Программная инженерия

Объектно-ориентированный анализ

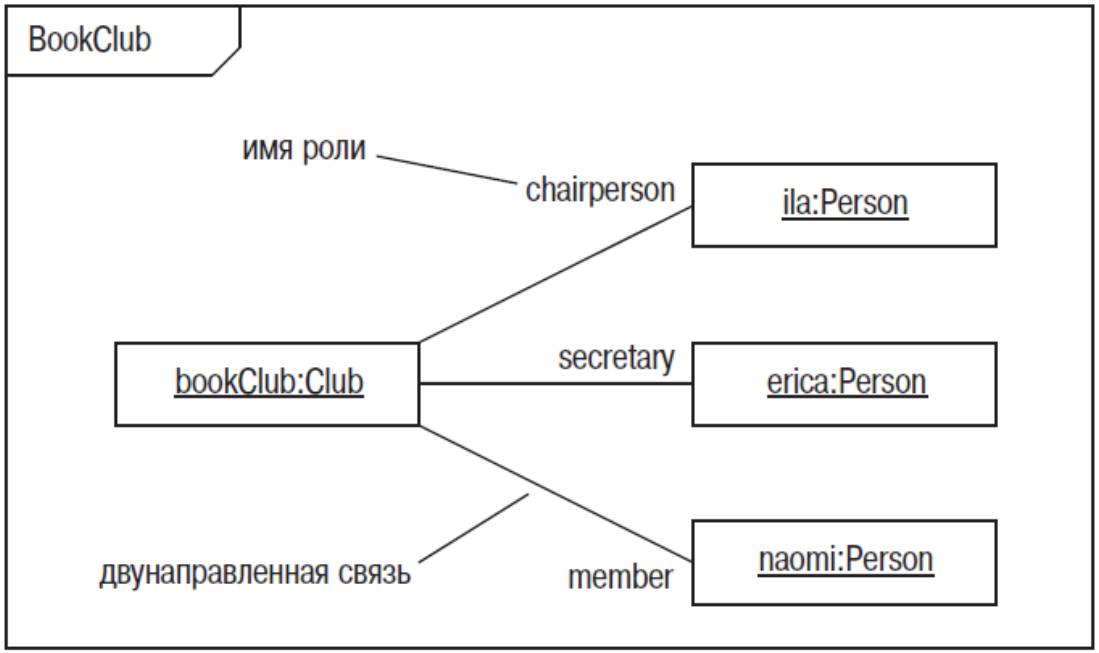
Отношения: связи и ассоциации

Отношения между объектами и между классами

- Отношения – это семантические (значимые) связи между элементами модели.
- Для формирования ОО системы необходимо объединить объекты и классы отношениями.
  - Отношения между **объектами** называются **связями**.
  - Отношения между **классами** называют **ассоциациями**.

