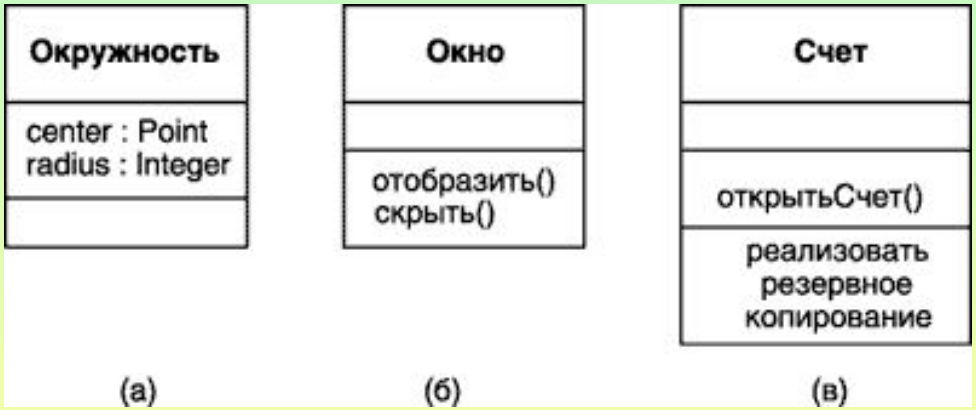
Диаграмма классов

Диаграмма *классов* предназначена для представления статической структуры модели системы в терминологии *классов* объектно-ориентированного программирования.

Варианты графического изображения класса на диаграмме классов



Примеры графического изображения конкретных классов

Класс может иметь или не иметь экземпляров или объектов. В зависимости от этого в языке UML различают *конкретные* и *абстрактные классы*.

Конкретный класс (concrete class) — *класс*, на основе которого могут быть непосредственно созданы экземпляры или объекты.

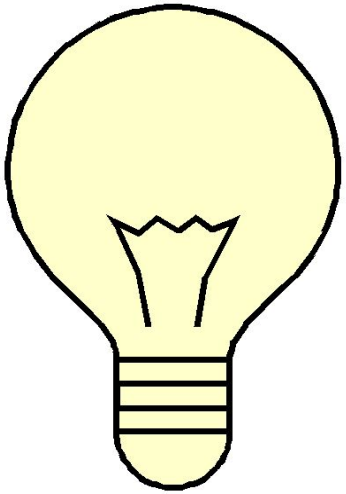
Абстрактный класс (abstract class) — *класс*, который не имеет экземпляров или объектов.

Обозначения на диаграммах классов

Формат строки *имени класса*:

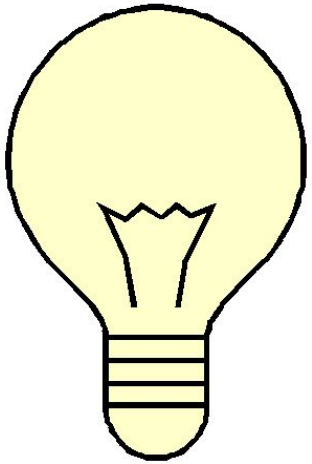
<Имя пакета>::<*Имя класса*>

Приклад абстракції



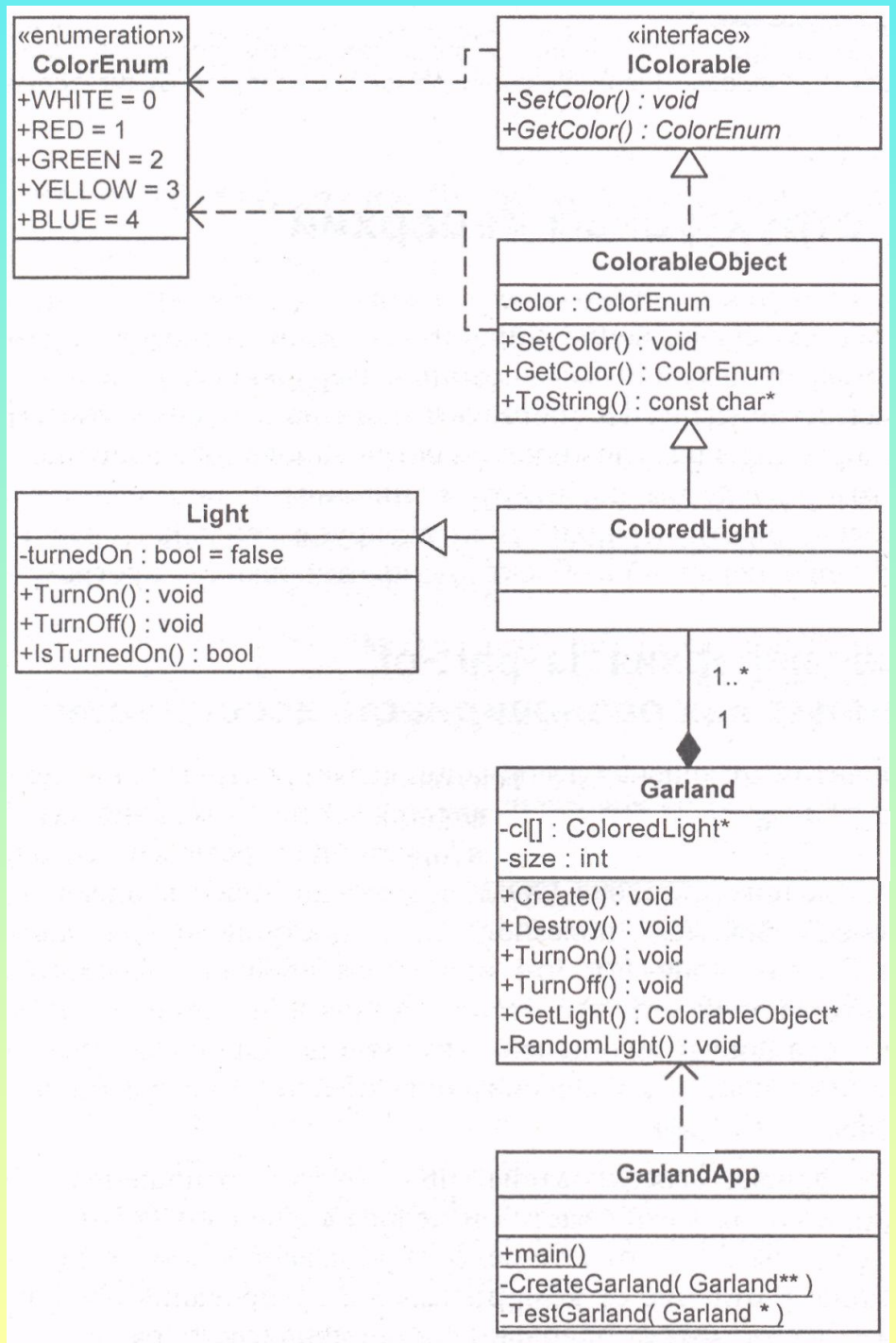
Лампа
-вкл : булен = false
+Включити() +Виключити() +Перевірити() : булен

Рис.2.1. Абстракція електричної лампи



Light
-turnedOn:bool = false
+TurnOn():void +TurnOff():void +IsTurnedOn():bool

Ієрархія класів проекту "Гірлянда"



Обозначения на диаграммах классов

Формат записи ***атрибута*** *класса:*

<квантор видимости> <имя атрибута> [кратность] : <тип атрибута> = <исходное значение> {строка-свойство}.

Квантор видимости

- "+" – обозначает *атрибут* с областью видимости типа **общедоступный** (public). *Атрибут* с этой областью видимости доступен или виден из любого другого *класса пакета*, в котором определена диаграмма.

