Диаграммы классов

- UML предлагает использовать **три уровня** диаграмм.
- Уровни зависят от степени детализации классов.

Уровни диаграмм классов:

- ***Концептуальный уровень*** **- этап анализа;**
- ***Уровень спецификаций*** **- этап проектирования;**
- ***Уровень реализации*** **- этап реализации.**

Краткое описание уровней:

- ***Концептуальный***
  - ✓ Диаграммы называют ***контекстными***.
  - ✓ Диаграммы демонстрируют связи между основными *понятиями* предметной области.

- ***Спецификаций***
  - ✓ Диаграммы отображают **связи объектов** классов.

- ***Реализации***
  - ✓ Диаграммы показывают ***особенности проектирования*** конкретных *классов*.

Обозначение класса на концептуальной диаграмме

Рецептов на предмет какую часть следует считать объектом, **не существует**.

Атрибуты - это существенные с точки зрения решаемой задачи характеристики объектов.

- ✓ Имя **атрибута** уникально **в пределах класса**.
- ✓ Имя **класса** уникально **в пределах пакета**.

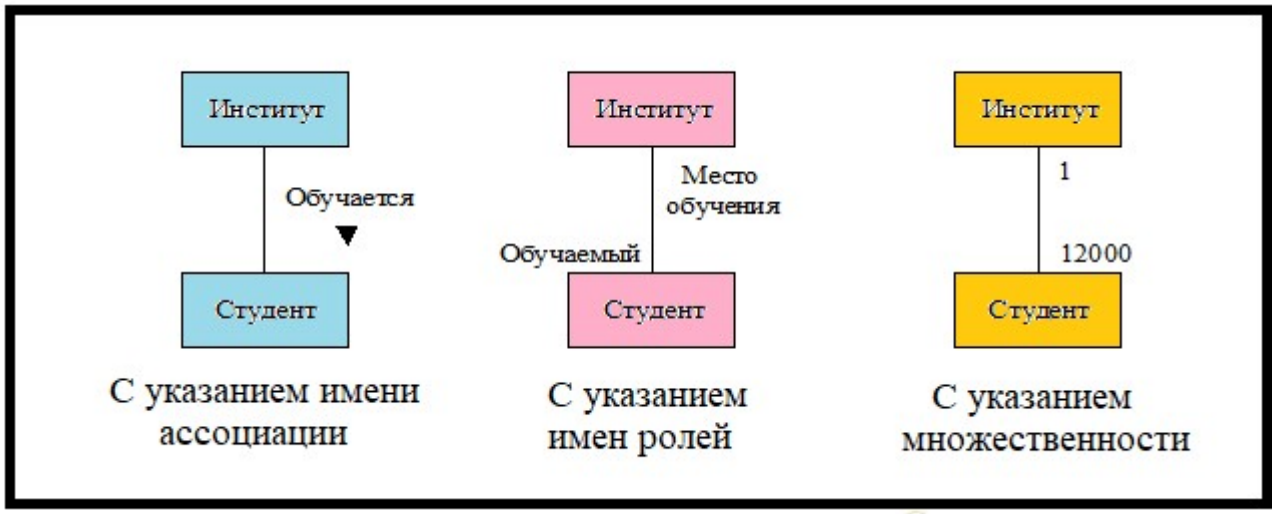
(Для указания, к какому пакету относится класс, добавляют имя пакета: **<Имя_пакета>::< Имя_класса >**)

∇ Под ***отношением*** классов понимают **статическую** связь между классами (не зависящую от времени).

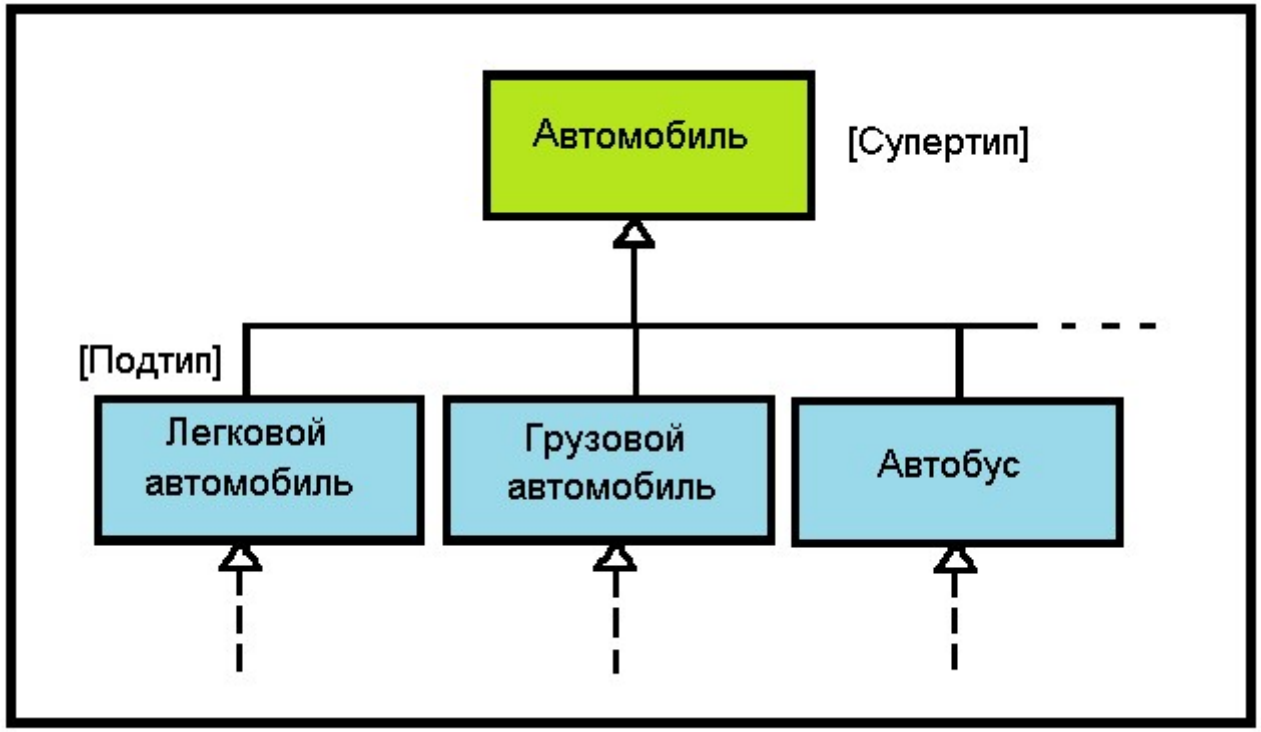
Основные виды отношений**: ассоциация и обобщение**.

