.Class')
tc.run_tests()
```

```
# Running tests for TestsFor.Adder
#
# Subtest: TestsFor.Adder
    # TestsFor.Adder.test_add()
    # Subtest: test_add
        ok 1 - 2.0001 plus 1.9999 equals 4
        not ok 2 - 1e100 plus 2e100 equals 3e100 # TODO FP precision error
        #     Failed (TODO) test (t/tests/TestsFor/Adder.lua at line 9)
        #          got: 3e+100
        #     expected: 3e+100
        1..2
    ok 1 - test_add
    1..1
ok 1 - TestsFor.Adder
# Running tests for TestsFor.Subtractor
#
# Subtest: TestsFor.Subtractor
    # TestsFor.Subtractor.test_subtract()
    # Subtest: test_subtract
        ok 1 - 1 minus 1 equals 0
        ok 2 - 0 minus -100 equals 100
        1..2
    ok 1 - test_subtract
    1..1
ok 2 - TestsFor.Subtractor
```

```
# Running tests for TestsFor.Adder
#
# Subtest: TestsFor.Adder
    # TestsFor.Adder.test_add()
    # Subtest: test_add
        ok 1 - 2.0001 plus 1.9999 equals 4
        not ok 2 - 1e100 plus 2e100 equals 3e100 # TODO FP precision error
        #     Failed (TODO) test (t/tests/TestsFor/Adder.lua at line 9)
        #          got: 3e+100
        #     expected: 3e+100
        1..2
    ok 1 - test_add
    1..1
ok 1 - TestsFor.Adder
# Running tests for TestsFor.Subtractor
#
# Subtest: TestsFor.Subtractor
    # TestsFor.Subtractor.test_subtract()
    # Subtest: test_subtract
        ok 1 - 1 minus 1 equals 0
        ok 2 - 0 minus -100 equals 100
        1..2
    ok 1 - test_subtract
    1..1
ok 2 - TestsFor.Subtractor
```

```
# Running tests for TestsFor.Adder
#
# Subtest: TestsFor.Adder
    # TestsFor.Adder.test_add()
    # Subtest: test_add
        ok 1 - 2.0001 plus 1.9999 equals 4
        not ok 2 - 1e100 plus 2e100 equals 3e100 # TODO FP precision error
        #     Failed (TODO) test (t/tests/TestsFor/Adder.lua at line 9)
        #          got: 3e+100
        #     expected: 3e+100
        1..2
    ok 1 - test_add
    1..1
ok 1 - TestsFor.Adder
# Running tests for TestsFor.Subtractor
#
# Subtest: TestsFor.Subtractor
    # TestsFor.Subtractor.test_subtract()
    # Subtest: test_subtract
        ok 1 - 1 minus 1 equals 0
        ok 2 - 0 minus -100 equals 100
        1..2
    ok 1 - test_subtract
    1..1
ok 2 - TestsFor.Subtractor
```

Как загрузить много тестовых классов?

t/tc.lua

```
require('TestsFor.Adder')
require('TestsFor.Subtractor')
require('TestsFor.Multiplier')
require('TestsFor.Divider')
require('TestsFor.Utils')
require('TestsFor.Context')
require('TestsFor.Character')
require('TestsFor.Station')
require('TestsFor.Travel')

-- ...

local tc = require('Test.Class')
tc.run_tests()
```

t/tc-load.lua

```
local tc = require('Test.Class')

tc.load_classes('t/tests')
tc.run_tests()
```

load_classes(dir)

- Рекурсивно обходит указанную директорию в поисках **.lua** файлов
- Для каждого найденного файла:
  - Преобразует имя файла в имя модуля
  - Загружает его при помощи **require()**
  - Если это наследник **Test.Class** - добавляет в список для запуска

t/tc-load.lua

```
local tc = require('Test.Class')

tc.load_classes('t/tests')
tc.run_tests()
```

run_tests(arg)

- Для каждого тестового класса из списка:
  - создает объект при помощи **new()**...
  - ...и запускает на нем тестовые методы
- arg - таблица с опциональным параметром:
  - include - регулярное выражение, ограничивающее методы для запуска

Подготовка / очистка тестовых данных

...для тестового класса

- test_startup() - подготовка
- test_shutdown() - очистка

...для тестового метода

- test_setup() - подготовка
  - self.current_method - имя метода
- test_teardown() - очистка

Наследование

**“Функции, которые используют базовый тип,
должны иметь возможность
использовать подтипы базового типа,
не зная об этом.”**

– Принцип подстановки Барбары Лисков (определение Роберта С. Мартина)

```
local _M = require('middleclass')('TestsFor.Parent', require('Test.Class'))
require 'Test.More'

function _M:test_parent()
  pass('I am a test of parent')
end

return _M
```

```
local _M = require('middleclass')('TestsFor.Parent', require('Test.Class'))
require 'Test.More'

function _M:test_parent()
  pass('I am a test of parent')
end

return _M
```

```
local _M = require('middleclass')('TestsFor.Parent', require('Test.Class'))
require 'Test.More'

function _M:test_parent()
  pass('I am a test of parent')
end

return _M
```

```
local _M = require('middleclass')('TestsFor.Parent', require('Test.Class'))
require 'Test.More'

function _M:test_parent()
  pass('I am a test of parent')
end

return _M
```

```
local _M = require('middleclass')('TestsFor.Child', require('TestsFor.Parent'))
require 'Test.More'

function _M:test_child()
  pass('I am a test of child')
end

return _M
```

```
local _M = require('middleclass')('TestsFor.Child', require('TestsFor.Parent'))
require 'Test.More'

function _M:test_child()
  pass('I am a test of child')
end

return _M
```

```
local _M = require('middleclass')('TestsFor.Child', require('TestsFor.Parent'))
require 'Test.More'

function _M:test_child()
  pass('I am a test of child')
end

return _M
```

```
t/tc-inheritance.lua ..
# Running tests for TestsFor.Parent
#
# Subtest: TestsFor.Parent
    # TestsFor.Parent.test_parent()
    # Subtest: test_parent
        ok 1 - I am a test of parent
        1..1
    ok 1 - test_parent
    1..1
ok 1 - TestsFor.Parent