- "#" – обозначает *атрибут* с областью видимости типа **защищенный** (protected). *Атрибут* с этой областью видимости недоступен или невиден для всех *классов*, за исключением *подклассов* данного *класса*.

- "-" – обозначает *атрибут* с областью видимости типа **закрытый** (private). *Атрибут* с этой областью видимости недоступен или невиден для всех *классов* без исключения.

- "~" – обозначает *атрибут* с областью видимости типа **пакетный** (package). *Атрибут* с этой областью видимости недоступен или невиден для всех *классов* за пределами пакета, в котором определен *класс*-владелец данного *атрибута*.

Обозначения на диаграммах классов

Формат записи *операции (**метода**) класса:*

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

- "+" – обозначает *метод* с областью видимости типа **общедоступный** (public). *Метод* с этой областью видимости доступен или виден из любого другого *класса* пакета, в котором определена диаграмма.
- "#" – обозначает *метод* с областью видимости типа **защищенный** (protected). *Метод* с этой областью видимости недоступен или невиден для всех *классов*, за исключением подклассов данного *класса*.
- "-" – обозначает *метод* с областью видимости типа **закрытый** (private). *Метод* с этой областью видимости недоступен или невиден для всех *классов* без исключения.
- "~" - обозначает *метод* с областью видимости типа **пакетный** (package). *Метод* с этой областью видимости недоступен или невиден для всех *классов* за пределами пакета, в котором определен *класс*-владелец данного *атрибута*.

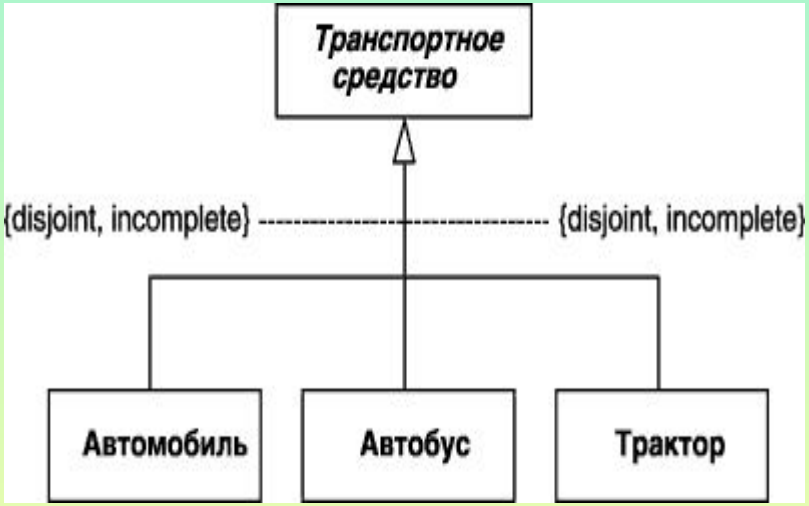
Формат записи параметров:

<направление параметра: *in, out, inout* > <имя параметра>: <выражение типа> = <значение параметра по умолчанию>

Диаграмма классов
Отношения классов

Обобщение (generalization) - отношение между более общим понятием и менее общим понятием.

Родитель, предок (parent) - в отношении обобщения более общий элемент. **Потомок** (child) - специализация одного из элементов отношения обобщения, называемого в этом случае родителем.



Ограничения

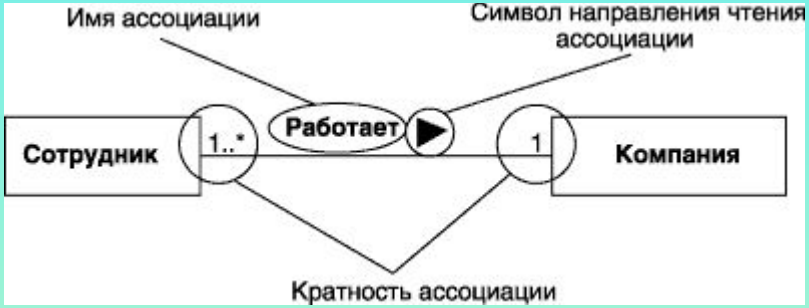
{**complete**} - в данном отношении *обобщения* специфицированы все классы-потомки, и других классов-потомков у данного класса-предка быть не может.

{**incomplete**} - означает, что на диаграмме указаны не все классы-потомки.

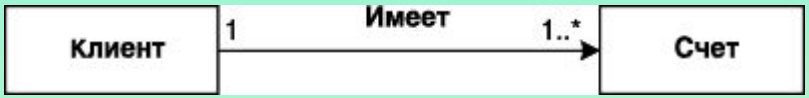
{**disjoint**} - означает, что классы-потомки не могут содержать объектов, одновременно являющихся экземплярами двух или более классов.

{**overlapping**} - предполагается, что отдельные экземпляры классов-потомков могут принадлежать одновременно нескольким классам.

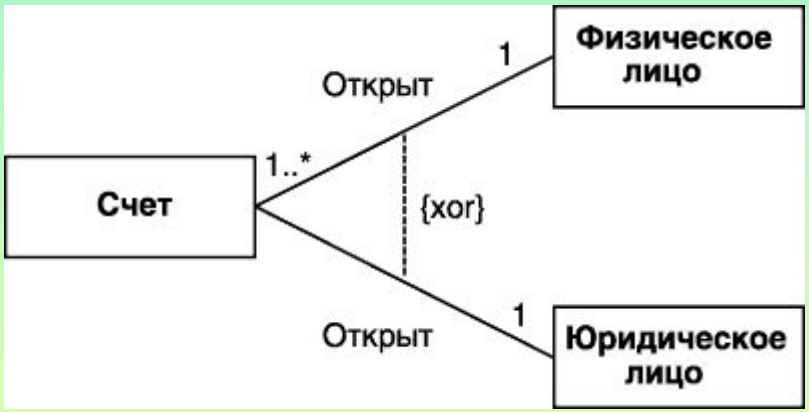
Диаграмма классов
Отношения классов



Графическое изображение *ненаправленной* бинарной ***ассоциации*** между классами



Графическое изображение *направленной* бинарной ***ассоциации*** между классами



Графическое изображение исключающей ***ассоциации*** между тремя классами

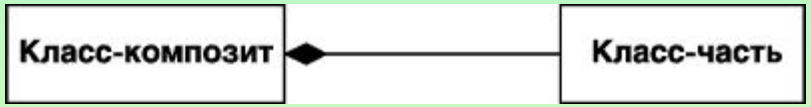
Графическое изображение тернарной ***ассоциации*** между тремя классами

Диаграмма классов
Отношения классов

Агрегация

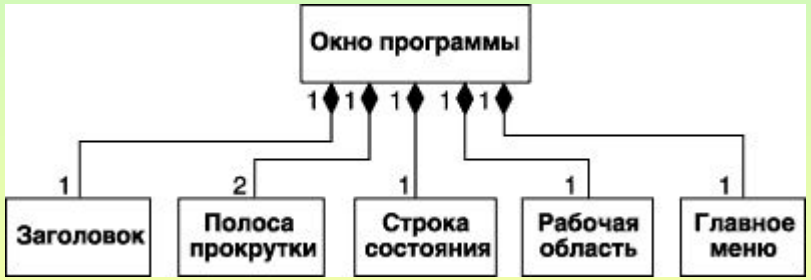
Агрегация (aggregation) - специальная форма **ассоциации**, которая служит для представления отношения типа "часть-целое" между агрегатом (целое) и его составной частью.

Отношение **агрегации** на примере системного блока ПК.



Композиция

Композиция (composition) - разновидность отношения **агрегации**, при которой составные части целого имеют такое же время жизни, что и само целое. Эти части уничтожаются вместе с уничтожением целого.

