

## Лекция 6

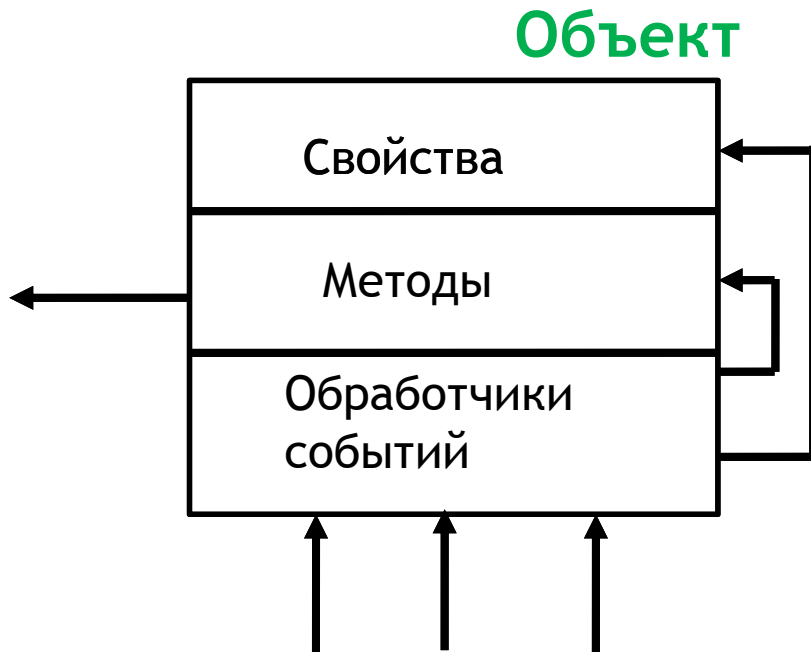
# Объектно-ориентированное программирование. Часть 2

# События и сообщения

Объекты могут:

- Взаимодействовать друг с другом посредством **сообщений**.
- Реагировать на определенные **события** (обрабатывать их), возникающие вследствие действий пользователя или других объектов.

*Хорошо подходит для графического интерфейса (обработчики событий нажатия на кнопку и т.п.)*



Тип переменной - множество отношений между данным типом и другими типами.

# Интерфейсы

В интерфейсе декларируется функциональность, которую должен реализовать каждый класс, реализующий этот интерфейс.



**Что здесь?**



**Что здесь?**





**Что здесь?**

Любая реализация интерфейса  
«Вилка для электрической розетки»

# Интерфейсы

*Примеры на PHP*

# Интерфейсы

Аналог абстрактного класса, содержащего только абстрактные методы. Интерфейсы — это облегченные классы, которые диктуют нам условия.

Интерфейсы можно наследовать друг от друга.

Класс может реализовывать несколько интерфейсов (наследоваться от нескольких классов нельзя).

Класс = тип данных + шаблон для создания экземпляров  
Интерфейс = тип данных

## Константы класса Статические свойства и методы

Определяются в контексте класса, а не объекта.

- Доступны из любой точки программы.
- Доступны каждому экземпляру данного класса.
- Не нужно создавать экземпляры класса только ради вызова простой функции.

*Примеры реализации на PHP*

# Способы взаимодействия между классами

- **Наследование**

- *Кошка является животным.*

- **Ассоциация**

В свойстве класса содержится ссылка на экземпляр(ы) другого класса

- ◆ **Композиция**

- *Квартира имеет комнату.*

- ◆ **Агрегация**

- *Автомобиль имеет колесо.*



Отношение  
**агрегации**



Отношение  
**КОМПОЗИЦИИ**

# *Примеры кода*

# КОМПОЗИЦИЯ

Объект А управляет временем жизни объекта В

```
class Engine
{
    private $power;

    public function __construct($power)
    {
        $this->power = $power;
    }
}

class Car
{
    private $model;
    private $engine;

    public function __construct($model, $power)
    {
        $this->model = $model;
        $this->engine = new Engine($power);
    }
}

$myCar = new Car( model: "Saturn VUE", power: 160);
```

## Плюсы

- Скрывает отношения использования объекта от глаз клиента.
- Делает API использования класса более простым.

## Минусы

- Отношение жесткое (один объект должен знать конкретный тип и иметь доступ к функции создания другого объекта).

# Агрегация

Объект А получает ссылку на объект В

```
class Engine
{
    private $power;

    public function __construct($power)
    {
        $this->power = $power;
    }
}

class Car
{
    private $model;
    private $engine;

    public function __construct($model, $engine)
    {
        $this->model = $model;
        $this->engine = $engine;
    }
}

$myCar = new Car( model: "Saturn VUE", new Engine( power: 160));
```

## Плюсы

- Более слабая связанность между объектом и его клиентом. Объекту не нужно знать, как именно создавать другой объект.
- Большая гибкость.

## Минусы

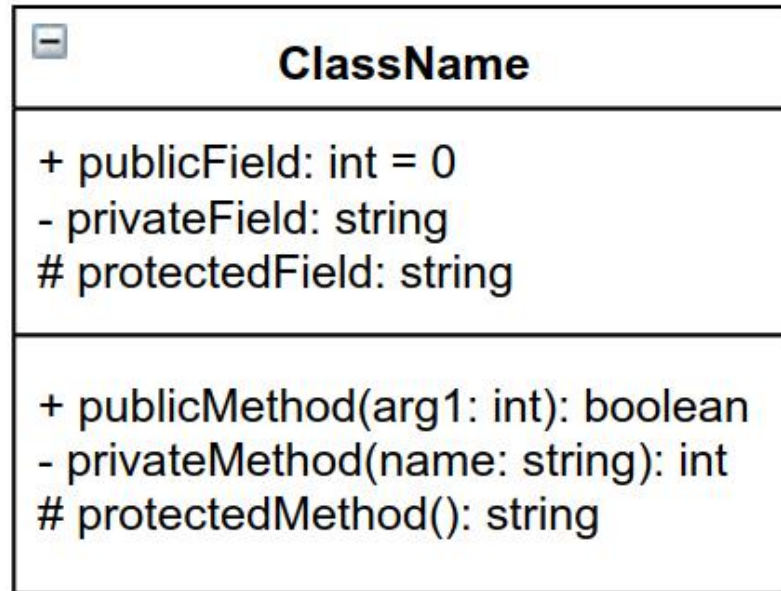
- Выставление наружу деталей реализации, увеличение сложности в работе клиентов.
- «Целое» не может самостоятельно заменить «составную часть».

