Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Лабораторная работа №4

«Шаблоны проектирования и модульное тестирование в Python»

по дисциплине «Базовые компоненты интернет-технологий»

Выполнил:

студент группы ИУ5-35Б

Герасимов А.Д.

2021 г.

Задание:

1. Необходимо для произвольной предметной области реализовать от одного до трех шаблонов проектирования: один порождающий, один
структурный и один поведенческий. В качестве справочника шаблоновможно использовать следующий каталог. Для сдачи лабораторной работы в минимальном варианте достаточно реализовать один паттерн.
2. Вместо реализации паттерна Вы можете написать тесты для своейпрограммы решения биквадратного уравнения. В этом случае, возможно, Вам потребуется доработать программу решения
биквадратного уравнения, чтобы она была пригодна для модульноготестирования.
3. В модульных тестах необходимо применить следующие технологии:
    - TDD - фреймворк.
    - BDD - фреймворк.

Лабораторная работа была сделана по пункту 2 и 3.

Текст программы:

Main.py

```
from TDD import TestShapeFunctions
from shapes import Rectangle, Square
from shapes import Circle
from ShapeFactory import ShapeFactory
from Shape import Shape
import unittest
from unittest import TestCase


def main():

  shape_factory = ShapeFactory()
  shape_name = input("Enter the name of the shape: ")

  shape = shape_factory.create_shape(shape_name)

  print(f"The type of object created: {type(shape)}")
  print(f"The area of the {shape_name} is: {shape.calculate_area()}")
  print(f"The perimeter of the {shape_name} is: {shape.calculate_perimeter()}")




if __name__ == '__main__':
  main()
```

shapes.py

```
from Shape import Shape
import abc
class Rectangle(Shape):
  def __init__(self, height, width):
    self.height = height
    self.width = width

  def calculate_area(self):
    return self.height * self.width

  def calculate_perimeter(self):
    return 2 * (self.height + self.width)
```

```
class Square(Shape):
  def __init__(self, width):
    self.width = width

  def calculate_area(self):
    return self.width ** 2

  def calculate_perimeter(self):
    return 4 * self.width

class Circle(Shape):
  def __init__(self, radius):
    self.radius = radius

  def calculate_area(self):
    return 3 * self.radius * self.radius

  def calculate_perimeter(self):
    return 2 * 3.14 * self.radius
```

shapesFactory.py

```
from shapes import Circle
from shapes import Rectangle
from shapes import Square
class ShapeFactory:
  def create_shape(self, name):
    if name == 'circle':
      radius = input("Enter the radius of the circle: ")
      return Circle(float(radius))

    elif name == 'rectangle':
      height = input("Enter the height of the rectangle: ")
      width = input("Enter the width of the rectangle: ")
      return Rectangle(int(height), int(width))

    elif name == 'square':
      width = input("Enter the width of the square: ")
      return Square(int(width))
```

shape.py

```
import abc
class Shape(metaclass=abc.ABCMeta):
  @abc.abstractmethod
```

```
  def calculate_area(self):
    pass

  @abc.abstractmethod
  def calculate_perimeter(self):
    pass
```

TDD.py

```
from shapes import Rectangle, Square
from shapes import Circle
import unittest



class TestShapeFunctions(unittest.TestCase):
    def test_calculate_area_of_rectangle(self):
        rec = Rectangle(2, 2)
        self.assertEqual(rec.calculate_area(), 4, "Should be 4")

    def test_calculate_perimeter_of_rectangle(self):
        rec = Rectangle(2, 2)
        self.assertEqual(rec.calculate_perimeter(), 8, "Should be 8")

    def test_calculate_area_of_Square(self):
        square = Square(2)
        self.assertEqual(square.calculate_area(), 4, "Should be 4")

    def test_calculate_perimeter_of_Square(self):
        square = Square(2)
        self.assertEqual(square.calculate_perimeter(), 8, "Should be 8")

    def test_calculate_area_of_circle(self):
        circle = Circle(2)
        self.assertEqual(circle.calculate_area(), 12, "Should be 12")

    def test_calculate_perimeter_of_circle(self):
        circle = Circle(2)
        self.assertEqual(circle.calculate_perimeter(), 12.56, "Should be 12.56")


suite = unittest.TestLoader().loadTestsFromTestCase(TestShapeFunctions)
unittest.TextTestRunner(verbosity=2).run(suite)
```

