Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

**Лабораторна робота № 5**

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «Патерни проектування»

Виконала:
студентка групи ІА-31
Кизим Є. К.

Київ 2025

# Зміст

**Мета:** Вивчити структуру шаблонів «Adapter», «Builder», «Command», «Chain of responsibility», «Prototype» та навчитися застосовувати їх в реалізації програмної системи.

**Завдання:**

- Ознайомитись з короткими теоретичними відомостями.
- Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.
- Реалізувати один з розглянутих шаблонів за обраною темою.
- Реалізувати не менше 3-х класів відповідно до обраної теми.
- Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму класів, яка представляє використання шаблону в реалізації системи, навести фрагменти коду по реалізації цього шаблону.

**Моя тема:** 27. Особиста бухгалтерія (state, prototype, decorator, bridge, flyweight, SOA) Програма повинна бути наочним засобом для ведення особистих фінансів: витрат і прибутку; з можливістю встановлення періодичних витрат / прибутку (зарплата і орендна плата); введення сканованих чеків з відповідними статтями витрат; побудова статистики; експорт/імпорт в Excel, реляційні джерела даних; різні рахунки; ведення єдиного фонду на всі рахунки (всією сім'єю) – на особливі потреби (ремонт, автомобіль, відпустка); можливість введення вкладів / кредитів для контролю банківських рахунків (звірка нарахованих відсотків з необхідними і т.д.).

## Теоретичні відомості

### 1. Огляд патернів

- **Adapter:** Використовується для адаптації інтерфейсу одного об'єкта до іншого, дозволяючи об'єктам з несумісними інтерфейсами працювати разом.

- **Builder:** Відділяє процес створення складного об'єкта від його представлення, що дозволяє використовувати один і той самий процес конструювання для створення різних об'єктів.
- **Command:** Перетворює запит (виклик методу) на самостійний об'єкт, що дозволяє параметризувати клієнтів чергами запитів, логуванням та підтримувати операції скасування.
- **Chain of Responsibility:** Дозволяє передавати запити послідовно через ланцюжок обробників, де кожен обробник вирішує, чи обробити запит, чи передати його далі.

## 2. Детальний опис шаблону «Prototype» (Прототип)

Для практичної реалізації в системі «Особиста бухгалтерія» було обрано породжувальний шаблон **Prototype**.

**Призначення:** Шаблон «Prototype» використовується для створення нових об'єктів шляхом копіювання вже існуючого екземпляра, який називається прототипом. Цей підхід дозволяє уникнути прямого використання оператора new та залежності клієнтського коду від конкретних класів об'єктів.

**Принцип роботи:** У базовому класі або інтерфейсі визначається метод Clone(). Конкретні класи реалізують цей метод, повертаючи копію самих себе. Це дозволяє маніпулювати об'єктами під час виконання програми шляхом налаштування відповідних прототипів, а також значно зменшує ієрархію успадкування .

**Структура шаблону:**

- **Client (Клієнт):** Створює новий об'єкт, звертаючись до прототипу з вимогою клонувати себе.
- **Prototype (Прототип):** Інтерфейс, що оголошує метод для клонування (Clone) .
- **ConcretePrototype (Конкретний Прототип):** Реалізує операцію клонування, копіюючи свої дані .

**Переваги використання:**

1. **Продуктивність:** Клонування складних об'єктів часто ефективніше, ніж їх створення з нуля та ініціалізація.
2. **Гнучкість:** Різні варіації об'єктів можна отримувати шляхом клонування налаштованих прототипів, а не створенням нових підкласів.
3. **Незалежність:** Клоновані об'єкти можна модифікувати незалежно, не впливаючи на оригінальний об'єкт (прототип).

**Недоліки:** Основним недоліком є складність реалізації «глибокого клонування» (deep copy), коли об'єкт має складну структуру та посилання на інші об'єкти, які теж потрібно рекурсивно копіювати.

У контексті даної лабораторної роботи використання патерну Prototype дозволяє реалізувати функціонал швидкого дублювання транзакцій (наприклад, для регулярних витрат), спрощуючи взаємодію користувача з системою.

