УВАЖАЕМЫЕ СТУДЕНТЫ! Изучите приведенную лекцию, законспектируйте основные сведения.

Ответы на вопросы, фотоотчет, предоставить преподавателю на e-mail: r.bigangel@gmail.com до 27.03.2023.

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)111-37-59, (Viber, WhatsApp), vk.com: https://vk.com/daykini

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

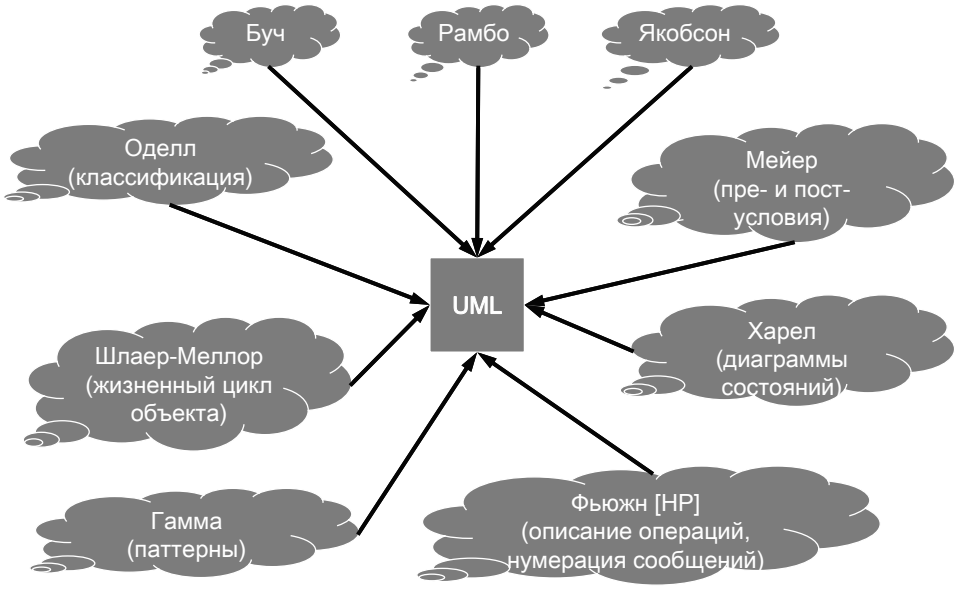
Лекция 9

Лекция Основные элементы нотации языка UML

Цель: Основные элементы нотации языка UML

Унифицированный язык моделирования UML (Unified Modeling Language) – это язык для определения, представления, проектирования и документирования программных систем, организационно-экономических систем, технических систем и других систем различной природы. UML содержит стандартный набор диаграмм и нотаций самых разнообразных видов.

UML является наследником методов объектно-ориентированного анализа и проектирования, появившихся в конце 1980-х и начале 1990-х годов. Создание UML началось в конце 1994 г., с объединения методов Booch и OMT (Object Modeling Technique) под эгидой компании Rational Software. К концу 1995 г. Гради Буч и Джеймс Рамбо создали первую спецификацию Unified Method, версия 0.8. Тогда же в 1995 г. к ним присоединился создатель метода OOSE (Object-Oriented Software Engineering) Ивар Якобсон. UML является унификацией методов Буча, Рамбо и Якобсона а также суммой передового опыта по разработке ПО:



Разработка UML преследовала следующие цели:

- предоставить разработчикам единый язык визуального моделирования;
- предусмотреть механизмы расширения и специализации языка;
- обеспечить независимость языка от языков программирования и процессов разработки;
- интегрировать накопленный практический опыт.

UML широко используется в индустрии ПО. Практически все мировые производители CASE-средств поддерживают UML в своих продуктах. В 1997 году Object Management Group (OMG) приняла стандарт UML 1.1. В 2004 году был пройден следующий важный этап – принят стандарт UML версии 2.0. В настоящее время UML проходит процесс стандартизации ISO. Текущая версия UML – 2.1.2.

