**Вопросы 10**

**1. 1 вопрос: Для чего используют многоуровневые архитектуры?**

Ответ: Многоуровневые архитектуры используются для создания более гибких, расширяемых и управляемых приложений путем разделения компонентов на различные уровни, каждый из которых отвечает за определенную функциональность.

**2. 2 вопрос: Опишите назначение слоев многоуровневой архитектуре. Business layer (уровень бизнес-логики), Data Access layer (уровень доступа к данным).**

Ответ: Уровень бизнес-логики, или Business layer, отвечает за обработку бизнес-логики и взаимодействие с уровнем доступа к данным, или Data Access layer, который обеспечивает доступ к данным из различных источников. Такое разделение компонентов позволяет увеличить гибкость и переносимость приложения.

**3. 3 вопрос: Поясните назначение и принцип использования паттерна Repository?**

Ответ: Паттерн Repository используется для абстрагирования доступа к данным от основного кода, что позволяет улучшить тестируемость и управляемость приложения. Он представляет собой слой между уровнем бизнес-логики и уровнем доступа к данным, который предоставляет единый интерфейс для работы с данными из различных источников.

**4. 4 вопрос: Поясните назначение и принцип использования паттерна Unit of Work?**

Ответ: Паттерн Unit of Work используется для объединения нескольких операций с данными в одну транзакцию, что позволяет обеспечить целостность данных и избежать ошибок при их обработке. Он представляет собой уровень абстракции над уровнем доступа к данным, который управляет жизненным циклом объектов и сохранением изменений.

**5. 5 вопрос: Опишите основные особенности библиотеки Entity Framework?**

Ответ: Библиотека Entity Framework представляет собой ORM-фреймворк, который позволяет упростить работу с базами данных. Она поддерживает работу с различными источниками данных, включая SQL Server, Oracle, MySQL и другие. Основными особенностями библиотеки являются легкость использования, возможность создания объектов модели данных, поддержка LINQ, механизмы отслеживания изменений и другие.

**6. 6 вопрос: Какие преимущества обеспечивает использование слабосвязанного кода?**

Ответ: Использование слабосвязанного кода позволяет упростить тестирование, повторное использование и изменение компонентов приложения. Это достигается путем разделения компонентов на отдельные модули, которые не зависят от других модулей или изменений в них. Это улучшает гибкость, управляемость и расширяемость приложения.

**Вопросы: 11**

**1. Что такое MVC, MVP, MVVM?**

MVC (Model-View-Controller), MVP (Model-View-Presenter), MVVM (Model-View-ViewModel) - это архитектурные шаблоны проектирования, используемые для разработки программных приложений.

**2. Объясните из каких компонентов состоит паттерн MVVM?**

Шаблон MVVM имеет три основных компонента: *модель*, которая  
представляет бизнес-логику приложения, *представление* пользовательского  
интерфейса XAML, *и представление-модель,* в котором содержится вся  
логика построения графического интерфейса и ссылка на модель, поэтому он  
выступает в качестве модели для представления

**3. Каким образом паттерн можно реализовать в WPF?**

- Определить классы для каждой части шаблона: модель (Model), представление (View) и представитель (ViewModel).

- Связать представление и представитель с помощью привязки данных (Data Binding). Для этого в XAML-коде нужно указать источники данных (поля и свойства представления, которые будут связаны с соответствующими свойствами представителя).

- Описать логику взаимодействия представления и представителя. Для этого можно использовать команды (ICommand) для связывания действий пользователя (например, щелчков мыши) с логикой обработки событий в представителе. Команды реализуются в представителе.

- Написать логику модели, которая определяет данные и бизнес-логику (например, методы для получения и сохранения данных). Модель не знает о представителе и представлении и не зависит от технологии, используемой в представлении.

**4. Объясните назначение интерфейса INotifyPropertyChanged?**

Интерфейс INotifyPropertyChanged определяет событие PropertyChanged, которое вызывается при изменении значения свойства. Когда свойство модели данных изменяется, оно вызывает событие PropertyChanged, уведомляя таким образом представление об изменении данных. Это позволяет представлению автоматически обновлять данные, связанные с этим свойством, так как значения свойства представления автоматически обновляются при изменении связанного свойства модели.

**5. Зачем нужен интерфейс ICommand и как паттерн MVVM использует  
команды в WPF-приложениях.**

Интерфейс ICommand определяет методы, необходимые для выполнения команд, связанных с действиями пользователя. Паттерн MVVM использует команды для связи логики и представления. Команды реализуются в представителе и выполняются в ответ на действия пользователя. Использование команд позволяет разделить логику действий пользователя (которая находится в представителе) от его реализации в представлении.

**6. Чем отличаются команды от событий?**

Главное отличие заключается в том, что события привязаны непосредственно к элементам интерфейса и обрабатываются в контексте элемента, в то время как команды являются независимыми от элементов и могут быть вызваны из различных частей приложения. Команды предлагают абстракцию действий, которая позволяет разделить логику выполнения команды от представления элемента интерфейса, обеспечивая более гибкое управление действиями в приложении.

