



Design patterns



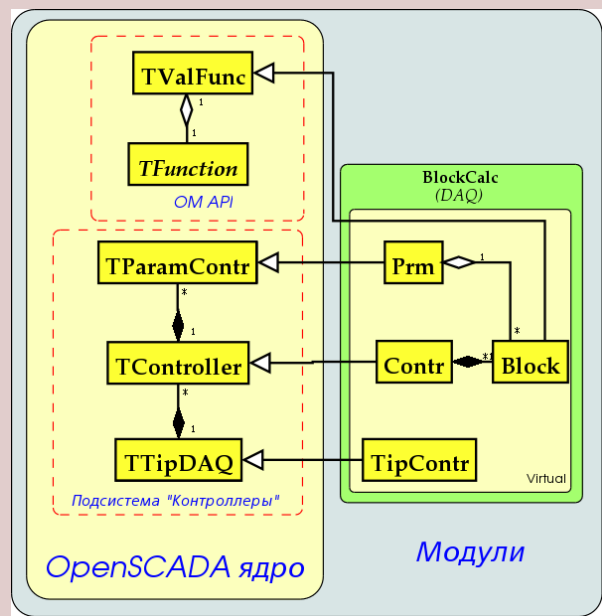
Типы паттернов проектирования

- **Порождающие паттерны** (**behavioral**), предназначенные для создания новых объектов в системе.
- **Структурные паттерны** (**creational**), решающие задачи компоновки системы на основе классов и объектов.
- **Паттерны поведения** (**structural**), предназначенные для распределения обязанностей между объектами в системе.

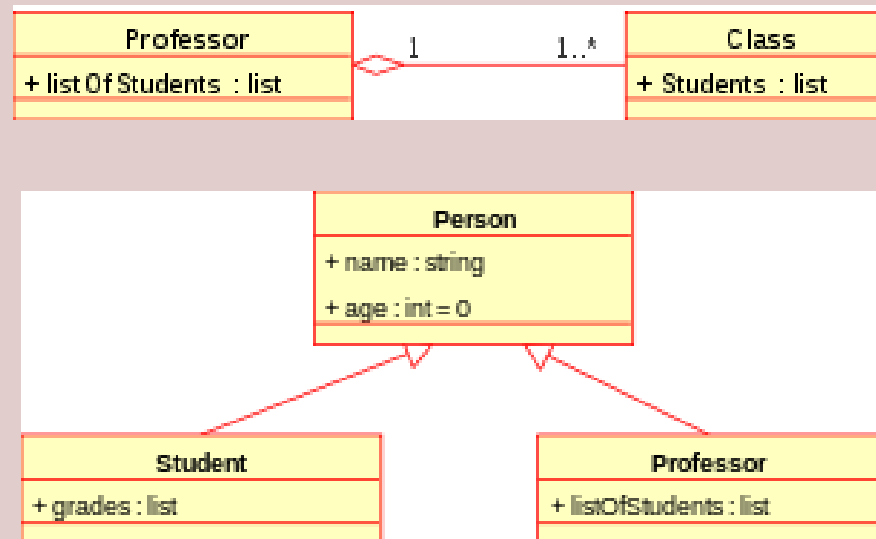
существует и масса других паттернов: паттерны кодирования, тестирования, модификации кода (а.к.а. рефакторинг), существуют архитектурные паттерны и многие другие

