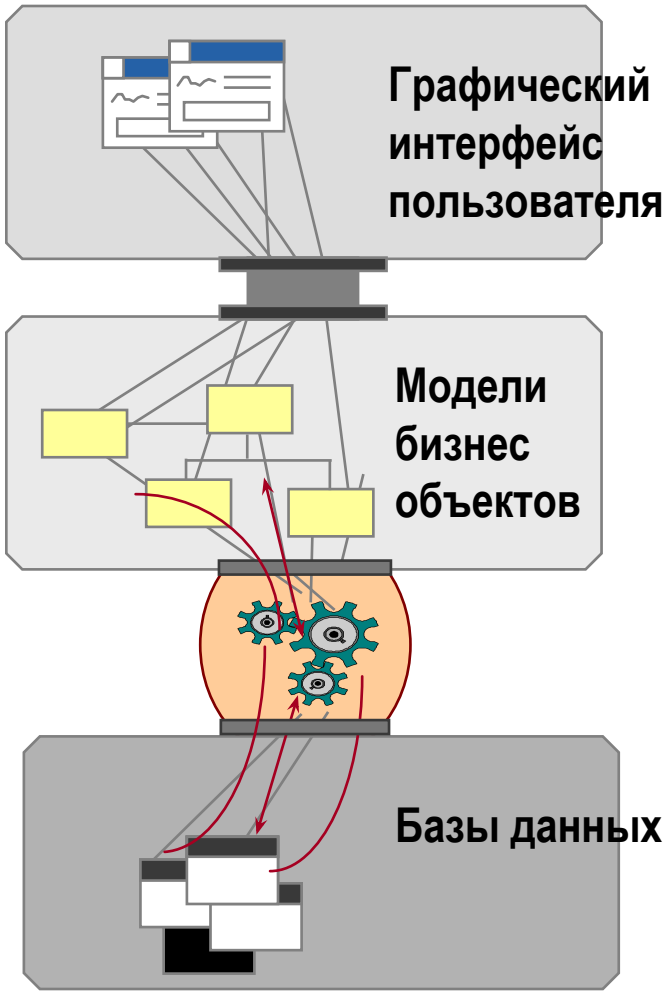
Учебный курс

Язык UML в анализе и проектировании программных систем и бизнес-процессов

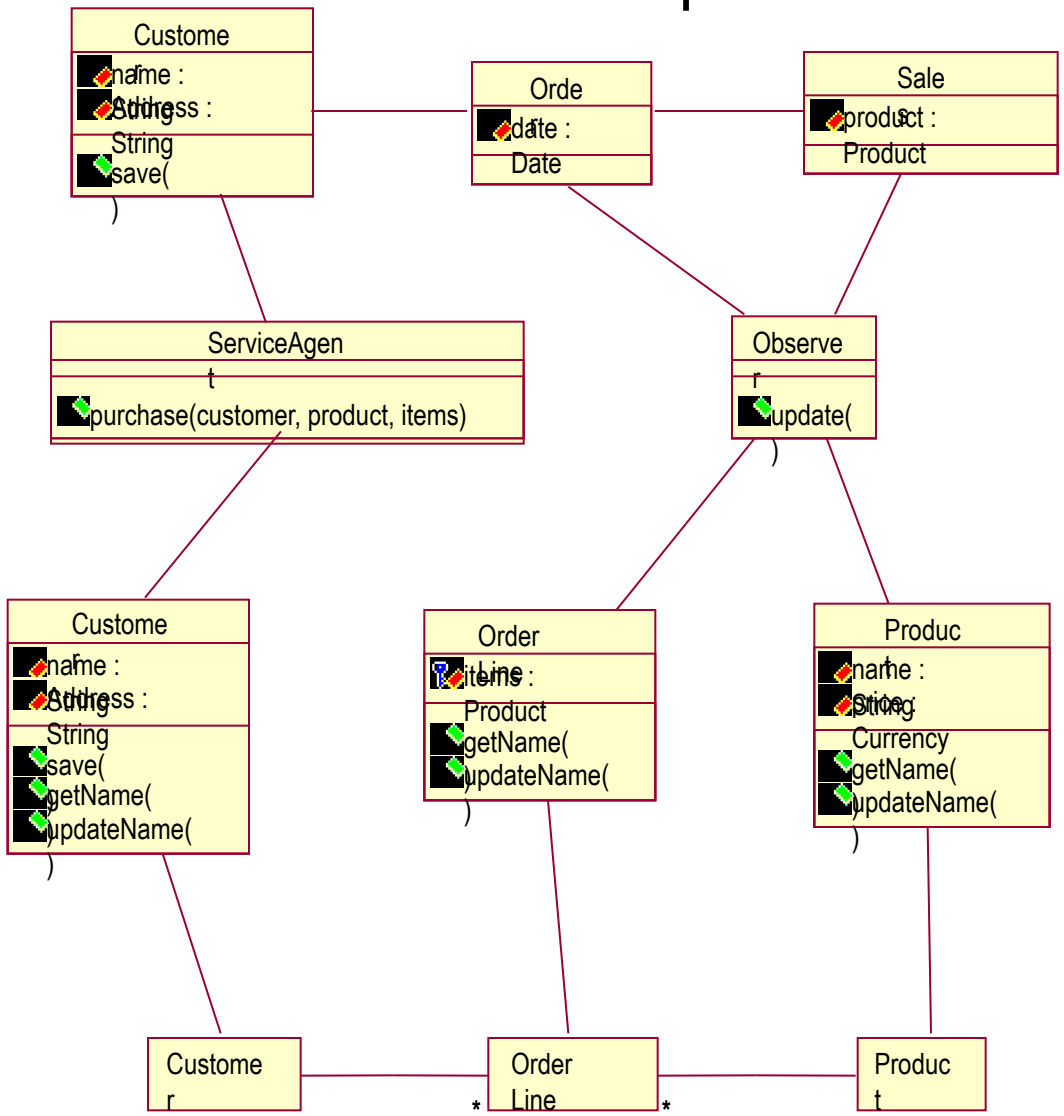
Лекция 3

Диаграмма классов языка UML 2

Проблема: Как представить архитектуру программной системы?



Архитектура программной системы в нотации UML



Уровень GUI

Промежуточный уровень

Уровень базы данных

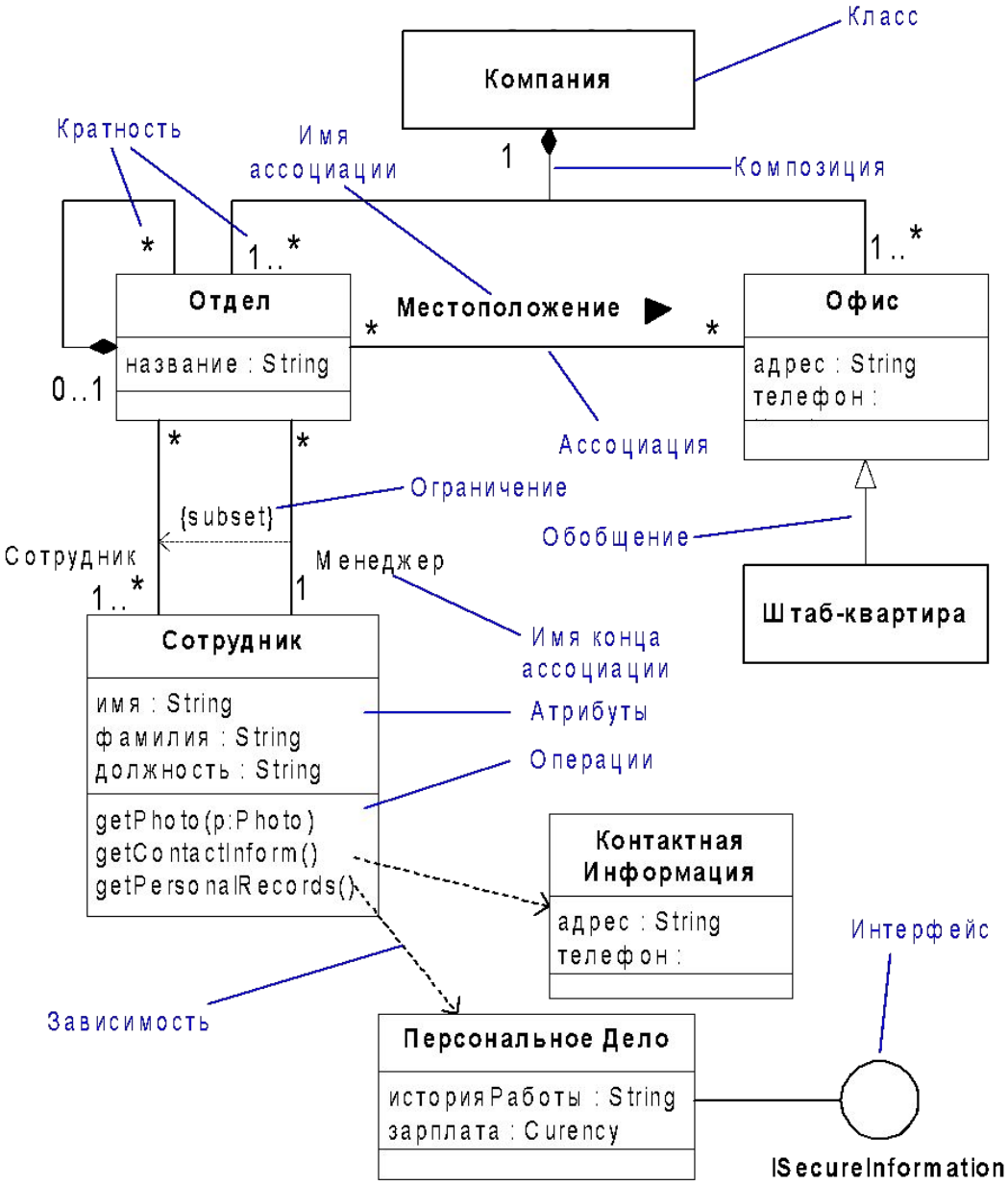
Диаграмма классов — основная логическая модель проектируемой системы

- *Диаграмма классов (class diagram)* — диаграмма, предназначенная для представления модели статической структуры программной системы в терминологии классов объектно-ориентированного программирования
- Диаграмма классов представляет собой граф, вершинами или узлами которого являются элементы типа “классификатор”, которые связаны различными типами структурных отношений
- *Классификатор (classifier)* – специальное понятие, предназначенное для классификации экземпляров, которые имеют общие характеристики

