Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки

Лабораторна робота №5
«Технології розроблення програмного забезпечення»
Тема: « E-mail клієнт»

Виконав:
Студент групи ІА-34
Марченко А.О.
Перевірив:
Мягкий Михайло Юрійович

Київ 2025

Тема: Патерни проектування.

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.

15. **E-mail клієнт** (singleton, builder, decorator, template method, interpreter, client-server)

Поштовий клієнт повинен нагадувати функціонал поштових програм Mozilla Thunderbird, The Bat і т.д. Він повинен сприймати і коректно обробляти pop3/smtp/imap протоколи, мати функції автонастройки основних поштових провайдерів для України (gmail, ukr.net, i.ua), розділяти повідомлення на папки/категорії/важливість, зберігати чернетки незавершених повідомлень, прикріплювати і обробляти прикріплені файли.

Теоретичні відомості

Шаблон «Adapter»

Призначення патерну: Шаблон "Adapter" (Адаптер) використовується для адаптації інтерфейсу одного об'єкту до іншого. Наприклад, існує декілька бібліотек для роботи з принтерами, проте кожна має різний інтерфейс (хоча однакові можливості і призначення). Має сенс розробити уніфікований інтерфейс (сканування, асинхронне сканування, двостороннє сканування, потокове сканування і тому подібне), і реалізувати відповідні адаптери для приведення бібліотек до уніфікованого інтерфейсу.

Це дозволить в програмі звертатися до загального інтерфейсу, а не приводити різні сценарії роботи залежно від способу реалізації бібліотеки. Адаптери також називаються "wrappers" (обгортками).

Переваги та недоліки:

+ Відокремлює інтерфейс або код перетворення даних від основної бізнес- логіки.

+ Можна добавляти нові адаптери не змінюючи код у класі Client.

- Умовним недоліком можна назвати збільшення кількості класів, але за рахунок використання патерна Адаптер програмний код, як привило, стає легше читати.

Шаблон «Builder»

Призначення патерну: Шаблон «Builder» (Будівельник) використовується для відділення процесу створення об'єкту від його представлення. Це доречно у випадках, коли об'єкт має складний процес створення (наприклад, Web-сторінка як елемент повної відповіді web-сервера) або коли об'єкт повинен мати декілька різних форм створення (наприклад, при конвертації тексту з формату у формат).

Переваги та недоліки:

+ Дозволяє використовувати один і той самий код для створення різноманітних продуктів.

- Клієнт буде прив’язаний до конкретних класів будівельників, тому що в інтерфейсі будівельника може не бути методу отримання результату.

Шаблон «Command»

Призначення патерну: Шаблон "command" (команда) перетворить звичайний виклик методу в клас [6]. Таким чином дії в системі стають повноправними об'єктами. Це зручно в наступних випадках:

• Коли потрібна розвинена система команд – відомо, що команди будуть добавлятися;

• Коли потрібна гнучка система команд – коли з'являється необхідність додавати командам можливість відміни, логування і інш.;

• Коли потрібна можливість складання ланцюжків команд або виклику команд в певний час.

Об'єкт команда сама по собі не виконує ніяких фактичних дій окрім перенаправлення запиту одержувачеві (тобто команди все ж виконуються одержувачем), однак ці об'єкти можуть зберігати дані для підтримки додаткових функцій відміни, логування і інш. Наприклад, команда вставки символу може запам'ятовувати символ, і при виклику відміни викликати відповідну функцію витирання символу. Можна також визначити параметр «застосовності» команди (наприклад, на картинці писати не можна) – і використати цей аттрибут для засвічування піктограми в меню.

Такий підхід до команд дозволяє побудувати дуже гнучку систему команд, що настроюється. У більшості додатків це буде зайвим (використовується спрощений варіант), проте життєво важливий в додатках з великою кількістю команд (редактори).

Переваги та недоліки:

+ Ініціатор виконання команди не знає деталей реалізації виконавця команди.

+ Підтримує операції скасування та повторення команд.

+ Послідовність команд можна логувати і при необхідності виконати цю послідовність ще раз.

+ Простота розширення за рахунок додавання нових команд без необхідності внесення змін в уже існуючий код (принцип відкритості-закритості).

