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 №4

Выполнила:
Студентка группы ИУ5-34Б
Глозман Варвара

Проверил:
Гапанюк Ю. Е.

2025 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.

Листинг программы:

__init__.py:

```
from .beverages import (
    Beverage,
    Coffee,
    Tea,
    BeverageComponent,
    SimpleBeverage,
    BeverageDecorator,
    MilkDecorator,
    SyrupDecorator,
)
from .factory import BeverageFactory, SimpleBeverageFactory

__all__ = [
    "Beverage",
    "Coffee",
    "Tea",
    "BeverageComponent",
    "SimpleBeverage",
    "BeverageDecorator",
    "MilkDecorator",
    "SyrupDecorator",
    "BeverageFactory",
    "SimpleBeverageFactory",
]
```

beverages.py:

```
from abc import ABC, abstractmethod

class Beverage(ABC):

    def prepare(self) -> str:
        steps = [
            self.boil_water(),
            self.brew(),
            self.pour_in_cup(),
```

```
            self.add_condiments(),
        ]
        return " | ".join(step for step in steps if step)

    def boil_water(self) -> str:
        return "Кипятим воду"

    @abstractmethod
    def brew(self) -> str:
        ...

    def pour_in_cup(self) -> str:
        return "Наливаем в чашку"

    @abstractmethod
    def add_condiments(self) -> str:
        ...


class Coffee(Beverage):
    def brew(self) -> str:
        return "Завариваем кофе"

    def add_condiments(self) -> str:
        return "Добавляем сахар и молоко"


class Tea(Beverage):
    def brew(self) -> str:
        return "Завариваем чай"

    def add_condiments(self) -> str:
        return "Добавляем лимон"



class BeverageComponent(ABC):

    @abstractmethod
    def cost(self) -> float:
        ...

    @abstractmethod
    def description(self) -> str:
        ...


class SimpleBeverage(BeverageComponent):

    def __init__(self, beverage: Beverage, base_cost: float, name: str):
        self._beverage = beverage
        self._base_cost = base_cost
        self._name = name
```

```
    def cost(self) -> float:
        return self._base_cost

    def description(self) -> str:
        return self._name


class BeverageDecorator(BeverageComponent):

    def __init__(self, component: BeverageComponent):
        self._component = component

    def cost(self) -> float:
        return self._component.cost()

    def description(self) -> str:
        return self._component.description()


class MilkDecorator(BeverageDecorator):
    def cost(self) -> float:
        return self._component.cost() + 20.0

    def description(self) -> str:
        return self._component.description() + " + молоко"


class SyrupDecorator(BeverageDecorator):
    def cost(self) -> float:
        return self._component.cost() + 15.0

    def description(self) -> str:
        return self._component.description() + " + сироп"
```

factory.py:

```
from abc import ABC, abstractmethod
from .beverages import Coffee, Tea, SimpleBeverage


class BeverageFactory(ABC):

    @abstractmethod
    def create_beverage(self, beverage_type: str):
        ...


class SimpleBeverageFactory(BeverageFactory):

    BASE_PRICES = {
        "coffee": 120.0,
        "tea": 80.0,
    }
```

```
    def create_beverage(self, beverage_type: str):
        beverage_type = beverage_type.lower()
        if beverage_type == "coffee":
            beverage = Coffee()
            return SimpleBeverage(beverage, self.BASE_PRICES["coffee"], "Кофе")
        if beverage_type == "tea":
            beverage = Tea()
            return SimpleBeverage(beverage, self.BASE_PRICES["tea"], "Чай")
        raise ValueError(f"Неизвестный тип напитка: {beverage_type}")
```

