

**Институт Автоматики и информационных технологий  
 Кафедра Программной инженерии**

**CSE5071 Шаблоны проектирования приложений**

**Билет №24**

ФИО студента: Рябинин Максим Вадимович

1. Вопрос №1.

Дайте описание 5 принципов проектирования SOLID:

- a. Single Responsibility Principle (Принцип единственной ответственности)  
Класс должен иметь только одну ответственность и, соответственно, только одну причину для изменения.  
Каждый модуль или компонент должен отвечать за одну часть функциональности системы.  
Это упрощает сопровождение, тестирование и понимание кода.  
Нарушение принципа приводит к сильной связанности и хрупкой архитектуре.
- b. Open/Closed Principle (Принцип открытости/закрытости)  
Программные сущности должны быть открыты для расширения, но закрыты для изменения.  
Добавление новой функциональности не должно требовать изменения существующего кода.  
Принцип обычно реализуется с помощью абстракций, наследования и полиморфизма.  
Это снижает риск внесения ошибок при развитии системы.
- c. Liskov Substitution Principle (Принцип подстановки Лисков)  
Объекты подклассов должны корректно заменять объекты базового класса без изменения поведения программы.  
Подкласс не должен усиливать предусловия или ослаблять постусловия базового класса.  
Нарушение принципа приводит к непредсказуемому поведению при использовании полиморфизма.  
Корректное соблюдение гарантирует надёжную и расширяемую иерархию классов.
- d. Interface Segregation Principle (Принцип разделения интерфейсов)  
Клиенты не должны зависеть от методов, которые они не используют.  
Лучше иметь несколько специализированных интерфейсов, чем один универсальный.

Это уменьшает связанность между компонентами системы.  
Принцип повышает гибкость и упрощает модификацию кода.

E.Dependency Inversion Principle (Принцип инверсии зависимостей)

Модули верхнего уровня не должны зависеть от модулей нижнего уровня — оба должны зависеть от абстракций.

Абстракции не должны зависеть от деталей реализации, детали должны зависеть от абстракций.

Принцип достигается через интерфейсы и внедрение зависимостей.

Он повышает тестируемость, расширяемость и устойчивость архитектуры системы.

2. Вопрос №2. Какой паттерн проектирования подходит для реализации описанного ниже технического задания

Observer

### 1. Описание проекта

Проект представляет собой систему управления биржевыми торговыми портфелем инвестора. Система должна уведомлять пользователей о различных событиях, связанных с их инвестициями и рынком (например, изменение цены акций, достижение заданных уровней для продажи/покупки, значительные изменения на рынке). Пользователи могут подписываться на интересующие их акции или другие ценные бумаги, чтобы оперативно получать информацию. Выбранный паттерн обеспечит централизованное управление подписками и автоматическое уведомление подписчиков о любых изменениях.

### 2. Цель и задачи

Цель проекта — создать систему для управления портфелем и мониторинга состояния рынка, в которой пользователи смогут подписываться на события, касающиеся интересующих их активов, и получать своевременные уведомления о любых изменениях. Выбранный паттерн должен обеспечить эффективное управление подписками и доставку уведомлений.

#### Задачи проекта:

- Разработать систему для отслеживания состояния акций и других активов.
- Реализовать механизм подписки и отписки от уведомлений для каждого актива.
- Обеспечить возможность отправки уведомлений подписчикам при изменении цен и других условиях.
- Добавить адаптацию уведомлений в зависимости от типа пользователя и уровня его подписки.

### 3. Требования к функционалу

## 1. Классы

## событий

## **рынка**

Создать классы для различных типов событий, на которые пользователи могут подписаться:

- o PriceChangeEvent: событие изменения цены актива (акции, облигации).
  - o ThresholdEvent: событие достижения определенного ценового уровня для актива (например, цена акций достигла уровня для продажи).
  - o MarketAlertEvent: системное событие, указывающее на значительные изменения на рынке (например, падение индекса на более чем 5%).
  - o DividendEvent: уведомление о выплате дивидендов для акций, на которые подписан пользователь.

## 2. Класс

Создать интерфейс, который будет определять методы для обработки уведомлений:

- `Update(event)`: метод, вызываемый при возникновении события, на которое подписан. В метод передается информация о событии и активах.

### 3. Класс

субъекта

## (Subject)

Создать интерфейс `Subject`, который будет представлять активы, отслеживаемые системой, и управлять списком подписчиков:

- o Attach: метод для добавления подписчика на уведомления для актива.
  - o Detach: метод для удаления подписчика.
  - o Notify(event): метод для уведомления всех подписчиков об изменениях состояния актива.

#### **4. Механизм подписки и отписки от уведомлений**

Каждый пользователь должен иметь возможность подписаться на изменения интересующих его активов и отписаться при необходимости:

- о Возможность подписки на уведомления об изменении цены, достижении определенного уровня и системные оповещения по выбранным активам.
  - о Возможность отписаться от уведомлений для конкретных событий или активов.

## **5. Адаптация уведомлений для различных типов пользователей**

Реализовать систему адаптации уведомлений в зависимости от типа пользователя и уровня подписки:

- о Приватные инвесторы получают уведомления о достижении уровней продажи/покупки и изменении цен активов.

- о Профессиональные инвесторы получают уведомления о всех изменениях и сигналы о критических событиях на рынке.
- о Финансовые аналитики могут получать детализированные отчеты о состоянии рынка, выпусках дивидендов и значительных изменениях по ключевым активам.

## 6. Поддержка различных каналов уведомлений

Реализовать отправку уведомлений через разные каналы:

- о EmailNotification: отправка уведомлений по электронной почте.
- о SMSNotification: отправка уведомлений через SMS.
- о PushNotification: отправка уведомлений в приложение на мобильном устройстве.

## 4. Пример использования

1. Пользователь подписывается на уведомления о ценовых изменениях и достижении уровней покупки/продажи для конкретной акции.
2. Когда цена достигает установленного порога, система генерирует событие ThresholdEvent.
3. Все подписчики, которые подписаны на это событие, получают уведомление через выбранный канал (например, push-уведомление или SMS).
4. Пользователь может просмотреть историю уведомлений для своих активов, включая события дивидендов и значительных изменений на рынке.

## 5. Ожидаемый результат

Результатом выполнения задания станет гибкая система управления уведомлениями для биржевой платформы,строенная с использованием выбранного паттерна. Система позволит пользователям подписываться на интересующие их события для активов, получать уведомления через различные каналы и иметь доступ к истории событий и отчетов.