Шаблон «Chain of Responsibility»

Призначення патерну: Шаблон «Chain of responsibility» (Ланцюжок відповідальності) частково можна спостерігати в житті, коли підписання

відповідного документу проходить від його складання у одного із співробітників компанії через менеджера і начальника до головного начальника, який ставить свій підпис.

Переваги та недоліки:

+ Зменшує залежність між клієнтом та обробниками: клієнт не знає хто обробить запит, а обробники не знають структуру ланцюжка.

+ Реалізовує додаткову гнучкість в обробці запиту: легко добавити або вилучити з ланцюжка нові обробники.

- Запит може залишитися ніким не опрацьованим: запит не має вказаного обробника, тому може бути не опрацьованим.

Шаблон «Prototype»

Призначення патерну: Шаблон «Prototype» (Прототип) використовується для створення об'єктів за «шаблоном» (чи «кресленням», «ескізом») шляхом копіювання шаблонного об'єкту, який називається прототипом. Для цього визначається метод «клонувати» в об'єктах цього класу.

Цей шаблон зручно використати, коли заздалегідь відомо як виглядатиме кінцевий об'єкт (мінімізується кількість змін до об'єкту шляхом створення шаблону), а також для видалення необхідності створення об'єкту – створення відбувається за рахунок клонування, і самій програмі немає необхідності знати, як створювати об'єкт.

Також, це дозволяє маніпулювати об'єктами під час виконання програми шляхом настроювання відповідних прототипів; значно зменшується ієрархія наслідування (оскільки в іншому випадку це були б не прототипи, а вкладені класи, що наслідують).

Переваги та недоліки:

+ За рахунок клонування складних об’єктів замість їх створення, підвищується продуктивність.

+ Різні варіації об’єктів можна отримувати за рахунок клонування, а не розширення ієрархії класів.

+ Вища гнучкість, тому що клоновані об’єкти можна модифікувати незалежно, не впливаючи на об’єкт з якого була зроблена копія.

- Реалізація глибокого клонування досить проблематична, коли об’єкт що клонується містить складну внутрішню структуру та посилання на інші об’єкти.

- Надмірне використання патерну Прототип може привести до ускладнення коду та проблем із супроводом такого коду.

