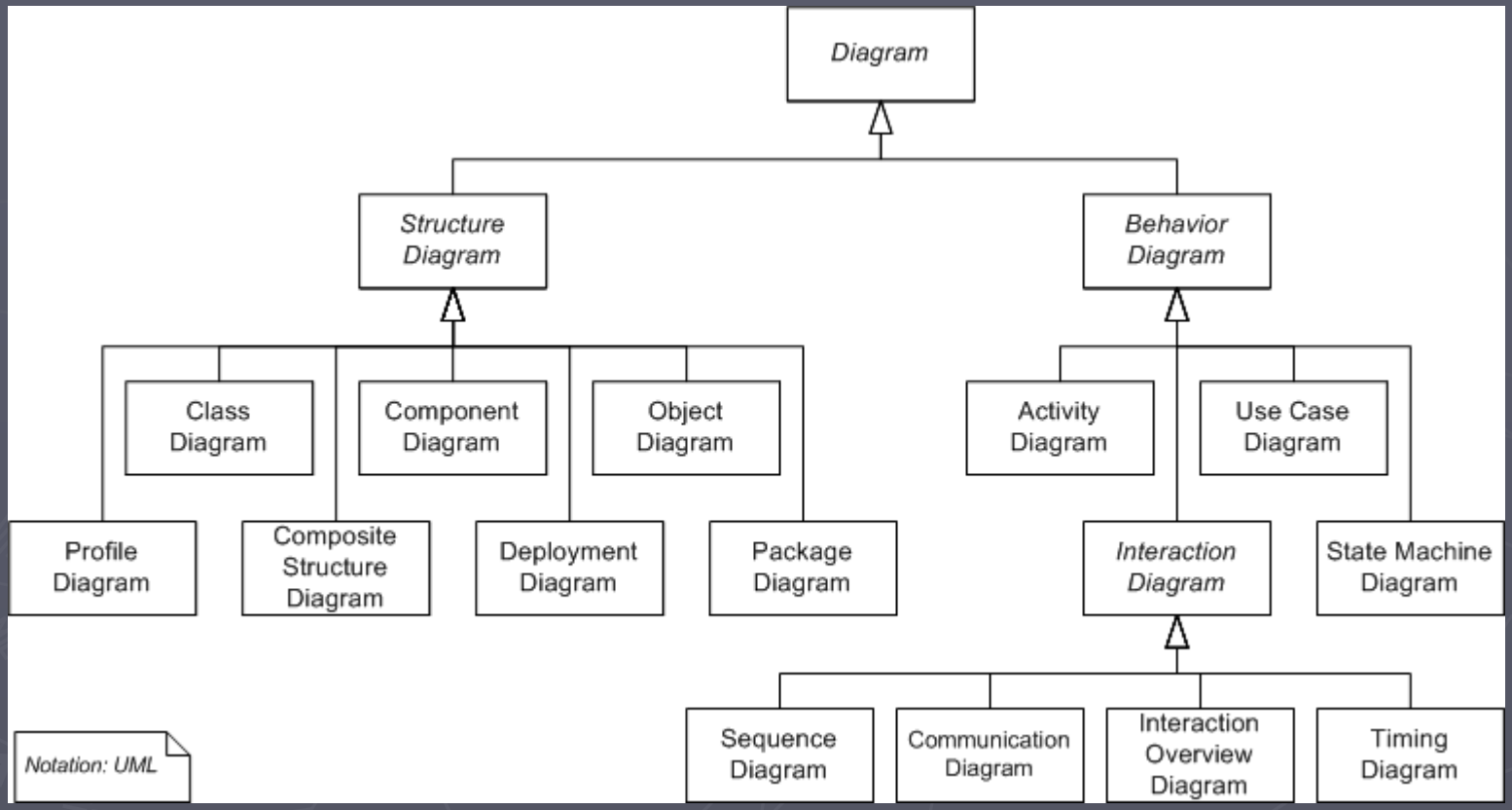
Проектирование отношений

Связи

- Связь представляет собой семантическую взаимосвязь между классами

- ассоциации
  - Агрегации
  - Композиции
- зависимости
- обобщения (наследования)