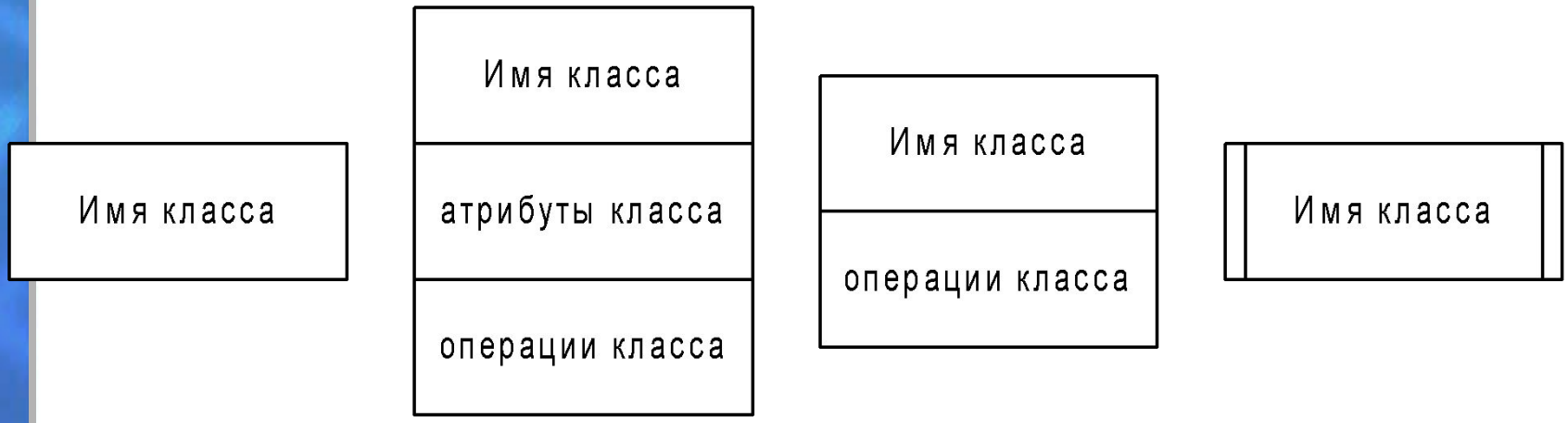
Характеристики классификатора

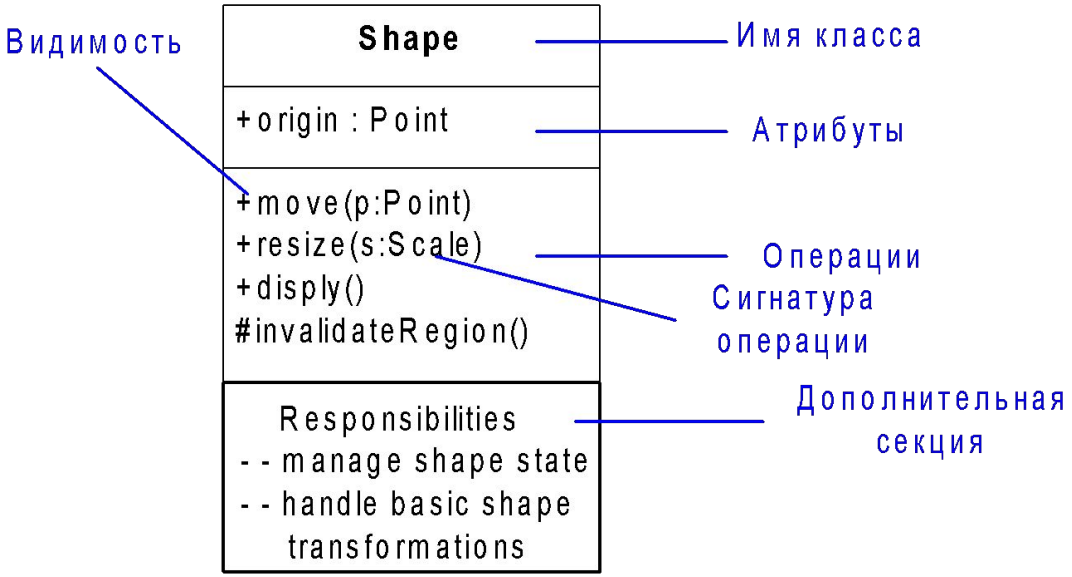
- *Характеристика (feature)* – понятие, предназначенное для спецификации особенностей структуры и поведения экземпляров классификаторов
- *Структурная характеристика (structural feature)* является типизированной характеристикой классификатора, которая специфицирует структуру его экземпляров
- *Характеристика поведения (behavioral feature)* является характеристикой классификатора, которая специфицирует некоторый аспект поведения его экземпляров
- *Класс (class)* – элемент модели, который описывает множество объектов, имеющих одинаковые спецификации характеристик, ограничений и семантики

Основные обозначения на диаграмме



Варианты графического изображения класса на диаграмме классов





Разновидности классов

Имя класса

- *Абстрактный* (abstract) класс не имеет экземпляров или объектов, для обозначения его имени используется наклонный шрифт (*курсив*)
- *Активный класс (active class)* – класс, каждый экземпляр которого имеет свою собственную нить управления
- *Пассивный класс (passive class)* – класс, каждый экземпляр которого выполняется в контексте некоторого другого объекта
- *Квалифицированное имя* (qualified name) используется для того, чтобы явно указать, к какому пакету относится тот или иной класс. Для этого применяется специальный символ в качестве разделителя имени – двойное двоеточие “::”
- Имя класса без символа разделителя называется *простым именем* класса

Атрибут (attribute) класса

- – служит для представления отдельной структурной характеристики или свойства, которое является общим для всех объектов данного класса
- <*атрибут*>::= [<*видимость*>] [‘/’] <*имя*> [‘:’ <*тип атрибута*>] [‘[‘<*кратность*>‘]’] [‘=’ <*значение по умолчанию*>] [‘{‘<*модификатор атрибута*> [‘,’ <*модификатор атрибута*>]* ’}’]
  - Где:
- <*видимость*>::= ‘+’ | ‘–‘ | ‘#’ | ‘~’.
- *видимость* (visibility) может принимать одно из 4-х возможных значений и отображаться либо посредством специального символа, либо соответствующего ключевого слова

Вид видимости

- + public (общедоступный). Общедоступный элемент является видимым всеми элементами, который имеют доступ к содержимому пространства имен, который им владеет.
- - private (закрытый). Закрытый элемент является видимым только внутри пространства имен, который им владеет.
- # protected (защищенный). Защищенный элемент является видимым для элементов, которые имеют отношение обобщения с пространством имен, который им владеет.
- ~ package (пакет). Элемент, помеченный как имеющий пакетную видимость, является видимым всеми элементами в ближайшем охватывающем пакете в предположении. За пределами ближайшего охватывающего пакета элемент, помеченный как имеющий пакетную видимость, не является видимым.

Проблема интерпретации видимости в языка UML

- «Ничто в языке UML не определяется так просто, и не интерпретируется так сложно, как видимость» (Мартин Фаулер)
- В C++ видимость «friend» (дружественная) обладает полным доступом ко всем элементам класса
- «…в C++ друзья прикасаются к закрытым частям друг друга»
- В Java видимость «package» (пакетная) обладает полным доступом ко всем классам данного пакета
- В Java разрешается помечать классы как:
  - общедоступные – элементы общедоступного класса могут использоваться любым классом, который импортирует пакет, содержащий исходный класс
  - пакетные – элементы пакетного класса могут использоваться только классами данного пакета

Элементы записи атрибута

- “/” означает, что атрибут является *производным* (derive). Значение производного атрибута может быть вычислено на основе значений других атрибутов этого или других классов. Поэтому данный атрибут называют иногда *вычислимым*. При использовании производных атрибутов разработчик должен явно указать процедуру или операцию для вычисления их значений.
- <*имя*> (name) 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 в обозначении атрибута, должно начинаться со срочной (малой) буквы и, как правило, не должно содержать пробелов.

Элементы записи атрибута

- <*тип атрибута*> (attribute type) есть имя классификатора, который является типом данного атрибута. Тип атрибута представляет собой имя некоторого типа данных, определенного или в пакете Типу атрибута должно предшествовать двоеточие
- <*кратность*> (multiplicity) атрибута характеризует общее количество конкретных значений для атрибута, которые могут быть заданы для объектов данного класса
- <*значение по умолчанию*> (default) – некоторое выражение, которое служит для задания начального значения или значений данного атрибута в момент создания отдельного экземпляра соответствующего класса. Конкретное значение по умолчанию должно соответствовать типу данного атрибута. Если этот терм не указан, то значение атрибута на момент создания нового экземпляра класса не определено.

Модификатор атрибута

- <*модификатор атрибута*> (attribute modifier) представляет собой текстовое выражение, которое придает дополнительную семантику данному атрибуту. При этом набор возможных модификаторов атрибутов в языке UML 2.x фиксирован и может быть представлен в следующем виде (БНФ):
- <*модификатор атрибута*>::= ‘readOnly’ | ‘union’ | ‘subsets‘ <*имя атрибута*> | ‘redefines’ <*имя атрибута*> | ‘ordered’ | ‘unique’ | <*ограничение атрибута*>.

Значения модификатора атрибута

- readOnly – атрибут является только для чтения
- union – атрибут является производным объединением его подмножеств
- subsets <*имя атрибута*> – атрибут является собственным подмножеством атрибута с именем <*имя атрибута*>
- redefines <*имя атрибута*> – атрибут переопределяет некоторый наследуемый атрибут с именем <*имя атрибута*>

Значения модификатора атрибута

- ordered – значения атрибута являются упорядоченными. Этот порядок означает, что существует отображение из множества положительных целых чисел в элементы этой коллекции значений. Если атрибут не является многозначным, то это значение не имеет семантического эффекта. При отсутствии этого модификатора атрибут специфицируется как неупорядоченный.
- unique – значения многозначного атрибута не могут иметь дубликатов, т.е. повторяться. Предполагается, что кратность соответствующего атрибута должна быть больше 1. Если атрибут не является многозначным, то значение unique не имеет семантического эффекта.
- <*ограничение атрибута*> Выражение, которое специфицирует некоторое ограничение, применяемое к данному атрибуту

Кратность

- *Кратность (multiplicity)* является спецификацией допустимой мощности множества при инстанцировании соответствующего элемента модели
- Спецификация кратности в нотации БНФ имеет следующий формат:
- *<кратность> ::= <диапазон-кратности> ['{' <указатель-упорядоченности> [',' <указатель-уникальности>] '}']*
- *<диапазон-кратности> ::= [<нижняя-граница>'..'] <верхняя-граница>*
- *<нижняя-граница> ::= <целое число> | <спецификация значения>*
- *<верхняя-граница> ::= '*' | <спецификация значения>*
- *<указатель-упорядоченности> ::= 'ordered' | 'unordered'*
- *<указатель-уникальности> ::= 'unique' | 'nonunique'*

Примеры записи атрибутов

- + имяСотрудника : String {readOnly}
- ~ датаРождения : Data {readOnly}
- # /возрастСотрудника : Integer
- + номерТелефона : Integer [1..*] {unique}
- – заработнаяПлата : Currency = 500.00

Операции класса

- *Операция (operation)* класса служит для представления отдельной характеристики поведения, которая является общей для всех объектов данного класса
- Общий формат записи отдельной операции класса следующий (БНФ):
- <*операция*>*::*=[<*видимость*>] <*имя операции*> ‘(‘ [<*список параметров*>] ‘)’ [‘:’ [<*тип возвращаемого результата*>] ‘{‘ <*свойство операции*> [‘,’ <*свойство операции*>]* ‘}’]
- Где:
- <*видимость*> ::= ‘+’ | ‘-‘ | ‘#’ | ‘~’
- <*имя операции*> (operation name) представляет собой строку текста, которая используется в качестве идентификатора соответствующей операции и поэтому должна быть уникальной для каждой операции данного класса

Формат записи операции класса

- <*список параметров*> (parameter list) представляет собой перечень разделенных запятыми формальных параметров операции и имеет следующий общий формат записи (БНФ):
- <*список параметров*>*::=* <*параметр*> [‘,’<*параметр*>]*.
- <*параметр*>*::=* [<*направление*>] <*имя параметра*> ‘:’ <*выражение типа*> [‘[‘<*кратность*>’]’] [‘=’ <*значение по умолчанию*>] [‘{‘ <*свойство параметра* > [‘,’ <*свойство параметра*>]* ‘}’]
- <*тип возвращаемого результата*> (return type) специфицирует тип значения, возвращаемого данной операцией

Параметры операции

- *Параметр (parameter)* является спецификацией аргумента, который используется при выполнении операции или при вызове характеристики поведения
- <*направление*>::= ‘**in**’ | ‘**out**’ | ‘**inout**’| ‘**return**‘. Если оно не указано, то по умолчанию принимается значение “in”.
- **in** – указывает на то, что значения этого параметра передаются в операцию вызывающим объектом.
- **inout** – указывает на то, что значения этого параметра передаются в операцию вызывающим объектом и затем обратно вызывающему объекту после окончания выполнения операции.
- **out** – указывает на то, что значения этого параметра передаются вызывающему объекту после окончания выполнения операции.
- **return** – указывает на то, что значения этого параметра передаются в качестве возвращаемых значений вызывающему объекту после окончания выполнения операции.

