1. Что такое паттерн проектирования в контексте разработки программного обеспечения? Перечислите три основных типа паттернов проектирования и дайте краткое описание каждого. Как паттерны проектирования способствуют написанию легко поддерживаемого и масштабируемого кода?

Шаблон проектирования – повторяемая архитектурная конструкция, которая решает проблемы проектирования в рамках часто повторяющегося контекста.

Порождающие паттерны определяют способы создания объектов, чтобы система не была зависима от способов создания, композиции и представления объектов.

Структурные паттерны определяют способы композиции объектов для обеспечения нового функционала и управления их взаимодействиями.

Поведенческие паттерны определяются способы организации паттернов таким образом, чтобы упростить коммуникацию и управление взаимодействиями внутри системы.

Паттерны помогают тем, что улучшают читаемость кода и как следствие – его изменение. Также паттерны увеличивают переиспользуемость кода, что помогает убирать повторы. И наконец паттерны способствуют масштабированию благодаря своей гибкой архитектуре и возможности легко дописать код.

1. Опишите паттерн Singleton и сценарии, в которых его применение оправдано. Какие проблемы могут возникнуть при использовании Singleton в многопоточной среде? Приведите пример реализации Singleton в Java/Kotlin

Одиночка – порождающий паттерн проектирования, который гарантирует, что у класса есть 1 экземпляр и предоставляет глобальную точку доступа.

Сценарии, в которых использование одиночки оправдано – работа с общими ресурсами, например, база данных. Шаблон гарантирует наличие 1 экземпляра. Также может быть использован в кешировании или настройке приложения.

Использование singleton в многопоточной среде может вызвать несколько проблем. Во-первых, несколько потоков могут одновременно создать или получить экземпляр класса, что приведет к сбою в программе. Во-вторых, неоднозначность состояния – одиночка может изменяться несколькими потоками параллельно без должной синхронизации, что приведет к несогласованности состояний объекта и внесению недопустимых изменений.

1. Что такое паттерн Наблюдатель и в каких случаях его стоит применять? Какие преимущества дает использование паттерна Наблюдатель? Приведите пример, где этот паттерн может быть использован в приложении.

Наблюдатель — это поведенческий паттерн проектирования, который позволяет определить зависимость ”один ко многим” между объектами так, чтобы при изменении состояния одного объекта все зависящие от него объекты были оповещены и обновлены автоматически.

Преимущества: отделение наблюдателей от объекта, уменьшение связанности между компонентами системы, модульность и расширяемость, упрощение обновления состояний.

Пример: использоваться может, например, при отправке сообщений в мессенджерах – пользователь отправляет сообщение, все сообщения сдвигаются – используется модель наблюдатель. Или когда по нажатию кнопки генерируются новые блоки на web-странице.

1. Объясните назначение и принцип работы паттерна Адаптер. В чем преимущество использования Адаптера при интеграции сторонних библиотек или API? Приведите пример кода, где паттерн Адаптер помогает взаимодействовать двум несовместимым интерфейсам.

Адаптер — это структурный паттерн программирования, который позволяет объектам с несовместимыми интерфейсами работать вместе.

Преимущества: совместимость,