Мельник Денис Сергеевич БПИ 238

Контрольная работа №1. Паттерны проектирования. Отчёт

Инструкция для запуска тестов находится в README.md. Их покрытие точно больше 65%.

Реализованная функциональность:

1) Основные требования к функциональности модуля: работа с доменной моделью: создание, редактирование и удаление счетов, категорий, операций (доходов/расходов) – реализовано в пунктах меню 1-9, создание, редактирование и удаление соответственно для классов счёта, категории и операции
2) Опциональная функциональность модуля
    a. Аналитика – подсчёт разницы доходов и расходов за выбранный период (пункт меню 10), группировка доходов и расходов по категориям (пункт меню 11).
    b. Импорт и экспорт данных – импорт или экспорт всех данных (массивы основных классов модели) в JSON, CSV, YAML (пункт меню 12)
    c. Управление данными – баланс считается с учётом начального баланса счёта, к нему прибавляются или отнимаются значения операций
    d. Статистика – каждая команда использует «Команда + декоратор», и пишет время своей работы в консоль после использования
3) Три основных класса: BankAccount, Category и Operation реализованы в файлах BankAccount.cs, Category.cs и Operation.cs
4) Паттерны
    a. Фасад – реализован class FinancialFacade в одноимённом файле, в нём происходит хранение и работа с массивами основных классов
    b. Команда + декоратор – команда Command и декоратор TimingDecorator оба реализуют интерфейс ICommand. Команда – все команды из меню, декоратор позволяет выводить в консоль время работы команды
    c. Шаблонный метод – используется при импорте данных (метод ImportData<T>), шаблонный класс – один из основных класов, который импортируется в данный момент. Классы: JsonImporter, CsvImporter, YamlImporter
    d. Посетитель – используется при экспорте данных, для 3 форматов файлов это классы JsonExportVisitor, CsvExportVisitor и YamlExportVisitor
    e. Фабрика – используется при создании операции (Operation), класс OperationFactory. Нужен для валидации параметра amount, чтобы он не был указан как отрицательный
    f. Прокси – используется при работе с файлами
5) Критерии оценки
    a. + 2 балла за полную реализацию основных требований к функциональности – это сделано и расписано выше

b. + 0.5 балла (max 3 балла по данному критерию) за каждый реализованный (из предложенных нами или выбранных вами) паттерн – все паттерны реализованы
c. + 2 балла за соблюдение принципов SOLID и GRASP – принципы соблюдены
d. + 1 балл если покрыто тестами более 65% кода – тесты реализованы в KR_1_MELNIK.Tests и имеют большой процент покрытия
e. + 1 балл за использование DI-контейнера – DI-контейнер используется при создании фабрики OperationFactory в методе CreateOperation в фасаде FinancialFacade
f. +/- 1 балл за очную защиту проекта семинаристу – готов защититься у семинариста

6) Отчёт
   a. Общая идею вашего решения (какой функционал реализовали, особенно если вносили изменения в функциональные требования) – функционал, как в тз, существуют счета с определённым балансом, категории операций (увеличивающие или уменьшающие баланс счёта) и сами операции, принадлежащие к определённой категории
   b. Опишите какие принципы из SOLID вы реализовали, скажите в каких классах (модулях)
      i. Single Responsibility Principle – каждый класс имеет свою обязанность, например классы CsvExporter, JsonExporter, YamlExporter работают только со своим форматом файлов
      ii. Open/Closed Principle – использование интерфейса IExportVisitor позволяет расширять формат экспорта без изменения существующего кода
      iii. Liskov Substitution Principle – JsonExporter, CsvExporter, YamlExporter реализуют IExportVisitor, что позволяет заменять их друг другом
      iv. Interface Segregation Principle – IExportVisitor отделяет экспорт данных от других операций; IValidationService выделяет валидацию в отдельный интерфейс
      v. Dependency Inversion Principle - использование IValidationService вместо жесткой привязки к ValidationService; OperationFactory получает IValidationService через DI
   c. Опишите какие принципы из GRASP вы реализовали, скажите в каких классах (модулях)
      i. Controller – класс Command управляет взаимодействием с пользователем
      ii. Information Expert – BankAccount, Category, Operation содержат логику, относящуюся к их данным
      iii. Polymorphism – IExportVisitor и ICommand позволяют использовать разные реализации без изменения кода

iv. Creator – OperationFactory отвечает за создание объектов Operation
v. Low Coupling - FinancialFacade скрывает сложность системы

d. Опишите какие паттерны GoF вы реализовали, обоснуйте их важность, скажите в каких классах (модулях) они реализованы – описано выше в разделе «шаблоны»