Программная инженерия

Объектно-ориентированный анализ

Анализ прецедента

- Деятельность UP «**Анализ прецедента**» включает:
  - создание классов анализа
  - реализации прецедентов
- **Классы анализа** – это классы, которые представляют четкую абстракцию предметной области и должны проецироваться на реальные бизнес-понятия;
- **Реализации прецедентов** – это кооперации объектов, показывающие, как системы взаимодействующих объектов могут реализовывать поведение системы, описанное в прецеденте.

Классы анализа

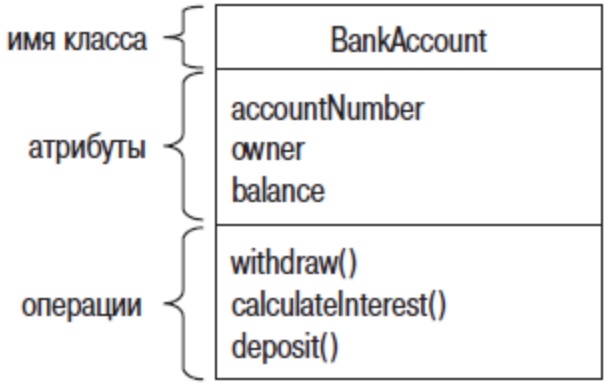
Класс анализа

- Класс анализа должен четко и однозначно проецироваться в реальное прикладное понятие.

- Следовательно, задача ОО аналитика – попытаться прояснить беспорядочные или несоответствующие прикладные понятия и превратить их в то, что может стать основой для класса анализа.

Состав класса анализа

- В классах анализа содержатся только ключевые атрибуты и обязанности, определенные на очень высоком уровне абстракции. *Указывают атрибуты, которые, возможно, будут присутствовать в проектных классах*.
- Однако одна операция уровня анализа очень часто разбивается на несколько операций уровня проекта.



Признаки хорошего класса анализа

- его имя отражает его назначение
- он является четкой абстракцией, моделирующей один конкретный элемент предметной области
- у него небольшой четко определенный набор обязанностей
- у него высокая внутренняя связность (cohesion)
- у него низкая связанность с другими классами (coupling)

Подходы к созданию классов анализа

- 3-5 методов в классе – оптимальное количество
- Классы не должны быть изолированными
- Не создавать много мелких классов (1-2 метода)
- Не объединять все в крупные классы (> 6 методов)
- Избегать всемогущих классов («…System» «…Controller»)
- Избегать глубокой иерархии наследования

Выявление классов анализа

Нет универсального подхода, дающего 100% результат, но существуют проверенные методы:

- Метод «Существительное/Глагол»
- Метод CRC-анализа
- Применение стереотипов RUP
- Применение готовых шаблонов классов анализа

Существительное/глагол

- Анализируются: модель требований; модель прецедентов; документы с постановкой задачи.
- Существительные и именные группы указывают на классы или атрибуты. Глаголы и глагольные группы служат признаком обязанностей или операций.
- Результат – список с потенциальными классами, атрибутами и обязанностями (методами). После чего – распределение атрибутов и методов по классам.

CRC-анализ

- CRC – class-responsibilities-collaborators (класс – обязанности – участники)
- CRC – это техника мозгового штурма, при которой важные моменты предметной области записываются на стикерах.

Имя класса: BankAccount	
Обязанности: поддерживать остаток	Участники: Bank

Стереотипы RUP

- Согласно RUP считается полезным поискать классы, которые можно обозначить стереотипами **«boundary» (граница), «control» (управление)** и **«entity» (сущность).**

Стереотип	Пиктограмма	Семантика
«boundary»		Класс, который является посредником во взаимодействии между системой и ее окружением.
«control»		Класс, инкапсулирующий характерное для прецедента поведение.
«entity»		Класс, используемый для моделирования постоянной информации о чем-то.

