Объектно-ориентированный подход к моделированию информационных систем

Объектная модель - концептуальная основа моделирования
Основные понятия:

- объекты и атрибуты
- целое и часть
- классы и экземпляры

- ***Объект*** *- осязаемая реальность (tangible entity) - предмет или явление, имеющие четко определяемое поведение*

- ***Класс*** *- множество объектов, связанных общностью структуры и поведения. Класс инкапсулирует (объединяет) в себе данные (атрибуты) и поведение (операции)*

Основные механизмы ООМ:

- абстрагирование
- инкапсуляция
- наследование
- модульность
- иерархия

Абстрагирование – выделение существенных характеристик некоторого объекта, которые отличают его от всех других видов объектов и четко определяют его концептуальные границы с точки зрения дальнейшего рассмотрения и анализа

Инкапсуляция – объединение данных (атрибутов) и поведения (операций) в рамках класса

Наследование – построение новых классов на основе существующих с возможностью добавления или переопределения данных и методов

Модульность – свойство системы, связанное с возможностью ее декомпозиции на ряд внутренне связных, но слабо связанных между собой модулей

Иерархия – ранжированная или упорядоченная система абстракций, расположение их по уровням

Полиморфизм – способность класса принадлежать более чем одному типу

Состояние объекта характеризуется перечнем (обычно неизменным) всех свойств данного объекта и текущими (обычно изменяемыми) значениями каждого из этих свойств. Тот факт, что всякий объект имеет состояние, означает, что всякий объект занимает определенное пространство (физически или в памяти компьютера).

К числу свойств относятся присущие объекту или приобретаемые им характеристики, черты, качества или способности, делающие данный объект самим собой. Эти свойства принято называть *атрибутами класса*

Наследование – это такое отношение между классами, когда один класс частично или полностью повторяет структуру и поведение другого класса (одиночное наследование) или других (множественное наследование) классов. Наследование устанавливает между классами иерархию "общее-частное".

Связи наследования также называют обобщениями (generalization) и изображают в виде больших белых стрелок от класса-потомка к классу-предку

Наследование

- Механизм наследования классов позволяет строить иерархии, в которых производные классы получают элементы родительских, или базовых, классов и могут дополнять их или изменять их свойства.
- Классы, находящиеся ближе к началу иерархии, объединяют в себе наиболее общие черты для всех нижележащих классов.
- По мере продвижения вниз по иерархии классы приобретают все больше конкретных черт:

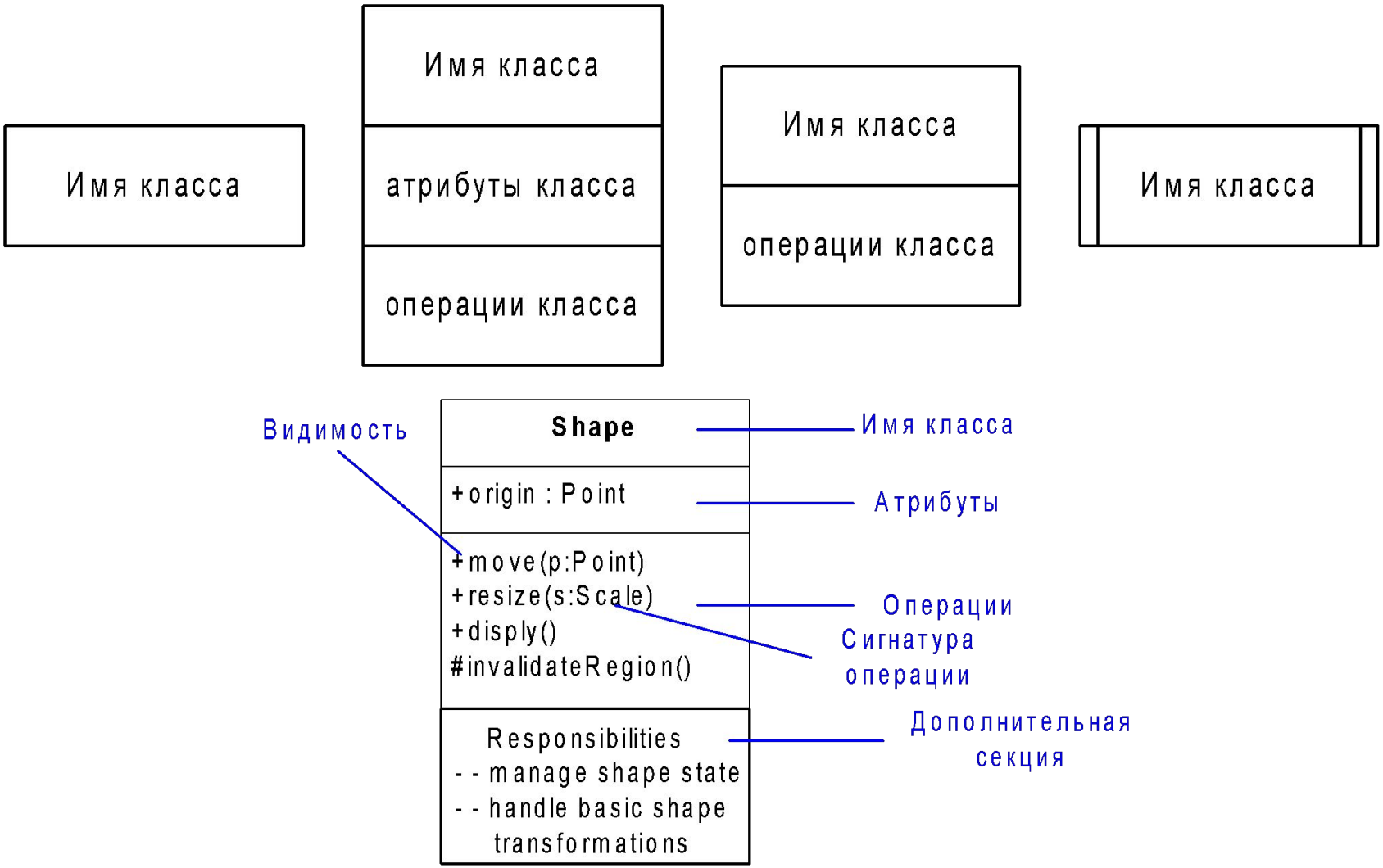
Язык UML в анализе и проектировании бизнес-процессов

Диаграмма классов языка UML 2

Диаграмма классов — основная логическая модель системы

- *Диаграмма классов (class diagram)* — диаграмма, предназначенная для представления модели статической структуры системы в терминологии классов объектно-ориентированного подхода
- 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

Варианты графического изображения класса на диаграмме классов



Примеры записи атрибутов

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

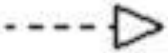
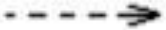
Операции класса

- *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

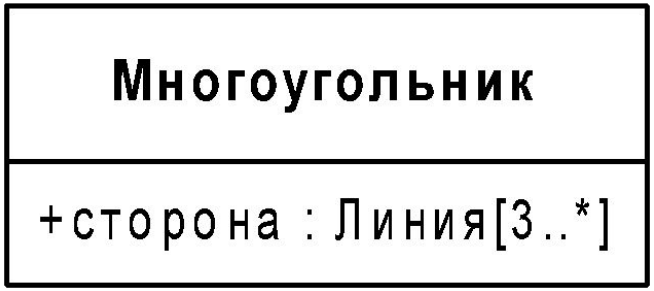
Примеры записи операций:

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

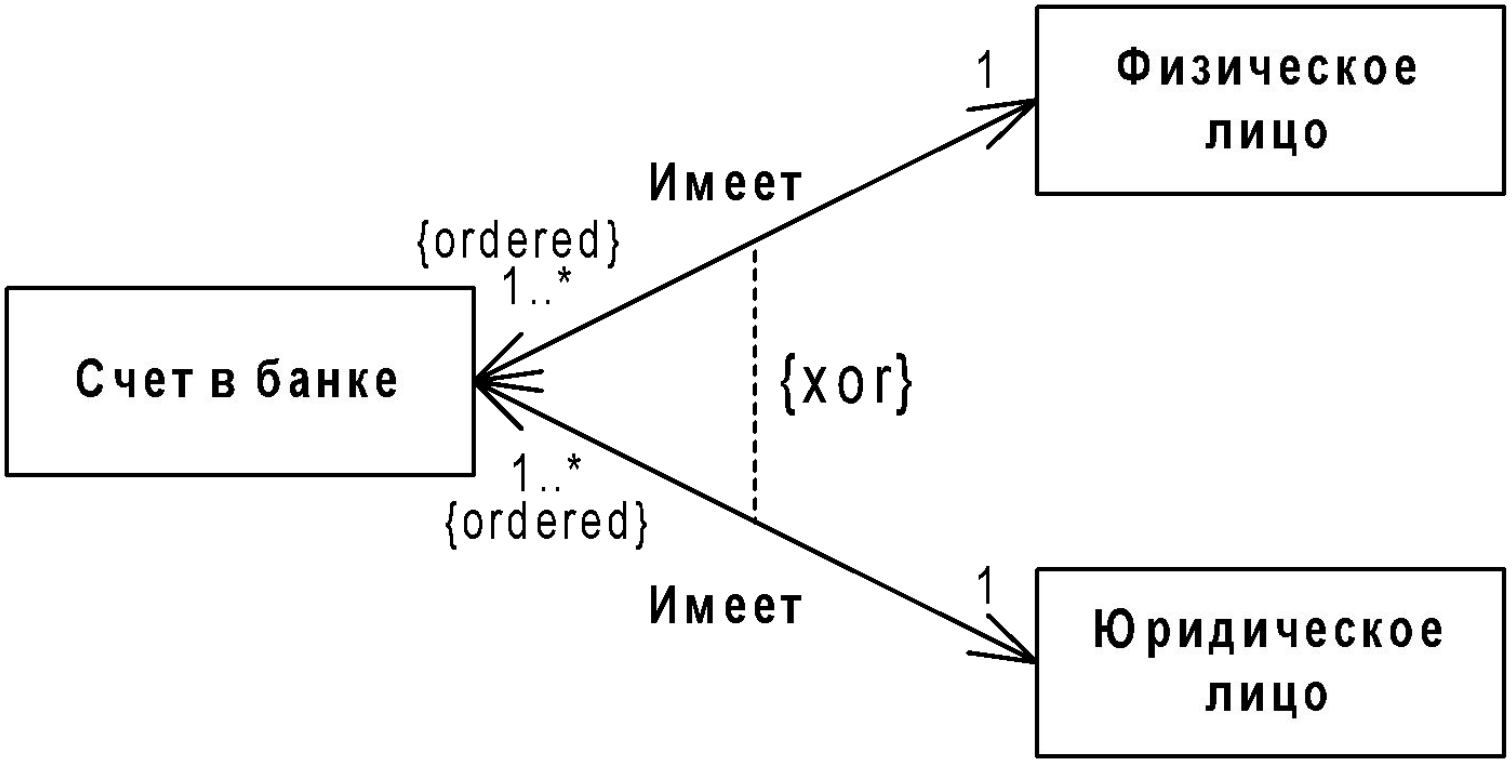
Отношения на диаграмме классов

