ЛАБОРАТОРНАЯ РАБОТА № 5
ФОРМИРОВАНИЕ МОДЕЛИ КЛАССОВ ПРОГРАММНОГО СРЕДСТВА С ИСПОЛЬЗОВАНИЕМ UML

1 Цель занятия

Научиться формировать диаграммы классов (class diagram) для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования.

2 Общие теоретические сведения

2.1. Класс

Класс (class) в языке UML служит для обозначения множества объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов. Графически класс изображается в виде прямоугольника, который дополнительно может быть разделен горизонтальными линиями на разделы или секции. В этих разделах могут указываться имя класса, атрибуты (переменные) и операции (методы).

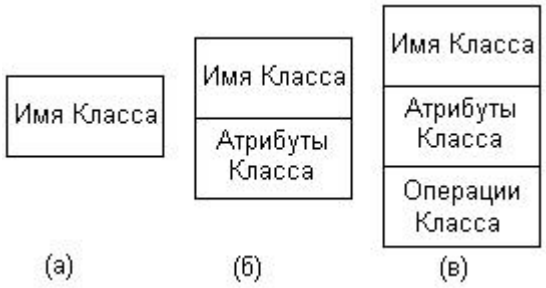

Рис. 1 Графическое изображение класса на диаграмме классов

Обязательным элементов обозначения класса является его имя. На начальных этапах разработки диаграммы отдельные классы могут обозначаться простым прямоугольником с указанием только имени соответствующего класса (рис. 1, а). По мере проработки отдельных компонентов диаграммы описания классов дополняются атрибутами (рис. 1, б) и операциями (рис. 1, в).

Предполагается, что окончательный вариант диаграммы содержит наиболее полное описание классов, которые состоят из трех разделов или секций. Иногда в обозначениях классов используется дополнительный четвертый раздел, в котором приводится семантическая информация справочного характера или явно указываются исключительные ситуации.

Даже если секция атрибутов и операций является пустой, в обозначении класса она выделяется горизонтальной линией, чтобы сразу отличить класс от других элементов языка UML. Примеры графического изображения классов на диаграмме классов приведены на рис. 2. В первом случае для класса "Прямоугольник" (рис. 2, а) указаны только его атрибуты — точки на координатной плоскости, которые определяют его расположение. Для класса "Окно" (рис. 2, б) указаны только его операции, секция атрибутов оставлена пустой. Для класса "Счет" (рис. 2, в)

дополнительно изображена четвертая секция, в которой указано исключение — отказ от обработки просроченной кредитной карточки.

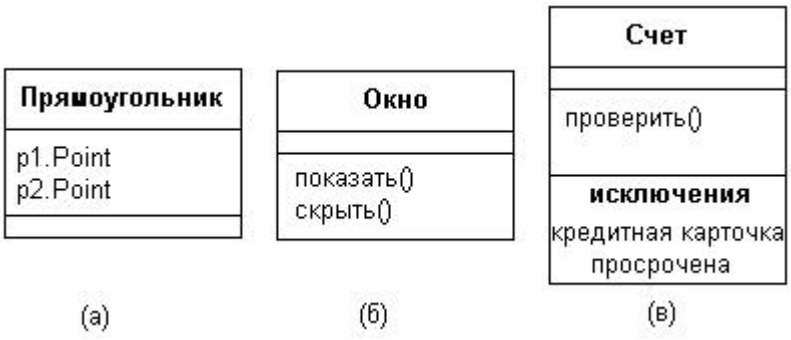

Рис. 2 Примеры графического изображения классов на диаграмме

В верхней секции указывается имя класса.

2.1.1 Атрибуты класса

Во второй сверху секции прямоугольника класса записываются его атрибуты (attributes) или свойства. 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:

– <квантор видимости><имя атрибута>[кратность]:

– <тип атрибута> = <исходное значение>{строка-свойство}

Квантор видимости может принимать одно из трех возможных значений и, соответственно, отображается 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 невиден для всех классов, за исключением подклассов данного класса.