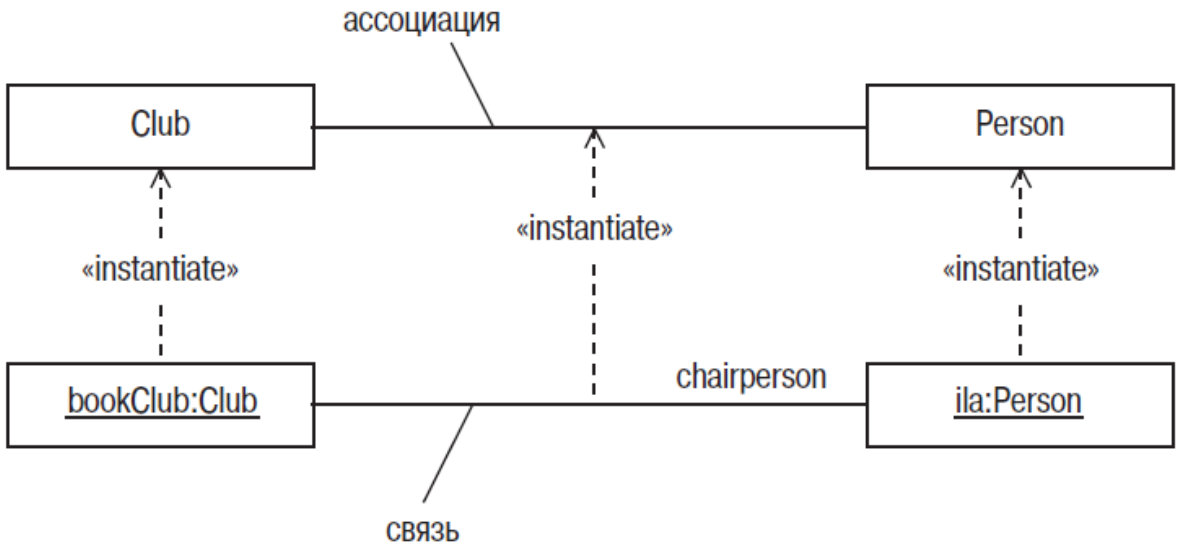
Диаграммы объектов

- Диаграммы объектов – это моментальные снимки работающей ОО системы.



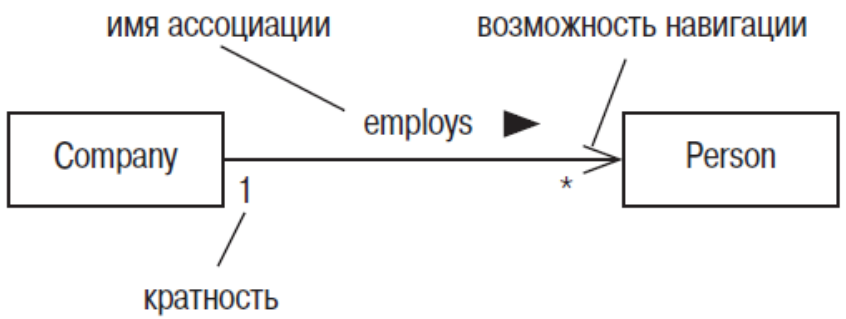
Диаграммы классов: ассоциация

- **Ассоциации** – это соединения между **классам**
  - связь – это экземпляр ассоциации, как объект – экземпляр класса.



