Московский государственный технический университет им. Н.Э. Баумана

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

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №4 «Шаблоны проектирования и модульное тестирование в Python»

Выполнил:
студент группы ИУ5-32Б
Еремихин Владислав

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) и тесты для них.
2. В модульных тестах необходимо применить следующие технологии:
    - TDD - фреймворк.
    - BDD - фреймворк.
    - Создание Mock-объектов.

Текст программы

Файл **main.py**:

```python
from chair_factory import ModernBuilder, VintageBuilder

if __name__ == "__main__":
    while True:
        print("Выберите стиль стула для производства: \n    1: Модерн\n    2: 
Винтаж")
        choice = int(input())
        if choice in [1, 2]:
            print("Введите вес: ")
            weight = int(input())
            print("Введите цену: ")
            cost = int(input())
            print()
            if choice == 1:
                chair = ModernBuilder().build_chair(weight, cost)
            else:
                chair = VintageBuilder().build_chair(weight, cost)
            print()
            chair.print_info()
            print()
            chair.print_desc()
            print()
            print()
        else:
            print("Неверный ввод - повторите попытку.")
            print()
        print("Выберите действие: \n    1: Повторить\n    2: Выход")
        print("\n")
        action = int(input())
        if action == 2:
            break
```

Пакет **chair_factory**:

Файл **__init__.py**:

```
from .factory import Chair, ModernBuilder, ModernChair, VintageBuilder,
VintageChair
```

Файл **factory.py**:

```
from abc import ABC, abstractmethod


class Chair(ABC):
    def __init__(self, weight, cost):
        self.weight = weight
        self.cost = cost

    def print_info(self):
        print("Информация о стуле:")
        print("      Вес: " + str(self.weight))
        print("     Цена: " + str(self.cost))

    @abstractmethod
    def print_desc(self):
        pass

    @abstractmethod
    def get_style_name(self):
        pass


class ModernChair(Chair):
    def print_desc(self):
        print("Описание: ")
        print(
            "    Объединяет в себе плавность линий и элементы из разнообразных
материалов (стекло, металл и др.). \n    Такая мебель выглядит довольно необычно,
импровизационно"
        )

    def get_style_name(self):
        return "modern"


class VintageChair(Chair):
    def print_desc(self):
        print("Описание: ")
        print(
            "    Имеет ярко выраженный античный акцент. Легкость форм,
изящность\n    деталей и изысканная простота – основные характеристики мебели в
данном стиле"
        )

    def get_style_name(self):
        return "vintage"


class ChairBuilder(ABC):
    @abstractmethod
    def build_chair(self, weight, cost):
        pass


class ModernBuilder(ChairBuilder):
    def building_process(self, weight, cost):
        print("Стул в стиле модерн произведен!")
        return ModernChair(weight, cost)

    def build_chair(self, weight, cost):
        return self.building_process(weight, cost)


class VintageBuilder(ChairBuilder):
    def building_process(self, weight, cost):
        print("Стул в стиле винтаж произведен!")
        return VintageChair(weight, cost)

    def build_chair(self, weight, cost):
        return self.building_process(weight, cost)
```

Каталог **tests**:

Файл **test_chair_factory.py**:

```
import unittest
from unittest import mock
from chair_factory import (
    Chair,
    ModernBuilder,
    ModernChair,
    VintageBuilder,
    VintageChair,
)


class TestChairBuilders(unittest.TestCase):
    def test_modern_builder(self):
        self.assertIsInstance(ModernBuilder().build_chair(1, 1), ModernChair)
```

```python
    def test_vintage_builder(self):
        self.assertIsInstance(VintageBuilder().build_chair(1, 1), VintageChair)

    def test_builder(self):
        self.assertIsInstance(VintageBuilder().build_chair(1, 1), Chair)

    def test_modern_chair_constructor(self):
        chair = mock.Mock()
        chair.weight = 1
        chair.cost = 1
        result = ModernBuilder().build_chair(1, 1)
        self.assertEqual(result.weight, chair.weight)
        self.assertEqual(result.cost, chair.cost)

    def test_vintage_builder_proccess(self):
        builder = VintageBuilder()
        builder.building_process = mock.MagicMock()
        builder.build_chair(2, 2)
        builder.building_process.assert_called_once_with(2, 2)


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

Файл **features/tests.feature**:

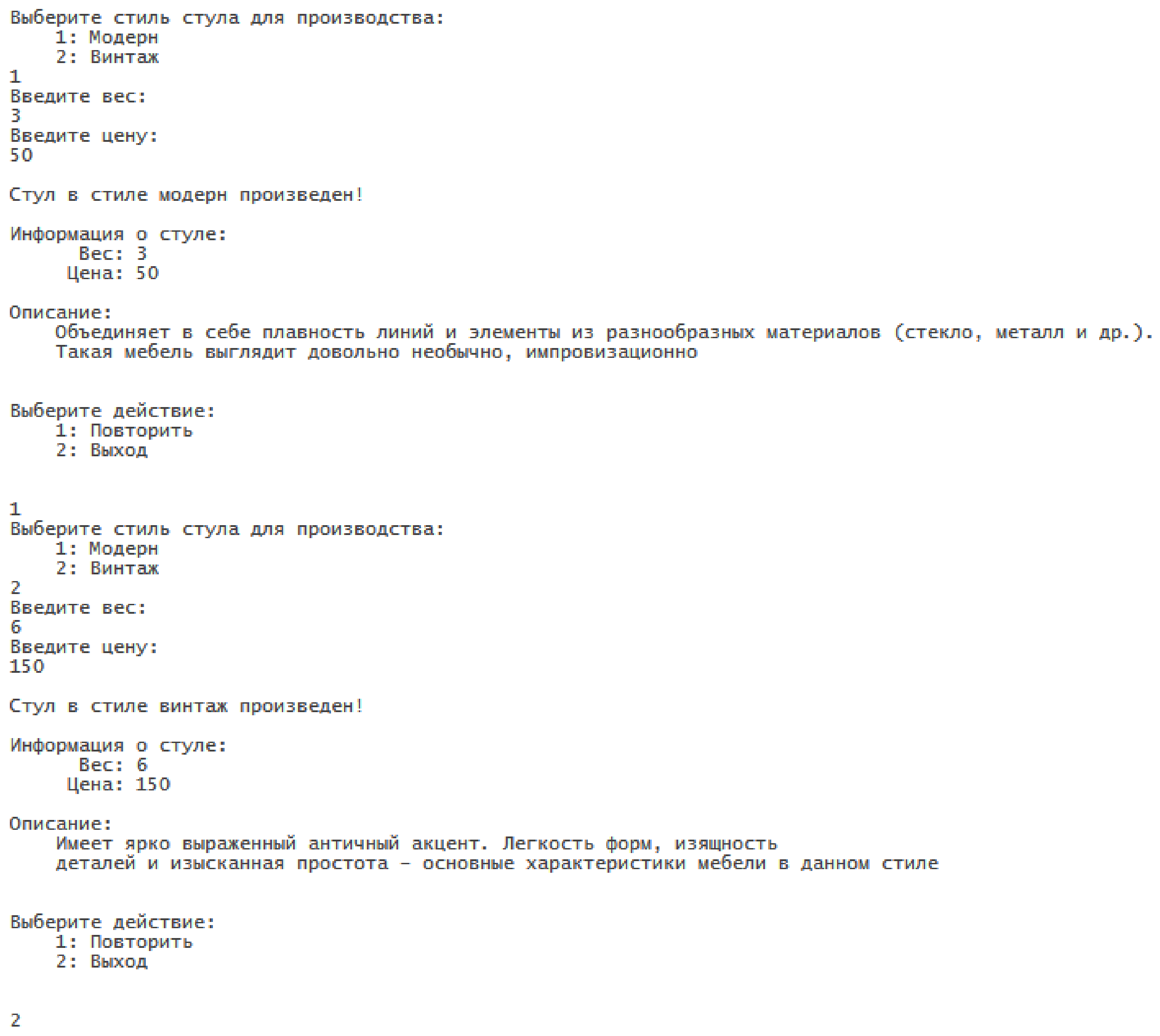
```
Feature: Modern or Vintage
  In order to extend the list of produced chairs
  As a chair factory manager
  I want my workers to decide in which factory
  they will create chairs, based on customer's style choice

  Scenario: Cheap vintage chair
    Given The chair style is vintage, weight is 3 lbs, cost is 50 dollars
    When We choosing factory we must choose appropriate one
    Then Produced chair should be vintage with given parameters: weight is 3 lbs,
cost is 50 dollars

  Scenario: Expensive vintage chair
    Given The chair style is vintage, weight is 6 lbs, cost is 350 dollars
    When We choosing factory we must choose appropriate one
    Then Produced chair should be vintage with given parameters: weight is 6 lbs,
cost is 350 dollars