Хід роботи

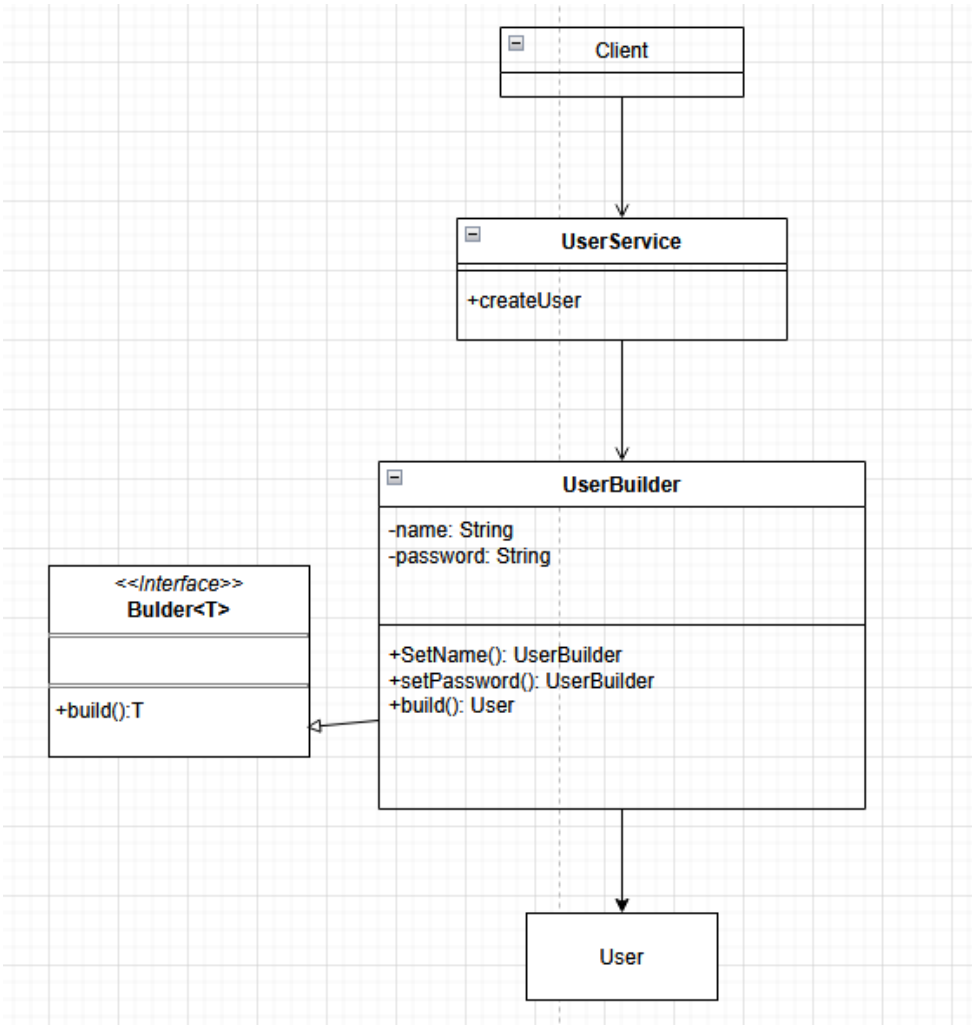


Рис 5.1 -Застосування патерну Builder для User

На діаграмі зображено основні елементи. Інтерфейс Builder<T> — це узагальнений інтерфейс, який визначає метод build(). Цей метод має повертати об’єкт типу T, тобто будь-який клас, який створюється за допомогою конкретного будівельника. Таким чином забезпечується універсальність і можливість використання спільного підходу для побудови різних типів об’єктів. Клас UserBuilder реалізує інтерфейс Builder<User> і відповідає за поетапне створення об’єкта користувача. У ньому є приватні поля name і password, а також публічні методи: setName, setPassword , build Клас User — це кінцевий продукт побудови. Він представляє готового користувача, який створюється через UserBuilder.

