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

Лабораторна робота №5

Технології розроблення програмного забезпечення

Шаблони «Adapter», «Builder», «Command», «Chain Of Responsibility», «Prototype»

Варіант 30

Виконав
студент групи ІА-13
Якін С.О.

Перевірив:
Мягкий М.Ю.

Київ 2023

Тема: Шаблони «Adapter», «Builder», «Command», «Chain Of Responsibility», «Prototype»

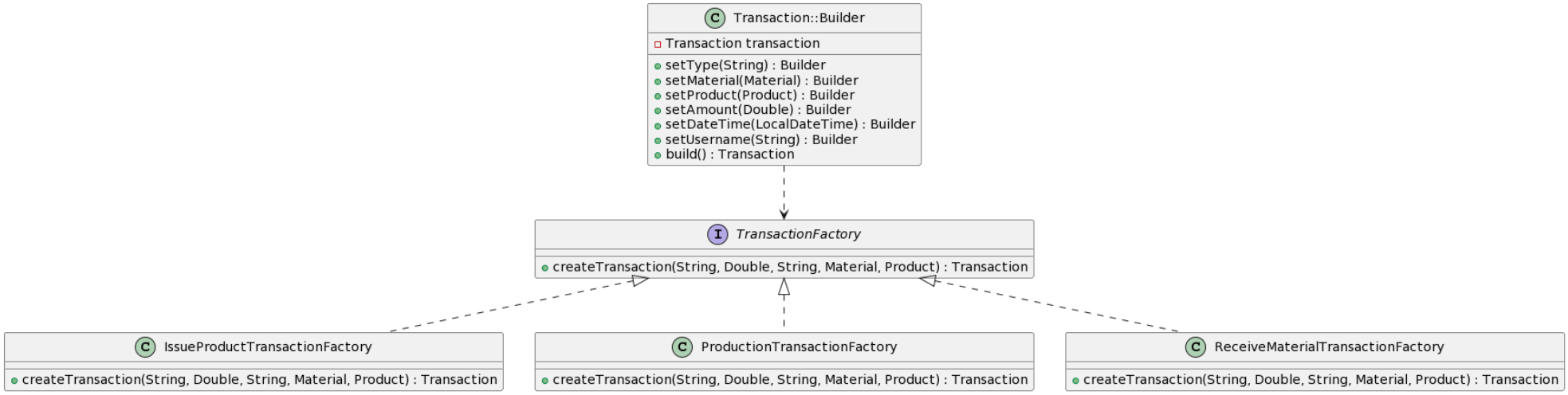
Завдання:

1. Ознайомитися з короткими теоретичними відомостями.
2. Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.
3. Застосування одного з розглянутих шаблонів при реалізації програми.

Хід роботи:

Тема: « Система складського обліку виробництва »;

Згідно мого варіанту, було виконано шаблон проектування «Builder». Було вирішено його реалізовувати у класі «TransactionBuilder», який створюватиме під наші потреби унікальний об’єкт класу Transaction.



(на рисунку показано також реалізацію «Factory Method»)

Основний клас, який використовує цей шаблон, це Transaction. Transaction - це сутність, яка представляє транзакцію на складі, включаючи деталі, такі як тип транзакції, продукт, матеріал, кількість, час та інформація про користувача. Внутрішній клас Transaction::Builder є реалізацією шаблону Builder. Цей клас використовується для створення екземплярів Transaction, дозволяючи встановлювати різні атрибути крок за кроком. Завдяки цьому підхід дозволяє уникнути складності конструктора з багатьма параметрами та робить код більш читабельним і зрозумілим. Використання Builder також дозволяє встановлювати тільки ті атрибути, які необхідні для конкретного випадку, забезпечуючи гнучкість у створенні об'єктів. На діаграмі показано, як TransactionFactory та його реалізації використовують Builder.

```
    protected Transaction() { }
    9 usages  serrb
    public static class Builder {
        8 usages
        private final Transaction transaction;
        3 usages  serrb
        public Builder() { transaction = new Transaction(); }
        serrb
        public Builder setType(String type) {
            transaction.type = type;
            return this;
        }
        serrb
        public Builder setMaterial(Material material) {
            transaction.material = material;
            return this;
        }
        serrb
        public Builder setProduct(Product product) {
            transaction.product = product;
            return this;
        }
        serrb
        public Builder setAmount(Double amount) {
            transaction.amount = amount;
            return this;
        }
        3 usages  serrb
        public Builder setDateTime(LocalDateTime dateTime) {
            transaction.dateTime = dateTime;
            return this;
        }
        serrb
        public Builder setUsername(String username) {
            transaction.username = username;
            return this;
        }
        3 usages  serrb
        public Transaction build() { return transaction; }
    }
}
```

Ініціалізація основних параметрів

```java
public Transaction createTransaction(String type,
                                     Double amount,
                                     String username,
                                     Material material,
                                     Product product) {
    return new Transaction.Builder()
            .setType("ISSUE")
            .setProduct(product)
            .setAmount(amount)
            .setDateTime(LocalDateTime.now())
            .setUsername(username)
            .build();
}
```

Приклад використання у коді

На прикладі показано процес створення транзакції з типом «Видача», яку ми власне потім можемо використовувати у наших звітах. Як ми бачимо у результаті створення власного Builder ми полегшили читаємість коду, та легкість модернізації у майбутньому.

Висновок: на цій лабораторній роботі я познайомився з такими паттернами, як «adapter», «builder», «command», «chain of responsibility», «prototype», а також розібрався у принципі їх роботи та сфері використання. На практиці спроектував та реалізував паттерн «builder» у своєму проекті.