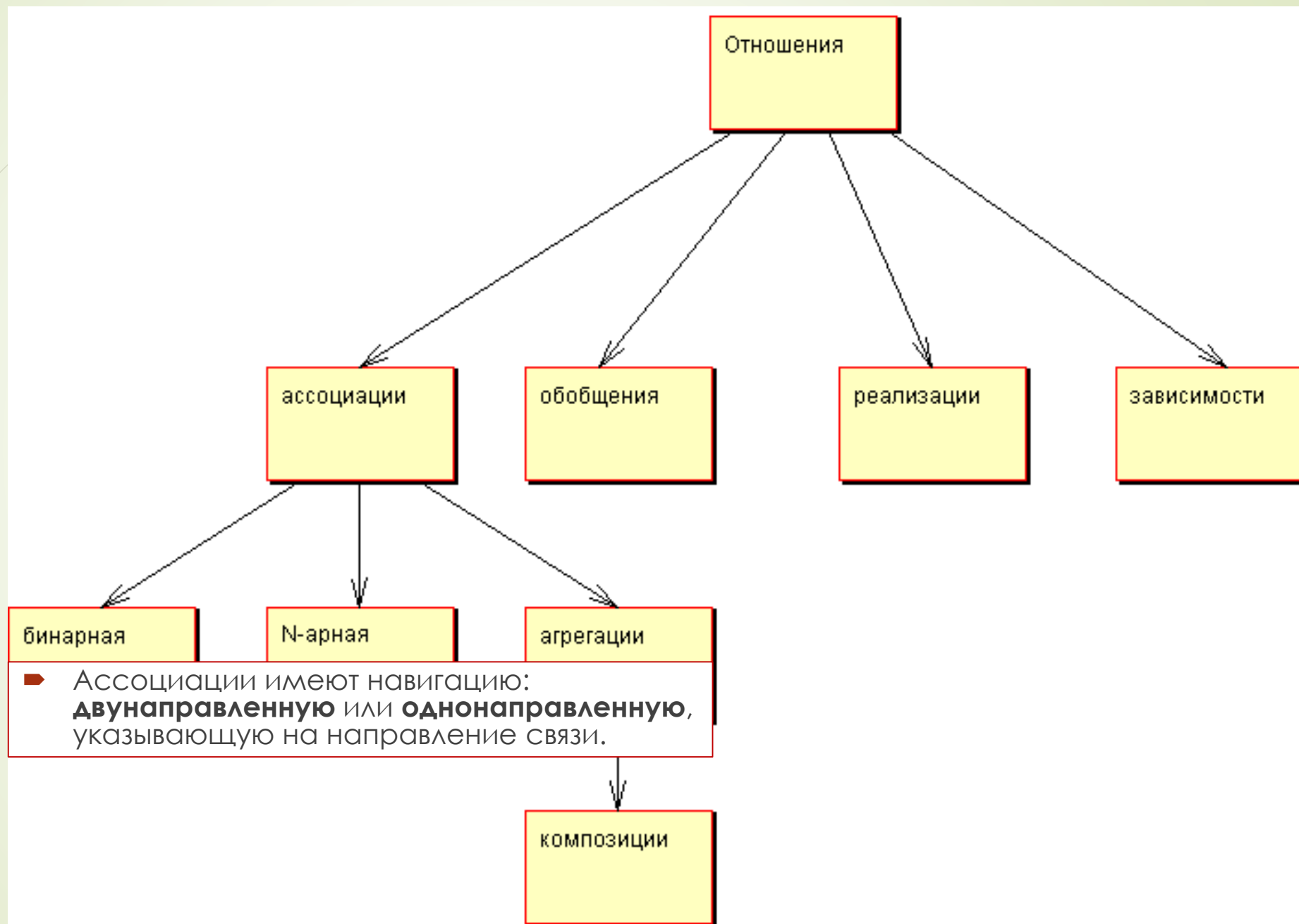
Диаграмма классов

Статический



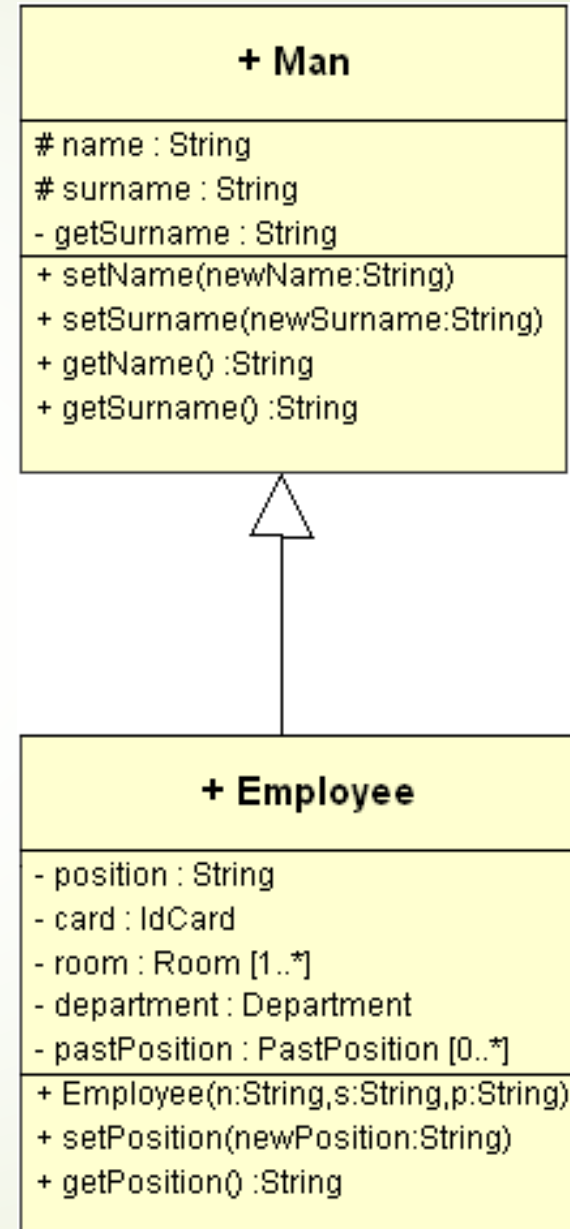
