

A decorative graphic on the left side of the slide. It features a vertical bar with a green-to-white gradient. To its right, a series of green squares of varying shades are arranged in a stepped pattern, resembling a staircase or a staircase-like structure. The squares are in different shades of green, from light to dark.

Диаграммы UML

Диаграмма классов
(*Class Diagram*)



Основные вопросы

- Что такое диаграмма классов
- Компоненты диаграммы классов и их назначение
- Пример диаграммы классов
- Расширение языка UML для построения моделей программного обеспечения и бизнес-систем



Диаграмма классов

- Является центральным звеном объектно-ориентированного подхода
- Содержит информацию об объектах системы и статических связях между объектами
- Отражает *декларативные знания* о предметной области
- Оперирует понятиями *класса, объекта, отношения, пакета*

Класс

- **Класс** – это множество объектов, которые обладают *одинаковой* структурой, поведением и отношениями с объектами из других классов.

Имя_класса

← Простейший вид класса состоит только из секции имени

Имя_класса

атрибуты класса

← Класс с указанием атрибутов (переменных)

Полное описание класса, состоящее из 3 разделов (секций) – секции имени, секции атрибутов, секции операций

Имя_класса

атрибуты класса

операции класса()



Класс

- **Имя класса** должно быть уникально
- Имя класса должно начинаться с заглавной буквы.
- Класс может не иметь экземпляров или объектов. В этом случае он называется **абстрактным классом**, а для обозначения его имени используется *курсив*



Атрибуты класса

- **Атрибут = свойство**, которое является общим для всех объектов данного класса
- Общий формат записи атрибутов:
*<квантор видимости> <имя атрибута>
[кратность]: <тип атрибута> =
<исходное значение> {строка-
свойство}*



Атрибуты класса.

Квантор видимости


- Квантор видимости может принимать одно из следующих значений: **+**, **#**, **-**, **~**.
- «**+**» - атрибут с областью видимости типа ***общедоступный*** (public).
- «**#**» - атрибут с областью видимости типа ***защищенный*** (protected).
- «**-**» - атрибут с областью видимости типа ***закрытый*** (private).
- «**~**» - атрибут с областью видимости типа ***пакетный*** (package).



Атрибуты класса.

Имя атрибута

- Представлено в виде *уникальной* строки текста
- Имя атрибута является единственным обязательным элементом в синтаксическом обозначении атрибута
- Должно начинаться со строчной буквы
- По практическим соображениям записывается *без пробелов*



Атрибуты класса.

Кратность атрибута

- ***Кратность атрибута*** характеризует общее количество конкретных атрибутов данного типа, входящих в состав отдельного класса.
- **Формат**: *[нижняя граница . . верхняя граница]*
- ***Примеры: [0..1], [0..*], [1..3, 5..7]***



Атрибуты класса. Тип атрибута

- Выражение, определяемое некоторым типом данных (например, в зависимости от языка программирования)
- В простейшем случае – *осмысленная строка текста*.
- Пример:
 - цвет: Color*
 - имяСотрудника[1..2]: String;*
 - видимость: Boolean*



Атрибуты класса.

Исходное значение


- Служит для задания некоторого начального значения в момент *создания* отдельного экземпляра класса

- Пример:

цвет: Color = (255, 0, 0)

*имяСотрудника[1..2]: String = 'Иван
Иванов';*

видимость: Boolean = истина



Атрибуты класса.

Строка-свойство

- Служит для указания **дополнительных свойств атрибута**, которые могут характеризовать особенности изменения значений атрибута в ходе выполнения соответствующей программы.
- Это значение принимается за **исходное значение атрибута**, которое не может быть изменено в дальнейшем.
- Пример:
заработнаяПлата: Currency = \$500 {frozen}



Операции класса

- Представляют собой некоторый сервис, который предоставляет каждый экземпляр класса или объект по требованию своих клиентов.

- Правила записи операций:

*<квантор видимости> <имя операции>
(список параметров): <выражение
типа возвращаемого значения>
{строка-свойство}*




Операции класса.

Список параметров

- *Список параметров является перечнем разделенных запятой формальных параметров, каждый из которых, в свою очередь, может быть представлен в следующем виде:*

*<вид параметра> <имя параметра> :
<выражение типа> = <значение
параметра по умолчанию>*



Операции класса.

