Test Driven Development — Unit Testing

Unit Testing, Part 2

Walter Cazzola

Dipartimento di Informatica e Comunicazione
Università degli Studi di Milano
e-mail: cazzola@dico.unimi.it

To implement the classes representing:

- equilateral triangles, circles, rectangles, squares and pentagons
- with the following characteristics/properties/capabilities.

1. they should deal with calculate_perimeter() and calculate_area()
   messages with the obvious meaning
2. the state must be private
3. a list of geometric shapes must be sortable by area and by perimeter
   (not at the same time, of course)
4. to add an hexagon class should maintain all the capabilities of the
   existing classes and correctly interact with them

Test Driven Development — Unit Testing

Case Study: Polygons

import unittest
from math import *
from shapes import *

def make_test_on_area_and_perimeter(a_shape, f, n, name):
    class TestCalculateAreaAndPerimeter(unittest.TestCase):
        def setUp(self):
            self.known_areas = [(side, f(side)) for side in range(1,10000)]
            self.known_perimeters = [(side, n*side) for side in range(1,10000)]
        def test_calculate_area(self):
            for side, area in self.known_areas:
                result = a_shape(side).calculate_area()
                self.assertTrue(-0.00001 < result - area < 0.00001)
        def test_calculate_perimeter(self):
            for side, perimeter in self.known_perimeters:
                result = a_shape(side).calculate_perimeter()
                self.assertEqual(result, perimeter)
        def tearDown(self):
            self.known_areas = []
            self.known_perimeters = []
    return TestCalculateAreaAndPerimeter

TestCalculateAreaAndPerimeterTriangle = make_test_on_area_and_perimeter(triangle.triangle, lambda x: x**2*sqrt(3)/4, 3, "triangle")
TestCalculateAreaAndPerimeterPentagon = make_test_on_area_and_perimeter(pentagon.pentagon, lambda x: x**2*sqrt(25+10*sqrt(5))/4 , 5, "pentagon")
TestCalculateAreaAndPerimeterHexagon = make_test_on_area_and_perimeter(hexagon.hexagon, lambda x: x**2*3*sqrt(3)/2, 6, "hexagon")
TestCalculateAreaAndPerimeterHeptagon = make_test_on_area_and_perimeter(heptagon.heptagon, lambda x: 7/4*x**2*tan(pi/7)**-1, 7, "heptagon")
TestCalculateAreaAndPerimeterCircle = make_test_on_area_and_perimeter(circle.circle, lambda x: x**2*pi, 2*pi, "circle")
TestCalculateAreaAndPerimeterSquare = make_test_on_area_and_perimeter(square.square, lambda x: x**2, 4, "square")

Test Driven Development — Unit Testing

Testing the Area & Perimeter Calculation on Good Inputs

import unittest
from math import *
from shapes import *

def make_test_on_good_inputs(a_shape, name):
    class TestCalculateAreaAndPerimeter(unittest.TestCase):
        def setUp(self):
            self.known_areas = [(side, f(side)) for side in range(1,10000)]
            self.known_perimeters = [(side, n*side) for side in range(1,10000)]
        def test_calculate_area(self):
            for side, area in self.known_areas:
                result = a_shape(side).calculate_area()
                self.assertTrue(-0.00001 < result - area < 0.00001)
        def tearDown(self):
            self.known_areas = []
            self.known_perimeters = []
    return TestCalculateAreaAndPerimeter

TestCalculateAreaAndPerimeterTriangle = make_test_on_good_inputs(triangle.triangle, "triangle")
TestCalculateAreaAndPerimeterPentagon = make_test_on_good_inputs(pentagon.pentagon, "pentagon")
TestCalculateAreaAndPerimeterHexagon = make_test_on_good_inputs(hexagon.hexagon, "hexagon")
TestCalculateAreaAndPerimeterHeptagon = make_test_on_good_inputs(heptagon.heptagon, "heptagon")
TestCalculateAreaAndPerimeterCircle = make_test_on_good_inputs(circle.circle, "circle")
TestCalculateAreaAndPerimeterSquare = make_test_on_good_inputs(square.square, "square")
Test Driven Development — Unit Testing

Walter Cazzola

Unit Testing

Polygons Must Be Sortable by Areas & Perimeters

slide 7 of 9

Test Driven Development — Unit Testing

Polygons Must Be Sortable by Areas & Perimeters

slide 8 of 9

Test Driven Development — Unit Testing

Organizing the Testing Phase

slide 9 of 9

Test Driven Development — Unit Testing

The Solution

slide 10 of 9

# Import

import unittest

# Triangle

class Triangle(unittest.TestCase):
    def test_calculate_area(self):
        pass

# Circle

class Circle(unittest.TestCase):
    def test_calculate_area(self):
        pass

# Hexagon

class Hexagon(unittest.TestCase):
    def test_calculate_area(self):
        pass

# Heptagon

class Heptagon(unittest.TestCase):
    def test_calculate_area(self):
        pass

# Pentagon

class Pentagon(unittest.TestCase):
    def test_calculate_area(self):
        pass

# Square

class Square(unittest.TestCase):
    def test_calculate_area(self):
        pass

# TestSuite

tests_on_good_inputs = unittest.TestSuite(
    [TriangleTest(), CircleTest(), HexagonTest(), HeptagonTest(), PentagonTest()])

tests_on_bad_inputs = unittest.TestSuite(
    [TriangleNegativeTests(), CircleNegativeTests(), HexagonNegativeTests(), HeptagonNegativeTests(), PentagonNegativeTests()])

shape_tests = unittest.TestSuite(
    [tests_on_good_inputs, tests_on_bad_inputs])

suite = unittest.TestSuite(
    [shape_tests])

unittest.TextTestRunner(verbosity=2).run(suite)
References

- Jennifer Campbell, Paul Gries, Jason Montojo, and Greg Wilson. 
  Practical Programming: An Introduction to Computer Science Using Python. 

- Mark Pilgrim. 
  Dive into Python 3. 

- Mark Summerfield. 
  Addison-Wesley, October 2009.