Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Системы обработки информации и управления (ИУ5)

О т ч е т

по лабораторной работе №4

Дисциплина: Разработка Интернет-Приложений

Студент гр. ИУ5-53Б _____ Ваксина И.Р.

(Подпись, дата) (И.О. Фамилия)

Преподаватель _____ Гапанюк Ю.Е.

(Подпись, дата) (И.О. Фамилия)

Москва, 2020

1. Цель работы

Изучение реализации шаблонов проектирования и возможностей модульного тестирования в языке Python.

2. Задание

1 Необходимо для произвольной предметной области реализовать три шаблона проектирования: один порождающий, один структурный и один поведенческий. В качестве справочника шаблонов можно использовать следующий каталог.

2 Для каждой реализации шаблона необходимо написать модульный тест. В модульных тестах необходимо применить следующие технологии:

a. TDD - фреймворк.

b. BDD - фреймворк.

c. Создание Mock-объектов.

3. Decorator

3.1. decorator.py

3.1.1. Текст программы

```
class Notification:
    def __init__(self, notification, email):
        self.notification = notification
        self.email = email

    def send_notification(self):
        print('Уведомление: ' + self.notification)
        print('Отправлена на почту: ' + self.email)
        return 'Отправлена на почту: ' + self.email


class SendToWhatsUp:
    def __init__(self, notifaer, whatsup_nickname):
        self.notifaer = notifaer
        self.whatsup_nickname = whatsup_nickname

    def send_notification(self):
        self.notifaer.send_notification()
        print('Отправлена на WhatsUp: ' + self.whatsup_nickname)
        return 'Отправлена на WhatsUp: ' + self.whatsup_nickname


class SendToTelegram:
    def __init__(self, notifaer, telegram_nickname):
```

```
        self.notifaer = notifaer
        self.telegram_nickname = telegram_nickname

    def send_notification(self):
        self.notifaer.send_notification()
        print('Отправлена на Telegram: ' + self.telegram_nickname)
        return 'Отправлена на Telegram: ' + self.telegram_nickname


class SendToVK:
    def __init__(self, notifaer, vk_nickname):
        self.notifaer = notifaer
        self.Vk_nickname = vk_nickname

    def send_notification(self):
        self.notifaer.send_notification()
        print('Отправлена в Vk: ' + self.Vk_nickname)
        return 'Отправлена в Vk: ' + self.Vk_nickname


class SendToSlack:
    def __init__(self, notifaer, slack_nickname):
        self.notifaer = notifaer
        self.slack_nickname = slack_nickname

    def send_notification(self):
        self.notifaer.send_notification()
        print('Отправлена на Slack: ' + self.slack_nickname)
        return 'Отправлена на Slack: ' + self.slack_nickname


class Client:
    def __init__(self, email, notifications_platforms, nickname=''):
        self.email = email
        self.notifications_platforms = notifications_platforms
        self.nickname = nickname


def create_notifaers(clients):
    notifaers = list()

    i = 1
    for client in clients:
        notifaer = Notification('message' + str(i), client.email)

        if 'Slack' in client.notifications_platforms:
            notifaer = SendToSlack(notifaer, client.nickname)

        if 'VK' in client.notifications_platforms:
            notifaer = SendToVK(notifaer, client.nickname)
```

```
        if 'WhatsUp' in client.notifications_platforms:
            notifaer = SendToWhatsUp(notifaer, client.nickname)

        if 'Telegram' in client.notifications_platforms:
            notifaer = SendToTelegram(notifaer, client.nickname)

        notifaers.append(notifaer)
        i += 1
    return notifaers


def notifay_all(notifaers):
    for notifaer in notifaers:
        notifaer.send_notification()
        print()
    # [lambda i: i.send_notification() for i in notifaers]


if __name__ == '__main__':
    client1 = Client('434f@dsf', ['Slack', 'Telegram'], 'client1')
    client2 = Client('fdsafd@dsf', ['WhatsUp', 'VK'], 'client2')
    client3 = Client('fdsjreio3432i@dsf', ['WhatsUp', 'VK', 'Slack'], 'client3')

    clients = list()
    clients.append(client1)
    clients.append(client2)
    clients.append(client3)
    notifay_all(create_notifaers(clients))
```

3.1.2. Результат выполнения

```
Отправлена на Telegram: client1

Уведомление: message2
Отправлена на почту: fdsafd@dsf
Отправлена в Vk: client2
Отправлена на WhatsUp: client2

Уведомление: message3
Отправлена на почту: fdsjreio3432i@dsf
Отправлена на Slack: client3
Отправлена в Vk: client3
Отправлена на WhatsUp: client3
```