Основные «строительные блоки» UML:

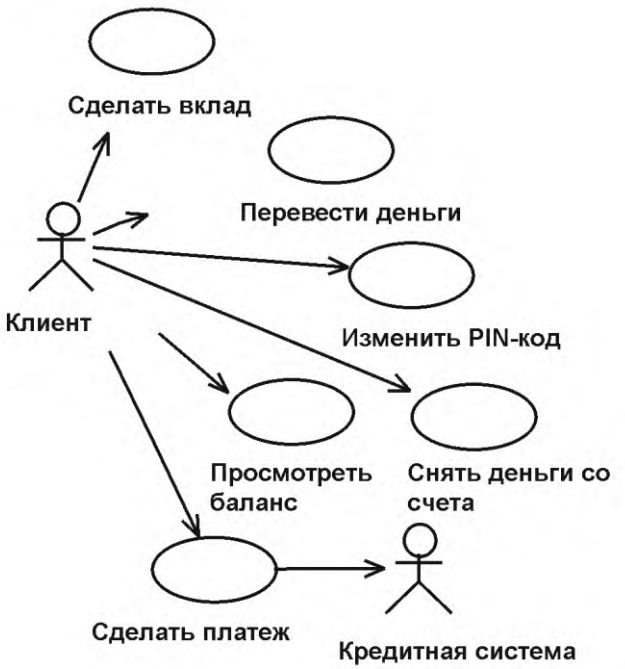
- элементы модели (классы, интерфейсы, компоненты, варианты использования и др.);
- связи (ассоциации, обобщения, зависимости и др.);
- механизмы расширения (стереотипы, ограничения, примечания, именованные значения);
- диаграммы.

Состав диаграмм UML 1.x:

- структурные:
    - диаграммы классов, моделирующие статическую структуру классов системы и связи между классами;
    - диаграммы компонентов, моделирующие иерархии компонентов ПО;
    - диаграммы размещения, моделирующие физическую архитектуру системы;
- поведенческие:
    - диаграммы вариантов использования, моделирующие бизнес-процессы и требования к ПО;
    - диаграммы взаимодействия (диаграммы последовательности и коммуникационные диаграммы), моделирующие обмен сообщениями между объектами;
    - диаграммы состояний, моделирующие поведение объектов;
    - диаграммы деятельности, моделирующие поведение системы в целом и потоки управления.

В UML 2.0 введены новые типы диаграмм, которых ранее не было: диаграммы обзора взаимодействия, временные диаграммы и диаграммы составных структур.

Вариант использования – это ответные действия ПО, являющиеся реакцией на событие, инициируемое извне. Вариант использования описывает типичное



взаимодействие между пользователем и ПО. Он отражает представление о поведении системы с точки зрения пользователя. На диаграммах варианты использования представляются в виде овалов.

Действующее лицо – это роль, которую пользователь играет по отношению к системе. На диаграммах вариантов использования они изображаются в виде стилизованных человеческих фигурок. Действующим лицом может быть пользователь-человек, внешняя программная система или время, если запуск каких-либо событий в системе зависит от времени.

Диаграммы вариантов использования показывают, какие действующие лица инициируют варианты использования (от них идет стрелка к варианту использования). Из диаграмм понятно, какие действующие лица получают данные в ходе выполнения варианта использования (к ним идет стрелка от варианта использования).

Диаграмма вариантов использования является самым общим представлением функциональных требований к системе. Детально функциональные требования описываются в документе, называемом «сценарий варианта использования» или «поток событий». Он подробно документирует процесса взаимодействия действующего лица с системой, реализуемого в рамках варианта использования.