## Хід роботи

### Діаграма класів реалізації патерну Prototype

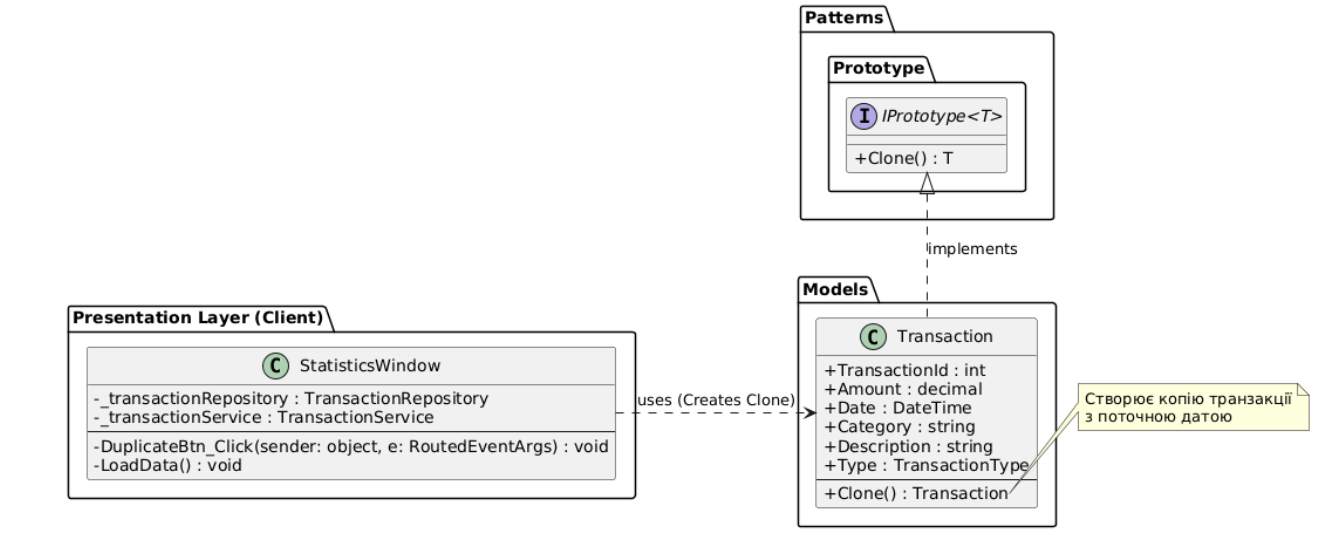



Рисунок 1 – Діаграма класів реалізації патерну Prototype

**Опис реалізації патерну:**

Для реалізації патерну було створено узагальнений інтерфейс IPrototype<T>, який містить метод Clone(). Клас Transaction імплементує цей інтерфейс, реалізуючи логіку поверхневого копіювання своїх полів (Amount, Category, Description), але при цьому встановлює нову дату (DateTime.Now) для створеного клона. Клієнтський клас StatisticsWindow використовує цей механізм для створення нових записів на основі існуючих без необхідності повторного введення даних користувачем.

**Фрагменти програмного коду:**

Інтерфейс прототипу (Patterns/Prototype/IPrototype.cs):

```
namespace personalAccounting.Patterns.Prototype
{
    public interface IPrototype<T>
```

```
    {
        T Clone();
    }
}
```

Реалізація прототипу в класі сутності (Models/Transaction.cs):

```
using System;
using personalAccounting.Patterns.Prototype;

namespace personalAccounting.Models
{
    public class Transaction : IPrototype<Transaction>
    {
        public int TransactionId { get; set; }
        public decimal Amount { get; set; }
        public DateTime Date { get; set; }
        public string Category { get; set; }
        public string Description { get; set; }

        public Transaction Clone()
        {
            return new Transaction
            {
                Amount = this.Amount,
                Type = this.Type,
                Category = this.Category,
                Description = this.Description,

                Date = DateTime.Now
```

```
            };
        }
    }
}
```

Використання патерну клієнтом (StatisticsWindow.xaml.cs):