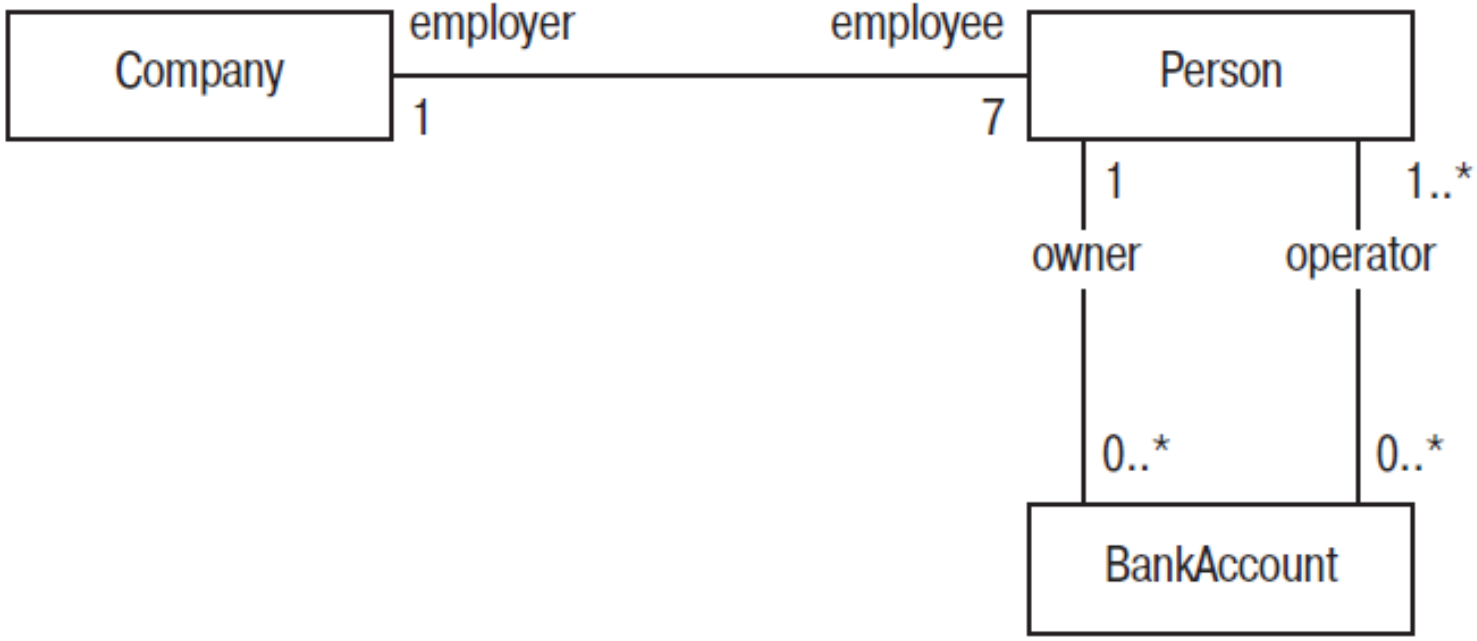
Синтаксис ассоциации

- Ассоциации могут иметь:
  - имя ассоциации
  - имена ролей
  - кратность (Минимум .. Максимум)
  - возможность навигации



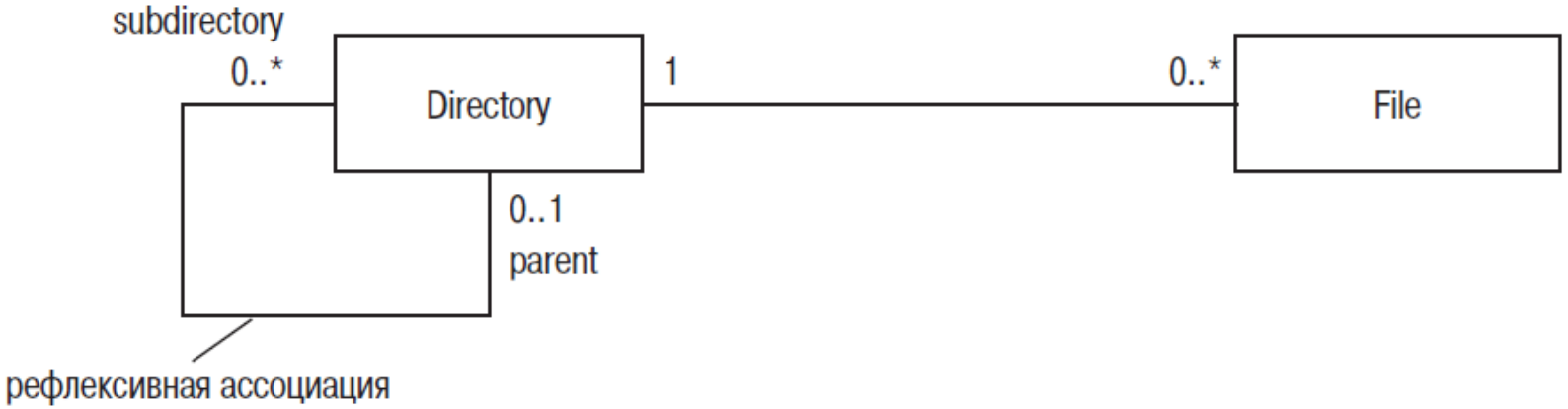
«Компания (Company) нанимает много Человек (Persons)»