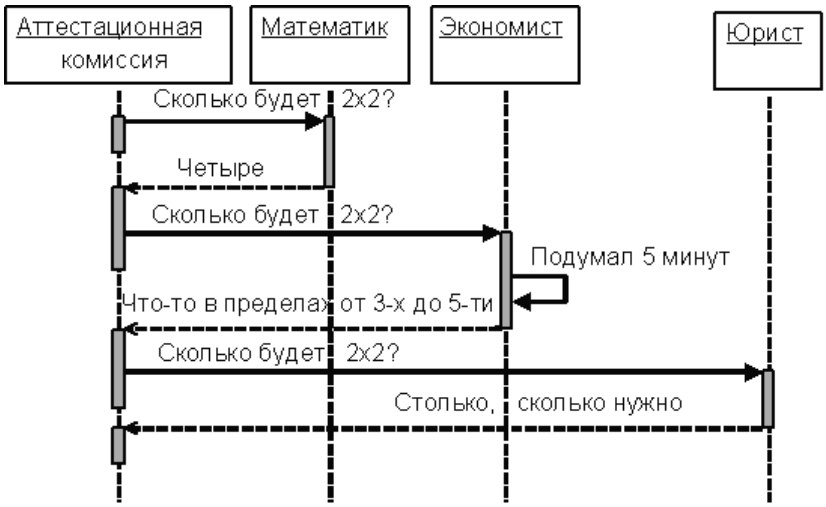
В диаграммах вариантов использования может присутствовать несколько типов связей:

- связи коммуникации (линия со стрелкой, обозначающая связь между вариантом использования и действующим лицом);
- связи включения (пунктирная линия со стрелкой, обозначающая включение многократно используемой функциональности, представленной в виде абстрактного варианта использования);
- связи расширения (пунктирная линия со стрелкой, указывающая на особый случай, описанный в абстрактном варианте использования);
- связи обобщения (линия с треугольным концом, показывающая, что у нескольких действующих лиц имеются общие черты и различия).

Диаграммы взаимодействия описывают поведение взаимодействующих групп объектов в рамках потока событий. На диаграмме отображается ряд

объектов и сообщения, которыми они обмениваются между собой. Сообщение – это средство, с помощью которого объект-отправитель запрашивает у объекта-получателя выполнение одной из его операций. Существует два вида диаграмм взаимодействия: диаграммы последовательности и коммуникационные диаграммы (ранее называемые кооперативными).

Диаграммы последовательности отражают временную последовательность событий, происходящих в рамках варианта использования. Все действующие лица и объекты системы изображаются в верхней части диаграммы. От каждого из них вниз проведена вертикальная черта – «линия жизни». Стрелки, соответствующие