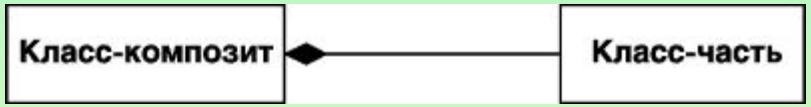


Отношение **композиции** на примере класса-композита Окно программы

Диаграмма классов
Отношения классов

Агрегация

Диаграмма классов для иллюстрации отношения ***агрегации*** на примере системного блока ПК



Композиция

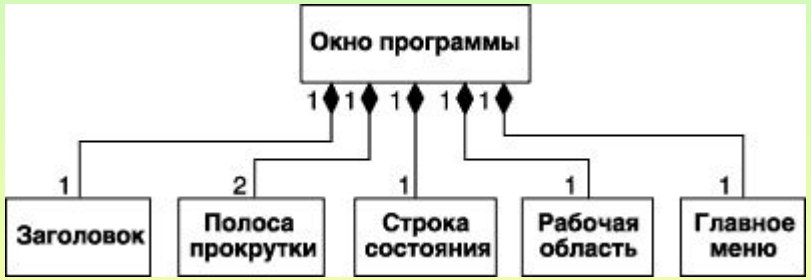
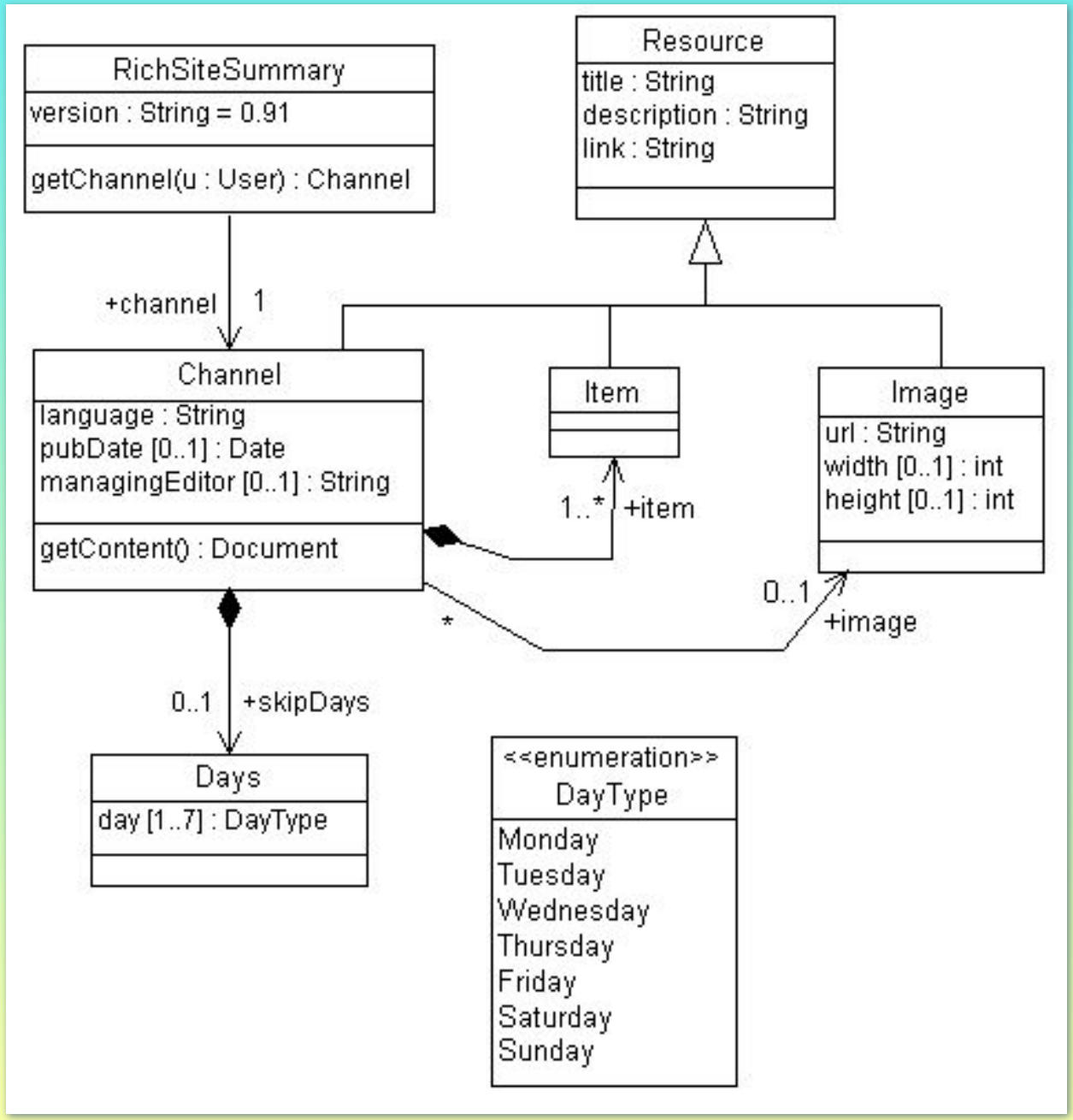
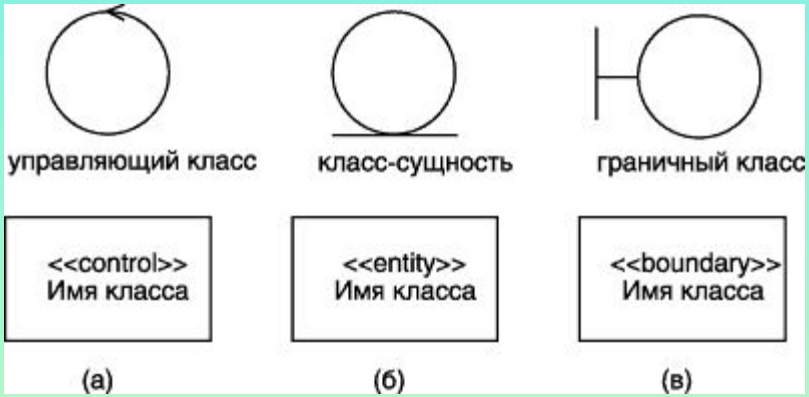


Диаграмма классов для иллюстрации отношения ***композиции*** на примере класса-композита Окно программы


RichSiteSummary
version : String = 0.91
getChannel(u : User) : Channel
Resource
title : String
description : String
link : String
+channel
1
Channel
language : String
pubDate [0..1] : Date
managingEditor [0..1] : String
getContent() : Document
Item
1..*
+item
Image
url : String
width [0..1] : int
height [0..1] : int
0..1
+image
*
0..1
+skipDays
Days
day [1..7] : DayType
<<enumeration>>
DayType
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday

Расширение языка UML для построения моделей программного обеспечения

Графические ***примитивы*** для уточнения семантики классов:



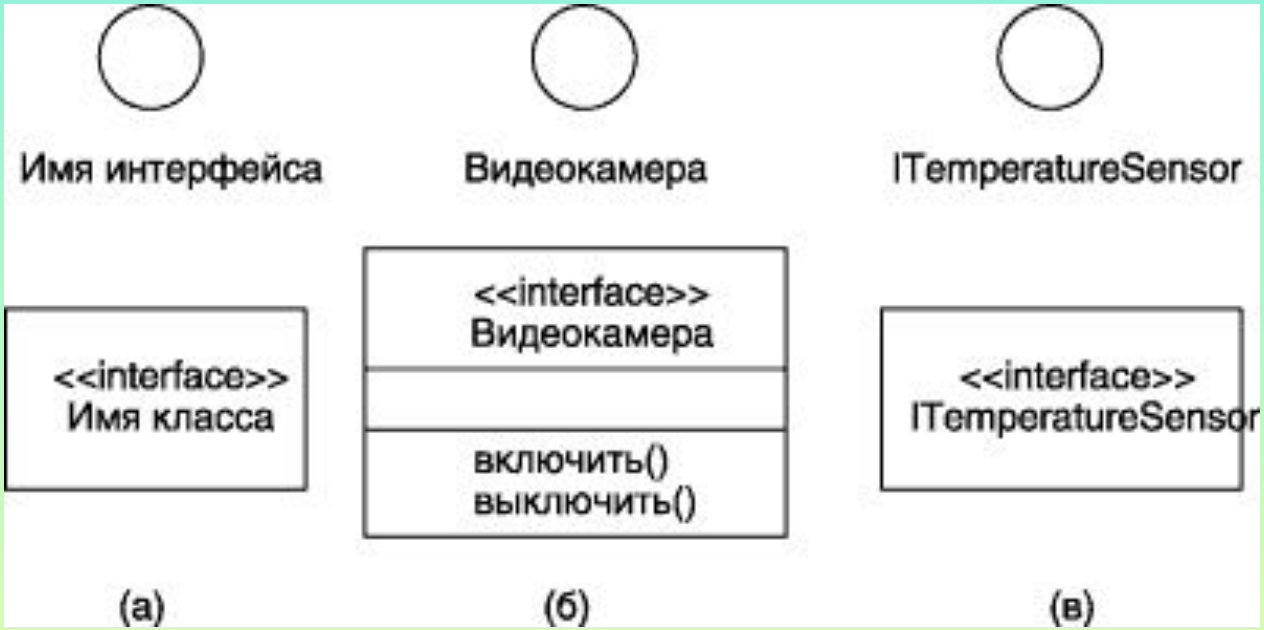
Управляющий класс (control class) — активный *класс*, отвечающий за координацию действий других *классов*. На каждой диаграмме *классов* должен быть хотя бы один управляющий *класс.*Управляющий *класс* может быть изображен в форме прямоугольника *класса* со стереотипом <<control>>

Класс-сущность (entity class) — пассивный класс, информация о котором должна храниться постоянно и не уничтожаться с выключением системы. Как правило, этот класс соответствует отдельной таблице базы данных. Класс-сущность может быть изображен также стандартным образом в форме прямоугольника класса со стереотипом <<entity>>

Граничный класс (boundary class) — класс, который располагается на границе системы с внешней средой и непосредственно взаимодействует с актерами, но является составной частью системы. Граничный класс может быть изображен также стандартным образом в форме прямоугольника класса со стереотипом <<boundary>>

Интерфейс

Интерфейс (interface) — именованное множество *операций*, которые характеризуют поведение отдельного элемента модели. Интерфейс является специальным случаем класса, у которого *имеются операции*, но *отсутствуют атрибуты*.



Интерфейсы на диаграмме служат для спецификации таких элементов модели, которые видимы извне, но их внутренняя структура остается скрытой от клиентов. Интерфейсы не могут содержать ни *атрибутов*, ни состояний, ни направленных ассоциаций. Они содержат только *операции* без указания особенностей их реализации.

Діаграма кооперації

Диаграмма *кооперации* предназначена для описания поведения системы на уровне отдельных ***объектов***, которые обмениваются между собой ***сообщениями***, чтобы достичь нужной цели или реализовать некоторый ***вариант использования***.

Кооперация (collaboration) — спецификация множества *объектов* отдельных классов, совместно взаимодействующих с целью реализации отдельных ***вариантов использования*** в общем контексте моделируемой системы.

На диаграмме *кооперации* размещаются:

1) *объекты*, представляющие собой экземпляры классов;
2) *связи* между ними, которые в свою очередь являются экземплярами *ассоциаций*;
3) *сообщения*.

Объекты и их графическое изображение на диаграммах кооперации

Объект (object) — сущность с хорошо определенными границами и индивидуальностью, которая инкапсулирует состояние и поведение.

Полное имя *объекта* представляет собой строку текста в формате:

<собственное имя объекта >'/'<Имя роли класса>:<Имя класса >

Варианты возможных записей полного имени *объекта*:

o : C – *объект* с собственным именем o, экземпляр класса C.

: C – анонимный объект, экземпляр класса C.

o :(или просто o) — объект-сирота с собственным именем o.

o / R : C — объект с собственным именем o, экземпляр класса C, играющий роль R.

/ R : C — **анонимный** объект, экземпляр класса C, играющий роль R.

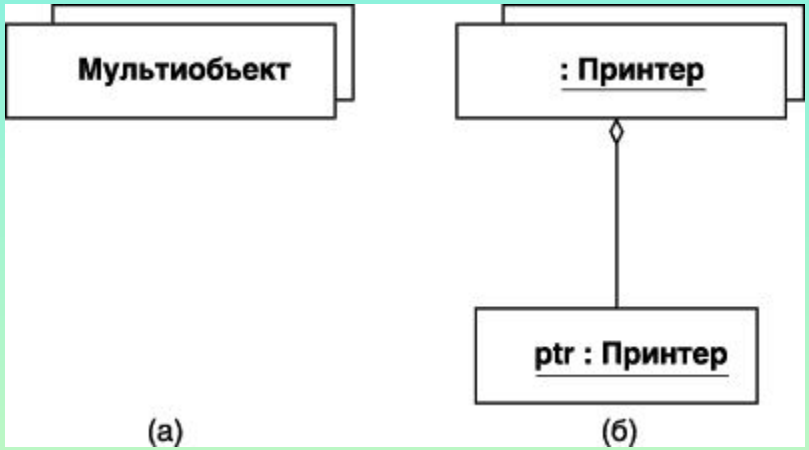
o / R — **объект-сирота** с собственным именем o, играющий роль R.

/ R — **анонимный** объект и одновременно объект-сирота, играющий роль R.

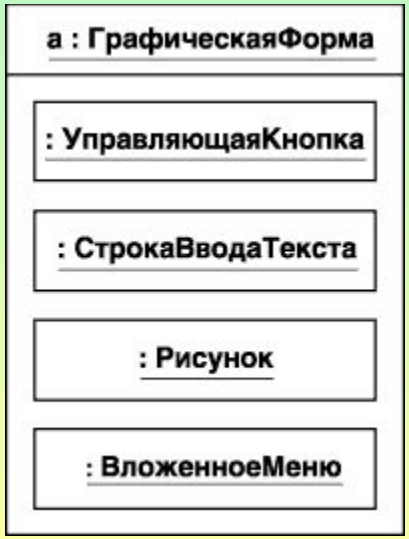
Объекты и их графическое изображение на диаграммах кооперации (2)

Активный объект (active object) имеет собственный процесс управления и может инициировать деятельность по управлению другими *объектами*.

Мультиобъект (multiobject) представляет собой множество анонимных *объектов*, которые могут быть образованы на основе одного класса.



Составной объект (composite object) или объект-композит предназначен для представления *объекта*, имеющего собственную структуру и внутренние потоки (нити) управления.



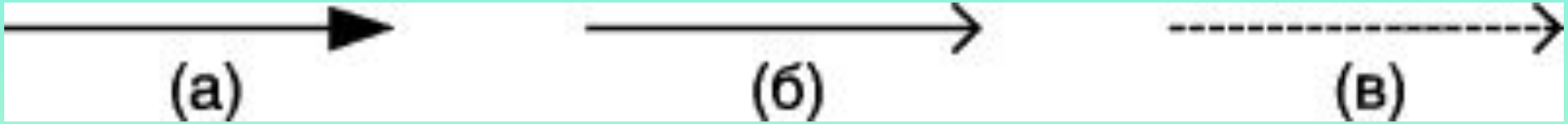
Связи на диаграмме кооперации

Связь (link) — любое семантическое отношение между некоторой совокупностью *объектов*.