Шаблоны классов анализа

- Базовые шаблоны (паттерны) могут предоставить готовые компоненты аналитической модели.
- Есть несколько книг, в которых приводятся примеры таких паттернов:
  - «Enterprise Patterns and MDA: Building Better Software with Archetype Patterns and UML», Jim Arlow, Ila Neustadt, AddisonWesley, 2004
  - Архитектура корпоративных программных приложений. Мартин Фаулер. Вильямс, 2007 г.
- Шаблоны Order, Party, Product, Rule …

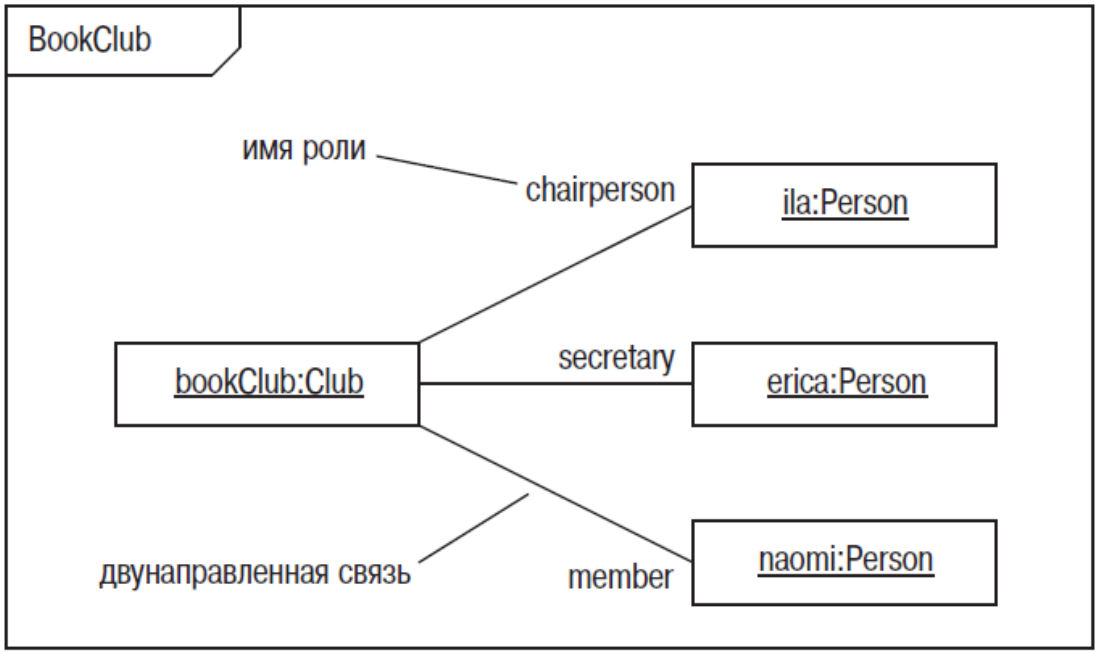
Отношения: связи и ассоциации

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**.

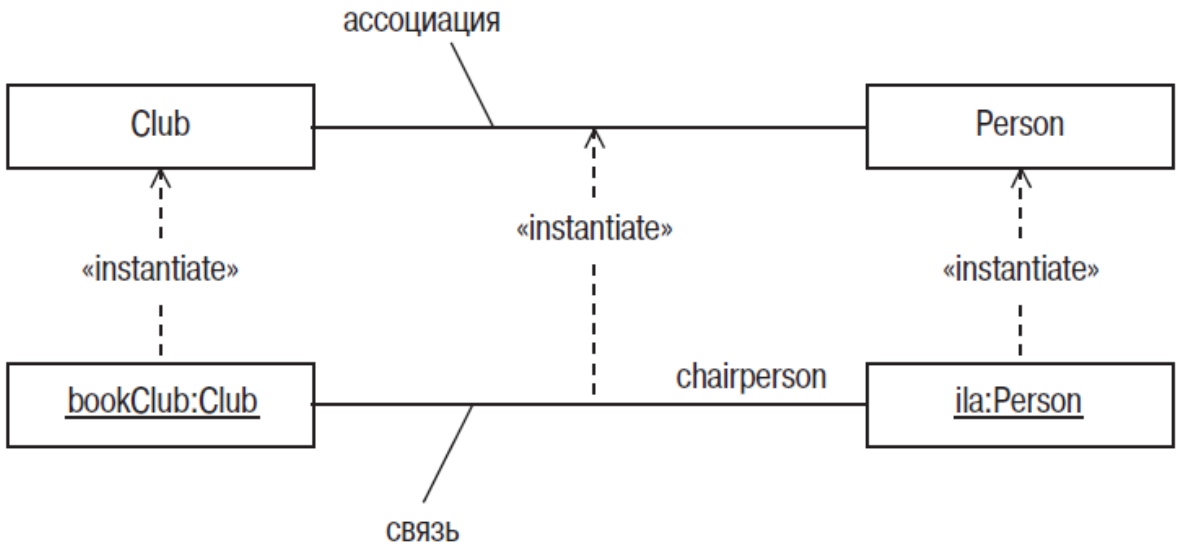
Диаграммы объектов

- Диаграммы объектов – это моментальные снимки работающей ОО системы.



