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
«Основные конструкции языка Python»

Выполнил:		Проверил:
студент группы ИУ5-32Б		преподаватель каф. ИУ5
Панов Г.Д.		Гапанюк Ю. Е.
Подпись и дата		Подпись и дата

Москва, 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 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.

Листинг программы:

main.py

```
from __future__ import annotations
from abc import ABC, abstractmethod


class Developer(ABC):

    @abstractmethod
    def factory_method(self):
        pass

    def some_operation(self) -> str:
        # Вызываем фабричный метод, чтобы получить объект-продукт
        product = self.factory_method()

        # Работаем с этим продуктом
        result = f"Creator: Today the company has created a new game:
{product.operation()}"

        return result


class ComputerDeveloper(Developer):
    def factory_method(self) -> Game:
        return ComputerGame()


class MobileDeveloper(Developer):
    def factory_method(self) -> Game:
        return MobileGame()
```

```
class Game(ABC):
    @abstractmethod
    def operation(self) -> str:
        pass


class ComputerGame(Game):
    def operation(self) -> str:
        return "Computer game created"


class MobileGame(Game):
    def operation(self) -> str:
        return "Mobile game created"


def client_code(creator: Developer):

    # print(f"Client: I'm not aware of the creator's class, but it still
works.\n"
    #       f"{creator.some_operation()}", end="")
    result = (f"Client: I'm not aware of the creator's class, but it still
works.\n"
          f"{creator.some_operation()}")
    return result


if __name__ == "__main__":
    print("App: Launched with the ComputerDeveloper.")
    print(client_code(ComputerDeveloper()))
    print("\n")

    print("App: Launched with the MobileDeveloper.")
    print(client_code(MobileDeveloper()))
    print("\n")
```

tests_tdd.py

```
import unittest
from main import client_code
from main import ComputerDeveloper
from main import MobileDeveloper
from unittest.mock import patch



class TestGameCreators(unittest.TestCase):
    def test_computer_developer(self):
        expected_result = "Client: I'm not aware of the creator's class,
but it still works." \
                          "\nCreator: Today the company has created a new
game: Computer game created"
        actual_result = client_code(ComputerDeveloper())
        self.assertEqual(expected_result, actual_result, "Something is
wrong with creating a computer game object")

    def test_mobile_developer(self):
        expected_result = "Client: I'm not aware of the creator's class,
```

```
but it still works." \
                            "\nCreator: Today the company has created a new
game: Mobile game created"
        actual_result = client_code(MobileDeveloper())
        self.assertEqual(expected_result, actual_result, "Something is
wrong with creating a mobile game object")


class TestComputerDeveloper(unittest.TestCase):
    @patch('main.ComputerGame.operation', return_value="Computer game
created")
    def test_operation(self, operation):
        self.assertEqual(operation(self), "Computer game created")




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

tests_bdd.py

```
import pytest
from pytest_bdd import scenario, given, when, then

@scenario("factory_method.feature", "Creation of new development
departments")
def test_start_working():
    pass

@given("Need to create a game")
def test_creation():
    pass

@when("The developer creates a game")
def test_transfer_to_client():
    pass

@then("We write that the game has been successfully created")
def test_result():
    pass
```

factory_method.feature

```
Feature: GamesDeveloper
    A game constructor pattern for developer.

    Scenario: Creation of new development departments
        Given Need to create a game
        When  The developer creates a game
        Then We write that the game has been successfully created
```

Пример работы:

```
C:\Users\User\PycharmProjects\lab04\venv\Scripts\python.exe C:/Users/User/PycharmProjects/lab04/main.py
App: Launched with the ComputerDeveloper.
Client: I'm not aware of the creator's class, but it still works.
Creator: Today the company has created a new game: Computer game created


App: Launched with the MobileDeveloper.
Client: I'm not aware of the creator's class, but it still works.
Creator: Today the company has created a new game: Mobile game created
```

```
✔ Tests passed: 3 of 3 tests – 0 ms
C:\Users\User\PycharmProjects\lab04\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm 2021.2
 .2\plugins\python\helpers\pycharm\_jb_pytest_runner.py" --path C:/Users/User/PycharmProjects/lab04/tests_tdd.py
Testing started at 21:08 ...
Launching pytest with arguments C:/Users/User/PycharmProjects/lab04/tests_tdd.py --no-header --no-summary -q in
 C:\Users\User\PycharmProjects\lab04

============================= test session starts =============================
collecting ... collected 3 items

tests_tdd.py::TestGameCreators::test_computer_developer PASSED           [ 33%]
tests_tdd.py::TestGameCreators::test_mobile_developer PASSED             [ 66%]
tests_tdd.py::TestComputerDeveloper::test_operation PASSED               [100%]

============================== 3 passed in 0.05s ==============================

Process finished with exit code 0
```

```
✔ Tests passed: 4 of 4 tests – 0 ms
C:\Users\User\PycharmProjects\lab04\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm 2021.2
 .2\plugins\python\helpers\pycharm\_jb_pytest_runner.py" --path C:/Users/User/PycharmProjects/lab04/tests_bdd.py
Testing started at 21:08 ...
Launching pytest with arguments C:/Users/User/PycharmProjects/lab04/tests_bdd.py --no-header --no-summary -q in
 C:\Users\User\PycharmProjects\lab04

============================= test session starts =============================
collecting ... collected 4 items

tests_bdd.py::test_creation PASSED                                       [ 25%]
tests_bdd.py::test_transfer_to_client PASSED                             [ 50%]
tests_bdd.py::test_result PASSED                                         [ 75%]
tests_bdd.py::test_start_working <- venv\lib\site-packages\pytest_bdd\scenario.py PASSED [100%]

============================== 4 passed in 0.02s ==============================

Process finished with exit code 0
```