И, наконец, знак "-" обозначает атрибут с областью видимости типа закрытый (private). Атрибут с этой областью видимости недоступен или невиден для всех классов без исключения.

Квантор видимости может быть опущен. В этом случае его отсутствие просто означает, что видимость атрибута не указывается.

Имя атрибута представляет собой строку текста, которая используется в качестве идентификатора соответствующего атрибута и поэтому должна быть уникальной в пределах данного класса. Имя атрибута является единственным обязательным элементом синтаксического обозначения атрибута.

Кратность атрибута характеризует общее количество конкретных атрибутов данного типа, входящих в состав отдельного класса. В общем случае кратность записывается в форме строки текста в квадратных скобках после имени соответствующего атрибута:

[нижняя_граница1 .. верхняя_граница1, нижняя_граница2.. верхняя_граница2, ..., нижняя_границаk .. верхняя_границаk],

где нижняя_граница и верхняя_граница являются положительными целыми числами, каждая пара которых служит для обозначения отдельного замкнутого интервала целых чисел, у которого нижняя (верхняя) граница равна значению нижняя_граница (верхняя_граница). В целом данное условное обозначение кратности соответствует теоретико-множественному объединению соответствующих интервалов. В качестве верхней_границы может использоваться специальный символ "*", который означает произвольное положительное целое число. Другими словами, это означает неограниченное сверху значение кратности соответствующего атрибута.

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.

Если кратность атрибута не указана, то по умолчанию принимается ее значение равное 1..1, т. е. в точности 1.

Тип атрибута представляет собой выражение, семантика которого определяется языком спецификации соответствующей модели. 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 некоторого начального значения для соответствующего атрибута в момент создания отдельного экземпляра класса.

При задании атрибутов могут быть использованы две дополнительные синтаксические конструкции – это подчеркивание строки атрибута и пояснительный текст в фигурных скобках.

Подчеркивание строки атрибута означает, что соответствующий атрибут может принимать подмножество значений из некоторой области значений атрибута, определяемой его типом. Эти значения можно рассматривать как набор однотипных записей или массив, которые в совокупности характеризуют каждый объект класса.

Строка-свойство служит для указания значений атрибута, которые не могут быть изменены в программе при работе с данным типом объектов. 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.

2.1.2 Операция

В третьей сверху секции прямоугольника записываются операции или методы класса. Операция (operation) представляет собой некоторый сервис, предоставляющий каждый экземпляр класса по определенному требованию. Совокупность операций характеризует функциональный аспект поведения 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:

– <квантор видимости><имя операции>(список параметров):

– <выражение типа возвращаемого значения>{строка-свойство}

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

Квантор видимости для операции может быть опущен. В этом случае его отсутствие просто означает, что видимость операции не указывается.

Имя операции представляет собой строку текста, которая используется в качестве идентификатора соответствующей операции и поэтому должна быть уникальной в пределах данного класса. Имя атрибута является единственным обязательным элементом синтаксического обозначения операции.

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

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

Здесь вид 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. Выражение типа является зависимой от конкретного языка программирования спецификацией типа возвращаемого значения для соответствующего формального параметра. Наконец, значение по умолчанию в общем случае представляет собой выражение для значения формального параметра, синтаксис которого зависит от конкретного языка программирования и подчиняется принятым в нем ограничениям.

Выражение типа возвращаемого значения также является зависимой от языка реализации спецификацией типа или типов значений параметров, которые возвращаются объектом после выполнения соответствующей операции. Двоеточие и выражение типа возвращаемого значения могут быть опущены, если операция не возвращает никакого значения. Для указания кратности возвращаемого значения данная спецификация может быть записана в виде списка отдельных выражений.

Строка-свойство служит для указания значений свойств, которые могут быть применены к данному элементу. Строка-свойство не является обязательной, она может отсутствовать, если никакие свойства не специфицированы.