Связи не имеют собственных имен, поскольку идентичны как экземпляры некоторой ***ассоциации***. Другими словами, все *связи* на диаграмме *кооперации* могут быть только анонимными и при необходимости записываются без двоеточия перед именем ассоциации. Однако *чаще всего имена связей* на диаграммах *кооперации* не указываются. Для *связей* не указывается также и кратность концевых точек. Однако другие обозначения специальных случаев отношений, такие как агрегация и композиция, могут присутствовать на отдельных концах *связей*.

Сообщения и их графическое изображение

Сообщение (message) — спецификация передачи информации от одного элемента модели к другому с ожиданием выполнения определенных действий со стороны принимающего элемента.



1) ***Сплошная линия с треугольной стрелкой*** (рис. а) обозначает вызов процедуры (операции) или передачу потока управления. *Сообщения* этого типа могут быть использованы, когда один из них передает *сообщение* этого типа и *ожидает*, пока не закончится некоторая последовательность действий, выполняемая вторым *объектом*.
2) ***Сплошная линия с V-образной стрелкой*** (рис.б) обозначает асинхронное *сообщение* в простом потоке управления. В этом случае клиент передает асинхронное *сообщение* и продолжает выполнять свою деятельность, *не ожидая* ответа от клиента.
3) ***Пунктирная линия с V-образной стрелкой*** (рис. в) обозначает *возврат* из вызова процедуры.

Каждое *сообщение* может быть помечено строкой текста, которая имеет следующий формат:
<Предшествующие сообщения> '/' <Выражение последовательности> ':'
<Возвращаемое значение := имя сообщения> <(Список аргументов)>

Предшествующие *сообщения* — это разделенные запятыми номера *сообщений*, записанные перед наклонной чертой: **<Номер *сообщения*',')* <'/'>**.

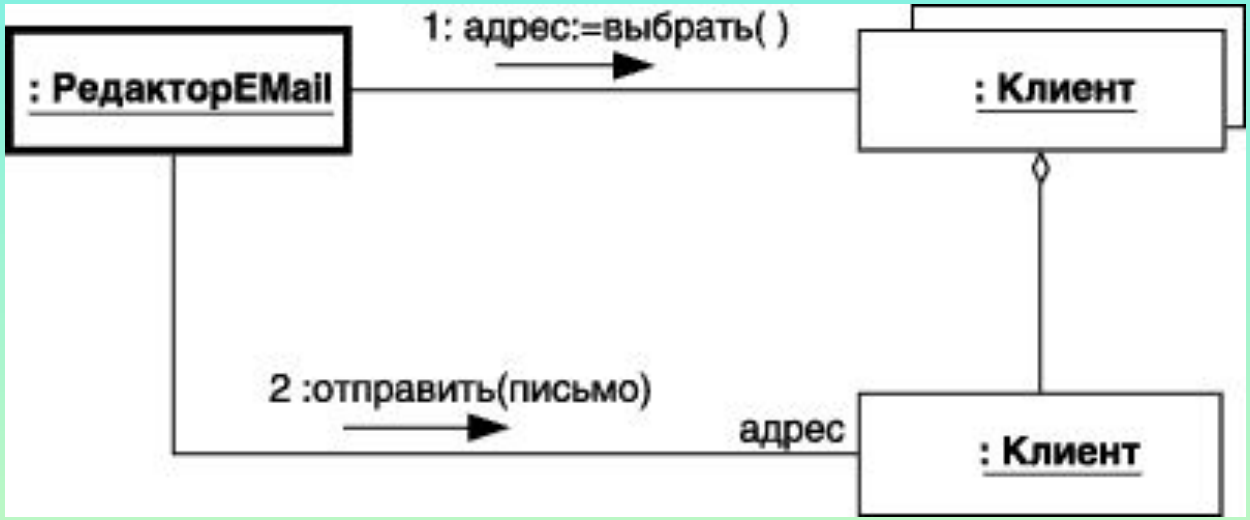
Выражение последовательности — это разделенный точками список отдельных термов последовательностей, после которого записывается **двоеточие**:
<Терм последовательности'.'…> ':'

Каждый терм последовательности имеет следующий синтаксис:
[Целое число | Имя] [Рекуррентность].

имя *сообщения*, записанное в сигнатуре после возвращаемого значения, означает имя события, которое инициируется объектом-получателем *сообщения* после его приема.

Список аргументов представляет собой разделенные запятыми и заключенные в круглые скобки действительные параметры той операции, вызов которой инициируется данным *сообщением*. Список аргументов может быть пустым, однако скобки все равно записываются.

В UML предусмотрены стандартные действия при получении соответствующего *сообщения*. Они могут быть явно указаны на диаграмме *кооперации* в форме ***стереотипа*** перед именем *сообщения*, к которому они относятся, или выше его: <<call>> (вызвать), <<return>> (возвратить), <<create>> (создать), <<destroy>> (уничтожить), <<send>> (послать)



Фрагмент диаграммы кооперации для выбора адреса клиента для отправки электронного письма

Стереотипные сообщения

<<call>> (вызвать) – *сообщение*, требующее вызова операции или процедуры объекта-получателя. Если *сообщение* с этим стереотипом рефлексивное, то оно инициирует локальный вызов операции у пославшего это *сообщение объекта*.

<<return>> (возвратить) – *сообщение*, возвращающее значение выполненной операции или процедуры вызвавшему ее *объекту*. Значение результата может инициировать ветвление потока управления.

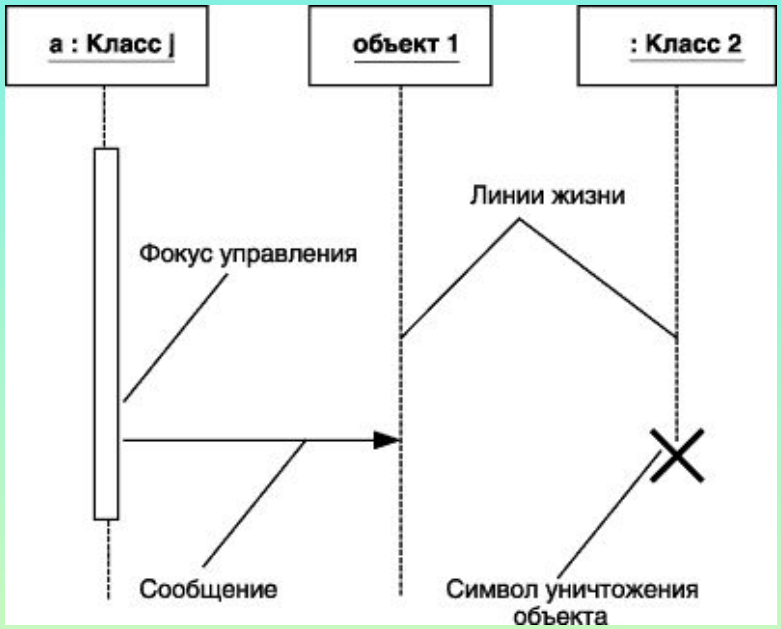
<<create>> (создать) – *сообщение*, требующее создания другого *объекта* для выполнения определенных действий. Созданный *объект* может стать активным (ему передается поток управления), а может остаться пассивным.

<<destroy>> (уничтожить) – *сообщение* с явным требованием уничтожить соответствующий *объект*. Посылается в том случае, когда необходимо прекратить нежелательные действия со стороны существующего в системе *объекта*, либо когда *объект* больше не нужен и должен освободить задействованные им системные ресурсы.

<<send>> (послать) – обозначает посылку другому *объекту* сигнала, который асинхронно инициируется одним *объектом* и принимается (перехватывается) другим. Отличие сигнала от *сообщения* заключается в том, что сигнал должен быть явно описан в том классе, *объект* которого инициирует его передачу.