# UML-диаграммы

**U**nified **M**odeling **L**anguage — графический синтаксис для описания объектно-ориентированных систем

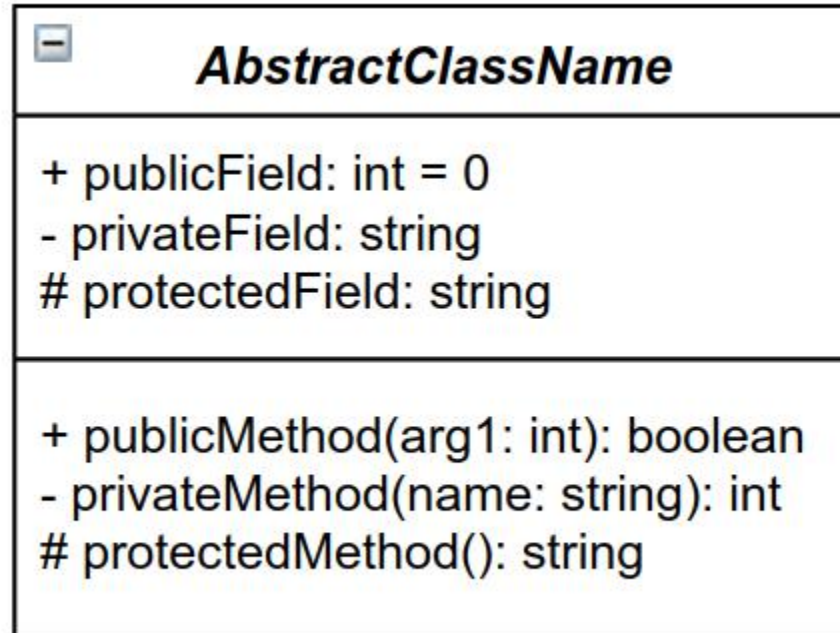
- 1) Диаграмма классов (class diagram)
- 2) Диаграмма использования (use case diagram)
- 3) Диаграмма автомата (stat machine diagram)
- 4) Диаграмма деятельности (activity diagram)
- 5) Диаграмма последовательности (sequence diagram)
- 6) Диаграмма коммуникации (communicatio diagram)

