**17. Объектная модель системы. Имена ролей, квалификаторы.**

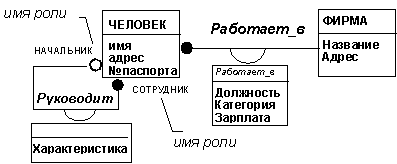
#### Объектная модель системы

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 прагматика разрабатываемой системы, что выражается в использовании терминологии прикладной области, связанной с использованием разрабатываемой системы.

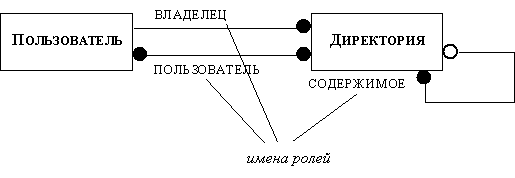
Имена ролей, квалификаторы

Роль определяет одну сторону зависимости. В бинарной зависимости определены две роли. Имя роли однозначно определяет одну сторону зависимости. Роли дают возможность рассматривать бинарную зависимость как связь между объектом и множеством зависимых объектов: каждая роль является обозначением объекта или множества объектов, связанных зависимостью с объектом на другом конце зависимости. Имя роли можно рассматривать как производный атрибут, множеством значений которого является множество связанных с этой ролью объектов. В бинарной зависимости пара имен ролей может использоваться для идентификации этой зависимости.

На рисунке [2.11](http://citforum.ck.ua/programming/oop_rsis/glava2_1_5.shtml#r2_11) имена начальник и сотрудник в зависимости руководит - это имена ролей; как уже было отмечено, эту зависимость удобнее назвать начальник-сотрудник.



Еще один пример имен ролей показан на рисунке 2.13.

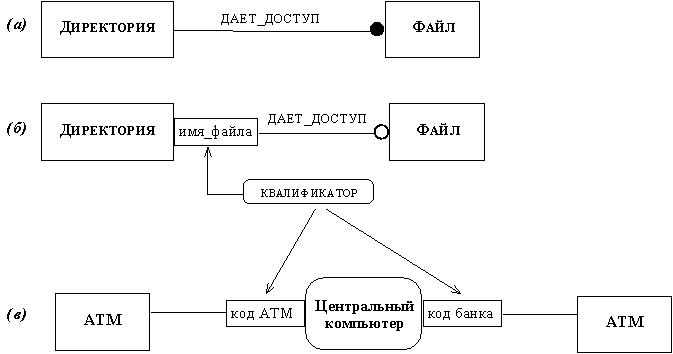


*Рис. 2.13. Имена ролей*

Пользователь может быть либо владельцем, либо зарегистрированным пользователем директории; директория может содержать в себе другие директории.

Имена ролей следует обязательно указывать в тех случаях, когда зависимость устанавливается между объектами одного и того же класса (как в случаях, показанных на рисунках [2.11](http://citforum.ck.ua/programming/oop_rsis/glava2_1_5.shtml#r2_11) и 2.13). Имена ролей должны быть уникальны, так как они используются для различения объектов, участвующих в зависимости.

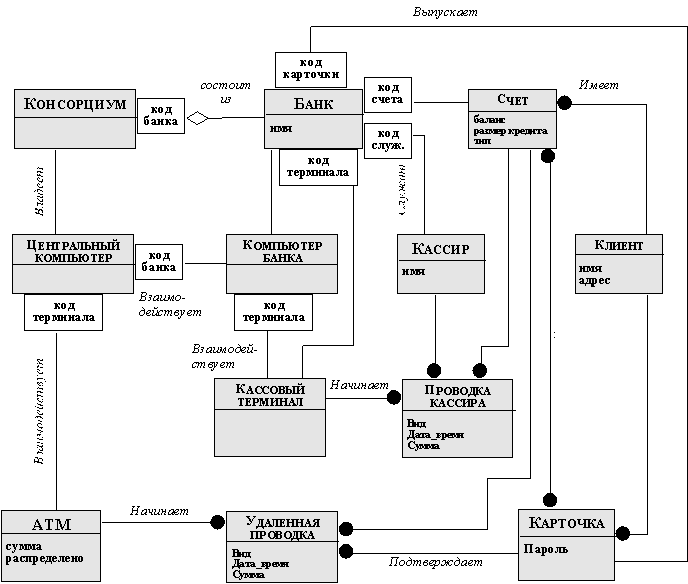
*Квалификатором* называется некоторый атрибут, который позволяет снизить эффективную кратность зависимости. Квалификаторы применяются в зависимостях типов "один ко многим" или "много ко многим". Так в примере, показанном на рисунке 2.14, использование квалификатора имя файла позволяет привести зависимость дает доступ от вида, приведенного на рисунке 2.14(а), к виду, показанному на рисунке 2.14(б), сократив число зависимых объектов до одного. Еще один пример использования квалификатора показан на рисунке 2.14(в): использование квалификаторов и здесь позволяет сократить кратность зависимости до одного объекта. Квалификаторы указываются на схемах в прямоугольничках, пририсованных к прямоугольнику, изображающему соответствующий класс.

