Тема №3

Диаграмма классов

Рассматриваемые вопросы:

1. Понятие диаграммы классов
2. Класс
3. Отношения между классами
4. Интерфейсы
5. Объекты
6. Шаблоны или параметризованные классы
7. Рекомендации по построению диаграмм классов

Понятие диаграммы классов

Диаграмма классов (class diagram) служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования.

На диаграмме не указывается информация о временных аспектах функционирования системы.

Диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы.

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:

Класс

Обязательным элементом обозначения класса является его имя.

На начальных этапах разработки диаграммы отдельные классы могут обозначаться простым прямоугольником с указанием только имени соответствующего класса.

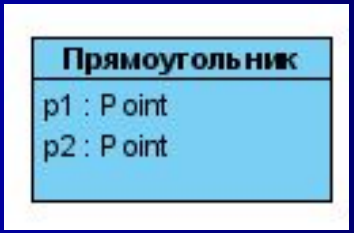
По мере проработки отдельных компонентов диаграммы описания классов дополняются атрибутами и операциями.

Иногда в обозначении классов используется дополнительный четвертый раздел, в котором приводится информация справочного характера или указываются исключительные ситуации.

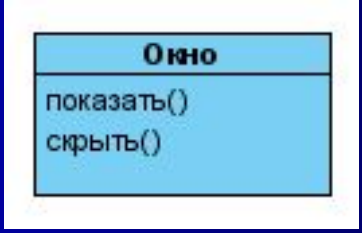
Класс

Примеры графических изображений классов:

- Для класса «Прямоугольник» указаны только его атрибуты – точки на координатной плоскости:



- Для класса «Окно» указаны только его операции:



Имя класса

Имя класса должно быть уникальным в пределах пакета, который описывается некоторой совокупностью диаграмм классов (возможно, одной диаграммой)

Рекомендуется в качестве имен использовать существительные, записанные по практическим соображениям без пробелов.

Именно имена классов образуют словарь предметной области при ООАП

Абстрактный класс

Класс может не иметь экземпляров или объектов. В этом случае он называется *абстрактным* классом, а для обозначения его имени используется наклонный шрифт (курсив):

В языке UML принято общее соглашение о том, что любой текст, относящийся к абстрактному элементу, записывается курсивом.

Атрибуты класса

В языке UML принята стандартная запись атрибутов класса.

Каждому атрибуту соответствует отдельная строка текста, которая состоит из *квантора видимости* атрибута, *имени* атрибута, его *кратности*, *типа значений* атрибута и, возможно, его *исходного значения*.

Квантор видимости может принимать одно из трех возможных значений и отображается при помощи специальных символов:

- символ «+» обозначает атрибут с областью видимости типа общедоступный (public);
- символ «#» обозначает атрибут с областью видимости типа защищенный (protected);
- символ «-» обозначает атрибут с областью видимости типа закрытый (private).

Атрибуты класса

Пример использования кванторов видимости:

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

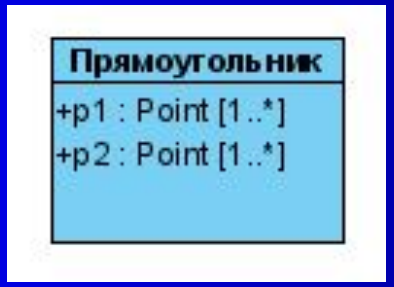
Эта ситуация отличается от принятых по умолчанию соглашений в традиционных языках программирования, когда отсутствие квантора видимости трактуется как public или private.

Атрибуты класса

Имя атрибута представляет собой строку текста, которая используется в качестве идентификатора соответствующего атрибута и должна быть уникальной в пределах данного класса.

Кратность атрибута характеризует общее количество конкретных атрибутов данного типа, входящих в состав отдельного класса.

Кратность записывается в форме строки текста в квадратных скобках после имени соответствующего атрибута:



Атрибуты класса

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 предполагает использовать для реализации данной модели.