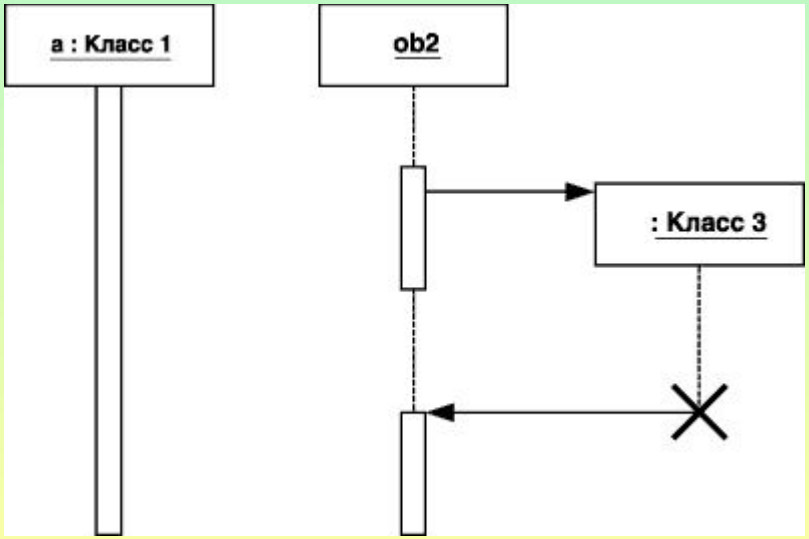
Діаграми послідовності

Диаграмма последовательности* (*sequence diagram*)** - диаграмма, на которой показаны взаимодействия ***объектов, упорядоченные по времени их проявления.



***Линия жизни объекта* (*object lifeline*)** - вертикальная линия, которая представляет существование объекта в течение определенного периода времени.

***Фокус управления* (*focus of control*)** - специальный символ, указывающий период времени, в течение которого объект выполняет некоторое действие, находясь в активном состоянии.



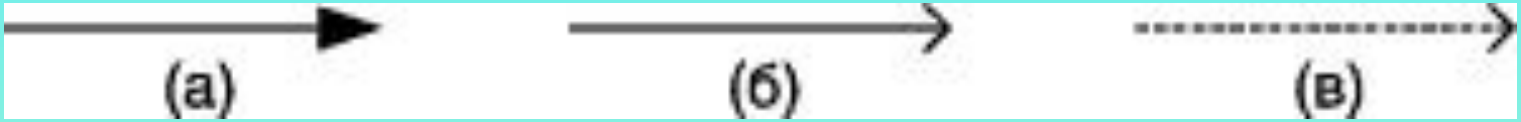
Для обозначения момента уничтожения *объекта* в языке UML применяется специальный символ в форме латинской буквы "X"

Діаграми послідовності (2)

Рефлексивные *сообщения (сообщения самому себе)* изображаются в форме *сообщения*, начало и конец которого соприкасаются с *линией жизни* или *фокусом управления* одного и того же *объекта*

Если в результате рефлексивного *сообщения* создается новый подпроцесс или нить управления, то говорят о **рекурсивном** или вложенном *фокусе управления*.

Сообщения на диаграмме последовательности

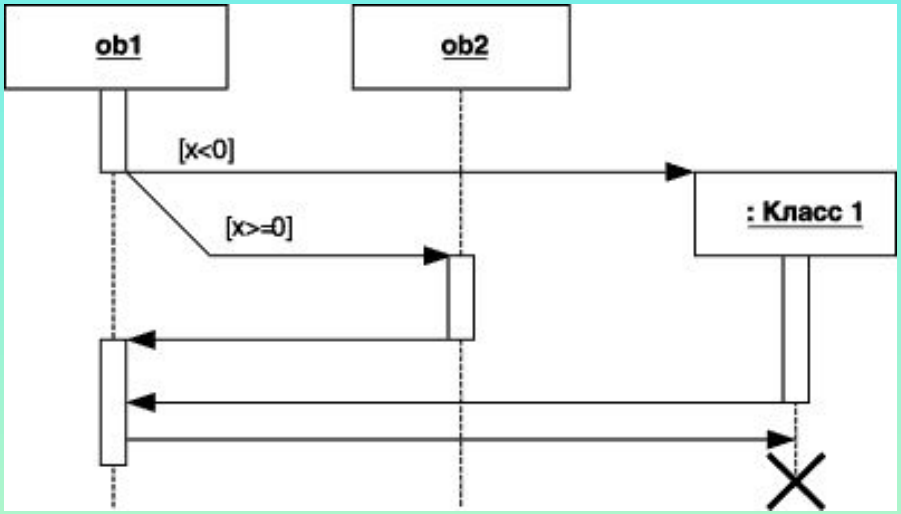


Первая разновидность *сообщения* (рис. а) используется для синхронного вызова процедур, выполнения операций или обозначения отдельных вложенных потоков управления. Принимающий *объект* может получить *фокус управления*, становясь в этом случае активным. Передающий *объект* может потерять *фокус управления* или остаться активным.

Вторая разновидность *сообщения* (рис.б) используется для обозначения простого асинхронного *сообщения*, которое передается в произвольный момент времени. Передача такого *сообщения* не сопровождается получением *фокуса управления* объектом-получателем.

Третья разновидность *сообщения* (рис. в) используется для возврата из вызова процедуры.

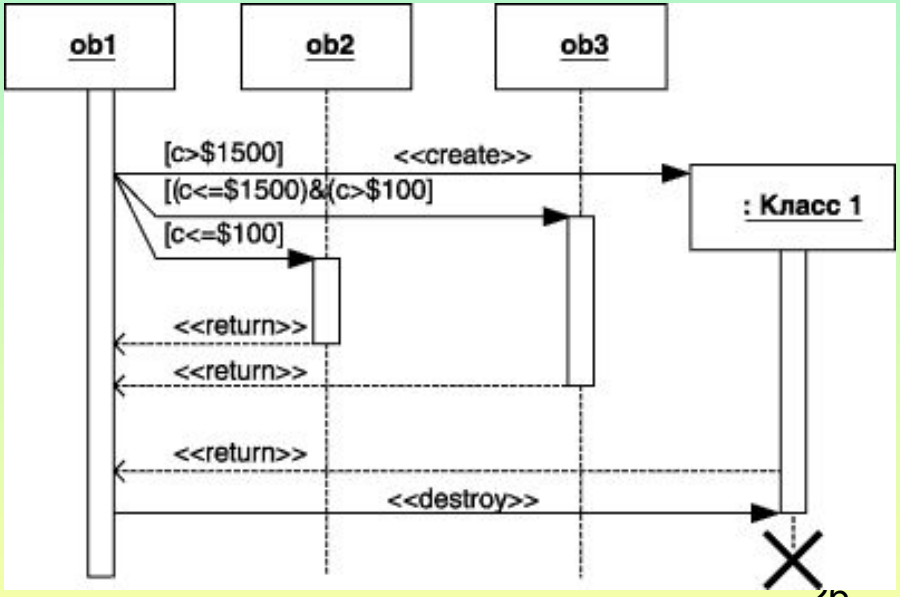
Ветвление потока управления



Для изображения *ветвления* используются две или более стрелки, выходящие из одной точки *фокуса управления объекта*. При этом рядом с каждой из них должно быть указано соответствующее ***условие ветви*** в форме ***булевского выражения***.

