**Лабораторная работа №76-77**

**«Использование поведенческих шаблонов»**

**Цель работы:** Научиться использовать поведенческие шаблоны.

**Задания для практического занятия:**

Используя поведенческий шаблон, выполните следующие задания:

1. Разработайте программу для обработки заказов, где от вас требуется принять заказ с товаром и совершить ряд проверок: Проверка наличия товара, проверка платежа, проверка адреса доставки. Каждый этап при этом должен быть отдельным обработчиком.

class Program

{

static void Main(string[] args)

{

OrderHandler productHandler = new ProductAvailabilityHandler();

OrderHandler paymentHandler = new PaymentVerificationHandler();

OrderHandler addressHandler = new DeliveryAddressHandler();

productHandler.SetNext(paymentHandler);

paymentHandler.SetNext(addressHandler);

var order = new Order(true, true, true);

productHandler.Handle(order);

}

}

public abstract class OrderHandler

{

protected OrderHandler nextHandler;

public void SetNext(OrderHandler handler)

{

nextHandler = handler;

}

public virtual void Handle(Order order)

{

nextHandler?.Handle(order);

}

}

public class ProductAvailabilityHandler : OrderHandler

{

public override void Handle(Order order)

{

if (order.ProductAvailable)

{

Console.WriteLine("Товар доступен для заказа.");

base.Handle(order);

}

else

{

Console.WriteLine("Товар недоступен.");

}

}

}

public class PaymentVerificationHandler : OrderHandler

{

public override void Handle(Order order)

{

if (order.PaymentSuccessful)

{

Console.WriteLine("Платеж подтвержден.");

base.Handle(order);

}

else

{

Console.WriteLine("Ошибка платежа.");

}

}

}

public class DeliveryAddressHandler : OrderHandler

{

public override void Handle(Order order)

{

if (order.AddressValid)

{

Console.WriteLine("Адрес доставки корректен.");

base.Handle(order);

}

else

{

Console.WriteLine("Некорректный адрес доставки.");

}

}

}

public class Order

{

public bool ProductAvailable { get; set; }

public bool PaymentSuccessful { get; set; }

public bool AddressValid { get; set; }

public Order(bool productAvailable, bool paymentSuccessful, bool addressValid)

{

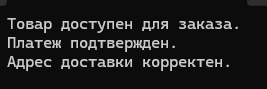
ProductAvailable = productAvailable;

PaymentSuccessful = paymentSuccessful;

AddressValid = addressValid;

}

}



**Контрольные вопросы**

**1.В чем различие поведенческих паттернов от порождающих и структурных?**

Поведенческие паттерны фокусируются на взаимодействии и ответственности объектов, упрощая управление коммуникацией между ними. Порождающие паттерны занимаются созданием объектов, а структурные паттерны определяют способ упорядочивания и организации объектов.

**2. Назовите основные принципы на которых строятся поведенческие паттерны.**

Основные принципы: инкапсуляция поведения, использование интерфейсов, делегирование ответственности, уменьшение связности между классами через четко определенные контракты.

**3. Как поведенческие паттерны помогают уменьшить связность между классами?**

Они позволяют объектам взаимодействовать через интерфейсы или абстракции, что делает их независимыми друг от друга, упрощая модификацию и тестирование кода.

**4. В чем суть паттерна стратегия?**

Паттерн стратегия определяет набор алгоритмов, инкапсулирует каждый из них и делает их взаимозаменяемыми, позволяя выбирать алгоритм динамически.

**5. В чем суть паттерна наблюдатель?**

Паттерн наблюдатель определяет зависимость "один ко многим", позволяя наблюдателям автоматически обновляться при изменении состояния субъекта.

**6. В чем суть паттерна команда?**

Паттерн команда инкапсулирует запрос как объект, позволяя сохранять, управлять и вызывать его позже, что упрощает выполнение операций.

**7. В чем суть паттерна цепочка обязанностей?**

Паттерн цепочка обязанностей позволяет передавать запрос по цепочке обработчиков, пока один из них не обработает его, что снижает связанность.

8. **Чем команда отличается от стратегии?**

Команда фокусируется на инкапсуляции запросов и выполнение действий, а стратегия определяет алгоритм выполнения задачи, давая возможность выбора.

9**. В чем разница между состоянием и стратегией?**

Состояние управляет поведением объекта в зависимости от его внутреннего состояния, а стратегия задает алгоритм выполнения задачи, не связанный со состоянием объекта.

**10. Когда прерывают цепочку обязанностей?**

Цепочку прерывают, когда запрос успешно обработан одним из обработчиков или когда ни один из них не смог его обработать, что может требовать обработки ошибок.

**Вывод:** Научился использовать поведенческие шаблоны.