Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №6
ТЕХНОЛОГІЇ РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Виконав
студент групи ІА–13:
Лапушенко А.К.

Київ 2023

Тема: *Шаблон фабричний метод "Factory Method".*
Варіант: POS(Point-of-sales)-system.

Хід роботи

Шаблон "фабричний метод" визначає інтерфейс для створення об'єктів певного базового типу. Це зручно, коли хочеться додати можливість створення об'єктів не базового типу, а деякого дочірнього. Фабричний метод у такому разі є зачіпкою для впровадження власного конструктора об'єктів. Основна ідея полягає саме в заміні об'єктів їх підтипами, що при цьому зберігає ту ж функціональність; інша частина поведінки об'єктів не є інтерфейсною (AnOperation) і дозволяє взаємодіяти із створеними об'єктами як з об'єктами базового типу.

В даній роботі шаблон "Factory Method" був використаний для реалізації комунікації між офіціантом та різними відділами кухонного процесу. Кожен відділ реалізує один інтерфейс, в якому є метод "показати повідомлення". Реалізація цього метода для різних відділів буде своєю, але суть одна і та ж сама – доставити повідомлення, яке складається зі страв, які треба приготувати, та їх кількості.

Інтерфейс

```
public interface Bar {
    void showMessage(OrderMenuItems item);
}
```

Відділи, які імплементують інтерфейс

```
public class DrinkBar implements Bar{
    @Override
    public void showMessage(OrderMenuItems item) {
        System.out.println("Shows on monitor");
        System.out.println(item.getMenuItem() + " : "+item.getQuantity());
    }
}
```

```
public class Kitchen implements Bar{
    @Override
    public void showMessage(OrderMenuItems item) {
        System.out.println("Shows on big monitor");
        System.out.println(item.getMenuItem() + " : "+item.getQuantity());
    }
}
```

```java
public class SushiBar implements Bar{
    @Override
    public void showMessage(OrderMenuItems item) {
        System.out.println("Printed on paper");
        System.out.println(item.getMenuItem() + " : "+item.getQuantity());
    }
}
```

Фабрика

```java
public class BarFactory {
    public Bar getBar(MenuItem menuItem){
        String name = menuItem.getName();
        if(contains(SushiItems.class, name)){
            return new SushiBar();
        }
        if(contains(KitchenItems.class, name)){
            return new Kitchen();
        }
        if(contains(DrinkItems.class, name)){
            return new DrinkBar();
        }
        return null;
    }

    private boolean contains(Class e, String name){
        return EnumUtils.findEnumInsensitiveCase(e, name) != null;
    }
```

Висновок

В цій роботі я навчився користуватись та використав паттерн “Factory method” у своєму застосунку, який сприяє розширенню та зміненню функцій комунікації офіціанта з іншими відділами кухонного процесу.

Контрольні питання

1. *Розкажіть про SOLID принципи проектування.*

1. Single Responsibility Principle (Принцип єдиної відповідальності).
2. Open Closed Principle (Принцип відкритості/закритості).
3. Liskov's Substitution Principle (Принцип підстановки Барбари Лисков).
4. Interface Segregation Principle (Принцип розподілу інтерфейсу).
5. Dependency Inversion Principle (Принцип інверсії залежностей).

2. *Навіщо використовується шаблон "фабрика" ?*

Для створення об'єктів певного базового типу. Це зручно, коли хочеться додати можливість створення об'єктів не базового типу, а деякого дочірнього. Фабричний метод у такому разі є зачіпкою для впровадження власного конструктора об'єктів. Основна ідея полягає саме в заміні об'єктів їх підтипами, що при цьому зберігає ту ж функціональність; інша частина поведінки об'єктів не є інтерфейсною (AnOperation) і дозволяє взаємодіяти із створеними об'єктами як з об'єктами базового типу.

3. *Чим відрізняється шаблон "фабрика" від "фабричного методу" ?*

   Фабрика має методи для кожного об’єкту. Фабричний метод має один метод, який повертає загальний інтерфейс.
   Наприклад, фабрика має методи “повернути апельсин” та “повернути яблуко”, які повертають об’єкти, а фабричний метод – “повернути фрукт” та від вхідних даних обирає, який фрукт повернути чи яблуко, чи апельсин, ці об’єкти реалізують один інтерфейс “фрукт”.

4. *Яку функціональність додає шаблон "оглядач"?*

   Коли один об'єкт змінює власний стан, усі інші об'єкти отримують про це сповіщення і мають можливість змінити власний стан також.

5. *Які обмеження використання шаблону "декоратор" ?*

   Важко конфігурувати об’єкти, які загорнуто в декілька обгорток одночасно.