Строка-свойство

- **Строка-свойство** служит для указания значений свойств, которые могут быть применены к данной операции.
- Например, для указания последовательности действий будет использована строка-свойство вида:

{concurrency = имя} ,

где *имя* может принимать одно из следующих значений:

- **sequential** (последовательная),
- **concurrent** (параллельная),
- **guarded** (охраняемая)



Операции класса. Примеры

- +нарисовать (форма : Многоугольник = прямоугольник, цветЗаливки : Color = (0, 0, 255));
- -изменитьСчетКлиента (номерСчета : Integer) : Currency;
- #выдатьСообщение() : ('Ошибка деления на ноль').



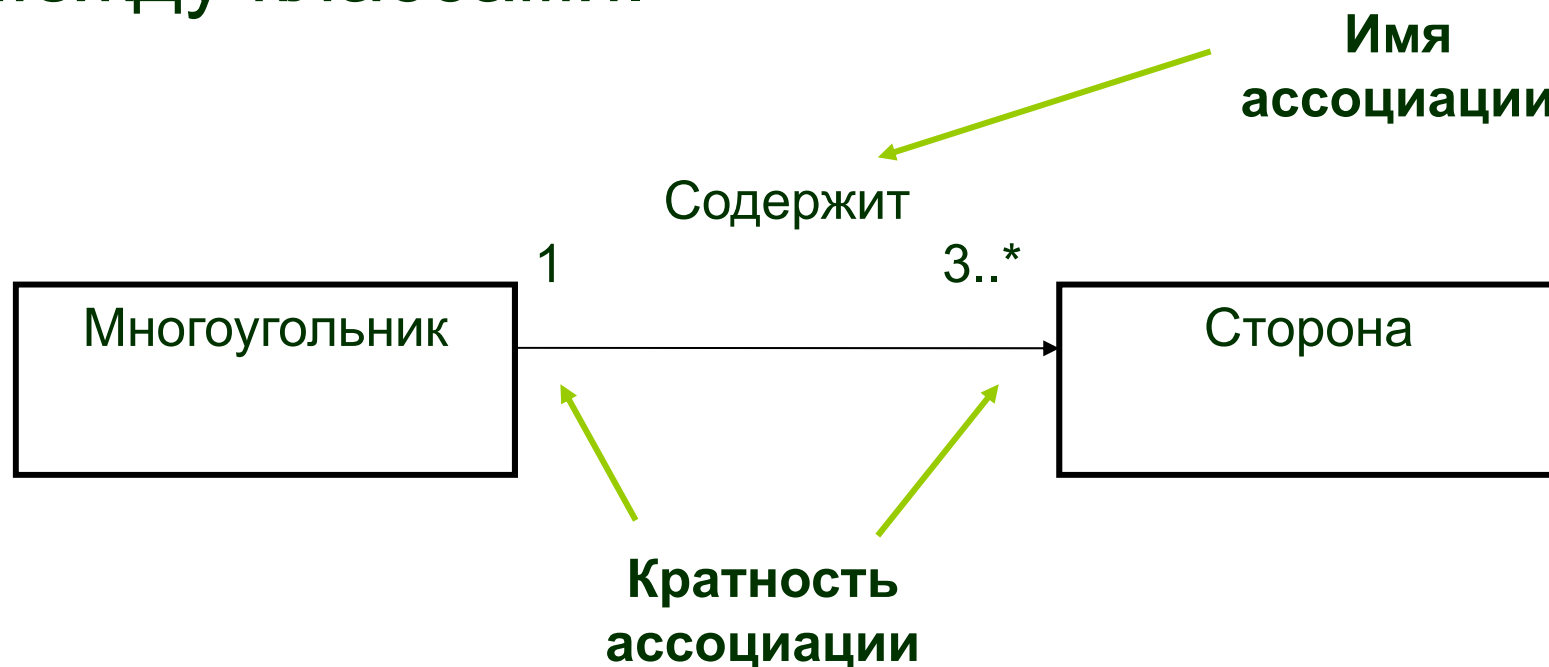
Отношения между классами

Базовыми отношениями на диаграмме классов являются:

- отношения **ассоциации** (*association*);
- отношения **обобщения** (*generalization*);
- отношения **агрегации** (*aggregation*);
- отношения **композиции** (*composition*);
- отношения **зависимости** (*dependency*).