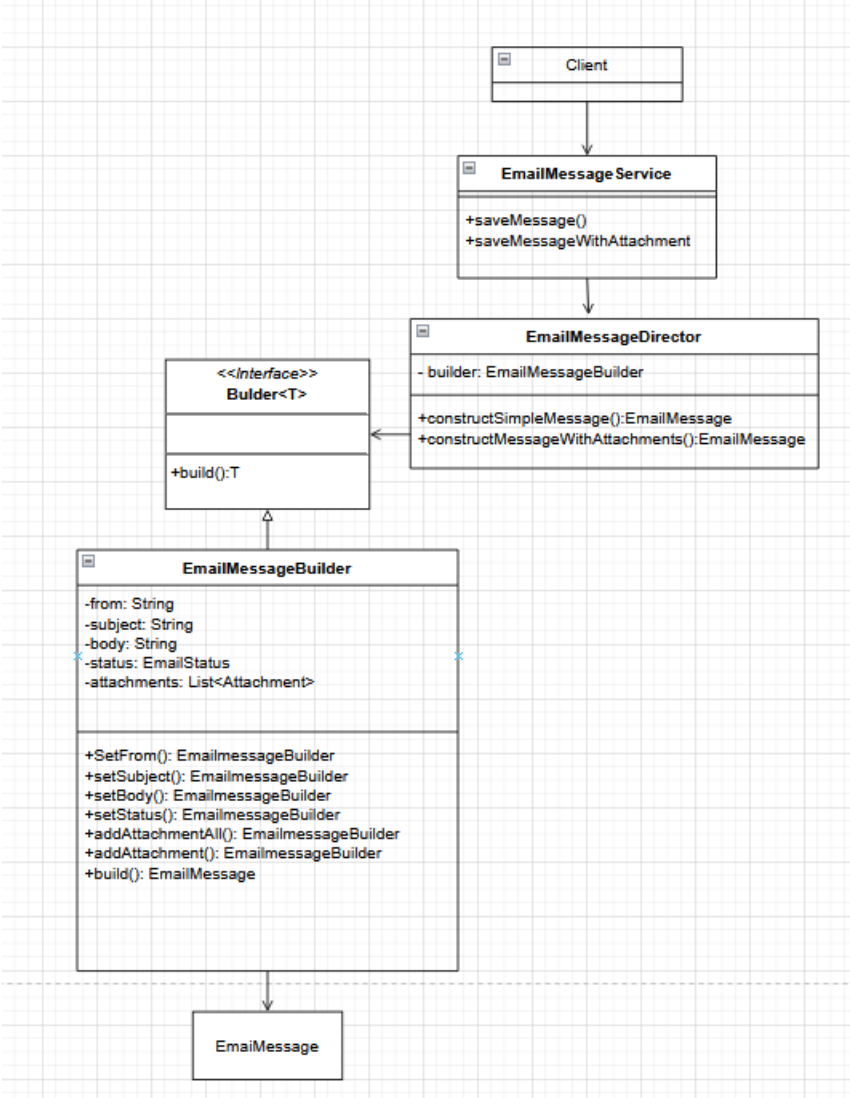


Рис 5.2 -Застосування патерну Builder для EmailMessage

Інтерфейс Builder<T> — це узагальнений інтерфейс, який визначає метод build(). Цей метод має повертати об’єкт типу T, тобто будь-який клас, який

створюється за допомогою конкретного будівельника. Клас EmailMessageBuilder — реалізує інтерфейс Builder<EmailMessage> і безпосередньо відповідає за побудову повідомлення. Він має такі поля: відправник, тема листа, текст повідомлення, статус повідомлення, список вкладень.

Клас має набір методів, кожен з яких задає певний параметр повідомлення (setFrom(), setSubject(), setBody(), setStatus() тощо). Наприкінці метод build() створює об’єкт EmailMessage з усіма встановленими параметрами.

Клас EmailMessageDirector — керує процесом побудови, визначаючи, які кроки виконуються для створення конкретного типу повідомлення.