# Диаграммы классов



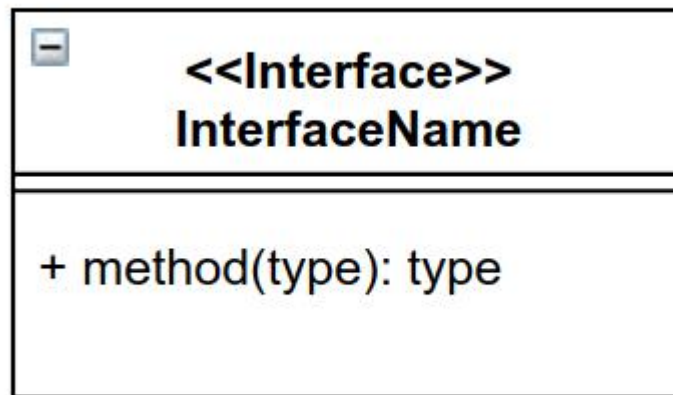
**Класс с атрибутами и операциями**

# Диаграммы классов



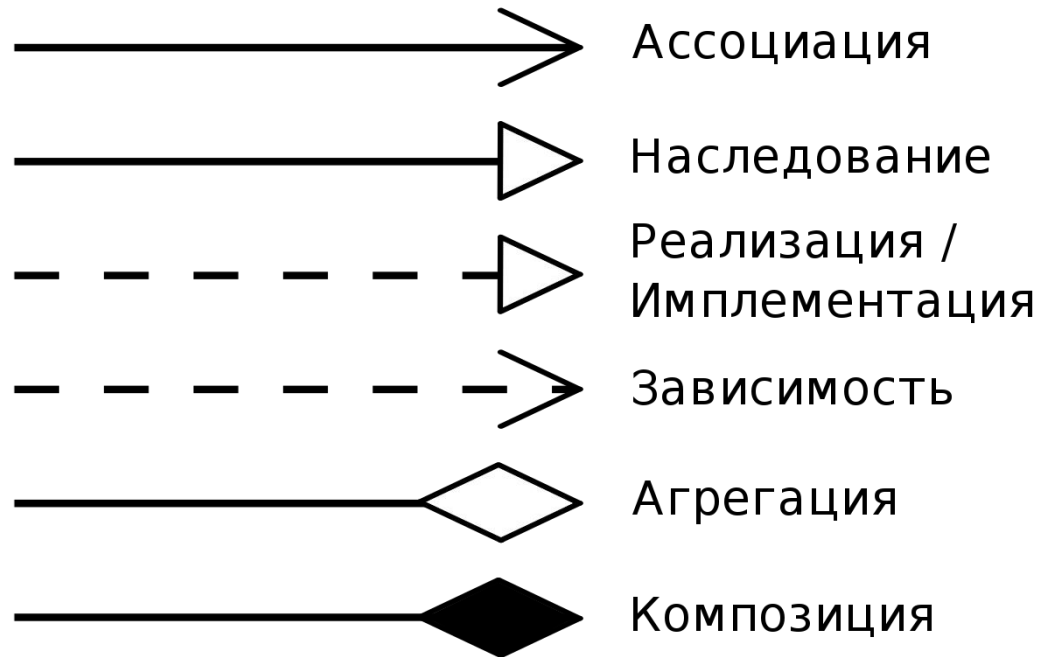
**Абстрактный класс**

# Диаграммы классов



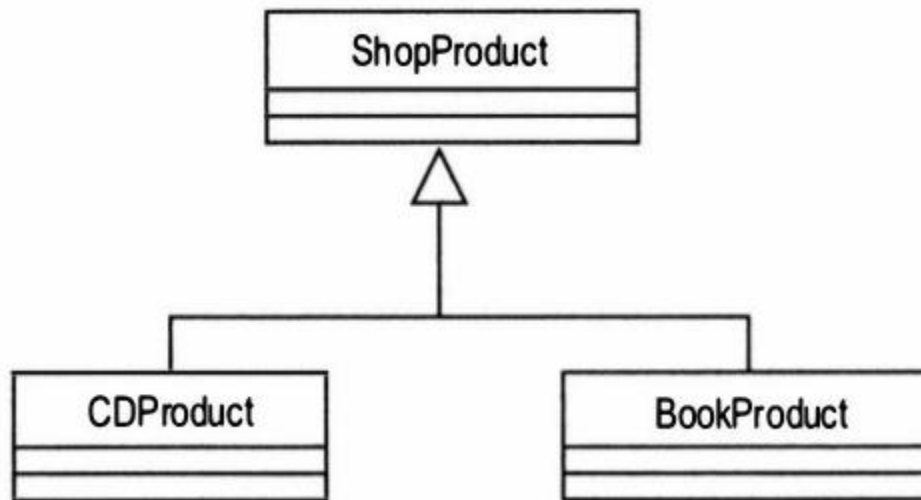
**Интерфейс**

# Диаграммы классов



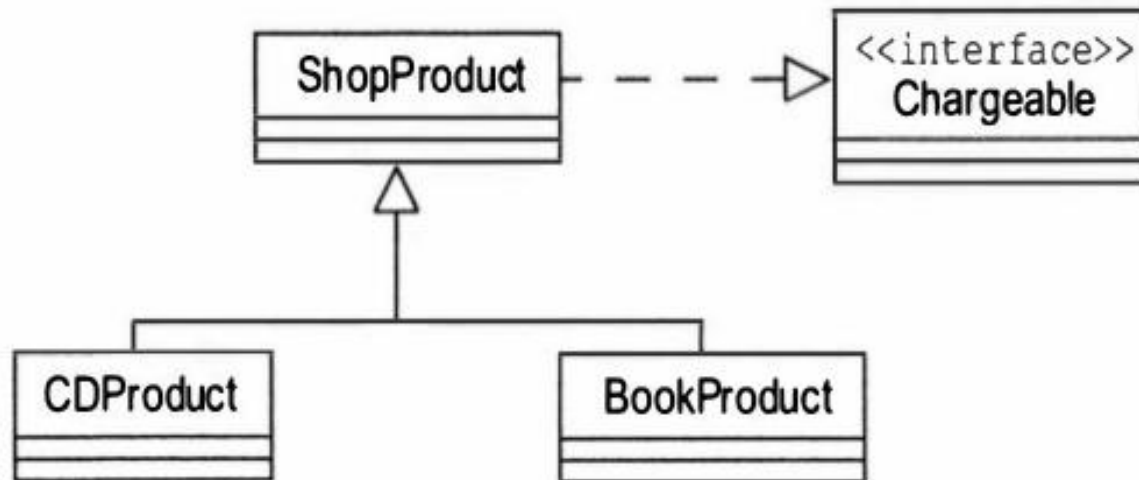
