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 №6

## Технології розроблення програмного забезпечення

*Шаблони «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator»*

Варіант 30

Виконав
студент групи ІА-13
Якін С. О.

Перевірив:
Мягкий М.Ю.

Київ 2023

**Тема:** Шаблони «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator»

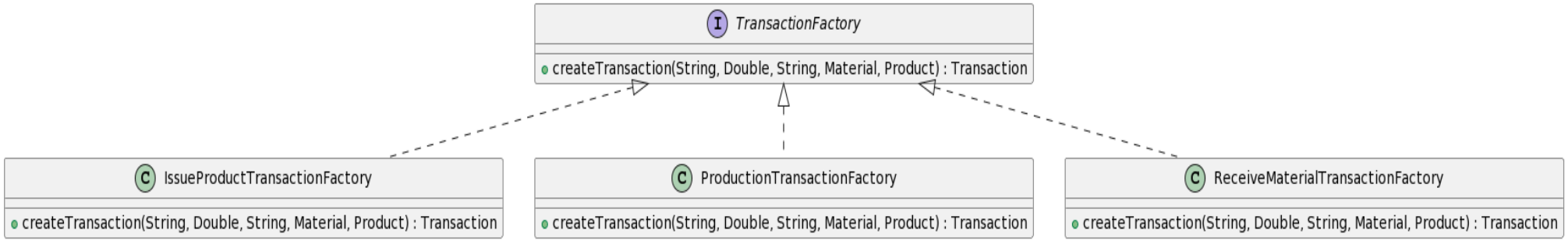
**Завдання:**

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 «Factory method». Було вирішено його реалізовувати для класу «Transaction», який створюватиме знімок стану нашого користувача, щоб в майбутньому могти повернутися до нього.




На діаграмі представлено застосування шаблону проектування "Factory Method" у контексті створення транзакцій у системі управління складом. Цей шаблон було вибрано для реалізації, оскільки він дозволяє гнучко змінювати типи транзакцій та ефективно вносити ці транзакції до таблиці, що є ключовим аспектом у системах, що обробляють різні види операцій. Інтерфейс TransactionFactory визначає метод createTransaction, який є основою для створення об'єктів транзакцій. Цей метод приймає параметри, необхідні для створення транзакції, такі як тип транзакції, кількість, інформація про користувача, матеріали та продукти. Використання інтерфейсу дозволяє абстрагувати процес створення транзакцій від конкретних класів.

Конкретні реалізації цього інтерфейсу, такі як IssueProductTransactionFactory, ProductionTransactionFactory і ReceiveMaterialTransactionFactory, кожна з яких відповідає за створення певного типу транзакції. Кожна з цих фабрик реалізує метод createTransaction відповідно до своїх специфік, забезпечуючи створення об'єктів транзакцій з відповідними характеристиками.

```java
package com.example.warehouse.factories.transaction;

import ...

no usages  serrb
public class IssueProductTransactionFactory  implements TransactionFactory {
    3 usages  serrb
    @Override
    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();
    }
}
```

**IssueProductTransactionFactory** може створювати транзакції видачі продукції

```java
package com.example.warehouse.factories.transaction;

import ...

2 usages  serrb *
public class ProductionTransactionFactory implements TransactionFactory {
    3 usages  serrb *
    @Override
    public Transaction createTransaction(String type,
                                         Double amount,
                                         String username,
                                         Material material,
                                         Product product) {
        return new Transaction.Builder()
                .setType("PRODUCTION")
                .setProduct(product)
                .setAmount(amount)
                .setDateTime(LocalDateTime.now())
                .setUsername(username)
                .build();
    }
}
```

**ProductionTransactionFactory** - транзакції, пов'язані з виробництвом

```java
package com.example.warehouse.factories.transaction;

import ...

no usages   serrb *
public class ReceiveMaterialTransactionFactory implements TransactionFactory {
    3 usages   serrb *
    @Override
    public Transaction createTransaction(String type,
                                         Double amount,
                                         String username,
                                         Material material,
                                         Product product) {
        return new Transaction.Builder()
                .setType("RECEIVE")
                .setMaterial(material)
                .setAmount(amount)
                .setDateTime(LocalDateTime.now())
                .setUsername(username)
                .build();
    }
}
```