Ассоциация - означает наличие связи между экземплярами классов или объектами.

(например, класс «студент» ассоциирован с классом «институт»).



Обобщение - означает, что два класса находятся в таком отношении, что любой объект **одного класса** (***подтипа***) обязательно является также и объектом **другого класса** (**супертипа**).



Виды классов

Классы-сущности - для представления сущностей реального мира или внутренних элементов системы.

- ✓ Обычно такие классы **не зависят от окружения** и могут использоваться в различных приложениях.

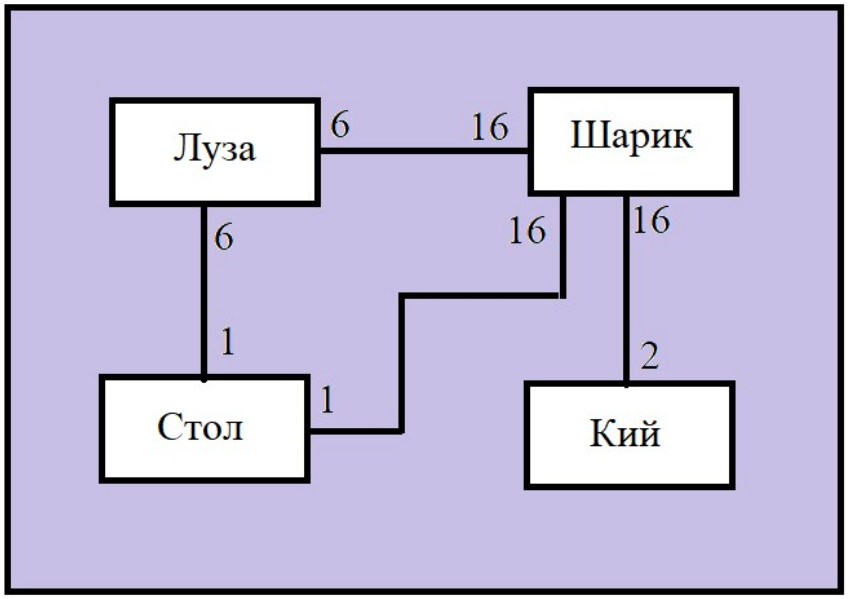
(Например, шарик в игре бильярд, процесс в ОС и др.)

Граничные (интерфейсные) классы - обеспечивают взаимодействие объектов внешнего мира с внутренними элементами системы

(Например, класс, реализующий пользовательский интерфейс и др.).

Управляющие классы - служат для **моделирования** последовательного поведения, заложенного в один или несколько **вариантов использования**.

(Вопрос: Нужен в игре бильярд управляющий класс?)

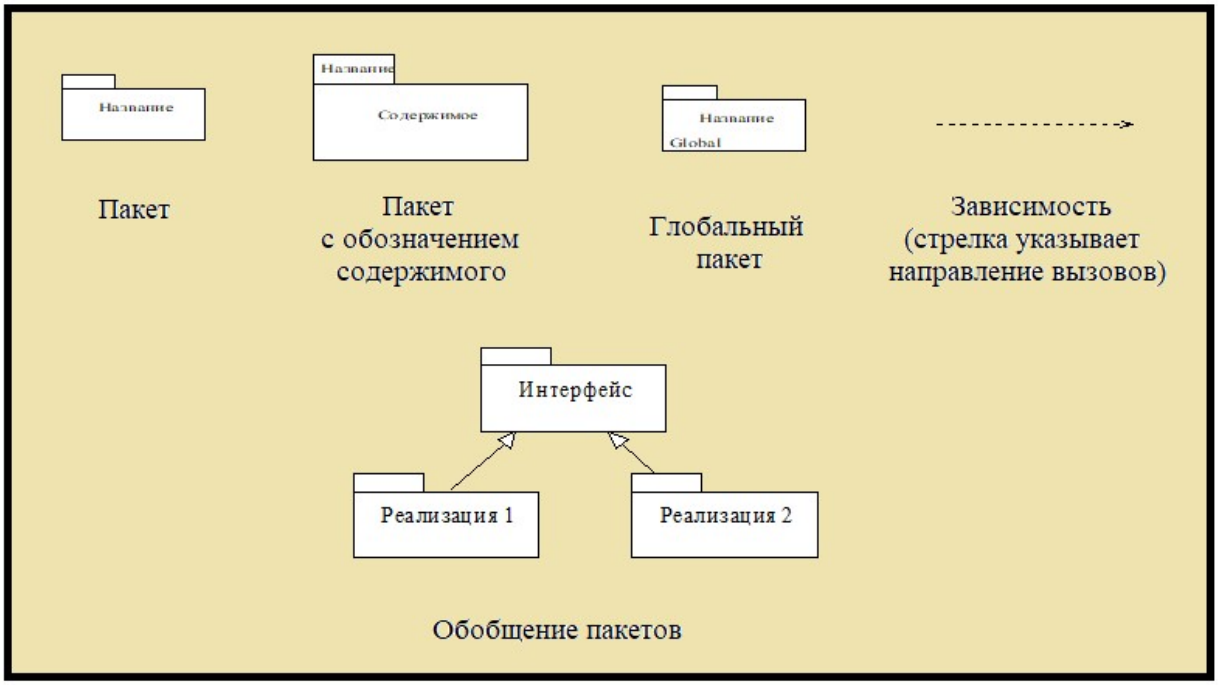


*Если количество классов-кандидатов **велико**, то их целесообразно разбить на группы – **пакеты**.

(Обычно объединяют классы и другие ресурсы по ***назначению)***

*Пакеты, от которых зависят другие пакеты программной системы, называют ***глобальными***.