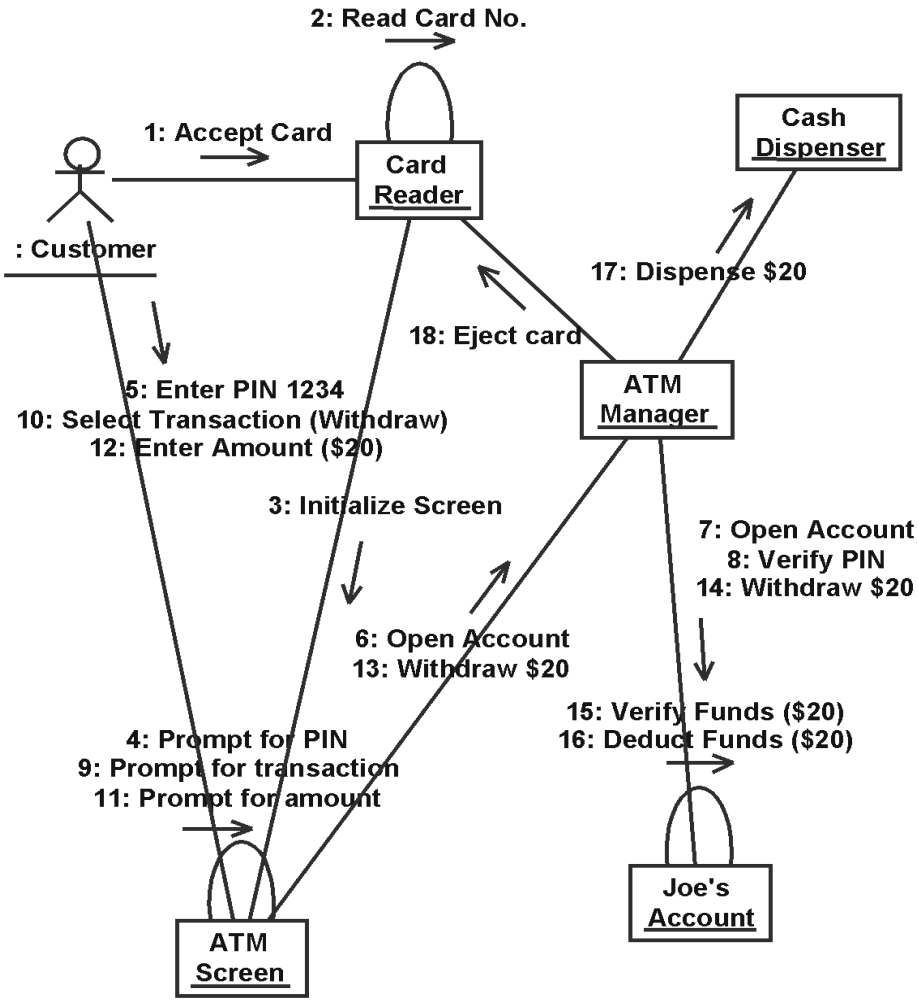
сообщениям, которые передаются между действующим лицом и объектом или между объектами, соединяют линии жизни отправителя и получателя сообщения. Порядок отправки сообщений соответствует их размещению на диаграмме сверху вниз.

В UML 2.0 диаграммы последовательности могут содержать блоки разных типов:

- alt – несколько альтернатив (каждая альтернатива – часть блока помеченная сторожевым условием);
- opt – необязательный блок (взаимодействие выполняемое при истинности сторожевого условия);
- par – параллельно выполняемые блоки;
- loop – цикл (пока истинно условие);
- region – критический участок;
- neg – неверное взаимодействие;

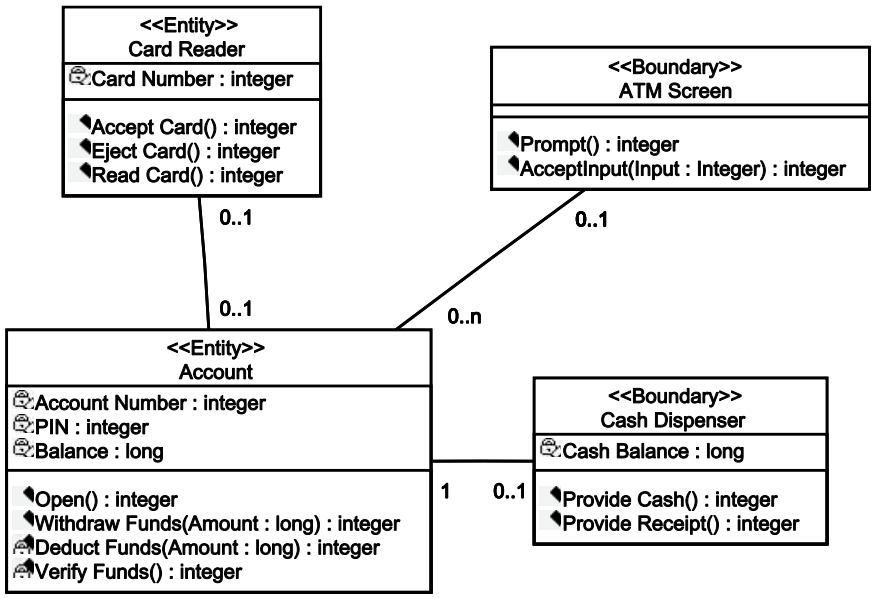
- ref – ссылка на субдиаграмму;
- sd – блок, включающий диаграмму целиком, используется для субдиаграмм.

