Лекция 3:

Элементы графической нотации диаграммы классов

Ключевые слова: место, класс, диаграмма, сущность предметной области, информация, class diagram, UML, статический элемент, атрибут, операция, представление, разделы, имя класса, ПО, конкретный класс, Окружность, длина, резервное копирование, Дополнение, предметной области, отношение, абстрактный класс, concrete class, шрифт, значение, аспект описания, специальный символ, синтаксис, имя пакета, квантор видимости, кратность, visibility, ключевое слово, произвольное, целое число, монотонно возрастающей, выражение, семантика, конструктор, запись, фигурные скобки, element, список, параметр, очередь, идентификатор, тип данных, software, development, modeling, NAME, интерфейс, прямоугольник

Центральное место в методологии ООАП занимает разработка логической модели системы в виде диаграммы классов. Диаграмма классов отражает, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы. С этой точки зрения диаграмма классов может служить дальнейшим развитием концептуальной модели проектируемой системы

Диаграмма классов (class diagram) — диаграмма языка UML, на которой представлена совокупность декларативных или статических элементов модели, таких как классы с атрибутами и операциями, а также связывающие их отношения.

Диаграмма классов предназначена для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. При этом диаграмма классов может содержать интерфейсы, пакеты, отношения и даже отдельные экземпляры классификаторов, такие как объекты и связи. Когда говорят о данной диаграмме, имеют в виду статическую структурную модель проектируемой системы, т.е. графическое представление таких структурных взаимосвязей логической модели системы, которые не зависят от времени.

Класс

Класс (class) — абстрактное описание множества однородных объектов, имеющих одинаковые атрибуты, операции и отношения с объектами других классов .

Графически класс в нотации языка UML изображается в виде прямоугольника, который дополнительно может быть разделен горизонтальными линиями на разделы или секции (рис. 3.1). В этих секциях могут указываться имя класса, атрибуты и операции класса.

Рис. 3.1. Варианты графического изображения класса на диаграмме классов

На начальных этапах разработки диаграммы отдельные классы могут обозначаться простым прямоугольником, в котором должно быть указано имя соответствующего класса (рис. 3.1, а). По мере проработки отдельных компонентов диаграммы описание классов дополняется атрибутами (рис. 3.1, б) и операциями (рис. 3.1, в). Четвертая секция (рис. 3.1, г) не обязательна и служит для размещения дополнительной информации справочного характера, например, об исключениях или ограничениях класса, сведения о разработчике или языке реализации. Предполагается, что окончательный вариант диаграммы содержит наиболее полное описание классов, которые состоят из трех или четырех секций.

Даже если секции атрибутов и операций пусты, в обозначении класса они должны быть выделены горизонтальной линией, с тем чтобы отличить класс от других элементов языка UML. Примеры графического изображения конкретных классов приведены на рис. 3.2. В первом случае для класса Окружность (рис. 3.2, а) указаны только его атрибуты – точка на координатной плоскости, которая определяет расположение ее центра и длина радиуса окружности. Для класса Окно (рис. 3.2, б) указаны только его операции, при этом секция его атрибутов оставлена пустой. Для класса Счет (рис. 3.2, в) дополнительно изображена четвертая секция, в которой указано требование – реализовать резервное копирование объектов этого класса.