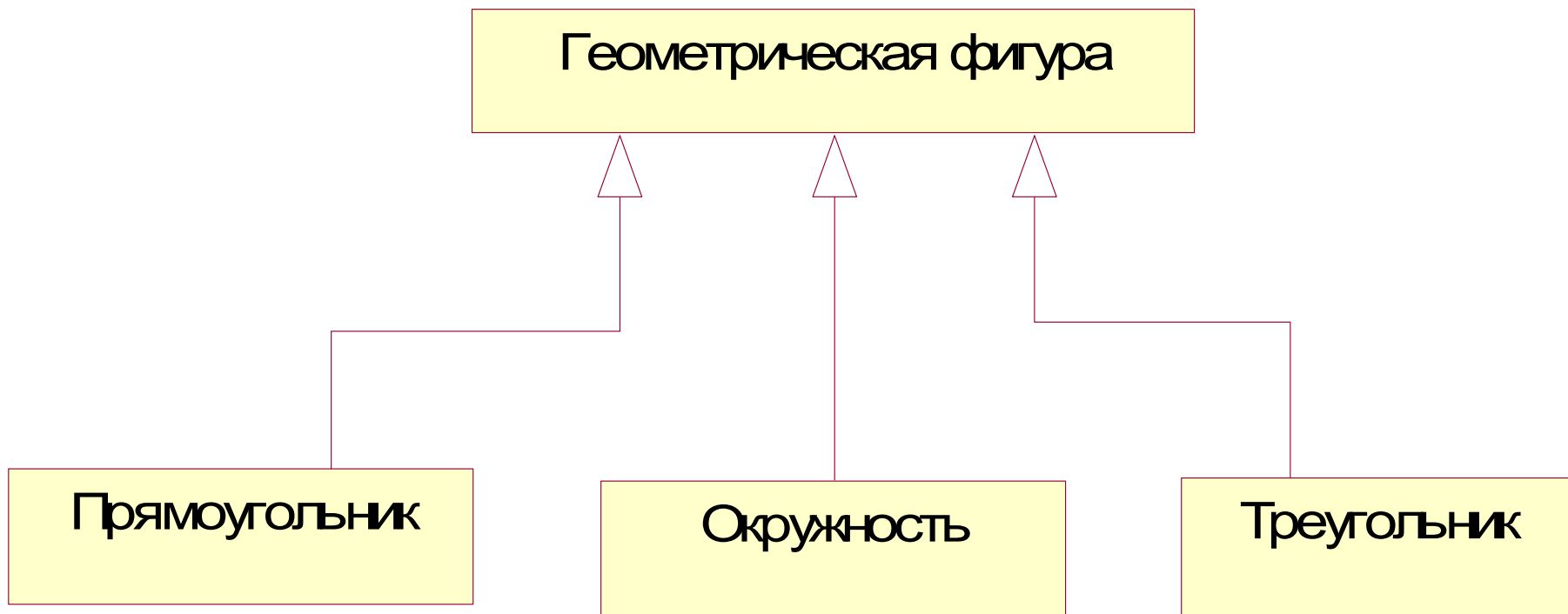
Отношение ассоциации

- **Отношение ассоциации** свидетельствует о наличии произвольного отношения между классами.