Примеры типов атрибутов классов:

- цвет:Color – здесь *цвет* является именем атрибута. Color – именем типа данного атрибута;
- имя_сотрудника[1..2]: String – здесь *имя_сотрудника* является именем атрибута, который служит для представления информации об имени (и отчестве) сотрудника;

Атрибуты класса

- *видимость:Boolean* – здесь *видимость* есть имя абстрактного атрибута (курсив), который может характеризовать наличие визуального представления соответствующего класса на экране монитора.

Исходное значение служит для задания некоторого начального значения для соответствующего атрибута в момент создания отдельного экземпляра класса.

Если исходное значение не указано, то значение соответствующего атрибута не определено на момент создания нового экземпляра класса.

Атрибуты класса

Примеры исходных значений атрибутов:

- цвет:Color = (255, 0, 0) – в RGB-модели это соответствует чистому красному цвету в качестве исходного значения для данного атрибута;
- имя_сотрудника[1..2]:String = Иван Иванович
- *видимость:Boolean = истина* – может соответствовать ситуации, когда в момент создания экземпляра класса создается видимое на экране монитора окно, соответствующее данному объекту.

Атрибуты класса

Подчеркивание строки атрибута означает, что соответствующий атрибут может принимать подмножество значений.

Примером может служить задание атрибута в виде номер_счета:Integer, что может означать для объекта *Сотрудник* наличие некоторого подмножества счетов, общее количество которых заранее не фиксируется.

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

Например, строка-свойство в записи атрибута заработная_плата:Currency = {$500} может служить для обозначения фиксированной заработной платы.

Операция

Операция представляет собой некоторый сервис, предоставляющий каждый экземпляр класса по определенному требованию.

Совокупность операций характеризует функциональный аспект поведения класса.

Каждой операции класса соответствует строка, состоящая из:

- квантора видимости операции;
- имени операции;
- выражения типа возвращаемого операцией значения;
- строка-свойство данной операции.

Операция

Для повышения производительности одни операции могут выполняться параллельно или одновременно, а другие – только последовательно.

Для указания параллельности выполнения операции используется строка-свойство вида «{concurrency=*имя*}», где *имя* принимает следующие значения:

- *последовательная* (sequential);
- *параллельная* (concurrent);
- *охраняемая* (guarded) – все обращения к данной операции должны быть строго упорядочены во времени с целью сохранения целостности объектов этого класса.

Операция

Примеры записи операций:

- **+создать()** – операция по созданию отдельного объекта класса, которая является общедоступной и не содержит формальных параметров;
- **+нарисовать(форма:Многоугольник = прямоугольник, цвет_заливки:Color = (0,0,255))** – может обозначать операцию по изображению на экране монитора прямоугольной области синего цвета;
- **запросить_счет_клиента(номер_счета: Integer):Currency** – обозначает операцию по установлению наличия средств на текущем счете клиента банка.

Отношения между классами

Базовыми отношениями между классами в языке UML являются:

- ***Отношение зависимости (dependency relationship);***
- ***Отношение ассоциации (association relationship);***
- ***Отношение обобщения (generalization relationship);***
- ***Отношение реализации (realization relationship).***

Отношение зависимости

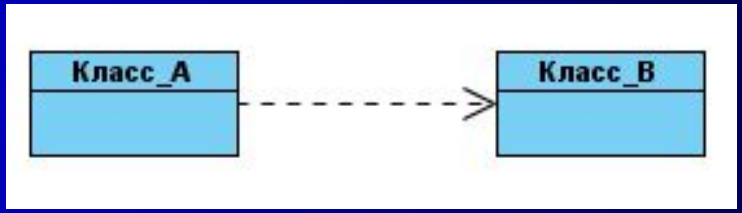
Отношение зависимости в общем случае указывает некоторое семантическое отношение между двумя классами, которое не является отношением ассоциации, обобщения или реализации.

Отношение зависимости используется в ситуации, когда изменение одного элемента модели может потребовать изменение другого зависимого от него элемента модели.

Отношение зависимости

Отношение зависимости графически изображается пунктирной линией между соответствующими элементами со стрелкой на одном из ее концов.

На диаграмме классов стрелка направлена от класса-клиента зависимости к независимому классу или классу источнику:



Отношение зависимости

Стрелка может помечаться необязательным, но стандартным ключевым словом в кавычках и необязательным индивидуальным именем.