Параметры операции

- <*имя параметра*> (parameter name) представляет собой идентификатор формального параметра, при записи которого необходимо следовать правилам задания имен атрибутов
- <*выражение типа*> (type expression) является спецификацией типа данных для возможных значений соответствующего формального параметра. Этот терм аналогичен рассмотренному выше терму <*тип атрибута*>для атрибутов классов

Параметры операции

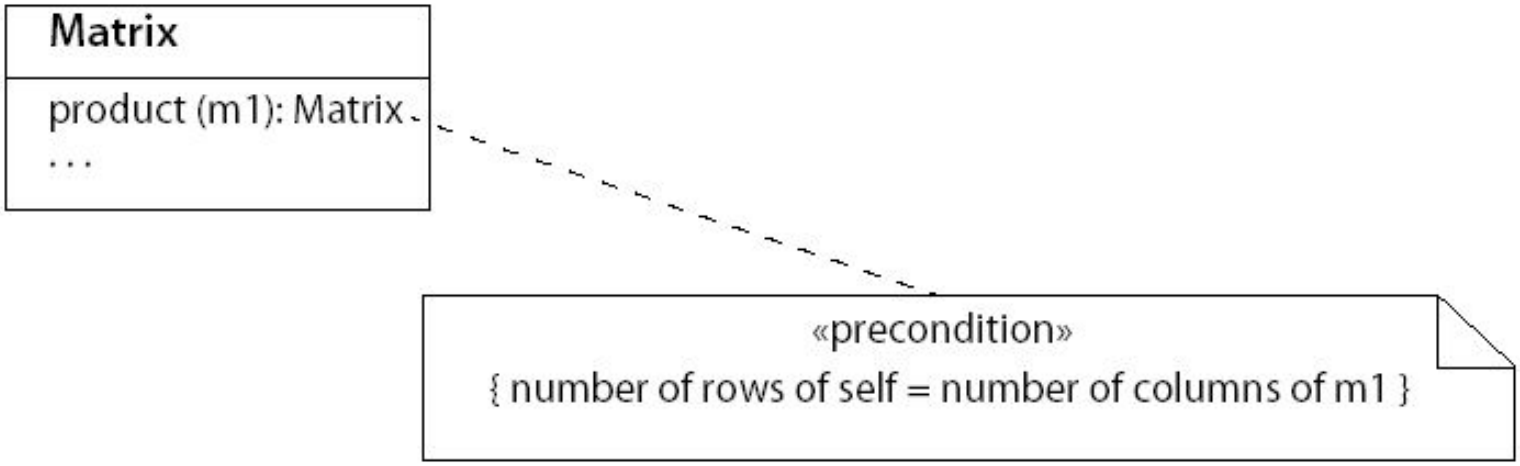
- <*кратность*> (multiplicity) характеризует общее количество конкретных параметров с данным именем, которые могут принадлежать тому или иному объекту данного класса
- <*значение по умолчанию*> (default) представляет собой некоторое выражение, которое специфицирует конкретное значение по умолчанию для данного формального параметра.
- <*свойство параметра*> (parameter property) указывает дополнительные свойства значений данного формального параметра. В качестве значений свойств параметра могут быть использованы модификаторы атрибутов

Свойства операций

- redefines <*имя операции*> – данная операция переопределяет некоторую наследуемую операцию с именем <*имя операции*>
- query – данная операция не изменяет состояния моделируемой системы и, соответственно, не имеет побочного эффекта
- ordered – значения возвращаемого параметра являются упорядоченными Предполагается, что кратность данного возвращаемого параметра должна быть больше 1
- unique – значения возвращаемого параметра не могут повторяться. Предполагается, что кратность данного возвращаемого параметра должна быть больше 1.
- <*ограничение*> – выражение, которое специфицирует некоторое ограничение, применяемое к данной операции

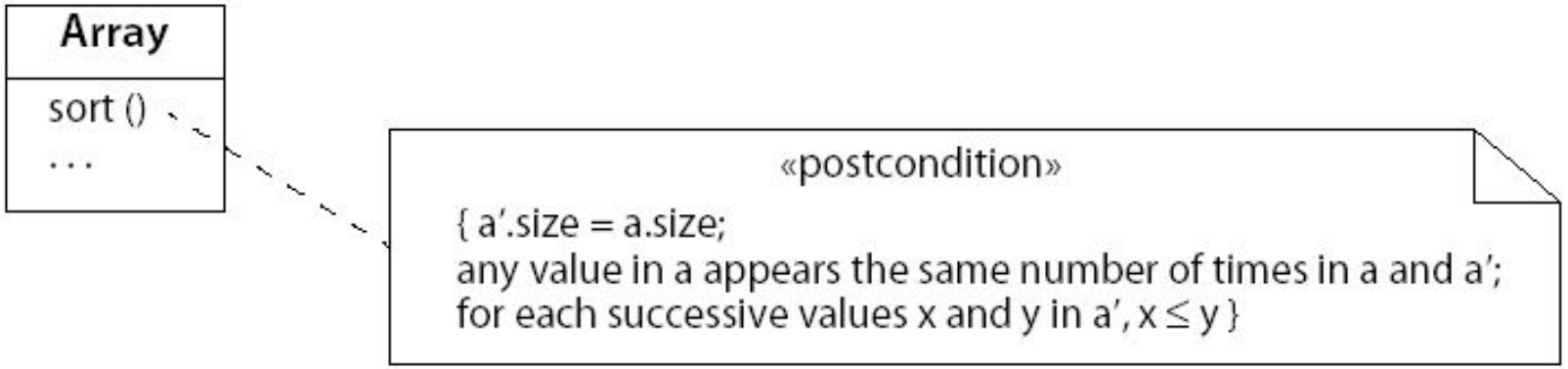
Предусловие (precondition) операции

- – определяет условие, которое должно быть истинным, когда эта операция вызывается



Постусловие (postcondition) операции

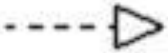
- – определяет условие, которое должно быть истинным, когда вызов операции успешно завершился, в предположении, что все предусловия были удовлетворены



Примеры записи операций:

- +добавить(**in** номерТелефона : Integer [*] {unique})
- –изменить(**in** заработнаяПлата : Currency)
- +создать() : Boolean
- toString(return : String)
- toString( ) : String

Отношения на диаграмме классов

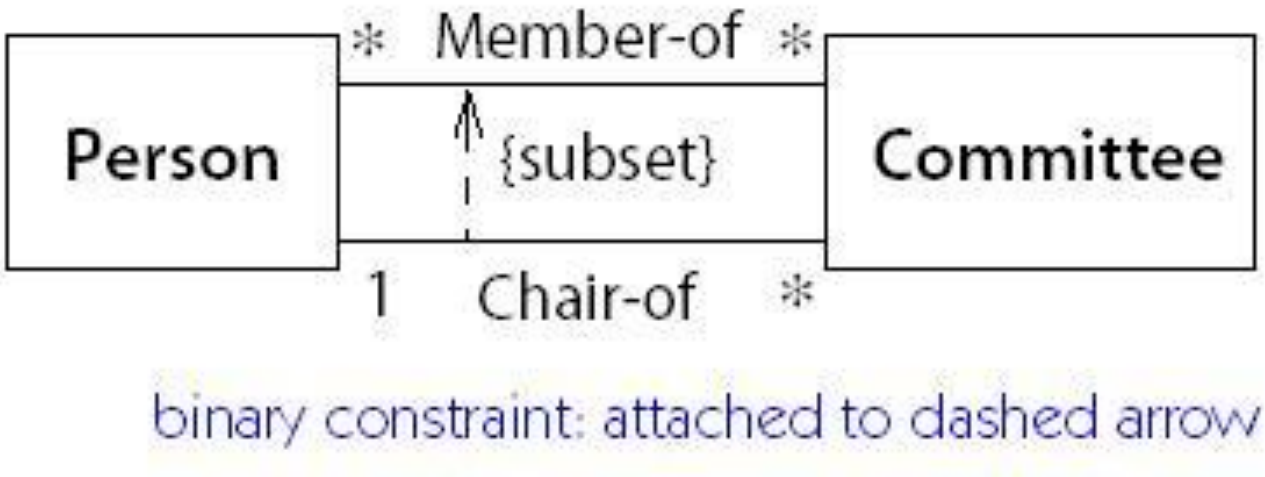
Relationship	*Function*	*Notation*
association	Представление произвольного отношения между экземплярами классов	
generalization	Отношение типа "Общее-Частное", обладающая свойством наследования свойств	
aggregation	Отношение типа "Часть-Целое"	
composition	Более сильная форма отношения типа "Часть-Целое"	
realization	Отношение между спецификацией и ее выполнением	
dependency	Направленное отношение между двумя элементами модели с открытой семаникой	

Ассоциация

- *Ассоциация (association)* – произвольное отношение или взаимосвязь между классами
- *Имя конца ассоциации* специфицирует *роль* (role), которую играет класс, расположенный на соответствующем конце рассматриваемой ассоциации
- *Видимость конца ассоциации* специфицирует возможность доступа к соответствующему концу ассоциации с других ее концов
- *Кратность конца ассоциации* специфицирует возможное количество экземпляров соответствующего класса, которое может соотноситься с одним экземпляром класса на другом конце этой ассоциации
- *Символ наличия навигации* (navigable) изображается с помощью простой стрелки в форме буквы «V» на конце ассоциации
- *Символ отсутствия навигации* (non navigable) изображается с помощью буквы «X» на линии у конца ассоциации

Строка свойство (property string)

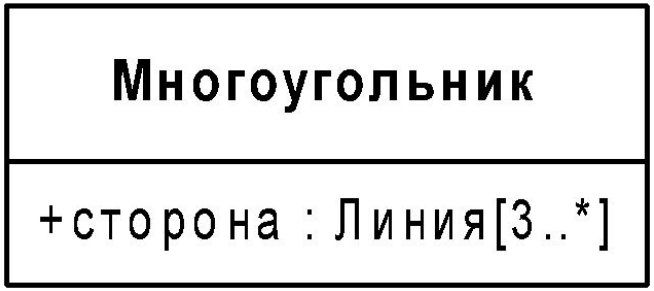
- {subset <*имя элемента*>} – конец ассоциации представляет собой некоторое подмножество <имя элемента>, в качестве которого может выступать имя конца ассоциации или атрибута класса



