**Диаграмма классов (class diagram)**

Вообще-то, понятие класса нам уже знакомо, но, пожалуй, не лишним будет поговорить о классах еще раз. Классики о классах говорят очень просто и понятно:

**Класс (class)** - категория вещей, которые имеют общие атрибуты и операции.

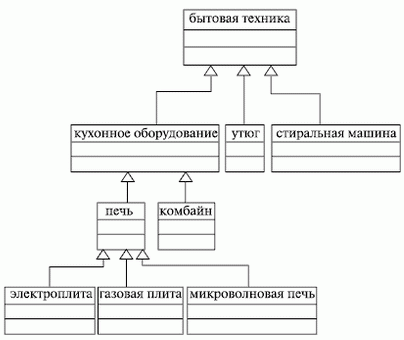
Продолжая тему, скажем, что классы - это строительные блоки любой объектно-ориентированной системы. Они представляют собой описание совокупности объектов с общими атрибутами, операциями, отношениями и семантикой. При проектировании объектно-ориентированных систем диаграммы классов обязательны.

Классы используются в процессе анализа предметной области для составления словаря предметной области разрабатываемой системы. Это могут быть как абстрактные понятия предметной области, так и классы, на которые опирается разработка и которые описывают программные или аппаратные сущности.

Диаграмма классов - это набор статических, декларативных элементов модели. Диаграммы классов могут применяться и при прямом проектировании, то есть в процессе разработки новой системы, и при обратном проектировании - описании существующих и используемых систем. Информация с диаграммы классов напрямую отображается в исходный код приложения - в большинстве существующих инструментов UML-моделирования возможна кодогенерация для определенного языка программирования (обычно Java или C++). Таким образом, диаграмма классов - конечный результат проектирования и отправная точка процесса разработки.

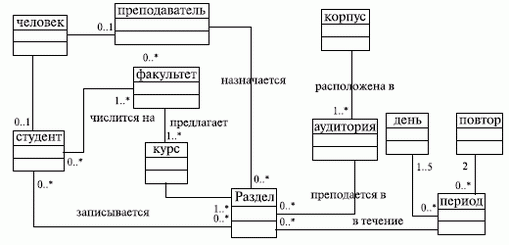
Но мы опять заговорились, а, как известно, лучше один раз увидеть, чем сто раз услышать. Мы уже знаем, как классы обозначаются в UML, но пока еще не видели ни одной диаграммы "с их участием". Итак, посмотрим на примеры диаграмм классов.

Первый пример ([рис. 2.5](http://www.intuit.ru/studies/courses/1007/229/lecture/3175?page=2#image.2.5)) весьма прост. Как видим, он, хоть и немного однобоко, иллюстрирует с помощью операции наследования или генерализации "генеалогическое древо" бытовой техники. Думаем, мы бы поняли смысл этой диаграммы, даже если бы ничего не знали о классах и не занимались программированием вообще.



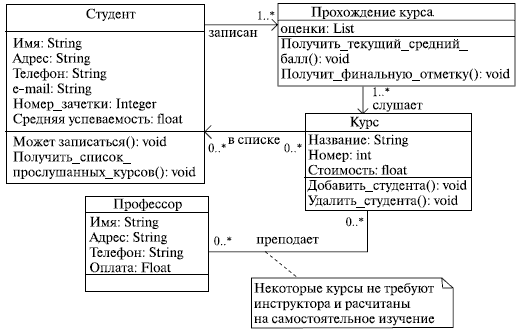
**Рис. 2.5.**

Рассмотрим еще пример ([рис. 2.6](http://www.intuit.ru/studies/courses/1007/229/lecture/3175?page=2#image.2.6)):



**Рис. 2.6.**

И опять-таки смысл этой диаграммы ясен без особых пояснений. Даже бегло рассмотрев ее, можно легко догадаться, что она описывает предметную область задачи об автоматизации работы некоего вуза или учебного центра. Обратите внимание на обозначения кратности на концах связей. А теперь немного усложним задачу ([рис. 2.7](http://www.intuit.ru/studies/courses/1007/229/lecture/3175?page=2#image.2.7)):



**Рис. 2.7.**

Как видим, здесь уже все более серьезно - кроме кратности обозначены свойства (и их типы) и операции, и вообще, эта диаграмма производит впечатление набора классов для реализации, а не просто описания предметной области, как предыдущие. Но, тем не менее, все равно все просто и понятно.

Отметим, что более детально о диаграмме классов мы поговорим в следующей лекции. Там мы подробно разберем нотацию этого вида диаграмм и познакомимся с улучшениями, внесенными текущей версией UML.