## Отношения между классами

# Диаграммы классов



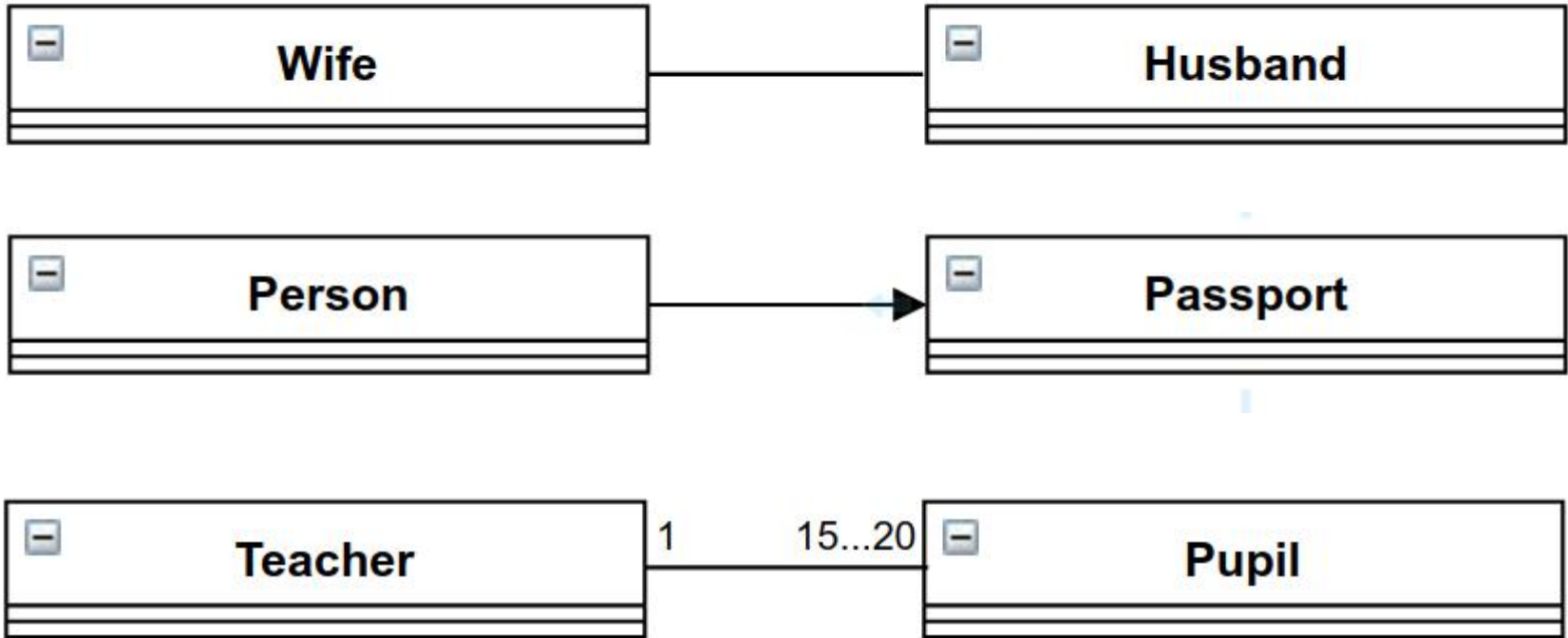
**Отношение наследования (обобщение)**

# Диаграммы классов



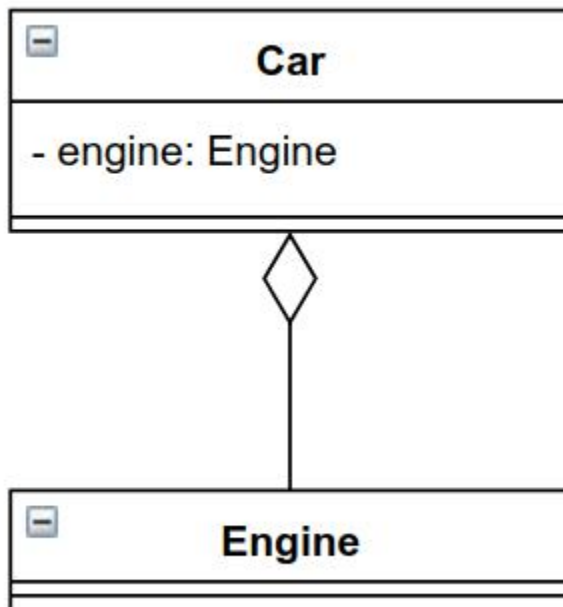
**Отношение реализации**

# Диаграммы классов



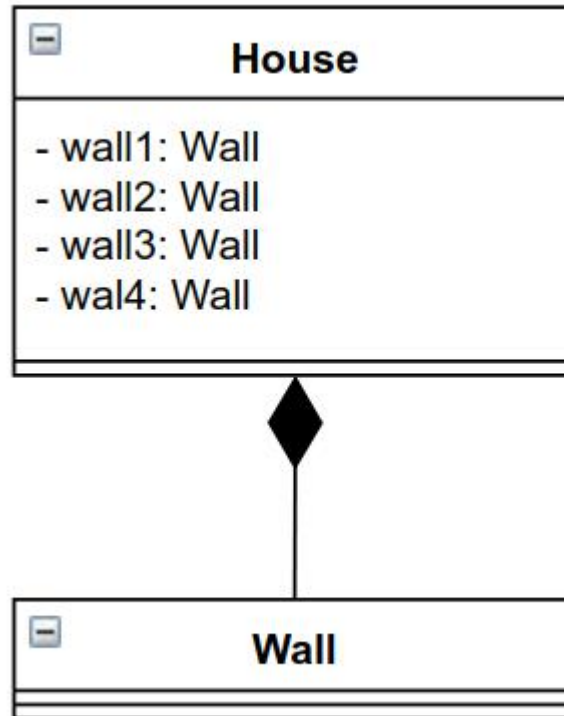
