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 «ГУИМЦ»

Кафедра ИУ5 «Системы обработки информации и управления»

Дисциплина «Базовые компоненты ИТ»

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

«Шаблоны проектирования и модульное тестирование в Python»

Студент: Амосов П.А., группа ИУ5Ц-52Б

Преподаватель: Канев А.И.

2022г.

1. Описание задания

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.

2. Текст программы

field.py

```
from __future__ import annotations


def field(items: tuple[dict] | list[dict], *args: str):

    if type(items) not in (list, tuple):
        raise TypeError("Wrong goods format! Try correct types.")

    if not (len(args) > 0):
        raise RuntimeError('No keys specified to print!')

    if tuple(arg for arg in args if arg in items[0].keys()) != args:
        raise KeyError('Selected keys are not in goods list!')

    if len(args) == 1:
        for product in items:
            yield product[args[0]]
    else:
        for product in items:
            yield {arg: product[arg] for arg in args}


if __name__ == "__main__":
```

```
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
    ]

    print("Testing...")

    # Testing
    assert tuple(i for i in field(goods, 'title')) == ('Ковер', 'Диван для
отдыха'), \
        "\n[TestError][#1] -> -Failed-"
    assert tuple(i for i in field(goods, 'title', 'price')) == ({'title':
'Ковер', 'price': 2000},
                                                               {'title':
'Диван для отдыха', 'price': 5300}), \
        "\n[TestError][#2] -> -Failed-"

    print("All tests passed!\n")

    print("Run with args ('title, color'):")
    # Run
    for callback in field(goods, 'title', "color"):
        print(callback)
```

Testing...

All tests passed!

Run with args ('title, color'):

{'title': 'Ковер', 'color': 'green'}

{'title': 'Диван для отдыха', 'color': 'black'}

Process finished with exit code 0

BDD

Tests.py

```
from field import field
from json import loads as load_jstr
from behave import *


@given('a goods list')
def given_a_list(context):
    context.list = load_jstr(context.text)


@given("fields #1 {f1} and #2 {f2}")
def given_a_fields(context, f1: {str}, f2: {str}):
    context.F1 = f1
    context.F2 = f2
```

```
@given("fields #1 {f1}")
def given_a_field(context, f1: {str}):
    context.F1 = f1
    context.F2 = None


@when('we call function with that data we get result')
def func_call(context):
    if context.F2 is not None:
        context.data = tuple(i for i in field(context.list, context.F1,
context.F2))
        return
    context.data = tuple(i for i in field(context.list, context.F1))


@Then("we can assert this data with the tuple")
def check_data(context):
    assert context.data == tuple(load_jstr(context.text))
```

Process finished with exit code 0

Tdd

unittests.py

```
import unittest
from field import field

goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
    ]

class TestEquation(unittest.TestCase):


    def test_calculate(self):
        self.assertEqual(tuple(i for i in field(goods, 'title')), ('Ковер',
'Диван для отдыха'))
        self.assertEqual(tuple(i for i in field(goods, 'title', 'price')),
({'title': 'Ковер', 'price': 2000},

{'title': 'Диван для отдыха',

'price': 5300}))

    def test_type(self):
        with self.assertRaises(TypeError) as e:
            tuple(i for i in field('', 'title'))

    def test_runtime(self):
        with self.assertRaises(RuntimeError) as e:
            tuple(i for i in field(goods))

    def test_key(self):
        with (self.assertRaises(KeyError)) as e:
            tuple(i for i in field(goods, 'title1', 'price'))


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

Testing started at 14:42 ...

Ran 4 tests in 0.003s

OK

Launching unittests with arguments python -m unittest C:/Users/User/Desktop/п/БКИТ/LR5/TDD/unittests.py in C:\Users\User\Desktop\п\БКИТ\LR5\TDD

Process finished with exit code 0