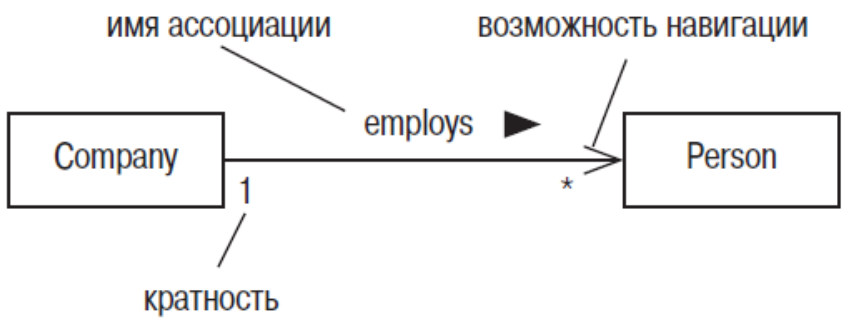
Диаграммы классов: ассоциация

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



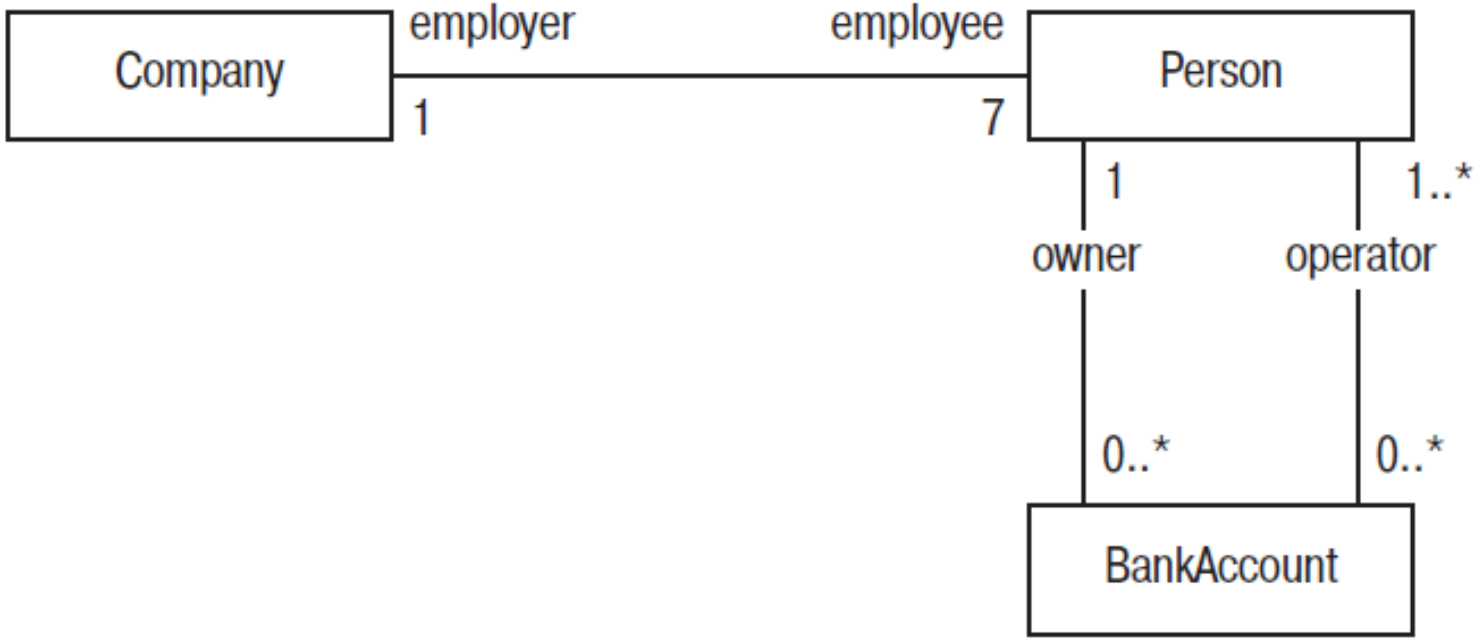
Синтаксис ассоциации

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



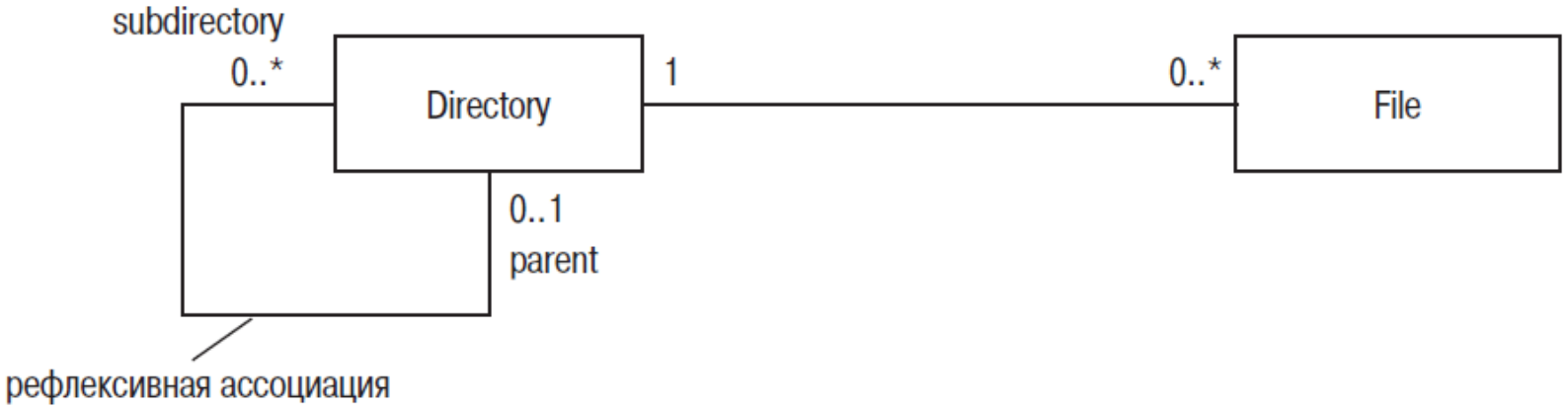
«Компания (Company) нанимает много Человек (Persons)»