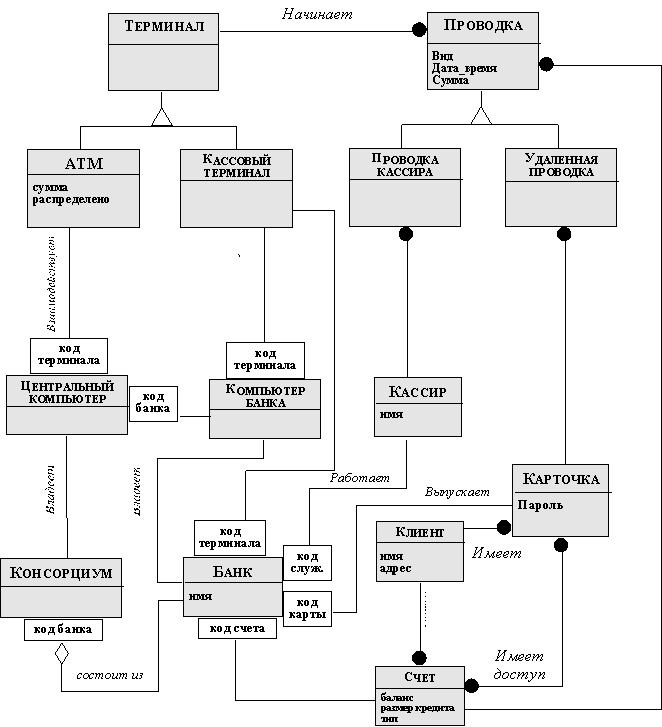


*Рис. 2.14. Использование квалификаторов*

Рисунок 2.14(в) может быть проинтерпретирован следующим образом: центральныйкомпьютер + код ATM определяют конкретную ATM (отметим, что код ATM - имя одного из атрибутов класса ATM, а не класса центральный компьютер); аналогично центральный компьютер + код банка определяют конкретный компьютер банка. Использование квалификаторов повышает точность описания семантики и наглядность описания зависимостей.

#### 18.Построение объектной модели. Организация системы классов, используя наследование.

 Организация системы классов с использованием наследования  
  
В рассматриваемом примере естественно определить суперклассы для объектов, определяющих различные терминалы: кассовый терминал и БМ (банкомат), и для объектов, определяющих проводки: проводка кассира и удаленная проводка (с банкомата).   
  
Внеся соответствующие изменения, получим объектную диаграмму, представленную на рисунке 1.4.   
  
  
  
Рисунок 1.3 - Объектная диаграмма для банковской сети после уточнения атрибутов и добавления квалификаторов

  
  
Рисунок 1.4 - Объектная диаграмма для банковской с учетом наследования

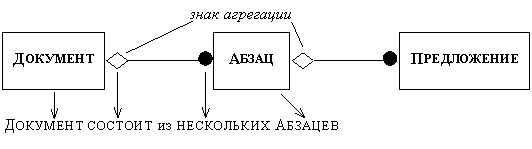
19.Объектная модель системы Агрегация.