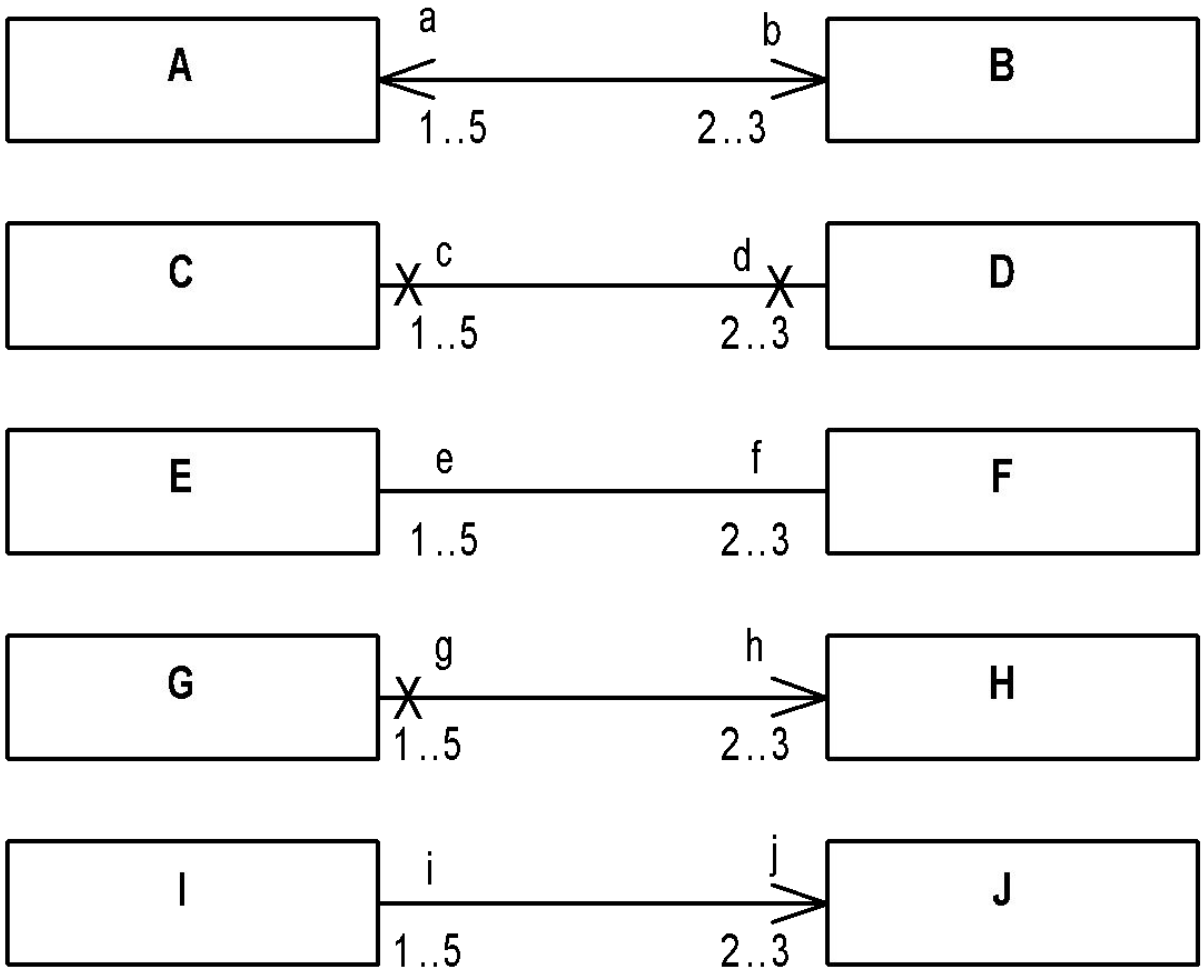
Строка свойство (property string)

- {redefined <*имя конца ассоциации*>} – конец ассоциации переопределяет другой конец ассоциации с именем <имя конца ассоциации>
- {union} – конец ассоциации является производным и определяется посредством объединения своих подмножеств
- {ordered} – конец ассоциации представляет собой некоторое упорядоченное множество
- {bag} – конец ассоциации представляет собой мультимножество или совокупность, в которой допускается представлять один и тот же элемент более одного раза
- {sequence} или {seq} – конец ассоциации представляет собой некоторую последовательность или упорядоченное мультимножество

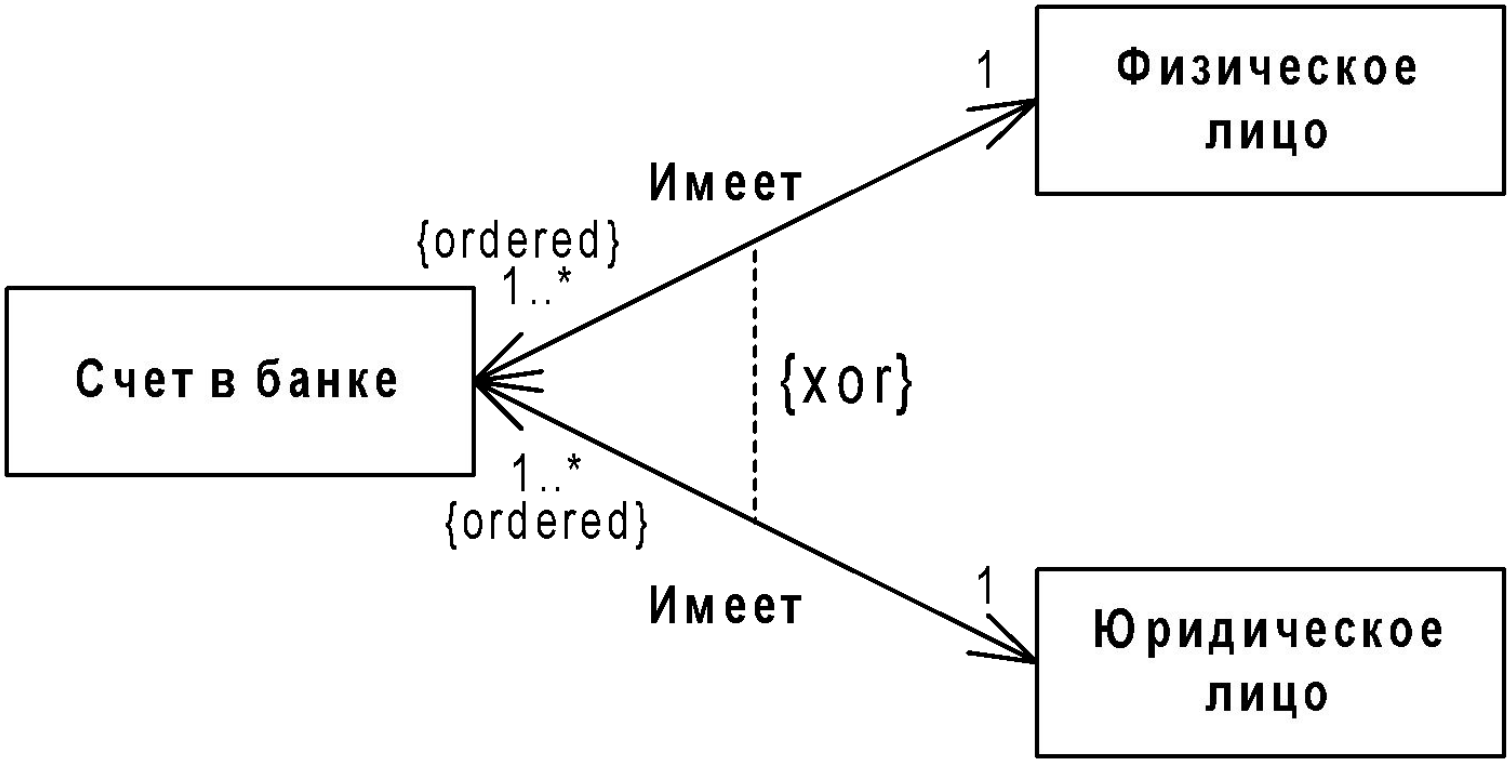
Ассоциация с навигацией и эквивалентное ему представление класса с атрибутом



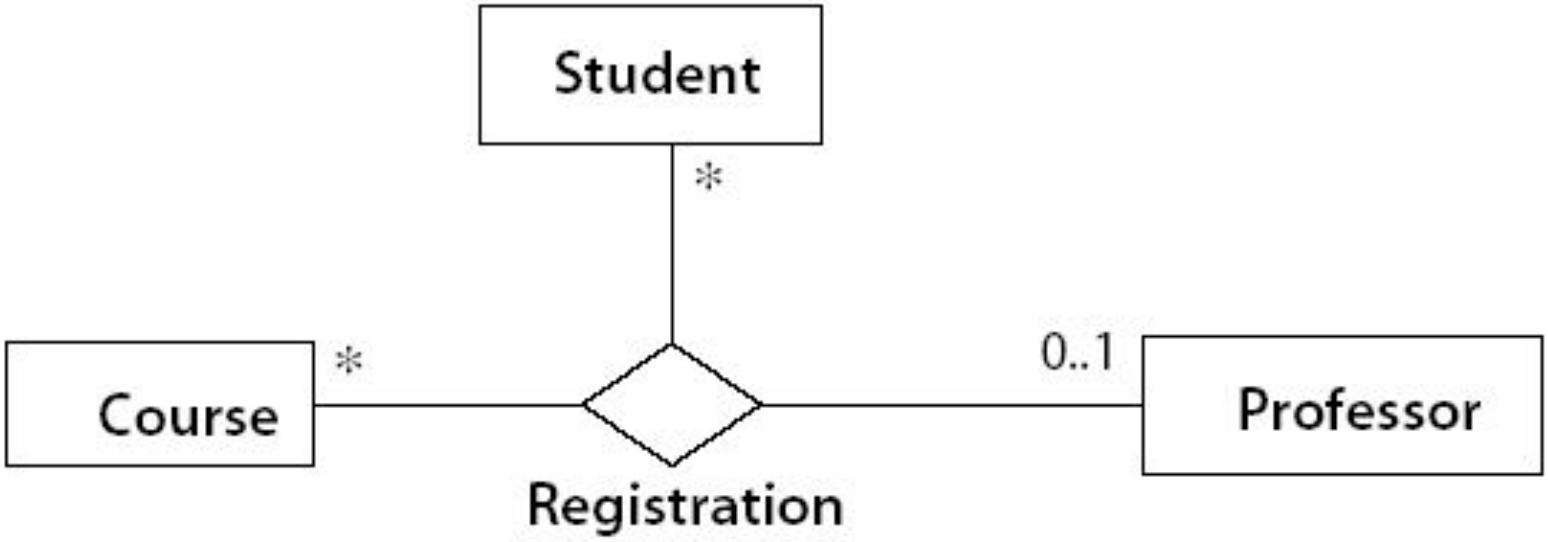
Варианты изображения навигации и кратности у концов ассоциации



