Контрольная работа 1. Мусаев Умахан Рашидович.

Немного о программе:

Общая идея и функционал

В проекте реализовано приложение для учета финансов, включающее работу с банковскими счетами, категориями расходов/доходов и операциями. Добавлены функции создания, редактирования и удаления объектов, а также импорта/экспорта данных в разных форматах (CSV, JSON, YAML). Код разделен на модули: основная логика работы с данными (класс FinancialFacade) и набор вспомогательных классов/меню (MenuHandlers, InputHelpers и т.д.), что упрощает поддержку и расширение функционала.

Принципы SOLID и GRASP

- **Single Responsibility (SRP): класс FinancialFacade отвечает за операции высокого уровня над финансовыми данными, а классы вида Repository занимаются доступом к данным. Это разделение задач делает код более структурированным.**
- **Open-Closed (OCP): при необходимости добавить новую реализацию репозиториев или форматы импорта/экспорта можно наследовать и расширить имеющиеся классы, не изменяя основной код.**
- **Low Coupling / High Cohesion (GRASP): каждый класс сконцентрирован на своей задаче, а взаимодействие между модулями сведено к минимуму за счет четких интерфейсов (IBankAccountRepository, ICategoryRepository, IOperationRepository).**

Паттерны GoF

- **Facade: класс FinancialFacade скрывает сложность работы с несколькими репозиториями, предоставляя упрощенный интерфейс для операций с финансовой логикой.**
- **Proxy: классы вида BankAccountRepositoryProxy выступают «прокси», перехватывая вызовы и при необходимости выполняя дополнительную логику (например, кэширование или логирование).**

Обоснованность паттернов: Facade упрощает взаимодействие между слоями приложения, а Proxy обеспечивает гибкость и контроль при доступе к репозиториям.

Для работы программы нужно скачать NuGet пакет: YamlDotNet

Описание основных классов

1. **FinancialFacade**

Фасад для управления всеми финансовыми объектами (банковские счета, категории, операции). Предоставляет единый интерфейс для создания, обновления и удаления данных, а также содержит логику расчета балансов, получения статистики и прочих операций.

2. **MenuHandlers**

Статический класс, отвечающий за логику взаимодействия с пользователем: вывод меню, обработку выбора пунктов и вызов соответствующих методов фасада для выполнения операций (например, создание счета, редактирование операции и т.д.).

3. InputHelpers

Статический класс с методами для безопасного и удобного ввода данных (чисел, дат, строк, enum-типов). Упрощает чтение пользовательского ввода, уменьшает дублирование кода в меню.

4. BankAccount

Класс, представляющий банковский счет. Имеет свойства идентификатора, названия счета, баланса и методы для обновления имени и изменения баланса.

5. Category

Класс, соответствующий категории операции, например «Доход» или «Расход». Содержит свойства идентификатора, типа категории и названия. Имеет метод для изменения имени категории.

6. Operation

Класс, описывающий финансовую операцию (доход или расход). Содержит идентификатор, дату, сумму, описание, тип операции и связывает ее с банковским счетом и категорией.

7. Репозитории (интерфейсы IBankAccountRepository, ICategoryRepository, IOperationRepository)

Определяют методы для управления данными (создание, обновление, удаление и получение) в зависимости от типа сущности (счета, категории, операции). Позволяют легко менять способ хранения данных (например, база данных, файлы и т.д.).

Реализованные функции:

1. Управление банковскими счетами

- **Создание, редактирование и удаление счетов.**
- **Просмотр списка всех счетов.**

2. Работа с категориями

- **Создание, редактирование и удаление категорий (доходы/расходы).**
- **Просмотр всех доступных категорий.**

3. Операции

- **Создание новых операций с указанием даты, суммы, категории и банковского счета.**
- **Редактирование и удаление существующих операций.**
- **Просмотр всех операций.**

4. Аналитика

- **Подсчет разницы доходов и расходов за указанный период.**
- **Группировка операций по категориям с выводом детальной статистики.**

5. **Импорт и экспорт**

- **Экспорт данных в формате CSV, JSON или YAML.**
- **Импорт данных из указанных форматов.**

6. **Пересчет балансов**

   Пересчет баланса по всем счетам в случае несоответствий.

Все основные функции реализованы.

В папке primer есть файлы с данными в формате json,csv и yaml для примера.