Ruby初級者向けレッスン第 9 回（解答）

かずひこ@株式会社 ネットワーク応用通信研究所，コウザイ@小波ゼミ，サカイ@小波ゼミ

2006 年 10 月 7 日

Step 1

- テストコード (test_stack.rb)

```
require 'stack'
require 'test/unit'
class TestStack < Test::Unit::TestCase
  def setup
    @stack = Stack.new
  end

  def test_empty?
    assert(@stack.empty?, 'a new stack is empty.')
  end
end
```

- スタッククラスのコード (stack.rb)

```
class Stack
  def empty?
    true
  end
end
```

Step 2

- テストコード (test_stack.rb)

```
require 'stack'
require 'test/unit'
class TestStack < Test::Unit::TestCase
  def setup
    @stack = Stack.new
  end
```

```
  def test_empty?
    assert(@stack.empty?, 'a new stack is empty.')
  end

  def test_push_and_pop
    @stack.push(3)
    assert_equal(3, @stack.pop, 'pop returns the last value.')
  end
end
```

- スタッククラスのコード (stack.rb)

```
class Stack
  def empty?
    true
  end

  def push(val)
  end

  def pop
    return 3
  end
end
```

Step 3

- テストコード (test_stack.rb)

```
require 'stack'
require 'test/unit'
class TestStack < Test::Unit::TestCase
  def setup
    @stack = Stack.new
  end

  def test_empty?
    assert(@stack.empty?, 'a new stack is empty.')
  end

  def test_push_and_pop
    @stack.push(3)
```

```
    assert_equal(3, @stack.pop, 'pop returns the last value.')
  end

  def test_push_and_size
    @stack.push(3)
    assert_equal(1, @stack.size, 'push increments the size.')
  end
end
```

- スタッククラスのコード (stack.rb)

```
class Stack
  def empty?
    true
  end

  def push(val)
  end

  def pop
    return 3
  end

  def size
    1
  end
end
```

Step 4

- テストコード (test_stack.rb)

```
require 'stack'
require 'test/unit'
class TestStack < Test::Unit::TestCase
  def setup
    @stack = Stack.new
  end

  def test_empty?
    assert(@stack.empty?, 'a new stack is empty.')
  end
```

```
  def test_push_and_pop
    @stack.push(3)
    assert_equal(3, @stack.pop, 'pop returns the last value.')
  end

  def test_push_and_size
    @stack.push(3)
    assert_equal(1, @stack.size, 'push increments the size.')
    @stack.push(5)
    assert_equal(2, @stack.size, 'push increments the size.')
  end
end
```

- スタッククラスのコード (stack.rb)

```
class Stack
  def initialize
    @size = 0
  end

  def empty?
    true
  end

  def push(val)
    @size += 1
  end

  def pop
    return 3
  end

  def size
    return @size
  end
end
```

Step 5

- テストコード (test_stack.rb)

```
require 'stack'
require 'test/unit'
```

```
class TestStack < Test::Unit::TestCase
  def setup
    @stack = Stack.new
  end

  def test_empty?
    assert(@stack.empty?, 'a new stack is empty.')
  end

  def test_push_and_pop
    @stack.push(3)
    assert_equal(3, @stack.pop, 'pop returns the last value.')
  end

  def test_push_and_size
    @stack.push(3)
    assert_equal(1, @stack.size, 'push increments the size.')
    @stack.push(5)
    assert_equal(2, @stack.size, 'push increments the size.')
  end

  def test_push_and_empty?
    @stack.push(3)
    assert_equal(false, @stack.empty?, 'a stack with data is not empty.')
  end
end
```

- スタッククラスのコード (stack.rb)

```
class Stack
  def initialize
    @size = 0
  end

  def empty?
    return @size == 0
  end

  def push(val)
    @size += 1
  end

  def pop
```

```
    return 3
  end

  def size
    return @size
  end
end
```

