Макет



Ответы на Контрольные вопросы:

1. **В чем отличие предварительного проектирования от детального?**
   * **Предварительное проектирование** заключается в создании общей структуры системы, определении её архитектуры и базовых компонентов. На этом этапе разрабатываются основные модули и компоненты системы, их взаимосвязи, интерфейсы, а также выбираются архитектурные паттерны. Целью является создание базовой модели системы.
   * **Детальное проектирование** включает в себя проработку каждого отдельного компонента системы с высоким уровнем детализации. Здесь разрабатываются конкретные алгоритмы, интерфейсы между модулями, структуры данных, и прописываются все мелкие детали, необходимые для реализации системы. Детальное проектирование приводит к точной спецификации всех элементов системы.
2. **Перечислите и кратко опишите архитектурные системные паттерны:**
   * **Монолитная архитектура:** система реализована в виде одного большого блока кода, который управляет всеми аспектами приложения. Подходит для небольших приложений, но сложно масштабируется.
   * **Клиент-сервер:** система разделена на два основных компонента: клиент, который запрашивает данные, и сервер, который их предоставляет. Удобно для распределенных систем.
   * **Микросервисы:** система разбита на небольшие независимые сервисы, которые могут разрабатываться и разворачиваться автономно. Этот подход улучшает масштабируемость и гибкость.
   * **Слойная архитектура:** система делится на несколько логических уровней (например, уровень представления, бизнес-логика и база данных), что упрощает поддержку и развитие.
   * **Шинная архитектура (Event-Driven):** компоненты взаимодействуют друг с другом через сообщения, передаваемые по единой шине. Используется для систем с высокой нагрузкой и требующих масштабирования.
3. **Перечислите и кратко опишите паттерны управления:**
   * **Singleton (Одиночка):** обеспечивает, чтобы у класса был только один экземпляр и предоставляет к нему глобальную точку доступа.
   * **Factory Method (Фабричный метод):** определяет интерфейс для создания объектов, позволяя подклассам решать, какой класс создавать.
   * **Observer (Наблюдатель):** объект управляет списком наблюдателей и уведомляет их об изменениях своего состояния, используется для реализации реактивных систем.
   * **Strategy (Стратегия):** позволяет изменять поведение класса во время выполнения программы, выбирая одну из нескольких возможных стратегий (алгоритмов).
4. **Что такое «связность модуля»? Перечислите и кратко опишите типы связности модуля:**

**Связность модуля** (Cohesion) — это мера того, насколько сильно элементы внутри одного модуля связаны друг с другом. Чем выше связность, тем более сконцентрирован на одной задаче модуль, что считается хорошей практикой.

Типы связности:

* + **Функциональная связность:** элементы модуля вместе выполняют одну функцию — наивысшая степень связности.
  + **Последовательная связность:** выход одной функции служит входом для другой функции внутри модуля.
  + **Коммуникационная связность:** элементы используют одни и те же данные или структуру данных.
  + **Процедурная связность:** функции модуля выполняются последовательно, но не обязательно обрабатывают одни и те же данные.
  + **Временная связность:** элементы модуля должны выполняться в определенный момент времени, например, при инициализации.
  + **Логическая связность:** в модуле собраны функции, которые выполняют схожие задачи, но работают независимо.
  + **Случайная связность:** функции в модуле не имеют связи друг с другом — наименьшая степень связности.

1. **Что такое «сцепление модулей»? Перечислите и кратко опишите типы сцепления модулей:**

**Сцепление модулей** (Coupling) — это мера зависимости между модулями системы. Сильное сцепление затрудняет поддержку и изменение системы, тогда как слабое сцепление облегчает это.

Типы сцепления:

* + **Контентное сцепление:** один модуль зависит от внутренней реализации другого модуля — наихудший тип сцепления.
  + **Общее сцепление:** модули разделяют глобальные данные.
  + **Внешнее сцепление:** модули используют общий интерфейс или структуру данных.
  + **Контрольное сцепление:** один модуль управляет выполнением другого.
  + **Штучное сцепление (Message Coupling):** модули взаимодействуют через отправку сообщений или передачу данных — наиболее слабое сцепление.

1. **Опишите следующие фундаментальные паттерны проектирования:**
   * **Делегирование:** позволяет объекту передавать выполнение конкретной задачи другому объекту. Это помогает разделять ответственность между объектами.
   * **Неизменяемый объект (Immutable Object):** объект, состояние которого нельзя изменить после создания. Такие объекты повышают безопасность и упрощают многопоточную разработку.
   * **Интерфейс (Interface):** определяет контракт, который должны соблюдать классы, реализующие этот интерфейс, что помогает абстрагировать конкретные реализации.
   * **MVC (Model-View-Controller):** архитектурный паттерн, разделяющий логику приложения на три компонента: модель (работает с данными), представление (отображает данные) и контроллер (обрабатывает пользовательский ввод). Этот паттерн способствует разделению ответственности и упрощает поддержку приложения.