**Разбор сложных вопросов и подготовка к экзамену:**

**1) Являются ли интрфейсы и классы строительными блоками ЮМЛ. Как я понял нет, но не понял почему, и чем тогда они являются? Они же есть на схемах.**

**Или классы и интерфесы и это описание сущностей, который уже является строительным блоком?**

**ОТВЕТ ПРЕПОДА:**

**наверное, правильно вы написали - классы и интерфейсы это описание сущности. А сущность это уже строительный блок**

**Единственная схема где они выступают сами как блоки - это диаграмма классов**

**2)Что такое квантор(~) в диаграмме классов, надо ли про это знать?**

**ОТВЕТ ПРЕПОДА:**

**)~ пакетная видимость, в с++ нет специального спецификатора. Это можно решить используя пространства имен**

**3)каноническим в языке UML являются (8 шт):**

**-Диаграмма вариантов использования**

**-Диаграмма классов**

**-Диаграмма состояний**

**-Диаграмма деятельности**

**-Диаграмма последовательности**

**-Диаграмма кооперации**

**-Диаграмма компонентов**

**-Диаграмма развертывания**

**А что является не каноническим ??**

**-Диаграмма объектов**

**ОТВЕТ ПРЕПОДА:**

**4)Что значит кратность атрибута? Это что то типа одного значения на один атрибут, или только один атрибут вообще на весь класс, или вобще нечто иное.**

**ОТВЕТ ПРЕПОДА:**

**Кратность (multiplicity) — спецификация области значений допустимой мощности, которой могут обладать соответствующие множества.**

**Кратность атрибута характеризует общее количество конкретных атрибутов данного типа, входящих в состав отдельного класса. В общем случае кратность записывается в форме строки текста из цифр в квадратных скобках после имени соответствующего атрибута, при этом цифры разделяются двумя точками: [нижняя граница .. верхняя граница], где нижняя и верхняя границы положительные целые числа. Каждая такая пара служит для обозначения отдельного замкнутого интервала целых чисел, у которого нижняя (верхняя) граница равна значению нижней границы (верхней). В качестве верхней границы может использоваться специальный символ " \* " (звездочка), который означает произвольное положительное целое число, т.е. неограниченное сверху значение кратности соответствующего атрибута.**

**Интервалов кратности для отдельного атрибута может быть несколько. В этом случае их совместное использование соответствует теоретико-множественному объединению соответствующих интервалов. Значения кратности из интервала следуют в монотонно возрастающем порядке без пропуска отдельных чисел, лежащих между нижней и верхней границами. При этом придерживаются следующего правила: соответствующие нижние и верхние границы интервалов включаются в значение кратности.**

**Если в качестве кратности указывается единственное число, то кратность атрибута принимается равной данному числу. Если же указывается единственный знак " \* ", то это означает, что кратность атрибута может быть произвольным положительным целым числом или нулем. В языке UML кратность широко используется также для задания ролей ассоциаций, составных объектов и значений атрибутов. Если кратность атрибута не указана, то по умолчанию в языке UML принимается ее значение равное [1..1], т.е. в точности 1.**

**5)Различия Ассоциация и агрегация**

**Ассоциация - свзяь между двумя независимыми классами**

**Агрегация - это форма ассоциации где один независимый класс содержится в другом.**

**Верно?**

**На диаграмме классов обе связи идут как полоя ромб-стрелка? Или у них разное обозначени?**

**Композиция получается тоже одна из форм асоциации, только жесткая, или она само по себе?**

**ОТВЕТ ПРЕПОДА:**

**ассоциация это когда внутри каждого из классов может быть элемент связанного класса. Обозначается линией или простой стрелкой**

**Объекты классов могут использоваться по отдельности.**

**Агрегация-более жесткая ассоциация, объект агрегируемый не используется без агрегатора, что-то вроде составной части. На схеме пустой ромб у агрегатора**

**Композиция более жесткая агрегация. Когда жизненный цикл зап.части (агрегируемого объекта) определяется агрегатором. На схеме закрашенный ромб.**

