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

Выполнил:
студент группы ИУ5-31Б
Коваленко
Алексей Викторович

Подпись:_____

Дата:_____

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий
Евгеньевич

Подпись:_____

Дата:_____

Москва, 2021 г.

Лабораторная работа №1

Описание задания

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

Папка patterns

Файл Bulder.py

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

class Builder(ABC):

    @property
    @abstractmethod
    def product(self) -> None:
        pass

    @abstractmethod
    def produce_hammer(self,amount: int) -> None:
        pass

    @abstractmethod
    def produce_screwdriver(self,amount: int) -> None:
        pass

    @abstractmethod
    def produce_screw(self,amount: int,type: string) -> None:
        pass

    @abstractmethod
    def produce_nail(self,amount: int,type: string) -> None:
        pass

    @abstractmethod
    def produce_drill(self,amount: int,model: string) -> None:
        pass

    @abstractmethod
    def produce_drill_head(self,amount: int,type: string) -> None:
        pass

    @abstractmethod
    def produce_plank(self,amount: int, type: string) -> None:
        pass



class CartBuilder(Builder):

    def __init__(self) -> None:
        self.reset()

    def reset(self) -> None:
        self._product = Product1()

    @property
    def product(self) -> Product1:
        return self._product

    def produce_hammer(self,amount: int) -> None:
```

```
        self._product.add({'hammer':amount})

    def produce_screwdriver(self,amount: int) -> None:
        self._product.add({'screwdriver':amount})

    def produce_screw(self,amount: int,type: string) -> None:
        self._product.add({'screw':amount,'size':type})

    def produce_nail(self,amount: int,type: string) -> None:
        self._product.add({'nail':amount,'size':type})

    def produce_drill(self,amount: int,model: string) -> None:
        self._product.add({'drill':amount,'model':model})

    def produce_drill_head(self,amount: int,type: string) -> None:
        self._product.add({'drillhead':amount,'type':type})

    def produce_plank(self,amount: int, type:string) -> None:
        self._product.add({'plank':amount,'material':type})



class Product1():

    def __init__(self) -> None:
        self.size = 0
        self.parts = []

    def add(self, part: Any) -> None:
        self.size+=1
        self.parts.append(part)

    def print_info(self) -> None:
        print(f"Размер корзины:{self.size} \nТовары корзины:")
        for  i in range(len(self.parts)):
            print (', '.join([f'{key.capitalize()}: {value}' for key, value
in self.parts[i].items()]))


    def get_amount(self):
        return self.size

    def get_cart(self):
        return self.parts


class Director:


    def __init__(self) -> None:
        self._builder = None

    @property
    def builder(self) -> Builder:
        return self._builder

    @builder.setter
    def builder(self, builder: Builder) -> None:
        self._builder = builder


    def build_hammer_cart(self) -> None:
        self.builder.produce_hammer(1)
```

```
        self.builder.produce_nail(100,'medium')

    def build_drill_cart(self) -> None:
        self.builder.produce_drill(1,'classic A1')
        self.builder.produce_drill_head(2,'small')
        self.builder.produce_drill_head(2,'medium')
        self.builder.produce_drill_head(2,'large')

    def build_screw_cart(self) -> None:
        self.builder.produce_screwdriver(2)
        self.builder.produce_screw(50,'small')
        self.builder.produce_screw(50,'medium')

    def build_starter_kit(self) -> None:
        self.builder.produce_hammer(1)
        self.builder.produce_drill(1,'classic A1')
        self.builder.produce_drill_head(1,'small')
        self.builder.produce_drill_head(1,'medium')
        self.builder.produce_drill_head(1,'large')
        self.builder.produce_screwdriver(2)



def Auto_create():
    director=Director()
    builder = CartBuilder()
    director.builder=builder
    print ("Список товаров: nail, drill, drill head, hammer, screwdriver,
screw, plank.\n Также список готорвых наборов:/"
           " starter kit, drill kit, hammer kit, screw kit.\n Для выхода
введите 0.")
    while (True):
        item = str(input("Введите элемент который вы хотите добавить: "))
        match item:
            case "hammer":
                amount = (int(input("Введите количество: ")))
                builder.produce_hammer(amount)
            case "nail":
                amount = (int(input("Введите количество: ")))
                size = (str(input("Введите размер: ")))
                builder.produce_nail(amount,size)
            case "screwdriver":
                amount = (int(input("Введите количество: ")))
                builder.produce_screwdriver(amount)
            case "screw":
                amount = (int(input("Введите количество: ")))
                size = (str(input("Введите размер: ")))
                builder.produce_screw(amount, size)
            case "drill":
                amount = (int(input("Введите количество: ")))
                size = (str(input("Введите модель: ")))
                builder.produce_drill(amount, size)
            case "drillhead":
                amount = (int(input("Введите количество: ")))
                size = (str(input("Введите размер: ")))
                builder.produce_drill_head(amount, size)
            case "plank":
                amount = (int(input("Введите количество: ")))
                size = (str(input("Введите древесину: ")))
                builder.produce_plank(amount, size)
            case "starter kit":
                director.build_starter_kit()
            case "drill kit":
                director.build_drill_cart()
```