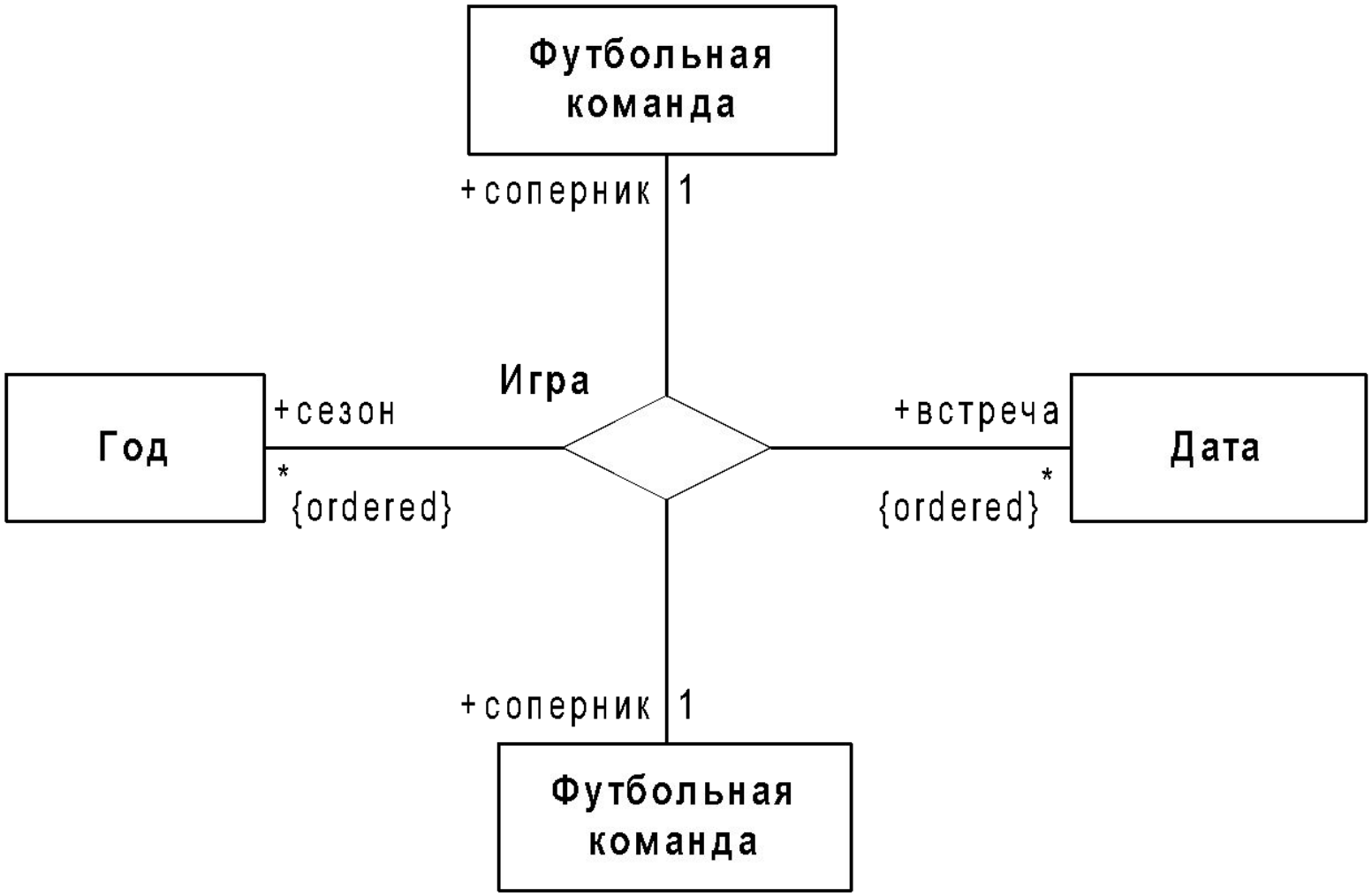
Исключающая ассоциация между тремя классами



Пример тернарной ассоциации



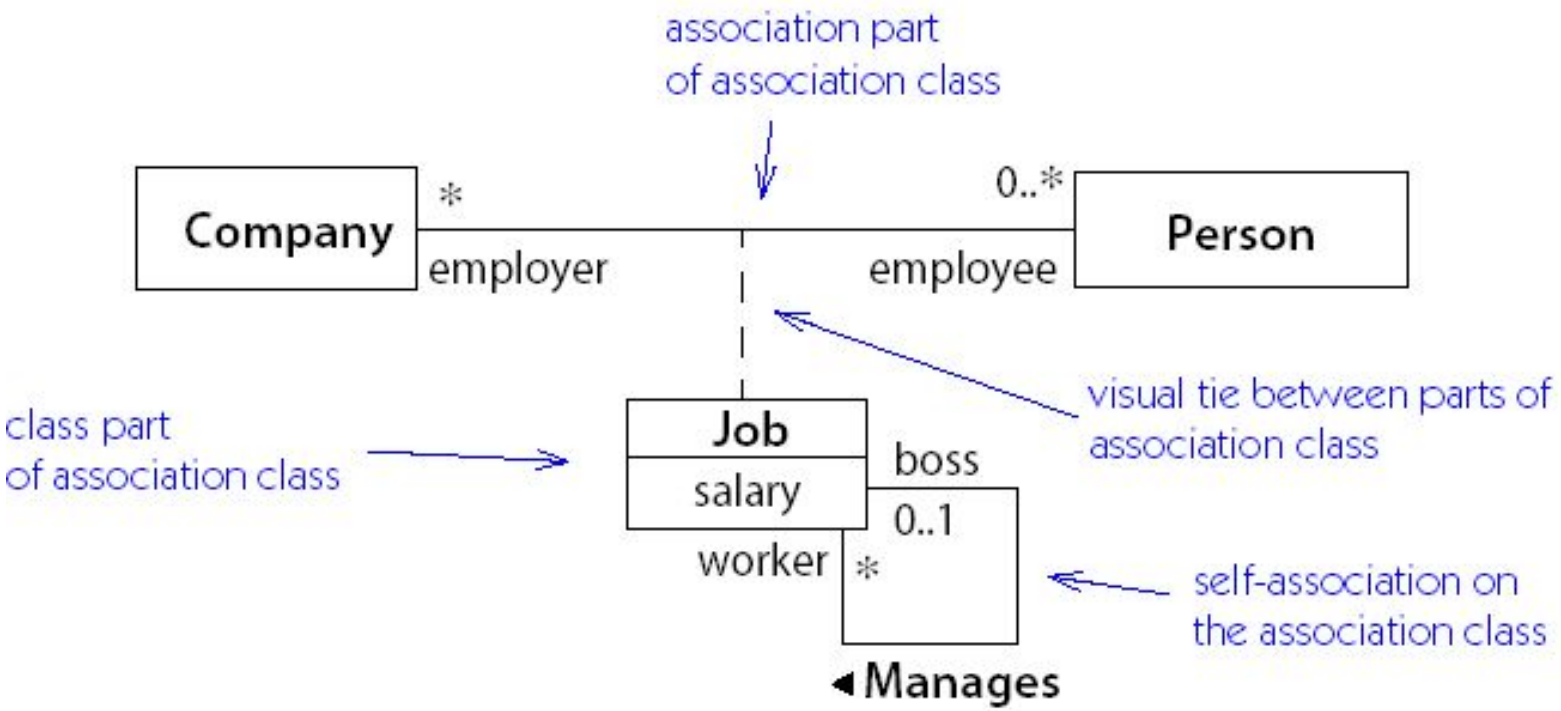
Пример 4-арной ассоциации



Ассоциация класс *(association class)*

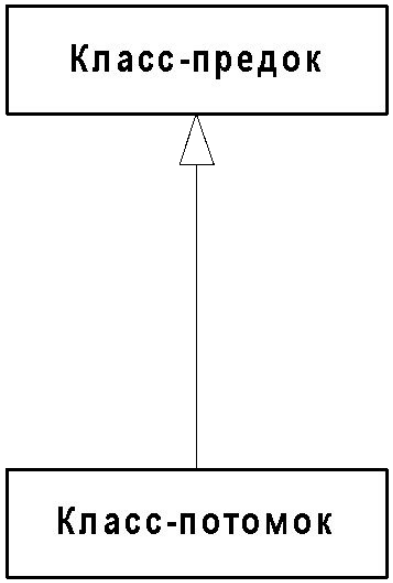
- – элемент модели, который имеет свойства как ассоциации, так и класса, и предназначенный для спецификации дополнительных свойств ассоциации в форме атрибутов и, возможно, операций класса.

Примеры ассоциации класса и рефлексивной ассоциации

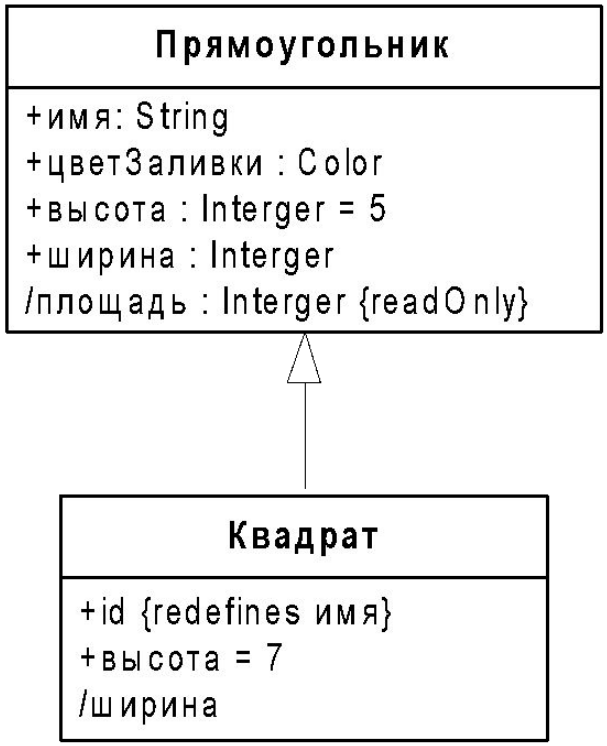


Обобщение *(generalization)*

- – таксономическое отношение между более общим классификатором (родителем или предком) и более специальным классификатором (дочерним или потомком)

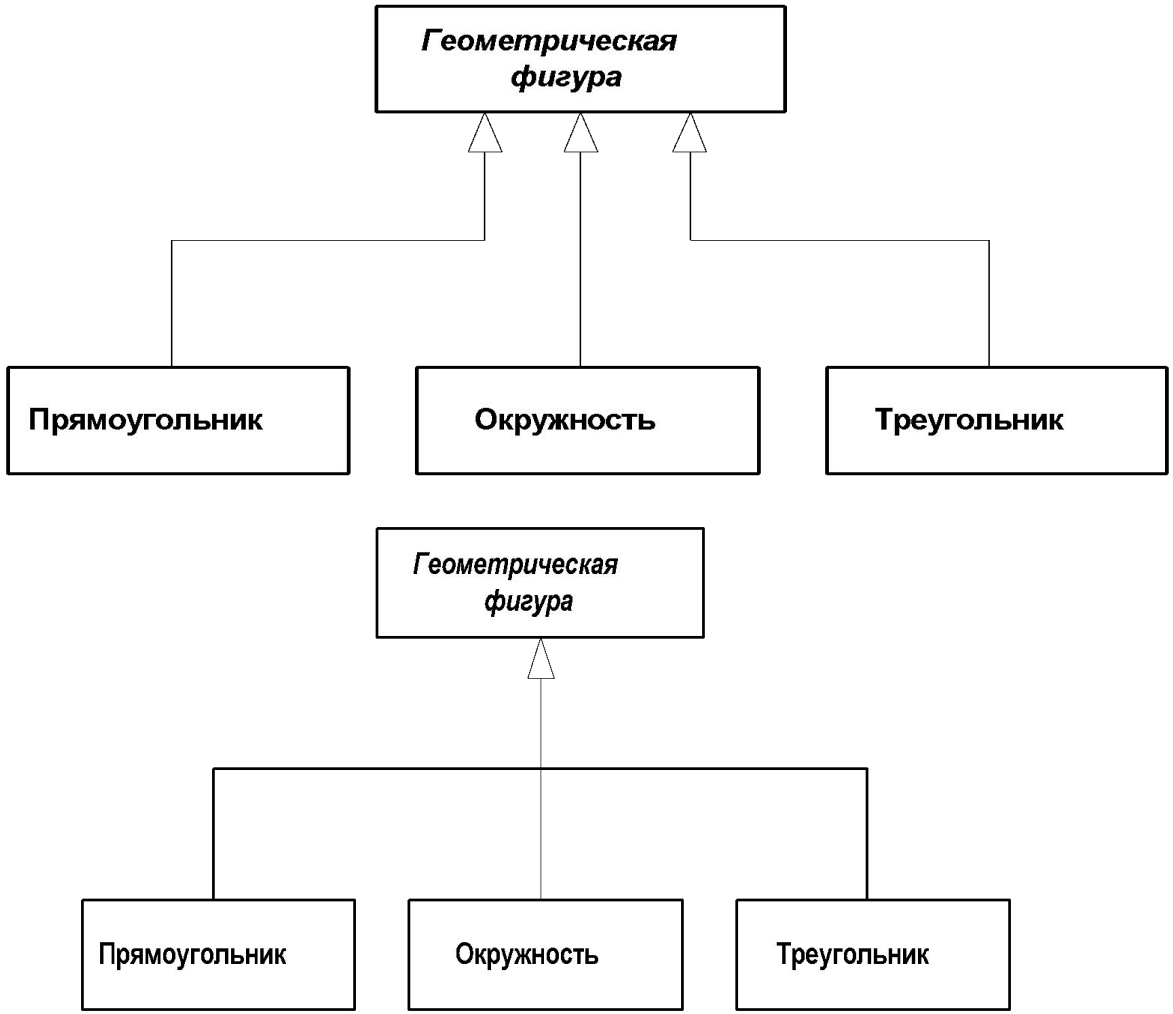


(а)