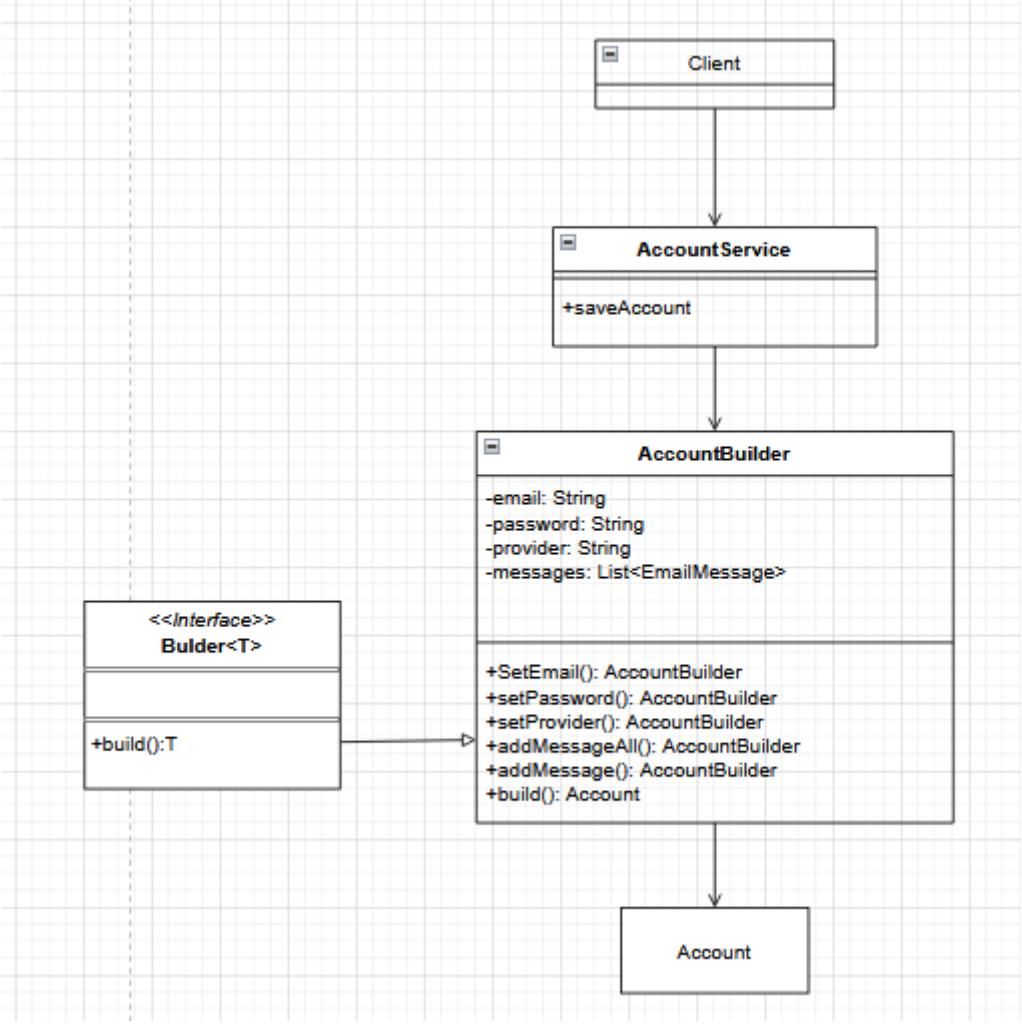


Рис 5.3 -Застосування патерну Builder дляAccount

Інтерфейс Builder<T> — це узагальнений інтерфейс, який визначає метод build(). Цей метод має повертати об’єкт типу T, тобто будь-який клас, який створюється за допомогою конкретного будівельника. Клас AccountBuilder —

реалізує інтерфейс Builder<Account> і безпосередньо відповідає за побудову об’єкта облікового запису. Він містить такі поля: адреса електронної пошти користувача, пароль, провайдер поштової служби, список повідомлень типу EmailMessage, що належать цьому акаунту.

Клас має набір методів, які поступово задають усі властивості: setEmail(), setPassword(), setProvider() — для встановлення основних параметрів, naddMessageAll() і addMessage() — для додавання одного або кількох повідомлень до акаунту. Метод build() створює та повертає готовий об’єкт типу Account.

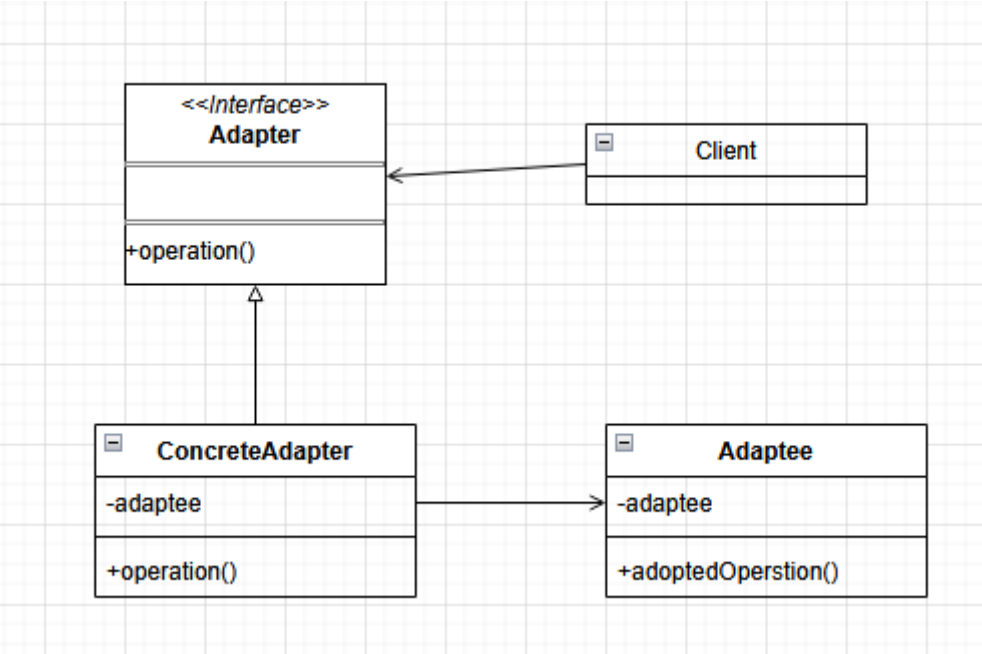
Посилання на код:
https://github.com/BeautifulBublik/TRPZ/tree/master/EmailClient/src/main/java/com/example/emailclient

Питання до лабораторної роботи

1. Яке призначення шаблону «Адаптер»?

   Призначення шаблону «Адаптер» — забезпечити взаємодію між об'єктами з несумісними інтерфейсами, перетворюючи інтерфейс одного класу на інший, зрозумілий для клієнта. Це дозволяє об'єднати класи, які не можуть працювати разом "напряму", наприклад, при використанні стороннього класу, інтерфейс якого не відповідає решті коду програми.

2. Нарисуйте структуру шаблону «Адаптер».



