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)

Базовые компоненты интернет технологий

Отчет по лабораторной работе №4

Студент: Макеев В. А.
Группа: ИУ5Ц-54Б

Преподаватель: Гапанюк Ю. Е.

2021 г.

Лабораторная работа №4

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

1. builder.py

```
from abc import ABC, abstractmethod
from enum import Enum, auto


class BrandType(Enum):
    MSI = auto()
    ASUS = auto()
    ACER = auto()
    HP = auto()

class CpuType(Enum):
    I5 = auto()
    I7 = auto()
    I9 = auto()

class RamType(Enum):
    RAM8 = auto()
    RAM16 = auto()

class CardType(Enum):
    GTX1650 = auto()
    GTX1650ti = auto()
    GTX1660ti = auto()
    RTX2060 = auto()

class Laptop:
    def __init__(self, name):
        self.name = name
        self.brand = None
        self.cpu = None
        self.ram = None
        self.card = None
        self.cost = None

    def __str__(self):
       info: str = f"Laptop name: {self.name} \n" \
                   f"{self.brand} \n" \
                   f"{self.cpu} \n" \
                   f"{self.ram} \n" \
                   f"{self.card} \n" \
                   f"Cost: {self.cost} rub"

       return info
```

```python
class Builder(ABC):

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

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

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

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


class MSIGL62LapBuilder(Builder):

    def __init__(self):
        self.laptop = Laptop("MSI GL62")
        self.laptop.cost = 50000

    def add_brand(self) -> None:
        self.laptop.brand = BrandType.MSI

    def add_cpu(self) -> None:
        self.laptop.cpu = CpuType.I5

    def add_ram(self) -> None:
        self.laptop.ram = RamType.RAM8

    def add_card(self) -> None:
        self.laptop.card = CardType.GTX1650

    def get_lap(self) -> Laptop:
        return self.laptop


class ASUSTUFLapBuilder(Builder):

    def __init__(self):
        self.laptop = Laptop("ASUS TUF")
        self.laptop.cost = 100000

    def add_brand(self) -> None:
        self.laptop.brand = BrandType.ASUS

    def add_cpu(self) -> None:
        self.laptop.cpu = CpuType.I7

    def add_ram(self) -> None:
        self.laptop.ram = RamType.RAM16
```

```
    def add_card(self) -> None:
        self.laptop.card = CardType.RTX2060

    def get_lap(self) -> Laptop:
        return self.laptop


class Director:
    def __init__(self):
        self.builder = None

    def set_builder(self, builder: Builder):
        self.builder = builder

    def make_lap(self):
        if not self.builder:
            raise ValueError("Builder didn't set")
        self.builder.add_brand()
        self.builder.add_cpu()
        self.builder.add_ram()
        self.builder.add_card()


def check_cost(name1):
    for it1 in (MSIGL62LapBuilder, ASUSTUFLapBuilder):
        director1 = Director()
        builder1 = it1()
        director1.set_builder(builder1)
        director1.make_lap()
        laptop1 = builder1.get_lap()
        if laptop1.name == name1:
            return laptop1.cost

def sum_cost(x):
    for it1 in (MSIGL62LapBuilder, ASUSTUFLapBuilder):
        director1 = Director()
        builder1 = it1()
        director1.set_builder(builder1)
        director1.make_lap()
        laptop1 = builder1.get_lap()
        x = x + laptop1.cost
    return x


if __name__ == "__main__":
    print("Объекты:")
    director = Director()
    for it in (MSIGL62LapBuilder, ASUSTUFLapBuilder):
        builder = it()
        director.set_builder(builder)
```

```
        director.make_lap()
        laptop = builder.get_lap()
        print(laptop)
        print('---------------')
    name = "ASUS TUF"
    print(name, "Cost:", check_cost(name))
    x = 0
    print('sum = ', sum_cost(x))
```

2. mock.py

```
from builder import *
from unittest import TestCase
from unittest.mock import patch
import unittest

class TestCost(TestCase):
    @patch('builder.sum_cost', return_value=150000)
    def test_sum_cost(self, x):
        self.assertEqual(sum_cost(0), 150000)

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

3. tdd.py

```
import unittest
import sys, os

sys.path.append(os.getcwd())
from builder import *

class TestCost(unittest.TestCase):

    def test_cost(self):
        self.assertEqual(check_cost("ASUS TUF"), 100000)
    def test_cost1(self):
        self.assertEqual(check_cost("MSI GL62"), 50000)

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

4. steps.py

```
from behave import given, when, then
```

```
from builder import *

@given('I have sum = {x:g}')
def step(context, x):
    context.x = x

@when('I sum the cost')
def step(context):
    context.x = sum_cost(context.x)

@then('I expect to get result = {result:g}')
def step(context, result):
    assert context.x == result
```

5. build.feature

```
Feature: Test

  Scenario: Test sum_cost
    Given  I have sum = 0
    When I sum the cost
    Then I expect to get result = 150000
```

Тест программы

1. **builder.py**

```
Объекты:
Laptop name: MSI GL62
BrandType.MSI
CpuType.I5
RamType.RAM8
CardType.GTX1650
Cost: 50000 rub
---------------
Laptop name: ASUS TUF
BrandType.ASUS
CpuType.I7
RamType.RAM16
CardType.RTX2060
Cost: 100000 rub
---------------
ASUS TUF Cost: 100000
sum =  150000
```

2. **mock.py**

```
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
```

3. **tdd.py**

```
Ran 2 tests in 0.001s

OK
```

4. **steps.py**

```
PS C:\Users\vital\OneDrive - bmstu.ru\Рабочий стол\учебы\БКИТ\lab_4> behave
Feature: Test # features/steps/build.feature:1

  Scenario: Test sum_cost                # features/steps/build.feature:3
    Given I have sum = 0                 # features/steps/steps.py:4
    When I sum the cost                  # features/steps/steps.py:8
    Then I expect to get result = 150000 # features/steps/steps.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 0m0.002s
```