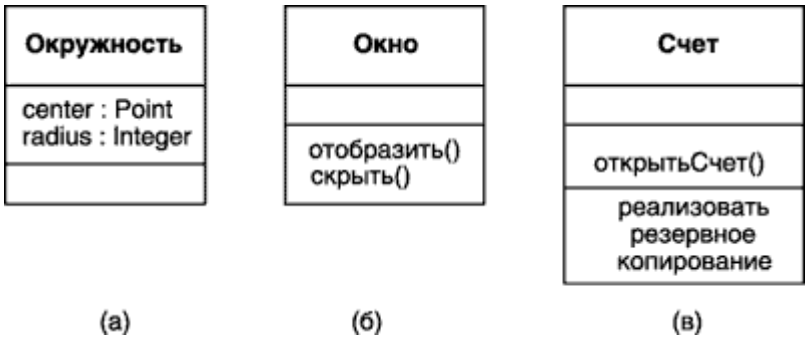

Рис. 3.2. Примеры графического изображения конкретных классов

Имя класса

Имя класса должно быть уникальным в пределах пакета, который может содержать одну или несколько диаграмм классов. Имя указывается в самой верхней секции прямоугольника, поэтому она часто называется секцией имени класса. В дополнение к общему правилу именования элементов языка UML, имя класса записывается по центру секции имени полужирным шрифтом и должно начинаться с заглавной буквы. Рекомендуется в качестве имен классов использовать существительные, записанные по практическим соображениям без пробелов. Необходимо помнить, что имена классов образуют словарь предметной области при ООАП.

В секции имени класса могут также находиться стереотипы или ссылки на стандартные шаблоны, от которых образован данный класс и, соответственно, от которых он наследует атрибуты и операции. В этой секции может также приводиться информация о разработчике данного класса и статус состояния разработки. Здесь также могут записываться и другие общие свойства этого класса, имеющие отношение к другим классам диаграммы или стандартным элементам языка UML.

Класс может иметь или не иметь экземпляров или объектов. В зависимости от этого в языке UML различают конкретные и абстрактные классы.

Конкретный класс (concrete class) — класс, на основе которого могут быть непосредственно созданы экземпляры или объекты.

Рассмотренные выше обозначения относятся к конкретным классам. От них следует отличать абстрактные классы.

Абстрактный класс (abstract class) — класс, который не имеет экземпляров или объектов.

Для обозначения имени абстрактного класса используется наклонный шрифт (курсив). В языке UML принято общее соглашение о том, что любой текст, относящийся к абстрактному элементу, записывается курсивом. Это имеет принципиальное значение, поскольку является семантическим аспектом описания абстрактных элементов языка UML.

В некоторых случаях необходимо явно указать, к какому пакету относится тот или иной класс. Для этой цели используется специальный символ разделитель – двойное двоеточие - (::). Синтаксис строки имени класса в этом случае будет следующий: <Имя пакета>::< Имя класса >. Другими словами, перед именем класса должно быть явно указано имя пакета, к которому его следует отнести. Например, если определен пакет с именем Банк, то класс Счет в этом банке может быть записан в виде: Банк::Счет.

Атрибуты класса

Атрибут (attribute) — содержательная характеристика класса, описывающая множество значений, которые могут принимать отдельные объекты этого класса .

Атрибут класса служит для представления отдельного свойства или признака, который является общим для всех объектов данного класса. Атрибуты класса записываются во второй сверху секции прямоугольника класса. Эту секцию часто называют секцией атрибутов.

В языке UML принята определенная стандартизация записи атрибутов класса, которая подчиняется некоторым синтаксическим правилам. Каждому атрибуту класса соответствует отдельная строка текста, которая состоит из квантора видимости атрибута, имени атрибута, его кратности, типа значений атрибута и, возможно, его исходного значения. Общий формат записи отдельного атрибута класса следующий:

<квантор видимости> <имя атрибута> [кратность] :
<тип атрибута> = <исходное значение> {строка-свойство}.

Видимость (visibility) — качественная характеристика описания элементов класса, характеризующая потенциальную возможность других объектов модели оказывать влияние на отдельные аспекты поведения данного класса.

Видимость в языке UML специфицируется с помощью квантора видимости (visibility), который может принимать одно из 4-х возможных значений и отображаться при помощи специальных символов.

- Символ " + " – обозначает атрибут с областью видимости типа общедоступный (public). Атрибут с этой областью видимости доступен или виден из любого другого класса пакета, в котором определена диаграмма.