UML (Unified Modeling Language) - унифицированный язык моделирования



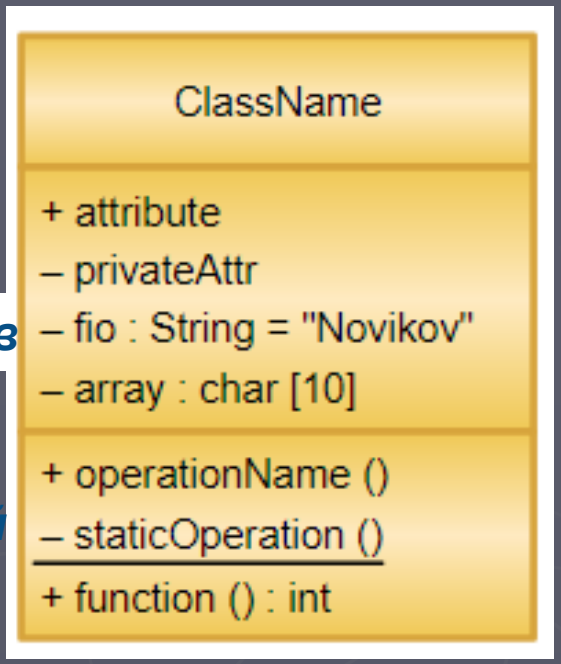
https://www.omg.org/spec/UML/2.5/PDF/

Классы

секция имени

секция атрибутов

секция операций



видимость ИМЯ кратность : тип = начальное_значение {свойства}

Стандартные стереотипы классов

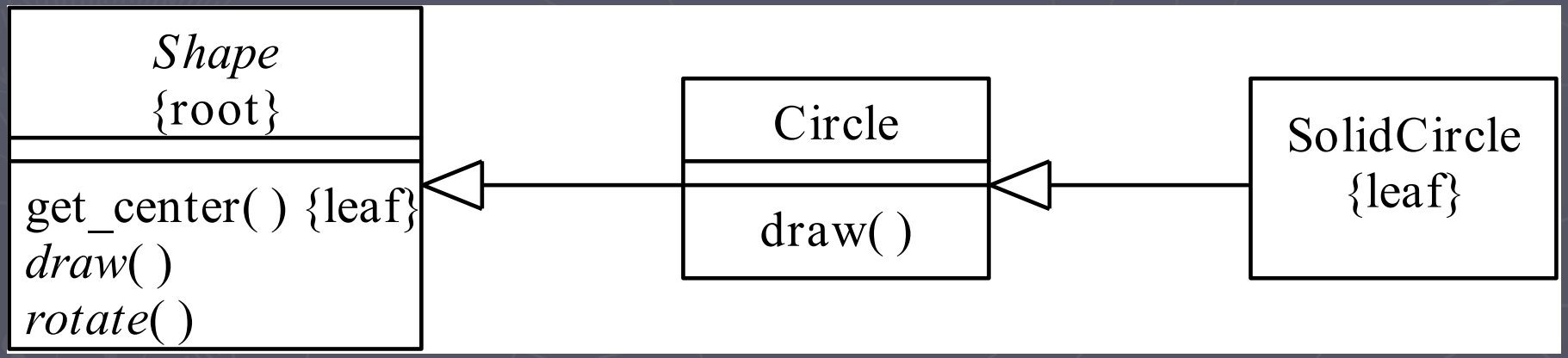
Стереотип	Описание
«actor»	Действующее лицо
«auxiliary»	Вспомогательный класс
«enumeration»	Перечислимый тип данных
«exception»	Исключение (только в UML 1)
«focus»	Основной класс
«implementationClass»	Реализация класса
«interface»	Все составляющие абстрактные
«metaclass»	Экземпляры являются классами
«powertype»	Метакласс, экземплярами которого являются все наследники данного класса (только в UML 1)
«process»	Активный класс
«thread»	Активный класс (только в UML 1)
«signal»	Класс, экземплярами которого являются сигналы
«stereotype»	Новый элемент на основе существующего
«type»	Тип данных
«dataType»	Тип данных
«utility»	Нет экземпляров, служба

- *открытый* (обозначается знаком `+` или ключевым словом `public`);
- *защищенный* (обозначается знаком `#` или ключевым словом `protected`);
- *закрытый* (обозначается знаком `–` или ключевым словом `private`).
- *пакетный* (обозначается знаком `~` или ключевым словом `package`).

а) Обобщение

Генерализация-расширение-наследование

- показывают связи наследования между классами
- Является IS A



б)Композиция

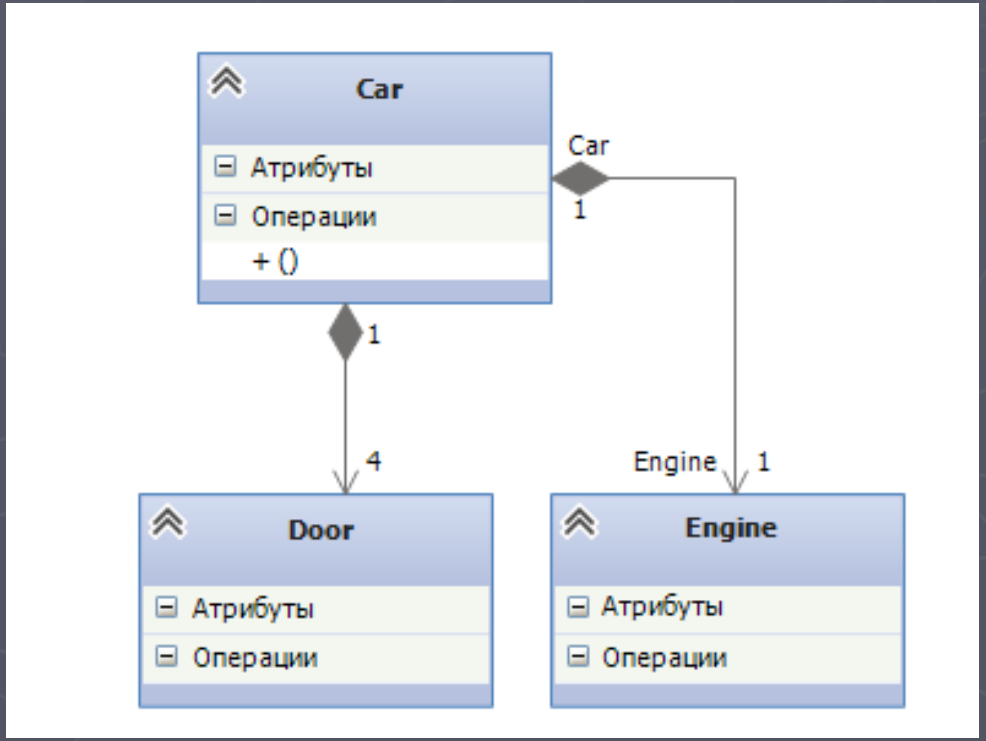
Композиция **(composition) – это ассоциация между классом А (часть) и классом В(целое), которая дополнительно накладывает более сильные ограничения в сравнении с агрегацией: композиционно часть А может входить только в одно целое В, часть существует, только пока существует целое и прекращает свое существование вместе с целым.**

Состоит из

Отношение «содержит как часть»

HAS-A

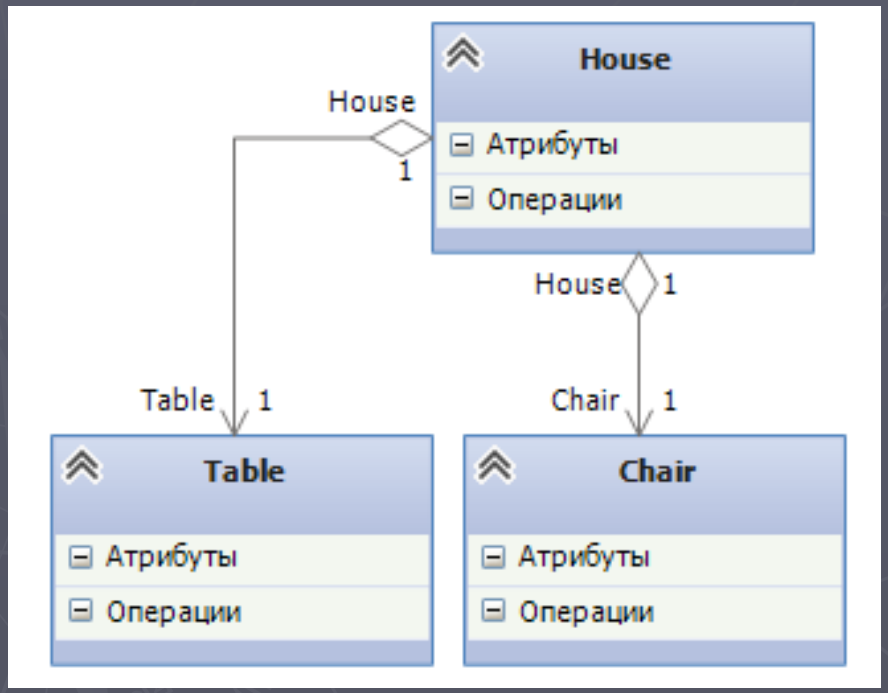
Целое контролирует время жизни своей составной части (часть не существует без целого) - сильно связана