Операция с областью действия на весь класс показывается подчеркиванием имени и строки выражения типа. По умолчанию под областью операции понимается объект класса. В этом случае имя и строка выражения типа операции не подчеркиваются.

Операция, которая не может изменять состояние системы и, соответственно, не имеет никакого побочного эффекта, обозначается строкой-свойством "{запрос}" ("{query}"). В противном случае операция может изменять состояние системы, хотя нет никаких гарантий, что она будет это делать.

Для повышения производительности системы одни операции могут выполняться параллельно или одновременно, а другие – только последовательно. В этом случае для указания параллельности выполнения операции используется строка-свойство вида "{concurrency = имя}", где имя может принимать одно из следующих значений: последовательная (sequential), параллельная (concurrent), охраняемая (guarded). При этом придерживаются следующей семантики для данных значений:

– последовательная (sequential) – для данной операции необходимо обеспечить ее единственное выполнение в системе, одновременное выполнение других операций может привести к ошибкам или нарушениям целостности объектов класса;

– параллельная (concurrent) – данная операция в силу своих особенностей может выполняться параллельно с другими операциями в системе, при этом параллельность должна поддерживаться на уровне реализации модели;

– охраняемая (guarded) – все обращения к данной операции должны быть строго упорядочены во времени с целью сохранения целостности объектов данного класса, при этом могут быть приняты дополнительные меры по контролю исключительных ситуаций на этапе ее выполнения.

С целью сокращения обозначений допускается использование одного имени в качестве строки-свойства для указания соответствующего значения параллельности. Отсутствие данной строки-свойства означает, что семантика параллельности для операции не определена. Поэтому следует предположить худший с точки зрения производительности случай, когда данная операция требует последовательного выполнения.

Появление сигнатуры операции на самом верхнем уровне объявляет эту операцию на весь класс, при этом данная операция наследуется всеми потомками данного класса. Если в некотором классе операция не выполняется (т.е. некоторый метод не применяется), то такая операция может быть помечена как абстрактная "{abstract}". Другой способ показать абстрактный характер операции — записать ее сигнатуру курсивом. Подчиненное появление записи данной операции без свойства {абстрактная} указывает на тот факт, что соответствующий класс-потомок может выполнять данную операцию в качестве своего "метода.

Если для некоторой операции необходимо дополнительно указать особенности ее реализации (например, алгоритм), то это может быть сделано в форме примечания, записанного в виде текста, который присоединяется к записи операции в соответствующей секции класса. Если объекты класса принимают и реагируют на некоторый сигнал, то запись данной операции помечается ключевым словом "сигнал" ("signal"). Это обозначение равнозначно обозначению некоторой операции. Реакция объекта на прием сигнала может быть показана в виде некоторого автомата. Кроме других случаев эта нотация может быть использована, чтобы показать реакцию объектов класса на ошибочные ситуации или исключения, которые могут моделироваться как сигналы или сообщения.

Поведение операции может быть указано дополнительно в форме присоединенного к операции примечания. В этом случае текст примечания заключается в скобки, если он представляет собой формальную спецификацию на некотором языке программирования и соответствует элементу "семантическое ограничение языка UML".

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

2.3. Отношения между классами

Кроме внутреннего устройства или структуры классов на соответствующей диаграмме указываются различные отношения между классами. При этом совокупность типов таких отношений фиксирована в языке UML и предопределена семантикой этих типов отношений. Базовыми отношениями или связями в языке UML являются:

– отношение зависимости (dependency relationship);

– отношение ассоциации (association relationship);

– отношение обобщения (generalization relationship);

– отношение реализации (realization relationship).

Каждое из этих отношений имеет собственное графическое представление на диаграмме, которое отражает взаимосвязи между объектами соответствующих классов.

2.2.1 Отношение зависимости

Отношение зависимости в общем случае указывает некоторое семантическое отношение между двумя элементами модели или двумя множествами таких элементов, которое не является отношением ассоциации, обобщения или реализации. Оно касается только самих элементов модели и не требует множества отдельных примеров для пояснения своего смысла. Отношение зависимости используется в такой ситуации, когда некоторое изменение одного элемента модели может потребовать изменения другого зависимого от него элемента модели.

