Московский государственный технический университет им. Н.Э. Баумана

Факультет Радиотехнический
Кафедра “Системы обработки информации и управления”

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №4
“Шаблоны проектирования и модульное тестирование в Python.”

Выполнил:
студент группы РТ5-31Б:
Каландаров Алим
Шамильевич
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий
Евгеньевич
Подпись и дата:

Москва, 2025 г.

Постановка задачи

Необходимо для произвольной предметной области реализовать от одного до трех шаблонов проектирования: один порождающий, один структурный и один поведенческий.

Я решил реализовать реализовывать шаблоны по предметной области отправки сообщений

В модульных тестах необходимо применить следующие технологии:

- TDD - фреймворк.
- BDD - фреймворк.
- Создание Mock-объектов.

Текст программы

Adapter.py

```
from notifications import Notification
from third_party import ThirdPartyMessenger


class ThirdPartyAdapter(Notification):
    def __init__(self, messenger: ThirdPartyMessenger):
        self._messenger = messenger

    def send(self, message: str) -> bool:
        status = self._messenger.push(message)
        return status == 200
```

notifications.py

```
from abc import ABC, abstractmethod


class Notification(ABC):
    @abstractmethod
    def send(self, message: str) -> bool:
        pass


class EmailNotification(Notification):
    def send(self, message: str) -> bool:
        # имитация отправки email
        print(f"Email sent: {message}")
        return True


class SMSNotification(Notification):
    def send(self, message: str) -> bool:
```

```
        # имитация отправки sms
        print(f"SMS sent: {message}")
        return True


class NotificationFactory:
    @staticmethod
    def create(notification_type: str) -> Notification:
        if notification_type == "email":
            return EmailNotification()
        if notification_type == "sms":
            return SMSNotification()
        raise ValueError("Unknown notification type")
```

sender.py

```
from strategy import SendStrategy


class NotificationSender:
    def __init__(self, strategy: SendStrategy):
        self._strategy = strategy

    def set_strategy(self, strategy: SendStrategy):
        self._strategy = strategy

    def send(self, message: str):
        self._strategy.execute(message)
```

strategy.py

```
from abc import ABC, abstractmethod


class SendStrategy(ABC):
    @abstractmethod
    def execute(self, message: str) -> None:
        pass


class ImmediateStrategy(SendStrategy):
    def execute(self, message: str) -> None:
        print(f"Immediate send: {message}")


class DelayedStrategy(SendStrategy):
    def execute(self, message: str) -> None:
        print(f"Delayed send: {message}")
```

test_adapter.py

```
from adapter import ThirdPartyAdapter
from third_party import ThirdPartyMessenger


def test_third_party_adapter_send():
    messenger = ThirdPartyMessenger()
    adapter = ThirdPartyAdapter(messenger)

    assert adapter.send("Hello Adapter") is True
```

test_notifications.bdd

```
import pytest
from pytest_bdd import scenario, given, when, then
from notifications import NotificationFactory


@scenario(
    "features/notification.feature",
    "Send email notification"
)
def test_send_email():
    pass


@pytest.fixture
def context():
    return {}


@given("a notification factory")
def given_factory(context):
    context["factory"] = NotificationFactory()


@when('I create an "email" notification')
def when_create_notification(context):
    factory = context["factory"]
    context["notification"] = factory.create("email")


@then("the notification should be sent successfully")
def then_send_notification(context):
    notification = context["notification"]
    assert notification.send("Hello BDD") is True
```

test_notifications_mock.py

```
from unittest.mock import Mock
from notifications import Notification


def test_notification_send_called():
    mock_notification = Mock(spec=Notification)

    mock_notification.send.return_value = True

    result = mock_notification.send("Hello")

    mock_notification.send.assert_called_once_with("Hello")
    assert result is True
```

test_notifications_tdd.py

```
import pytest
from notifications import NotificationFactory, EmailNotification, SMSNotification


def test_factory_creates_email_notification():
    notification = NotificationFactory.create("email")
    assert isinstance(notification, EmailNotification)


def test_factory_creates_sms_notification():
    notification = NotificationFactory.create("sms")
    assert isinstance(notification, SMSNotification)


def test_factory_raises_error_for_unknown_type():
    with pytest.raises(ValueError):
        NotificationFactory.create("push")
```

Test_strategy.py

```
from unittest.mock import Mock
from sender import NotificationSender
from strategy import SendStrategy


def test_strategy_execution_called():
    mock_strategy = Mock(spec=SendStrategy)
    sender = NotificationSender(mock_strategy)
```

```
    sender.send("Hello Strategy")

    mock_strategy.execute.assert_called_once_with("Hello Strategy")
```

Third_party.py

```
class ThirdPartyMessenger:
    def push(self, text: str) -> int:
        print(f"Third-party push: {text}")
        return 200
```

notifications_feature.txt

```
==========================================
Feature: Notification sending
  Scenario: Send email notification
   Given a notification factory
When I create an "email" notification
    Then the notification should be sent successfully
==================================
```

Анализ результатов

Bdd тест

```
C:\Users\Mashiro\Projects\Labs\3_sem\PIKYAP\Lab4>pytest test_notifications_bdd.py
============================================== test session starts ==============
platform win32 -- Python 3.12.6, pytest-9.0.2, pluggy-1.6.0
rootdir: C:\Users\Mashiro\Projects\Labs\3_sem\PIKYAP\Lab4
plugins: bdd-8.1.0
collected 1 item

test_notifications_bdd.py .

============================================== 1 passed in 0.05s ===============

C:\Users\Mashiro\Projects\Labs\3_sem\PIKYAP\Lab4>
```

Tdd тест

```
C:\Users\Mashiro\Projects\Labs\3_sem\PIKYAP\Lab4>pytest test_notifications_tdd.py
============================================== test session starts ==============
platform win32 -- Python 3.12.6, pytest-9.0.2, pluggy-1.6.0
rootdir: C:\Users\Mashiro\Projects\Labs\3_sem\PIKYAP\Lab4
plugins: bdd-8.1.0
collected 3 items

test_notifications_tdd.py ...

============================================== 3 passed in 0.05s ===============

C:\Users\Mashiro\Projects\Labs\3_sem\PIKYAP\Lab4>
```

Mock объект тест