Корпоративно-информационные системы
Новиков Александр Константинович
631 группа
Про язык UML (исправленное)

UML (Unified Modeling Language) – это графический способ представления и моделирования процессов и систем. Разработан с целью наглядного моделирования и показательного проектирования того или иного продукта.

UML – это язык не программирования, а моделирования (язык блок схем, по которым можно написать код).

UML состоит из:

1. Объекты;
2. Отношения, связывающие объекты;
3. Блок-схемы или диаграммы, полученными объединением объектов по отношениям;

Диаграммы бывают двух основных категорий:

1. Структурные: статическая структура ПО или системы. Показывают иерархию объектов и то, как они связаны и взаимодействуют друг с другом.
2. Поведенческие: показывают функциональные возможности и демонстрируют, что будет происходить в моделируемой системе.

Структурные диаграммы бывают:

1. Диаграммы классов: используется для изображения логической и физической структуры системы и показывает ее классы. Классы обозначаются в виде прямоугольников.
    a. У каждого класса есть три секции:
        i. Верхняя — Имя класса (иногда стереотип)
        ii. Средняя — Атрибуты класса (поименованные свойства, у которых есть тип). Если перед названием атрибута стоит минус, то он private (не виден извне).
        iii. Нижняя — Методы или операции класса (услуга)
2. У экземпляра класса имя подчёркнуто и перед ним стоит двоеточие
    a. Между классами есть отношения:
        i. Зависимость (меняется один => меняется другой)
        ii. Ассоциация (показывает связность, принадлежность)
        iii. Агрегация (нежесткая иерархия классов, уберём внешний, внутренние останутся)
        iv. Композиция (жесткая иерархия классов, уберём внешний, внутренние уничтожатся)
        v. Наследование (на основе предка стоится класс ребёнок)
        vi. Реализация (набор соотношений и правил класса)
3. Диаграммы объектов. Проверка диаграммы классов на практике.
4. Диаграммы компонентов. Показывают логические группы элементов и их взаимосвязи. Упрощают представление о сложной системе, разделяя ее на простые компоненты.
5. Составные структурные диаграммы. Сложноорганизованные структурные диаграммы.
6. Диаграмма развертывания. Наглядное представление о том, где именно развернут каждый программный компонент.

7. Диаграмма пакетов. Показывают взаимосвязь между различными крупными компонентами, которые образуют сложную систему.
8. Диаграммы профиля. Помогают создавать новые свойства и семантику для диаграмм UML путем определения пользовательских стереотипов, теговых значений и ограничений.

Поведенческие диаграммы бывают:

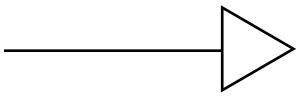
1. Диаграммы деятельности. Изображают пошаговый процесс с четким началом и концом.
2. Диаграммы вариантов использования (взаимодействия). Описывается, именно **что** делает система, а не как.
3. Обзорные диаграммы взаимодействия. Диаграммы деятельности, составленные из разных диаграмм взаимодействия.
4. Временные диаграммы. Отображения событий по шкале времени. Хронология.
5. Диаграммы конечного автомата (диаграммы состояний). Описывают поведение одного объекта и то, как оно изменяется в зависимости от внутренних и внешних событий.
6. Диаграммы последовательности. Просто раскрывают структуру системы.
7. Диаграммы связи (сотрудничества). Подчёркивает связь между объектами.

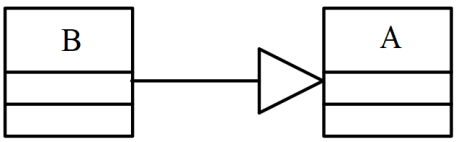
Три принципа ООП

1. Инкапсуляция (наш объект – капсула, в котором спрятаны атрибуты, и есть интерфейс – видимые извне операции, поименованные операции)
2. Наследование (можно создать пустой класс и унаследовать ему атрибуты, операции и связи от другого класса)
3. Полиморфизм: разные реализации одного и того же интерфейса. Связан с лучшей практикой: использованием компонентной архитектуры.

Компонента – модуль, публикующий свои интерфейсы. Состоит из двух частей: интерфейс и модуль его реализующий.