Обозначения пакетов в UML:

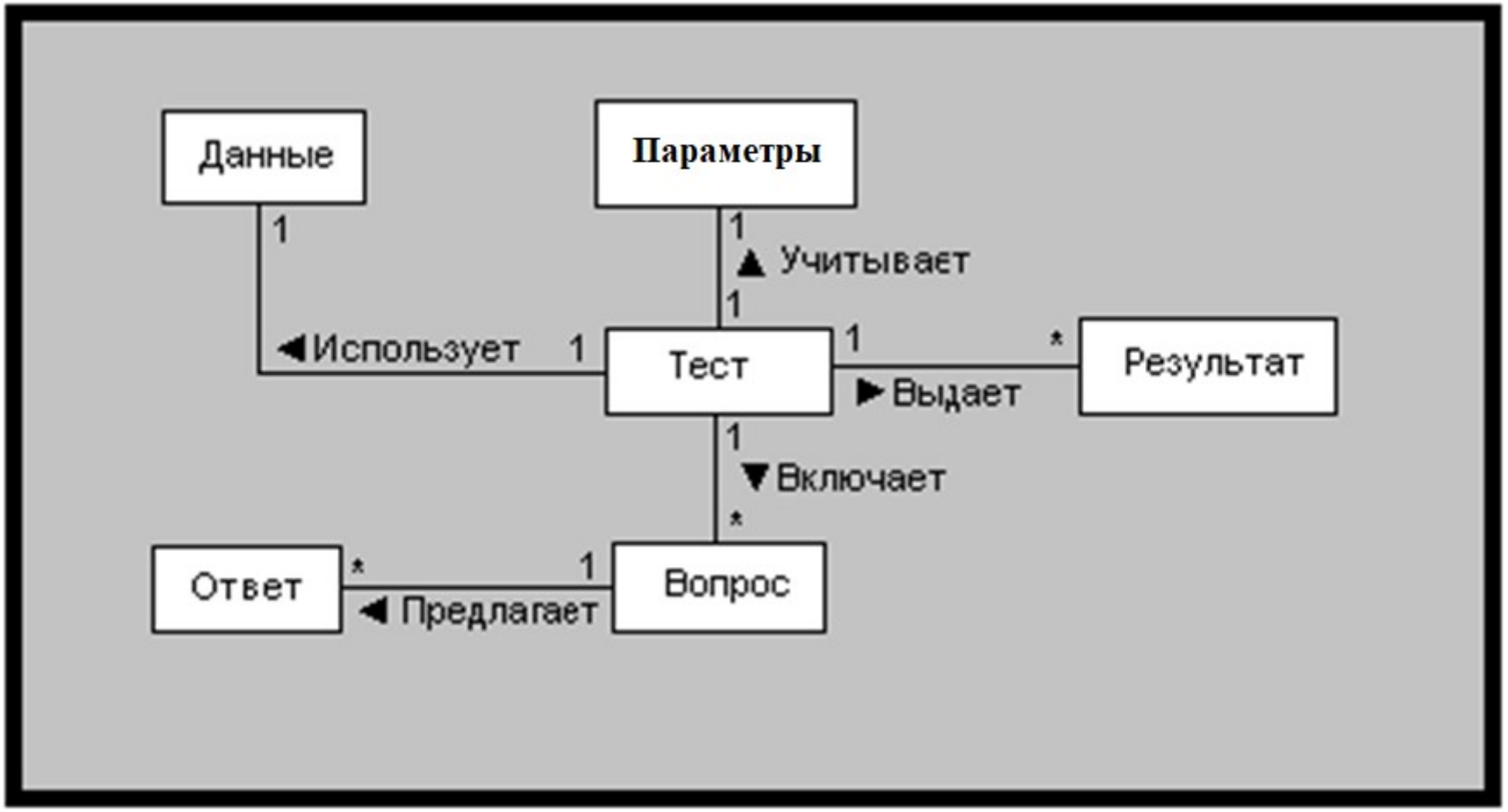


✓ После определения структуры **переходят к проектированию классов**, входящих в каждый пакет.

Виды зависимости классов:

- **объекты** одного класса **посылают сообщения** объектам другого класса;
- **объекты** одного класса **обращаются к данным** объектов другого;
- **объекты** одного класса **используют объекты** другого в списке параметров методов
- и т.п.

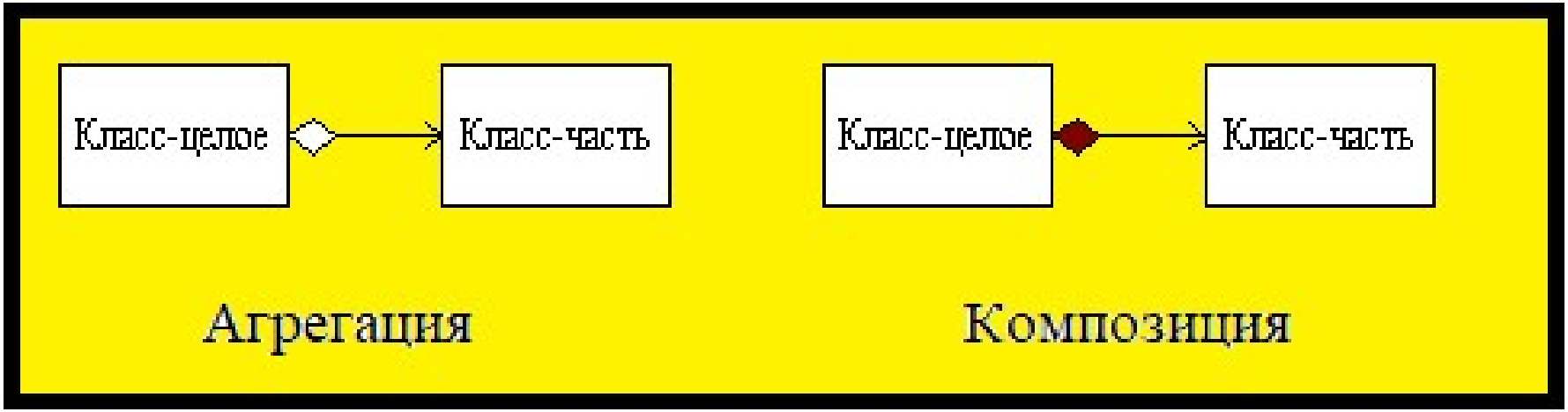
Пример



Исходная диаграмма классов

Уточнение отношений классов

- ✓ Процесс проектирования классов **начинают с уточнения отношений** между ними.
- ✓ На этапе проектирования (кроме ассоциации и обобщения) различают : **агрегацию и композицию**.