  Scenario: Cheap modern chair
    Given The chair style is modern, weight is 2 lbs, cost is 55 dollars
    When We choosing factory we must choose appropriate one
    Then Produced chair should be modern with given parameters: weight is 2 lbs,
cost is 55 dollars
```

```
  Scenario: Expensive modern chair
    Given The chair style is modern, weight is 7 lbs, cost is 150 dollars
    When We choosing factory we must choose appropriate one
    Then Produced chair should be modern with given parameters: weight is 7 lbs,
cost is 150 dollars
```

Файл **features/steps/test_builders.py**:

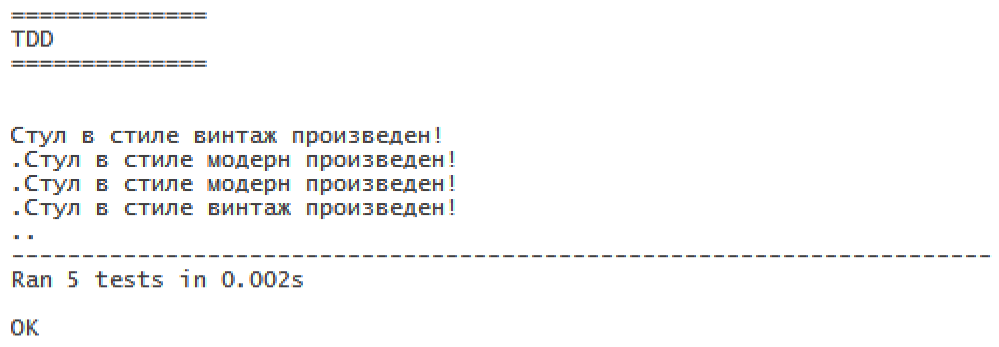
```
from behave import given, when, then
from chair_factory import VintageBuilder, ModernBuilder, ModernChair, VintageChair


@given("The chair style is {style}, weight is {weight} lbs, cost is {cost}
dollars")
def have_chair_params(context, style, weight, cost):
    context.style = style
    context.weight = weight
    context.cost = cost

@when("We choosing factory we must choose appropriate one")
def build_chair(context):
    if context.style == "vintage":
        context.chair = VintageBuilder().build_chair(context.weight, context.cost)
    if context.style == "modern":
        context.chair = ModernBuilder().build_chair(context.weight, context.cost)

@then(
    "Produced chair should be {style} with given parameters: weight is {weight}
lbs, cost is {cost} dollars"
)
def expect_result(context, style, weight, cost):
    if style == "vintage":
        assert isinstance(context.chair, VintageChair)
    if style == "modern":
        assert isinstance(context.chair, ModernChair)
    assert context.chair.weight == weight
    assert context.chair.cost == cost
    print(str(style))
    print(context.chair.get_style_name())
    assert context.chair.get_style_name() == str(style)
```

Пример выполнения программы

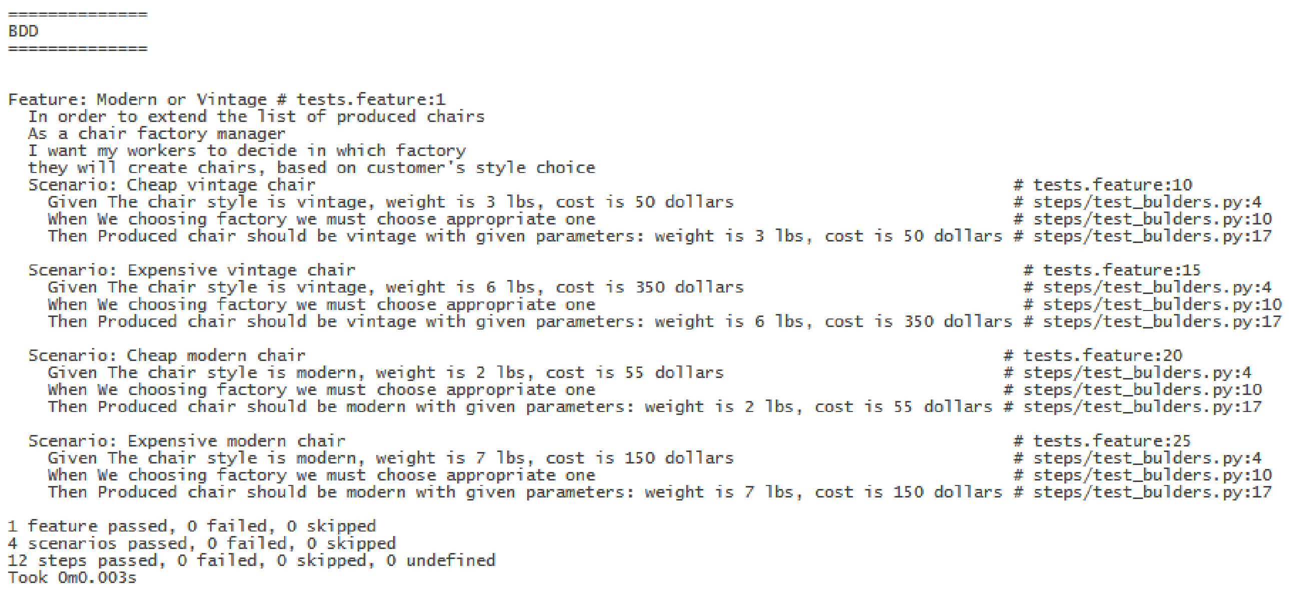