test_tdd_beverages.py:

```
import pytest
from coffee.beverages import Coffee, Tea, SimpleBeverage, MilkDecorator, SyrupDecorator


def test_coffee_prepare_template_method():
    coffee = Coffee()
    process = coffee.prepare()

    assert "Кипятим воду" in process
    assert "Завариваем кофе" in process
    assert "Добавляем сахар и молоко" in process


def test_tea_prepare_template_method():
    tea = Tea()
    process = tea.prepare()

    assert "Кипятим воду" in process
    assert "Завариваем чай" in process
    assert "Добавляем лимон" in process


def test_decorator_cost_and_description():
    base = SimpleBeverage(Coffee(), base_cost=120.0, name="Кофе")
    with_milk = MilkDecorator(base)
    with_milk_and_syrup = SyrupDecorator(with_milk)

    assert with_milk_and_syrup.cost() == pytest.approx(155.0)
    assert with_milk_and_syrup.description() == "Кофе + молоко + сироп"
```

test_tdd_factory.py:

```
import pytest
from coffee.factory import SimpleBeverageFactory
from coffee.beverages import SimpleBeverage


def test_factory_creates_coffee():
    factory = SimpleBeverageFactory()
```

```
    beverage = factory.create_beverage("coffee")

    assert isinstance(beverage, SimpleBeverage)
    assert beverage.description() == "Кофе"
    assert beverage.cost() > 0


def test_factory_unknown_type():
    factory = SimpleBeverageFactory()

    with pytest.raises(ValueError):
        factory.create_beverage("unknown")
```

test_mocks.py:

```
def test_factory_uses_base_prices_mock(mocker):
    from coffee.factory import SimpleBeverageFactory

    factory = SimpleBeverageFactory()
    mocker.patch.object(factory, "BASE_PRICES", {"coffee": 1.0, "tea": 1.0})

    beverage = factory.create_beverage("coffee")

    assert beverage.cost() == 1.0
```

test_bdd_order.py:

```
import pytest
from pytest_bdd import given, when, then, scenario, parsers, scenarios
from coffee.factory import SimpleBeverageFactory
from coffee.beverages import MilkDecorator, SyrupDecorator

@scenario("features/order_coffee.feature", "Клиент заказывает кофе с молоком и сиропом")
def test_order_coffee():
    pass

@given("в системе есть фабрика напитков", target_fixture="beverage_factory")
def beverage_factory():
    return SimpleBeverageFactory()

@when(parsers.parse('клиент заказывает "{kind}" с молоком и сиропом'),
target_fixture="order_with_addons")
def order_with_addons(beverage_factory, kind):
    base = beverage_factory.create_beverage(kind)
    with_milk = MilkDecorator(base)
    with_syrup = SyrupDecorator(with_milk)
    return with_syrup

@then(parsers.parse("итоговая стоимость заказа равна {price:f}"))
def check_price(order_with_addons, price):
    assert order_with_addons.cost() == price

@then(parsers.parse('описание заказа равно "{description}"'))
```

```python
def check_description(order_with_addons, description):
    assert order_with_addons.description() == description
```

order_coffee.feature:

```
Feature: Оформление заказа кофе
  Scenario: Клиент заказывает кофе с молоком и сиропом
    Given в системе есть фабрика напитков
    When клиент заказывает "coffee" с молоком и сиропом
    Then итоговая стоимость заказа равна 155.0
    And описание заказа равно "Кофе + молоко + сироп"
```

pyproject.toml:

```
[project]
name = "coffee-shop-patterns"
version = "0.1.0"
requires-python = ">=3.10"

[project.dependencies]
pytest = "^8.0.0"
pytest-bdd = "^7.0.0"
pytest-mock = "^3.14.0"

[tool.pytest.ini_options]
testpaths = ["tests"]
```

Результат выполнения:

```
============================= test session starts ==============================
platform darwin -- Python 3.13.6, pytest-9.0.2, pluggy-1.6.0
rootdir: /Users/nikitakelcin/Varya/coffee_shop
configfile: pyproject.toml
testpaths: tests
plugins: mock-3.15.1, bdd-8.1.0
collected 7 items

tests/bdd/test_bdd_order.py .                                            [ 14%]
tests/bdd/test_mocks.py .                                                [ 28%]
tests/test_tdd_beverages.py ...                                          [ 71%]
tests/test_tdd_factory.py ..                                             [100%]

============================== 7 passed in 0.01s ===============================
```