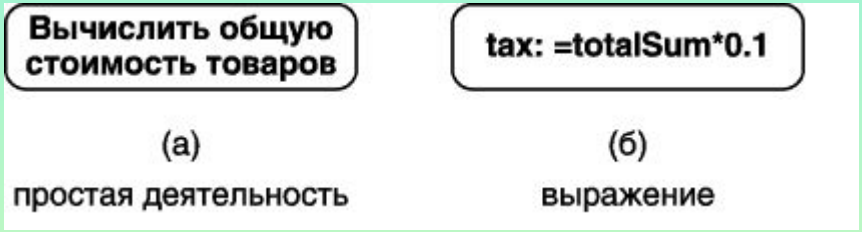
На *диаграммах последовательности* при записи *сообщений* также могут использоваться ***стереотипы***, рассмотренные ранее при построении диаграммы кооперации

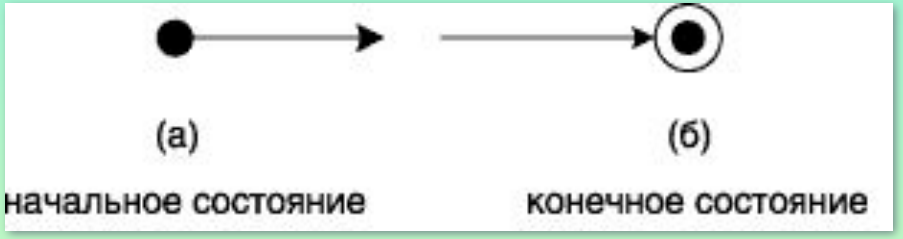
С*ообщения* могут иметь собственное имя, в качестве которого выступает имя операции, вызов которой инициируют эти *сообщения* у принимающего *объекта*. В этом случае рядом со стрелкой записывается имя операции с круглыми скобками, в которых могут указываться параметры или аргументы соответствующей операции.



Діаграми діяльності

Диаграмма деятельности предназначена для моделирования поведения систем, хотя время в явном виде отсутствует на этой диаграмме. На ***диаграмме деятельности*** отображается логика или последовательность ***перехода*** от одной ***деятельности*** к другой, при этом внимание фиксируется на результате деятельности. ***Диаграмма деятельности позволяет*** детализировать особенности *алгоритмической* и *процедурной* реализации выполняемых системой операций. Диаграммы деятельности - частный случай диаграмм состояний.



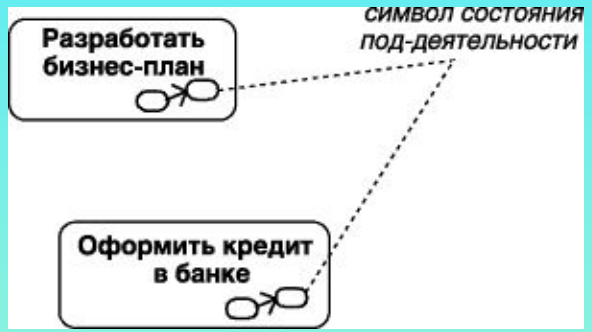


(а) Состояние деятельности (activity state) - состояние в графе деятельности, которое служит для представления *процедурной* последовательности действий, требующих определенного времени.

(б) Состояние действия (action state) - специальный случай состояния с некоторым входным действием и, по крайней мере, одним выходящим из состояния переходом (моделирование ***шага*** выполнения алгоритма или процедуры).

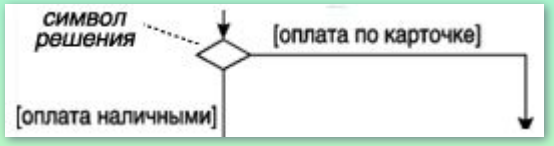
Диаграмма деятельности должна иметь единственное **начальное (а)** и **конечное (б)** состояния. Каждая деятельность начинается в начальном состоянии и заканчивается в конечном состоянии. Диаграмму деятельности принято располагать таким образом, чтобы действия следовали сверху вниз или слева направо.

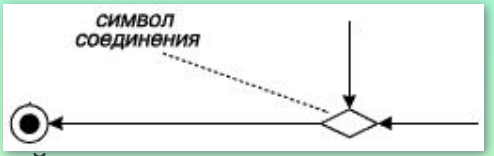
Состояние под-деятельности (subactivity state) - состояние в графе деятельности, которое служит для представления неатомарной последовательности шагов процесса.



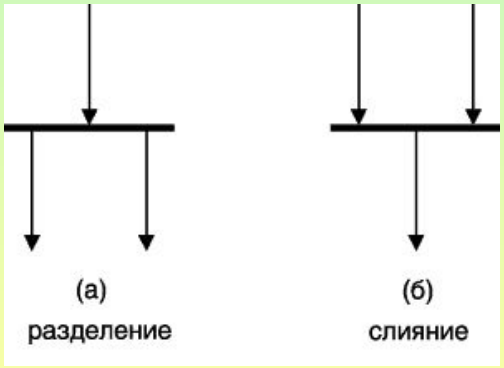
Переход изображается сплошной линией со стрелкой.

Если переходов несколько, то для каждого из таких переходов должно быть явно записано собственное **сторожевое условие в прямых скобках**. Такая ситуация называется ветвлением, а для ее обозначения применяется символ **решения** (decision) в форме небольшого ромба, внутри которого нет никакого текста.