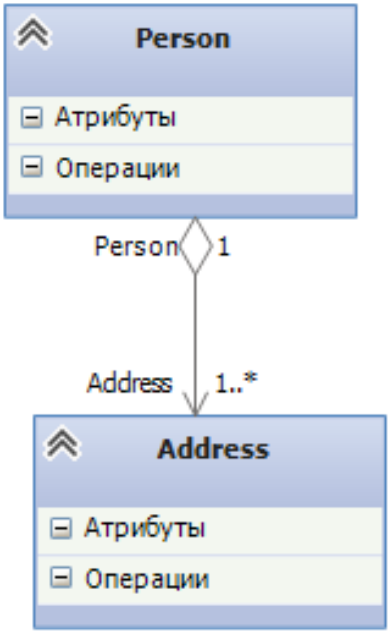
Целое контролирует время жизни своей составной части (часть не существует без целого) - сильно связана

B) Агрегация

- связь между целым и его частью
- Включает в себя - contains
- Объекты м.б. равноправные



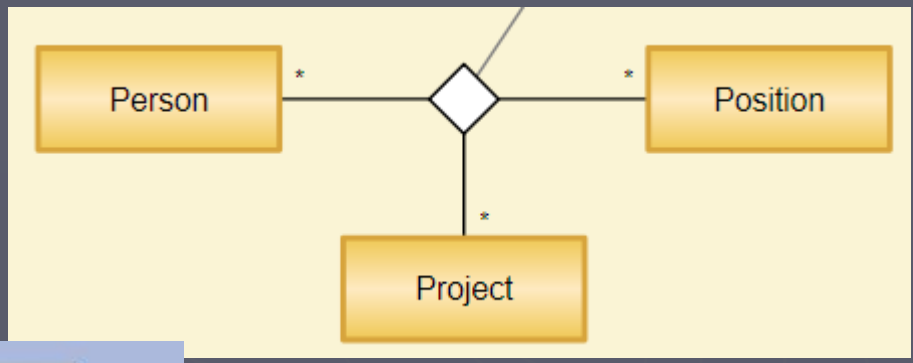
Целое хоть и содержит свою составную часть, время их жизни не связано

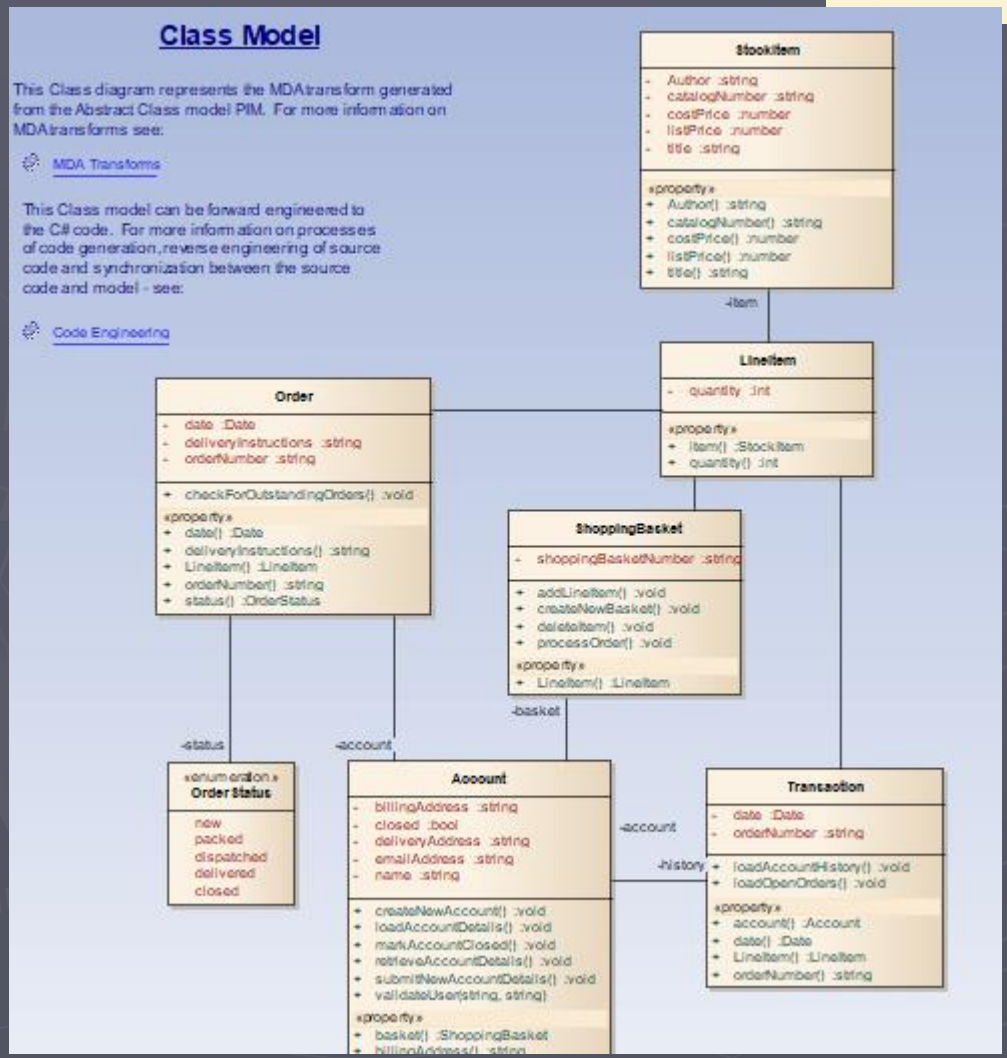


Агрегация **(aggregation) – это ассоциация между классом A (часть) и классом B (целое), которая означает, что экземпляры (один или несколько) класса A входят в состав экземпляра класса B.**