Применение ограничений



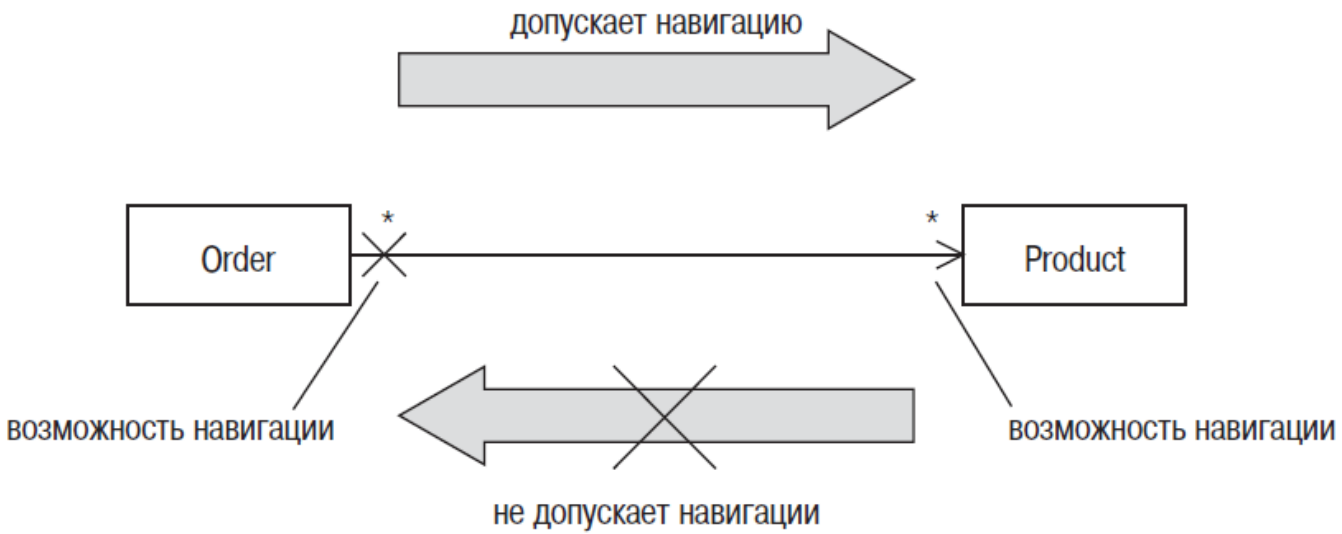
Рефлексивные ассоциации

- 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

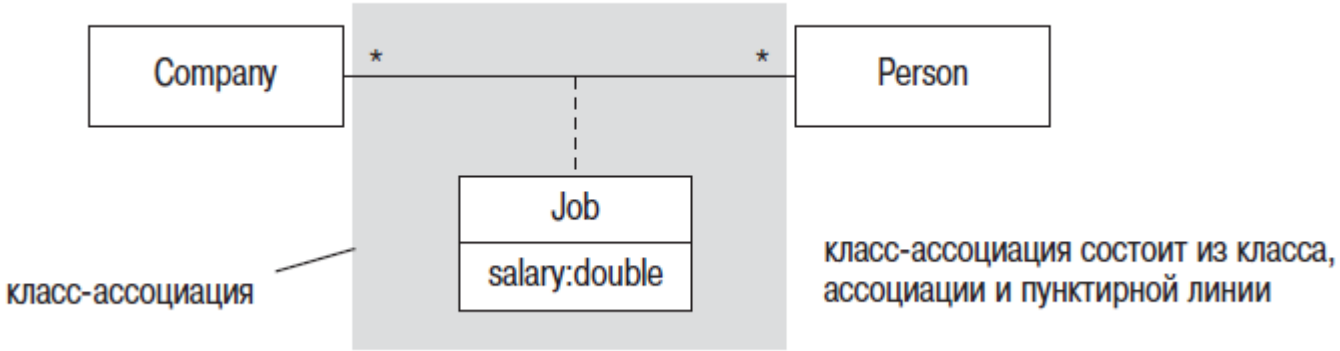
Ассоциация и атрибуты

- 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.