Отношение зависимости графически изображается пунктирной линией между соответствующими элементами со стрелкой на одном из ее концов ("—>" или "<—"). На диаграмме классов данное отношение связывает отдельные классы между собой, при этом стрелка направлена от класса-клиента зависимости к независимому классу или классу-источнику (рис. 3). На данном рисунке изображены два класса: Класс_А и Кяасс_Б, при этом Класс_Б является источником некоторой зависимости, а Класс_А — клиентом этой зависимости.

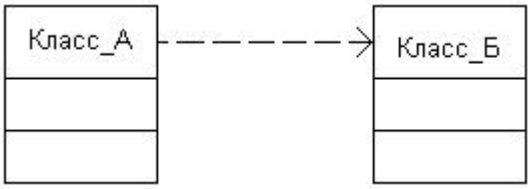

Рис. 3 Графическое изображение отношения зависимости на диаграмме классов

В качестве класса-клиента и класса-источника зависимости могут выступать целые множества элементов модели. В этом случае одна линия со стрелкой, выходящая от источника зависимости, расщепляется в некоторой точке на несколько отдельных линий, каждая из которых имеет отдельную стрелку для класса-клиента. Например, если функционирование Класса_С зависит от особенностей реализации Класса_А и Класса_Б, то данная зависимость может быть изображена следующим образом (рис. 5.4).

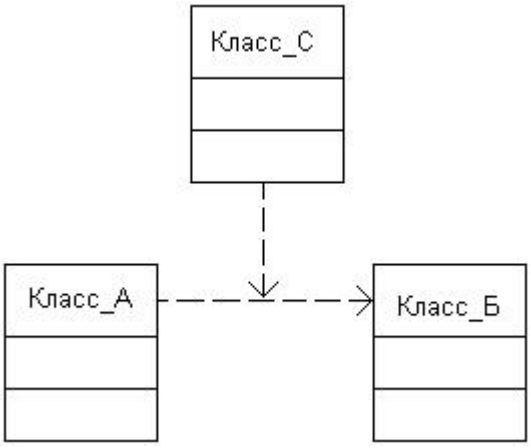

Рис. 4 Графическое представление зависимости между классом-клиентом (Класс_С) и классами-источниками (Класс_А и Класс_Б)

Стрелка может помечаться необязательным, но стандартным ключевым словом в кавычках и необязательным индивидуальным именем. Для отношения зависимости предопределены ключевые слова, которые обозначают некоторые специальные виды зависимостей. Эти ключевые слова (стереотипы) записываются в кавычках рядом со стрелкой, которая соответствует данной зависимости.

2.2.2 Отношение ассоциации

Отношение ассоциации соответствует наличию некоторого отношения между классами. Данное отношение обозначается сплошной линией с дополнительными специальными символами, которые характеризуют отдельные свойства конкретной ассоциации. В качестве дополнительных специальных символов могут использоваться имя ассоциации, а также имена и кратность классов-ролей ассоциации. Имя ассоциации является необязательным элементом ее обозначения. Если оно задано, то записывается с заглавной (большой) буквы рядом с линией соответствующей ассоциации.

Наиболее простой случай данного отношения – бинарная ассоциация. Она связывает в точности два класса и, как исключение, может связывать класс с самим собой. Для бинарной ассоциации на диаграмме может быть указан порядок следования классов с использованием треугольника в форме стрелки рядом с именем данной ассоциации. Направление этой стрелки указывает на порядок классов, один из которых является первым (со стороны треугольника), а другой — вторым (со стороны вершины треугольника). Отсутствие данной стрелки рядом с именем ассоциации означает, что порядок следования классов в рассматриваемом отношении не определен.