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

Шаблон "Адаптер" включає класи Адаптер (Adapter), Цільовий інтерфейс і Адаптований клас (Adaptee). Взаємодія полягає в тому, що Адаптер використовує інтерфейс Цільового інтерфейсу для роботи з Адаптованим класом (Adaptee), якого, як правило, не можна безпосередньо змінити. Таким чином, Адаптер дозволяє двом несумісним інтерфейсам працювати разом.

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

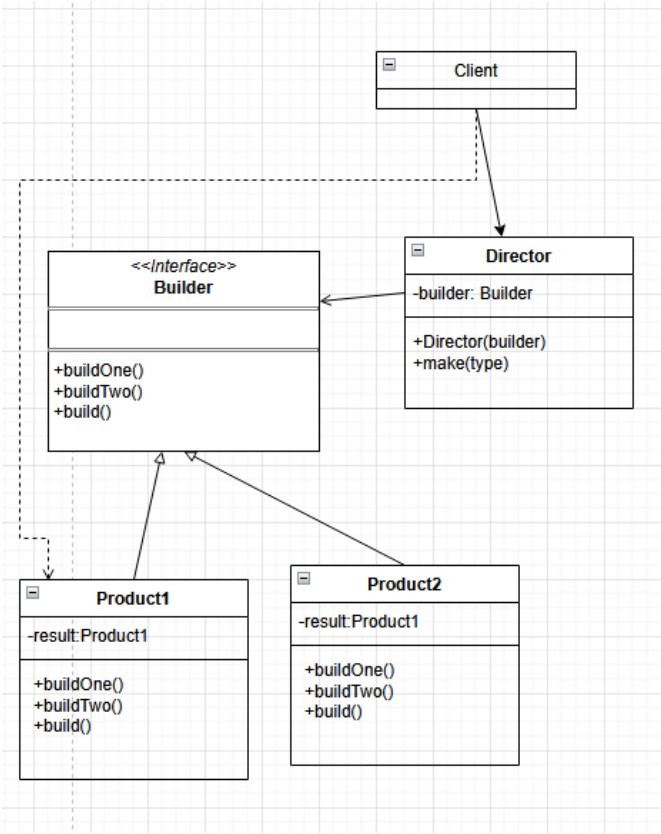
Різниця між реалізацією паттерна "Адаптер" на рівні класів та об'єктів полягає в тому, що адаптер на рівні класів використовує успадкування для зв'язування об'єктів, тоді як адаптер на рівні об'єктів використовує композицію для зв'язування об'єктів. Адаптер на рівні класів може мати тільки один клас-клієнт, а адаптер на рівні об'єктів може мати кілька класів-клієнтів.

5. Яке призначення шаблону «Будівельник»?

Шаблон «Будівельник» (Builder) — це шаблон проєктування, який дозволяє створювати складні об'єкти покроково, приховуючи процес конструювання від клієнта і дозволяючи отримати різні подання одного й того ж об'єкта. Його призначення – відокремити алгоритм побудови

складного об'єкта від його частин, що дозволяє використовувати той самий код побудови для отримання різних версій об'єкта

6. Нарисуйте структуру шаблону «Будівельник».



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

Шаблон проєктування «Будівельник» складається з чотирьох основних класів: Будівельник (інтерфейс для створення частин продукту), Конкретний будівельник (реалізує інтерфейс Builder, створює і збирає продукт), Керівник (користується інтерфейсом Builder для конструювання об'єкта) і Продукт (складний об'єкт, який будується). Взаємодія між ними полягає в тому, що Керівник викликає методи Конкретного будівельника для поетапної побудови Продукту, а Будівельник забезпечує стандартизований інтерфейс для цього процесу

8. У яких випадках варто застосовувати шаблон «Будівельник»"?