```
            case "hammer kit":
                director.build_hammer_cart()
            case "screw kit":
                director.build_screw_cart()
            case "0":
                break
            case _:
                print("Ошибка запроса, введите корректные данные!")

    return builder.product

def pars_build(builder, director, name, *args):
    match name:
        case "hammer":
            amount = args[0]
            builder.produce_hammer(amount)
        case "nail":
            amount = args[0]
            size = args[1]
            builder.produce_nail(amount,size)
        case "screwdriver":
            amount = args[0]
            builder.produce_screwdriver(amount)
        case "screw":
            amount = args[0]
            size = args[1]
            builder.produce_screw(amount, size)
        case "drill":
            amount = args[0]
            size = args[1]
            builder.produce_drill(amount, size)
        case "drillhead":
            amount = args[0]
            size = args[1]
            builder.produce_drill_head(amount, size)
        case "plank":
            amount = args[0]
            size = args[1]
            builder.produce_plank(amount, size)
        case "starter kit":
            director.build_starter_kit()
        case "drill kit":
            director.build_drill_cart()
        case "hammer kit":
            director.build_hammer_cart()
        case "screw kit":
            director.build_screw_cart()
        case _:
            return builder
    return builder
```

Файл build tests.py

```
import Bulder

builder1 = Bulder.CartBuilder()
director = Bulder.Director()
director.builder = builder1

director.build_hammer_cart()
result=builder1.product

def test_amount():
```

```
    assert result.get_amount() == 2

def test_cart():

    assert result.get_cart() == [{'hammer':1},{'nail':100,'size':'medium'}]
```

Файл mocktest.py

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

class TestBuilder(TestCase):
    @patch('Bulder.CartBuilder')
    def test_pars_built(self, b):
        secb = CartBuilder()
        secb.produce_hammer(3)
        secb.produce_screwdriver(2)
        b.product.return_value = secb.product
        prod = Product1()
        prod.size=2
        prod.parts=[{'hammer':3},{'screwdriver':2}]
        res =b.product()
        self.assertIsNotNone(res)
        self.assertIsInstance(res, Product1)
        self.assertEqual(res.parts,prod.parts)
```

Папка Features

Файл build.feature

```
Feature: My first tests as Cart Builder tests.

  Scenario: Build basic cart
      Given I want to order starter kit and 4 oak planks
      When I end my order
      Then I expect to see cart with 7 items, which consist of 1 hammer, 1
drill, 1 small  drill head, 1 medium  drill head, 1 big drill  head, 2
screwdrivers, 4 oak planks.
```

Папка Steps

Файл step.py

```
# -*- coding: utf-8 -*-
from behave import given, when, then
import patterns.Bulder
import sys

builder = patterns.Bulder.CartBuilder()
director = patterns.Bulder.Director()
director.builder = builder

@given(u'I want to order {object1} and {amount} {material} {object2}s')
def step_impl(context, object1, amount, material, object2):
    context.object1 = object1
    context.amount = amount
    context.material = material
    context.object2 = object2
```

```
@when(u'I end my order')
def step_impl(context):
    context.cart=patterns.Bulder.pars_build(builder,director,context.object1)

context.cart=patterns.Bulder.pars_build(builder,director,context.object2,cont
ext.amount,context.material)



@then(u'I expect to see cart with 7 items, which consist of {amount1}
{object1}, {amount2} {object2}, {amount3} \
{size1}  {object3}, {amount4} {size2}  {object4}, {amount5} {size3}
{object5}, {amount6} {object6}s, {amount7} {material} {object7}s.')
def step_impl(context, amount1, object1, amount2, object2, amount3, size1,
object3, amount4, size2, object4, amount5, size3, object5, amount6,\
                  object6, amount7, material, object7):
    result = patterns.Bulder.pars_build(builder,director,object1,amount1)
    result =
patterns.Bulder.pars_build(builder,director,object2,amount2,'classic')
    result =
patterns.Bulder.pars_build(builder,director,object3,amount3,size1)
    result =
patterns.Bulder.pars_build(builder,director,object4,amount4,size2)
    result =
patterns.Bulder.pars_build(builder,director,object5,amount5,size3)
    result = patterns.Bulder.pars_build(builder,director,object6,amount6)
    result =
patterns.Bulder.pars_build(builder,director,object7,amount7,material)

    assert context.cart == result
```

Экранные формы с примерами выполнения программы

TDD:

```
Launching pytest with arguments E:/Oreo/4lab/patterns/build tests.py --no-header --no-summary -q

============================= test session starts ==============================
collecting ... collected 2 items

build tests.py::test_amount PASSED                                      [ 50%]
build tests.py::test_cart PASSED                                        [100%]

============================== 2 passed in 0.08s ===============================

Process finished with exit code 0
```

BDD:

```
    When I end my order
    Then I expect to see cart with 7 items, which consist of 1 hammer, 1 drill, 1 small  drill head, 1 medium  drill head, 1 big drill  head

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
```

Mock:

```
============================= test session starts ==============================
collecting ... collected 1 item

mocktest.py::TestBuilder::test_pars_built PASSED                        [100%]

============================== 1 passed in 0.41s ===============================

Process finished with exit code 0
```