N-арная ассоциация графически обозначается ромбом, от которого ведут линии к символам классов данной ассоциации. В этом случае ромб соединяется с символами соответствующих классов сплошными линиями. Обычно линии проводятся от вершин ромба или от середины его сторон. Имя N-арной ассоциации записывается рядом с ромбом соответствующей ассоциации.

Порядок классов в N-арной ассоциации, в отличие от порядка множеств в отношении, на диаграмме не фиксируется. Некоторый класс может быть присоединен к ромбу пунктирной линией. Это означает, что данный класс обеспечивает поддержку свойств соответствующей N-арной ассоциации, а сама N-арная ассоциация имеет атрибуты, операции и/или ассоциации. N-арная ассоциация не может содержать символ агрегации ни для какой из своих ролей.

Следующий элемент обозначений – кратность отдельных классов, являющихся концами ассоциации. Кратность отдельного класса обозначается в виде интервала целых чисел, аналогично кратности атрибутов и операций классов. Интервал записывается рядом с концом ассоциации и для N-арной ассоциации означает потенциальное число отдельных экземпляров или значений кортежей этой ассоциации, которые могут иметь место, когда остальные N-1 экземпляров или значений классов фиксированы.

Что касается других свойств отношения, ассоциации, то в случае их наличия, они могут рассматриваться в качестве атрибутов класса ассоциации и могут быть указаны на диаграмме обычным для класса способом в соответствующей секции прямоугольника класса.

Частным случаем отношения ассоциации является так называемая исключающая ассоциация (Xor-association). Семантика данной ассоциации указывает на тот факт, что из нескольких потенциально возможных вариантов данной ассоциации в каждый момент времени может использоваться только один ее экземпляр. На диаграмме классов исключающая ассоциация изображается пунктирной линией, соединяющей две и более ассоциации, рядом с которой записывается строка-ограничение "{xor}".

Например, счет в банке может быть открыт для клиента, в качестве которого может выступать физическое лицо (индивидум) или компания, что изображается с помощью исключающей ассоциации (рис. 5).

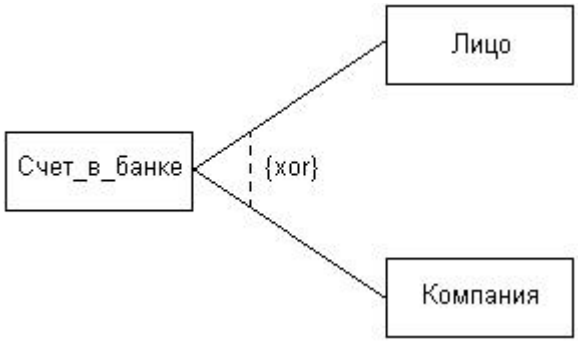

Рис. 5 Графическое изображение исключающей ассоциации между тремя классами

Специальной формой или частным случаем отношения ассоциации является отношение агрегации, которое, в свою очередь, тоже имеет специальную форму — отношение композиции. Поскольку эти отношения имеют свои специальные

обозначения и относятся к базовым понятиям языка UML, рассмотрим их последовательно.

2.2.3 Отношение агрегации

Отношение агрегации имеет место между несколькими классами в том случае, если один из классов представляет собой некоторую сущность, включающую в себя в качестве составных частей другие сущности.

Данное отношение имеет фундаментальное значение для описания структуры сложных систем, поскольку применяется для представления системных взаимосвязей типа "часть-целое". Раскрывая внутреннюю структуру системы, отношение агрегации показывает, из каких компонентов состоит система и как они связаны между собой. С точки зрения модели отдельные части системы могут выступать как в виде элементов, так и в виде подсистем, которые, в свою очередь, тоже могут образовывать составные компоненты или подсистемы. Это отношение по своей сути описывает декомпозицию или разбиение сложной системы на более простые составные части, которые также могут быть подвергнуты декомпозиции, если в этом возникнет необходимость в последующем.

