Московский государственный технический
университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»
Отчет по лабораторной работе №2
«Объектно-ориентированные возможности языка Python»

Выполнил:
студент группы ИУ5-31Б
Маркин Денис

Проверил:
преподаватель каф. ИУ5
Нардид Анатолий Николаевич

Москва, 2024 г.

Постановка задачи

1. Необходимо для произвольной предметной области реализовать от одного до трех шаблонов проектирования: один порождающий, один структурный и один поведенческий. Для сдачи лабораторной работы в минимальном варианте достаточно реализовать один паттерн.
2. В модульных тестах необходимо применить следующие технологии:
    - TDD - фреймворк.
    - BDD - фреймворк.
    - Создание Mock-объектов.

Текст Программы

Behavioral.py

```
from abc import ABC, abstractmethod
#State Шаблон состояния
class State(ABC):
    @abstractmethod
    def handle(self):
        pass


class Outside(State):
    def handle(self):
        return "Студент не в ВУЗе"


class Inside(State):
    def handle(self):
        return "Студент учится"


class Walk(State):
```

```
    def handle(self):
        return "Студент в пути"


class Window(State):
    def handle(self):
        return "Студент в ВУЗе, но сейчас у него окно"


class Student:
    def __init__(self):
        self.state = Outside()

    def set_state(self, state: State):
        self.state = state

    def where(self):
        return self.state.handle()

me = Student()
print(me.where())
me.set_state(Inside())
print(me.where())
me.set_state(Window())
print(me.where())
me.set_state(Outside())
print(me.where())
me.set_state(Walk())
print(me.where())
```

Creational.py

```
#Singleton Порождающий шаблон
class Shop:
    instance = None #Unity moment
    def __new__(cls):
        if cls.instance is None:
            cls.instance = super(Shop, cls).__new__(cls)
            cls.instance.products = []
        return cls.instance

    def add_product(self, _p):
        self.products.append(_p)

    def get_products(self):
        return self.products

# Пример использования
shop = Shop()
shop.add_product("Bananas")
shop.add_product("Apples")
another_shop = Shop()
another_shop.add_product("Oranges")
print(shop.get_products())
print(another_shop.get_products())
print("equal" if another_shop == shop else "mistake!")
```

Struct.py

```
from abc import ABC, abstractmethod
#Bridge Структурный
```

```
class Color(ABC): #Абстракция
    @abstractmethod
    def get_color(self):
        pass

class Red(Color): #Реализация
    def get_color(self):
        return "красный"

class Blue(Color): #Реализация
    def get_color(self):
        return "синий"

class Shape(ABC): #Абстракция с ссылкой на реализацию
    def __init__(self, color: Color):
        self.color = color
    @abstractmethod
    def info(self):
        pass

class Square(Shape): #Ну дальше ты пон
    def info(self):
        return f"Квадрат, цвет: {self.color.get_color()}"

class Circle(Shape):
    def info(self):
        return f"Окружность, цвет: {self.color.get_color()}"
```

```
red = Red()
blue = Blue()
square_red = Square(red)
circle_blue = Circle(blue)
print(square_red.info())
print(circle_blue.info())
```

test_Behavioral.py

```
import unittest

# Тест пройден Сначала тесты, потом разработка
from Behavioral import Student, Outside, Inside, Window, Walk
class TestStudent(unittest.TestCase):

    def setUp(self):
        self.student = Student()

    def test_initial_state(self):
        self.assertEqual(self.student.where(), "Студент не в ВУЗе")

    def test_state_inside(self):
        self.student.set_state(Inside())
        self.assertEqual(self.student.where(), "Студент учится")

    def test_state_window(self):
        self.student.set_state(Window())
        self.assertEqual(self.student.where(), "Студент в ВУЗе, но сейчас у
него окно")
```

```
        def test_state_outside(self):
            self.student.set_state(Outside())
            self.assertEqual(self.student.where(), "Студент не в ВУЗе")

        def test_state_walk(self):
            self.student.set_state(Walk())
            self.assertEqual(self.student.where(), "Студент в пути")

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

test_Creational.py

```
import unittest
from unittest.mock import patch
from Creational import Shop
class TestShopSingleton(unittest.TestCase):

    @patch.object(Shop, 'add_product')
    def test_add_product_using_mock(self, mock_add_product):
        shop = Shop()
        mock_add_product("Bananas")
        mock_add_product("Apples")
        self.assertEqual(mock_add_product.call_count, 2)
        shop.add_product("Bananas")
        shop.add_product("Apples")

        self.assertIn("Bananas", shop.get_products())
        self.assertIn("Apples", shop.get_products())
```

```
    def test_singleton_property(self):
        shop1 = Shop()
        shop2 = Shop()
        shop1.add_product("Grapes")
        self.assertIs(shop1, shop2)
        self.assertIn("Grapes", shop2.get_products())

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

features/steps/steps.py

```
from behave import given, when, then
from Struct import Red, Blue, Square, Circle

@given('I have a red color')
def step_given_red_color(context):
    context.color = Red()

@when('I create a square with that color')
def step_when_create_square(context):
    context.shape = Square(context.color)

@then('it should return "{expected_output}"')
def step_then_check_output(context, expected_output):
    assert context.shape.info() == expected_output
```

Результаты работы программы

```
Квадрат, цвет: красный
Окружность, цвет: синий
```

Struct.py

```
['Bananas', 'Apples', 'Oranges']
['Bananas', 'Apples', 'Oranges']
equal
```

Creational.py

```
Студент не в ВУЗе
Студент учится
Студент в ВУЗе, но сейчас у него окно
Студент не в ВУЗе
Студент в пути
```

Behavioral.py

```
Студент не в ВУЗе
Студент учится
Студент в ВУЗе, но сейчас у него окно
Студент не в ВУЗе
Студент в пути
.....
----------------------------------------------------------------------
Ran 5 tests in 0.001s

OK
```

Test_Behavioral.py

```
['Bananas', 'Apples', 'Oranges']
['Bananas', 'Apples', 'Oranges']
equal
..
----------------------------------------------------------------------
Ran 2 tests in 0.002s

OK
```

Test_Creational.py

```
Квадрат, цвет: красный
Окружность, цвет: синий
Feature: Square color # features/Shape_color.feature:1

  Scenario: Create a red square                    # features/Shape_color.feature:2
    Given I have a red color                       # features/steps/Step.py:4
    When I create a square with that color         # features/steps/Step.py:8
    Then it should return "Квадрат, цвет: красный" # features/steps/Step.py:12

1 feature passed, 0 failed, 0 skipped
1 scenario passed, 0 failed, 0 skipped
3 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.010s
```

features