Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №5-6
«Шаблоны проектирования и модульное тестирование в Python»

Выполнил:
студент группы ИУ5-25Б
Хрипков Т.А.
Подпись и дата:

Проверил:
Подпись и дата:

Москва, 2024 г

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. В модульных тестах необходимо применить следующие технологии:
    - TDD - фреймворк.
    - BDD - фреймворк.
    - Создание Mock-объектов.

Текст программы:

Classes.py

```
from abc import ABC, abstractmethod
import sys


class Book(ABC):
    @abstractmethod
    def get_description(self):
        pass

class Fiction(Book):
    def get_description(self):
        return "This is a fiction book."

class NonFiction(Book):
    def get_description(self):
        return "This is a non-fiction book."

class BookFactory(ABC):
    @abstractmethod
    def create_book(self) -> Book:
        pass

class FictionBookFactory(BookFactory):
    def create_book(self) -> Book:
        return Fiction()

class NonFictionBookFactory(BookFactory):
    def create_book(self) -> Book:
        return NonFiction()

print("////////////////////////////////////////////////////////////////////
//////////////////")

class BookDecorator(Book):
    def __init__(self, book: Book):
```

```
        self._book = book

    def get_description(self):
        return self._book.get_description()

class IllustratedBookDecorator(BookDecorator):
    def get_description(self):
        return f"{self._book.get_description()} It has illustrations."

print("////////////////////////////////////////////////////////////////////
//////////////////")


class Observer(ABC):
    @abstractmethod
    def update(self, book: Book):
        pass


class BookStore(Observer):
    def update(self, book: Book):
        print(f"New book added: {book.get_description()}")


class Librarian:
    def __init__(self):
        self._observers = []

    def subscribe(self, observer: Observer):
        self._observers.append(observer)

    def unsubscribe(self, observer: Observer):
        self._observers.remove(observer)

    def notify(self, book: Book):
        for observer in self._observers:
            observer.update(book)
```

Mock.py

```
from classes import Fiction, NonFiction, FictionBookFactory,
NonFictionBookFactory
from classes import IllustratedBookDecorator, BookStore, Librarian
import unittest
from unittest.mock import Mock

class TestBookFactory(unittest.TestCase):
    def test_create_fiction_book(self):
        factory = FictionBookFactory()
        book = factory.create_book()
        self.assertEqual(book.get_description(), "This is a fiction book.")
```

```
class TestBookDecorator(unittest.TestCase):
    def test_illustrated_book_decorator(self):
        factory = FictionBookFactory()
        book = factory.create_book()
        illustrated_book = IllustratedBookDecorator(book)
        self.assertEqual(illustrated_book.get_description(), "This is a
fiction book. It has illustrations.")

class TestObserver(unittest.TestCase):
    def test_observer_notification(self):
        librarian = Librarian()
        mock_store = Mock(spec=BookStore)
        librarian.subscribe(mock_store)

        factory = FictionBookFactory()
        new_book = factory.create_book()
        librarian.notify(new_book)

        mock_store.update.assert_called_once_with(new_book)

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

TDD.py

```
from classes import Fiction, NonFiction, FictionBookFactory,
NonFictionBookFactory
from classes import IllustratedBookDecorator, BookStore, Librarian
import pytest
from io import StringIO
import sys

# Фабрика
def test_fiction_book_factory():
    factory = FictionBookFactory()
    book = factory.create_book()
    assert isinstance(book, Fiction)
    assert book.get_description() == "This is a fiction book."
    print("test_fiction_book_factory passed")


def test_non_fiction_book_factory():
    factory = NonFictionBookFactory()
    book = factory.create_book()
    assert isinstance(book, NonFiction)
    assert book.get_description() == "This is a non-fiction book."
    print("test_non_fiction_book_factory passed")


# Декоратор
```

```
def test_illustrated_book_decorator():
    book = Fiction()
    decorated_book = IllustratedBookDecorator(book)
    assert decorated_book.get_description() == "This is a fiction book. It
has illustrations."
    print("test_illustrated_book_decorator passed")


#  Наблюдатель
def test_book_store_observer():
    librarian = Librarian()
    book_store = BookStore()

    librarian.subscribe(book_store)

    book = Fiction()

    output = StringIO()
    sys.stdout = output

    librarian.notify(book)

    sys.stdout = sys.__stdout__

    assert "New book added: This is a fiction book." in output.getvalue()
    print("test_book_store_observer passed")


# Проверка отписки
def test_unsubscribe_observer():
    librarian = Librarian()
    book_store = BookStore()

    librarian.subscribe(book_store)
    librarian.unsubscribe(book_store)

    output = StringIO()
    sys.stdout = output

    librarian.notify(Fiction())
    sys.stdout = sys.__stdout__

    assert output.getvalue() == ""
    print("test_unsubscribe_observer passed")

test_fiction_book_factory()
test_non_fiction_book_factory()
test_illustrated_book_decorator()
test_book_store_observer()
test_unsubscribe_observer()
```

Пример работы:

TDD

```
(venv) PS C:\Users\tima\Piton\lab5-6\Tests> python TDD.py
////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////
test_fiction_book_factory passed
test_non_fiction_book_factory passed
test_illustrated_book_decorator passed
test_book_store_observer passed
test_unsubscribe_observer passed
(venv) PS C:\Users\tima\Piton\lab5-6\Tests>
```

Mock

```
(venv) PS C:\Users\tima\Piton\lab5-6\Tests> python Mock.py
////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////
...
----------------------------------------------------------------------
Ran 3 tests in 0.002s

OK
(venv) PS C:\Users\tima\Piton\lab5-6\Tests>
```