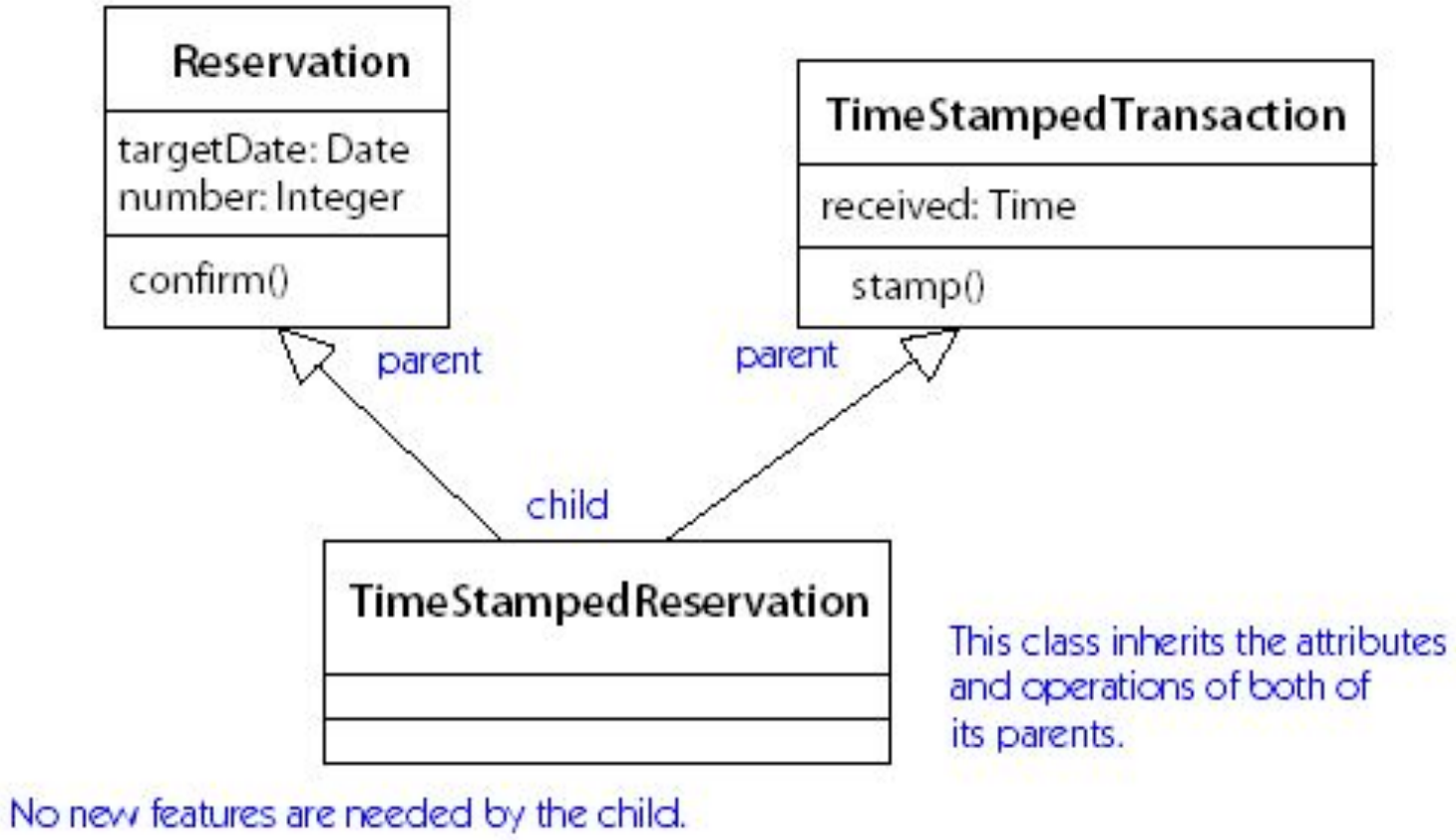


(б)

Примеры отношения обобщения



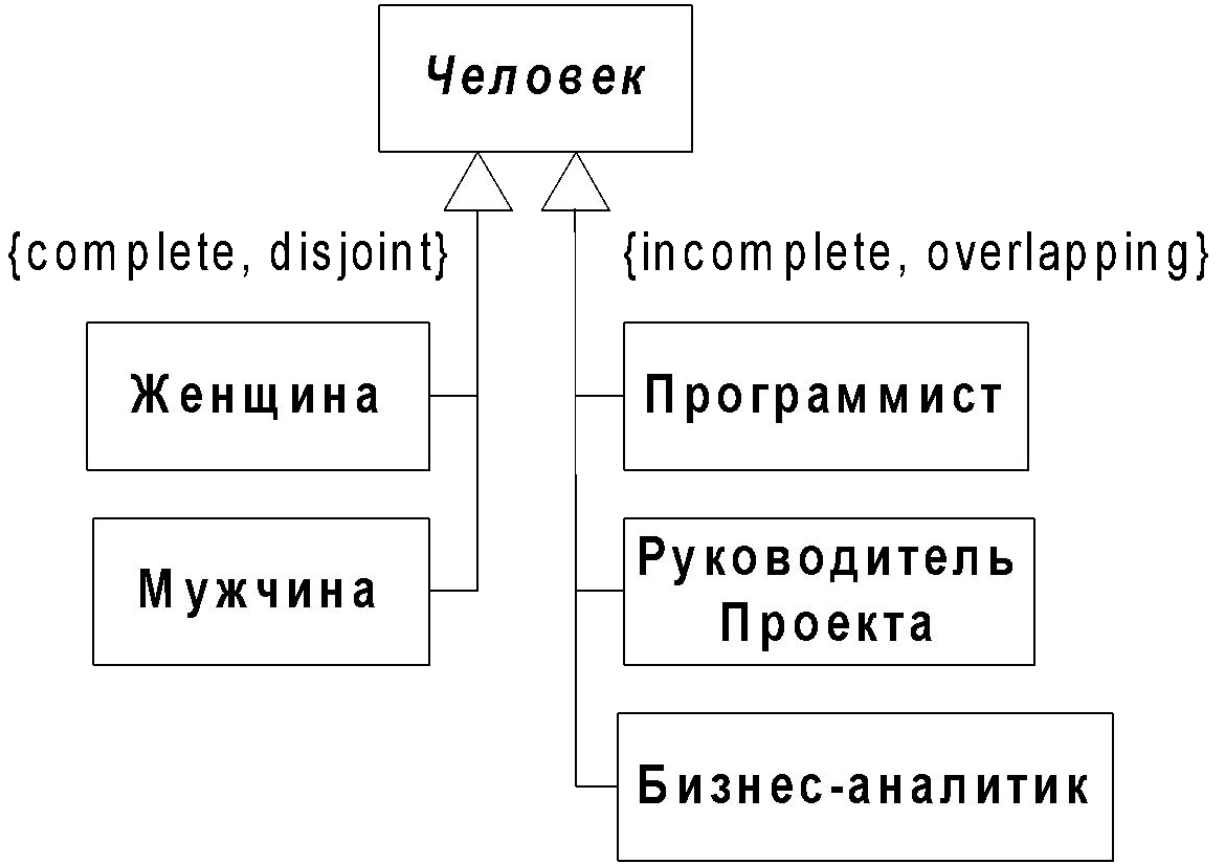
Множественное наследование – в языке UML разрешено



Множество обобщения *(generalization set)*

- – элемент модели, экземпляры которого определяют коллекции подмножеств отношения обобщения
- {complete, disjoint} — означает, что данное множество обобщения является покрывающим и его специальные классы не имеют общих экземпляров
- {incomplete, disjoint} — означает, что данное множество обобщения не является покрывающим и его специальные классы не имеют общих экземпляров (предполагается по умолчанию)
- {complete, overlapping} — означает, что данное множество обобщения является покрывающим и его специальные классы имеют общие экземпляры
- {incomplete, overlapping} — означает, что данное множество обобщения не является покрывающим и его специальные классы имеют общие экземпляры
- По умолчанию - {incomplete, disjoint}

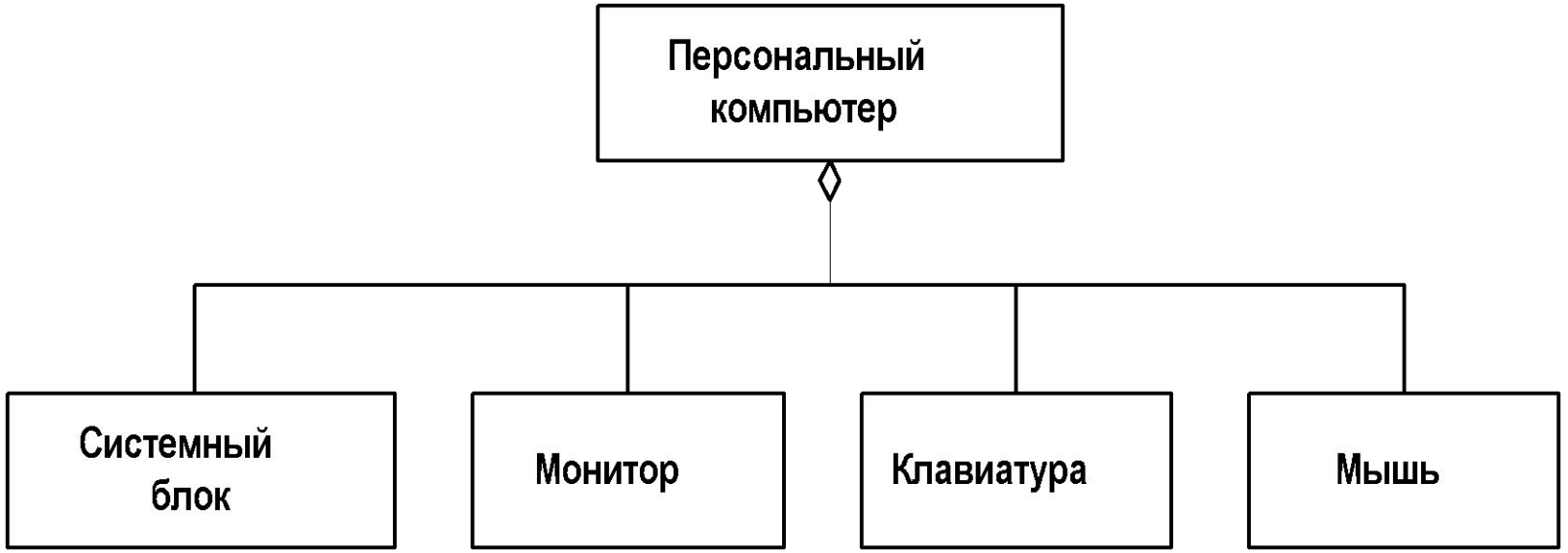
Примеры ограничений на множество обобщения