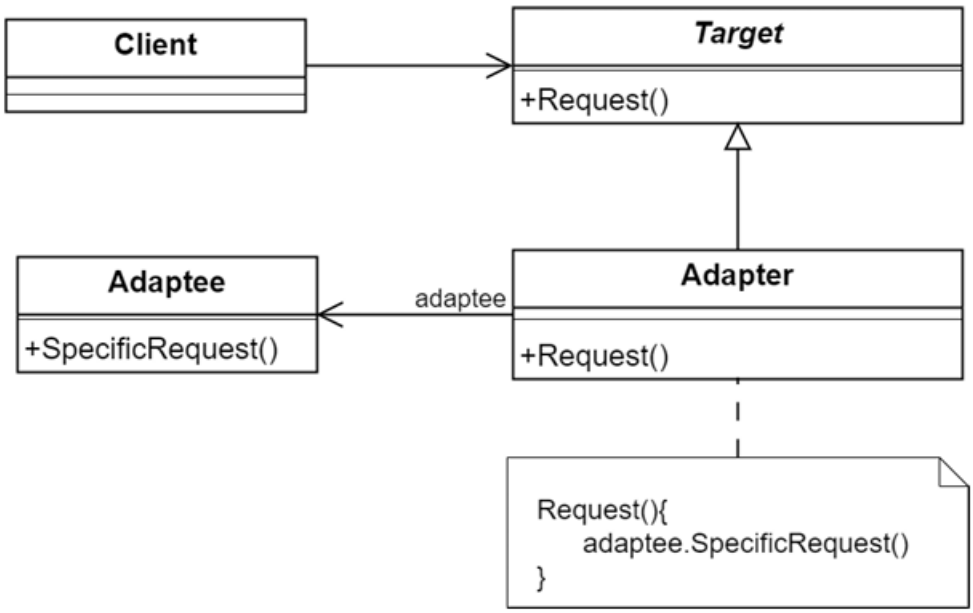
```
private void DuplicateBtn_Click(object sender, RoutedEventArgs e)
{
    if (StatsGrid.SelectedItem is Transaction selectedTransaction)
    {
        Transaction clone = selectedTransaction.Clone();

        bool success = _transactionService.AddExpense(clone, 1);

        if (success)
        {
            MessageBox.Show("Транзакцію успішно продубльовано!", "Успіх");
            LoadData();
        }
    }
    else
    {
        MessageBox.Show("Будь ласка, виберіть витрату для дублювання.", "Увага");
    }
}
```

## Контрольні питання

**1. Яке призначення шаблону «Адаптер»?**

Він дозволяє об'єктам з несумісними інтерфейсами працювати разом. Адаптер конвертує інтерфейс одного класу в інший інтерфейс, який очікує клієнт.

**2. Нарисуйте структуру шаблону «Адаптер».**




**3. Які класи входять в шаблон «Адаптер», та яка між ними взаємодія?**

Структура патерну включає чотири ключові елементи: Client, Target, Adaptee та Adapter. Механізм роботи базується на тому, що клас Adapter імплементує інтерфейс Target, якого очікує клієнт, і водночас зберігає посилання на екземпляр класу Adaptee. При виклику методів інтерфейсу Target адаптер делегує виконання відповідним методам об'єкта Adaptee. Це 10 забезпечує інкапсуляцію специфічної реалізації та дозволяє клієнтському коду взаємодіяти з різними класами через єдиний інтерфейс.

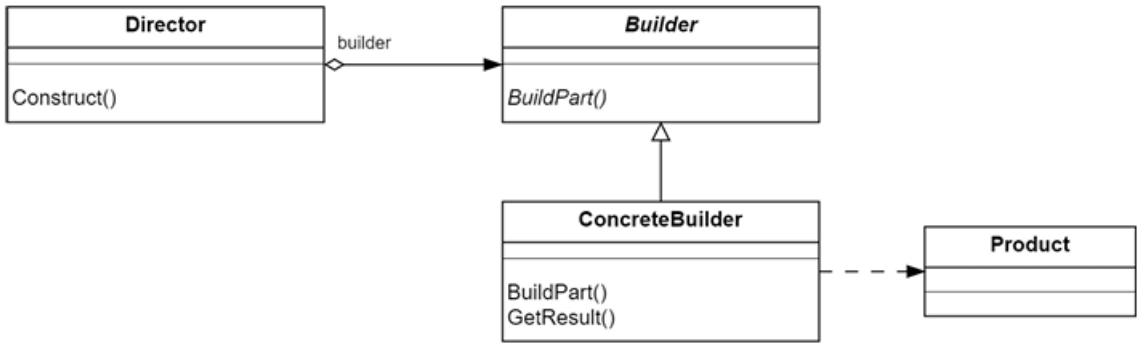
**4. Яка різниця між реалізацією «Адаптера» на рівні об’єктів та на рівні класів?**

Існує два підходи до реалізації патерну. Адаптер рівня об'єктів базується на механізмі композиції: клас-адаптер містить поле з посиланням на об'єкт Adaptee і перенаправляє йому виконання запитів. Натомість адаптер рівня класів реалізується через множинне успадкування, розширюючи одночасно інтерфейс клієнта та клас адаптованого компонента

**5. Яке призначення шаблону «Будівельник»?**

Він відокремлює конструювання складного об'єкта від його представлення. Це дозволяє використовувати один і той самий процес конструювання для створення різних об'єктів.

