Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №5

по дисципліні «Технології розроблення програмного забезпечення»

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

Виконав:
студент групи ІА-01
Мартюк М.К.

Перевірив:
вик. кафедри ІСТ
Колеснік В. М.

Дата здачі: 03.11.2022

Захищено з балом _____

Київ 2022

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 їх взаємодій для досягнення конкретних функціональних можливостей.

3. Застосування одного з розглянутих шаблонів при реалізації програми.Хід роботи:

2. Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.

Реалізація:

```java
import lombok.*;

import java.time.LocalDateTime;

@Data
//@Builder
@NoArgsConstructor
@AllArgsConstructor
public class Order {

    private Integer id;
    private LocalDateTime start;
    private LocalDateTime finish;
    private OrderStatus status;
    private Integer clientProfileId;
    private Integer barberProfileId;
    private Integer barbershopId;

    public Order(OrderBuilder orderBuilder) {
        this.id = orderBuilder.id;
        this.barberProfileId = orderBuilder.barberProfileId;
        this.start = orderBuilder.start;
        this.finish = orderBuilder.finish;
        this.status = orderBuilder.status;
        this.barbershopId = orderBuilder.barbershopId;
        this.clientProfileId = orderBuilder.clientProfileId;
    }

    public static OrderBuilder builder() {
        return new OrderBuilder();
    }

    @NoArgsConstructor(access = AccessLevel.PRIVATE)
    public static class OrderBuilder {
        private Integer id;
```

```
        private LocalDateTime start;
        private LocalDateTime finish;
        private OrderStatus status;
        private Integer clientProfileId;
        private Integer barberProfileId;
        private Integer barbershopId;

        public OrderBuilder id(Integer id) {
            this.id = id;
            return this;
        }

        public OrderBuilder start(LocalDateTime start) {
            this.start = start;
            return this;
        }

        public OrderBuilder finish(LocalDateTime finish) {
            this.finish = finish;
            return this;
        }

        public OrderBuilder status(OrderStatus status) {
            this.status = status;
            return this;
        }

        public OrderBuilder clientProfileId(Integer clientProfileId) {
            this.clientProfileId = clientProfileId;
            return this;
        }

        public OrderBuilder barberProfileId(Integer barberProfileId) {
            this.barberProfileId = barberProfileId;
            return this;
        }

        public OrderBuilder barbershopId(Integer barbershopId) {
            this.barbershopId = barbershopId;
            return this;
        }

        public Order build() {
            return new Order(this);
        }
    }

}
```

Використання:

```
import com.example.hairsalon.entity.Order;
import com.example.hairsalon.entity.OrderStatus;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.stereotype.Component;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.time.LocalDateTime;
```

```java
@Component
public class OrderRowMapper implements RowMapper<Order> {

    @Override
    public Order mapRow(ResultSet rs, int rowNum) throws SQLException {
        Integer orderId = rs.getObject("id", Integer.class);
        LocalDateTime start = rs.getTimestamp("start").toLocalDateTime();
        LocalDateTime finish = rs.getTimestamp("finish").toLocalDateTime();
        OrderStatus orderStatus = OrderStatus.valueOf(rs.getString("status"));
        Integer clientId = rs.getObject("client_id", Integer.class);
        Integer barberId = rs.getObject("barber_id", Integer.class);
        Integer barbershopId = rs.getObject("barbershop_id", Integer.class);

        return Order.builder()
                .id(orderId)
                .start(start)
                .finish(finish)
                .status(orderStatus)
                .clientProfileId(clientId)
                .barberProfileId(barberId)
                .barbershopId(barbershopId)
                .build();
    }
}
```

Висновок: на цій лабораторній роботі було реалізовано частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей, а також застосовано один з розглянутих шаблонів для реалізації програми.