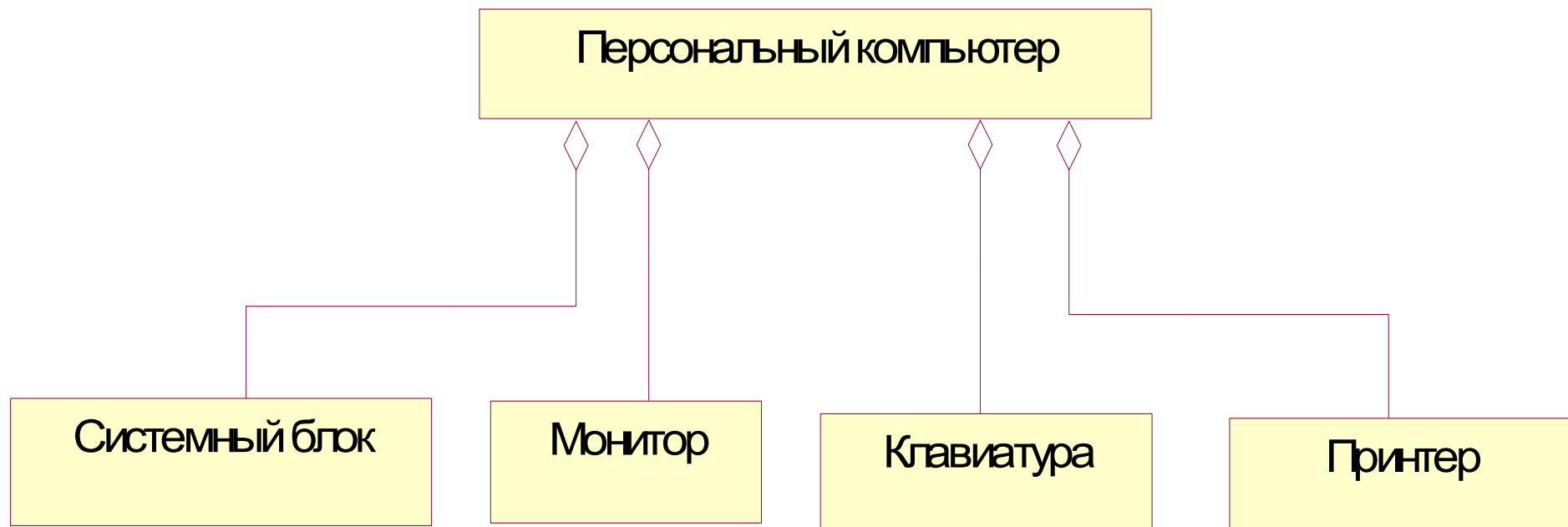
Отношение обобщения

- Является отношением *классификации* между более общим элементом (родителем или предком) и более частным или специальным элементом (дочерним или потомком)



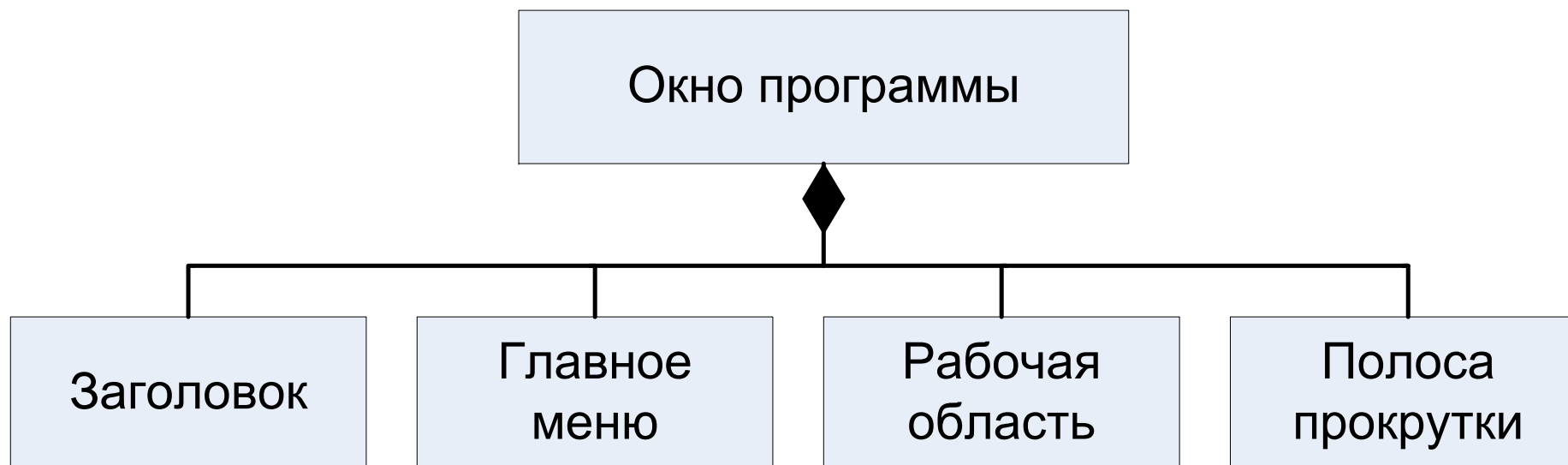
Отношение агрегации

- Смысл: один из классов представляет собой некоторую сущность, которая **включает** в себя в качестве составных частей другие сущности.
- Применяется для представления системных взаимосвязей типа «часть-целое».