Применение ограничений



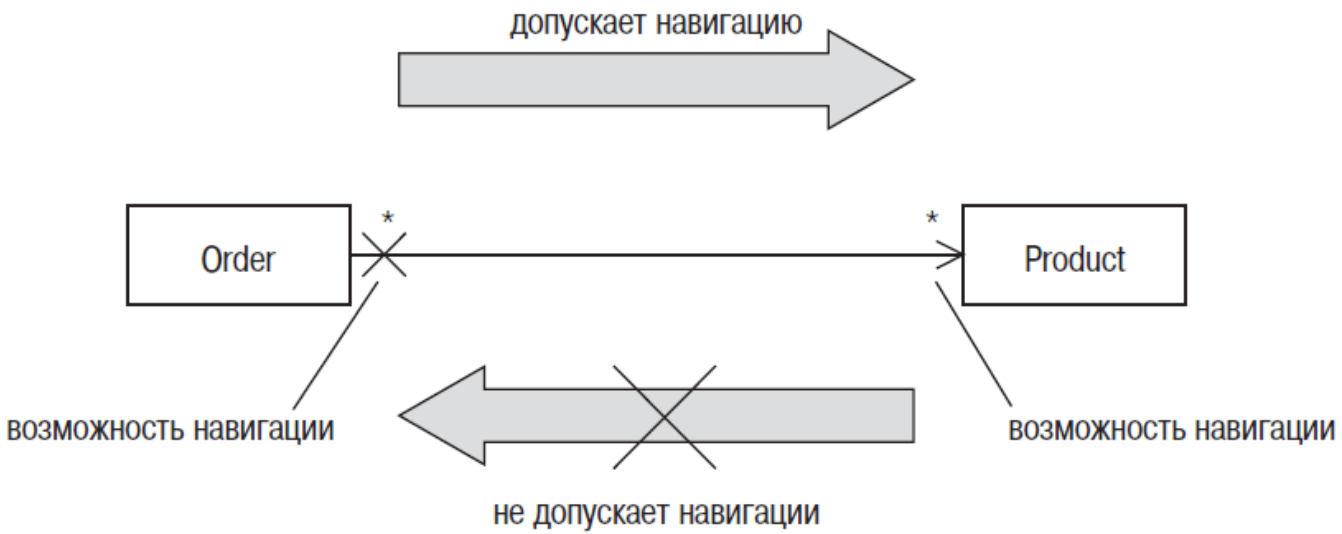
Рефлексивные ассоциации

- Если класс имеет ассоциацию с самим собой, это рефлексивная ассоциация.



Возможность навигации

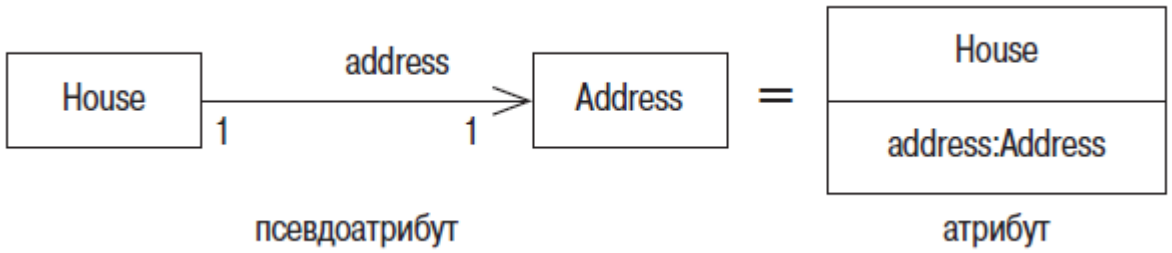
- Возможность навигации показывает, что объекты исходного класса «знают об» объектах целевого класса.
  - **сообщения могут посылаться только в направлении, в котором указывает стрелка**



объект Product не хранит списка объектов Order

Ассоциация и атрибуты

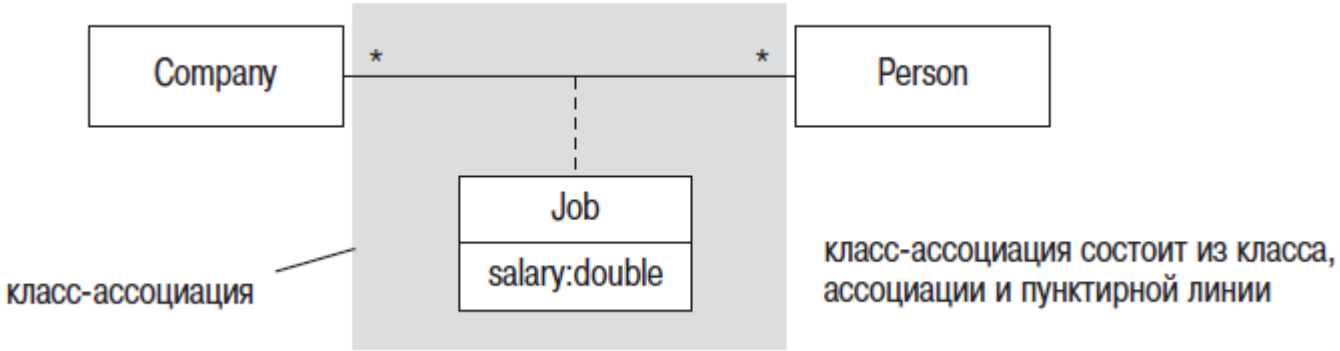
- Ассоциация между исходным и целевым классами означает, что **объекты исходного класса могут сохранять объектную ссылку на объекты целевого класса**.



- Если кратность ассоциации 1 ко многим, то она может быть реализована массивом или коллекцией.