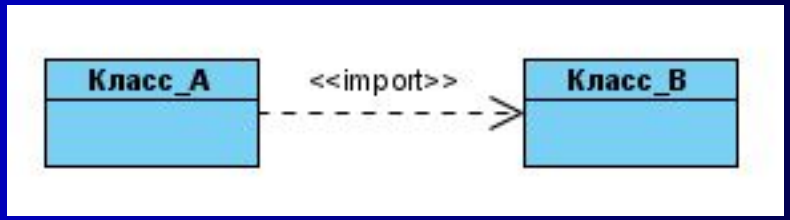
Для отношения зависимости предопределены ключевые слова, которые обозначают некоторые специальные виды зависимостей (*стереотипы*):

- «access» - служит для обозначения доступности открытых атрибутов и операций класса-источника для классов-клиентов;
- «derive» - атрибуты класса-клиента могут быть вычислены по атрибутам класса-источника;
- «refine» - указывает, что класс-клиент служит уточнением класса-источника в силу причин исторического характера;

Отношение зависимости

- «bind» - класс-клиент может использовать некоторый шаблон для своей последующей параметризации;
- «import» - открытые атрибуты и операции класса-источника становятся частью класса-клиента, как если бы они были объявлены непосредственно в нем.

Пример:



Отношение ассоциации

Отношение ассоциации соответствует наличию некоторого отношения между классами. Данное отношение обозначается сплошной линией с дополнительными специальными символами:

- имя ассоциации;
- имена и кратность классов-ролей ассоциации.

Имя ассоциации является необязательным элементом ее обозначения.

Отношение ассоциации

Наиболее простой случай – бинарная ассоциация.

Она связывает в точности два класса и, как исключение, класс с самим собой.

Пример бинарной ассоциации – отношение между классами «Компания» и «Сотрудник»:

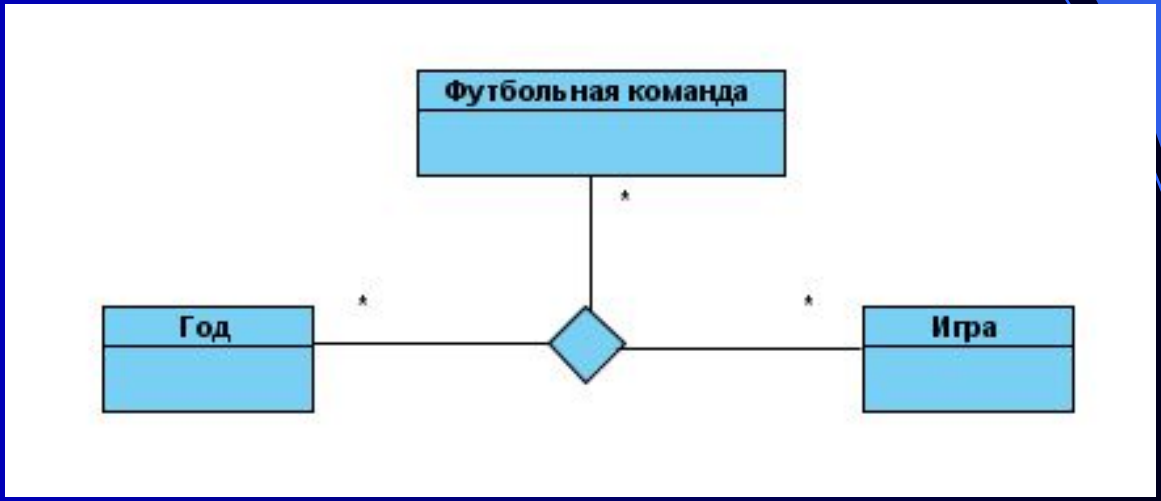
Отношение ассоциации

Тернарная ассоциация и ассоциация более высокой арности в общем случае называются *N-арной ассоциацией*.

Такая ассоциация связывает некоторым отношением три и более классов.

Бинарная ассоциация является частным случаем N-арной ассоциации, когда значение N=2, и имеет свое собственное обозначение.

Пример:



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.

Используется для описания структуры сложных систем, т.к. применяется для представления взаимосвязей типа «часть-целое».

