Отношения между классами

Диаграмма классов UML

Пример. Отдел кадров предприятия

Назначение диаграммы классов

- Диаграмма классов UML позволяет обозначать отношения между классами и их экземплярами
- Для чего они нужны?
 - Они нужны для реализации программной модели прикладной области

Обобщение Ассоциация Зависимость Реализация

Создание класса

вне класса разрешен)

+ Man Открытый • Класс «Мап» (человек) (публичный) # name : String • Программная модель роли -# surname : String getSurname: String человек Защищенный (доступ + setName(newName:String) только для классов-+ setSurname(newSurname:String) наследников) + getName() :String + getSurname():String Свойства (атрибуты) Закрытый (доступ из вне класса запрещен) Поведение (методы) Открытый (доступ из

Обобщение (наследование)

+ Man

name : String

surname: String

- getSurname : String
- + setName(newName:String)
- + setSurname(newSurname:String)
- + getName():String
- + getSurname() :String

- Класс «Мап» (человек) более общий
- Класс «Employee» (сотрудник) более специализированный

Класс «Employee» наследует свойства и методы класса «Man»

Класс «Employee» добавляет свои свойства и методы

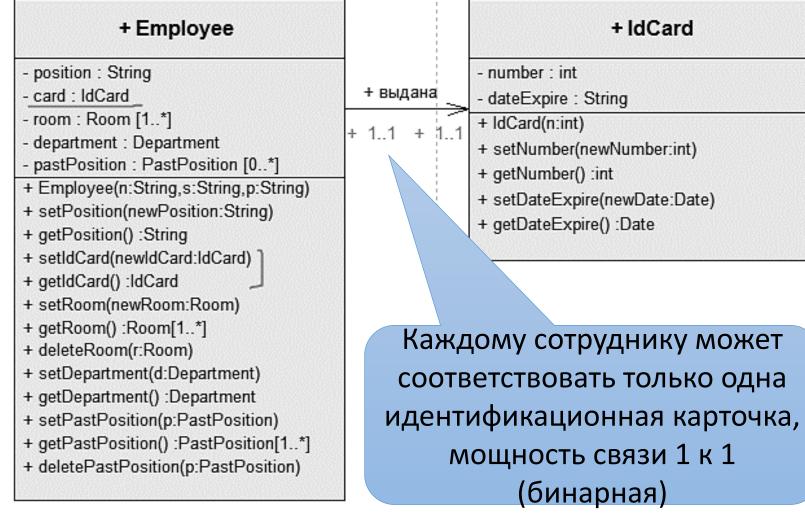
Может также изменить (переопределить) наследуемые методы

+ 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

Ассоциация (бинарная)

• Ассоциация показывает отношения между объектамиэкземплярами классов



Класс «IdCard» представляет идентификационную карточку(пропуск) сотрудника

Ассоциация (N-арная)

+ 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)1
- + getRoom() :Room[1..*]
- + deleteRoom(r:Room)



+ Room

- number: int
- + Room(n:int)
- + setNumber(newNumner:int)
- + getNumber() :int

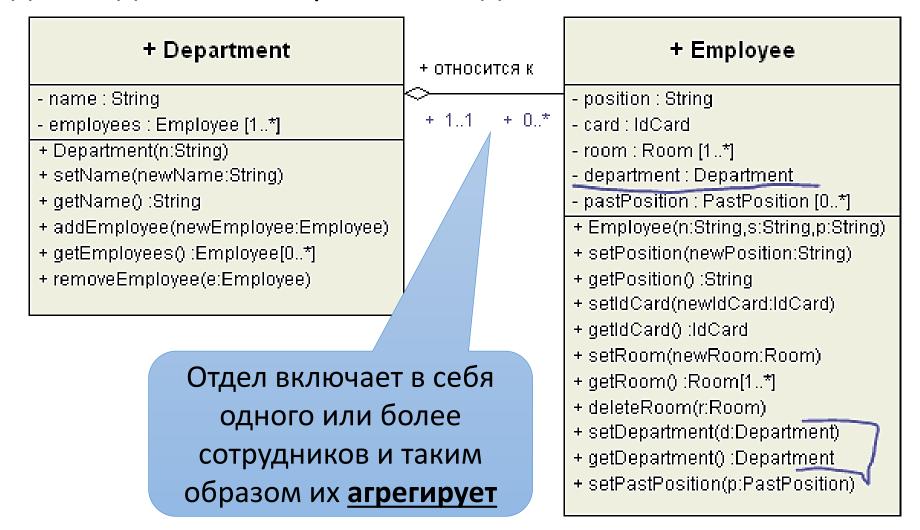
- В организации положено закреплять за работниками помещения.
- Добавляем новый класс Room

Каждому объекты работник (Employee) может соответствовать несколько рабочих помещений. Мощность связи один-ко-многим. Навигация от Employee к Room

Агрегация

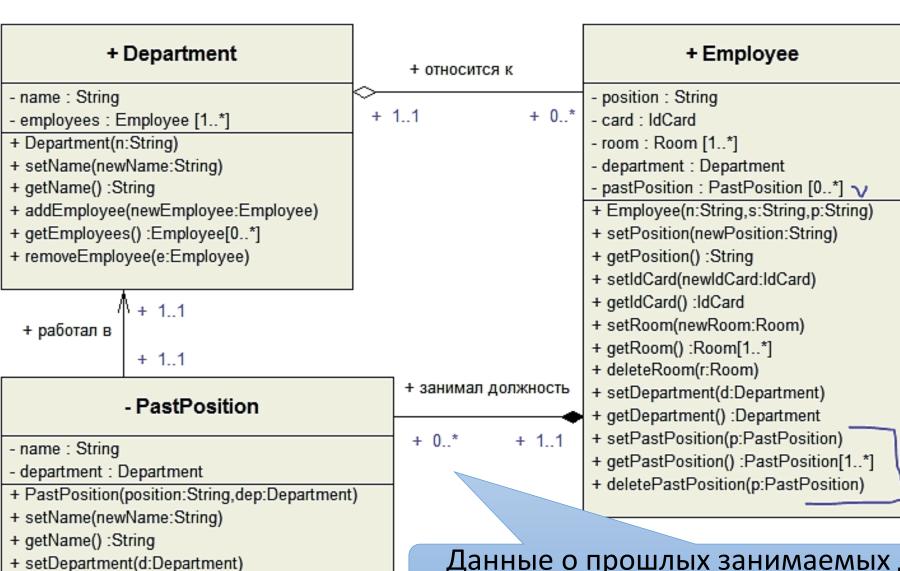
• Класс Department(отдел) — предприятие структурировано по отделам. В каждом отделе может работать один или более

человек



Композиция

+ getDepartment() :Department



- Требование: хранить данные о прежней занимаемой должности на предприятии.
- Введем новый класс «pastPosition»

Данные о прошлых занимаемых должностях являются частью данных о сотруднике

Зависимость

+ Employee

- position : String

card : IdCard

- room : Room [1..*]

- department : Department

- pastPosition : PastPosition [0..*]

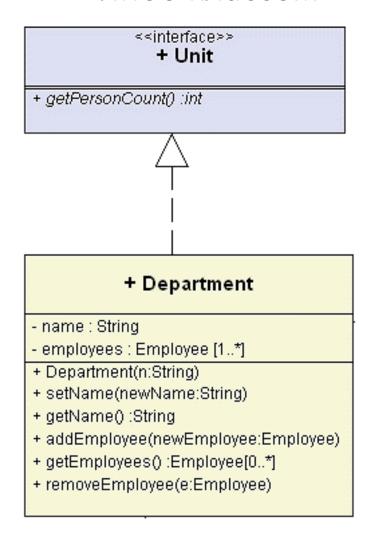
- + Employee(n:String,s:String,p:String)
- + setPosition(newPosition:String)
- + getPosition():String
- + setIdCard(newIdCard:IdCard)
- + getIdCard():IdCard
- + setRoom(newRoom:Room)
- + getRoom() :Room[1..*]
- + deleteRoom(r:Room)
- + setDepartment(d:Department)
- + getDepartment() :Department
- + setPastPosition(p:PastPosition)
- + getPastPosition() :PastPosition[1..*]
- + deletePastPosition(p:PastPosition)

- Для организации диалога с пользователем введем в систему класс «Menu»
 - метод «showEmployees» показывает список сотрудников и их должности
 - Параметром для метода является массив объектов «Employee»
- Таким образом, изменения внесенные в класс «Employee» могут потребовать и изменения класса «Menu»



Реализация

• Объявление интерфейса и возможность его реализации каким-либо классом



Интерфейс «Unit» представляет собой самую абстрактную единицу деления организации

В каждой единице деления работает какое-то количество сотрудников, поэтому метод для получения количества работающих людей будет актуален для каждого класса, реализующего интерфейс «Unit»

Итоговая диаграмма