**6)Пока разбирался с юмл наткнулся на странное противоречие:**

**Стандарт UML допускает при необходимости дополнять UML расширениями. Среди таких есть расширение Эриксона - Пенкера, разработанное специально для моделирования бизнес-процессов.**

**То есть это не стандарт? а уже расширение, но...**

**Предназначение UML:**

**Для визуализации объектно-ориентированных систем**

**Для моделирования сущностей и их взаимосвязей**

**Для визуального моделирования и проектирования различных систем в ключе объектно-ориентированных концепций**

**Для визуализации бизнес-процессов**

**Как так, то есть моделирования бизнес-процессов это всеголишь расширение но при этом язык юмл для него предназначен. Где я упуская логику?**

**ОТВЕТ ПРЕПОДА:**

**Во многих вещах необходимо искать компромисс. UML удобный инструмент для проектирования, описания и не только программирования.**

**Это скорее обобщение и попытка стандартизировать в общемировом масштабе некие документы**

**общение и представление между разработчиками.**

**7)** **-- Еще один момент обнаружил, читая диаграммы классов паттернов, обнуружил интересную вещь. У всех наследников дублируются методы и поля родителя внутри рамки класса. Это так и надо или это просто считается хорошим тоном? Ведь по сути и так понятно что если один клас наслдеуется от другого он и так наследует и поля и методы.**

**ОТВЕТ ПРЕПОДА:**

**Поля реже. Чаще методы. Так как это может быть указано на свою астрактную реализацию данного метода.**

**ВАЖНЫЕ ТЕРМЕНЫ:**

* **Полая стрелочка - объединение(наследование)**
* **Пунктирная стрелка - использование**
* **Стрелка темный ромб - включает (композиция)**
* **Стрелка пустой ромб - агрегация(ассоциация)**
* **UML - это нотация, используемая для описания элементов данных**
* **Понятие «суперкласс» – это понятие, которое применимо для…**

**-Применимо только для иерархии классов (наследования)**

**-Применимо для класса, у которого есть подклассы**

**-НЕ применимо для класса, который является ключевым классом**

* **Полюс ассоциации… - Может иметь имя и кратность**
* **Модель взаимодействия (последовательность, деятельность)**
* **Модель поведения (деятельность, состояние, использование)**
* **«наследование» в UML?**

**-Наличие у разных классов, входящих в иерархию классов, общих атрибутов и операций**

**-Преемственность атрибутов и методов наследуемого класса**

**Объяснение: Паттерн Фасад ? это структурный паттерн проектирования, предоставляющий простой интерфейс к сложной системе классов, библиотеке или фреймворку, а Адаптер позволяет объектам с несовместимыми интерфейсами работать вместе.**

**Фасад задаёт новый интерфейс, тогда как Адаптер повторно использует старый. Адаптер оборачивает только один класс, а Фасад оборачивает целую подсистему. Более того, Адаптер позволяет двум существующим интерфейсам работать сообща, вместо того, чтобы задать полностью новый.**

**Какие сущности относятся к поведенческим?**

**- Деятельность**

**- Прецедент**

**-Состояние**

**-Сообщение**

**Выберите правильные утверждения по отношению к параметрическому полиморфизму:**

**ВЕРНО:**

**Позволяет определять функцию или тип данных обобщённо, так что значения обрабатываются идентично вне зависимости от их типа5427 / 10573**

**Доступен в нескольких объектно-ориентированных языках, где он часто идет под названием "дженерик" или "шаблоны"5242 / 10573**

**НЕВРНО:**

**Поддерживается во многих языках посредством перегрузки функций и методов4442 / 10573**

**Позволяет давать одинаковые имена программным сущностям с различным поведением4174 / 10573**

**Не поддерживается в языках Java и С++**

**Какими бывают отношения между классами (укажите все подходящие варианты)?**

**Наследование16432 / 17567**

**Агрегация12966 / 1756**

**Ассоциация9423 / 17567**

**Пояснение: Еще выделяют отношения: композиция и класс-метакласс.**

**НЕТ:**

**Специализация3029 / 17567**

**Абстракция5161 / 17567**

**Иерархическое наследование(hierarchical inheritance):**

**Пояснение: Одномереное наследование - содержит один базовый класс и один производный класс. Иерархическое наследование - содержит один базовый класс и несколько производных классов одного и того же базового класса. Многоуровневое наследование - содержит класс, производный от производного класса. Множественное наследование - содержит несколько базовых классов и производный класс.**