```
Выберите стиль стула для производства:
    1: Модерн
    2: Винтаж
1
Введите вес:
3
Введите цену:
50

Стул в стиле модерн произведен!

Информация о стуле:
      Вес: 3
     Цена: 50

Описание:
    Объединяет в себе плавность линий и элементы из разнообразных материалов (стекло, металл и др.).
    Такая мебель выглядит довольно необычно, импровизационно


Выберите действие:
    1: Повторить
    2: Выход


1
Выберите стиль стула для производства:
    1: Модерн
    2: Винтаж
2
Введите вес:
6
Введите цену:
150

Стул в стиле винтаж произведен!

Информация о стуле:
      Вес: 6
     Цена: 150

Описание:
    Имеет ярко выраженный античный акцент. Легкость форм, изящность
    деталей и изысканная простота - основные характеристики мебели в данном стиле


Выберите действие:
    1: Повторить
    2: Выход


2
```

Результаты тестов:



```
===============
TDD
===============


Стул в стиле винтаж произведен!
.Стул в стиле модерн произведен!
.Стул в стиле модерн произведен!
.Стул в стиле винтаж произведен!
..
----------------------------------------------------------------------
Ran 5 tests in 0.002s

OK
```



```
==============
BDD
==============


Feature: Modern or Vintage # tests.feature:1
  In order to extend the list of produced chairs
  As a chair factory manager
  I want my workers to decide in which factory
  they will create chairs, based on customer's style choice
  Scenario: Cheap vintage chair                                                                       # tests.feature:10
    Given The chair style is vintage, weight is 3 lbs, cost is 50 dollars                            # steps/test_bulders.py:4
    When We choosing factory we must choose appropriate one                                          # steps/test_bulders.py:10
    Then Produced chair should be vintage with given parameters: weight is 3 lbs, cost is 50 dollars # steps/test_bulders.py:17

  Scenario: Expensive vintage chair                                                                    # tests.feature:15
    Given The chair style is vintage, weight is 6 lbs, cost is 350 dollars                            # steps/test_bulders.py:4
    When We choosing factory we must choose appropriate one                                           # steps/test_bulders.py:10
    Then Produced chair should be vintage with given parameters: weight is 6 lbs, cost is 350 dollars # steps/test_bulders.py:17

  Scenario: Cheap modern chair                                                                       # tests.feature:20
    Given The chair style is modern, weight is 2 lbs, cost is 55 dollars                            # steps/test_bulders.py:4
    When We choosing factory we must choose appropriate one                                         # steps/test_bulders.py:10
    Then Produced chair should be modern with given parameters: weight is 2 lbs, cost is 55 dollars # steps/test_bulders.py:17

  Scenario: Expensive modern chair                                                                    # tests.feature:25
    Given The chair style is modern, weight is 7 lbs, cost is 150 dollars                            # steps/test_bulders.py:4
    When We choosing factory we must choose appropriate one                                          # steps/test_bulders.py:10
    Then Produced chair should be modern with given parameters: weight is 7 lbs, cost is 150 dollars # steps/test_bulders.py:17

1 feature passed, 0 failed, 0 skipped
4 scenarios passed, 0 failed, 0 skipped
12 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.003s
```