Московский государственный технический университет им. Н.Э. Баумана

Факультет «Радиотехнический»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчёт по Лабораторной работе №4
Вариант №19

Выполнил:
студент группы РТ5-31Б
Сафонов Федор
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2021 г

Описание задания

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

Текст программы

- fabric.py

```
import math
from abc import ABC, abstractmethod
import unittest

import sys
import os
conf_path = os.getcwd()
print(conf_path)
sys.path.append(conf_path)
from Tests.tdd import *

class Creator(ABC):
    @abstractmethod
    def factory_method(self):
        pass

class TruckCreator(Creator):
    def factory_method(self):
        return Truck()

class ShipCreator(Creator):
    def factory_method(self):
        return Ship()

class Transport(ABC):
    @abstractmethod
    def deliver(self, *args):
```

```
        pass

    @abstractmethod
    def take_item(self, *args):
        pass

    @abstractmethod
    def count_time(self):
        pass

class Truck(Transport): # Грузовик
    speed = 10 # скорость
    dist = None # расстояние
    item = None # Товар

    def deliver(self, dist):
        self.dist = dist

    def take_item(self, item):
        self.item = item

    def count_time(self):
        return math.ceil(self.dist / self.speed) # округление в большую сторону

class Ship(Transport): # Корабль
    speed = 5 # скорость
    dist = None # расстояние
    item = None # Товар

    def deliver(self, dist):
        self.dist = dist

    def take_item(self, item):
        self.item = item

    def count_time(self):
        return math.ceil(self.dist / self.speed) # округление в большую сторону

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

- tdd.py

```
import unittest
from Lab4.fabric import *

class MyTestCase(unittest.TestCase):
    truck = None
    ship = None
    item1 = "pencil"
    item2 = "ball"
    dist1 = 100
    dist2 = 77

    @classmethod
```

```
    def setUp(self):
        self.truck = TruckCreator().factory_method()
        self.truck.take_item(self.item1)
        self.truck.deliver(self.dist1)
        self.ship = ShipCreator().factory_method()
        self.ship.take_item(self.item2)
        self.ship.deliver(self.dist2)

    def test_not_none(self):
        self.assertIsNotNone(self.ship)
        self.assertIsNotNone(self.truck)

    def test_item(self):
        self.assertEqual(self.truck.item, self.item1)
        self.assertEqual(self.ship.item, self.item2)

    def test_upper(self):
        self.assertTrue(self.truck.count_time() * self.truck.speed
                        >= self.truck.dist)
        self.assertTrue(self.ship.count_time() * self.ship.speed
                        >= self.ship.dist)

    def test_instance(self):
        self.assertIsInstance(self.truck, Transport)
        self.assertIsInstance(self.ship, Transport)


    @classmethod
    def tearDownClass(self):
        del self.ship
        del self.truck
```

- calculate.feature

```
Feature: My first feature file using radish
    In order to test my awesome software
    I need an awesome BDD tool like radish
    to test my software.

    Scenario: Test my calculator
        Given I have a truck with speed 10 km/h
        When I want it to deliver for 100 km
        Then I expect the time to be 100 h
```

- step.py

```python
from radish import given, when, then
from Lab4.fabric import *

@given("I have a truck with speed {speed:g} km/h")
def have_numbers(step, speed):
    step.context.speed = speed

@when("I want it to deliver for {dist:g} km")
def sum_numbers(step, dist):
    step.context.result = dist

@then("I expect the result to be {result:g} h")
def expect_result(step, result):
    assert TruckCreator().factory_method().count_time(step.context.dist) == result

    assert CarCreator().factory_method().count_time(step.context.dist) == result
```

Вывод программы

```
....
----------------------------------------------------------------------
Ran 4 tests in 0.000s

OK

Process finished with exit code 0
```