## Отношение ассоциации

# Диаграммы классов



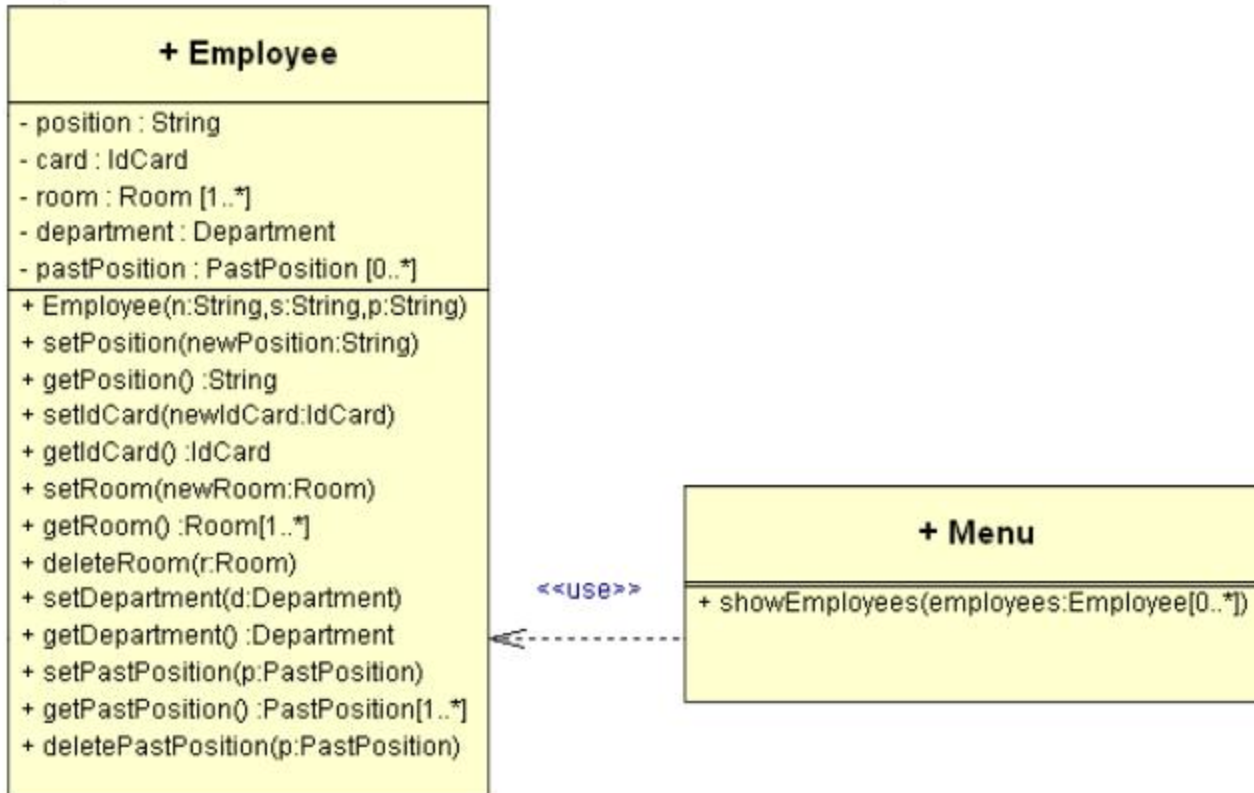
**Отношение агрегации**

# Диаграммы классов



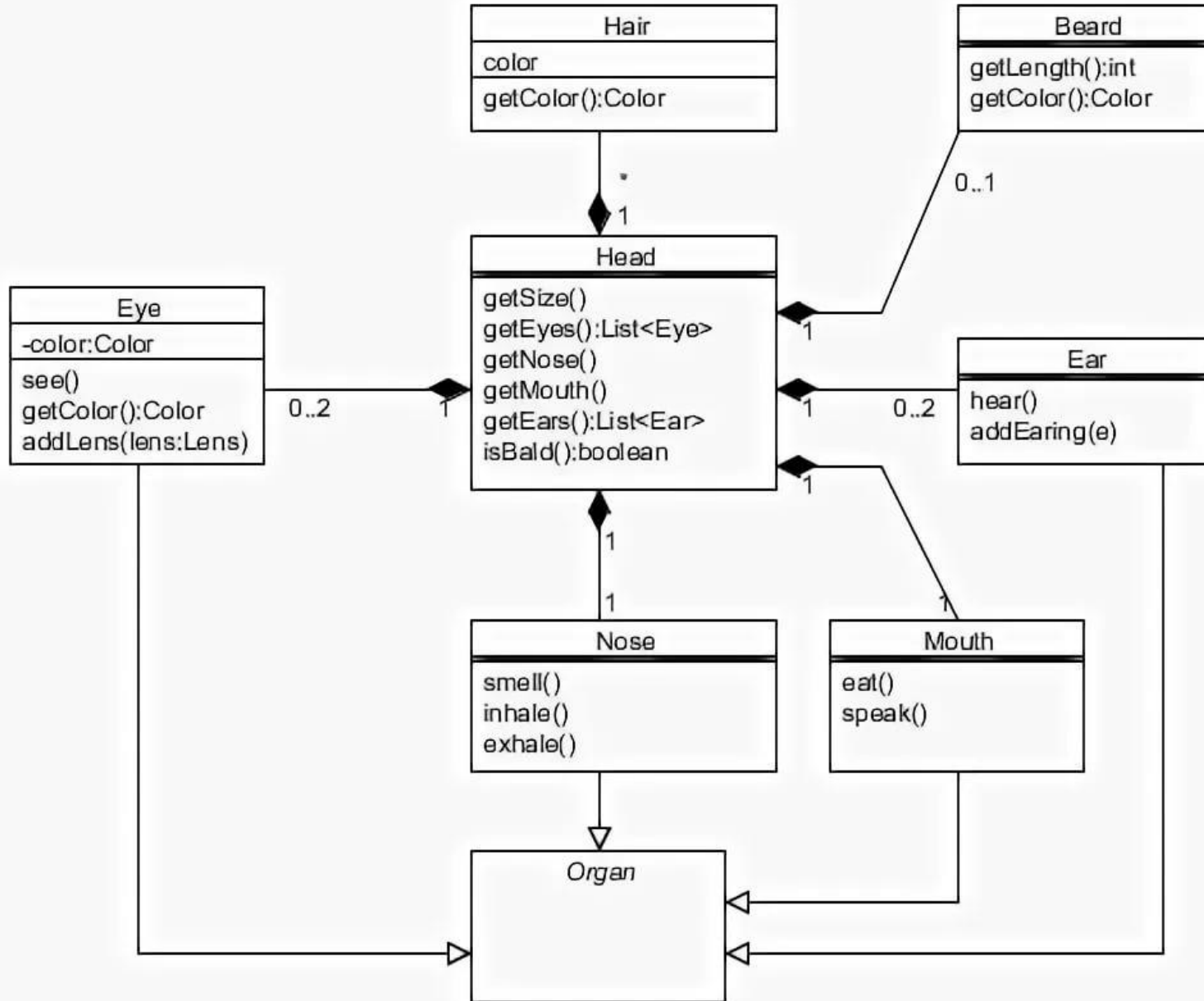
**Отношение композиции**

# Диаграммы классов



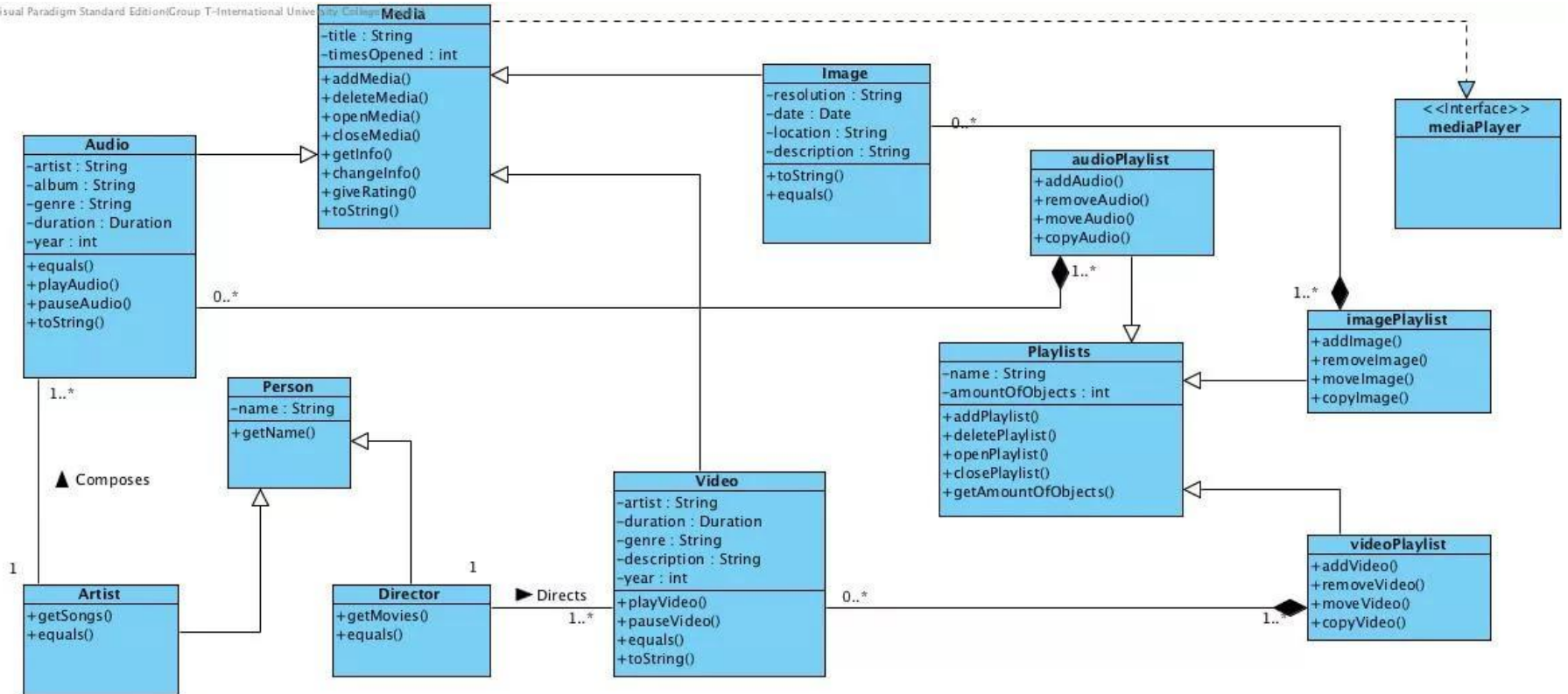
**Отношение использования  
(зависимости)**