Условные обозначения специальных видов ассоциации

Описание

Агрегацией называют ассоциацию между **целым и его частью** или **частями**.

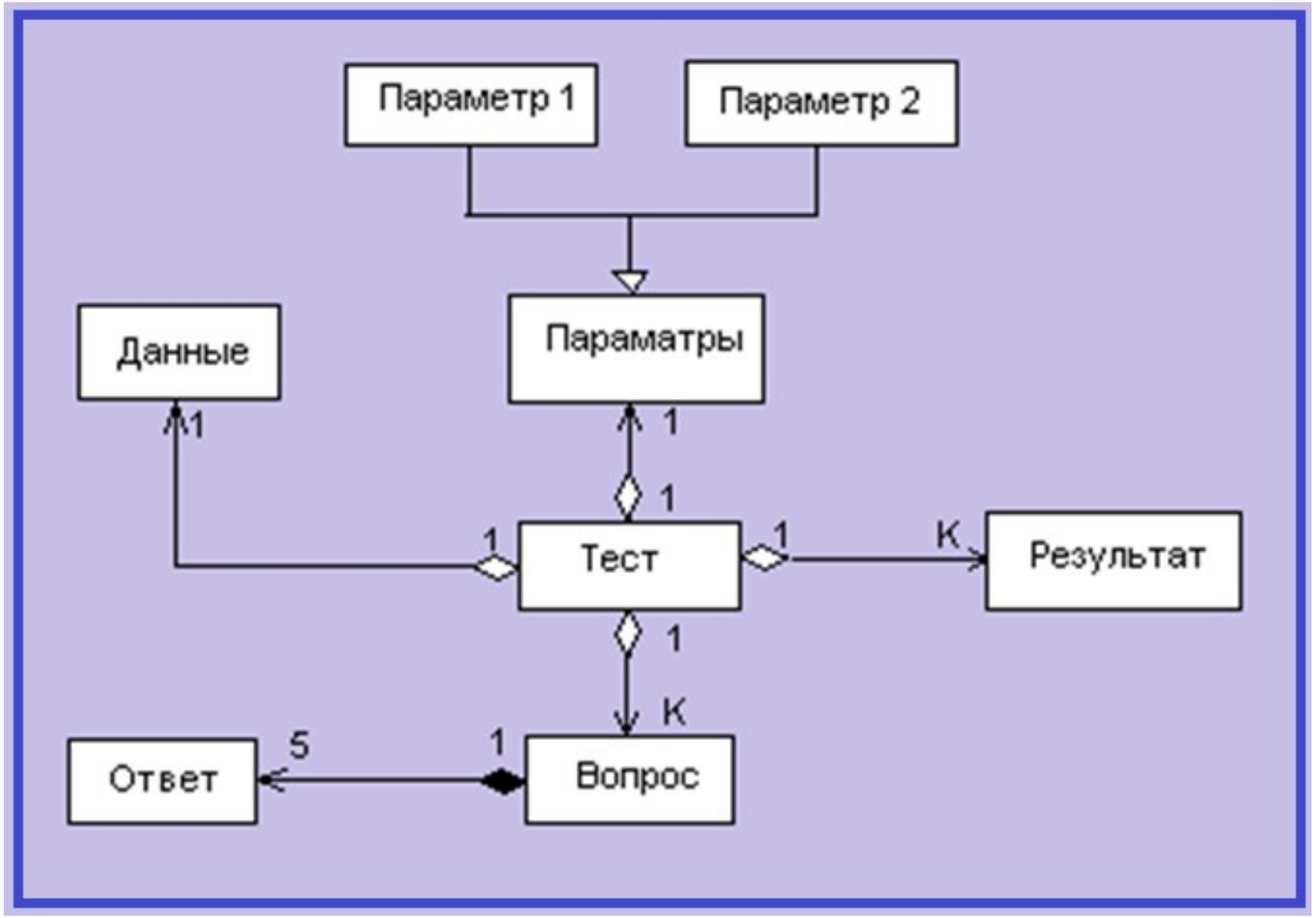
*(*Агрегацию **вместо** ассоциации указывают, если отношение «целое-часть» в конкретном случае существенно.)

Композиция – более **сильная разновидность агрегации**, которая подразумевает, что объект-часть может принадлежать только единственному целому.

(Объект-часть при этом создается и уничтожается только вместе со своим целым.)

При необходимости можно **уточнить направление передачи сообщений** *(показать навигацию ассоциации)*.

Пример уточненной диаграммы классов:



Проектирование классов

Завершают проектирование классов, **когда** окончательно определены ***структура*** и ***поведение*** объектов.

Условное обозначение класса в UML

Атрибут может включать:

<признак видимости> <имя>:<тип>=<значение по умолчанию>

Значения признаков видимости:

«+» – общедоступный (виден из любого класса пакета);

«#» – защищенный (виден только подклассами);

«-» – скрытый (невиден для всех классов).

Операции **-** основные **действия**, реализуемые классом.

В отличие от методов операции **не всегда** реализуются.

Описание:

<признак видимости> <имя>(<список параметров>):
<тип возвращаемого значения>

Ответственность - краткое неформальное перечисление основных функций объектов класса.

