**Теоретична частина до проєкту "Bank Account Refactoring"**

1. Мета проєкту

Метою даного проєкту є створення програмного продукту банківської системи, що дозволяє здійснювати базові фінансові операції (поповнення, зняття коштів, перегляд балансу, історії транзакцій) з використанням чистого, рефакторингованого та модульного коду. Особливий акцент зроблено на впровадженні принципів рефакторингу, покращенні архітектури коду, впровадженні шаблонів проєктування та покритті тестами.

2. Застосовані техніки рефакторингу

- Extract Method — виділення частин логіки в окремі методи  
- Extract Class — винесення функціональності в окремі класи  
- Extract Module — створення окремих файлів для логіки (наприклад, services/)  
- Replace Magic Numbers — заміна чисел на константи  
- Move Method — перенесення методів у відповідні класи  
- Introduce Enum — створення enums для покращення читабельності  
- Factory Pattern — фабрика для створення акаунтів  
- Decorator Pattern — для логування дій  
- Command Pattern — для команд меню  
- Clean Architecture — поділ на шари: логіка / UI / сервіси / збереження

3. Як змінювався код

Початковий код мав усю логіку в одному файлі, без поділу на класи, функції та структуру. Після кожного кроку від ref1\_code.py до ref10\_code.py код поступово розбивався на модулі, функції, класи, прибирались повторення, створювались тести, підключались патерни, додавалась памʼять. Фінально — створено GUI застосунок.

4. Додаткові реалізації

- Памʼять акаунту через JSON (збереження балансу та історії)  
- Вивід у лог-файл усіх дій користувача  
- Перемикання теми інтерфейсу (темна/світла) в GUI  
- Заокруглені кнопки в інтерфейсі  
- Можливість розтягувати вікно GUI

5. Команди для запуску

- Запуск CLI: python main.py  
- Запуск GUI: python GUI\_app.py  
- Запуск тестів: python -m unittest discover -s tests

6. Додаткові пояснення

- \_\_init\_\_.py — потрібен для позначення директорій як Python-пакетів  
- PYTHONPATH=. — дозволяє імпортувати модулі з проєкту з будь-якої точки запуску  
- customtkinter — бібліотека для створення сучасного GUI з кастомними елементами  
- unittest — модуль стандартної бібліотеки Python для тестування  
- pylint — інструмент перевірки стилю і якості коду (я отримав 10/10)

7. Висновок

Проєкт повністю відповідає вимогам — 15 технік рефакторингу, 20 модульних тестів, чистий та структурований код, GUI-версія, якісна архітектура. Програму зручно використовувати, вона легко підтримується і чудово підходить для демонстрації принципів якісної розробки.

8. Теоретичні основи рефакторингу

Рефакторинг — це процес зміни внутрішньої структури програми без зміни її зовнішньої поведінки. Метою рефакторингу є покращення читабельності коду, його супровідності, повторного використання та зменшення складності.

Типові ознаки, що вказують на потребу у рефакторингу (так звані code smells):  
- Дублювання коду (Duplicate Code)  
- Занадто довгі методи (Long Method)  
- Класи з багатьма обов’язками (Large Class)  
- Залежність від магічних чисел (Magic Numbers)  
- Тісне зʼєднання (Tight Coupling)

Рефакторинг ґрунтується на принципах:  
- SRP (Single Responsibility Principle) — один клас виконує одну роль  
- DRY (Don’t Repeat Yourself) — уникнення дублювання  
- KISS (Keep It Simple, Stupid) — простота реалізацій  
- SOLID — набір з 5 принципів для якісного ООП-дизайну

9. Теоретичні основи модульного тестування

Модульне тестування — це процес тестування окремих компонентів (функцій, методів, класів) з метою перевірки правильності їхньої поведінки. У Python для цього часто використовують стандартний модуль `unittest`.

Переваги модульного тестування:  
- Автоматична перевірка працездатності коду  
- Швидке виявлення помилок  
- Зменшення ризику регресій після змін у коді  
- Документація логіки очікуваної поведінки

Основні компоненти тестів на Python:  
- `TestCase` — клас, що містить тести  
- `assertEqual`, `assertTrue`, `assertRaises` — методи перевірки  
- `setUp()` — підготовка до тестів  
- `tearDown()` — очищення після тестів

У нашому проєкті реалізовано 20 модульних тестів, які перевіряють:  
- Поповнення рахунку  
- Зняття коштів  
- Обробку помилок (відʼємна сума, недостатньо коштів)  
- Збереження історії  
- Роботу JSON-файлу збереження  
- Тестування логування і валідації вводу