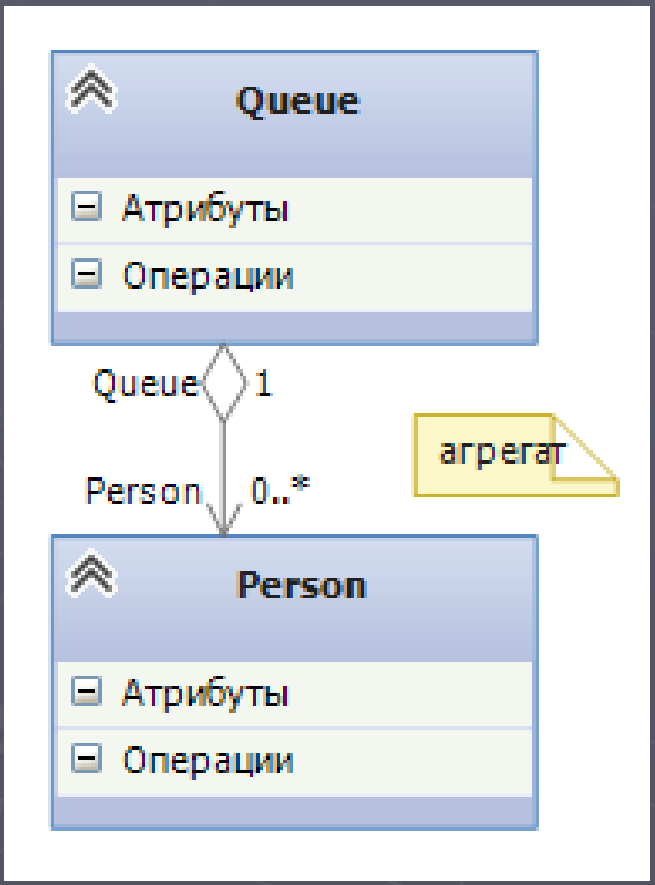
Целое хоть и содержит свою составную часть, время их жизни не связано