Очевидно, что рассматриваемое в таком аспекте деление системы на составные части представляет собой некоторую иерархию ее компонентов, однако данная иерархия принципиально отличается от иерархии, порождаемой отношением обобщения. Отличие заключается в том, что части системы никак не обязаны наследовать ее свойства и поведение, поскольку являются вполне самостоятельными сущностями. Более того, части целого обладают своими собственными атрибутами и операциями, которые существенно отличаются от атрибутов и операций целого.

В качестве примера отношения агрегации рассмотрим взаимосвязь типа "часть-целое".

Рис. 6 Графическое изображение отношения агрегации в языке UML

Еще одним примером отношения агрегации может служить известное каждому из читателей деление персонального компьютера на составные части: системный блок, монитор, клавиатуру и мышь. Используя обозначения языка UML, компонентный состав ПК можно представить в виде соответствующей диаграммы классов (рис. 7), которая в данном случае иллюстрирует отношение агрегации.

Рис. 7 Диаграмма классов для иллюстрации отношения агрегации на примере ПК

2.2.4 Отношение композиции

Отношение композиции, как уже упоминалось ранее, является частным случаем отношения агрегации. Это отношение служит для выделения специальной формы отношения "часть-целое", при которой составляющие части в некотором смысле находятся внутри целого. Специфика взаимосвязи между ними заключается в том, что части не могут выступать в отрыве от целого, т. е. с уничтожением целого уничтожаются и все его составные части.

Графически отношение композиции изображается сплошной линией, один из концов которой представляет собой закрашенный внутри ромб. Этот ромб указывает на тот из классов, который представляет собой класс-композицию или "целое". Остальные классы являются его "частями" (рис. 8).

Рис. 8 Графическое изображение отношения композиции в языке UML

В качестве дополнительных обозначений для отношений композиции и агрегации могут использоваться дополнительные обозначения, применяемые для отношения ассоциации. А именно, указание кратности класса ассоциации и имени данной ассоциации, которые не являются обязательными.

2.2.5 Отношение обобщения

Отношение обобщения является обычным таксономическим отношением между более общим элементом (родителем или предком) и более частным или специальным элементом (дочерним или потомком). Данное отношение может использоваться для представления взаимосвязей между пакетами, классами, вариантами использования и другими элементами языка UML.

Применительно к диаграмме классов данное отношение описывает иерархическое строение классов и наследование их свойств и поведения. При этом предполагается, что класс-потомок обладает всеми свойствами и поведением класса-предка, а также имеет свои собственные свойства и поведение, которые отсутствуют у класса-предка. На диаграммах отношение обобщения обозначается сплошной линией с треугольной стрелкой на одном из концов (рис. 9). Стрелка указывает на более общий класс (класс-предок или суперкласс), а ее отсутствие – на более специальный класс (класс-потомок или подкласс).

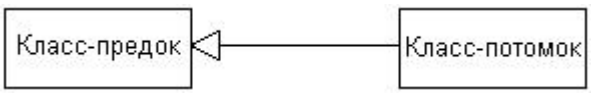

Рис. 9 Графическое изображение отношения обобщения в языке UML

Как правило, на диаграмме может указываться несколько линий для одного отношения обобщения, что отражает его таксономический характер. В этом случае более общий класс разбивается на подклассы одним отношением Обобщения.

С целью упрощения обозначений на диаграмме классов совокупность линий, обозначающих одно и то же отношение обобщения, может быть объединена в одну линию. В этом случае данные отдельные линии изображаются сходящимися к единственной стрелке, имеющей с ними общую точку пересечения (рис. 10).