# Running tests for TestsFor.Child
# ...
```

```
# Running tests for TestsFor.Child
#
# Subtest: TestsFor.Child
    # TestsFor.Child.test_child()
    # Subtest: test_child
        ok 1 - I am a test of child
        1..1
    ok 1 - test_child
    # TestsFor.Child.test_parent()
    # Subtest: test_parent
        ok 1 - I am a test of parent
        1..1
    ok 2 - test_parent
    1..2
ok 2 - TestsFor.Child
1..2
ok
All tests successful.
Files=1, Tests=2,  0 wallclock secs ( 0.02 usr +  0.00 sys =  0.02 CPU)
Result: PASS
```

Mixins

lib/MyTestMixin.lua

```
require 'Test.More'

return {

  pi = 3.1415,

  test_from_mixin = function()
    pass('This test always passes')
  end
}
```

t/tests/TestsFor/Mixin.lua

```
local _M = require('middleclass')('TestsFor.Mixin', require('Test.Class'))
  :include(require('MyTestMixin'))

function _M:test_pi()
  pass('PI is: ' .. self.pi)
end

return _M
```

t/tests/TestsFor/Mixin.lua

```
local _M = require('middleclass')('TestsFor.Mixin', require('Test.Class'))
  :include(require('MyTestMixin'))

function _M:test_pi()
  pass('PI is: ' .. self.pi)
end

return _M
```

t/tests/TestsFor/Mixin.lua

```
local _M = require('middleclass')('TestsFor.Mixin', require('Test.Class'))
  :include(require('MyTestMixin'))

function _M:test_pi()
  pass('PI is: ' .. self.pi)
end

return _M
```

```
t/tc-mixin.lua ..
# Running tests for TestsFor.Mixin
#
# Subtest: TestsFor.Mixin
    # TestsFor.Mixin.test_from_mixin()
    # Subtest: test_from_mixin
        ok 1 - This test always passes
        1..1
    ok 1 - test_from_mixin
    # TestsFor.Mixin.test_pi()
    # Subtest: test_pi
        ok 1 - PI is: 3.1415
        1..1
    ok 2 - test_pi
    1..2
ok 1 - TestsFor.Mixin
1..1
ok
All tests successful.
Files=1, Tests=1,  0 wallclock secs ( 0.02 usr +  0.00 sys =  0.02 CPU)
Result: PASS
```

```
t/tc-mixin.lua ..
# Running tests for TestsFor.Mixin
#
# Subtest: TestsFor.Mixin
    # TestsFor.Mixin.test_from_mixin()
    # Subtest: test_from_mixin
        ok 1 - This test always passes
        1..1
    ok 1 - test_from_mixin
    # TestsFor.Mixin.test_pi()
    # Subtest: test_pi
        ok 1 - PI is: 3.1415
        1..1
    ok 2 - test_pi
    1..2
ok 1 - TestsFor.Mixin
1..1
ok
All tests successful.
Files=1, Tests=1,  0 wallclock secs ( 0.02 usr +  0.00 sys =  0.02 CPU)
Result: PASS
```

```
t/tc-mixin.lua ..
# Running tests for TestsFor.Mixin
#
# Subtest: TestsFor.Mixin
    # TestsFor.Mixin.test_from_mixin()
    # Subtest: test_from_mixin
        ok 1 - This test always passes
        1..1
    ok 1 - test_from_mixin
    # TestsFor.Mixin.test_pi()
    # Subtest: test_pi
        ok 1 - PI is: 3.1415
        1..1
    ok 2 - test_pi
    1..2
ok 1 - TestsFor.Mixin
1..1
ok
All tests successful.
Files=1, Tests=1,  0 wallclock secs ( 0.02 usr +  0.00 sys =  0.02 CPU)
Result: PASS
```

Ок, с Test.Class мы можем...

С Test.Class мы можем

- Написать юнит-тест
- Запустить тесты:
  - Для выбранных или для всех классов в директории
  - Для всех или выбранных методов
- Подготавливать / очищать данные для тестов
- Использовать наследование и примеси

	Лаконичные assert-ы	Структури-зация	Наследование	Подготовка/ очистка данных	Выборочный запуск
Test.More	✓	✓	✗	✗	✗
Busted	✗	✓	✗	✓	✗
LuaUnit	✗	✓	✗	✓	✗
Test.Class	✓	✓	✓	✓	✓

Test.Class - особенности

- Работает только на *nix-based системах (сейчас)
- Нет тестов, документации
- Ранняя альфа версия - возможны (и наверняка будут) изменения

Использовать
на свой страх и риск ;-)

Test.Class - wishlist / plans

- Поддержка тестового плана
- "Автоплан" по количеству тестовых классов / методов
- Пропуск тестовых классов / методов
- Меньше boilerplate в начале тестовых классов

Скачать (бесплатно, без sms)

- Busted: luarocks.org
- LuaUnit: luarocks.org
- lua-TestMore: luarocks.org
- github.com/ichesnokov/lua-TestClass

Вопросы?

Спасибо!