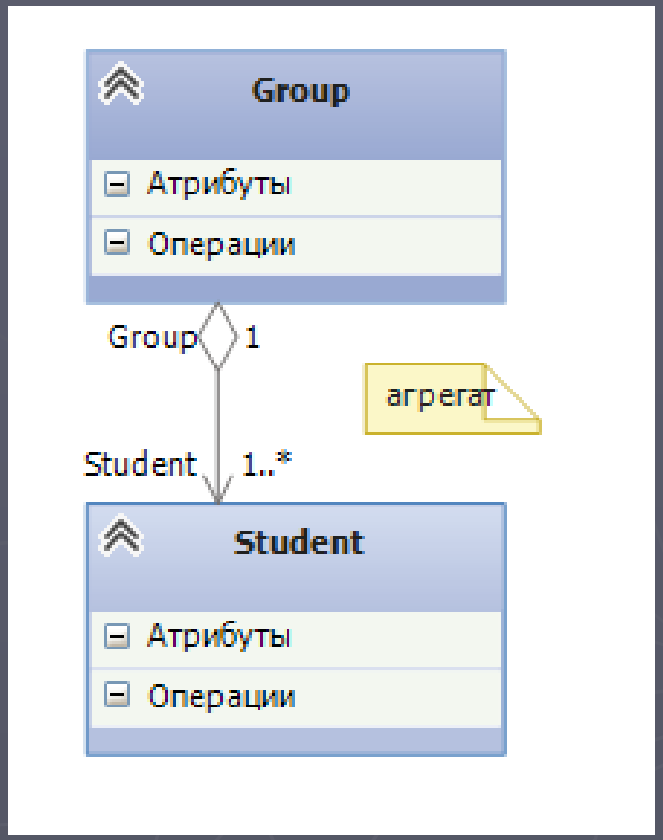
Многополюсная ассоциации





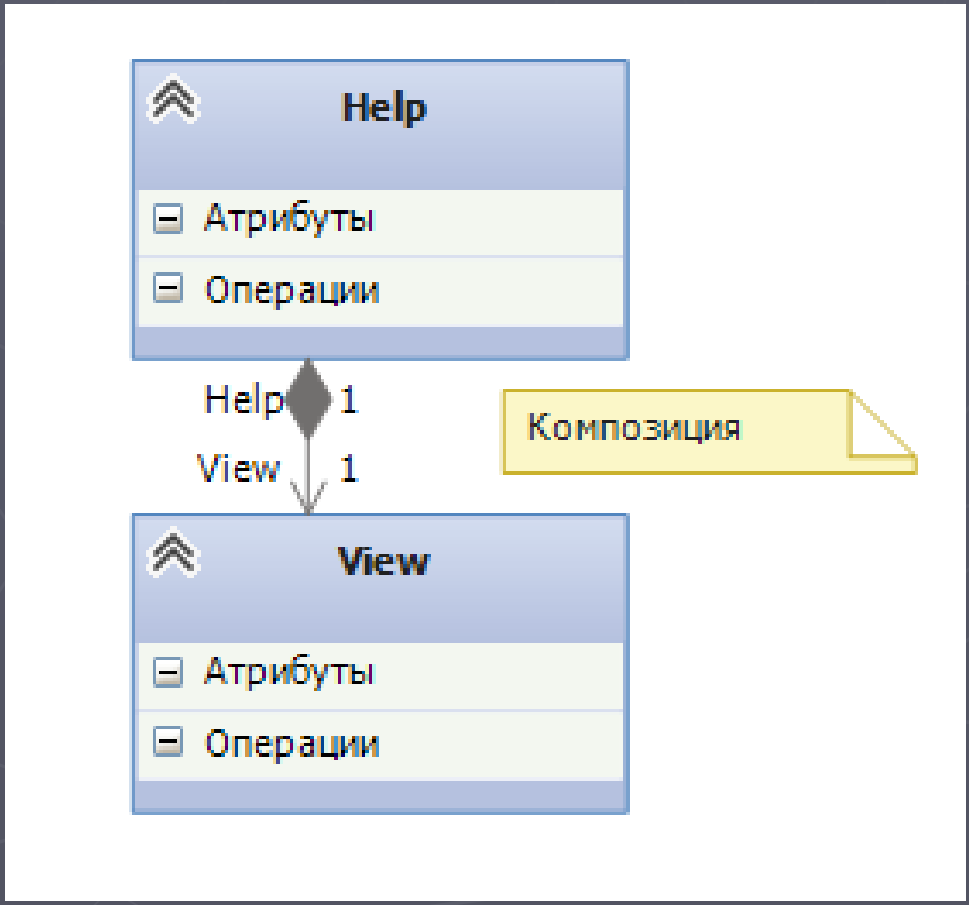
Группа – студент





Человек - очередь

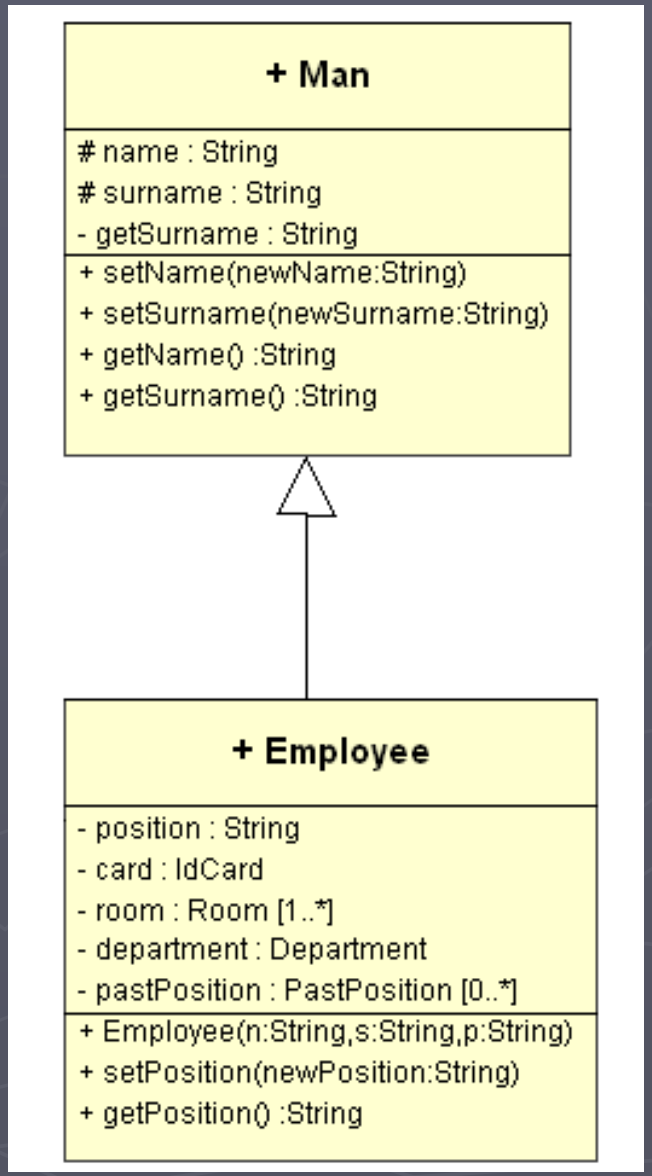
Помощь – окно просмотра



?

- Человек – работник

Человек – цех- завод



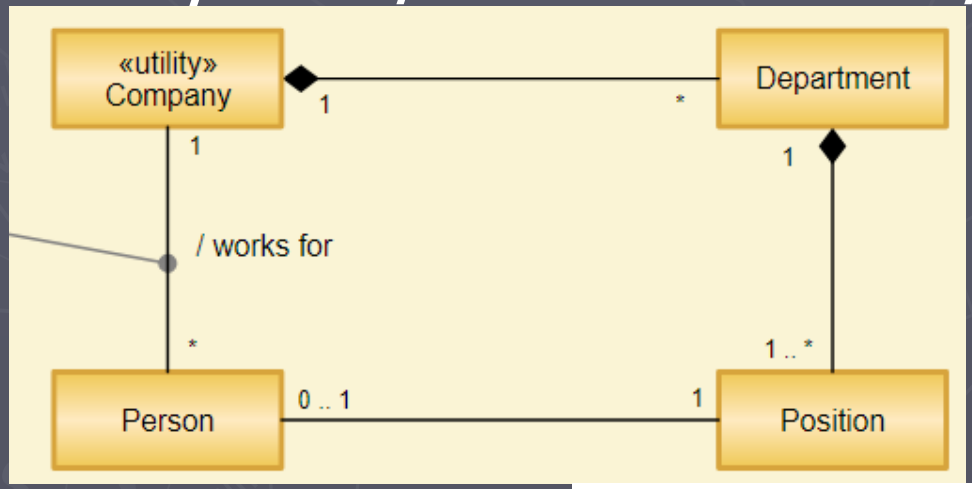
Отличие

- **композиция** может быть частью одного и только одного целого
- **агрегация** может быть частью нескольких объектов.

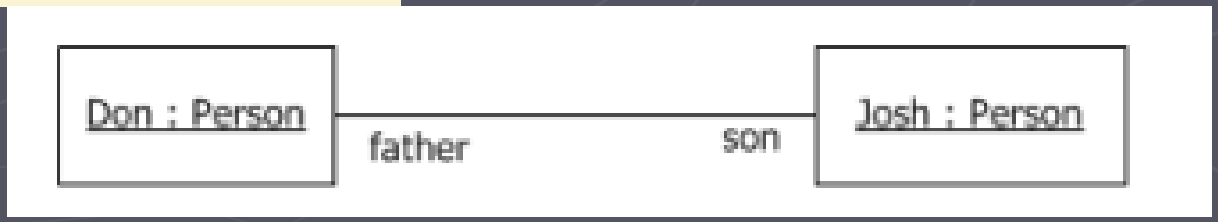
- **Г) Ассоциации и их дополнения**

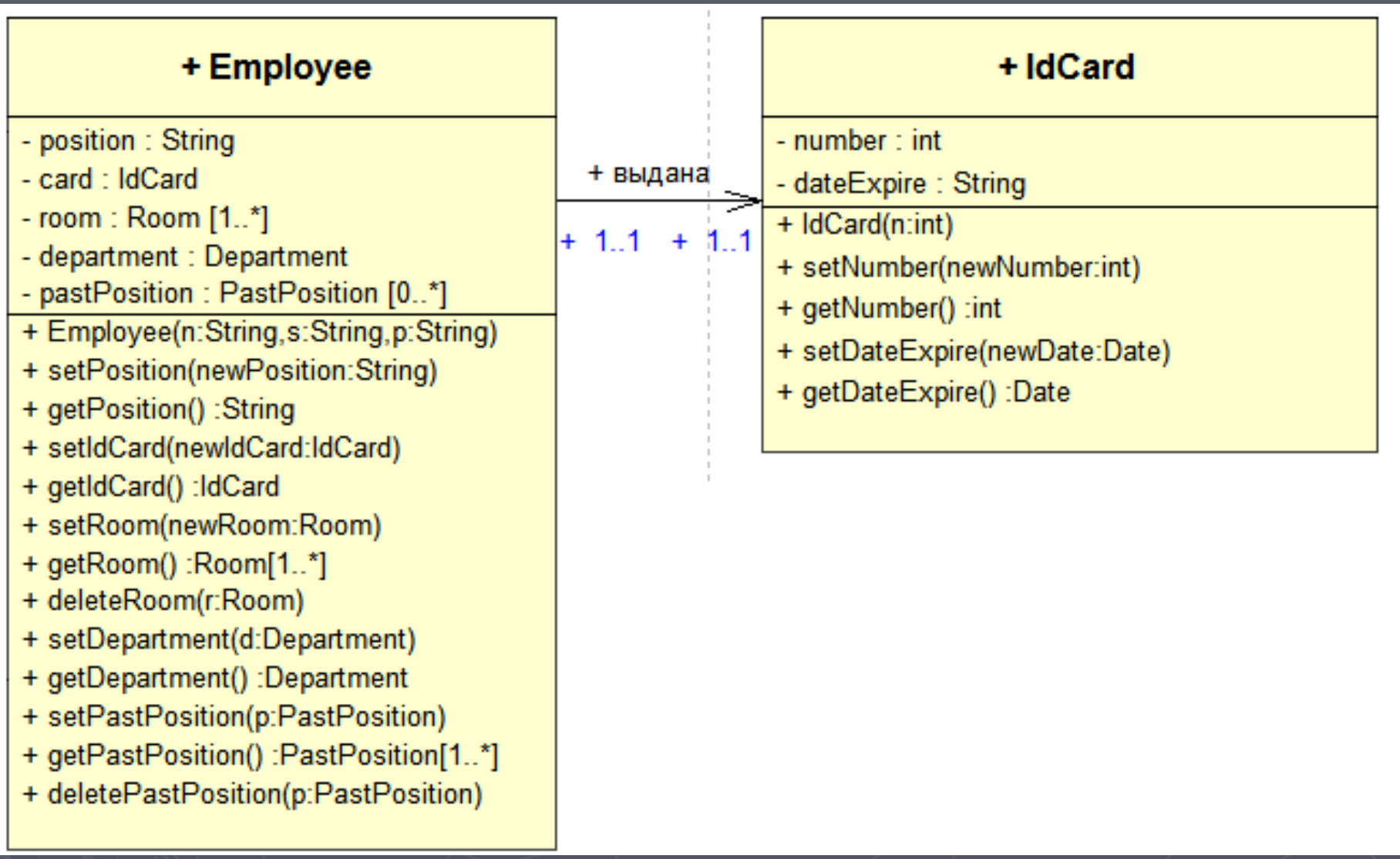
экземпляры одного класса связаны с экземплярами другого класса

