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 №4
ТЕХНОЛОГІЇ РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

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

Київ 2023

Тема: *Шаблон «strategy».*
Варіант: POS(Point-of-sales)-system.

Хід роботи

Шаблон «Strategy» (Стратегія) дозволяє змінювати деякий алгоритм поведінки об'єкта іншим алгоритмом, що досягає ту ж мету іншим способом.
В даній роботі шаблон «Strategy» був використаний для реалізації оплати замовлення. Користувач може вказати два варіанти оплати:

1. Готівкою. Система не може взаємодіяти з фізичними грошима, тому при вказанні цього пункту замовлення стає автоматично оплаченим. Усі грошові операції обробляються користувачем без взаємодії із застосунком.
2. Банківською картою. При вказанні клієнтом цього пункту надсилається запит до POS-терміналу, який підключений до системи, із вказанням суми яку треба оплатити. Якщо оплата не проходить або операція відміняється на терміналі, то статус замовлення не змінюється.

Після того як статус замовлення змінюється на “Сплачено”, тоді, і тільки тоді, його можна буде закрити, після чого він зникає з головної сторінки користувача та зберігається у базі даних.
В даній роботі, не було реалізовано підключення до POS-терміналу, тому що в загальному доступі немає інформації про їх підключення та немає фізичного пристрою за для тестування. Замість цього є взаємодія з терміналом застосунку. Є три варіанти роботи з ним:

1. Написати ‘p’ – payed, сплачено. Тобто транзакція пройшла успішно. Статус замовлення змінюється на “Сплачено”.
2. Написати ‘c’ – canceled, відмінено. Тобто транзакція була відмінена. Статус замовлення не змінюється.
3. Якщо написати будь-що інше, тоді система відповість, щоб користувач спробував знову, тому що виникла помилка. Це буде відбуватися поки не буде виконано 1-ий або 2-ий варіант.

```
public interface PayStrategy {
    boolean pay(double paymentAmount);
}
```

```java
public class PayCash implements PayStrategy{

    @Override
    public boolean pay(double paymentAmount) {
        return true;
    }
}
```

```java
public class PayCard implements PayStrategy{

    @Override
    public boolean pay(double paymentAmount) {
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        boolean isPayed = false;
        try {
            boolean isDone = false;
            while(!isDone){
                String s = reader.readLine();
                switch (s) {
                    case "c" -> {
                        System.out.println("Canceled");
                        isDone = true;
                    }
                    case "p" -> {
                        System.out.println("Payed");
                        isPayed = true;
                        isDone = true;
                    }
                    default -> {
                        System.out.println("Something goes wrong");
                        System.out.println("Try again");
                    }
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return isPayed;
    }
}
```

Висновок

В цій роботі я навчився користуватись та використав паттерн “Strategy” у своєму застосунку, який сприяє легкому розширенню в майбутньому функції оплати.