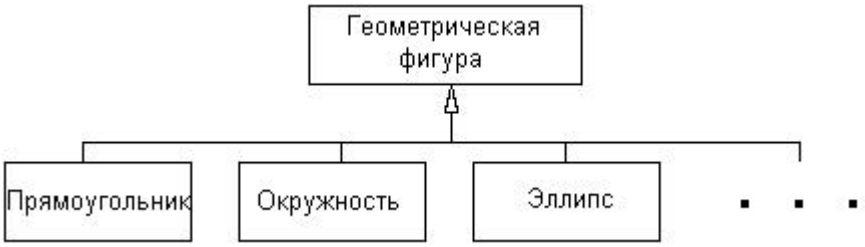

Рис. 10 Вариант графического изображения отношения обобщения классов для случая объединения отдельных линий

Рядом со стрелкой обобщения может размещаться строка текста, указывающая на некоторые дополнительные свойства этого отношения. Данный текст будет относиться ко всем линиям обобщения, которые идут к классам-потомкам. Другими словами, отмеченное свойство касается всех подклассов данного отношения. При этом текст следует рассматривать как ограничение, и тогда он записывается в фигурных скобках.

В качестве ограничений могут быть использованы следующие ключевые слова языка UML:

{complete} – означает, что в данном отношении обобщения специфицированы все классы-потомки, и других классов-потомков у данного класса-предка быть не может. На соответствующей диаграмме классов это можно указать явно, записав рядом с линией обобщения данную строку-ограничение;

{disjoint} – означает, что классы-потомки не могут содержать объектов, одновременно являющихся экземплярами двух или более классов. В этом случае рядом с линией обобщения можно записать данную строку-ограничение;

{incomplete} – означает случай, противоположный первому. А именно, предполагается, что на диаграмме указаны не все классы-потомки. В последующем возможно восполнить их перечень не изменяя уже построенную диаграмму;

{overlapping} – означает, что отдельные экземпляры классов-потомков могут принадлежать одновременно нескольким классам.

С учетом возможности использования строк-ограничений диаграмма классов (рис. 10) может быть изображена без многоточий и без потери информации (рис. 11).

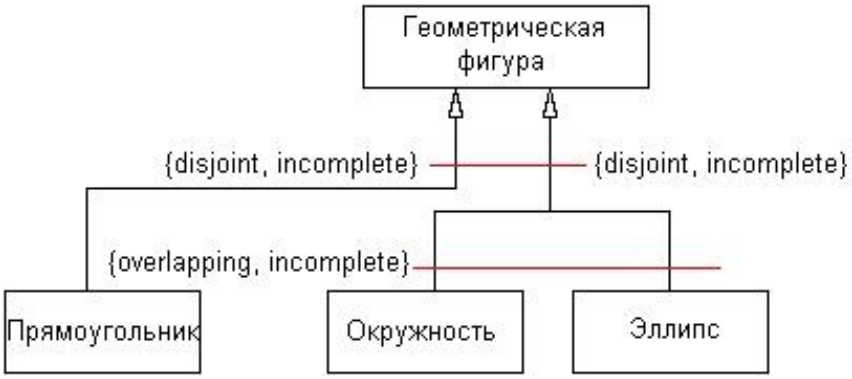

Рис. 11 Вариант графического изображения отношения обобщения классов с использованием строки-ограничения

2.3. Интерфейсы

Интерфейсы являются элементами диаграммы вариантов использования и были рассмотрены в лабораторной работе № 1. Однако при построении диаграммы классов отдельные интерфейсы могут уточняться и в этом случае для их изображения используется специальный графический символ – прямоугольник класса с ключевым словом или стереотипом "interface" (рис. 12). При этом секция атрибутов у прямоугольника отсутствует, а указывается только секция операций.

Рис. 12 Пример графического изображения интерфейса на диаграмме классов

2.4. Объекты

Объект (object) является отдельным экземпляром класса, который создается на этапе выполнения программы. Он имеет свое собственное имя и конкретные значения атрибутов. В силу самых различных причин может возникнуть необходимость показать взаимосвязи не только между классами модели, но и между отдельными объектами, реализующими эти классы. В данном случае может быть разработана диаграмма объектов, которая, хотя и не является канонической в метамодели языка UML, но имеет самостоятельное назначение.