- *Имя ассоциации*
- *Кратность полюса ассоциации*
- *Агрегация и композиция*



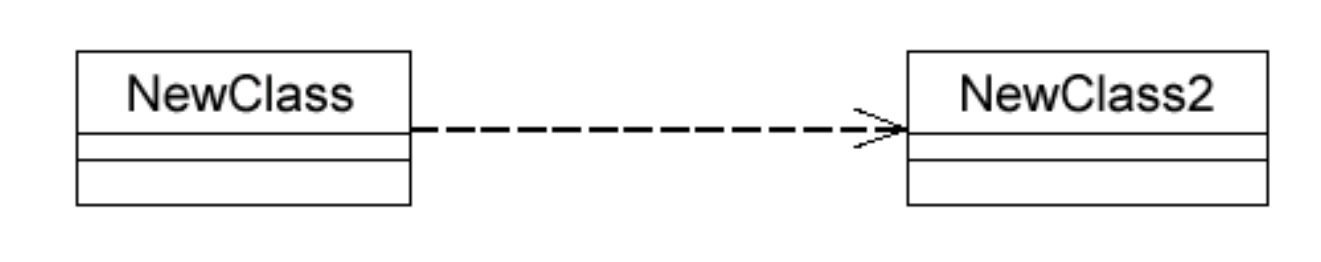
Ассоциация показывает отношения между объектами-экземплярами класса двунаправленные или однонаправленные