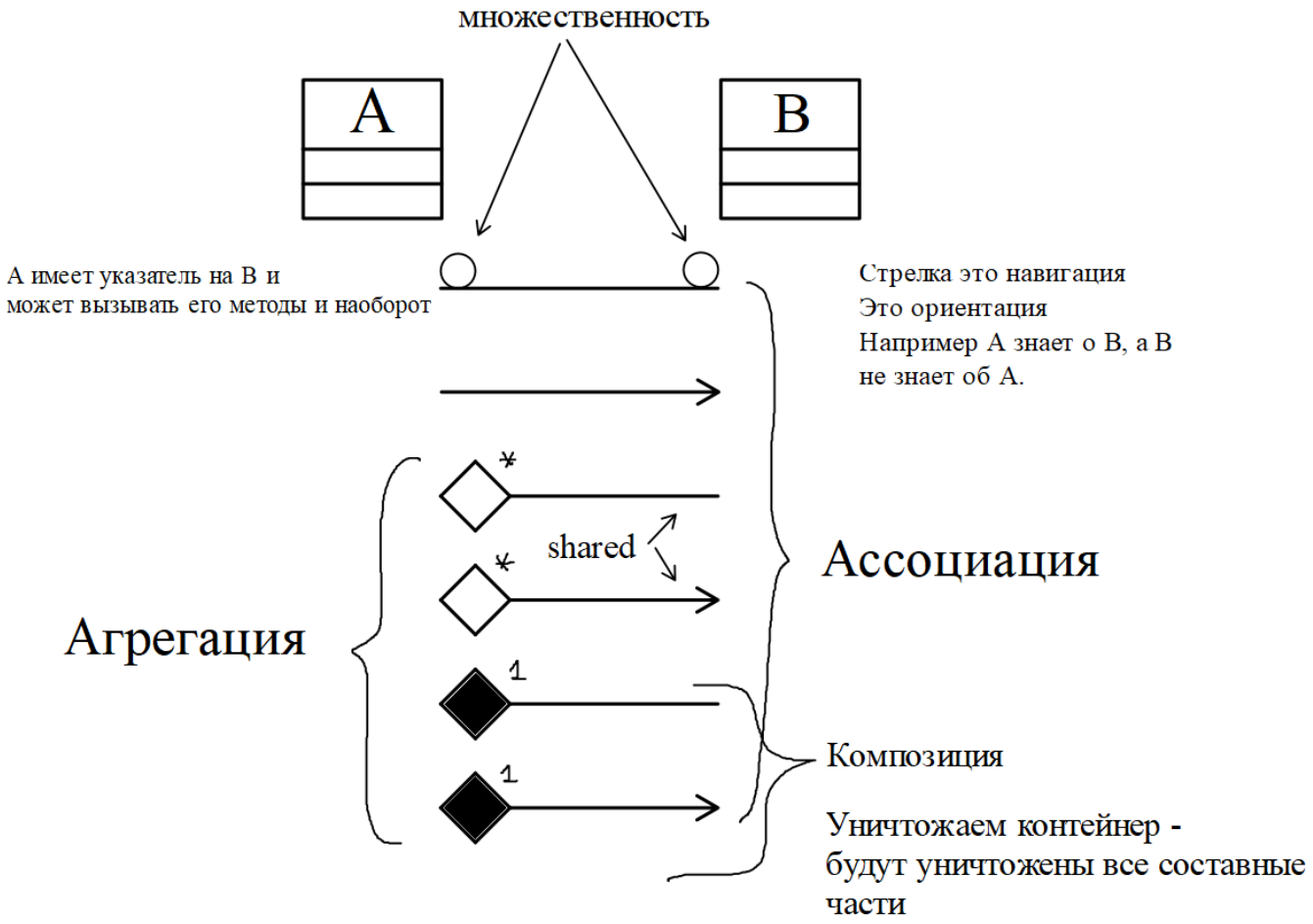
Интерфейс – поименованный набор операций. Мы публикуем операции компоненты в реестр, когда создаём её. Когда нужна какая-то операция, другая программа ищет интерфейс в реестре. Интерфейс связан с некоторым модулем, поэтому далее происходит загрузка этого модуля и передача ему управления над операцией. Значит можно поправить модуль, не меняя интерфейс и переопубликовать компоненту так, что оставшиеся части системы не узнают, что что-то поменялось.

Наследование обозначается: 



наследник родитель

Конец ассоциации – роль, роль имеет множественность.



Стереотип – это некоторое общее название элементов, метатип, создающий эквивалент нового класса в метамодели UML. Элементы модели, представляющие работников, инструкции, документы и стратегии.