**/\*\*\*/**

**Отношение композиции (4 варианта)**

**-частный случай агрегации**

**-является отношением part-of**

**-является отношением has-a**

**-указывает на физическое включение**

**Всего существует четыре основных вида отношений между классами:**

**q ассоциация (фиксирует структурные отношения — связи между экземплярами классов);**

**q зависимость (отображает влияние одного класса на другой класс);**

**q обобщение-специализация («is а»-отношение);**

**q целое-часть («part of»-отношение).**

**C++. Отношения наследования в ООП. Что чему соответствует?**

**Читаю "Язык Программирования С++. Лекции и Упражнения." — Прата.**

**Там упоминается (на сколько смог заметить) пять отношений наследования:**

**is-a — является - наследование**

**has-a — содержит - отношение типа "композиция"**

**uses-a — использует - отношение типа "агрегация"**

**is-like-a — подобен – два класса от одного родителя**

**is-implemented-as-a — реализован как - разделение на абстракцию и реализацию**

**1. is-a - наследование**

**class Car : public Vehicle {**

**// автомобиль является транспортом**

**};**

**2. has-a - отношение типа "композиция"**

**class Car {**

**Engine v8; // автомобиль имеет (содержит) двигатель**

**};**

**3. uses-a - отношение типа "агрегация"**

**class Driver {**

**Car\* myCar; // водитель использует автомобиль**

**};**

**4. is-like-a**

**class Square : public Figure;**

**class Rectangle : public Figure;**

**// квадрат и прямоугольник похожи по свойствам, но это разные фигуры**

**5. is-implemented-as-a**

**class Engine { // абстракция**

**public:**

**virtual void start() = 0;**

**protected:**

**float power;**

**};**

**class V8 : public Engine { // реализация**

**virtual void start() {**

**// wroom wroom**

**}**

**};**

**// Двигатель ДВС реализован как 8-ми цилиндровый двигатель V-конфигурации**

**/\*\*\*\*\*\*/**

**Основных отношений большинство объектно-ориентированных языков программирования поддерживает следующие отношения:**

**АССОЦИАЦИИ обеспечивают взаимодействия объектов, принадлежащих разным классам. Они являются клеем, соединяющим воедино все элементы программной системы. Благодаря ассоциациям мы получаем работающую систему. Без ассоциаций система превращается в набор изолированных классов-одиночек.**

**НАСЛЕДОВАНИЕ — наиболее популярная разновидность отношения обобщение-специализация. Альтернативой наследованию считается делегирование. При делегировании объекты делегируют свое поведение родственным объектам. При этом классы становятся не нужны.**

**АГРЕГАЦИЯ обеспечивает отношения целое-часть, объявляемые для экземпляров классов.**

**ЗАВИСИМОСТЬ часто представляется в виде частной формы — использования, которое фиксирует отношение между клиентом, запрашивающим услугу, и сервером, предоставляющим эту услугу.**

**КОНКРЕТИЗАЦИЯ выражает другую разновидность отношения обобщение-специализация. Применяется в таких языках, как Ada 95, C++, Эйфель.**

**ОТНОШЕНИЯ метаклассов поддерживаются в языках SmallTalk и CLOS. Метакласс — это класс классов, понятие, позволяющее обращаться с классами как с объектами.**

**РЕАЛИЗАЦИЯ определяет отношение, при котором класс-приемник обеспечивает свою собственную реализацию интерфейса другого класса-источника. Иными словами, здесь идет речь о наследовании интерфейса. Семантически реализация — это «скрещивание» отношений зависимости и обобщения-специализации.**

**Кстати возник такой вопрос, скорее даже потребность, если возможно.**

**Хотелось бы вместе с вами по решать задачи с использованием паттернов, но без использования кода, а порисовать диаграммы класса и на их примерах посмотреть конкретные места где возникают эти шаблоны. Только с более подробным описанием класса, что бы потом этом можно было самостоятельно оформит в код.**