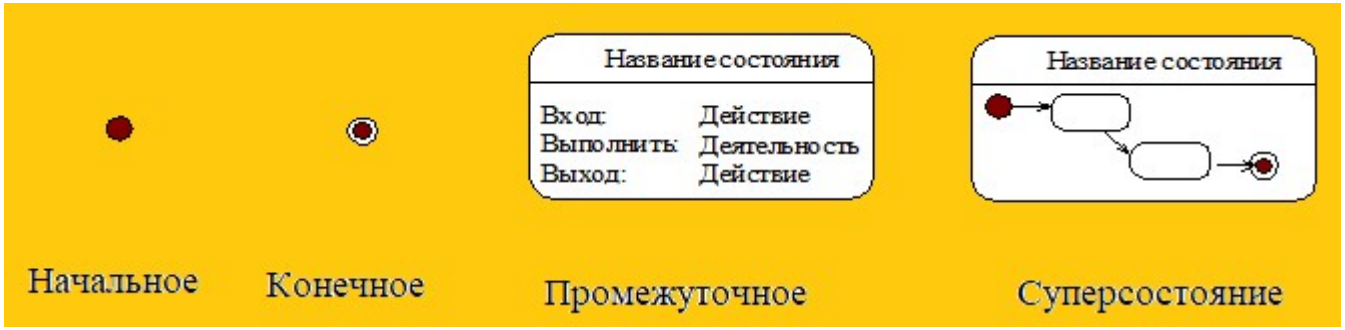
Для объектов ***сложного поведения*** целесообразно разрабатывать **диаграммы состояний**.

Диаграмма состояний объекта

Показывает:

- ✓ состояния объекта;
- ✓ возможные переходы;
- ✓ события или сообщения (вызывающие переход).

Обозначения:



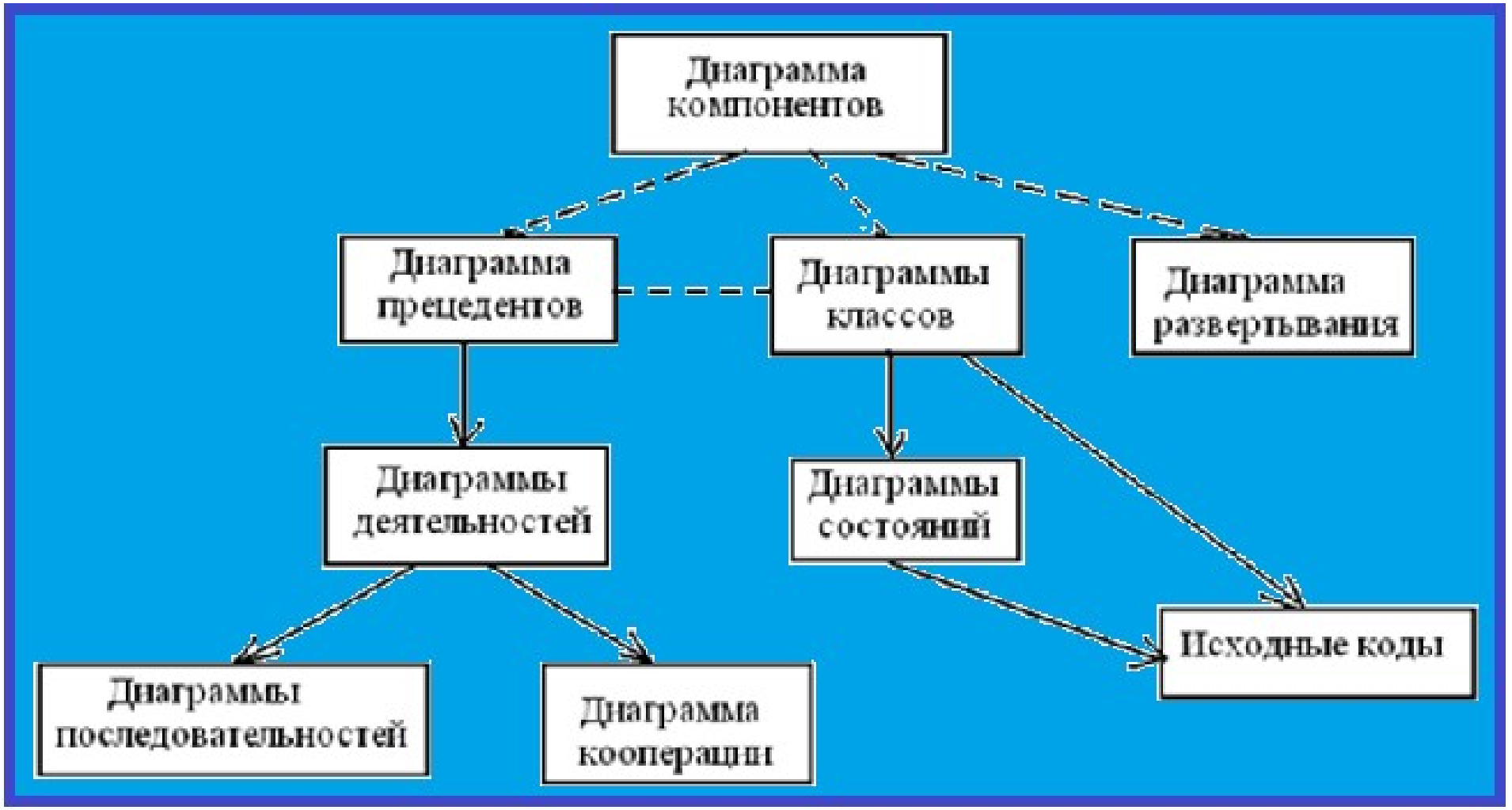
Переход - линия со стрелкой.

Линия помечается меткой: **<Событие> [<Условие>]/<Действие>**

Объект не может одновременно перейти в два разных состояния, поэтому условия должны быть взаимоисключающими.

Обобщенная схема процесса разработки диаграмм

Полный проект представляет собой совокупность моделей **логического** и **физического** представлений.



Физическое представление (**диаграммы реализации**)

Включает:

- *диаграмму* ***компонентов****;*
- *диаграмму* ***развертывания****.*

Диаграмма компонентов:

- оперирует понятиями ***компонент*** и ***зависимость***.
- обеспечивает согласованный переход от логического представления к конкретной реализации проекта в форме программного кода.

• Под компонентами понимают **физические заменяемые** части системы.