Аналитический





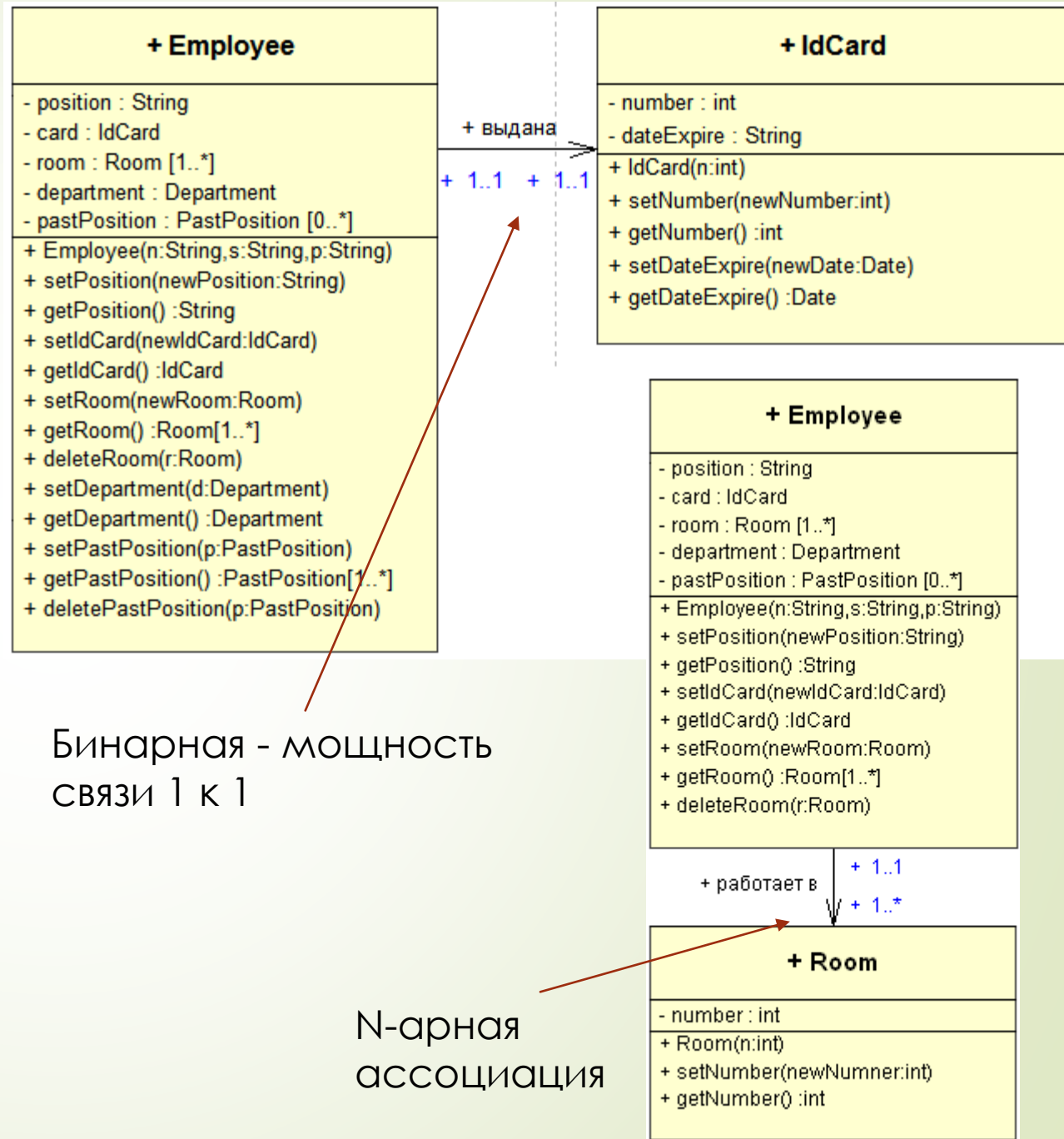
Обобщение

- Отношение обобщения — это наследование.



Ассоциация

- Ассоциация показывает отношения между объектами-экземплярами класса.

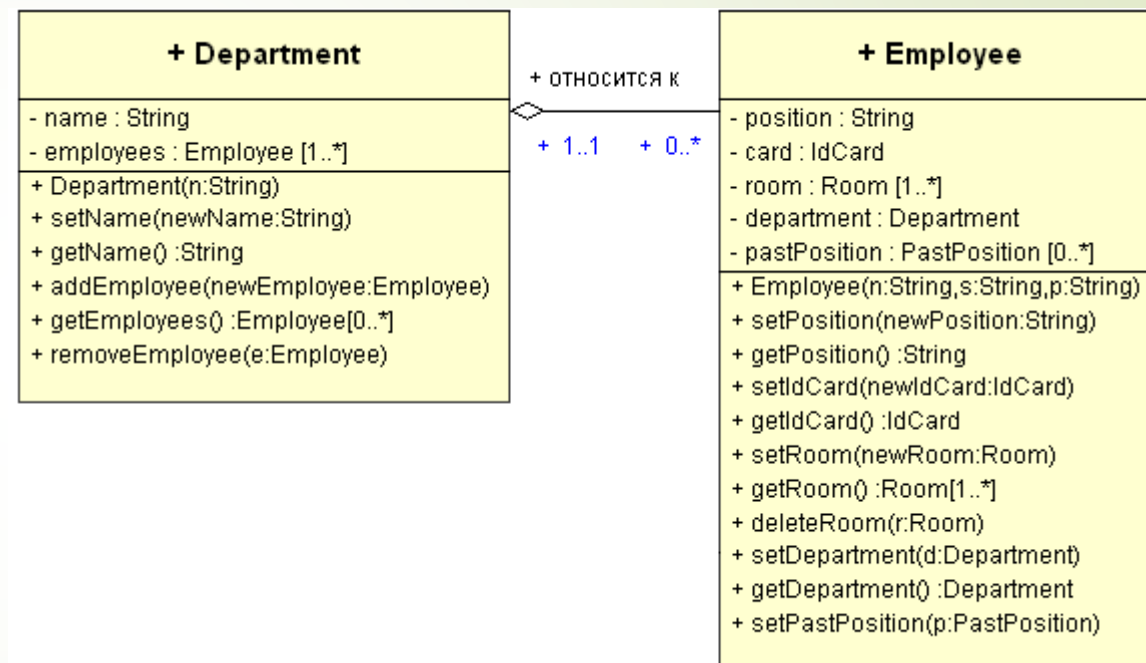


Бинарная - мощность
связи 1 к 1

N-арная
ассоциация

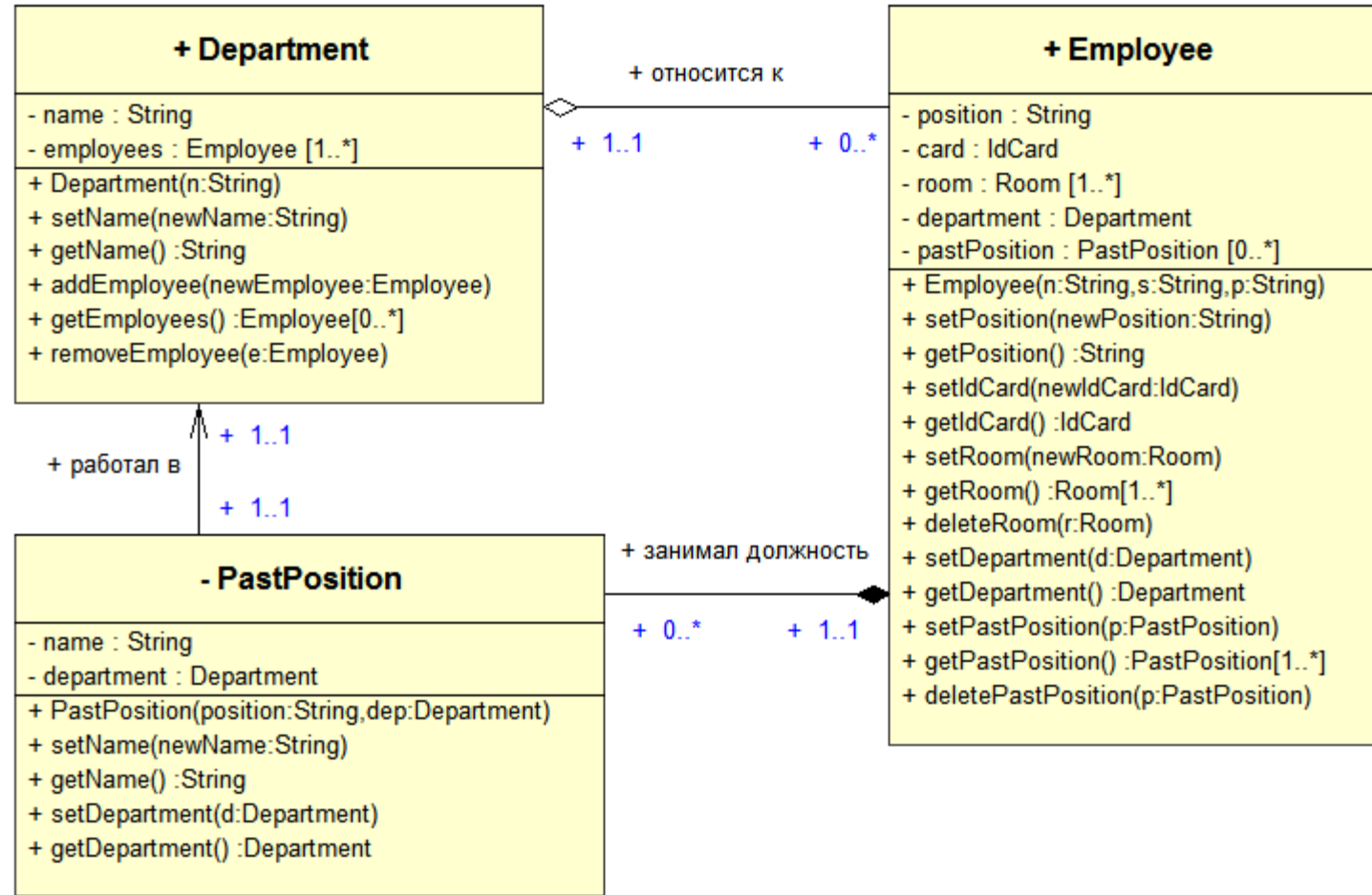
Агрегация

- Структурное отношение между равноправными сущностями
- Навигация на диаграмме не показана, значит она является двунаправленной



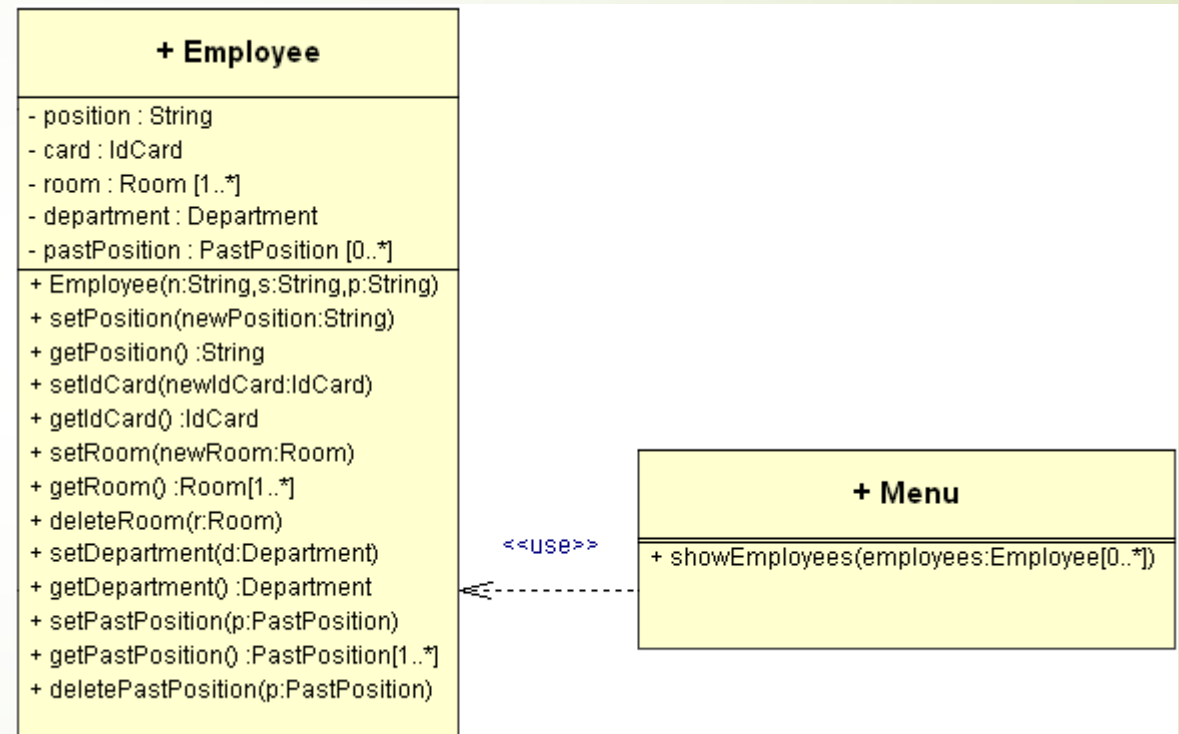
Композиция

- Имеет жёсткую зависимость времени существования экземпляров класса контейнера и экземпляров содержащихся классов.



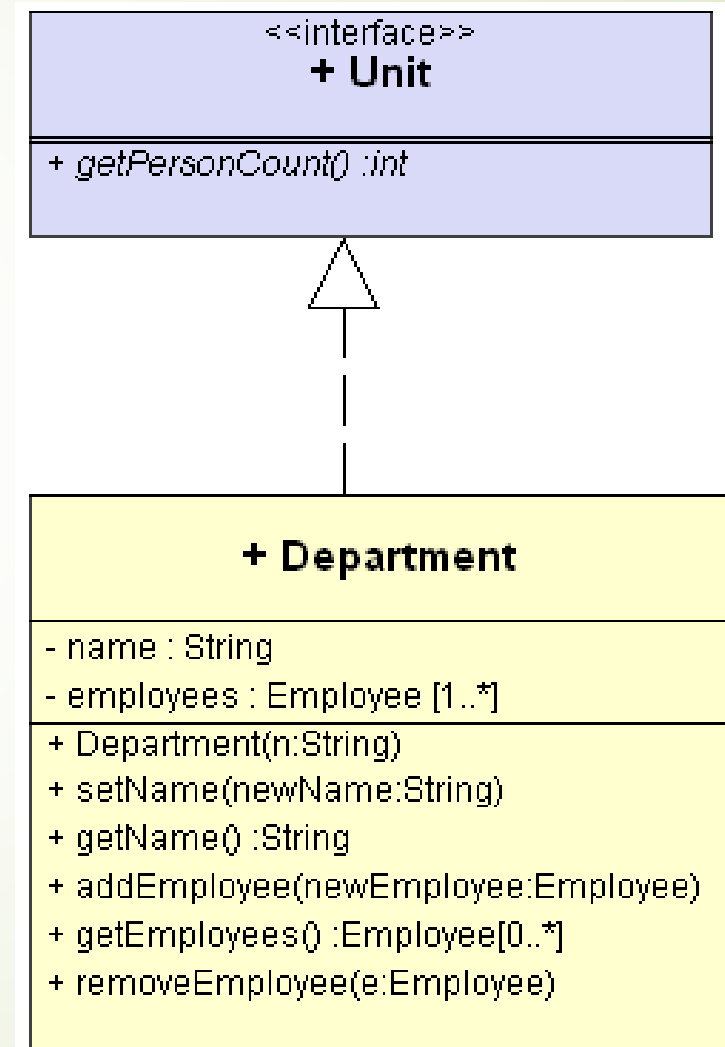
Зависимость

- Слабая форма отношения использования, при котором изменение одного влечёт за собой изменение другого, причём обратное не обязательно.



Реализация

- отношение между двумя элементами модели, в котором один элемент (клиент) реализует поведение, заданное другим (поставщиком).



Порождающие паттерны

- для создания объектов, позволяя системе оставаться независимой как от самого процесса порождения, так и от типов порождаемых объектов.



2 способа

Красивый

```
Warrior * createWarrior( Warrior_ID id )
{
    Warrior * p;
    switch (id)
    {
        case Archer_ID:
            p = new Archer();
            break;
        case Horseman_ID:
            p = new Horseman();
            break;
        ...
    }
    return p;
}
```

Не очень

```
class Warrior
{
public:
    virtual function() = 0;
    virtual ~Warrior() {}
};

class Infantryman: public Warrior{
    ...
};

class Archer: public Warrior{
    ...
};

class Horseman: public Warrior{
    ...
};
```



ОСНОВНЫЕ

- Паттерн Singleton контролирует создание единственного экземпляра некоторого класса и предоставляет доступ к нему.
- Паттерн Factory Method - вводится полиморфный класс Factory, в котором определяется интерфейс фабричного метода, а ответственность за создание объектов конкретных классов переносится на производные от Factory классы, в которых этот метод переопределяется.
- Паттерн Abstract Factory использует несколько фабричных методов и предназначен для создания целого семейства или группы взаимосвязанных объектов.
- Паттерн Prototype создает новые объекты с помощью прототипов. Прототип - некоторый объект, умеющий создавать по запросу копию самого себя.
- Паттерн Object Pool используется в случае, когда создание объекта требует больших затрат или может быть создано только ограниченное количество объектов некоторого класса.