Агрегация *(aggregation)*

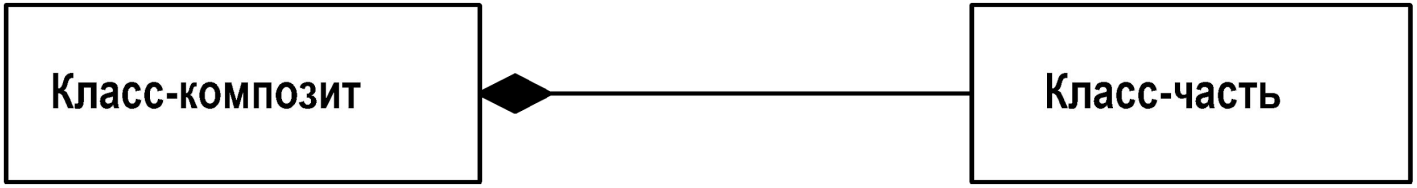
- – направленное отношение между двумя классами, предназначенное для представления ситуации, когда один из классов представляет собой некоторую сущность, которая включает в себя в качестве составных частей другие сущности

Пример отношения агрегации

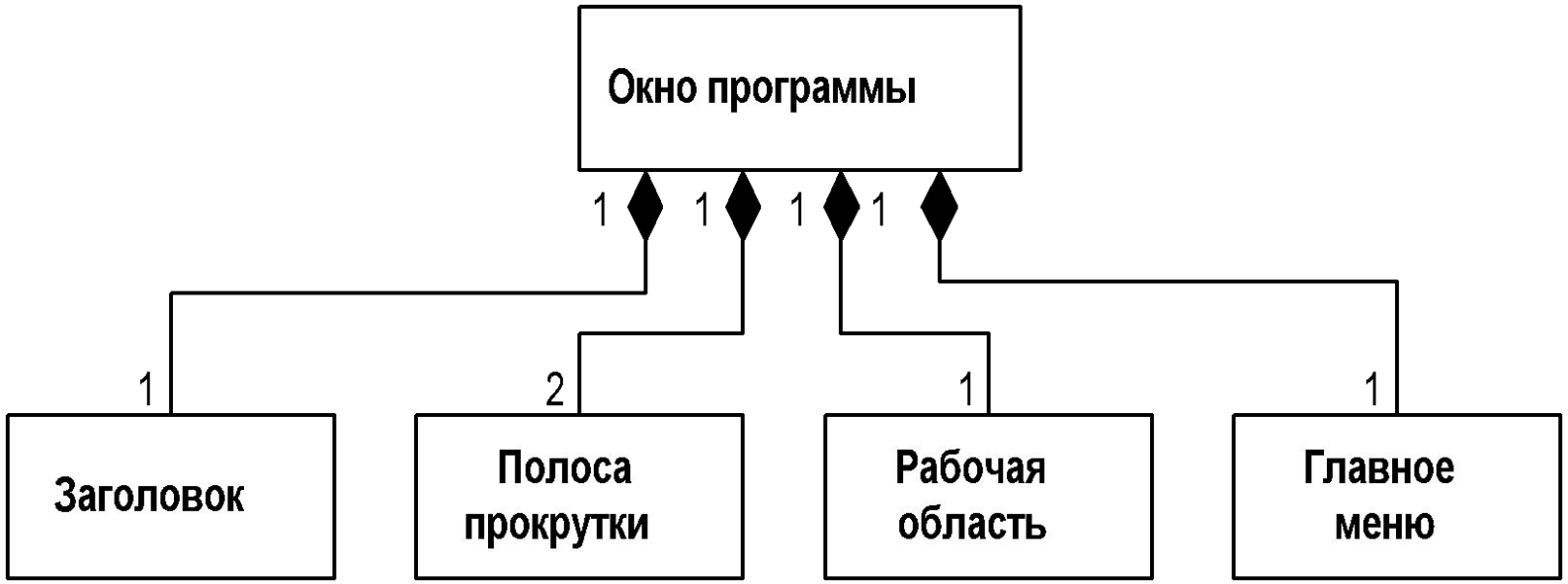


Композиция *(composition)*

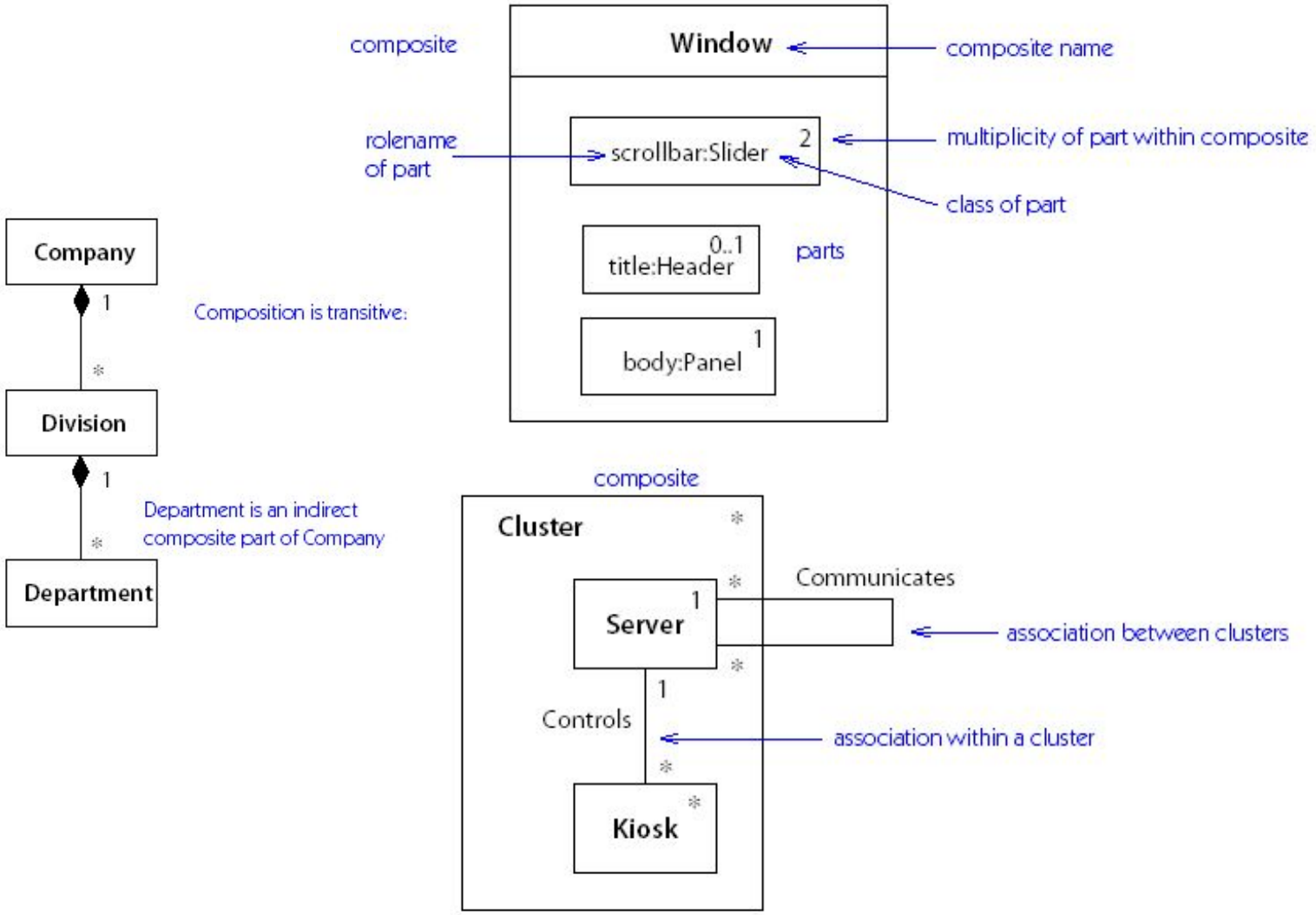
- или *композитная агрегация* предназначена для спецификации более сильной формы отношения "часть-целое", при которой с уничтожением объекта класса-контейнера уничтожаются и все объекты, являющимися его составными частями.