Компоненты бывают:

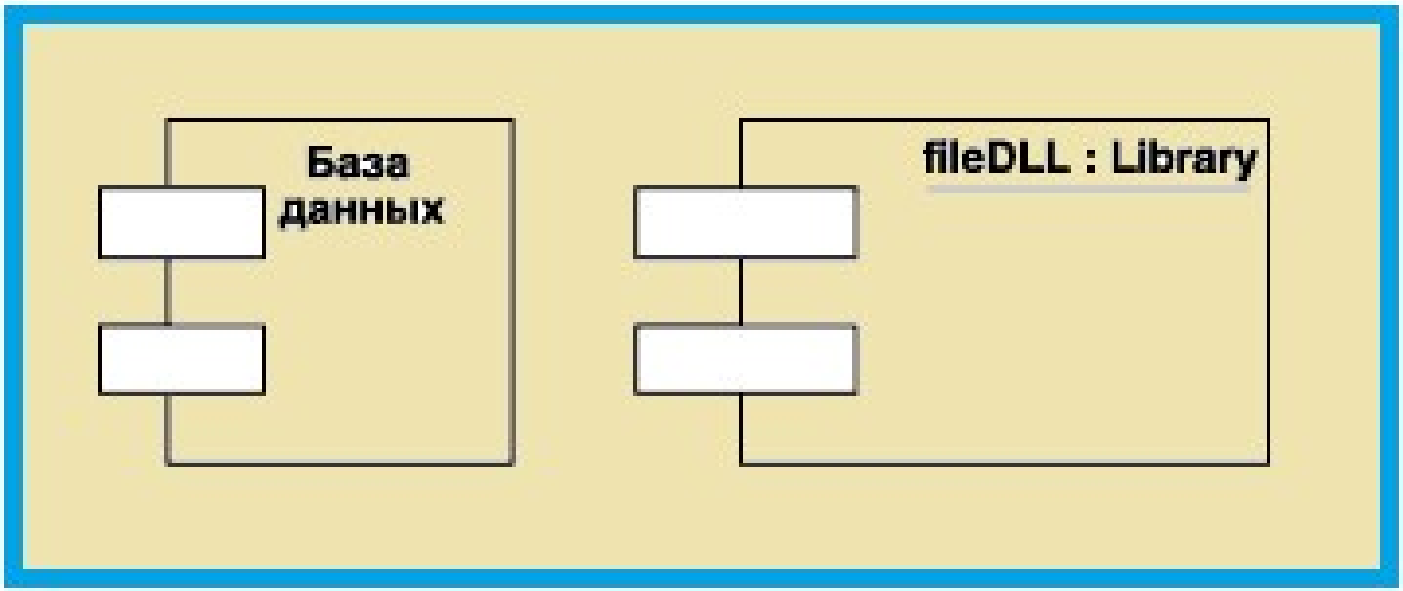
✓ существующие только на этапе компиляции;

✓ существующие на этапе его исполнения.

Компонентами могут быть:

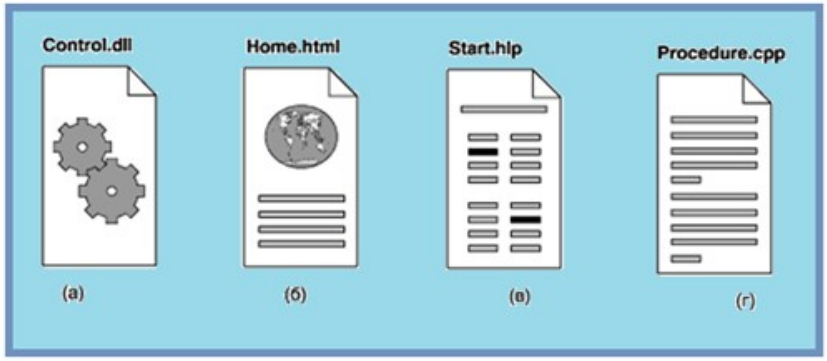
- ∇ *исполняемый код* отдельного модуля;
- ∇ *командные* файлы;
- ∇ файлы, содержащие *интерпретируемые скрипты*.

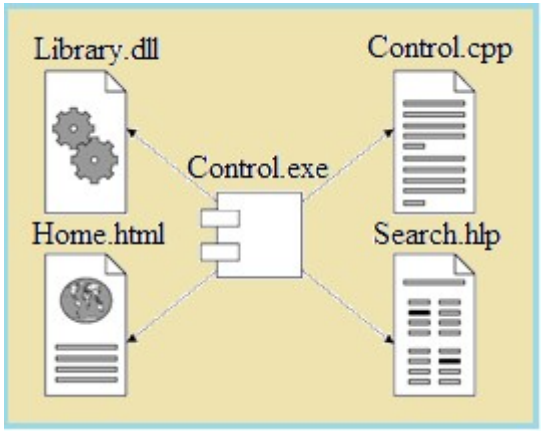
Графическое представление:



Ниже приведены:

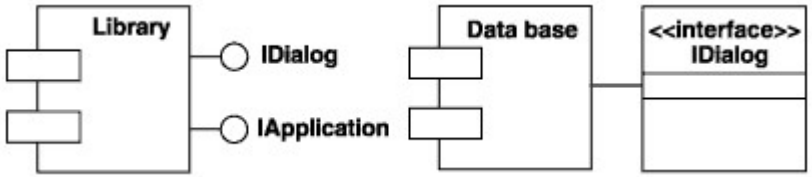
- динамически подключаемые библиотеки (а);
- Web-страницы на языке разметки гипертекста (б);
- файлы справки (в);
- файлы с исходными текстами программ (г).