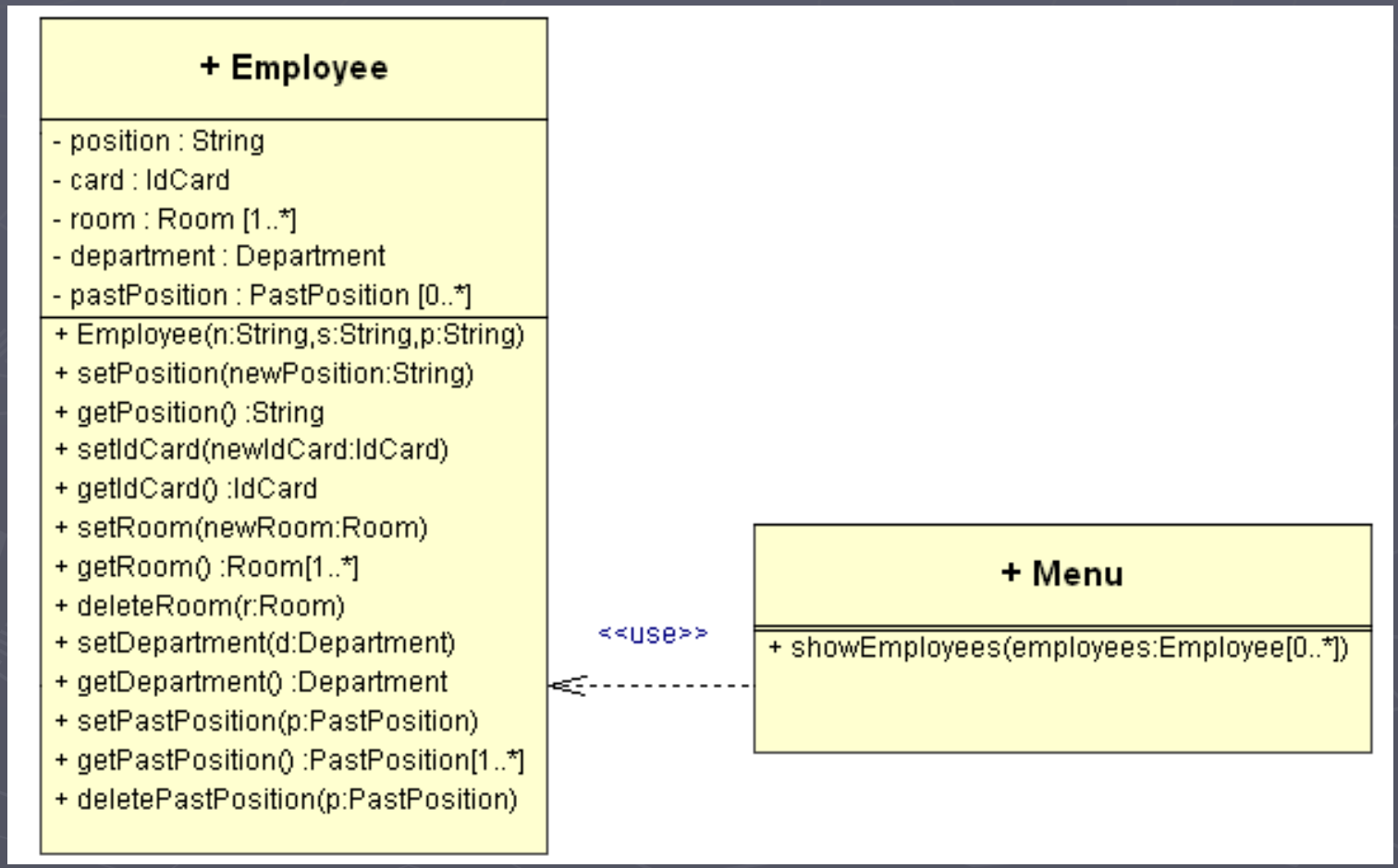

+ Employee
- position : String
- card : IdCard
- room : Room [1..*]
- department : Department
- pastPosition : PastPosition [0..*]
+ Employee(n:String,s:String,p:String)
+ setPosition(newPosition:String)
+ getPosition() :String
+ setIdCard(newIdCard:IdCard)
+ getIdCard() :IdCard
+ setRoom(newRoom:Room)
+ getRoom() :Room[1..*]
+ deleteRoom(r:Room)
+ setDepartment(d:Department)
+ getDepartment() :Department
+ setPastPosition(p:PastPosition)
+ getPastPosition() :PastPosition[1..*]
+ deletePastPosition(p:PastPosition)
+ выдана
+ 1..1
+ 1..1
+ IdCard
- number : int
- dateExpire : String
+ IdCard(n:int)
+ setNumber(newNumber:int)
+ getNumber() :int
+ setDateExpire(newDate:Date)
+ getDateExpire() :Date