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	

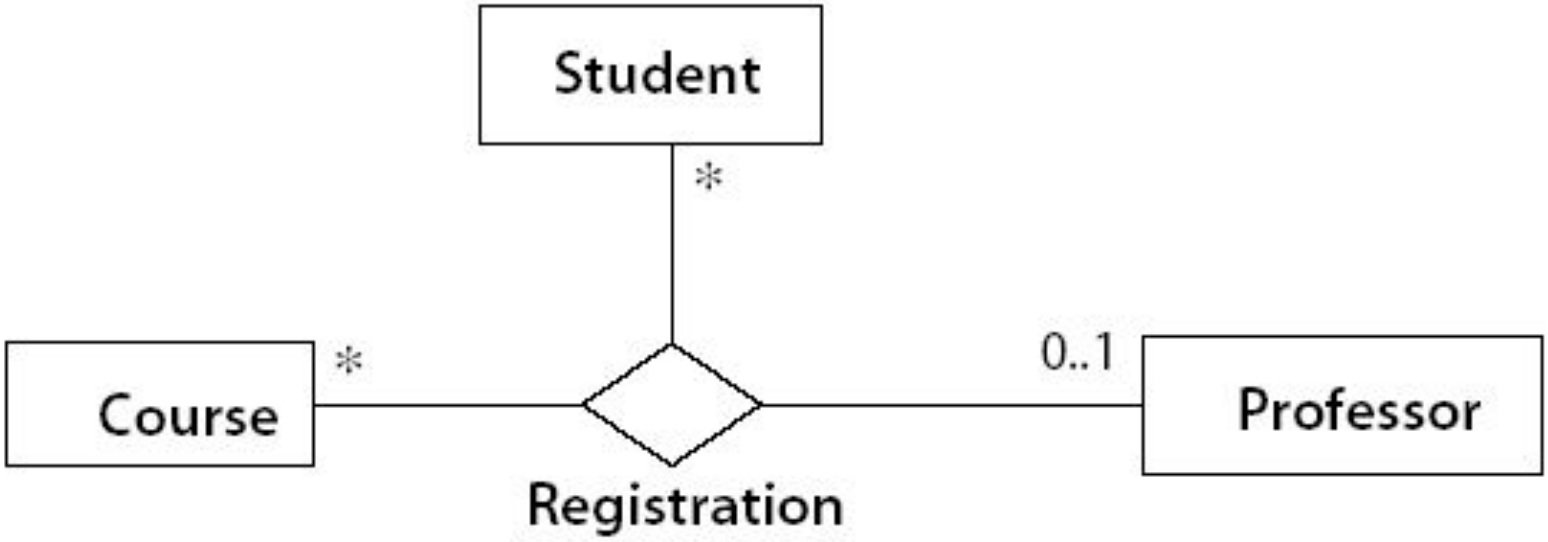
Ассоциация с навигацией и эквивалентное ему представление класса с атрибутом