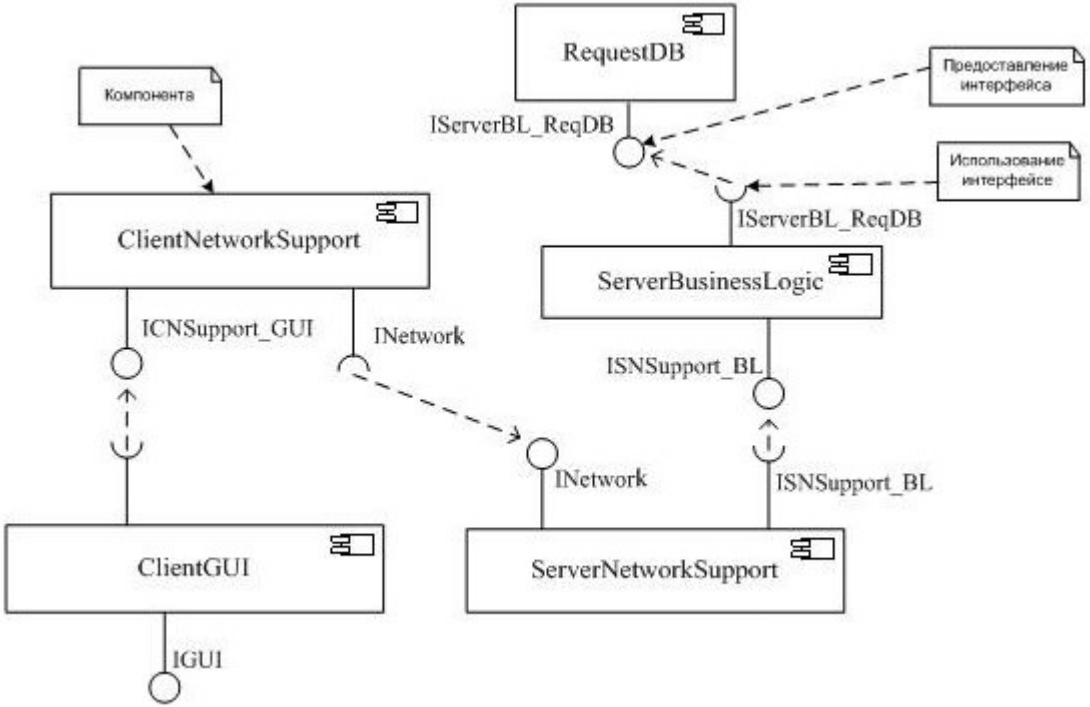


Можно перед именем *компонента* указать текст.

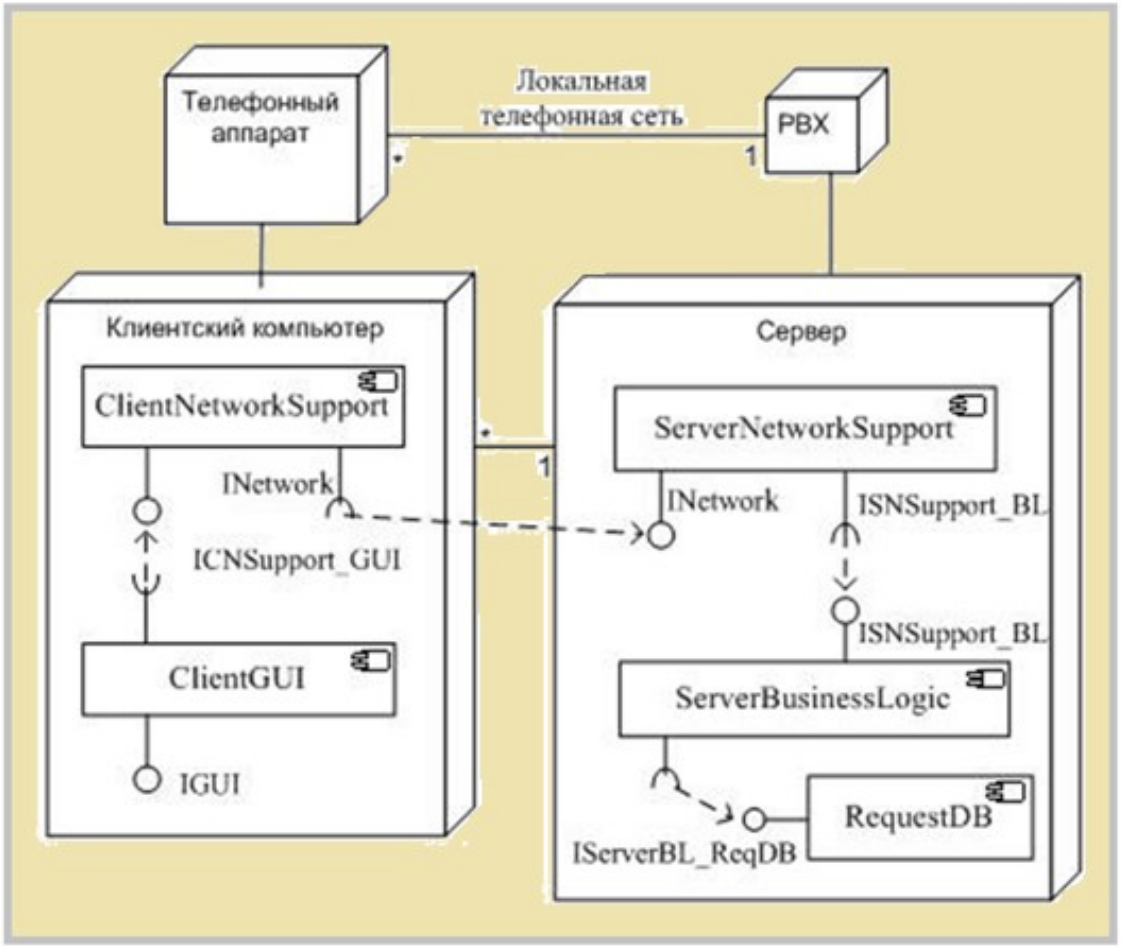
На диаграмме *компонентов* можно отображать *интерфейсы*.