Вторым видом диаграмм взаимодействия являются *коммуникационные диаграммы* (в UML 1 их называли *кооперативными*). Как и диаграммы

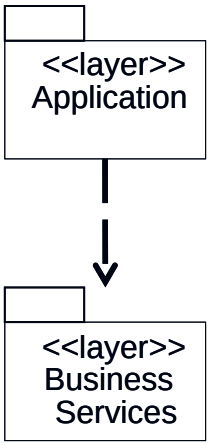


последовательности, они отображают поток событий варианта использования. На коммуникационных диаграммам внимание сконцентрировано на связях между объектами. Из них легче понять связи между объектами, однако, труднее уяснить последовательность событий. Объекты и/или действующие лица, обменивающиеся сообщениями соединяются линиями, над которым в виде стрелок обозначаются сообщения. Нумерация сообщений указывает их последовательность во времени.

Диаграмма классов определяет классы системы и различного рода связи,



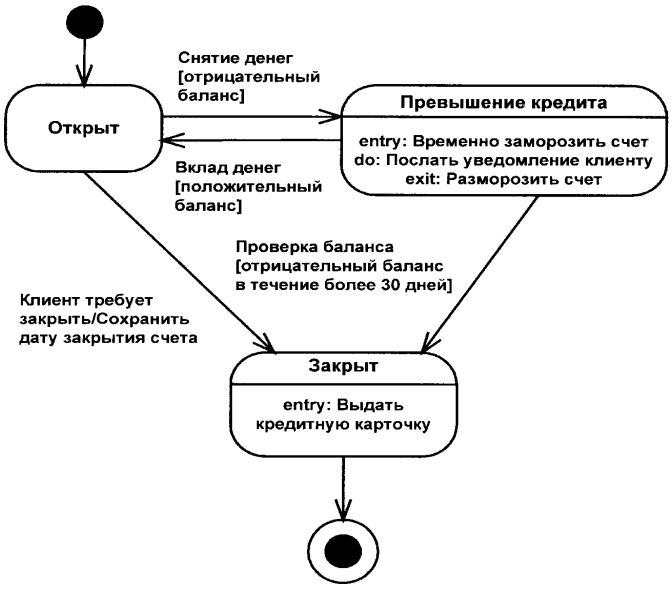
которые существуют между ними (ассоциации, агрегации, композиции, зависимости, обобщения, реализации). На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами. Классы изображаются в виде прямоугольников, ассоциации – в виде линий со стрелками, агрегации и композиции – в виде линий с ромбом на конце, связь обобщения – в виде линии с треугольником на конце, зависимость – в виде пунктирной линии со стрелкой. Ответственность классов изображается на диаграммах с помощью стереотипов. Класс может быть помечен как граничный (boundary), если он отвечает за взаимодействие с пользователем или внешней системой. Класс-контроллер реализует бизнес-логику приложения. Класс-сущность отвечает за представление данных.



Для группировки классов, обладающих некоторой общностью, применяются пакеты. Пакет – общий механизм для организации элементов модели в группы. Каждый пакет – это группа элементов модели, иногда сопровождаемая диаграммами, поясняющими структуру группы. Каждый элемент модели может входить только в один пакет. Диаграммы пакетов отображают зависимости между пакетами, возникающие, если элемент одного пакета зависит от элемента другого.

Пакеты также используются для представления подсистем. Подсистема – это комбинация пакета (поскольку она включает некоторое множество классов) и класса (поскольку она обладает поведением, т.е. реализует набор операций, которые определены в ее интерфейсах). Связь между подсистемой и интерфейсом называется связью реализации.