Step 6

- テストコード (test_stack.rb)

```
require 'stack'
require 'test/unit'
class TestStack < Test::Unit::TestCase
  def setup
    @stack = Stack.new
  end

  def test_empty?
    assert(@stack.empty?, 'a new stack is empty.')
  end

  def test_push_and_pop
    @stack.push(3)
    assert_equal(3, @stack.pop, 'pop returns the last value.')
  end

  def test_push_and_size
    @stack.push(3)
    assert_equal(1, @stack.size, 'push increments the size.')
    @stack.push(5)
    assert_equal(2, @stack.size, 'push increments the size.')
  end

  def test_push_and_empty?
    @stack.push(3)
    assert_equal(false, @stack.empty?, 'a stack with data is not empty.')
  end

  def test_empty_pop
    assert_raise(Stack::EmptyStackError,
                 'to pop a empty stack raise an error.') {@stack.pop}
```

```
  end
end
```

- スタッククラスのコード (stack.rb)

```
class Stack
  class EmptyStackError < StandardError; end

  def initialize
    @size = 0
  end

  def empty?
    return @size == 0
  end

  def push(val)
    @size += 1
  end

  def pop
    raise EmptyStackError if empty?
    return 3
  end

  def size
    return @size
  end
end
```

Step 7

- テストコード (test_stack.rb)

```
require 'stack'
require 'test/unit'
class TestStack < Test::Unit::TestCase
  def setup
    @stack = Stack.new
  end

  def test_empty?
    assert(@stack.empty?, 'a new stack is empty.')
```

```
  end

  def test_push_and_pop
    @stack.push(3)
    assert_equal(3, @stack.pop, 'pop returns the last value.')
  end

  def test_push_and_size
    @stack.push(3)
    assert_equal(1, @stack.size, 'push increments the size.')
    @stack.push(5)
    assert_equal(2, @stack.size, 'push increments the size.')
  end

  def test_push_and_empty?
    @stack.push(3)
    assert_equal(false, @stack.empty?, 'a stack with data is not empty.')
  end

  def test_empty_pop
    assert_raise(Stack::EmptyStackError,
                 'to pop a empty stack raise an error.') {@stack.pop}
  end

  def test_push_push_pop_and_size
    @stack.push(3)
    @stack.push(5)
    @stack.pop
    assert_equal(1, @stack.size, 'pop decrements the size.')
  end
end
```

- スタッククラスのコード (stack.rb)

```
class Stack
  class EmptyStackError < StandardError; end

  def initialize
    @size = 0
  end

  def empty?
    return @size == 0
```

```
  end

  def push(val)
    @size += 1
  end

  def pop
    raise EmptyStackError if empty?
    @size -= 1
    return 3
  end

  def size
    return @size
  end
end
```

Step 8

- テストコード (test_stack.rb)

```
require 'stack'
require 'test/unit'
class TestStack < Test::Unit::TestCase
  def setup
    @stack = Stack.new
  end

  def test_empty?
    assert(@stack.empty?, 'a new stack is empty.')
  end

  def test_push_and_pop
    @stack.push(3)
    assert_equal(3, @stack.pop, 'pop returns the last value.')
  end

  def test_push_and_size
    @stack.push(3)
    assert_equal(1, @stack.size, 'push increments the size.')
    @stack.push(5)
    assert_equal(2, @stack.size, 'push increments the size.')
  end
```

```
  def test_push_and_empty?
    @stack.push(3)
    assert_equal(false, @stack.empty?, 'a stack with data is not empty.')
  end

  def test_empty_pop
    assert_raise(Stack::EmptyStackError,
                 'to pop a empty stack raise an error.') {@stack.pop}
  end

  def test_push_push_pop_and_size
    @stack.push(3)
    @stack.push(5)
    @stack.pop
    assert_equal(1, @stack.size, 'pop decrements the size.')
  end

  def test_push_push_and_pop
    @stack.push(3)
    @stack.push(5)
    assert_equal(5, @stack.pop, 'pop returns the last value.')
  end
end
```

- スタッククラスのコード (stack.rb)

```
class Stack
  class EmptyStackError < StandardError; end

  def initialize
    @size = 0
    @values = Array.new
  end

  def empty?
    return @size == 0
  end

  def push(val)
    @size += 1
    @values[@size - 1] = val
  end
```

```
  def pop
    raise EmptyStackError if empty?
    val = @values[@size - 1]
    @size -= 1
    return val
  end

  def size
    return @size
  end
end
```