Пример:



Примеры диаграмм развертывания



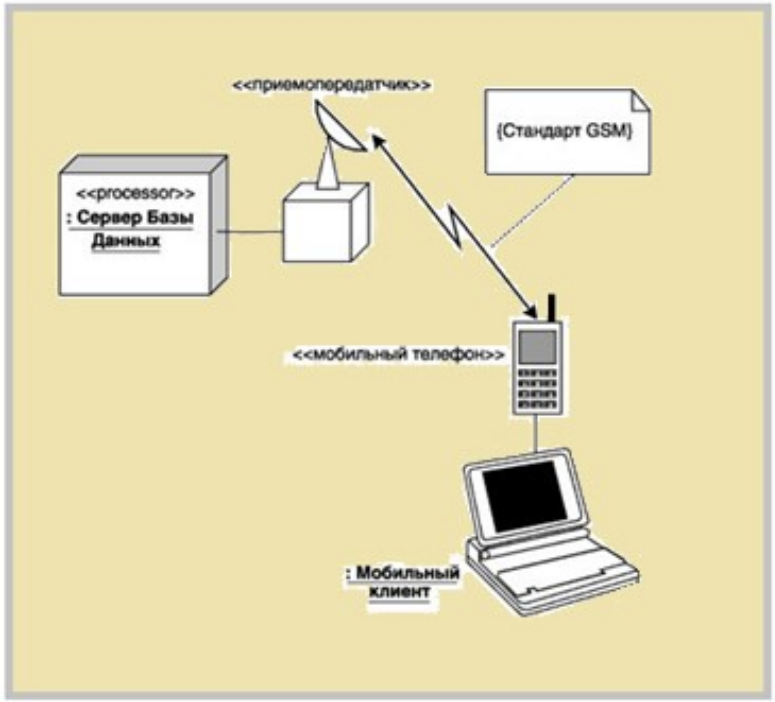
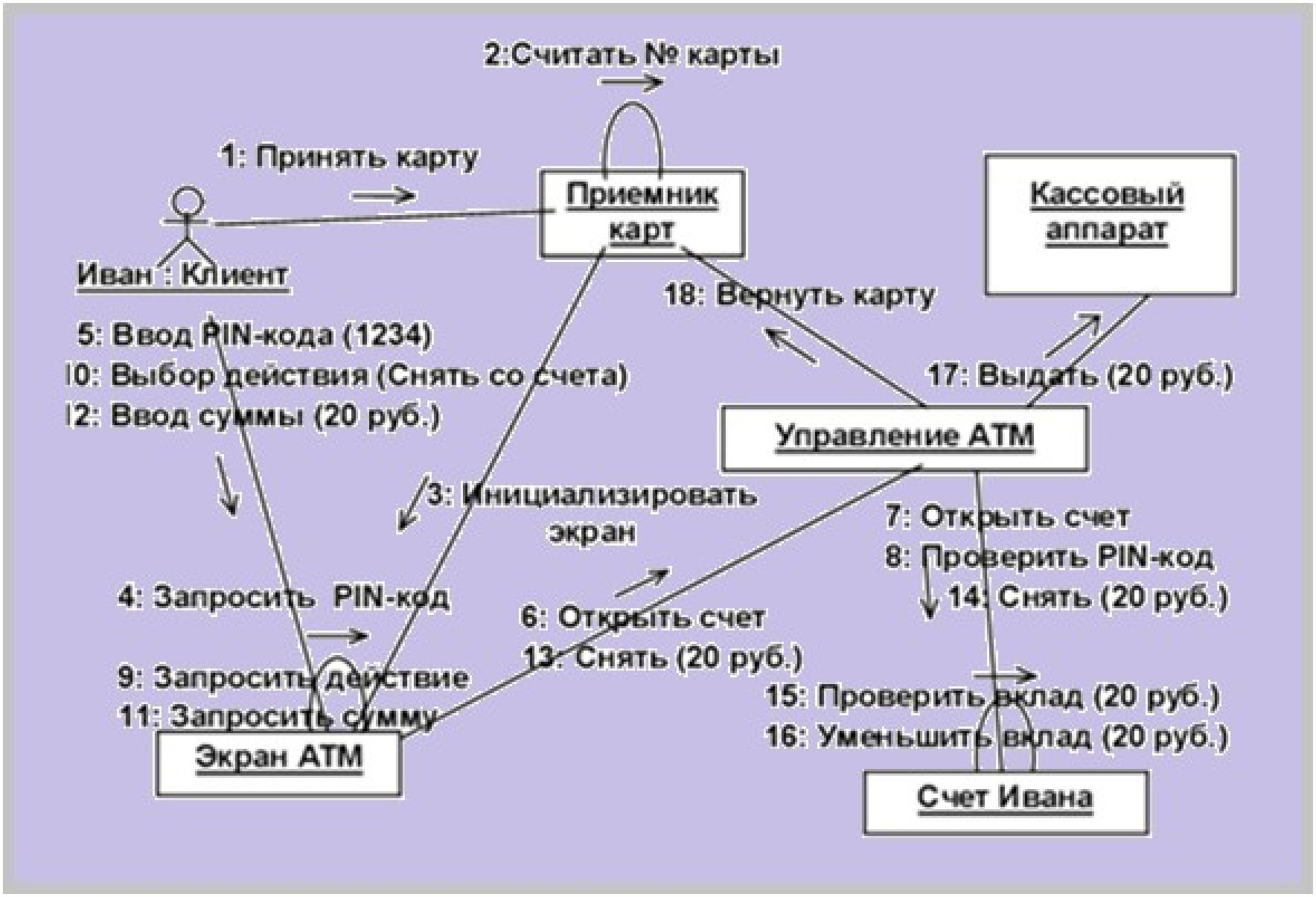


Диаграмма кооперации (взаимодействия см. ранее)

- ✓ Диаграммы кооперации **отображают поток событий** через конкретный сценарий варианта использования.

- ✓ По информативности **пересекается** с диаграммой последовательности (так же стрелки обозначают сообщения, обмен которыми осуществляется в рамках варианта использования)

Временная последовательность сообщений указывается **путем нумерации**.


2:Считать № карты
1: Принять карту
Приемник карт
Кассовый аппарат
Иван : Клиент
18: Вернуть карту
5: Ввод PIN-кода (1234)
10: Выбор действия (Снять со счета)
12: Ввод суммы (20 руб.)
17: Выдать (20 руб.)
Управление ATM
3: Инициализировать экран
7: Открыть счет
8: Проверить PIN-код
14: Снять (20 руб.)
4: Запросить PIN-код
6: Открыть счет
13: Снять (20 руб.)
9: Запросить действие
11: Запросить сумму
15: Проверить вклад (20 руб.)
16: Уменьшить вклад (20 руб.)
Экран ATM
Счет Ивана