Диаграммы состояний определяют все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате наступления некоторых событий. На каждой диаграмме состояний имеется одно начальное состояние и одно (или более одного) финальное. Начальное состояние выделено черной точкой, оно соответствует состоянию



объекта, когда он только что был создан. Финальное состояние обозначается черной точкой в белом кружке, оно соответствует состоянию объекта непосредственно перед его уничтожением. Когда объект находится в каком-то конкретном состоянии, могут выполняться различные процессы. Они, называются деятельностями состояния и указываются на диаграмме. Деятельность состояния – это прерываемое поведение. Оно может выполняться до своего завершения, пока объект находится в данном состоянии, или может быть прервано переходом объекта в другое состояние. Деятельность состояния изображают внутри самого состояния, ей должно предшествовать слово do и двоеточие. Для состояния могут быть указаны входное и выходное действия. Входное действие выполняется, когда объект переходит в данное состояние, как часть этого перехода. В отличие

от деятельности, входное действие рассматривается как непрерываемое. Входное действие также показывают внутри состояния, ему предшествует слово entry и двоеточие. Выходное действие осуществляется как составная часть выхода из данного состояния. Оно также является непрерываемым. Его изображают внутри состояния, ему предшествует слово exit и двоеточие.

Переходом (transition) называется перемещение объекта из одного состояния в другое. На диаграмме все переходы изображают в виде линий со стрелками. Объект может перейти в то же состояние, в котором он в настоящий момент находится. С переходом можно связать событие, ограждающее условие и действие. Событие вызывает переход из одного состояния в другое. Событие размещают на диаграмме вдоль линии перехода. Ограничивающие условия определяют, когда переход может произойти, а когда нет. Их изображают на диаграмме вдоль линии перехода после имени события, заключая в квадратные скобки. Действие, являющееся частью перехода, изображают вдоль линии перехода после имени события, ему предшествует косая черта. Для некоторых состояний указывают вложенные подсостояния. В них может быть указано так называемое историческое состояние, переход в которое означает возврат к предыдущему активному подсостоянию.

Диаграмма размещения используется менеджером проекта, пользователями, архитектором системы и эксплуатационным персоналом, чтобы понять физическое размещение системы и расположение ее отдельных подсистем.

Механизмы расширения UML предназначены для того, чтобы разработчики могли адаптировать язык моделирования к своим конкретным нуждам, не меняя при этом его метамодель. Наличие механизмов расширения принципиально отличает UML от других средств моделирования. К механизмам расширения UML относятся:

- стереотипы;
- теги (ключевые слова);
- примечания;
- ограничения.

Стереотип – это новый тип элемента модели, который определяется на

основе уже существующего элемента. Стереотипы расширяют нотацию модели, могут применяться к любым элементам модели и представляются в виде текстовой метки или пиктограммы.

Стереотипы классов – это механизм, позволяющий разделять классы на категории. Например, основными стереотипами, используемыми в процессе анализа системы, являются: Boundary (граничный класс), Entity (класс-сущность) и Control (управляющий класс).

Помимо упомянутых стереотипов, разработчики ПО могут создавать свои собственные наборы стереотипов, формируя тем самым специализированные подмножества UML (например, для описания бизнес-процессов, Web-приложений, баз данных и т.д.). Такие подмножества (наборы стереотипов) в стандарте языка UML носят название профилей языка.

Теги (именованные значения) – специальные термины, используемые спецификации ограничений и свойств, такие как disjoint, complete, incomplete и др., могут сопровождаться указанием значения свойства, например, author=Вася или location=server.

Примечание – элемент диаграммы для комментария или другой текстовой информации. Примечание может содержать дополнительные сведения об элементах модели (с ними его соединяет пунктирная линия).

Ограничение – это семантическое ограничение, имеющее вид текстового выражения на естественном или формальном языке (OCL – Object Constraint Language), которое невозможно выразить с помощью нотации UML. Средства OCL не предназначены для описания процессов вычисления выражений, а только лишь фиксируют необходимость выполнения тех или иных условий применительно к отдельным компонентам моделей. Он может быть использован для решения следующих задач:

- описание инвариантов классов и типов в модели классов;
- описание пред- и постусловий в операциях и методах;
- описание ограничивающих условий элементов модели;
- навигация по структуре модели;
    - спецификация ограничений на операции.