**Курс:** Шаблоны проектирования приложений

**Тема:** Модуль 07 Паттерны поведения. Стратегия. Наблюдатель

*Баллы:*

**Задача:**

Создать сложное приложение, которое симулирует работу системы бронирования путешествий. В системе пользователи могут выбирать различные типы транспорта (самолет, поезд, автобус и т.д.), а также различные методы расчета стоимости поездки в зависимости от множества факторов (расстояние, класс обслуживания, тип транспорта и дополнительные услуги).

**Описание:**

* **Контекст**: Система бронирования путешествий, которая может использовать различные стратегии расчета стоимости поездки.
* **Стратегии**: Каждая стратегия будет представлять собой алгоритм для расчета стоимости поездки с учетом различных факторов (расстояние, тип транспорта, класс обслуживания и другие).
* **Дополнительные факторы**: В задачу вводятся дополнительные требования, такие как класс обслуживания (эконом, бизнес), наличие скидок, скидки для детей или пенсионеров, количество пассажиров и пр.

**Цели:**

1. Построить гибкую систему, в которой можно добавлять новые алгоритмы расчета стоимости поездки (стратегии) без изменения основного кода.
2. Организовать архитектуру таким образом, чтобы клиент мог легко менять стратегии на лету.
3. Поддержать дополнительные факторы, влияющие на расчет стоимости поездки (скидки, дополнительные услуги и т.д.).

**Шаги выполнения:**

1. **Создайте интерфейс** ICostCalculationStrategy, который будет представлять стратегию расчета стоимости поездки.
2. **Создайте несколько классов-стратегий**, которые реализуют интерфейс ICostCalculationStrategy. Каждая стратегия будет рассчитывать стоимость в зависимости от типа транспорта: самолет, поезд, автобус и т.д.
3. **Создайте класс контекста** TravelBookingContext, который будет использовать различные стратегии для расчета стоимости поездки в зависимости от выбора пользователя.
4. **Добавьте возможность выбора дополнительных факторов**, влияющих на стоимость, таких как скидки (например, скидка для детей или пенсионеров) и класс обслуживания (эконом, бизнес).
5. **Напишите клиентский код**, который будет взаимодействовать с контекстом, позволяя пользователю выбрать тип транспорта, ввести необходимые данные и рассчитать стоимость поездки.

**Дополнительные условия:**

1. **Расширяемость**: Добавьте новые виды транспорта или расширьте существующие стратегии с учетом дополнительных факторов (например, учитывайте топливо, налоги, региональные коэффициенты).
2. **Обработка ошибок**: Включите валидацию пользовательских данных и обработку ошибок (например, неверные входные значения, отсутствие стратегии).
3. **Расширение на более сложные факторы**: Добавьте возможность расчета для более сложных сценариев, таких как наличие багажа, бронирование групповых поездок или поездок с пересадками.

**Требования:**

* В коде должно быть минимум 3 различных стратегии.
* Клиент должен иметь возможность настраивать контекст для выбора подходящей стратегии.
* Необходимо учитывать дополнительные факторы, такие как скидки, класс обслуживания и количество пассажиров.
* Программа должна быть гибкой и расширяемой для добавления новых стратегий.

**Задача:**

Необходимо разработать систему управления биржевыми торгами на C#, которая использует паттерн **Наблюдатель (Observer)**. Система должна позволять различным пользователям подписываться на обновления цен акций и получать уведомления при изменении стоимости акций в режиме реального времени. Ваша задача — реализовать гибкую архитектуру, позволяющую добавлять новых наблюдателей (например, торговых роботов, уведомления по электронной почте, мобильные приложения) и легко управлять подписками.

**Описание:**

1. **Субъект (Биржа)**: Система управления биржевыми торгами, которая хранит информацию о различных акциях и изменениях их цен. При изменении цены акции биржа уведомляет всех подписчиков об изменении.
2. **Наблюдатели (Трейдеры, Роботы и т.д.)**: Это объекты, которые подписаны на изменение цен акций. Каждый наблюдатель будет реагировать по-своему: один может просто выводить информацию, другой может совершать действия на основе определенных условий (например, автоматически покупать или продавать акции).
3. **Клиентский код**: Клиент должен иметь возможность добавлять и удалять наблюдателей на лету, а также подписывать каждого наблюдателя только на интересующие его акции.

**Цели:**

1. Построить систему, которая позволяет подписчикам (наблюдателям) подписываться на изменения цен конкретных акций.
2. Обеспечить возможность гибкого добавления и удаления наблюдателей на лету.
3. Разработать стратегию, по которой биржа может одновременно обрабатывать множество акций и уведомлять наблюдателей только о тех акциях, на которые они подписаны.
4. Предусмотреть возможность расширения системы, добавив новых наблюдателей с различными сценариями обработки данных.

**Шаги выполнения:**

1. Создайте интерфейс IObserver для наблюдателей
2. Создайте интерфейс ISubject для субъекта (Биржа)
3. Реализуйте класс StockExchange, который будет играть роль субъекта и хранить информацию о различных акциях
4. Создайте несколько классов-наблюдателей:
   1. Трейдер, который реагирует на изменение цены акции, выводя информацию на экран.
   2. Автоматический торговый робот, который автоматически покупает или продает акции в зависимости от определенных условий (например, цена выше или ниже определенного значения).
5. Реализуйте клиентский код, который будет имитировать биржевые торги и взаимодействовать с наблюдателями

**Дополнительные условия:**

1. **Фильтрация уведомлений**: Добавьте возможность для наблюдателей фильтровать уведомления (например, чтобы они получали уведомления только при превышении определенной цены).
2. **Асинхронные уведомления**: Добавьте возможность асинхронного уведомления наблюдателей, чтобы имитировать реальное время в биржевых торгах.
3. **Расширяемость системы**: Реализуйте поддержку подписки на несколько акций одновременно и возможность гибкого расширения системы новыми типами наблюдателей.
4. **Логирование**: Добавьте логирование событий, таких как добавление и удаление наблюдателей, а также изменения цен акций.

**Дополнительные факторы:**

1. **Отчеты по подписчикам**: Реализуйте возможность генерации отчетов, которые показывают, какие наблюдатели подписаны на те или иные акции и как часто они получают уведомления.
2. **Реальные сценарии**: Добавьте сценарии изменения цен акций в реальном времени с использованием событий (например, новостные события, выход финансовых отчетов).

**Требования:**

* Система должна поддерживать подписку наблюдателей на различные акции.
* При изменении цены акции система должна уведомлять только тех наблюдателей, которые подписаны на конкретную акцию.
* Наблюдатели должны получать данные о конкретной акции и ее новой цене.