- Символ " # " – обозначает атрибут с областью видимости типа защищенный (protected). Атрибут с этой областью видимости недоступен или не виден для всех классов, за исключением подклассов данного класса.
- Символ " - " – обозначает атрибут с областью видимости типа закрытый (private). Атрибут с этой областью видимости недоступен или не виден для всех классов без исключения.
- И, наконец, символ " ~ " - обозначает атрибут с областью видимости типа пакетный (package). Атрибут с этой областью видимости недоступен или не виден для всех классов за пределами пакета, в котором определен класс -владелец данного атрибута.

Квантор видимости может быть опущен. Его отсутствие означает, что видимость атрибута не указывается. Эта ситуация отличается от принятых по умолчанию соглашений в традиционных языках программирования, когда отсутствие квантора видимости трактуется как public или private. Однако вместо условных графических обозначений можно записывать соответствующее ключевое слово: public, protected, private, package.

Имя атрибута представляет собой строку текста, которая используется в качестве идентификатора соответствующего атрибута и поэтому должна быть уникальной в пределах данного класса. Имя атрибута - единственный обязательный элемент синтаксического обозначения атрибута, должно начинаться со строчной (малой) буквы и не должно содержать пробелов.

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.

Тип атрибута представляет собой выражение, семантика которого определяется некоторым типом данных, определенным в пакете Типы данных языка UML или самим разработчиком. В нотации UML тип атрибута иногда определяется в зависимости от языка программирования, который

предполагается использовать для реализации данной модели. В простейшем случае тип атрибута указывается строкой текста, имеющей осмысленное значение в пределах пакета или модели, к которым относится рассматриваемый класс.

Исходное значение служит для задания начального значения соответствующего атрибута в момент создания отдельного экземпляра класса. Здесь необходимо придерживаться правила принадлежности значения типу конкретного атрибута. Если исходное значение не указано, то значение соответствующего атрибута не определено на момент создания нового экземпляра класса. С другой стороны, конструктор объекта может переопределять исходное значение в процессе выполнения программы, если в этом возникает необходимость.

При задании атрибутов могут быть использованы дополнительные синтаксические конструкции — это подчеркивание строки атрибута, пояснительный текст в фигурных скобках и косая черта перед именем атрибута. Подчеркивание строки атрибута означает, что соответствующий атрибут общий для всех объектов данного класса, т.е. его значение у всех создаваемых объектов одинаковое (аналог ключевого слова static в некоторых языках программирования).

Пояснительный текст в фигурных скобках может означать две различные конструкции. Если в этой строке имеется знак равенства, то вся запись Строка-свойство служит для указания дополнительных свойств атрибута, которые могут характеризовать особенности изменения значений атрибута в ходе выполнения программы. Фигурные скобки как раз и обозначают фиксированное значение соответствующего атрибута для класса в целом, которое должны принимать все вновь создаваемые экземпляры класса без исключения. Это значение принимается за исходное значение атрибута, которое не может быть переопределено в последующем. Отсутствие строки-свойства по умолчанию трактуется так, что значение соответствующего атрибута может быть изменено в программе.

Знак " / " перед именем атрибута указывает на то, что данный атрибут является производным от некоторого другого атрибута этого же класса.

Производный атрибут (derived element) — атрибут класса, значение которого для отдельных объектов может быть вычислено посредством значений других атрибутов этого же объекта.

Операции класса

Операция (operation) - это сервис, предоставляемый каждым экземпляром или объектом класса по требованию своих клиентов, в качестве которых могут выступать другие объекты, в том числе и экземпляры данного класса .