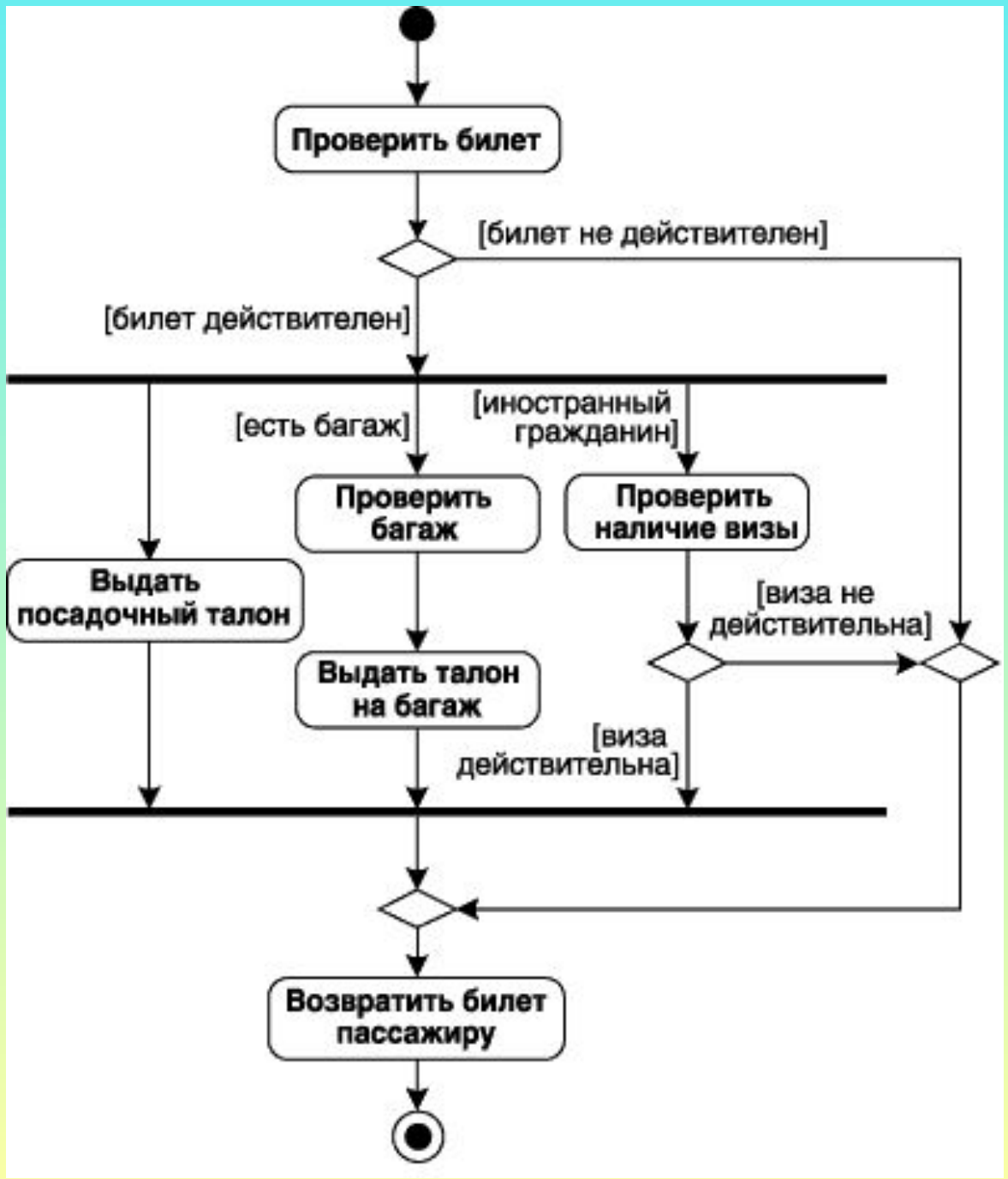


Для объединения альтернативных ветвей также используется символ в форме ромба, который в этом случае называют **соединением** (merge). Входящих стрелок у символа **соединения** (слияние) может быть несколько. Выходить из **соединения** может только одна стрелка.



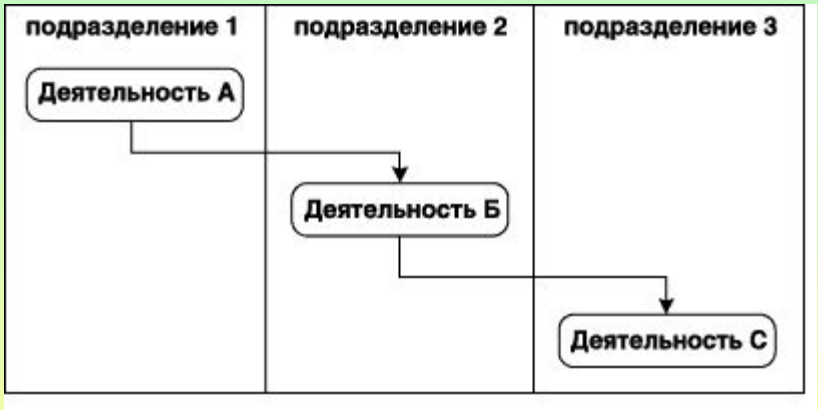
Для изображения разделения и слияния **параллельных** вычислений или потоков управления используется специальный символ - прямая черта.

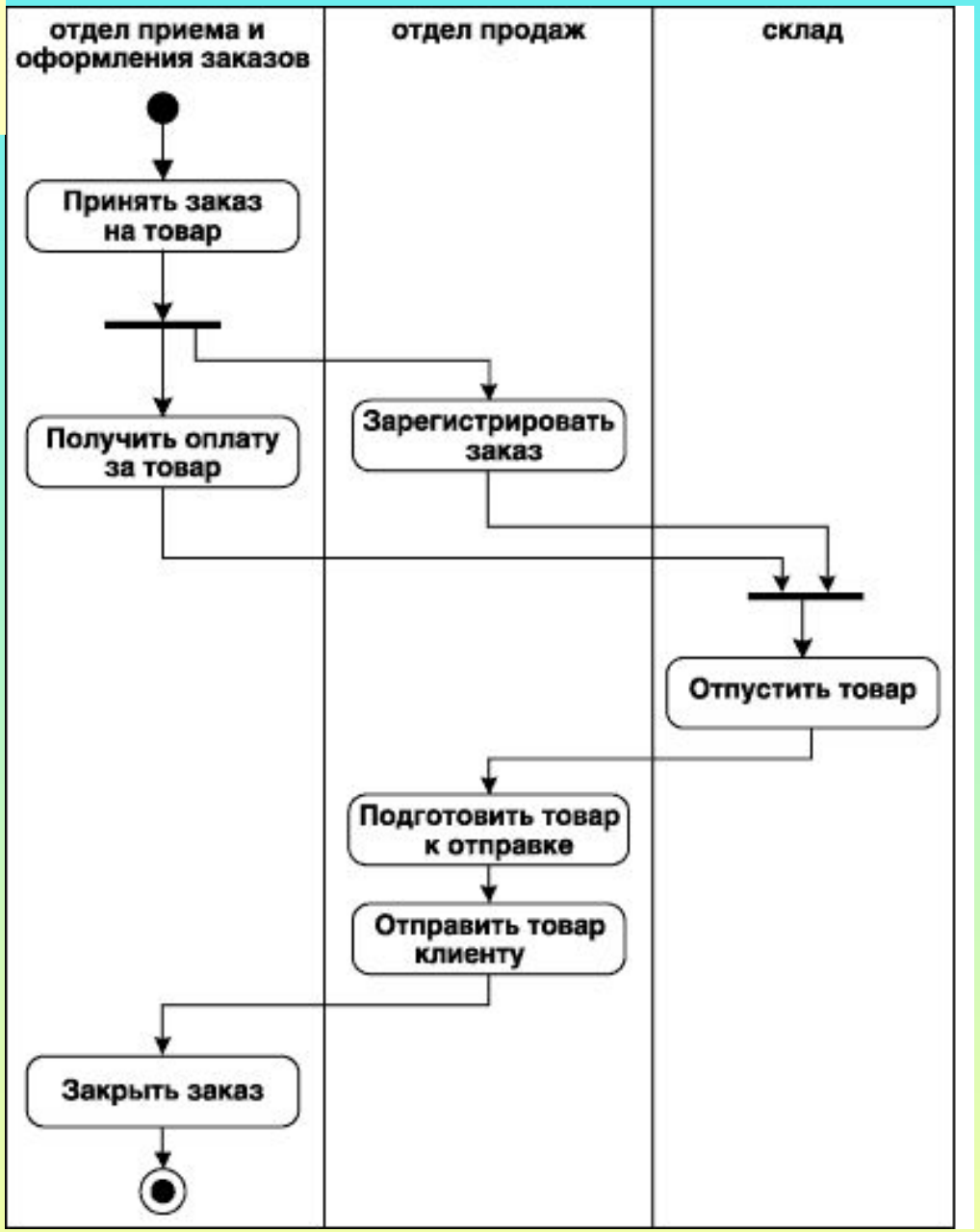

Вычислить общую стоимость товаров
[оплата по карточке]
[оплата наличными]
символ решения
Проверить сумму баланса
Оплатить стоимость товара наличными
[стоимость товаров>баланс]
[стоимость товаров<=баланс]
Оплатить стоимость товара кредитной карточкой
символ соединения


Проверить билет
[билет не действителен]
[билет действителен]
[есть багаж]
[иностранный гражданин]
Проверить багаж
Проверить наличие визы
Выдать посадочный талон
[виза не действительна]
Выдать талон на багаж
[виза действительна]
Возвратить билет пассажиру

Дорожки на диаграмме деятельности

Дорожка (swimlane) - графическая область диаграммы деятельности, содержащая элементы модели, ответственность за выполнение которых принадлежит отдельным подсистемам.





Объекты на диаграмме деятельности

Иногда возникает необходимость явно указать объекты на диаграмме деятельности.
Базовым графическим представлением объекта в нотации языка UML является прямоугольник класса, с тем отличием, что имя объекта подчеркивается. На диаграммах деятельности после имени может указываться характеристика состояния объекта в прямых скобках. Такие прямоугольники объектов присоединяются к состояниям действия ***отношением зависимости*** пунктирной линией со стрелкой.