Пример отношения композиции

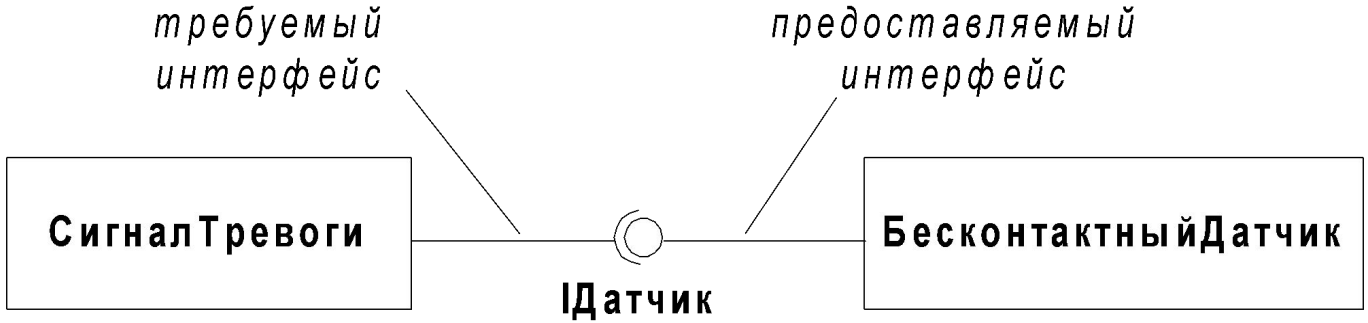


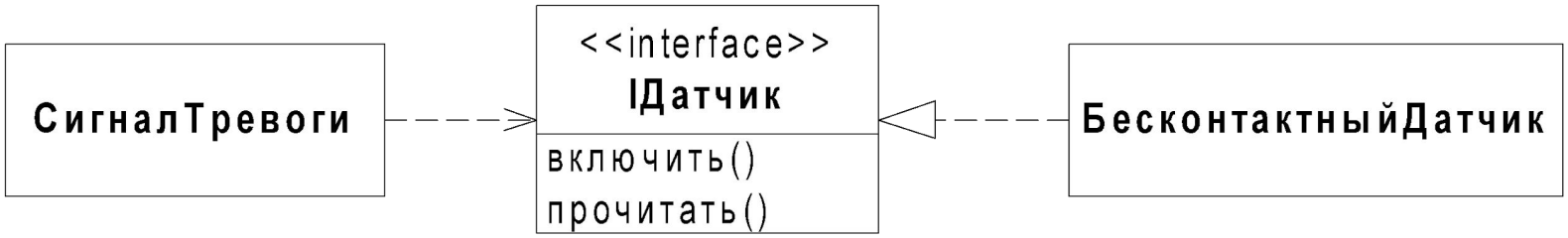
Варианты обозначения композиции



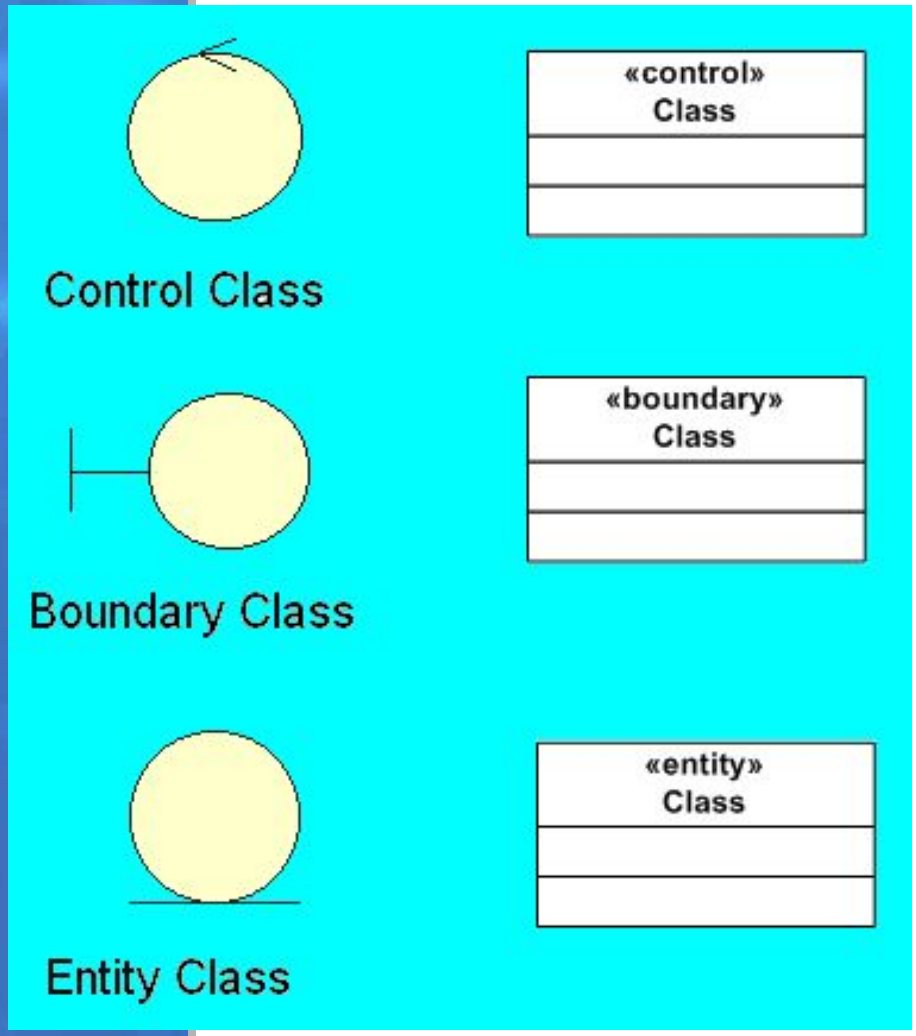
Интерфейс *(interface)*

- – вид класса, который представляет собой объявление множества общедоступных характеристик и обязанностей.





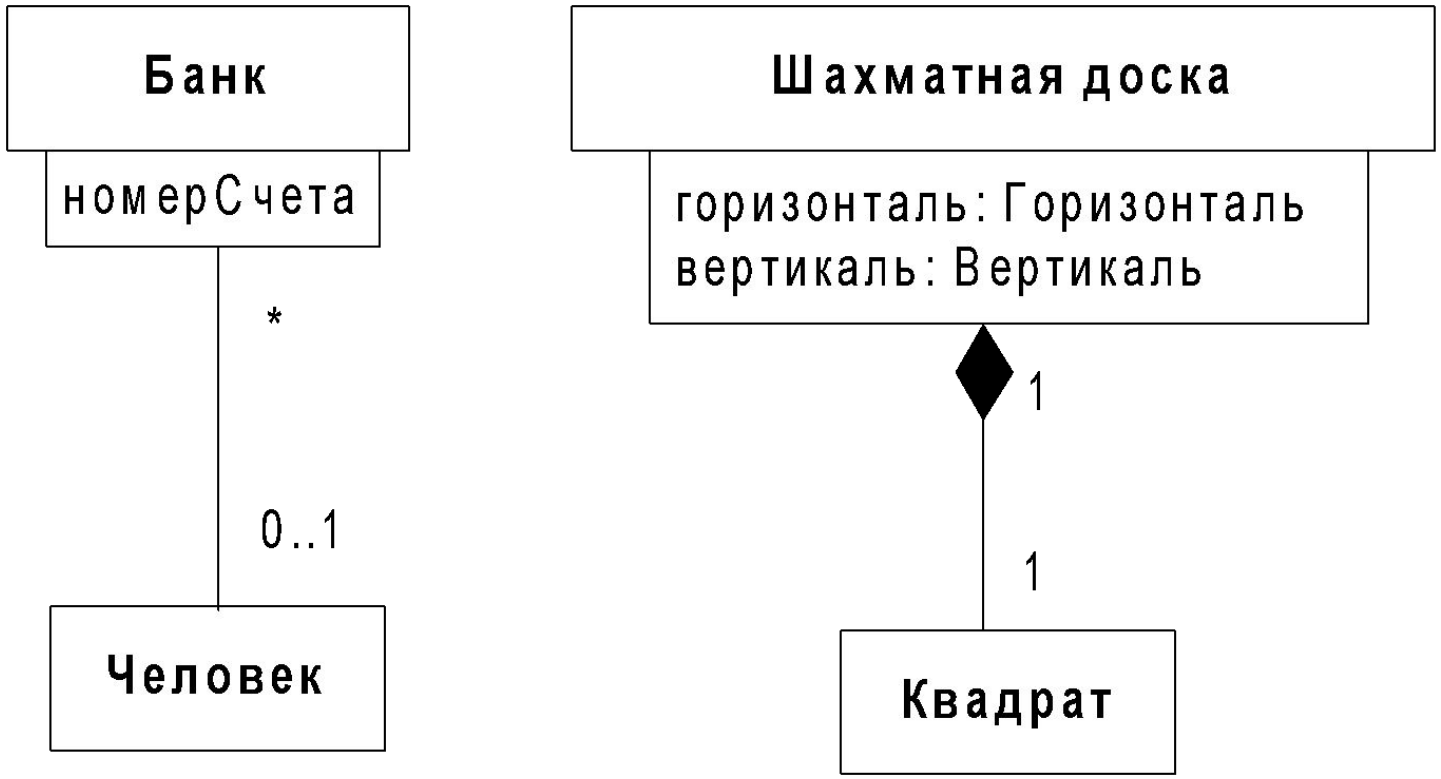
UML Profile for Software Development Processes



- Управляющий класс отвечает за координацию действий других классов. Этому классу посылают мало сообщений, а он рассылает много сообщений
- Граничный класс располагается на границе системы с внешней средой.
- Класс-сущность содержит информацию, которая хранится постоянно и не уничтожается с выключением системы

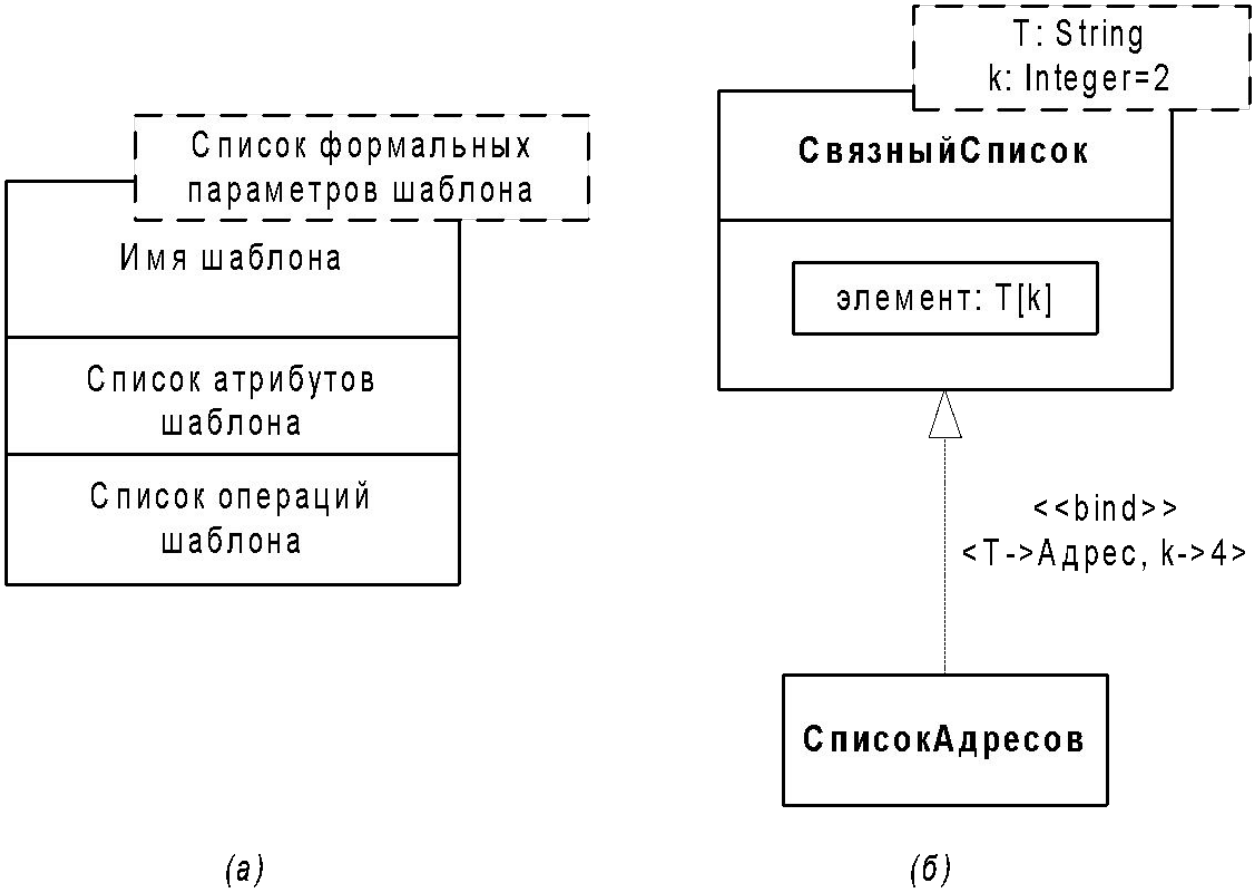
Квалификатор *(qualifier)*

- *Квалификатор* (qualifier) объявляет разбиение множества ассоциированных экземпляров относительно экземпляра на квалифицированном конце ассоциации



Шаблон *(template)*

- – классификатор, который в своем описании имеет несколько формальных параметров



Самостоятельное задание №2

- Выполнить текущее тестирование: вопросы 12-18
- Разработать диаграмму классов для ATM
  - Изобразить следующие классы: CardReader, Screen, Keyboard, Printer, CashDispenser, ATMController, BankController, Transaction, IATMBank.
  - Специфицировать для этих классов атрибуты и операции
  - Изобразить отношения между классами