Отчет по разработке модуля "Учет финансов"

1. Общая идея решения

Разработан модуль "Учет финансов", включающий следующие функциональные возможности:

- **Работа с доменной моделью: Создание, редактирование и удаление счетов, категорий и операций (доходов/расходов).**
- **Аналитика: Подсчет разницы доходов и расходов за выбранный период, группировка по категориям.**
- **Импорт и экспорт данных: Поддержка CSV, JSON, YAML.**
- **Управление данными: Пересчет баланса при несоответствиях.**
- **Статистика: Измерение времени выполнения сценариев.**

Дополнительно:

- **Улучшен процесс создания категорий: Теперь при создании отображается ID категории.**
- **Оптимизирован экспорт данных в CSV: Учитывается правильное форматирование и экранирование символов.**

2. Принципы SOLID и GRASP

SOLID:

1. **Single Responsibility Principle (SRP):**
   - **Каждый класс выполняет единственную задачу (например, `CsvDataExporter` отвечает только за экспорт CSV).**
2. **Open/Closed Principle (OCP):**
   - **Возможность расширять систему новыми типами экспорта/импорта, не изменяя существующий код (`IDataExportVisitor` и `IDataImportVisitor`).**
3. **Liskov Substitution Principle (LSP):**
   - **Экспорт и импорт поддерживают полиморфизм, что позволяет использовать их взаимозаменяемо.**
4. **Interface Segregation Principle (ISP):**
   - **Разделение интерфейсов для экспорта и импорта данных (`IDataExportVisitor`, `IDataImportVisitor`).**
5. **Dependency Inversion Principle (DIP):**
   - **Использование `FinancialFacade` для взаимодействия с доменной моделью вместо работы с низкоуровневыми классами напрямую.**

GRASP:

1. **Controller** – FinancialFacade управляет операциями, не перегружая другие классы.
2. **Creator** – фабрика FinancialEntityFactory отвечает за создание объектов (Category, Operation).
3. **Polymorphism** – различные классы импорта/экспорта реализуют общий интерфейс.

3. Паттерны GoF

1. **Factory Method** (FinancialEntityFactory)
   - Используется для создания экземпляров Category, Operation, BankAccount, уменьшая зависимость от конкретных реализаций.
2. **Strategy** (IDataExportVisitor, IDataImportVisitor)
   - Позволяет динамически выбирать стратегию экспорта/импорта данных.
3. **Template Method** (DataImporter)
   - Определяет шаблон процесса импорта, позволяя подклассам (CsvDataImporter, JsonDataImporter, YamlDataImporter) переопределять детали парсинга.

4. Инструкция по запуску

Требования:

- .NET 7.0+
- Newtonsoft.Json, YamlDotNet (установлены через NuGet)

Запуск:

1. Склонируйте репозиторий и перейдите в папку проекта:
2. Соберите проект:
3. Запустите приложение:
4. Следуйте инструкциям в консоли для работы с системой учета финансов.