



**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
по дисциплине «Базовые компоненты интернет-технологий»**

**Выполнила:
студентка группы ИУ5- 32Б
Андреева А. А.**

**Проверил:
Канев А.И.**

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.

Текст программы

Порождающий шаблон проектирования Абстрактная фабрика

main.py

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

class DigitalFactory(ABC):

    @abstractmethod
    def createPhone(self) -> Phone:
        pass

    @abstractmethod
    def createLaptop(self) -> Laptop:
        pass

class AppleFactory(DigitalFactory):

    def createPhone(self) -> Phone:
        return iPhone()

    def createLaptop(self) -> Laptop:
        return MacBook()

class SamsungFactory(DigitalFactory):

    def createPhone(self) -> Phone:
        return SamsungGalaxy()

    def createLaptop(self) -> Laptop:
        return SamsungBook()

class Phone(ABC):

    @abstractmethod
    def chargePhone(self) -> str:
        pass

class iPhone(Phone):

    def chargePhone(self) -> str:
        return "Charging iPhone."

class SamsungGalaxy(Phone):

    def chargePhone(self) -> str:
        return "Charging Samsung Galaxy."

class Laptop(ABC):

    @abstractmethod
    def chargeLaptop(self) -> None:
        pass

    @abstractmethod
    def usbPhone(self, collaborator: Phone) -> None:
```

```

        pass

class MacBook(Laptop):
    def chargeLaptop(self) -> str:
        return "Charging Macbook."

    def usbPhone(self, collaborator: Phone) -> str:
        result = collaborator.chargePhone()
        return f"MacBook connected with iPhone and ({result})"

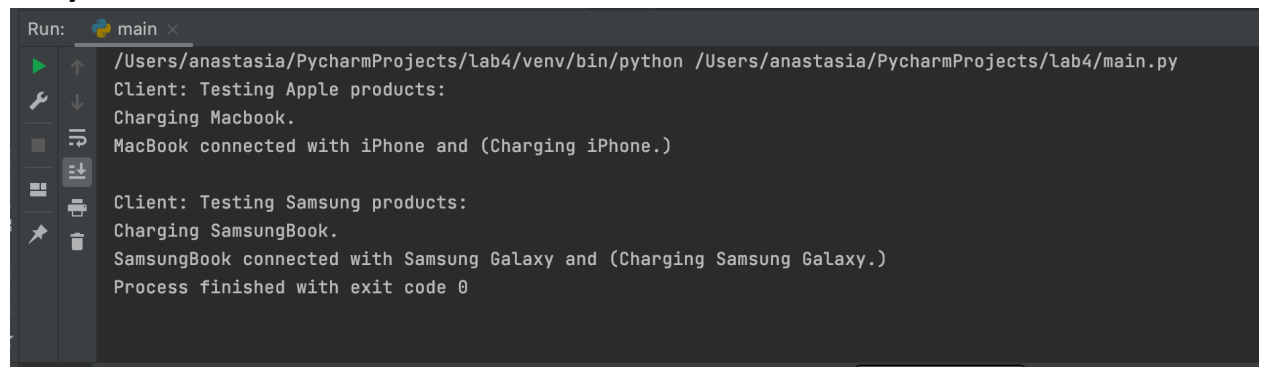
class SamsungBook(Laptop):
    def chargeLaptop(self) -> str:
        return "Charging SamsungBook."
    def usbPhone(self, collaborator: Phone) -> str:
        result = collaborator.chargePhone()
        return f"SamsungBook connected with Samsung Galaxy and ({result})"

def Store(factory: DigitalFactory) -> None:
    phone = factory.createPhone()
    laptop = factory.createLaptop()

    print(f"{laptop.chargeLaptop()}")
    print(f"{laptop.usbPhone(phone)}", end="")
if __name__ == '__main__':
    print("Client: Testing Apple products:")
    Store(AppleFactory())
    print("\n")
    print("Client: Testing Samsung products:")
    Store(SamsungFactory())

```

Результат



```

Run: main x
/Users/anastasia/PycharmProjects/lab4/venv/bin/python /Users/anastasia/PycharmProjects/lab4/main.py
Client: Testing Apple products:
Charging Macbook.
MacBook connected with iPhone and (Charging iPhone.)