# Диаграммы классов

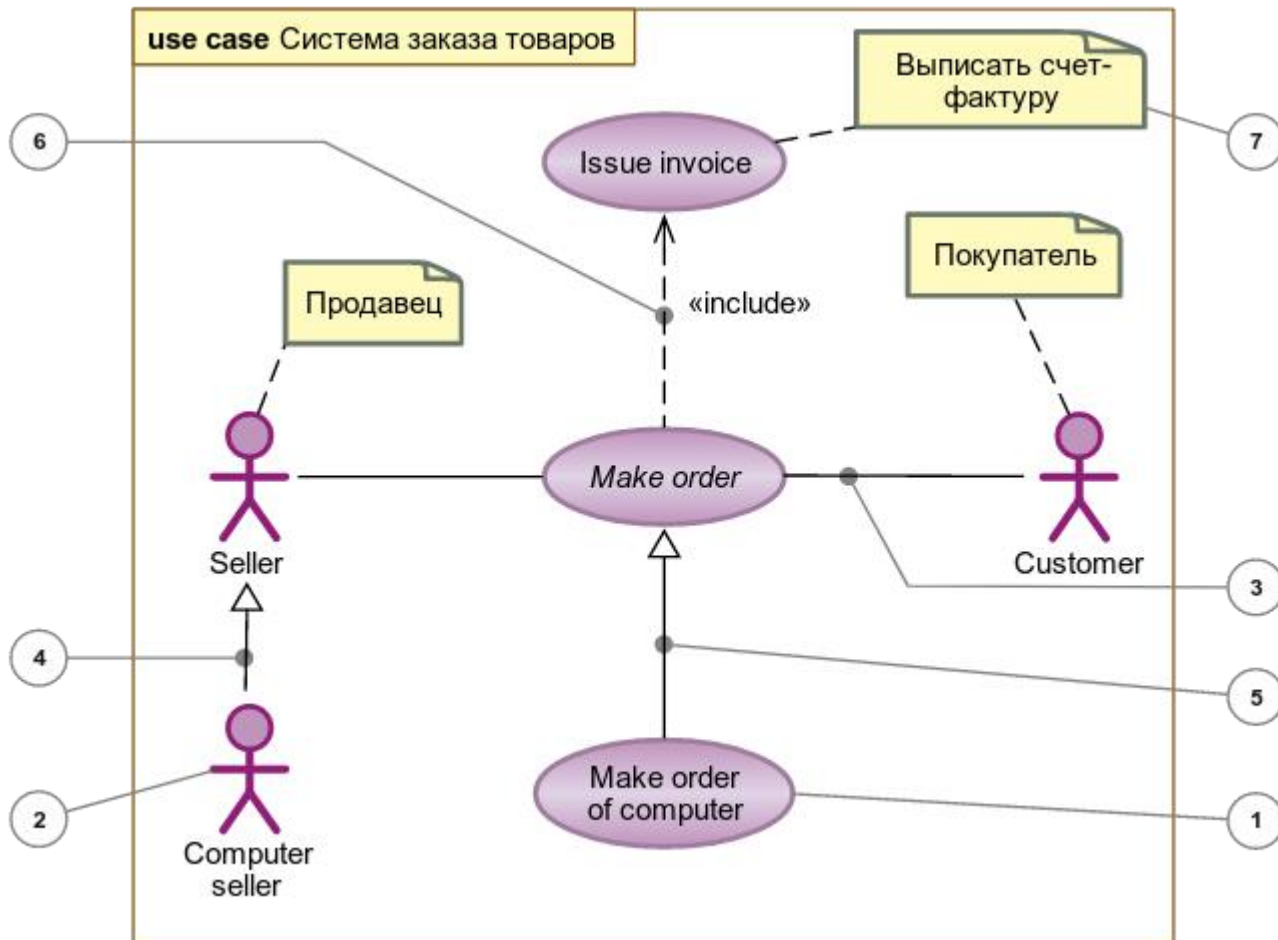


# Диаграммы классов

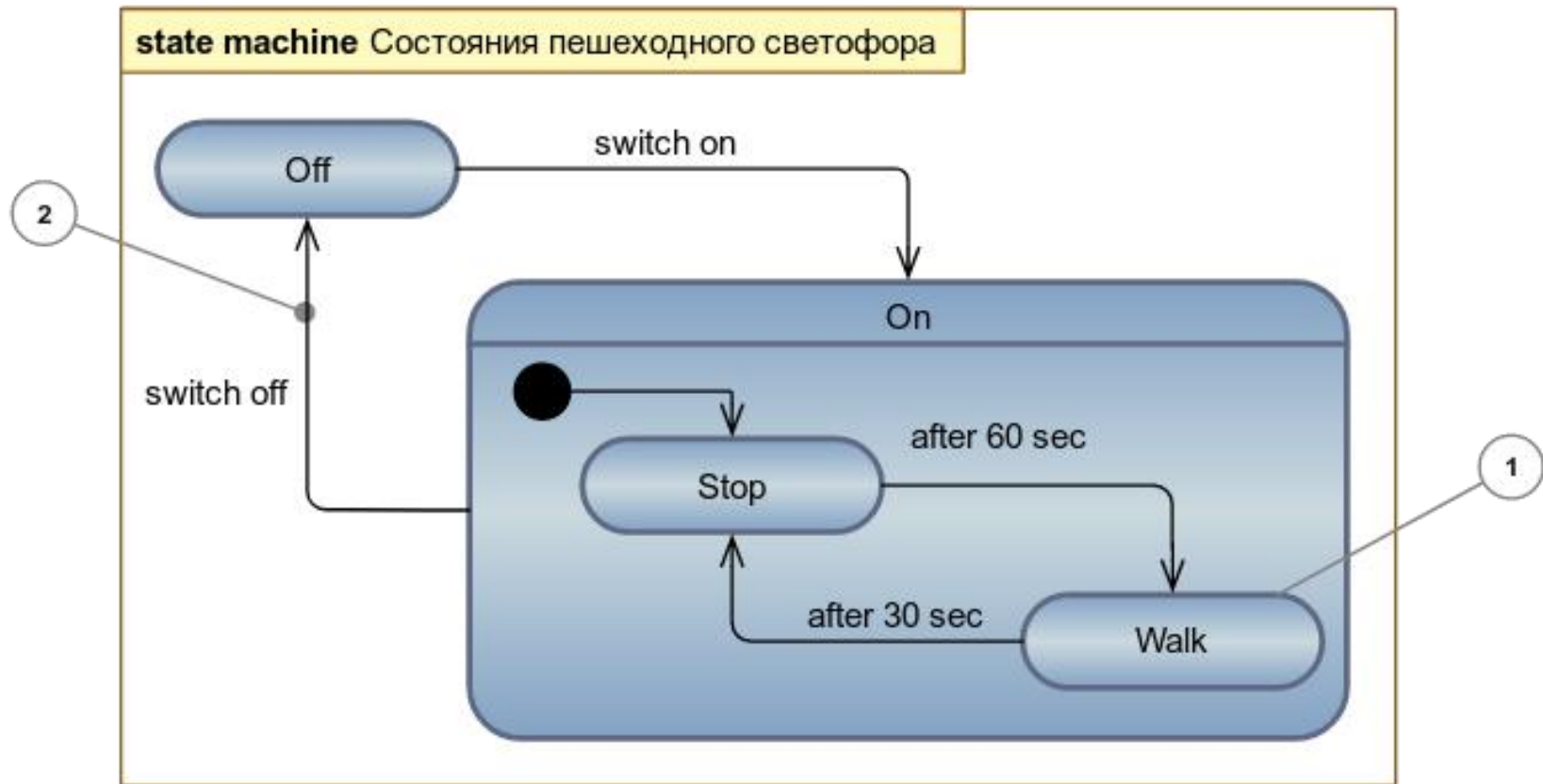
Visual Paradigm Standard Edition (Group T-International Unive