Для графического изображения объектов используется такой же символ прямоугольника, что и для классов. Отличия проявляются при указании имен объектов, которые в случае объектов обязательно подчеркиваются (рис. 13). При этом запись имени объекта представляет собой строку текста "имя объекта:имя класса", разделенную двоеточием (рис. 13 а, б). Имя объекта может отсутствовать, в этом случае предполагается, что объект является анонимным, и двоеточие указывает на данное обстоятельство (рис. 13, г). Отсутствовать может и имя класса. Тогда указывается просто имя объекта (рис. 13, в). Атрибуты объектов принимают конкретные значения.

При изображении диаграммы объектов нужно помнить, что каждый объект представляет собой экземпляр соответствующего класса, а отношения между объектами описываются с помощью связей (links), которые являются экземплярами соответствующих отношений. При этом все связи изображаются сплошными линиями.

Рис. 13 Пример графического изображения объектов на диаграммах языка UML

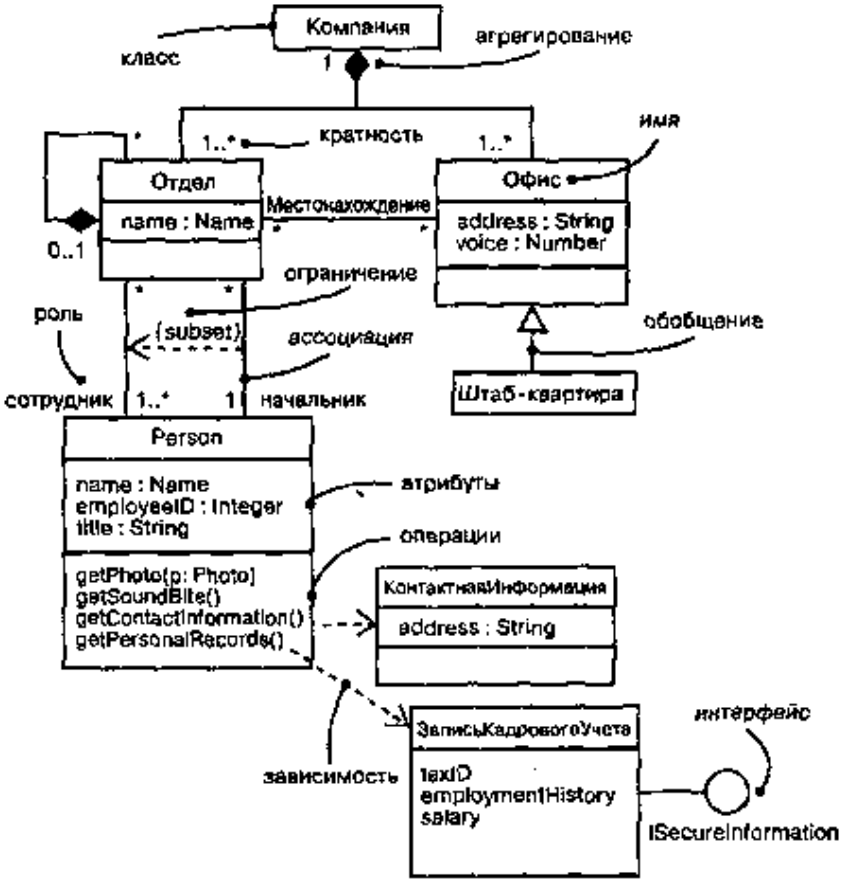

Рис. 14 Пример диаграммы классов кадрового учёта

3 Задачи для самостоятельного решения студентами

3.1 По варианту, разработанному на предыдущей лабораторной работе, выявить все классы, которые необходимы для программного средства, разработать дополнительные для связи и развития.

3.2 Разработать диаграмму классов с использованием программы Rational Rose, полностью учитывающую все варианты (в виде методов) диаграммы вариантов использования с использованием программы Rational Rose. Классы в диаграмме (не менее 7 штук) должны иметь не менее 3-х видов ассоциаций. Общее число свойств всех классов не менее 30, методов не менее 20. Все отношения должны быть поименованы. Должны присутствовать классы с атрибутами и классы с операциями.

4 Контрольные вопросы

1