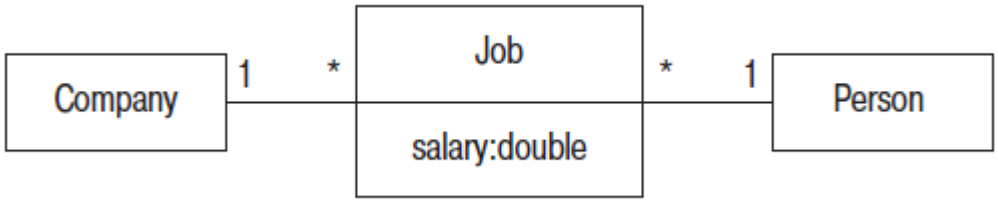
Ассоциации многие-ко-многим

- Отношение *-* реализуется посредством класса-ассоциации
- Класс-ассоциация означает, что в любой момент времени между **любыми двумя объектами может существовать только одна связь**.



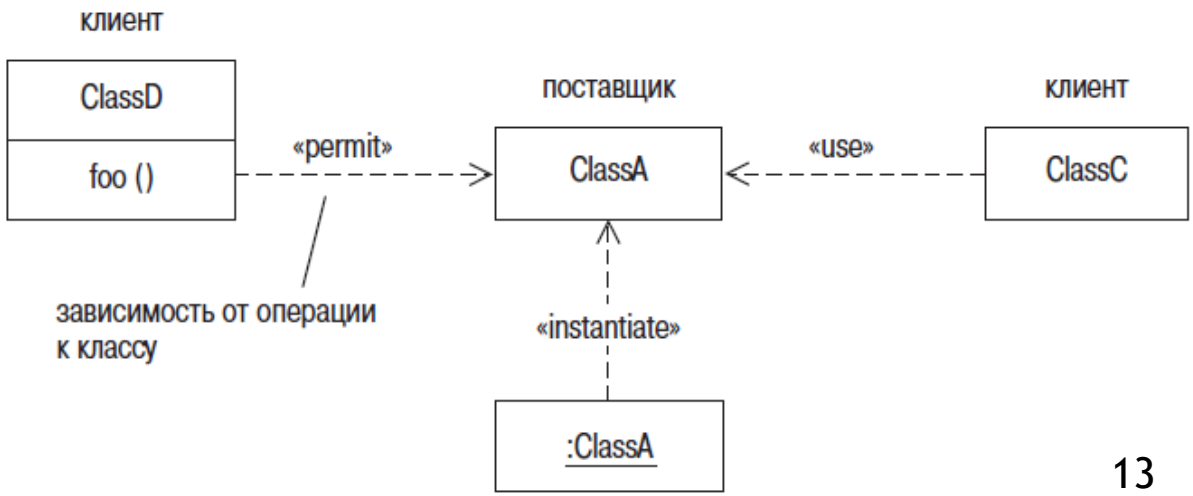
Материализация отношения многие-ко-многим

- **Материализованное отношение** делает возможным существование более одной связи между любыми двумя объектами в конкретный момент времени.



Зависимости

- Зависимость обозначает отношение между двумя или более элементами модели, при котором изменение одного элемента (поставщика) может повлиять или предоставить информацию, необходимую другому элементу (клиенту).
- Обозначается пунктирной стрелкой, возможно с уточнением типа зависимости:
  - Usage
  - Abstraction
  - Permission



Виды зависимостей

Usage	
use	клиент каким-то образом использует поставщика
call	операция-клиент вызывает операцию-поставщика
parameter	поставщик является параметром операции клиента
send	отправление поставщика (сигнал) в некоторую неопределенную цель
instantiate	клиент – это экземпляр поставщика
Abstraction	
trace	поставщик и клиент представляют одно понятие, но находятся в разных моделях
substitute	клиент во время выполнения может заменять поставщика
refine	клиент представляет собой уточнение поставщика
derive	возможность получения одной сущности как производной от другой
Permission	
access	разрешает одному пакету доступ ко всему открытому содержимому другого пакета
import	access + объединение пространств имен
permit	Клиентский элемент имеет доступ к элементу-поставщику не зависимо от объявленной видимости последнего