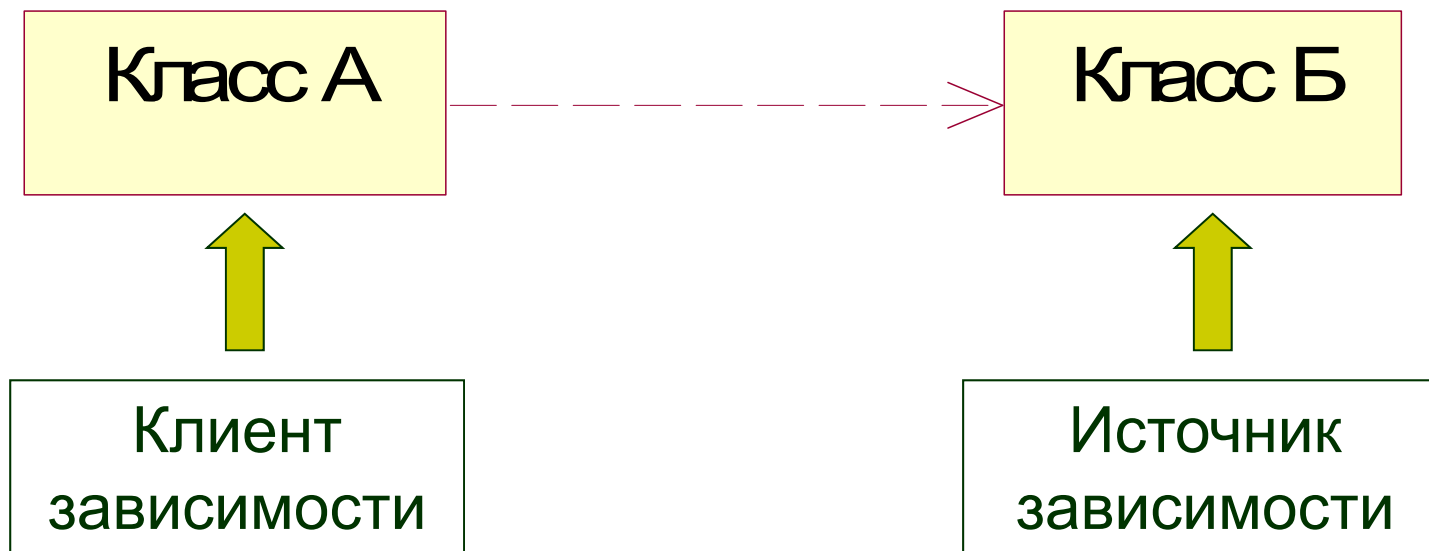
Отношение композиции

- Является частным случаем отношения агрегации.
- Части не могут выступить в отрыве от целого, т.е. с уничтожением целого уничтожаются составные части.



Отношение зависимости

- Используется в такой ситуации, когда некоторое изменение одного элемента модели может потребовать изменения другого элемента.