Операции класса записываются в третьей сверху секции прямоугольника класса, которую часто называют секцией операций. Совокупность операций характеризует функциональный аспект поведения всех объектов данного класса. Запись операций класса в языке UML также стандартизована и подчиняется определенным синтаксическим правилам. При этом каждой операции класса соответствует отдельная строка, которая состоит из квантора видимости операции, имени операции, выражения типа возвращаемого операцией значения и, возможно, строка-свойство данной операции. Общий формат записи отдельной операции класса следующий:

<квантор видимости> <имя операции>(
список параметров):
<выражение типа возвращаемого значения>

{строка-свойство}

Квантор видимости, как и в случае атрибутов класса, может принимать одно из четырех возможных значений и, соответственно, отображается при помощи специального символа либо ключевого слова. Символ " + " обозначает операцию с областью видимости типа общедоступный (public). Символ " # " обозначает операцию с областью видимости типа защищенный (protected). Символ " - " используется для обозначения операции с областью видимости типа закрытый (private). И, наконец, символ " ~ " используется для обозначения операции с областью видимости типа пакетный (package).

Квантор видимости для операции может быть опущен. В этом случае его отсутствие просто означает, что видимость операции не указывается. Вместо условных графических обозначений также можно записывать соответствующее ключевое слово: public, protected, private, package.

Имя операции представляет собой строку текста, которая используется в качестве идентификатора соответствующей операции и поэтому должна быть уникальной в пределах данного класса. Имя операции - единственный обязательный элемент синтаксического обозначения операции, должно начинаться со строчной (малой) буквы, и, как правило, записываться без пробелов.

Список параметров является перечнем разделенных запятой формальных параметров, каждый из которых, в свою очередь, может быть представлен в следующем виде:

<направление параметра> <имя параметра>:
<выражение типа> =
<значение параметра по умолчанию>.

Параметр (parameter) — спецификация переменной операции, которая может быть изменена, передана или возвращена.

Параметр может включать имя, тип, направление и значение по умолчанию. Направление параметра — есть одно из ключевых слов in, out или inout со значением in по умолчанию, в случае если вид параметра не указывается. Имя параметра есть идентификатор соответствующего формального параметра, при записи которого следуют правилам задания имен атрибутов. Выражение типа является спецификацией типа данных для допустимых значений соответствующего формального параметра. Наконец, значение по умолчанию в общем случае представляет собой некоторое конкретное значение для этого формального параметра.

Выражение типа возвращаемого значения также указывает на тип данных значения, которое возвращается объектом после выполнения соответствующей операции. Две точки и выражение типа возвращаемого значения могут быть опущены, если операция не возвращает никакого значения. Для указания нескольких возвращаемых значений данный элемент спецификации операции может быть записан в виде списка отдельных выражений.

Операция с областью действия на весь класс показывается подчеркиванием имени и строки выражения типа. В этом случае под областью действия операции понимаются все объекты этого класса. В этом случае вся строка записи операции подчеркивается.

Строка-свойство служит для указания значений свойств, которые могут быть применены к данной операции. Строка-свойство может отсутствовать, если свойства не специфицированы.

Список формальных параметров и тип возвращаемого значения не обязателен. Квантор видимости атрибутов и операций может быть указан в виде специального значка или символа, которые используются для графического представления моделей в инструментальном средстве. Еще раз следует напомнить, что имена операций, так же как атрибутов и параметров, записываются со строчной (малой) буквы, а их типы параметров — с заглавной (большой) буквы. При этом обязательной частью строки записи операции является наличие имени операции и круглых скобок.

Расширение языка UML для построения моделей программного обеспечения и бизнес-систем

Одним из несомненных достоинств языка UML является наличие механизмов расширения, которые позволяют ввести в рассмотрение дополнительные графические обозначения, ориентированные для решения задач из определенной предметной области. Язык UML содержит два специальных расширения: профиль для процесса разработки программного обеспечения (The UML Profile for Software Development Processes) и профиль для бизнес-моделирования (The UML Profile for Business Modeling).