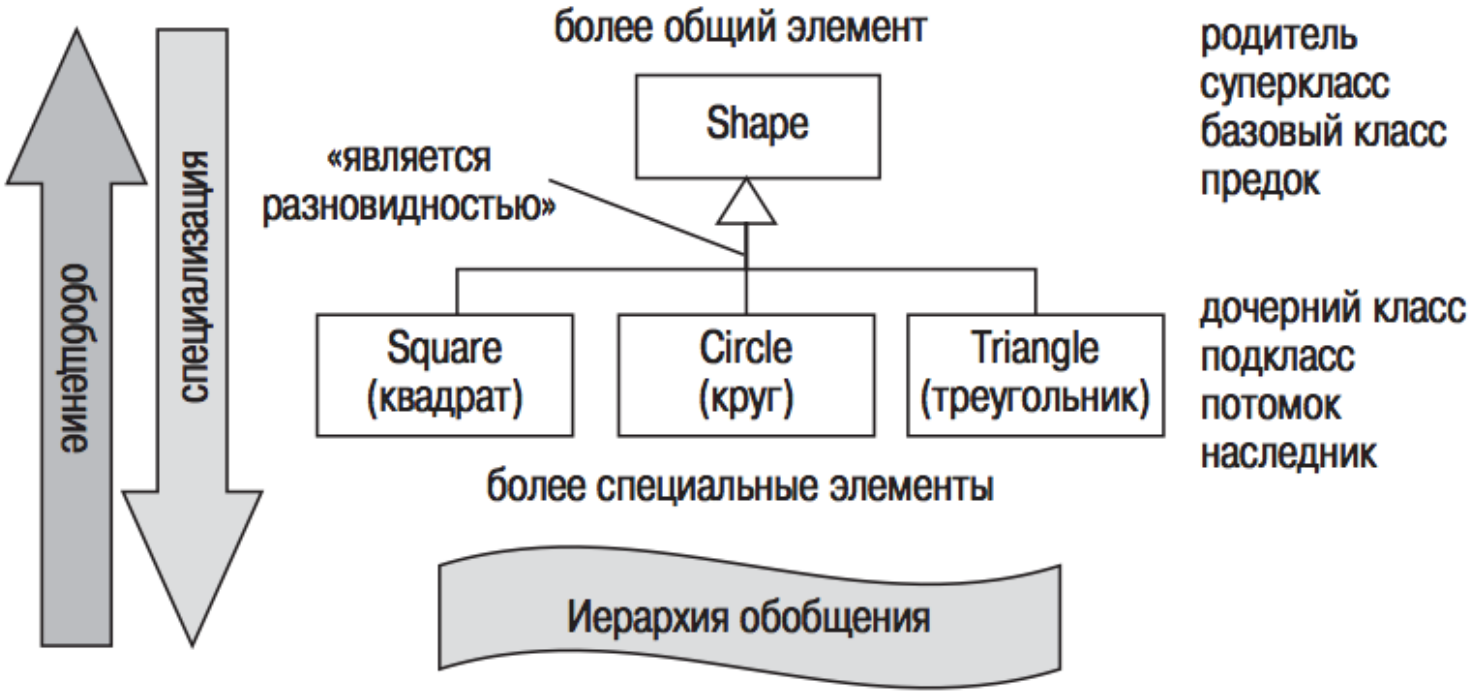
Отношения: наследование

Обобщение

- В основе наследования лежит концепция обобщения.
- Обобщение – это отношение между более общей сущностью и более специальной сущностью, когда более специальный элемент *полностью совместим* с более общим элементом, но содержит большее количество информации.
- Два элемента подчиняются принципу замещаемости: более специальный элемент может использоваться *везде*, где предполагается использование более общего элемента, без нарушения системы (принцип подстановки Барбары Лисков).

Обобщение

Иерархия обобщения создается путем обобщения более специальных сущностей и специализации более общих сущностей.



Наследование

- Наследование: это отношение между классами, посредством которого дочерние наследуют все возможности родительских (более общих) классов.
  - атрибуты;
  - операции;
  - отношения;
  - ограничения.

Наследование

- Подклассы переопределяют унаследованные операции, предоставляя новую операцию с такой же сигнатурой.
- Чтобы переопределить операцию класса-родителя, дочерний класс должен предоставить операцию с *точно* такой же сигнатурой, что и у переопределяемой операции класса-родителя.
- Отсутствие реализации операции можно обозначить, сделав ее *абстрактной операцией*. В UML для этого имя операции просто записывается *курсивом*.
- Класс с одной или более абстрактными операциями является неполноценным, поскольку в нем есть операции, не имеющие реализации. Такой класс называется *абстрактным*, и его имя также *выделяется курсивом*.

Наследование