Отчёт

По лр-5

Дисциплина «Парадигмы и конструкции языков программирования»

Студент: Якубович Анна

Группа: ИБМЗ-23Б

Преподаватель: Гапанюк Ю.Е.

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:
    1. TDD - фреймворк.
    2. BDD - фреймворк.
    3. Создание Mock-объектов.

Код программы:

```
from abc import ABC, abstractmethod

from unittest.mock import Mock

import unittest



# Фабричный метод (Порождающий)

class Button(ABC):

    @abstractmethod

    def render(self): pass

class PrimaryButton(Button):

    def render(self):

        return "<button class='btn-primary'>Click me</button>"

class SecondaryButton(Button):

    def render(self):

        return "<button class='btn-secondary'>Click me</button>"

class ButtonFactory:

    @staticmethod

    def create_button(button_type):

        if button_type == "primary":

            return PrimaryButton()

        elif button_type == "secondary":
```

```
            return SecondaryButton()
        else:
            raise ValueError("Unknown button type")
# Декоратор (Структурный)
class ButtonDecorator(Button):
    def __init__(self, button):
        self._button = button

    def render(self):
        return self._button.render()
class DisabledDecorator(ButtonDecorator):
    def render(self):
        return self._button.render().replace('>', ' disabled>')
class LargeDecorator(ButtonDecorator):
    def render(self):
        return self._button.render().replace('btn-', 'btn-lg ')
# Наблюдатель (Поведенческий)
class ButtonObserver(ABC):
    @abstractmethod
    def on_click(self, button_type): pass
class ClickLogger(ButtonObserver):
    def on_click(self, button_type):
        print(f"Button {button_type} was clicked!")
class AnalyticsTracker(ButtonObserver):
    def on_click(self, button_type):
        print(f"Analytics: {button_type} button click tracked")
class ClickableButton:
    def __init__(self, button):
        self._button = button
        self._observers = []

    def add_observer(self, observer):
        self._observers.append(observer)

    def click(self):
```

```
        html = self._button.render()
        for observer in self._observers:
            observer.on_click(self._button.__class__.__name__)
        return html
# TDD тесты
class TestButtonSystem(unittest.TestCase):
    def test_factory_creates_correct_buttons(self):
        primary = ButtonFactory.create_button("primary")
        secondary = ButtonFactory.create_button("secondary")

        self.assertIsInstance(primary, PrimaryButton)
        self.assertIsInstance(secondary, SecondaryButton)
        self.assertIn("btn-primary", primary.render())
        self.assertIn("btn-secondary", secondary.render())

    def test_decorator_adds_features(self):
        button = PrimaryButton()
        disabled_button = DisabledDecorator(button)

        self.assertIn("disabled", disabled_button.render())

    def test_observer_notified_on_click(self):
        button = PrimaryButton()
        clickable = ClickableButton(button)
        mock_observer = Mock(spec=ButtonObserver)

        clickable.add_observer(mock_observer)
        clickable.click()

        mock_observer.on_click.assert_called_once()
# Mock тесты
class TestButtonSystemWithMocks(unittest.TestCase):
    def test_mock_button_factory(self):
        mock_factory = Mock(spec=ButtonFactory)
        mock_button = Mock(spec=Button)
```

```
        mock_factory.create_button.return_value = mock_button


        result = mock_factory.create_button("primary")
        mock_factory.create_button.assert_called_with("primary")
        self.assertEqual(result, mock_button)


    def test_mock_observer(self):
        mock_observer = Mock(spec=ButtonObserver)
        button = PrimaryButton()
        clickable = ClickableButton(button)


        clickable.add_observer(mock_observer)
        clickable.click()


        mock_observer.on_click.assert_called_once()
```

```
# BDD тесты
def test_bdd_scenarios():
    print("\n--- BDD  ---")
    print("Сценарий: Создание стилизованной кнопки")
    button = ButtonFactory.create_button("primary")
    decorated_button = LargeDecorator(DisabledDecorator(button))
    result = decorated_button.render()
    print(f"Результат: {result}")
    assert "btn-lg" in result and "disabled" in result
    print("Успех: Создана большая отключенная основная кнопка")


    print("\nСценарий: Отслеживание кликов на кнопке")
    button = PrimaryButton()
    clickable = ClickableButton(button)
    logger = ClickLogger()
    analytics = AnalyticsTracker()


    clickable.add_observer(logger)
    clickable.add_observer(analytics)
```

```
    print("Кликаем на кнопку...")

    clickable.click()

    print("Успех: Все наблюдатели получили уведомление")

if __name__ == "__main__":

    print("=== Демонстрация паттернов ===")

    primary = ButtonFactory.create_button("primary")

    print("1. Фабрика:", primary.render())

    decorated = LargeDecorator(DisabledDecorator(primary))

    print("2. Декоратор:", decorated.render())

    clickable = ClickableButton(primary)

    clickable.add_observer(ClickLogger())

    print("3. Наблюдатель:")

    clickable.click()



    print("\n=== Запуск тестов ===")

    unittest.main(exit=False)

    test_bdd_scenarios()
```

Пример результата работы программы:

```
1. Фабрика: <button class='btn-primary'>Click me</button>
2. Декоратор: <button class='btn-lg primary' disabled>Click me</button disabled>
3. Наблюдатель:
Button PrimaryButton was clicked!

=== Запуск тестов ===
.....
----------------------------------------------------------------------
Ran 5 tests in 0.003s

OK

--- BDD  ---
Сценарий: Создание стилизованной кнопки
Результат: <button class='btn-lg primary' disabled>Click me</button disabled>
Успех: Создана большая отключенная основная кнопка

Сценарий: Отслеживание кликов на кнопке
Кликаем на кнопку...
Button PrimaryButton was clicked!
Analytics: PrimaryButton button click tracked
Успех: Все наблюдатели получили уведомление
```