steps.py

```python
from behave import given, when, then, step
from shapes import Rectangle, Square, Circle


# rectangle = Rectangle(2, 3)

@given("Rectangle has {height:d} and {width:d} cm")
def initial_impl(context, width, height):
    context.rectangle = Rectangle(width, height)


@when("I search area of rectangle")
def area_impl(context):
    context.result = context.rectangle.calculate_area()


@then("I should have area of Rectangle {result:d}")
def result_impl(context, result):
    assert context.result == result


# Square

@given("Square has {width:d} cm")
def initial_impl(context, width):
    context.square = Square(width)


@when("I search area of square")
def area_impl(context):
    context.result = context.square.calculate_area()


@then("I should have area of square {result:d}")
def result_impl(context, result):
    assert context.result == result


# Circle
@given("Circle has {radius:d} cm")
def initial_impl(context, radius):
    context.circle = Circle(radius)
```

```
@when("I search area of circle")
def area_impl(context):
    context.result = context.circle.calculate_area()


# Не сравнивает дробные значения!
@then("I have area of circle {result:d}")
def result_impl(context, result):
    assert context.result == result
```

square.feature

```
Feature: Square
    Scenario: aboba
        Given Square has 4 cm
        When I search area of square
        Then I should have area of square 16
```

circle.feature

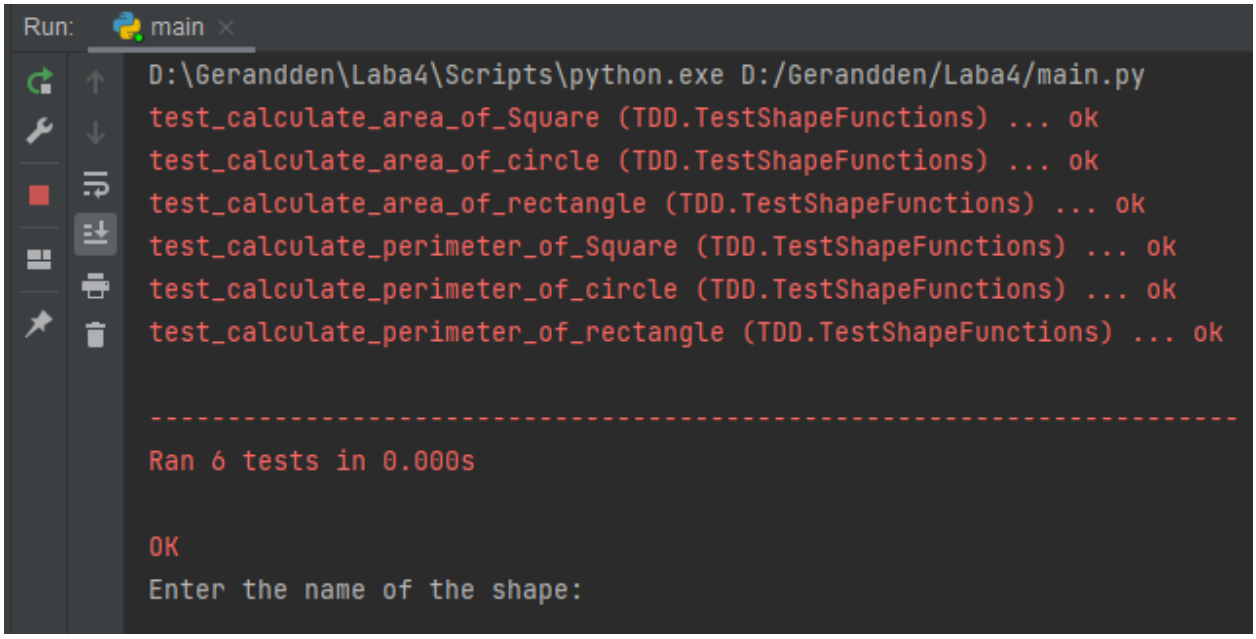
```
Feature: Circle
    Scenario: Arguments for given, when, then
        Given circle has 2 cm
        When I search area of circle
        Then I have area of circle 12
```

rectangle.feature

```
Feature: Rectangle
    Scenario: Arguments for given, when, then
        Given Rectangle has 2 and 3 cm
        When I search area of rectangle
        Then I should have area of Rectangle 6
```

Результаты тестирования:

TDD:



BDD:

```
Feature: Circle # features/circle.feature:1

  Scenario: Arguments for given, when, then  # features/circle.feature:2
    Given circle has 2 cm                    # features/steps/steps.py:40
    When I search area of circle             # features/steps/steps.py:45
    Then I have area of circle 12            # features/steps/steps.py:51

Feature: Rectangle # features/shapes.feature:1

  Scenario: Arguments for given, when, then  # features/shapes.feature:2
    Given Rectangle has 2 and 3 cm           # features/steps/steps.py:7
    When I search area of rectangle          # features/steps/steps.py:12
    Then I should have area of Rectangle 6   # features/steps/steps.py:17

Feature: Square # features/square.feature:1

  Scenario: aboba                          # features/square.feature:2
    Given Square has 4 cm                  # features/steps/steps.py:24
    When I search area of square           # features/steps/steps.py:29
    Then I should have area of square 16 # features/steps/steps.py:34

3 features passed, 0 failed, 0 skipped
3 scenarios passed, 0 failed, 0 skipped
9 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.002s
```

Результат выполнения программы: