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:
студент группы ИУ5-
32Б
Зайцев А.Д.

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. Для сдачи лабораторной работы в минимальном варианте достаточно реализовать один паттерн.
2) В модульных тестах необходимо применить следующие технологии:
    a. TDD - фреймворк.
    b. BDD - фреймворк.
    c. Создание Mock-объектов.

Текст программы:

Файл Logistic_company.py:

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

'''Создатель транспорта'''
class Creator(ABC):



    @abstractmethod
    def factory_method(self,delivertype,boxtype,weight):

        pass

    def creation(self,delivertype,boxtype,weight):
        product = self.factory_method(delivertype,boxtype,weight)
        product=product.deliver()
        return product #в данном случае возвращает строку с информацией о
доставке


class TruckCreator(Creator):


    def factory_method(self,delivertype,boxtype,weight) -> Transport:
        return Truck(delivertype,boxtype,weight)


class ShipCreator(Creator):
    def factory_method(self,delivertype,boxtype,weight) -> Transport:
        return Ship(delivertype,boxtype,weight)


'''Создатель транспорта'''
'''Транспор'''
class Transport(ABC):
```

```
    @abstractmethod
    def deliver(self,delivertype,boxtype,weight):
        pass

    def __init__(self,deliverytype,boxtype,weight):
        self.deliverytype=deliverytype
        self.boxtype=boxtype
        self.weight=weight

class Truck(Transport):

    def deliver(self):
        result = "Ваша посылка отправлена {0}\nТип доставки: {1}\nТип коробки:
{2}\nВес посылки {3} кг".format("грузовой
машиной",self.deliverytype,self.boxtype,self.weight)
        return result

class Ship(Transport):
    def deliver(self):
        result = "Ваша посылка отправлена {0}\nТип доставки: {1}\nТип коробки:
{2}\nВес посылки {3} кг".format("морским
судном",self.deliverytype,self.boxtype,self.weight)
        return result


'''Транспорт'''
```

Файл test_Truck.py:

```
import unittest
import unittest.mock
from Logistic_company import Truck


class TruckTest(unittest.TestCase):

    def test_weight(self):
        a=Truck(0,0,57)
        self.assertEqual(a.weight,57)
    def test_delivertype(self):
        a=Truck("1",0,0)
        self.assertEqual(a.deliverytype,"1")
    def test_boxtype(self):
        a=Truck(0,"1",0)
        self.assertEqual(a.boxtype,"1")
    def test_weight_mock(self):
        a=Truck(0,0,57)
        m=unittest.mock.Mock()
        m.weight = 57
        self.assertEqual(m.weight,a.weight)
```

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

Файл main.py:

```
import Logistic_company


if __name__ == "__main__":
    print("Выберите вид транспорта для доставки:\n1)Доставка наземным
транспортом\n2)Доставка морским транспортом")
    transporttype=int(input())
    print("Выберите тип доставки:\n1)Экспресс\n2)Обычная")
    deliverytype=int(input())
    if deliverytype == 1:
        deliverytype = "экспресс"
    else:
        deliverytype = "обычная"
    print("Выберите тип коробки:\n1)Безопасная(для хрупких грузов)\n2)Обычная")
    boxtype=int(input())
    if boxtype == 1:
        boxtype = "безопасная(для хрупких грузов)"
    else:
        boxtype = "обычная"
    print("Введите вес посылки в килограммах")
    weight=int(input())
    if transporttype == 1 :
        print(Logistic_company.TruckCreator().creation(deliverytype,boxtype,weigh
t))
    else:
        print(Logistic_company.ShipCreator().creation(deliverytype,boxtype,weight
))
```

Файл testBDD.py:

```
from behave import Given,When,Then
from Logistic_company import TruckCreator



@Given("ordering delivery for box with parametres deliverytype: {a} boxtype: {b}
weight: {c}")
def given_check(context,a,b,c):
    context.deliverytype = a
    context.boxtype = b
    context.weight = c