#### 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.

#### *Агрегация*

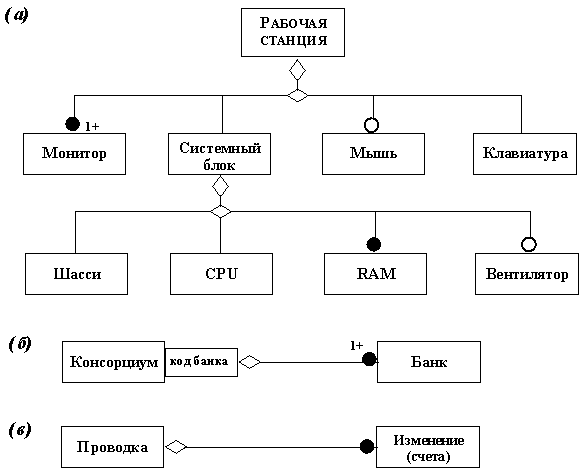
*Агрегация* - это зависимость между классом составных объектов и классами, представляющими компоненты этих объектов (отношение "целое"-"часть"). Агрегация обозначается ромбиком: на рисунке 2.15 приведен пример агрегации; этот пример интерпретируется следующим образом: документ состоит из нескольких (нуля, или более) абзацев; каждый абзац состоит из нескольких (нуля, или более) предложений.



*Рис. 2.15. Агрегация*

Наиболее важным свойством отношения агрегации является его *транзитивность* (если A есть часть B, а B есть часть C, то A есть часть C): так, из рисунка 2.15 можно заключить, что документ состоит из нескольких (нуля, или более) предложений. Легко видеть, что отношение агрегации *антисимметрично* (если A есть часть B, то B не есть часть A). Отметим также, что часть свойств целого может быть перенесена и на его части, возможно, с несущественными изменениями (например, контекст каждого оператора некоторой функции совпадает с внутренним контекстом всей функции).

Дальнейшие примеры агрегации показаны на рисунке 2.16. Отметим, что обе агрегации, показанные на рисунке 2.16(а), следует рассматривать не как зависимости между пятерками классов, а как четверки зависимостей между парами классов. Только при таком рассмотрении можно говорить о транзитивности и антисимметричности отношения агрегации.



*Рис. 2.16. Примеры агрегации*

20. Пример объектной модели. Дальнейшее исследование и усовершенствование модели.

***Дальнейшее исследование и усовершенствование модели***

Лишь в очень редких случаях построенная объектная модель сразу же оказывается корректной. Модель должна быть исследована и отлажена. Некоторые ошибки могут быть найдены при исследовании модели без компьютера, другие - при ее интерпретации совместно с динамической и функциональной моделями на компьютере (эти модели строятся после того, как объектная модель уже построена).

Здесь мы рассмотрим приемы бескомпьютерного поиска и исправления ошибок в объектной модели. В их основе лежат внешние признаки, по которым можно находить ошибки в модели; эти признаки могут быть объединены в следующие группы.

***Признаки пропущенного объекта (класса):***

* несимметричности связей и обобщений (наследований); для исправления ошибки необходимо добавить пропущенные классы;
* несоответствие атрибутов и операций у класса; для исправления ошибки необходимо расщепить класс на несколько других классов, так чтобы атрибуты и операции новых классов соответствовали друг другу;
* обнаружена операция, не имеющая удовлетворительного целевого класса; для исправления ошибки необходимо добавить пропущенный целевой класс;
* обнаружено несколько зависимостей с одинаковыми именами и назначением; для исправления ошибки необходимо сделать обобщение и добавить пропущенный суперкласс.

***Признаки ненужного (лишнего) класса:***

* нехватка атрибутов, операций и зависимостей у некоторого класса; для исправления ошибки необходимо подумать, не следует ли исключить такой класс.

***Признаки пропущенных зависимостей:***

* отсутствуют пути доступа к операциям; для исправления ошибки необходимо добавить новые зависимости, обеспечивающие возможности обслуживания соответствующих запросов.

***Признаки ненужных (лишних) зависимостей:***

* избыточная информация в зависимостях; для исправления ошибки необходимо исключить зависимости, не добавляющие новой информации, или пометить их как производные зависимости;
* не хватает операций, пересекающих зависимость; для исправления ошибки необходимо подумать, не следует ли исключить такую зависимость.

***Признаки неправильного размещения зависимостей:***

* имена ролей слишком широки или слишком узки для их классов; для исправления ошибки необходимо переместить зависимость вверх или вниз по иерархии классов.

***Признаки неправильного размещения атрибутов:***

* нет необходимости доступа к объекту по значениям одного из его атрибутов; для исправления ошибки необходимо рассмотреть нужно ли ввести квалифицированную зависимость.