**7. Расскажите о назначении и об основных возможностях следующих  
библиотек и фреймворков: Light Toolkit, Catel и Prism.**

Light Toolkit - это библиотека, которая содержит множество элементов управления, стилей и тем для создания пользовательского интерфейса в WPF. Она также предоставляет набор инструментов для работы с графикой, мультимедиа и привязкой данных.

Catel - это фреймворк для создания сложных приложений на WPF и Silverlight. Он предоставляет инструменты для управления жизненным циклом объекта, внедрения зависимостей, сериализации и валидации данных, также содержит много других возможностей для разработки высокопроизводительных приложений.

Prism - это фреймворк, который облегчает создание приложений, которые используют компонентную архитектуру и модель инверсии управления (IoC). Он содержит библиотеку инструментов для разработки гибких приложений, таких как привязка данных, обработка событий и управление модулями. Prism также предоставляет стандартные решения для повышения производительности и безопасности, а также для взаимодействия приложений между собой.

**Вопросы: 12**

**1. Что такое UML?**

Ответ: UML (Unified Modeling Language) - это стандартный язык моделирования, используемый для описания, визуализации, проектирования и документирования различных систем.

**2. Перечислите типы диаграмм UML.**

диаграммы классов,

деятельности,

последовательности,

состояний,

компонентов,

развертывания,

использования и пакетов.

**3. Какие отношения между классами могут быть на диаграмме классов?**

1. Наследование (inheritance) - отношение, в котором один класс наследует атрибуты и методы родительского класса. Обозначается сплошной линией с треугольником в сторону родительского класса.

2. Ассоциация (association) - отношение, которое связывает два класса и указывает на возможность взаимодействия между ними. Обозначается линией с указанием направления и типа ассоциации (один-к-одному, один-ко-многим, многие-ко-многим).

3. Агрегация (aggregation) - отношение, при котором один класс представляет собой контейнер, содержащий один или несколько экземпляров другого класса, которые могут существовать и без его контейнера. Обозначается линией с пустым ромбом в сторону класса-контейнера.

4. Композиция (composition) - отношение агрегации, при котором экземпляры класса-части могут существовать только внутри экземпляра класса-контейнера и при удалении экземпляра класса-контейнера все его экземпляры класса-частей также удаляются. Обозначается линией со залитым ромбом в сторону класса-контейнера.

5. Зависимость (dependency) - отношение, представляющее связь между двумя классами, при котором изменение одного класса влияет на другой. Обозначается пунктирной линией со стрелкой в сторону зависящего класса.

6. Реализация (realization) - отношение, в котором класс реализует (реализует методы) интерфейс, заданный другим классом. Обозначается пунктирной линией со стрелкой в сторону интерфейса и названием "реализует".

**4. Как обозначаются абстрактные классы на диаграмме классов?**

Ответ: Абстрактные классы на диаграмме классов обозначаются в виде класса со специальным украшением, например, ***курсивом*** или полужирным шрифтом.

**5. Как обозначаются интерфейсы на диаграмме классов?**

Интерфейсы на диаграмме классов обозначаются в виде пустых прямоугольников с названием интерфейса внутри.

Ответ: Интерфейсы на диаграмме классов обозначаются в виде класса с иконкой в виде множества пунктирных линий вокруг его границ.

**6. Как отображается доступность членов класса на диаграмме классов?**

Ответ: Доступность членов класса на диаграмме классов отображается в виде символов "+", "-", "#" или "~", что обозначает соответственно public (открытый), private (закрытый), protected (защищенный) и package (доступный внутри пакета).

**7. Что такое агрегация? Как обозначается?**

Ответ: Агрегация - это тип отношения между объектами, при котором один объект является составной частью другого объекта. На диаграмме это обозначается стрелкой с пустой головкой и на конце линии написано "part of".

**8. Что такое ассоциация?**

Ответ: Ассоциация - это отношение между объектами, при котором они могут взаимодействовать друг с другом. На диаграмме это обозначается линией, соединяющей объекты, и часто сопровождается описанием отношения.

**9. Какие обозначения используют на диаграмме последовательности?**

- актеры (люди, компьютерные системы, инструменты);

- объекты (предметы, которые передаются между актерами);

- сообщения (стрелки, которые показывают передачу объектов между актерами);

- активности (блоки, которые показывают действия, выполняемые актерами или объектами);

Ответ: На диаграмме последовательности используются обозначения для объектов, сообщений между объектами, фрагментов (например, условных операторов и циклов) и линий жизни объектов.

**10. Для чего нужна диаграмма последовательности?**

Ответ: Диаграмма последовательности используется для визуализации взаимодействия объектов в некоторой последовательности, например, в процессе выполнения какой-то операции или сценария. Она помогает разработчикам понять, какие объекты или актеры взаимодействуют между собой и как именно это происходит.

**11. Каково назначение диаграммы использования, пакетов и активности?**

Ответ: Диаграмма использования используется для описания взаимодействия между системой и ее пользователями или другими внешними системами.

Диаграмма пакетов - для описания структуры модели и связей между компонентами. Диаграмма активности используется для моделирования бизнес-процессов и последовательности действий, не обязательно в рамках компьютерных систем.