Примером отношения агрегации является взаимосвязь между сущностью «Грузовой автомобиль» и компонентами «Двигатель», «Шасси», «Кабина», «Кузов».

Отношение агрегации

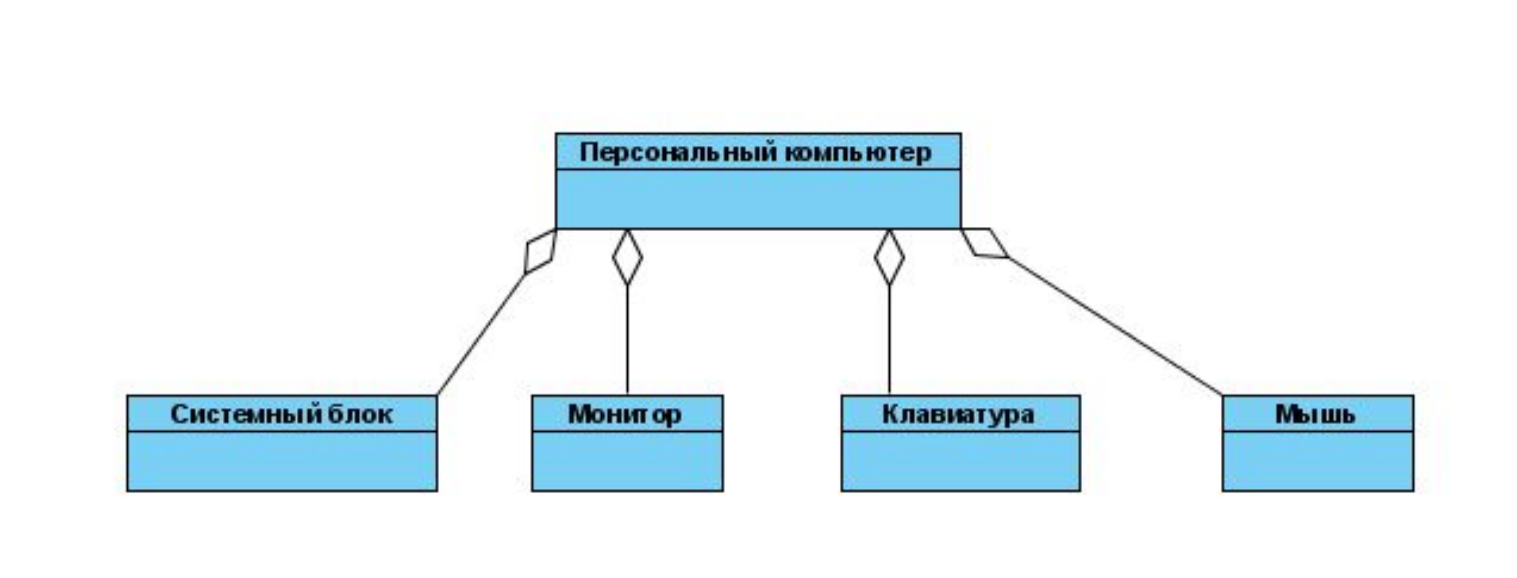
Графически отношение агрегации изображается сплошной линией, один из концов которой представляет собой незакрашенный ромб.

Ромб указывает на тот из классов, который представляет собой «целое». Остальные классы являются его «частями»:

Отношение агрегации

Примером отношения агрегации может служить деление персонального компьютера на составные части: системный блок, монитор, клавиатуру и мышь.

Используя обозначения языка UML, компонентный состав ПК можно представить в виде диаграммы классов:



Отношение композиции

Отношение композиции является частным случаем отношения агрегации.

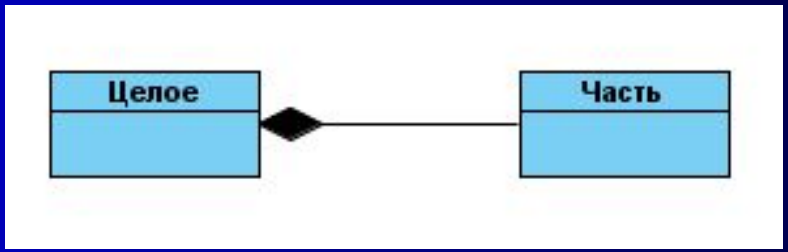
Это отношение служит для выделения специальной формы отношения «часть-целое», при которой составляющие части в некотором смысле находятся внутри целого.