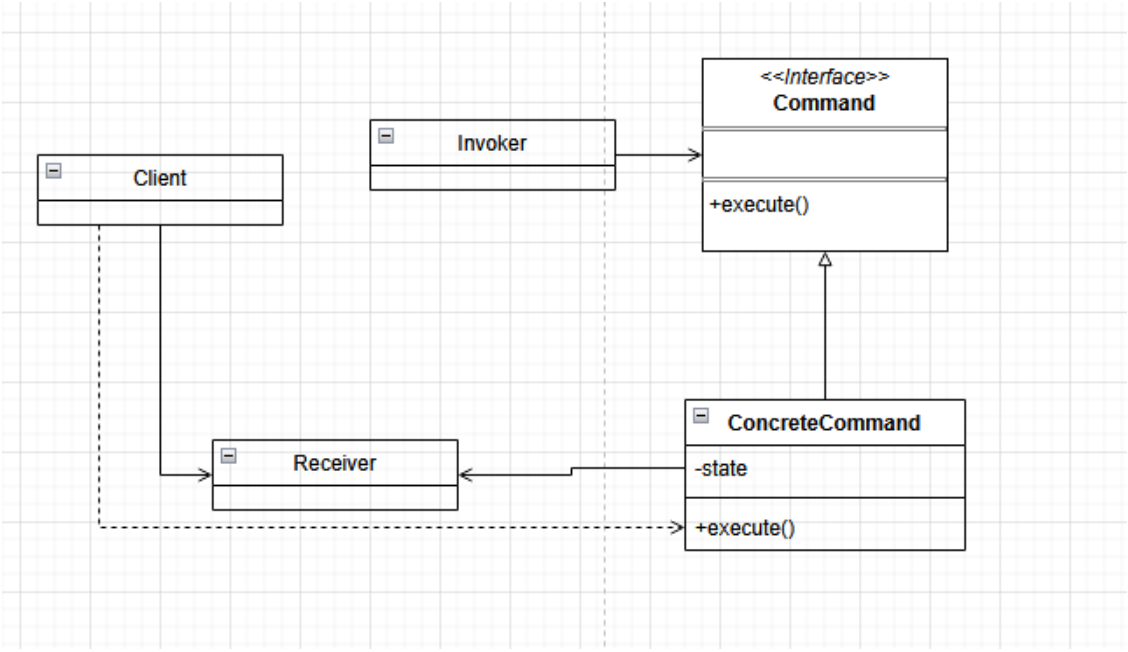
Шаблон «Будівельник» використовується, коли алгоритм створення складного об’єкта повинен бути незалежним від його складових частин та способу їх компонування. Він також корисний, коли процес

конструювання дозволяє створювати різні варіації одного й того самого об’єкта

9. Яке призначення шаблону «Команда»?

Шаблон «Команда» (Command) призначений для інкапсуляції запиту як об'єкта, що дозволяє параметризувати клієнтів різними запитами, ставити запити в чергу, відстежувати їх, а також підтримувати скасування операцій. Він відокремлює об'єкт, який ініціює дію, від об'єкта, який її виконує

10. Нарисуйте структуру шаблону «Команда».



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

Шаблон проєкту "Команда" (Command) включає чотири класи: Client (Клієнт), Invoker (Викликавач), Command (Команда), та Receiver (Одержувач). Client створює об'єкти ConcreteCommand (Конкретна команда), зв'язує їх з Receiver, а потім передає об'єкт ConcreteCommand через Invoker. Invoker зберігає посилання на об'єкт Command і викликає його метод execute(). ConcreteCommand реалізує інтерфейс Command і містить посилання на Receiver, викликаючи потрібний метод на ньому під час виконання. Receiver є об'єктом, який знає, як виконати операцію, яку потребує Command

12. Розкажіть як працює шаблон «Команда».

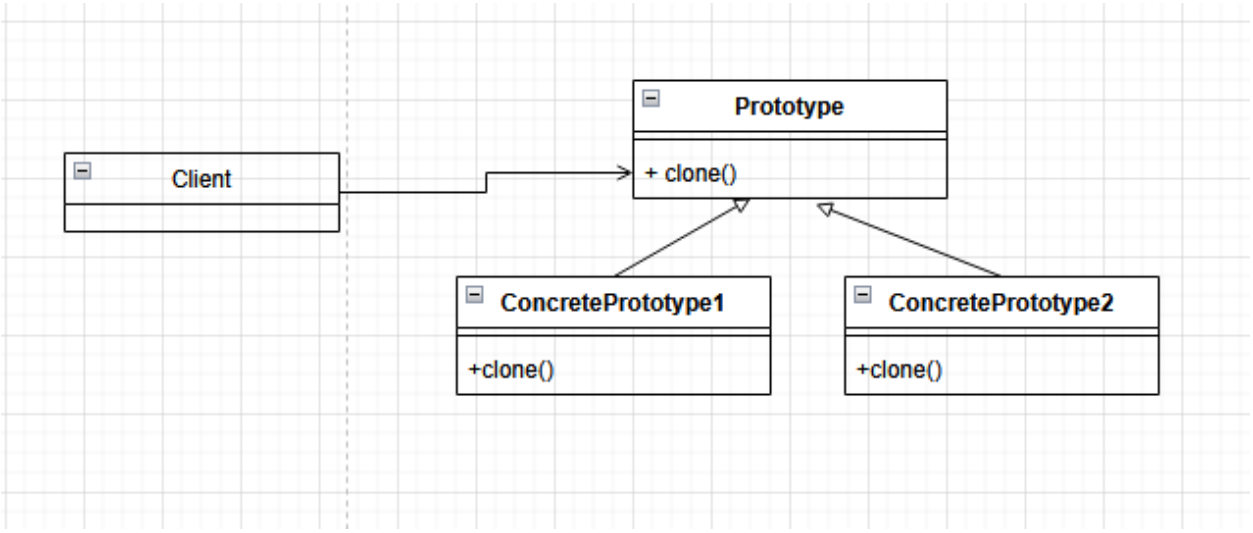
Робота шаблону відбувається так: клієнт створює об’єкт команди, передаючи йому посилання на отримувача. Потім клієнт передає цю команду відправнику. Коли потрібно виконати дію, відправник викликає метод execute() команди, а та в свою чергу звертається до отримувача, який виконує реальну операцію.

