**Курс:** Шаблоны проектирования приложений

**Тема:** Модуль 08 Паттерны поведения. Команда. Шаблонный метод. Посредник

**Цель:**

Изучить и реализовать паттерн **Команда (Command)** на языке C#. Ваша задача — разработать систему управления умным домом, где каждое действие (например, включение/выключение света, открытие/закрытие дверей) будет представлено в виде команды.

**Описание задачи:**

Вы будете разрабатывать приложение для управления различными устройствами в умном доме. Ваша задача — реализовать набор команд для управления этими устройствами, используя паттерн **Команда (Command)**. Команды будут инкапсулировать действия, такие как включение/выключение света, открытие/закрытие дверей и регулирование температуры. Программа должна поддерживать отмену последних команд.

**Структура программы:**

1. **Интерфейс ICommand** — общий интерфейс для всех команд.
2. **Реализации команд**:
   * Включение/выключение света.
   * Открытие/закрытие дверей.
   * Увеличение/уменьшение температуры.
3. **Класс Receiver** — устройство, выполняющее команду (например, свет, дверь или кондиционер).
4. **Класс Invoker** — объект, который вызывает команды и управляет отменой команд.
5. **Клиентский код** — демонстрация работы паттерна Команда.

**Шаги выполнения:**

1. Создайте интерфейс ICommand для всех команд
2. Создайте классы устройств Light, Door, Thermostat, которые будут выполнять команды
3. Реализуйте конкретные команды для управления устройствами
   1. **Команда для управления светом**
   2. Команда для управления дверью
   3. Команды для управления температурой
4. Реализуйте класс **Invoker**, который будет запускать команды и хранить историю для отмены
5. Напишите клиентский код для демонстрации работы программы

**Задания:**

1. Реализуйте приведенный выше код.
2. **Тестирование**:
   * Проверьте работу команд для всех устройств (свет, двери, термостат).
   * Убедитесь, что функция отмены работает корректно.
3. **Расширение функционала**:
   * Добавьте новый тип команды, например, включение/выключение телевизора или активация сигнализации.
   * Реализуйте возможность хранения истории нескольких последних команд и их отмены.
4. **Обработка ошибок**:
   * Добавьте обработку случаев, когда пытаются отменить команду, которая еще не была выполнена.

**Вопросы для самопроверки:**

1. Какие преимущества дает использование паттерна "Команда" в данном проекте?
2. Как можно добавить новые команды, не изменяя существующий код?
3. В чем заключается основное отличие паттерна "Команда" от простого вызова методов напрямую?

**Цель:**

Изучить и реализовать паттерн **Шаблонный метод (Template Method)** на языке C#. Ваша задача — разработать систему приготовления различных напитков (например, чая, кофе), где каждый напиток готовится по определенному шаблону, но с возможностью добавления уникальных шагов для каждого напитка.

**Описание задачи:**

Вы будете разрабатывать систему, в которой процесс приготовления напитков, таких как чай и кофе, будет разделен на несколько шагов. Большинство шагов будут общими для всех напитков, но некоторые — уникальными для каждого типа напитка. Программа должна использовать паттерн **Шаблонный метод**, чтобы определить общие шаги в базовом классе и позволить подклассам переопределять уникальные шаги.

**Структура программы:**

1. **Абстрактный класс Beverage** — содержит шаблонный метод для приготовления напитка и определяет последовательность шагов.
2. **Конкретные классы Tea и Coffee** — реализуют уникальные шаги для приготовления чая и кофе.
3. **Клиентский код** — демонстрирует процесс приготовления различных напитков.

**Шаги выполнения:**

1. Создайте абстрактный класс Beverage, который содержит шаблонный метод для приготовления напитка
2. Реализуйте конкретный класс **Tea**, который переопределяет уникальные шаги для чая
3. Реализуйте конкретный класс **Coffee**, который переопределяет уникальные шаги для кофе
4. Напишите клиентский код для демонстрации работы программы

**Задания:**

1. Реализуйте код по шагам выше.
2. **Тестирование**:
   * Проверьте процесс приготовления чая и кофе.
   * Убедитесь, что для кофе пользователь может отказаться от добавок.
3. **Расширение функционала**:
   * Добавьте новый напиток, например, **горячий шоколад**, с уникальными шагами приготовления.
   * Внедрите возможность изменять последовательность шагов для некоторых напитков, если требуется.
4. **Обработка ошибок**:
   * Добавьте проверку пользовательского ввода в методе CustomerWantsCondiments, чтобы обрабатывать некорректные ответы.

**Вопросы для самопроверки:**

1. Какие преимущества дает использование паттерна "Шаблонный метод" в данном проекте?
2. Как можно добавить новые напитки, не изменяя базовый класс?
3. В чем отличие перехватываемого метода (hook) от обычного абстрактного метода, и в каких случаях его использование уместно?

**Цель:**

Изучить и реализовать паттерн **Посредник (Mediator)** на языке C#. Ваша задача — разработать систему чата, где различные участники могут взаимодействовать друг с другом через посредника (чат-комнату), а не напрямую.

**Описание задачи:**

В рамках данного задания вы создадите чат-комнату, в которой несколько участников могут отправлять сообщения друг другу. Вместо того чтобы каждый участник отправлял сообщение напрямую другому участнику, он будет отправлять его через посредника (чат-комнату). Чат-комната будет управлять доставкой сообщений участникам.

**Структура программы:**

1. **Интерфейс IMediator** — интерфейс для посредника, который определяет методы для отправки и получения сообщений.
2. **Класс ChatRoom (конкретный посредник)** — реализует логику взаимодействия участников через чат.
3. **Класс User (участники чата)** — участники, которые взаимодействуют через посредника.
4. **Клиентский код** — демонстрация работы паттерна Посредник.

**Шаги выполнения:**

1. Создайте интерфейс IMediator, который будет определять взаимодействие между участниками
2. Реализуйте класс **ChatRoom**, который будет посредником между пользователями
3. Создайте класс **User**, который будет представлять участников чата
4. Реализуйте конкретные классы пользователей чата (например, обычный пользователь)
5. Напишите клиентский код для демонстрации работы программы

**Задания:**

1. Реализуйте код по шагам выше.
2. **Тестирование**:
   * Проверьте процесс отправки и получения сообщений между несколькими пользователями.
   * Убедитесь, что сообщение отправляется всем пользователям, кроме отправителя.
3. **Расширение функционала**:
   * Добавьте возможность отправки личных сообщений (только одному участнику).
   * Добавьте систему уведомлений о том, что новый пользователь присоединился к чату или покинул его.
4. **Обработка ошибок**:
   * Добавьте обработку ошибок, если пользователь пытается отправить сообщение, не будучи частью чата.

**Вопросы для самопроверки:**

1. Какие преимущества дает использование паттерна "Посредник" в данном проекте?
2. Как можно добавить новые типы участников (например, администратора чата), не изменяя существующий код?
3. Как изменить посредника, чтобы поддерживались групповые или личные сообщения между пользователями?