**6. Нарисуйте структуру шаблону «Будівельник».**



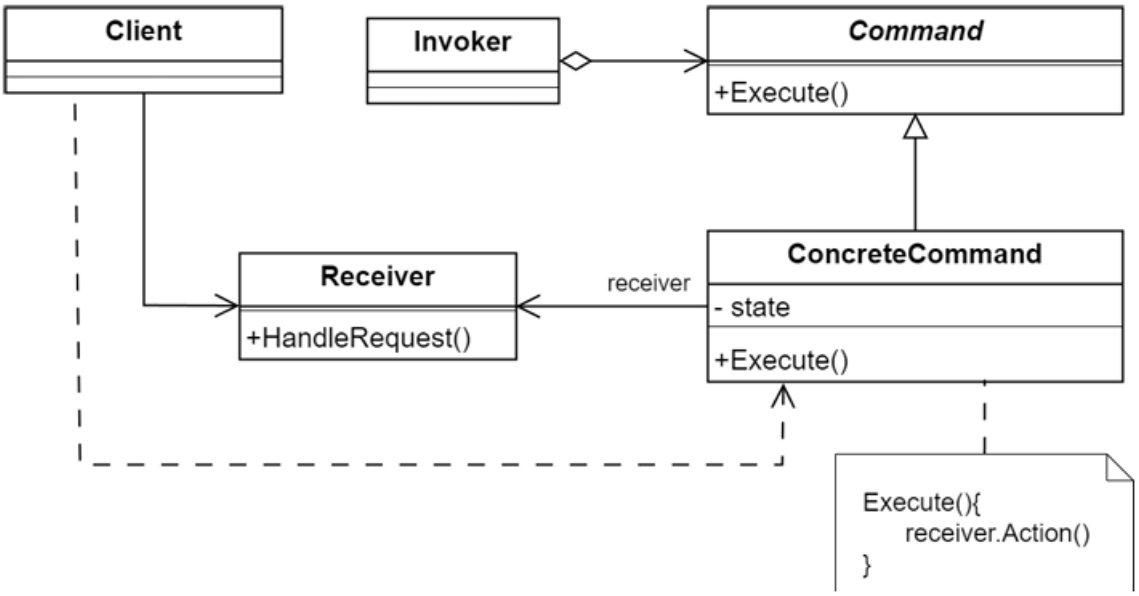

**7. Які класи входять в шаблон «Будівельник», та яка між ними взаємодія?**

Структура патерну включає чотири ключові елементи: Director, Builder, ConcreteBuilder та Product. Механізм роботи базується на тому, що Director (Розпорядник) керує алгоритмом створення об'єкта, делегуючи виконання конкретних кроків об'єкту ConcreteBuilder через загальний інтерфейс Builder. Будівельник послідовно накопичує стан майбутнього об'єкта Product, реалізуючи специфіку конструювання його частин. Це дозволяє ізолювати складний процес створення від самого об'єкта та використовувати один алгоритм конструювання для отримання різних результатів.

**8. У яких випадках варто застосовувати шаблон «Будівельник»"? 9. Яке призначення шаблону «Команда»?**

Шаблон використовується, коли алгоритм створення об'єкта є складним і має бути відокремленим від його складових частин (наприклад, генерація HTML-сторінок). Також він дозволяє створювати різні варіації продукту, використовуючи єдиний процес конструювання (наприклад, експорт даних у різні формати).

**10. Нарисуйте структуру шаблону «Команда».**




**11. Які класи входять в шаблон «Команда», та яка між ними взаємодія?**

У шаблон входять класи Client, Invoker, Command, Receiver та ConcreteCommand . Invoker викликає метод Execute у Command, а ConcreteCommand перенаправляє запит до Receiver, викликаючи метод Action .