**ReceiveMaterialTransactionFactory** - транзакції прийому матеріалів.

```java
public void issueProduct(Integer productId, Double amount, UserResponse currentUser) {
    Optional<Product> optionalProduct = productRepository.findById(productId);

    if (optionalProduct.isEmpty()) {
        throw new RuntimeException("Product with ID " + productId + " not found");
    }

    Product product = optionalProduct.get();

    //=== Do we have enough product ?===
    if (product.getAmount() < amount) {
        throw new RuntimeException("Not enough product in stock");
    }

    //===Logic of taking products out of warehouse==
    Double newAmount = product.getAmount() - amount;
    product.setAmount(newAmount);

    productRepository.save(product);

    Transaction transaction = transactionFactory.createTransaction( type: "ISSUE",
            amount, currentUser.getName(),  material: null, product);
    transactionRepository.save(transaction);
}
```

Приклад використання у сервісах

| | id | type | material_id | product_id | amount | date_time | username |
|---|---|---|---|---|---|---|---|
| 1 | 1 | RECEIVE | 1 | <null> | 45 | 2023-12-25 07:46:50.783643 | Sofia |
| 2 | 3 | RECEIVE | 2 | <null> | 100 | 2023-12-25 08:09:54.964044 | Sasha |
| 3 | 5 | RECEIVE | 1 | <null> | 25 | 2023-12-30 22:11:04.477726 | Sofia |
| 4 | 6 | RECEIVE | 1 | <null> | 25 | 2023-12-30 22:13:21.631382 | Sofia |
| 5 | 7 | RECEIVE | 1 | <null> | 25 | 2023-12-30 22:16:51.679970 | Sofia |
| 6 | 8 | RECEIVE | 1 | <null> | 25 | 2023-12-30 22:20:49.344002 | Sofia |
| 7 | 11 | RECEIVE | 3 | <null> | 500 | 2024-01-02 12:43:09.510161 | Sofia |
| 8 | 13 | PRODUCTION | <null> | 11 | 2 | 2024-01-05 07:59:56.705680 | Sofia |
| 9 | 14 | PRODUCTION | <null> | 10 | 5 | 2024-01-05 08:04:59.860628 | Sofia |
| 10 | 16 | PRODUCTION | <null> | 10 | 2 | 2024-01-05 17:49:36.637586 | Sofia |
| 11 | 17 | PRODUCTION | <null> | 13 | 3 | 2024-01-05 22:08:40.961675 | Sofia |
| 12 | 18 | PRODUCTION | <null> | 13 | 2 | 2024-01-05 22:12:17.844679 | Sofia |
| 13 | 19 | PRODUCTION | <null> | 14 | 2 | 2024-01-05 22:12:37.473017 | Sofia |
| 14 | 20 | RECEIVE | 1 | <null> | 200 | 2024-01-05 22:14:55.299604 | Sofia |
| 15 | 21 | ISSUE | <null> | 10 | 2 | 2024-01-05 22:15:23.042054 | Sofia |
| 16 | 22 | RECEIVE | <null> | 11 | 1 | 2024-01-06 05:48:44.200813 | Sofia |
| 17 | 23 | PRODUCTION | <null> | 13 | 1 | 2024-01-06 16:26:47.208410 | Sofia |
| 18 | 24 | PRODUCTION | <null> | 10 | 2 | 2024-01-06 16:27:20.347652 | Sofia |
| 19 | 25 | PRODUCTION | <null> | 14 | 1 | 2024-01-07 00:14:52.632969 | Sofia |
| 20 | 26 | PRODUCTION | <null> | 10 | 2 | 2024-01-07 00:16:06.214312 | Sofia |
| 21 | 27 | PRODUCTION | <null> | 10 | 1 | 2024-01-07 00:17:02.955724 | Sofia |
| 22 | 28 | PRODUCTION | <null> | <null> | 100 | 2024-01-07 00:17:34.206647 | Sofia |
| 23 | 29 | PRODUCTION | <null> | <null> | 100 | 2024-01-07 00:17:54.224497 | Sofia |

Результат заносення Транзакцій у БД

**Висновок:** на цій лабораторній роботі я познайомився з такими паттернами, як «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator», а також розібрався у принципі їх роботи та сфері використання. На практиці спроектував та реалізував паттерн «Factory method» у своєму проекті.