Специфика: части не могут выступать в отрыве от целого, т. е. с уничтожением целого уничтожаются и все его составные части.

Отношение композиции

Графически отношение композиции изображается сплошной линией, один из концов которой представляет собой закрашенный внутри ромб.

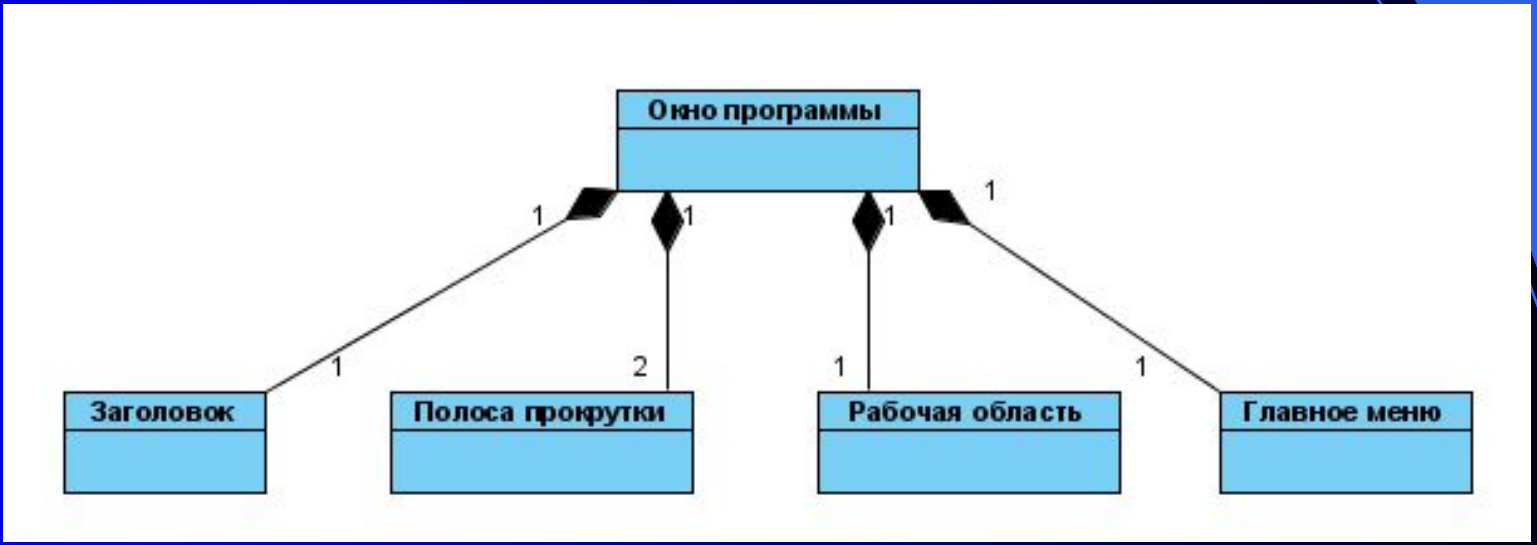
Этот ромб указывает на тот из классов, который представляет собой класс-композицию или «целое»:



Отношение композиции

В качестве дополнительных обозначений для отношений композиции и агрегации могут использоваться дополнительные обозначения, применяемые для отношения ассоциации: *кратность, имя.*

Пример:

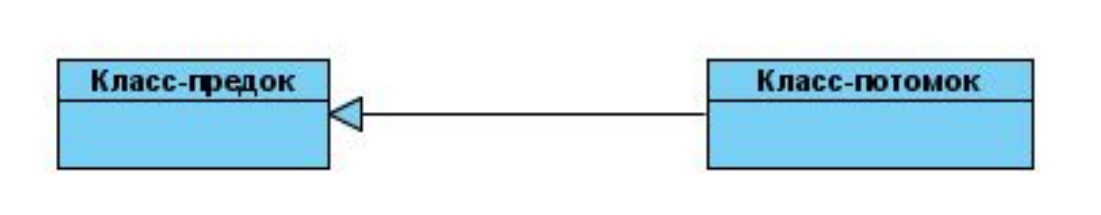


Отношение обобщения

Отношение обобщения является таксономическим отношением между более общим элементом и более частным или специальным элементом.