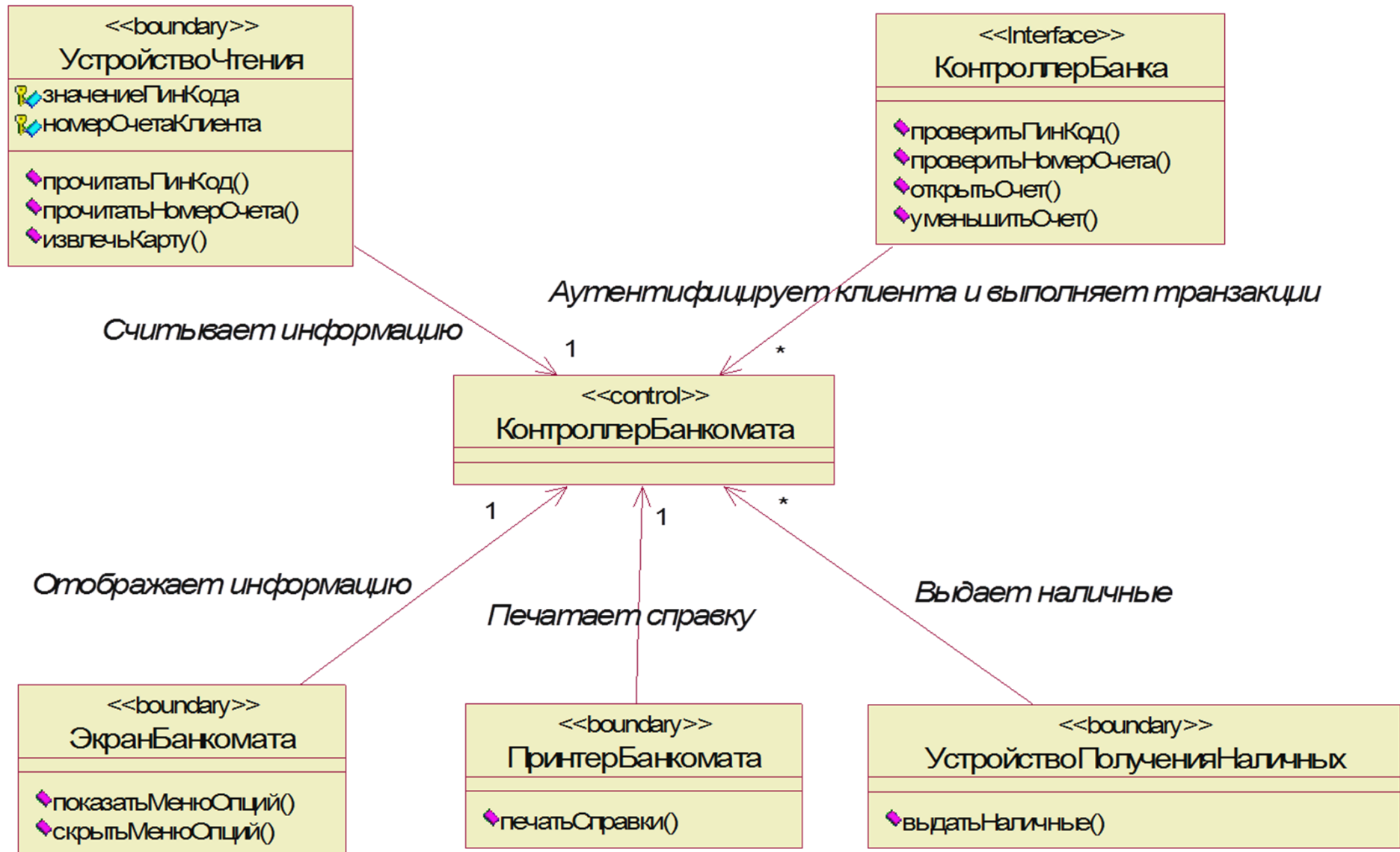
Пакеты

- служат для **группировки** элементов модели
- Любой пакет владеет своими элементами
- любой элемент может принадлежать *только одному пакету*