Наприклад, у текстовому редакторі кнопки «Копіювати», «Вставити» і «Скасувати» можуть бути реалізовані як команди. Кожна кнопка викликає свій об’єкт-команду, який знає, що саме потрібно зробити і як це зробити, але сам інтерфейс кнопки нічого не знає про деталі реалізації.

13. Яке призначення шаблону «Прототип»?

Шаблон «Прототип» — це породжувальний шаблон проектування, який дозволяє створювати нові об'єкти шляхом клонування (копіювання) існуючих об'єктів-прототипів, а не створення нових об'єктів з нуля. Це корисно для створення складних або ресурсоємних об'єктів, зменшує код ініціалізації та дозволяє створювати об'єкти під час виконання програми

14. Нарисуйте структуру шаблону «Прототип».



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

Шаблон проєктування «Прототип» складається з трьох класів: Prototype, ConcretePrototype і Client. Взаємодія між ними полягає в

тому, що Client запитує Prototype про клонування себе, а ConcretePrototype (який реалізує Prototype) створює копію самого себе для передачі клієнту

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

Шаблон «Ланцюжок відповідальності» (Chain of Responsibility) використовується тоді, коли потрібно, щоб запит проходив через послідовність обробників, і кожен з них міг або обробити запит, або передати його далі по ланцюжку. Це дозволяє уникнути жорсткої прив'язки відправника запиту до конкретного обробника та зробити систему гнучкішою.

У вебфреймворках (Spring, Express.js) вхідний запит часто проходить через послідовність фільтрів або middleware. Кожен елемент ланцюга може: перевірити авторизацію користувача, розпарсити тіло запиту, перевірити токен доступу, виконати логування, і, якщо умови виконані, передати запит далі.

Висновок:Було реалізовано практичне застосування структурних та поведінкових шаблонів проєктування, зокрема «Builder», а також розглянуто інші патерни — «Adapter», «Command», «Chain of Responsibility» та «Prototype». Основна мета роботи полягала у вивченні структури цих шаблонів та формуванні навичок їх використання під час побудови програмних систем.

AccountBuilder — відповідає за поетапне створення об'єкта користувацького акаунта, який містить такі поля, як email, пароль, провайдер та список повідомлень. Використання шаблону Builder у цьому класі дозволило спростити процес створення складного об'єкта з багатьма параметрами, зробивши код більш читабельним, гнучким і безпечним щодо помилок при ініціалізації.

EmailMessageBuilder — використовується для побудови об'єктів електронних повідомлень. Даний білдер дозволяє гнучко додавати дані про

відправника, тему, тіло, статус повідомлення та вкладення. Завдяки цьому підхід забезпечує зручне формування як простих, так і складних повідомлень, а також реалізує можливість розширення через додатковий клас EmailMessageDirector, який координує процес створення повідомлень різних типів.

UserBuilder — реалізує побудову користувачів системи. Завдяки шаблону Builder можливо поетапно створити об’єкт користувача, визначаючи лише необхідні параметри, що особливо зручно при реєстрації або додаванні нових користувачів у систему без надлишкових конструкторів.