Client: Testing Samsung products:
Charging SamsungBook.
SamsungBook connected with Samsung Galaxy and (Charging Samsung Galaxy.)
Process finished with exit code 0

```

Тесты

tdd.py

```

import unittest
from main import *

class test_DigitalFactory(unittest.TestCase):

    def test_ab_factory_is_working(self):
        factory = AppleFactory
        phone = factory.createPhone(self)
        laptop = factory.createLaptop(self)

```

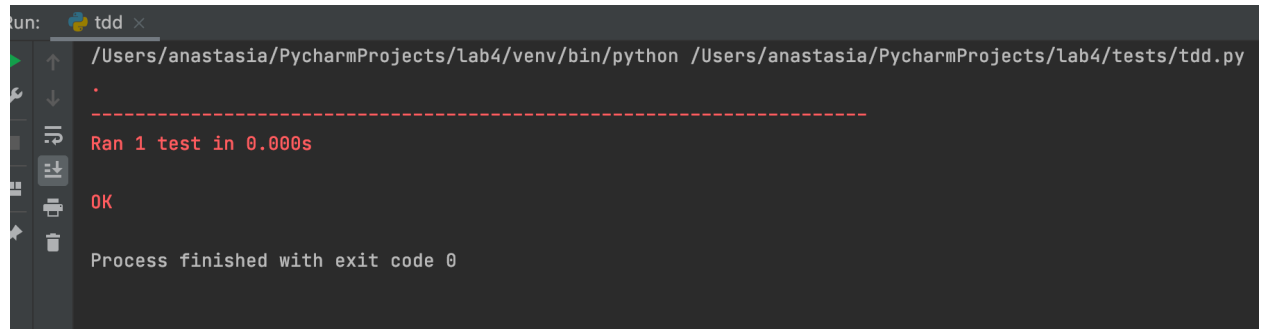
```

        self.assertEqual(laptop.usbPhone(phone), "MacBook connected with
iPhone and (Charging iPhone.)")

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

```

Результат



```

run: tdd x
/Users/anastasia/PycharmProjects/lab4/venv/bin/python /Users/anastasia/PycharmProjects/lab4/tests/tdd.py
.
-----
Ran 1 test in 0.000s
OK
Process finished with exit code 0

```

features.steps.test_feature.py

```

from behave import *
from tdd_test.TDD_test import *

@given("I have Mouse for 800 rubles and Keyboard for 1920 rubles")
def have_prices(context):
    context.a = TestPartCost()

@when("I put them into ControlDivices")
def ControlDivices_combine(context):
    context.a.test_part_cost_is_working()

@then("I expect ControlDivices to cost 2720 rubles")
def check_result(context):
    pass

```

test_feature.feature

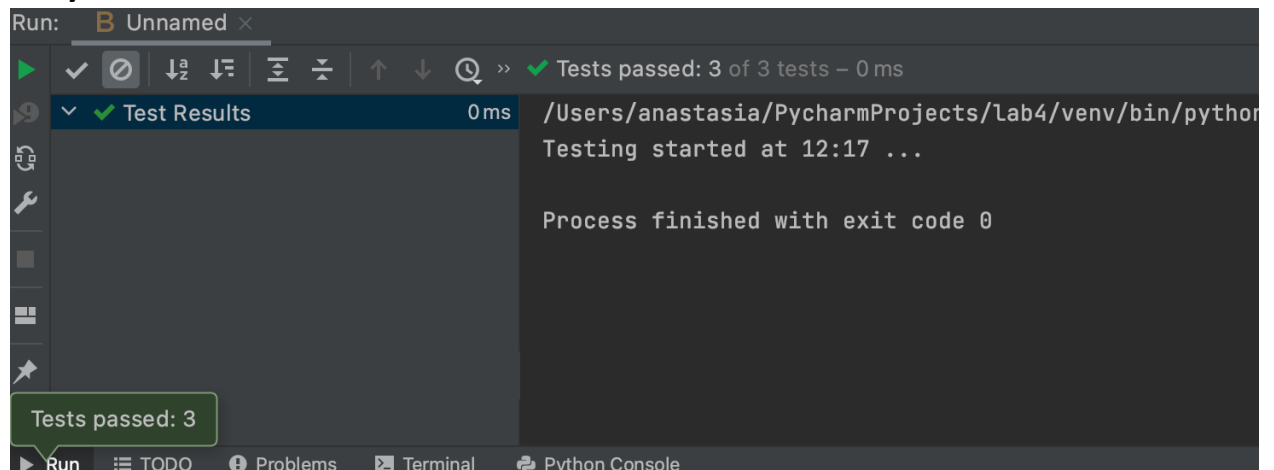
```

Feature: Test

    Scenario: Test function
        Given We have a store when we can charge phone from its factory laptop
        When We choose name of the factory Apple factory
        Then We expect iPhone connected with Macbook and charging

```

Результат



moke_test.py

```
import unittest
from unittest.mock import patch
from main import *

class TestDigitalFactory(unittest.TestCase):
    @patch('main.MacBook.chargeLaptop', return_value = 'Charge Android')
    def test_charge_macbook(self, chargeLaptop):
        factory = AppleFactory
        laptop = factory.createLaptop(self)

        self.assertEqual(laptop.chargeLaptop(), 'Charge Android')
# Он заменяет фактическую функцию chargeLaptop ложной функцией, которая ведет
# себя именно так, как мы хотим.
# В течение всего теста функция chargeLaptop заменяется на mock

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

Результат