Данное отношение описывает иерархическое строение классов и наследование их свойств и поведения.

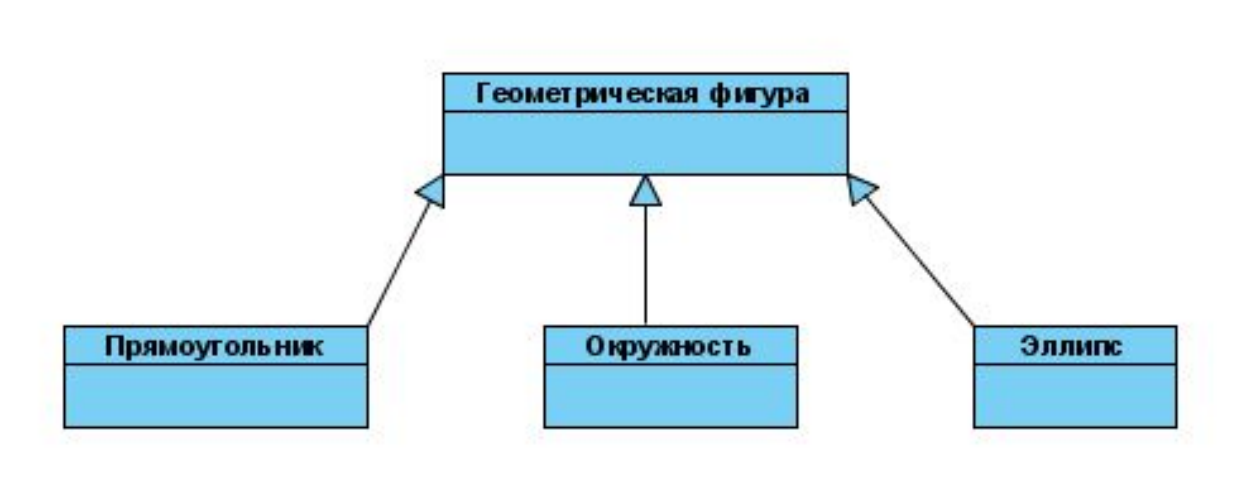
На диаграммах отношение обобщения обозначается сплошной линией с треугольной стрелкой на одном из концов. Стрелка указывает на более общий класс:



Отношение обобщения

На диаграмме может указываться несколько линий для одного отношения обобщения, что отражает его таксономический характер.

Пример: Класс «*Геометрическая_фигура_на_плоскости*» (абстрактный класс) выступает в качестве суперклассов для подклассов «Прямоугольник», «Окружность», «Эллипс»:



Отношение обобщения

Рядом со стрелкой обобщения могут указываться *ограничения*:

- {complete} – означает, что в данном отношении обобщения специфицированы все классы-потомки, и других классов-потомков у данного класса-предка быть не может.

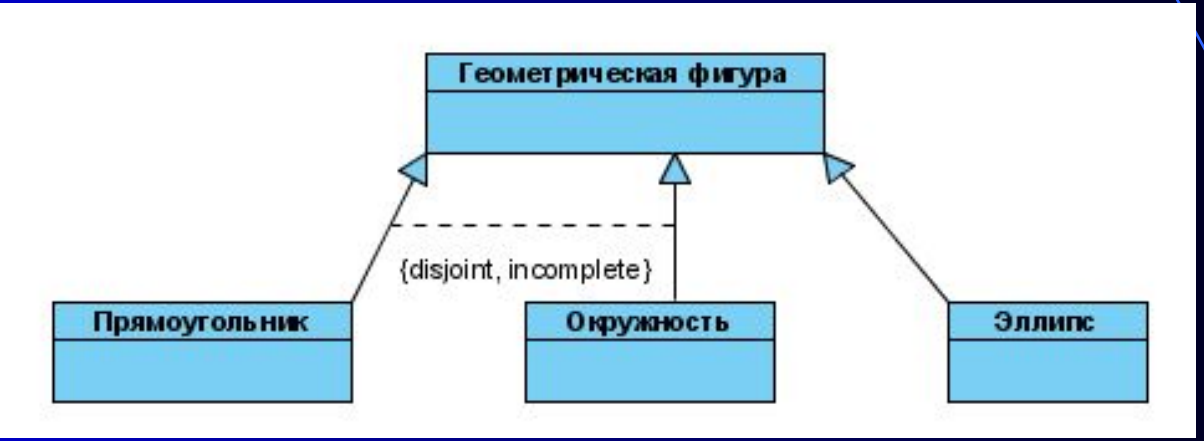
  Пример: класс «Клиент_банка» является предком для двух классов «Физическое_лицо» и «Компания».