В рамках первого из них предложено три специальных графических примитива, которые могут быть использованы для уточнения семантики отдельных классов при построении различных диаграмм:

- Управляющий класс (control class) — класс, отвечающий за координацию действий других классов. На каждой диаграмме классов должен быть хотя бы один управляющий класс, причем количество посылаемых объектам управляющего класса сообщений мало, по сравнению с числом рассылаемых ими. Управляющий класс отвечает за координацию действий других классов. У каждой диаграммы классов должен быть хотя бы один управляющий класс, контролирующий последовательность выполнения действий этого варианта использования. Как правило, данный класс является активным и инициирует рассылку множества сообщений другим классам модели. Кроме специального обозначения управляющий класс может быть изображен в форме прямоугольника класса со стереотипом <<control>> (рис. 3.3, а).
- Класс -сущность (entity class) — пассивный класс, информация о котором должна храниться постоянно и не уничтожаться с выключением системы. Класс -сущность содержит информацию, которая должна храниться постоянно и не уничтожается с уничтожением объектов данного класса или прекращением работы моделируемой системы, связанные с выключением системы или завершением программы. Как правило, этот класс соответствует отдельной таблице базы данных. В этом случае его атрибуты являются полями таблицы, а операции – присоединенными или хранимыми процедурами. Этот класс пассивный и лишь принимает сообщения от других классов модели. Класс -сущность может быть изображен также стандартным образом в форме прямоугольника класса со стереотипом <<entity>> (рис. 3.3, б).
- Граничный класс (boundary class) — класс, который располагается на границе системы с внешней средой и непосредственно взаимодействует с актерами, но является составной частью системы. Граничный класс может быть изображен также стандартным образом в форме прямоугольника класса со стереотипом <<boundary>> (рис. 3.3, в).

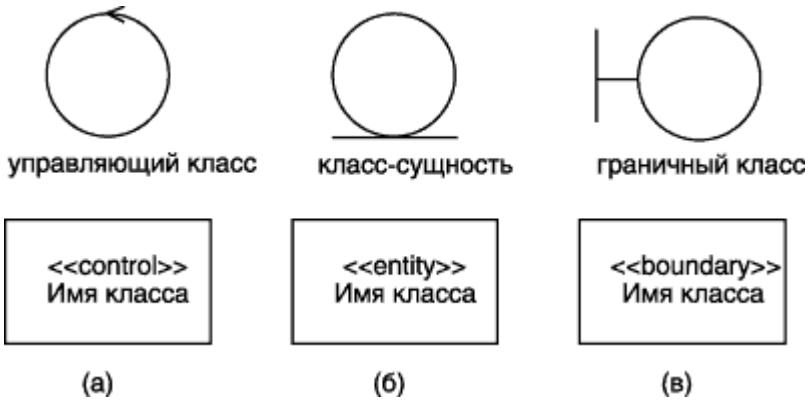

Рис. 3.3. Графическое изображение классов для моделирования программного обеспечения

В рамках второго профиля также предложено три специальных графических примитива, которые могут быть использованы для уточнения семантики отдельных классов при построении моделей бизнес-систем:

- Сотрудник (business worker) — класс, служащий на диаграмме классов для представления любого сотрудника, который является элементом бизнес-системы и взаимодействует с другими сотрудниками при реализации бизнес-процесса. Этот класс также может быть изображен в форме прямоугольника класса со стереотипом <<worker>> или <<internalWorker>> (рис. 3.4, а).
- Сотрудник для связи с окружением (caseworker) – класс, служащий для представления в бизнес-системе такого сотрудника, который, являясь элементом бизнес-системы, непосредственно взаимодействует с актерами (бизнес-актерами) при реализации бизнес-процесса. Этот класс также может быть изображен в форме прямоугольника класса со стереотипом <<caseWorker>> (рис. 3.4, б).
- Бизнес-сущность (business entity) — специальный случай класса -сущности, который также не инициирует никаких сообщений. Этот класс служит для сохранения информации о результатах выполнения бизнес-процесса в моделируемой бизнес-системе или организации. Этот класс также может быть изображен в форме прямоугольника класса со стереотипом <<business entity>> (рис. 3.4, в).