д) Зависимость

- всегда однонаправленны

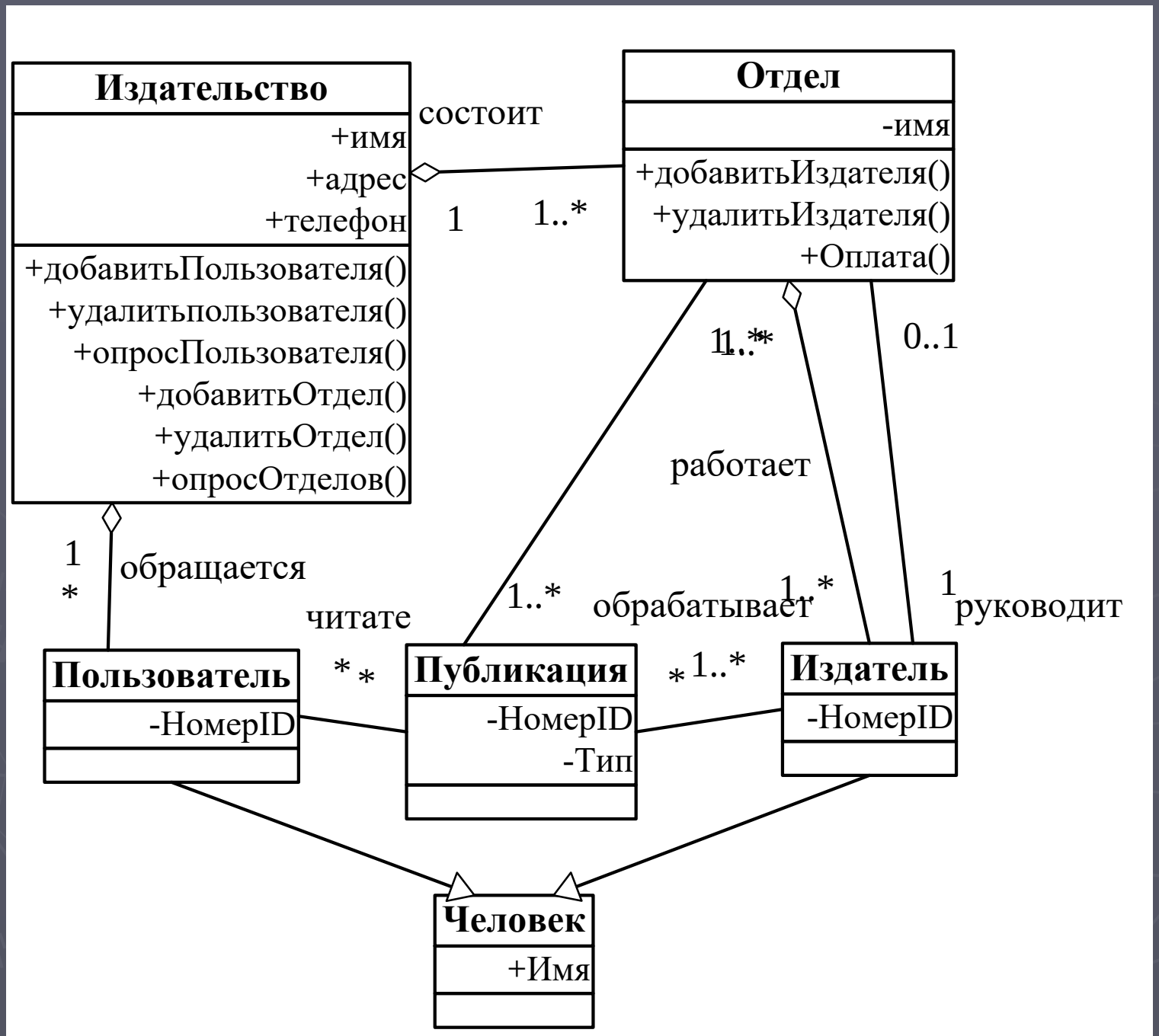


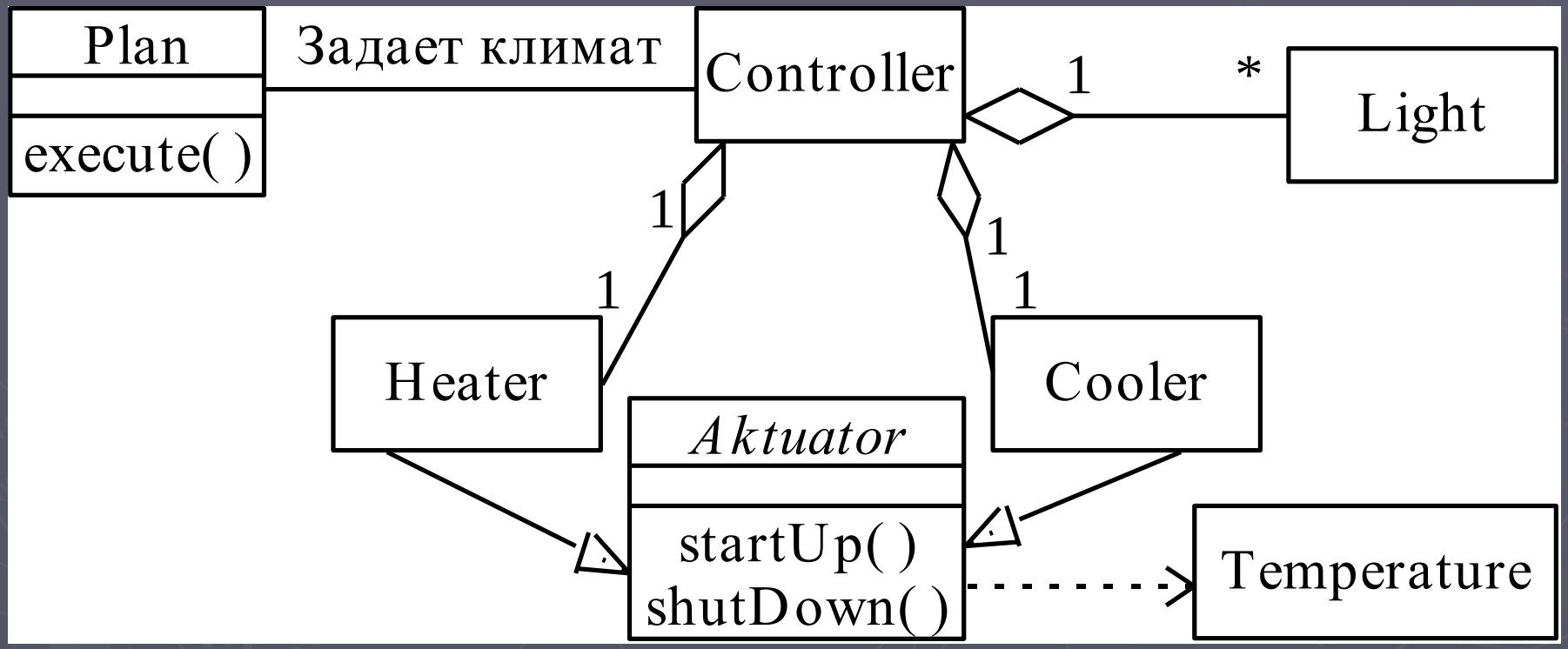
Работники - отображения



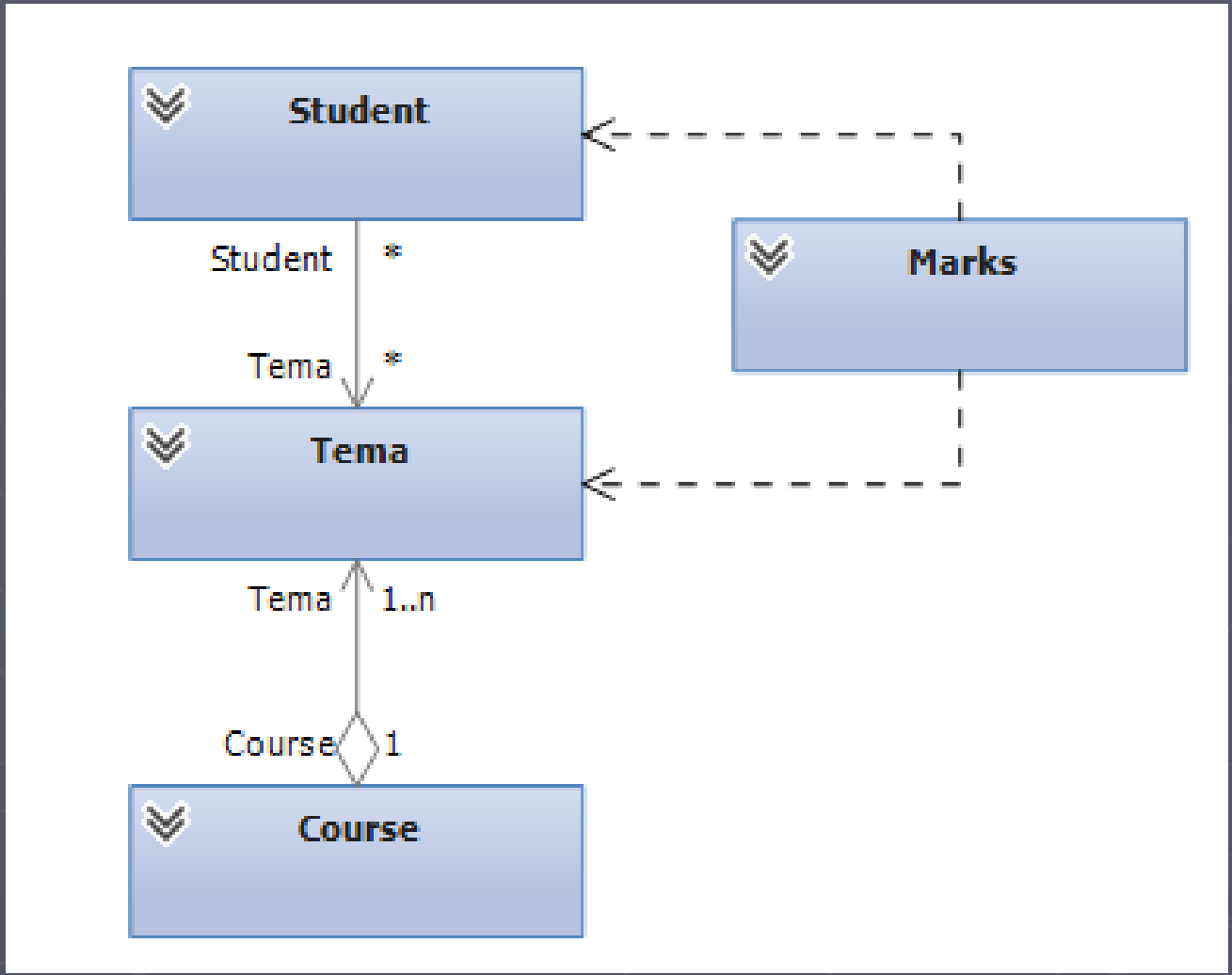
Типы отношений между классами

- обобщение\специализация (общее-частное)
- целое\часть
- семантическое, смысловые отношения или ассоциации


Издательство
+имя
+адрес
+телефон
+добавитьПользователя()
+удалитьпользователя()
+опросПользователя()
+добавитьОтдел()
+удалитьОтдел()
+опросОтделов()
состоит
1
1..*
Отдел
-имя
+добавитьИздателя()
+удалитьИздателя()
+Оплата()
1..*
0..1
работает
1
*
обращается
читате
1..*
обрабатывает
1..*
1
руководит
Пользователь
-НомерID
* *
Публикация
-НомерID
-Тип
* 1..*
Издатель
-НомерID
Человек
+Имя


Plan
execute( )
Задает климат
Controller
1
*
Light
1
1
1
1
Heater
Cooler
Aktuator
startUp( )
shutDown( )
Temperature

- Студент
- Тема
- Курс
- Оценки



Диаграммы классов (принципы)

- Описывать структуру удобнее параллельно с описанием поведения.
- Каждая итерация должна быть небольшим уточнением
- Не обязательно включать в модель все классы сразу
- Не обязательно определять все составляющие класса сразу.
- Не обязательно определять все отношения между классами сразу