- {disjoint} – означает, что классы-потомки не могут содержать объектов, одновременно являющихся экземплярами двух или более классов (см. предыдущий пример);

Отношение обобщения

- {incomplete} – означает случай, противоположный первому. Предполагается, что на диаграмме указаны не все классы-потомки;
- {overlapping} – означает, что отдельные экземпляры классов-потомков могут принадлежать одновременно нескольким классам.

Пример изображения ограничений:



Интерфейсы

Интерфейсы на диаграммах классов могут изображаться с помощью специального графического символа – прямоугольника класса с ключевым словом или стереотипом «interface»:

У интерфейсов секция атрибутов у прямоугольника отсутствует, а указывается только секция операций.

Диаграмма объектов

Объект (object) является отдельным экземпляром класса, который создается на этапе выполнения программы.

Он имеет собственное имя и конкретные значения атрибутов.

Диаграмма объектов не является канонической диаграммой UML, но имеет самостоятельное назначение

Для графического изображения объектов используется такой же символ прямоугольника, что и для классов.

Диаграмма объектов

Запись имени объекта представляет собой строку текста «имя_объекта:имя_класса», разделенную двоеточием:

Имя объекта может отсутствовать, в этом случае предполагается, что объект является анонимным, и двоеточие указывает на данное обстоятельство:

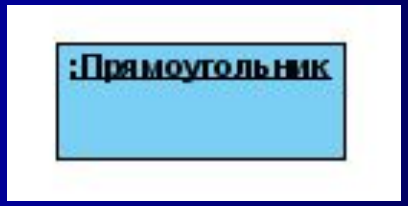
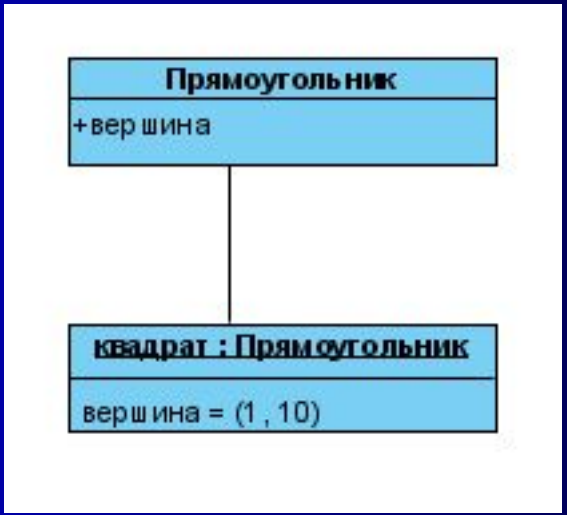


Диаграмма объектов

Атрибуты объектов принимают конкретные значения:



Шаблоны или параметризованные классы

Шаблон (template) или *параметризованный класс* (parametrized class) предназначен для обозначения такого класса, который имеет один (или более) нефиксированный формальный параметр.

Графически шаблон изображается прямоугольником, к верхнему правому углу которого присоединен маленький прямоугольник из пунктирных линий:

Шаблоны или параметризованные классы

Шаблон не может быть непосредственно использован в качестве класса, поскольку содержит неопределенные параметры.

Чаще всего в качестве шаблона выступает некоторый суперкласс, параметры которого уточняются в его классах-потомках.

Пример: Класс «Адрес» может быть получен из шаблона «Связный_список» на основе актуализации формальных параметров «S, k, l» фактическими атрибутами «улица, дом, квартира».

Шаблоны или параметризованные классы