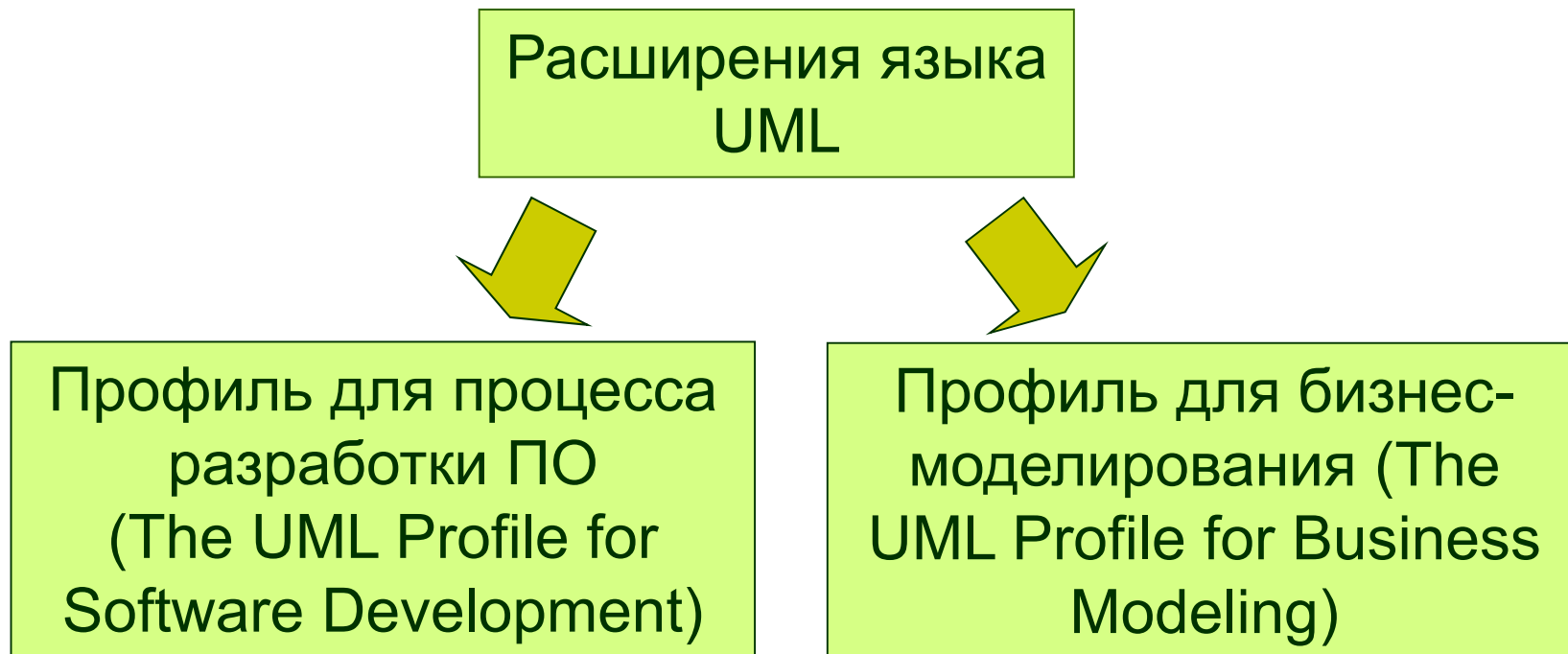
имя пакета

Пример диаграммы классов





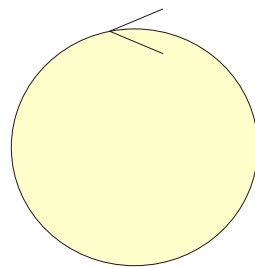
Расширения языка UML





Профиль для процесса разработки ПО

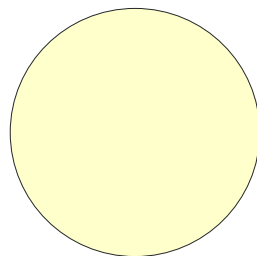
- **Управляющий класс (control)** – отвечает за координацию действий других классов.



NewClass

Профиль для процесса разработки ПО

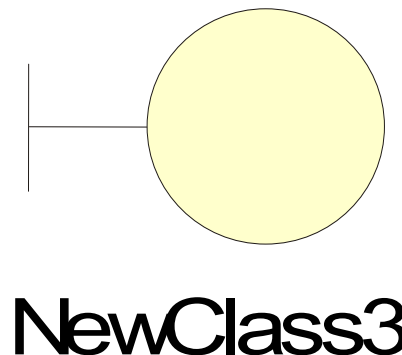
- **Класс-сущность (entity)** содержит информацию, которая должна храниться **постоянно** и не уничтожаться с уничтожением объектов данного класса или прекращением работы моделируемой системы.



NewClass2

Профиль для процесса разработки ПО

- **Граничный класс (boundary)** – располагается на границе системы с внешней средой, но является составной частью системы.



Интерфейс (interface)

- в контексте языка UML является специальным случаем класса, у которого имеются только операции и отсутствуют атрибуты.