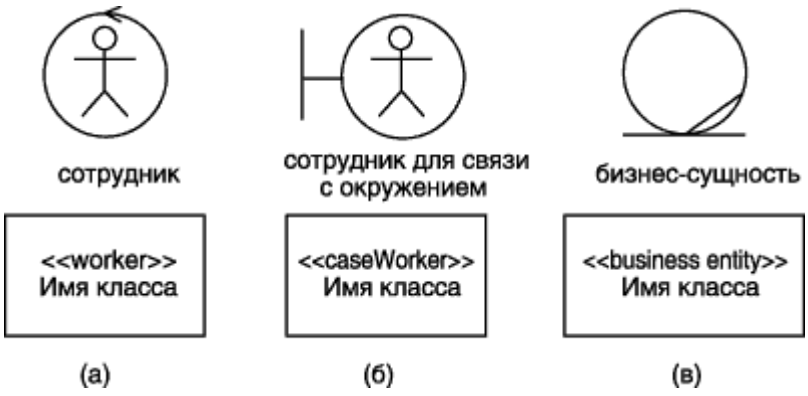

Рис. 3.4. Графическое изображение классов для моделирования бизнес-систем

Интерфейс

Интерфейс (interface) — именованное множество операций, которые характеризуют поведение отдельного элемента модели.

Интерфейс в контексте языка UML является специальным случаем класса, у которого имеются операции, но отсутствуют атрибуты. Для обозначения интерфейса используется специальный графический символ окружность или стандартный способ – прямоугольник класса со стереотипом <<interface>> (рис. 3.5).

На диаграмме вариантов использования интерфейс изображается в виде маленького круга, рядом с которым записывается его имя (рис. 3.5, а). В качестве имени может использоваться существительное, которое характеризует соответствующую информацию или сервис, например, " Датчик температуры ", " Форма ввода ", " Сирена ", " Видеокамера " (рис. 3.5, б). С учетом языка реализации модели имя интерфейса, как и имена других классов, рекомендуется записывать на английском и начинать с заглавной буквы I, например, ITemperatureSensor, IsecureInformation (рис. 3.5, в).

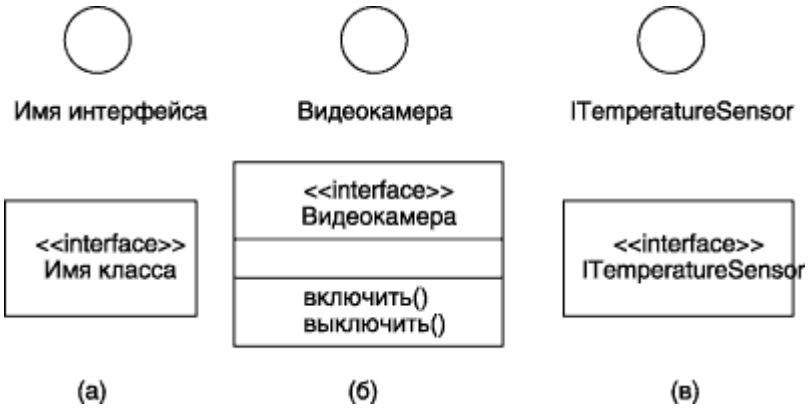

Рис. 3.5. Примеры графического изображения интерфейсов на диаграммах классов

Интерфейсы на диаграмме служат для спецификации таких элементов модели, которые видимы извне, но их внутренняя структура остается скрытой от клиентов. Интерфейсы не могут содержать ни атрибутов, ни состояний, ни направленных ассоциаций. Они содержат только операции без указания особенностей их реализации. Формально интерфейс не только отделяет спецификацию операций системы от их реализации, но и определяет общие границы проектируемой системы. В последующем интерфейс может быть уточнен явным указанием тех операций, которые специфицируют отдельный аспект поведения системы. Графическое изображение интерфейсов в форме окружности могут использоваться и на других типах канонических диаграмм, например, диаграммах компонентов и развертывания.