3.2. test_decorator.py

3.2.1. Текст программы

```
from decorator import *
```

```
import unittest


class DecoratorsTests(unittest.TestCase):
    def test_send_to_email(self):

        for i in range(0, 10):
            with self.subTest(i=i):
                notification = Notification('test' + str(i), 'email' + str(i))
                self.assertEqual('Отправлена на почту: email' + str(i),
                                 notification.send_notification())

    def test_send_to_whatsup(self):

        for i in range(0, 10):
            with self.subTest(i=i):
                notification = Notification('test' + str(i), 'email' + str(i))
                notification_whats_up = SendToWhatsUp(notification, 'nickname' + str(
i))
                self.assertEqual('Отправлена на WhatsUp: nickname' + str(i),
                                 notification_whats_up.send_notification())

    def test_send_to_slack(self):

        for i in range(0, 10):
            with self.subTest(i=i):
                notification = Notification('test' + str(i), 'email' + str(i))
                notification_whats_up = SendToWhatsUp(notification, 'nickname' + str(
i))
                notification_slack = SendToSlack(notification_whats_up, 'nickname' +
str(i))
                self.assertEqual('Отправлена на Slack: nickname' + str(i),
                                 notification_slack.send_notification())


    def test_notify_all_users(self):
        client1 = Client('434f@dsf', ['Slack', 'Telegram'], 'client1')
        client2 = Client('fdsafd@dsf', ['WhatsUp', 'VK'], 'client2')
        client3 = Client('fdsjreio3432i@dsf', ['WhatsUp', 'VK', 'Slack'], 'client3')

        clients = list()
        clients.append(client1)
        clients.append(client2)
        clients.append(client3)
        notifay_all(create_notifaers(clients))


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

3.2.2. Результат выполнения

```
PS C:\Users\Ия\OneDrive\Рабочий стол\lab4> python test_decorator.
py
Уведомление: message1
Отправлена на почту: 434f@dsf
Отправлена на Slack: client1
Отправлена на Telegram: client1

Уведомление: message2
Отправлена на почту: fdsafd@dsf
Отправлена в Vk: client2
Отправлена на WhatsUp: client2

Уведомление: message3
Отправлена на почту: fdsjreio3432i@dsf
Отправлена на Slack: client3
Отправлена в Vk: client3
Отправлена на WhatsUp: client3

.Уведомление: test0
Отправлена на почту: email0
Уведомление: test1
Отправлена на почту: email1
Уведомление: test2
Отправлена на почту: email2
Уведомление: test3
Отправлена на почту: email3
Уведомление: test4
Отправлена на почту: email4
Уведомление: test5
Отправлена на почту: email5
Уведомление: test6
Отправлена на почту: email6
Уведомление: test7
Отправлена на почту: email7
Уведомление: test8
```

```
.
----------------------------------------------------------------------
-----
Ran 4 tests in 0.189s

OK
```

4. Singleton

4.1. singleton.py

4.1.1. Текст программы

```
class MetaSingleton(type):
    _instances = {}

    def __call__(cls, *args, **kwargs):
        if cls not in cls._instances:
            instance = super().__call__(*args, **kwargs)
            cls._instances[cls] = instance
        return cls._instances[cls]


class DbConnection:
    DB = list()

    def select(self, request):
        print('Произведен такой запрос к базе данных:' + request)
        print('Получены какие-то данные...')
        print(self.DB)
        return self.DB

    def insert(self, request):
        self.DB.append(request)
        print('Произведен такой запрос к базе данных:' + request)
        print('Данные:' + ' В BD вставлены такие-то данные...')


class Database(metaclass=MetaSingleton):
    connection = None

    def connect(self):
        if self.connection is None:
            self.connection = DbConnection()
            print('Соединение BD установленно')
            return True
        else:
            print('Соединение с BD уже установленно')
            return False

    def select(self, request):
        if self.connection is not None:
            return self.connection.select(request)
        else:
            print('Соединение с базой данных отсутствует')
            return False

    def insert(self, request):
        if self.connection is not None:
            self.connection.insert(request)
            return True
```

```
        else:
            print('Соединение с базой данных отсутствует')
            return False


if __name__ == '__main__':
    db_singleton_1 = Database()
    print(db_singleton_1)
    db_singleton_2 = Database()
    print(db_singleton_2)

    db_singleton_1.insert('INSERT INTO table1 2')
    db_singleton_2.connect()
    db_singleton_1.insert('INSERT INTO table1 2')
```

4.1.2. Результат выполнения

```
PS C:\Users\Ия\OneDrive\Рабочий стол\lab4> python singtоn.py
<__main__.Database object at 0x012471D8>
<__main__.Database object at 0x012471D8>
Соединение с базой данных отсутствует
Соединение BD установленно
Произведен такой запрос к базе данных:INSERT INTO table1 2
Данные: В BD вставлены такие-то данные...
```

4.2. test_singleton.py

4.2.1. Текст программы