Исключающая ассоциация между тремя классами

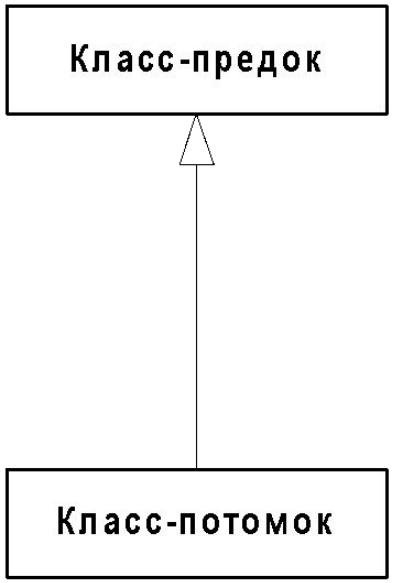


Пример тернарной ассоциации

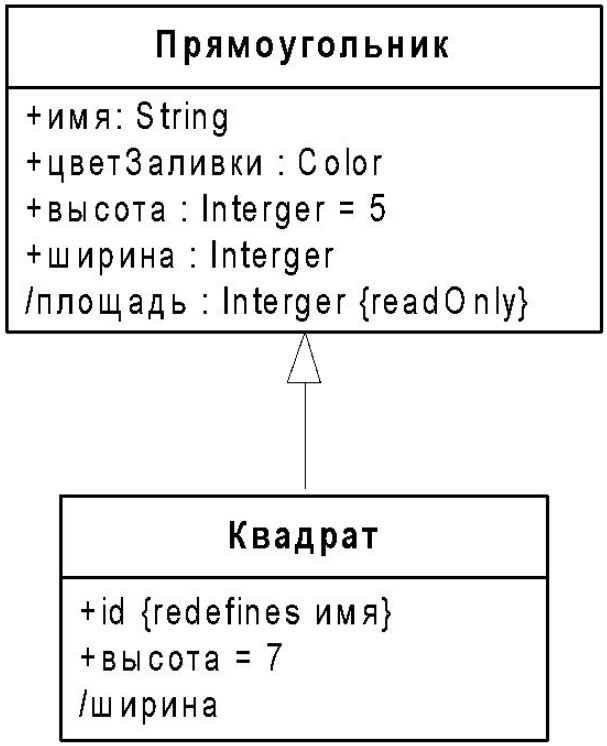


Обобщение *(generalization)*

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

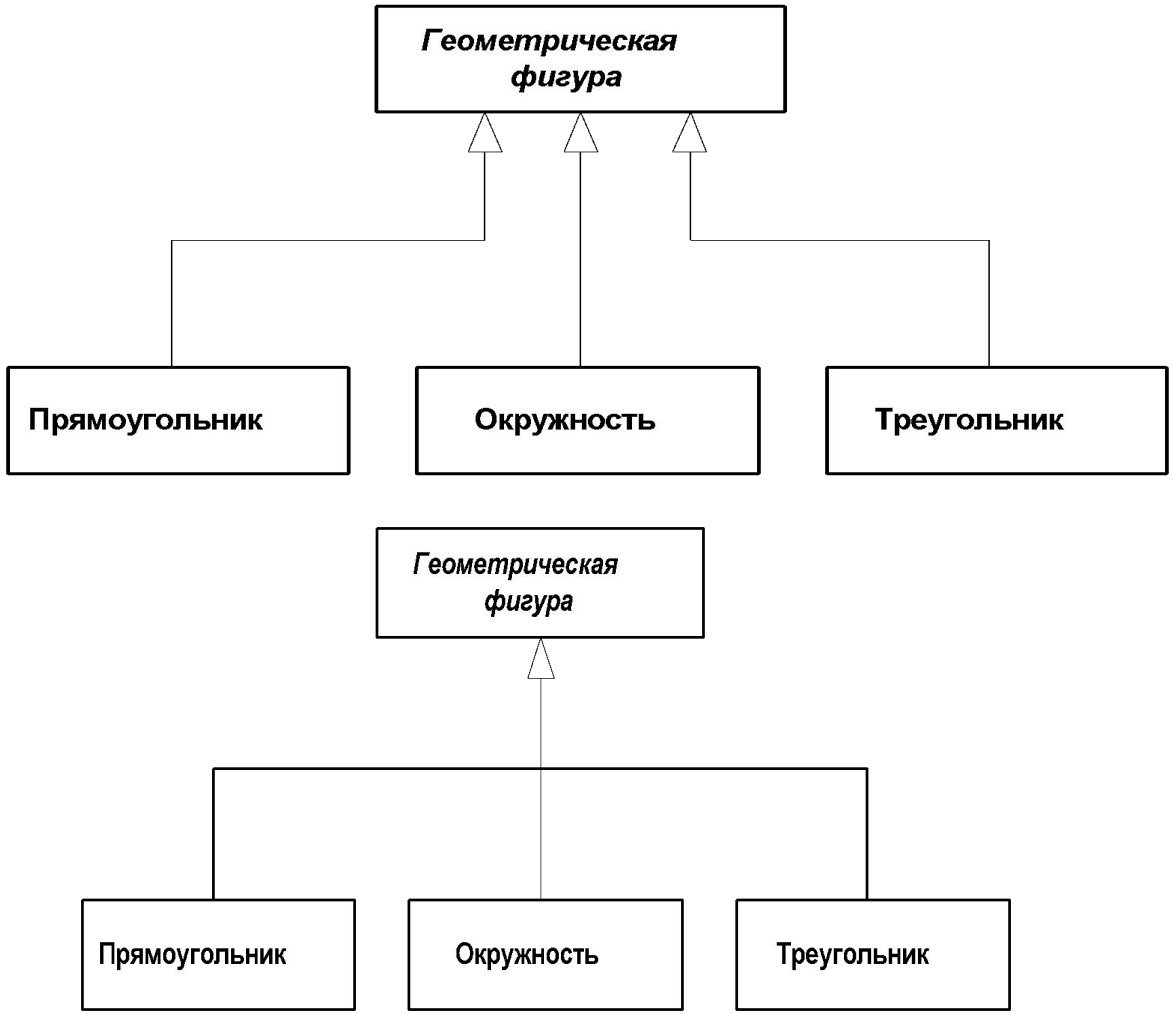


(а)



(б)