# Диаграмма использования (use case diagram)

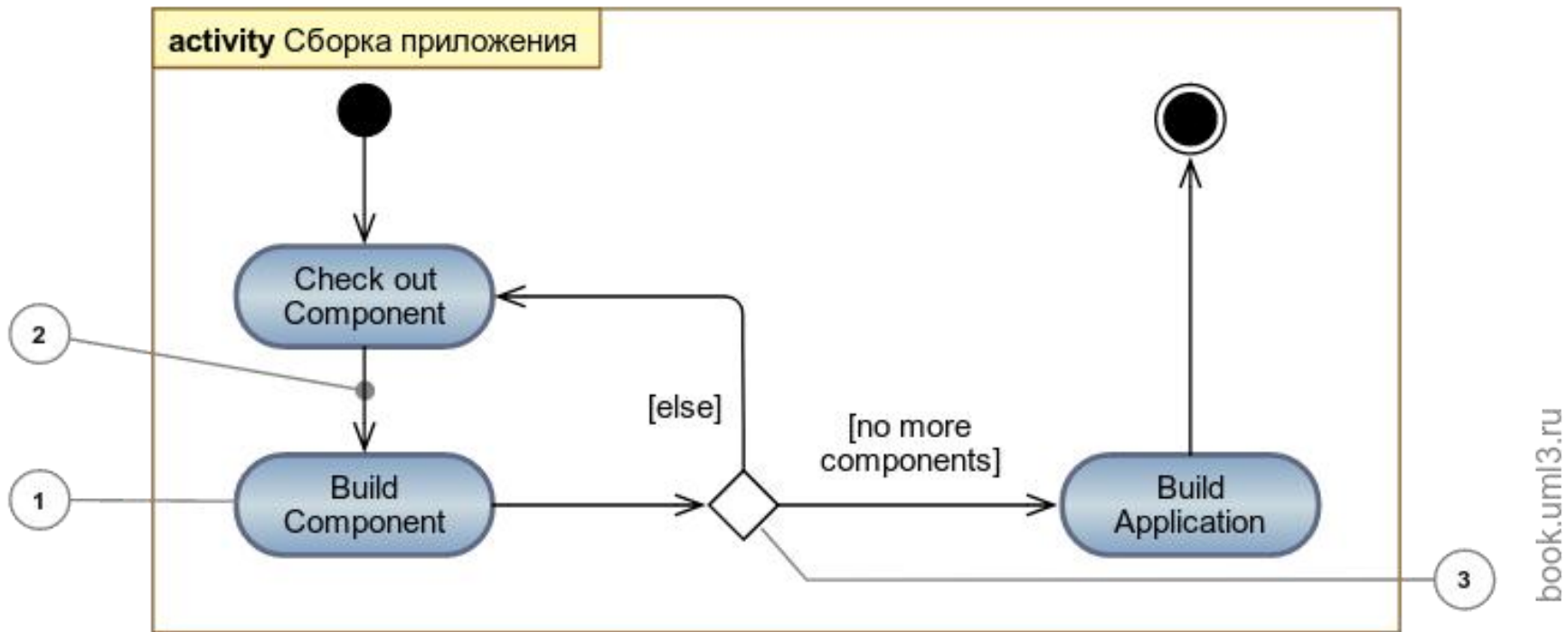


# Диаграмма автомата (state machine diagram)



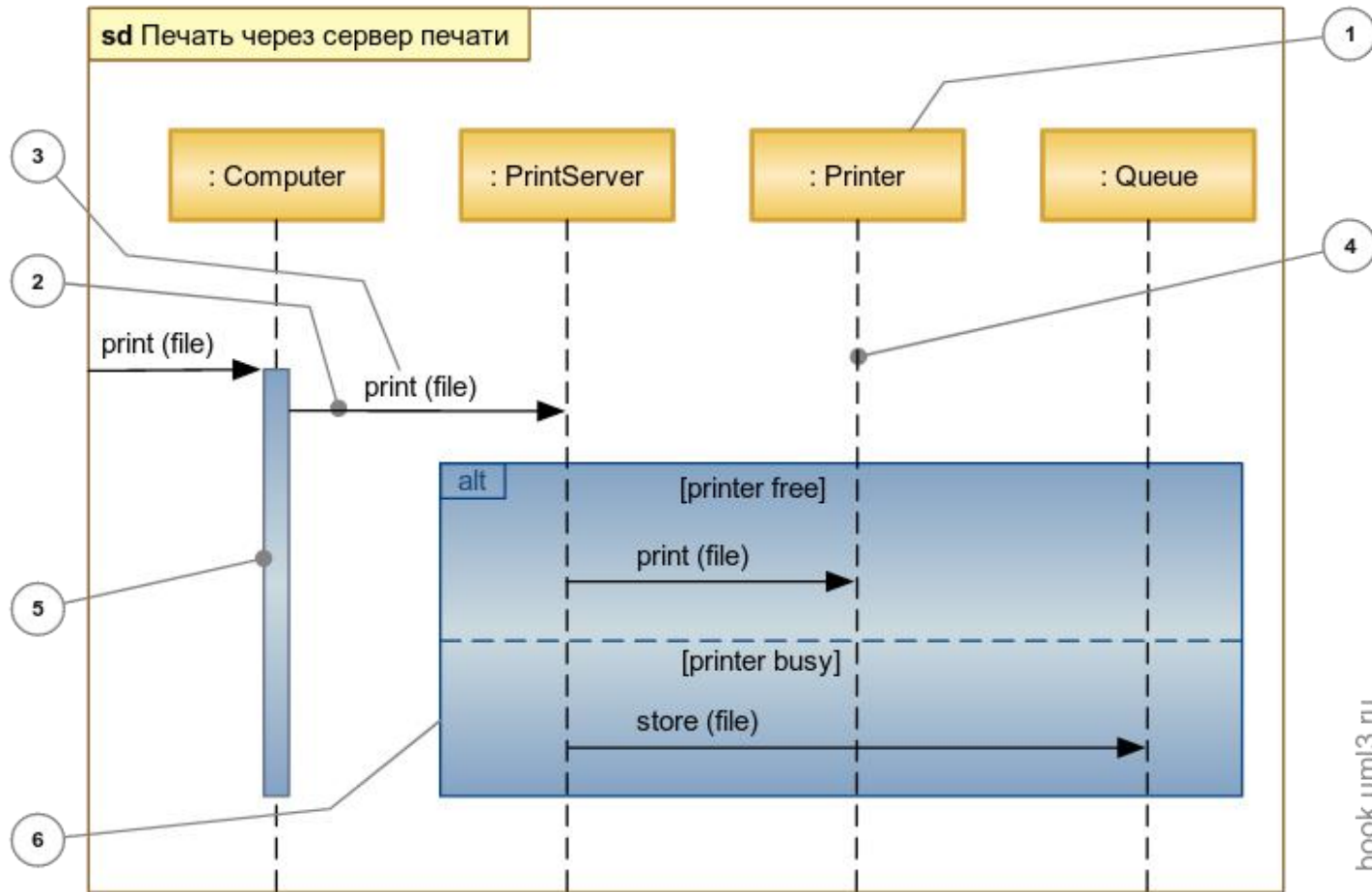
Описание поведения на основе явного выделения состояний и описания переходов между состояниями.

# Диаграмма деятельности (activity diagram)



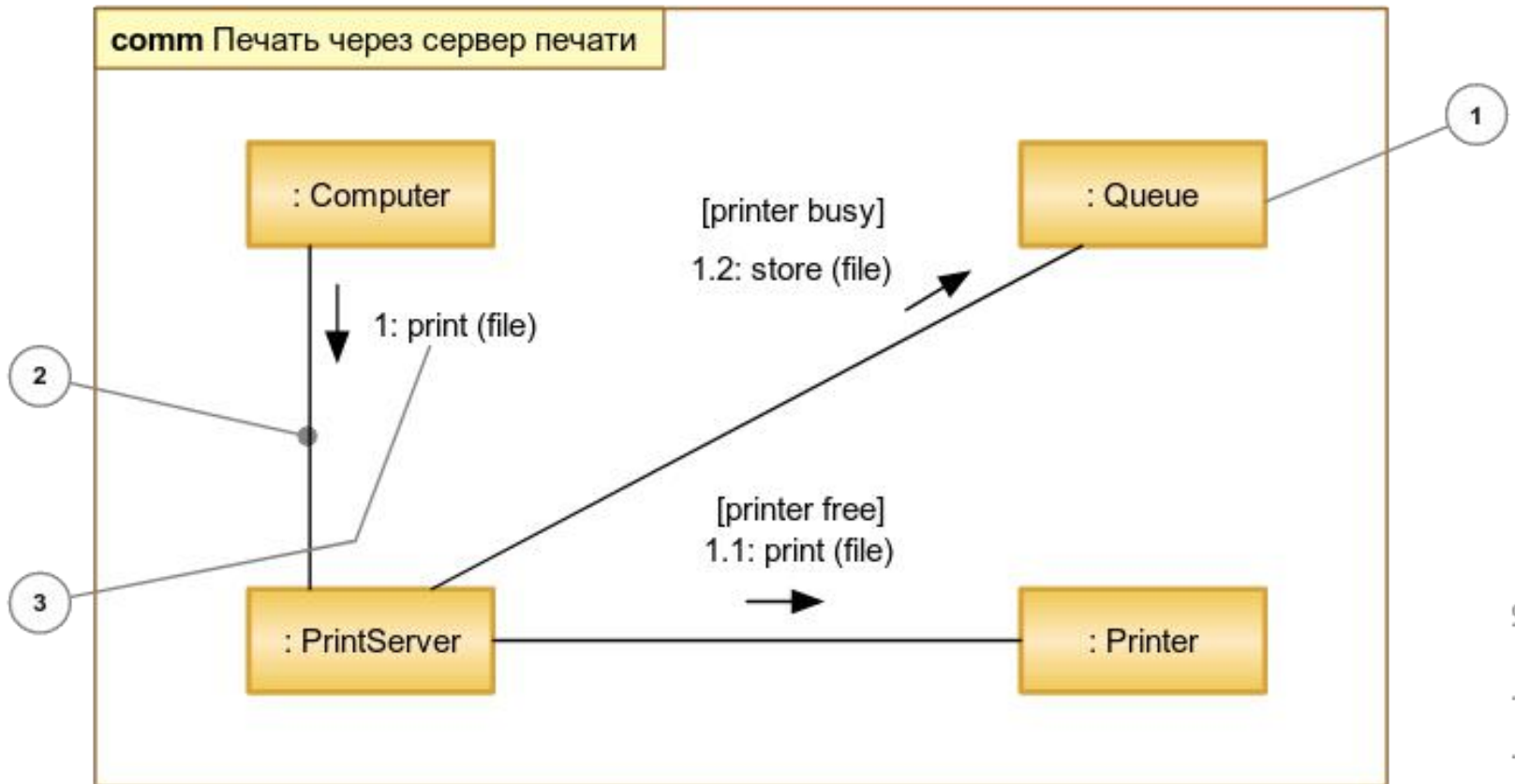
Описание поведения на основе указания потоков управления и потоков данных

# Диаграмма последовательности (sequence diagram)



Описание поведения системы на основе указания последовательности передаваемых сообщений

# Диаграмма коммуникации (communication diagram)



Акцент на структуре связей между конкретными экземплярами