Структурные паттерны

- решающие задачи как из классов и объектов образуются более крупные структуры
- Структурные шаблоны уровня класса используют наследование для составления композиций из интерфейсов и реализаций.

Наследование

- базовый класс определяет интерфейс, а подклассы - реализацию. Структуры на основе наследования получаются статичными.

Композиция

- структуры строятся путем объединения объектов некоторых классов. Композиция позволяет получать структуры, которые можно изменять во время выполнения.



ОСНОВНЫЕ

- Паттерн Adapter представляет собой программную обертку над уже существующими классами и предназначен для преобразования их интерфейсов к виду, пригодному для последующего использования в новом программном проекте.
- Паттерн Bridge отделяет абстракцию от реализации так, что то и другое можно изменять независимо.
- Паттерн Composite группирует схожие объекты в древовидные структуры. Рассматривает единообразно простые и сложные объекты.
- Паттерн Decorator используется для расширения функциональности объектов. Являясь гибкой альтернативой порождению классов, паттерн Decorator динамически добавляет объекту новые обязанности.



Паттерны поведения

- предназначенные для распределения обязанностей между объектами в системе; определяющие алгоритмы и способы реализации взаимодействия различных объектов и классов.
- Здесь важно то, как объекты получают информацию о существовании друг друга. Объекты-коллеги могут хранить ссылки друг на друга, но это усиливает степень связанности системы. При высокой связанности каждому объекту пришлось бы иметь информацию обо всех остальных. Некоторые из шаблонов решают эту проблему.

ОСНОВНЫЕ

- Паттерн Chain of Responsibility позволяет обработать запрос несколькими объектам-получателям. Получатели связываются в цепочку, и запрос передается по цепочке, пока не будет обработан каким-то объектом. Паттерн Chain of Responsibility позволяет также избежать жесткой зависимости между отправителем запроса и его получателями.
- Паттерн Command преобразовывает запрос на выполнение действия в отдельный объект-команду. Это придает системе гибкость: позволяет осуществлять динамическую замену команд, использовать сложные составные команды, осуществлять отмену операций.
- Паттерн Iterator предоставляет механизм обхода элементов составных объектов (коллекций) не раскрывая их внутреннего представления.
- Паттерн Interpreter предназначен для решения повторяющихся задач, которые можно описать некоторым языком. Для этого паттерн Interpreter описывает решаемую задачу в виде предложений этого языка, а затем интерпретирует их.
- Паттерн Mediator инкапсулирует взаимодействие совокупности объектов в отдельный объект-посредник. Уменьшает степень связанности взаимодействующих объектов - им не нужно хранить ссылки друг на друга.

ОСНОВНЫЕ

- Паттерн Memento получает и сохраняет за пределами объекта его внутреннее состояние так, чтобы позже можно было восстановить объект в таком же состоянии.
- Паттерн Observer определяет зависимость "один-ко-многим" между объектами так, что при изменении состояния одного объекта все зависящие от него объекты уведомляются и обновляются автоматически.
- Паттерн State позволяет объекту изменять свое поведение в зависимости от внутреннего состояния. Создается впечатление, что объект изменил свой класс. Паттерн State является объектно-ориентированной реализацией конечного автомата.
- Если поведение системы настраивается согласно одному из некоторого множества алгоритму, то применение паттерна Strategy переносит семейство алгоритмов в отдельную иерархию классов, что позволяет заменять один алгоритм другим в ходе выполнения программы. Кроме того, такую систему проще расширять и поддерживать.
- Паттерн Template Method определяет основу алгоритма и позволяет подклассам изменить некоторые шаги этого алгоритма без изменения его общей структуры.
- Паттерн Visitor определяет операцию, выполняемую на каждом элементе из некоторой структуры без изменения классов этих объектов.