```
from singtоn import *
import unittest


class SingletonTests(unittest.TestCase):
    def test_is_single_object(self):
        db_singleton_1 = Database()
        db_singleton_2 = Database()

        self.assertEqual(db_singleton_1, db_singleton_2)
        self.assertNotEqual(db_singleton_1, 2)

    def test_connection(self):
        db_singleton_1 = Database()

        self.assertTrue(db_singleton_1.connect())
        self.assertFalse(db_singleton_1.connect())

    def test_insert_select(self):
        db_singleton = Database()
        self.assertTrue(db_singleton.insert('INSERT INTO table1 2'))

        result = False
```

```
        if 'INSERT INTO table1 2' in db_singleton.connection.DB:
            result = True
        self.assertTrue(result)

    def test_select(self):
        requsts = ['INSERT INTO table1 2', 'requst2', 'requst3', 'requst4']
        db_singleton = Database()
        db_singleton.insert('requst2')
        db_singleton.insert('requst3')
        db_singleton.insert('requst4')

        self.assertEqual(requsts, db_singleton.select('requst'))


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

4.2.2. Результат выполнения

```
PS C:\Users\Ия\OneDrive\Рабочий стол\lab4> python test_singtоn.py

Соединение BD установленно
Соединение с BD уже установленно
.Произведен такой запрос к базе данных:INSERT INTO table1 2
Данные: В BD вставлены такие-то данные...
..Произведен такой запрос к базе данных:requst2
Данные: В BD вставлены такие-то данные...
Произведен такой запрос к базе данных:requst3
Данные: В BD вставлены такие-то данные...
Произведен такой запрос к базе данных:requst4
Данные: В BD вставлены такие-то данные...
Произведен такой запрос к базе данных:requst
Получены какие-то данные...
['INSERT INTO table1 2', 'requst2', 'requst3', 'requst4']
.
----------------------------------------------------------------------
Ran 4 tests in 0.019s

OK
```

5. Visitor

5.1. visitor.py

5.1.1. Текст программы

```
from time import sleep
from threading import *
from termcolor import colored
```

```python
# Поведенческий шаблон Наблюдатель


class InformationBoard:
    def __init__(self):
        self.objects = list()

    def add(self, new_obj):
        self.objects.append(new_obj)

    def start_watching(self):
        i = 0
        while True:
            if i >= len(self.objects):
                i = 0
            thread = ThreadForWindow(self.objects[i])
            if self.objects[i].is_ready():
                print(colored('Окно номер ' + str(self.objects[i].number()) + ' свобо
дно', self.objects[i].color))
                thread.start()
            # sleep(2)
            i += 1


class Window:
    def __init__(self, number, time_for_sleeping, color='white'):
        self.number_ = number
        self.is_ready_ = True
        self.time_for_sleeping_ = time_for_sleeping
        self.color = color

    def number(self):
        return self.number_

    def start_working(self):
        self.is_ready_ = False
        sleep(self.time_for_sleeping_)
        self.is_ready_ = True

    def is_ready(self):
        return self.is_ready_


class ThreadForWindow(Thread):
    def __init__(self, window):
        Thread.__init__(self)
        self.window = window

    def run(self):
        self.window.start_working()
```

```
if __name__ == '__main__':
    inform_board = InformationBoard()
    window1 = Window(1, 3, 'red')
    window2 = Window(2, 4, 'white')
    window3 = Window(3, 5, 'yellow')
    window4 = Window(4, 10, 'blue')

    inform_board.add(window1)
    inform_board.add(window2)
    inform_board.add(window3)
    inform_board.add(window4)

    inform_board.start_watching()
```

5.1.2. Результат выполнения

```
PS C:\Users\Ия\OneDrive\Рабочий стол\lab4> python visitor.py
Окно номер 1 свободно
Окно номер 2 свободно
Окно номер 3 свободно
Окно номер 4 свободно
Окно номер 1 свободно
Окно номер 2 свободно
Окно номер 3 свободно
Окно номер 1 свободно
Окно номер 2 свободно
Окно номер 1 свободно
Окно номер 3 свободно
Окно номер 4 свободно
Окно номер 2 свободно
Окно номер 1 свободно
```

5.2. test_visitor.py

5.2.1. Текст программы

```
from visitor import *
import unittest
from time import sleep


class VisitorTests(unittest.TestCase):
    def test_working(self):
        window1 = Window(1, 2, 'red')
        thread = ThreadForWindow(window1)
        thread.start()
        self.assertFalse(window1.is_ready())
        sleep(7)
        self.assertTrue(window1.is_ready())
```

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

5.2.2. Результат выполнения

```
PS C:\Users\Ия\OneDrive\Рабочий стол\lab4> python test_visitor.py

.
----------------------------------------------------------------------
Ran 1 test in 7.006s

OK
```