@When("we create truck delivery")
def delivery_creation(context):
```

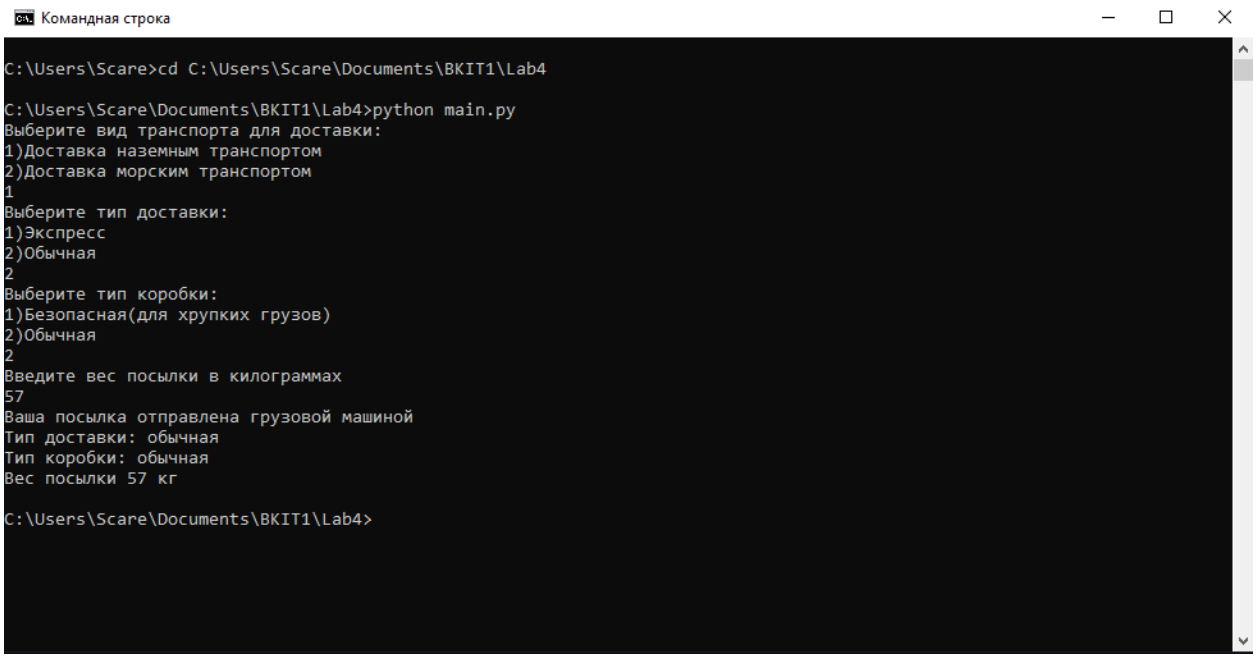
```
    result =
TruckCreator().creation(context.deliverytype,context.boxtype,context.weight)
    context.result=result

@Then("delivery check should be with given parameters")
def compare_results(context):
    assert(context.result == "Ваша посылка отправлена {0}\nТип доставки: {1}\nТип
коробки: {2}\nВес посылки {3} кг".format("грузовой
машиной",context.deliverytype,context.boxtype,context.weight))
```

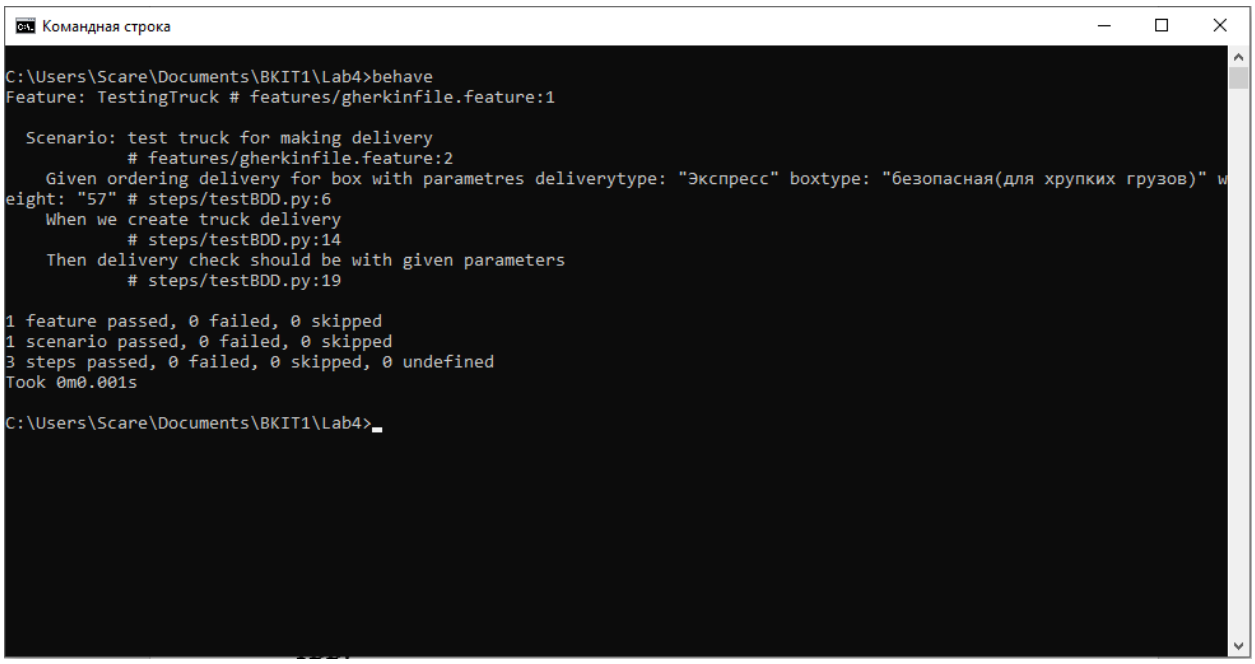
Файл gherkinfile.feature:

```
Feature: TestingTruck
    Scenario: test truck for making delivery
        Given ordering delivery for box with parametres deliverytype: "Экспресс"
boxtype: "безопасная(для хрупких грузов)" weight: "57"
        When we create truck delivery
        Then delivery check should be with given parameters
```

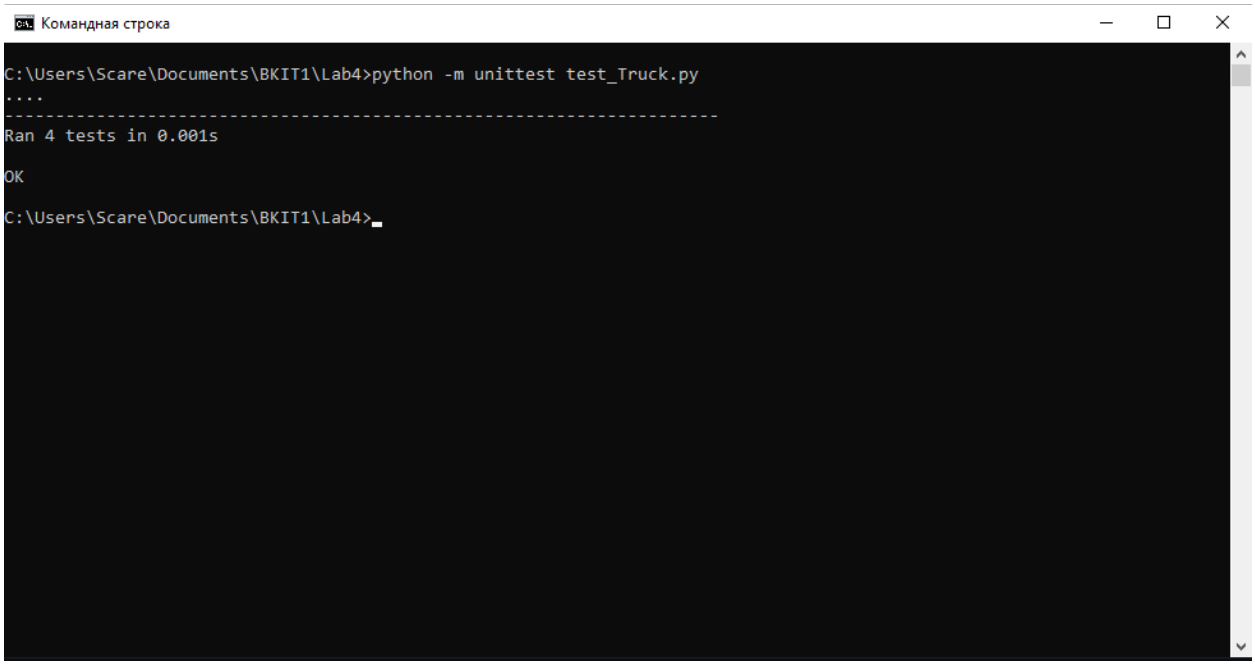
Примеры выполнения программы:



BDD:


Командная строка
C:\Users\Scare\Documents\BKIT1\Lab4>behave
Feature: TestingTruck # features/gherkinfile.feature:1
Scenario: test truck for making delivery
features/gherkinfile.feature:2
Given ordering delivery for box with parametres deliverytype: "Экспресс" boxtype: "безопасная(для хрупких грузов)" w
eight: "57" # steps/testBDD.py:6
When we create truck delivery
steps/testBDD.py:14
Then delivery check should be with given parameters
steps/testBDD.py:19
1 feature passed, 0 failed, 0 skipped
1 scenario passed, 0 failed, 0 skipped
3 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.001s
C:\Users\Scare\Documents\BKIT1\Lab4>

TDD:


Командная строка
C:\Users\Scare\Documents\BKIT1\Lab4>python -m unittest test_Truck.py
....
Ran 4 tests in 0.001s
OK
C:\Users\Scare\Documents\BKIT1\Lab4>