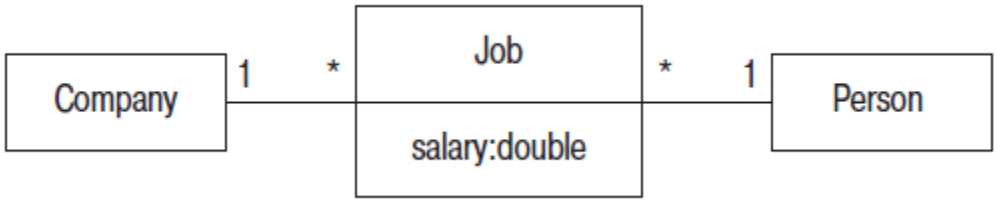
Ассоциации многие-ко-многим

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



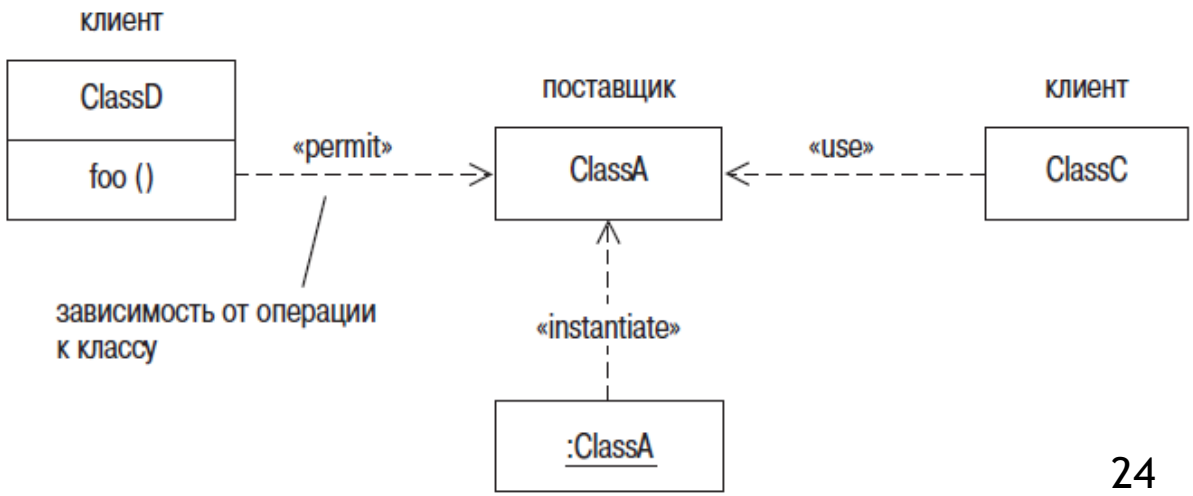
Материализация отношения многие-ко-многим

- **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