**Диаграмма объектов (object diagram)**

И снова, прежде чем говорить о новом виде диаграмм, введем определения нужных нам понятий. Итак, мы уже знаем, что такое класс. А что такое объект? Обратимся к классикам, которые об объектах говорят так же просто и понятно, как и о классах:

**Объект (object)** - экземпляр класса.

Zicom Mentor "говорит" об объектах более обстоятельно:

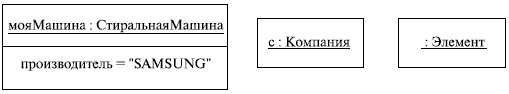
**Объект (object)** -

* конкретная материализация абстракции;
* сущность с хорошо определенными границами, в которой инкапсулированы состояние и поведение;
* экземпляр класса (вернее, классификатора - эктор, класс или интерфейс). Объект уникально идентифицируется значениями атрибутов, определяющими его состояние в данный момент времени.

"Второе" определение, по сути, просто расширяет "Бучевское". Да, действительно, объект - это экземпляр класса. Скажем, объектом класса "Микроволновая печь" из примера, приведенного выше, может быть и простейший прибор фирмы "Saturn" небольшой емкости и с механическим управлением, и навороченный агрегат с грилем, сенсорным управлением и системой трехмерного распределения энергии от Samsung или LG.

Еще пример - все мы являемся объектами класса "человек" и различимы между собой по таким признакам (значениям атрибутов), как имя, цвет волос, глаз, рост, вес, возраст и т. д. (в зависимости от того, какую задачу мы рассматриваем и какие свойства человека для нас в ней важны).

Как же обозначается объект в UML? А очень просто - объект, как и класс, обозначается прямоугольником, но его имя подчеркивается. Под словом имя здесь мы понимаем название объекта и наименование его класса, разделенные двоеточием. Для указания значений атрибутов объекта в его обозначении может быть предусмотрена специальная секция. Еще один нюанс состоит в том, что объект может быть анонимным: это нужно в том случае, если в данный момент не важно, какой именно объект данного класса принимает участие во взаимодействии. Примеры - на [рис. 2.8](http://www.intuit.ru/studies/courses/1007/229/lecture/3175?page=2#image.2.8).

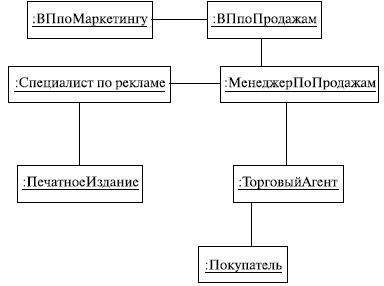


**Рис. 2.8.**

Итак, на определение основных понятий мы потратили довольно много времени, и пора бы уже вернуться к основному предмету нашего внимания - **диаграмме объектов**. Для чего нужны диаграммы объектов? Они показывают множество объектов - экземпляров классов (изображенных на диаграмме классов) и отношений между ними в некоторый момент времени. То есть диаграмма объектов - это своего рода снимок состояния системы в определенный момент времени, показывающий множество объектов, их состояния и отношения между ними в данный момент.

Таким образом, диаграммы объектов представляют статический вид системы с точки зрения проектирования и процессов, являясь основой для сценариев, описываемых диаграммами взаимодействия. Говоря другими словами, диаграмма объектов используется для пояснения и детализации диаграмм взаимодействия, например, диаграмм последовательностей. Впрочем, авторам курса очень редко доводилось применять этот тип диаграмм.

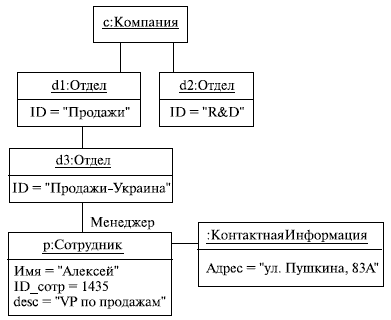
Приведем простейший пример такой диаграммы ([рис. 2.9](http://www.intuit.ru/studies/courses/1007/229/lecture/3175?page=2#image.2.9)).



**Рис. 2.9.**

О чем здесь идет речь, в принципе, понятно: некоторая фирма "раскручивает" новый товар или услугу. В этом процессе участвуют вице-президент по маркетингу, вице-президент по продажам, менеджер по продажам, торговый агент, специалист по рекламе, некое печатное издание и покупатель. Причем даже без указания сообщений, которыми обмениваются эти объекты, отлично видно, кто с кем взаимодействует. Кстати, обратите внимание, что на этой диаграмме все объекты анонимные!

Другой пример ([рис. 2.10](http://www.intuit.ru/studies/courses/1007/229/lecture/3175?page=2#image.2.10)).



**Рис. 2.10.**

Эта диаграмма тоже понятна в общих чертах даже без дополнительных объяснений. Здесь мы видим взаимосвязь объектов - организационных единиц в некоторой компании.

И наконец, последний пример ([рис. 2.11](http://www.intuit.ru/studies/courses/1007/229/lecture/3175?page=2#image.2.11)): диаграмма объектов учебной среды "Робот" для Turbo Pascal, в которой наше поколение школьников училось основам алгоритмизации.

Думаем, пока примеров достаточно и главной цели мы достигли - научили читателя различать диаграмму объектов. Кому-то может показаться, что мы уделили ей мало внимания, но, как уже было сказано выше, читатель вряд ли будет часто встречаться с этим типом диаграмм.