**12. Розкажіть як працює шаблон «Команда».**

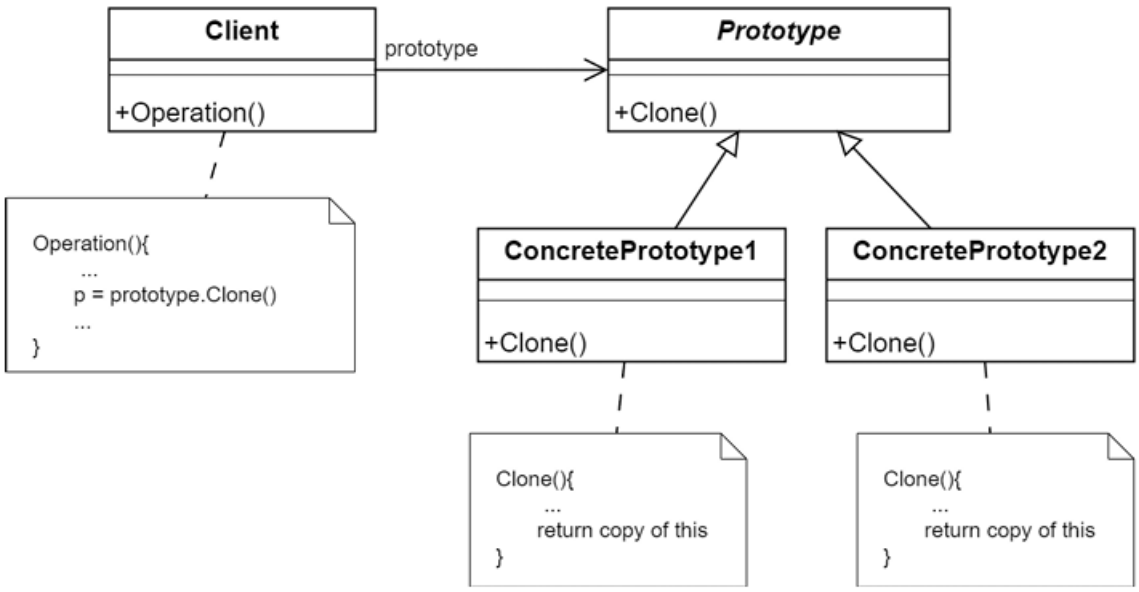
Основна задача об'єкта-команди — виступати посередником, який передає запит конкретному виконавцю, не виконуючи бізнес-логіку самостійно.

Завдяки цьому команда може зберігати параметри запиту, що дозволяє реалізувати історію дій або функцію їх скасування (Undo).
Схема взаємодії наступна: клієнт створює та налаштовує команду, після чого передає її об'єкту-ініціатору (Invoker). Останній же просто активує команду в потрібний момент часу.

**13. Яке призначення шаблону «Прототип»?**

Дозволяє копіювати об'єкти, не вдаючись у подробиці їхньої реалізації та не залежачи від їхніх класів. Це альтернатива створенню через new та налаштуванню з нуля.

**14. Нарисуйте структуру шаблону «Прототип».**




**15. Які класи входять в шаблон «Прототип», та яка між ними взаємодія?**

До шаблону входять класи Client, Prototype та конкретні реалізації ConcretePrototype1 і ConcretePrototype2 . Клієнт виконує операцію, викликаючи метод Clone у прототипу, який повертає копію самого себе.

**16. Які можна привести приклади використання шаблону «Ланцюжок відповідальності»?**

Веб-сервер: Вхідний запит (Аутентифікація -> Валідація даних -> Кешування -> Контролер).

**Висновки:** Під час виконання лабораторної роботи я ознайомилась з принципами роботи та структурою таких шаблонів проєктування, як «Adapter», «Builder», «Command», «Chain of Responsibility» та «Prototype».
Практична частина роботи полягала в реалізації породжувального шаблону **Prototype (Прототип)** у системі «Особиста бухгалтерія». Для цього було розроблено узагальнений інтерфейс IPrototype<T> та імплементовано метод Clone() у класі сутності Transaction. Це дозволило створити механізм копіювання об'єктів транзакцій, де нова копія отримує всі дані від оригіналу, але автоматично встановлює актуальну дату створення. Впровадження цього шаблону дозволило реалізувати функцію швидкого дублювання витрат у користувацькому інтерфейсі, що значно спрощує внесення регулярних платежів та зменшує залежність клієнтського коду від деталей ініціалізації об'єктів.

**Репозиторій:** посилання

# ДОДАТКИ

Додаток А

Демонстрація роботи програми: дублювання транзакції у списку