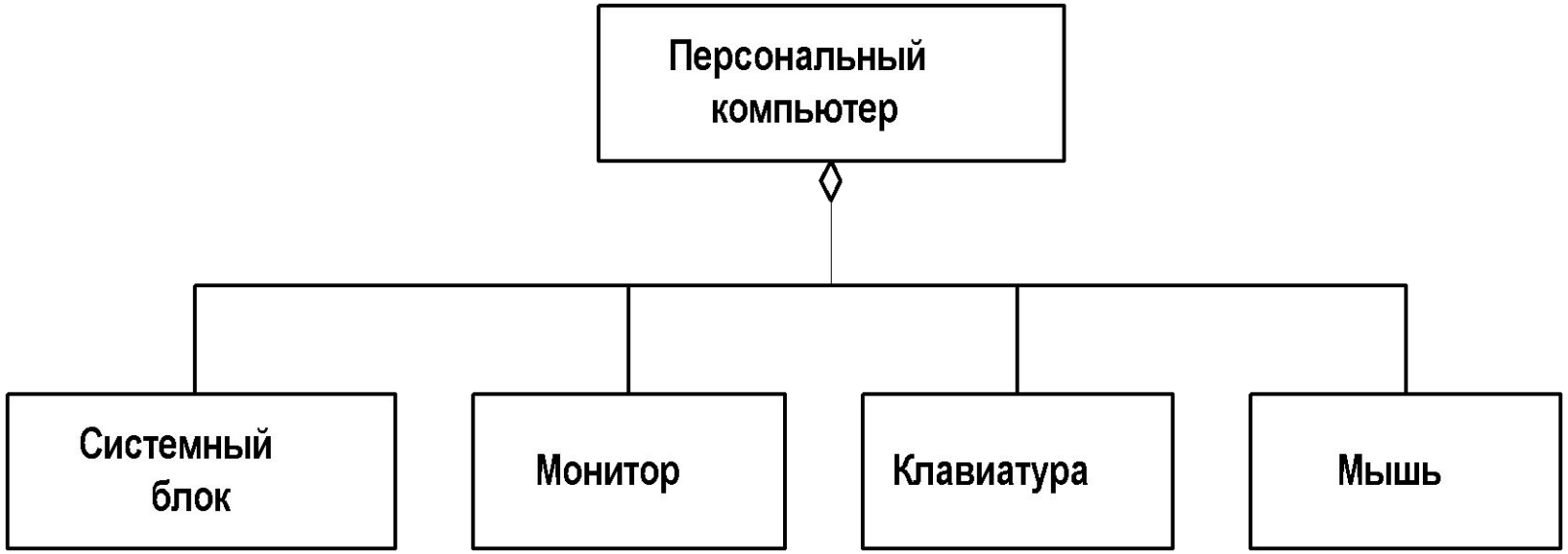
Примеры отношения обобщения



Агрегация *(aggregation)*

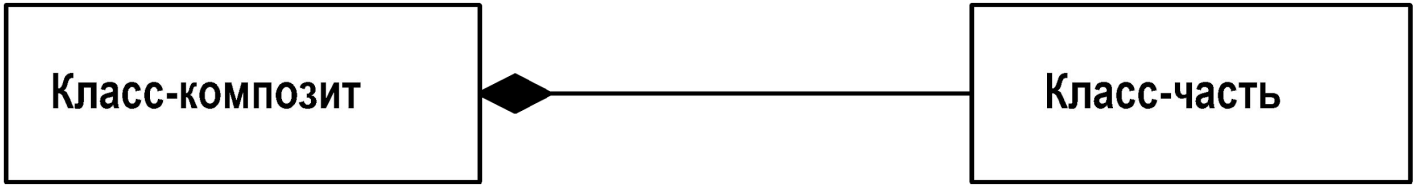
- – направленное отношение между двумя классами, предназначенное для представления ситуации, когда один из классов представляет собой некоторую сущность, которая включает в себя в качестве составных частей другие сущности

