**Объектная модель системы**

Объектная модель описывает структуру объектов, составляющих систему, их атрибуты, операции, взаимосвязи с другими объектами. В объектной модели должны быть отражены те понятия и объекты реального мира, которые важны для разрабатываемой системы. В объектной модели отражается прежде всего прагматика разрабатываемой системы, что выражается в использовании терминологии прикладной области, связанной с использованием разрабатываемой системы.

9. Объекты и классы. Атрибуты объектов.

По определению *объектом* можно называть понятие, абстракцию или любую вещь с четко очерченными границами, имеющую смысл в контексте рассматриваемой прикладной проблемы. Введение объектов преследует две цели:

* понимание прикладной задачи (проблемы);
* введение основы для реализации на компьютере.

Примеры объектов: форточка, Банк "Империал", Петр Сидоров, дело № 7461, сберкнижка и т.д.

Все объекты могут быть отличены один от другого: пусть у нас есть два яблока, имеющие одинаковый цвет, форму, вес и вкус; все равно это два яблока (а не одно), в чем легко убедиться, съев одно из них (другое останется). Между объектами можно установить отношение тождества: объекты, удовлетворяющие этому отношению, одинаковы (тождественны), как вышеупомянутые яблоки.

Два яблока из предыдущего примера принадлежат одному и тому же *классу* объектов (именно с этим связана их одинаковость). Цвет, форма, вес и вкус яблока - это его *атрибуты*: совокупность атрибутов и их значений (например, красное, овальное, стограммовое, кисло-сладкое) характеризует объект.

Все объекты одного и того же класса характеризуются одинаковыми наборами атрибутов. Однако объединение объектов в классы определяется не наборами атрибутов, а семантикой. Так, например, объекты конюшня и лошадь могут иметь одинаковые атрибуты: цена и возраст. При этом они могут относиться к одному классу, если рассматриваются в задаче просто как товар, либо к разным классам, что более естественно.

*Атрибут* - это значение, характеризующее объект в его классе. Примеры атрибутов: категория, баланс, кредит (атрибуты объектов класса счет); имя, возраст, вес (атрибуты объектов класса человек) и т.д.

Среди атрибутов различаются *постоянные атрибуты* (константы) и *переменные атрибуты*. Постоянные атрибуты характеризуют объект в его классе (например, номер счета, категория, имя человека и т.п.). Текущие значения переменных атрибутов характеризуют текущее *состояние* объекта (например, баланс счета, возраст человека и т.п.); изменяя значения этих атрибутов, мы изменяем состояние объекта.

10. Множественное наследование.

*Множественное наследование* позволяет классу иметь более одного суперкласса, наследуя свойства (атрибуты и операции) всех своих суперклассов. Класс, имеющий несколько суперклассов, называется объединенным классом. Свойства класса-предка, встречающегося более, чем один раз, в графе наследования, наследуются только в одном экземпляре. Конфликты между параллельными определениями порождают двусмысленности, которые должны разрешаться во время реализации. На практике следует избегать таких двусмысленностей или плохого понимания даже в тех случаях, когда конкретный язык программирования, выбранный для реализации системы, предоставляет возможность их разрешения, используя приоритеты или какие-либо другие средства.

В случае, если множественное наследование не поддерживается языком программирования, выбранным для реализации, оно может быть заменено одним из следующих способов.

Делегированием называется механизм реализации, в котором объект, ответственный за операцию, пересылает (делегирует) эту операцию другому объекту; в объектно-ориентированных языках делегирование реализуется путем присоединения методов непосредственно к объектам, а не к классам.

Возможны и другие способы замены множественного наследования. Во всех случаях при выборе способа замены множественного наследования нужно руководствоваться следующими правилами:

* если подкласс имеет несколько суперклассов, каждый из которых одинаково существен, лучше всего использовать делегирование
* если наиболее существенным является только один из суперклассов, а остальные не так важны, наилучшим способом является реализация множественного наследования через простое наследование и делегирование;
* если число возможных комбинаций групп наследуемых свойств невелико, можно использовать вложенное простое наследование; в случае большого числа комбинаций этот способ применять не следует;
* если один из суперклассов передает подклассу намного большее число свойств, чем остальные суперклассы, следует сохранить наследование по этому пути;
* если решено использовать вложенное простое наследование, то на первый уровень вложенности следует поместить наиболее существенный по передаче свойств суперкласс, затем наиболее существенный из оставшихся суперклассов и т.д.;
* следует избегать использования вложенного простого наследования , если это ведет к дублированию достаточно больших частей программы;
* следует помнить, что только вложенное простое наследование обеспечивает полную тождественность множественному наследованию.

11. Операции и методы

*Операция* - это функция (или преобразование), которую можно применять к объектам данного класса.

Все объекты данного класса используют один и тот же экземпляр каждой операции (т.е. увеличение количества объектов некоторого класса не приводит к увеличению количества загруженного программного кода). Объект, из которого вызвана операция, передается ей в качестве ее неявного аргумента (параметра).

Одна и та же операция может применяться к объектам разных классов: такая операция называется полиморфной, так как она может иметь разные формы для разных классов. Например, для объектов классов вектор и комплексное\_число можно определить операцию +; эта операция будет полиморфной, так как сложение векторов и сложение комплексных чисел разные операции.

Каждой операции соответствует метод - реализация этой операции для объектов данного класса. Таким образом, операция - это спецификация метода, метод - реализация операции. Например, в классе файл может быть определена операция печать (print). Эта операция может быть реализована разными методами: (а) печать двоичного файла; (б) печать текстового файла и др. Логически эти методы выполняют одну и ту же операцию, хотя реализуются они разными фрагментами кода.

Каждая операция имеет один неявный аргумент - объект к которому она применяется. Кроме того, операция может иметь и другие аргументы (параметры). Эти дополнительные аргументы параметризуют операцию, но не связаны с выбором метода. Метод связан только с классом и объектом.

Операция (и реализующие ее методы) определяется своей сигнатурой, которая включает, помимо имени операции, типы (классы) всех ее аргументов и тип (класс) результата (возвращаемого значения). Все методы, реализующие операцию должны иметь такую же сигнатуру, что и реализуемая ими операция.

Значения некоторых атрибутов объекта могут быть доступны только операциям этого объекта. Такие атрибуты называются *закрытыми.*

Запросы без аргументов (за исключением неявного аргумента - объекта, к которому применяется операция) могут рассматриваться как *производные атрибуты*. Значения производных атрибутов зависят от значений основных атрибутов. В этом их отличие от основных атрибутов, значения которых независимы. Следовательно, значения основных атрибутов объекта определяют как его состояние, так и значения его производных атрибутов. Так, например, длина, ширина и высота комнаты - ее основные атрибуты, а площадь и кубатура - производные атрибуты; такой атрибут как кубатура нужен для того, чтобы не вычислять кубатуру комнаты всякий раз, когда понадобится ее значение.

Выбор основных атрибутов объектов произволен, но в число основных атрибутов не следует включать такие атрибуты, значения которых определяются значениями других атрибутов, так что на самом деле они являются производными.

12. Связь объектов с базой данных.

В объектно-ориентированном проектировании мы имеем дело с множествами взаимосвязанных объектов. Каждый объект может рассматриваться как переменная или константа структурного типа (при таком рассмотрении методы, описываемые в объекте, трактуются как адреса функций, которые разрешено применять к этому объекту). Следовательно, множество объектов - это множество взаимосвязанных данных, т.е. нечто очень похожее на базу данных. Поэтому применение понятий баз данных часто оказывается полезным при объектно-ориентированном анализе и объектно-ориентированном проектировании прикладных программных систем.