Пример отношения агрегации

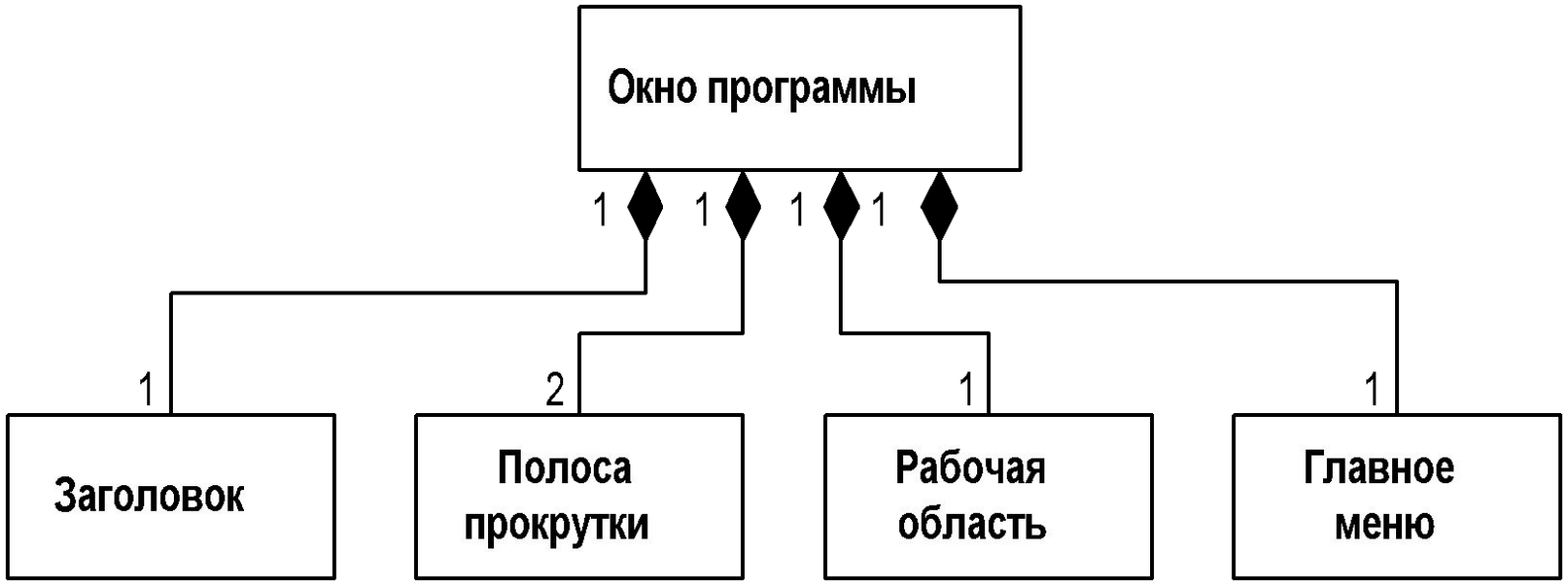


Композиция *(composition)*

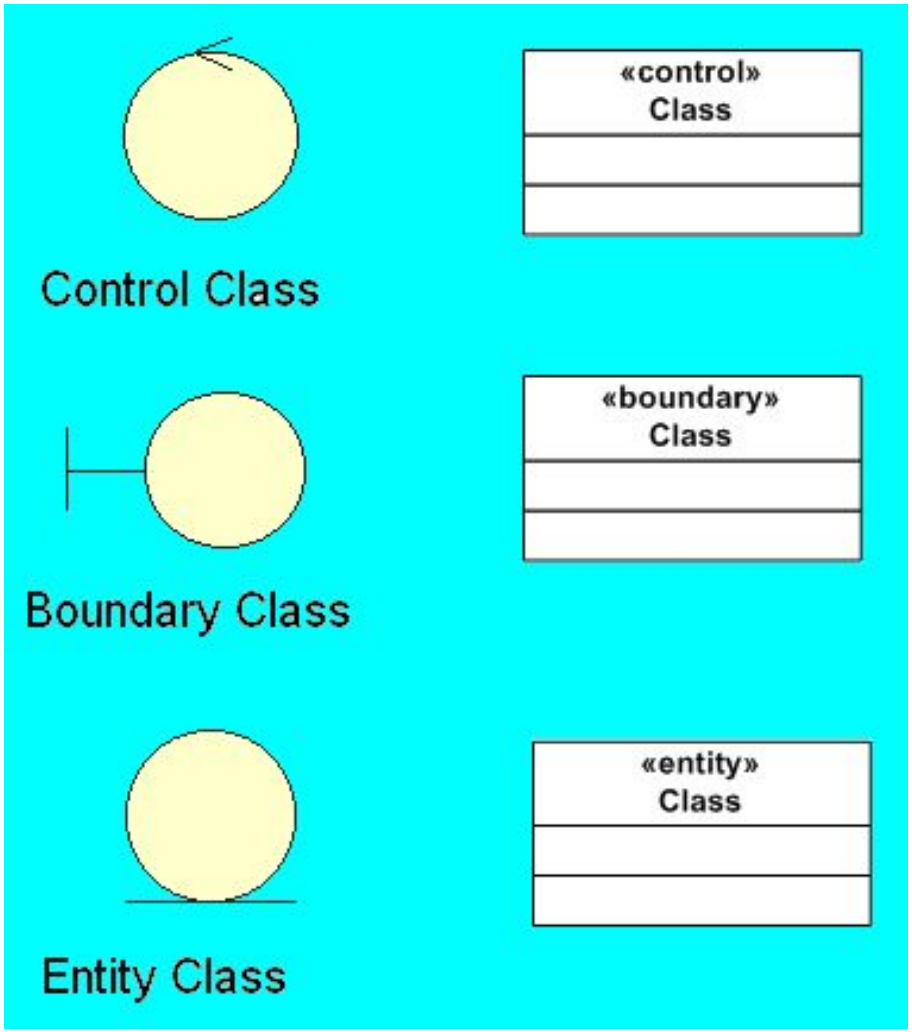
- или *композитная агрегация* предназначена для спецификации более сильной формы отношения "часть-целое", при которой с уничтожением объекта класса-контейнера уничтожаются и все объекты, являющимися его составными частями.



Пример отношения композиции



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

Основные обозначения на диаграмме классов