Методы визуального анализа и проектирования систем

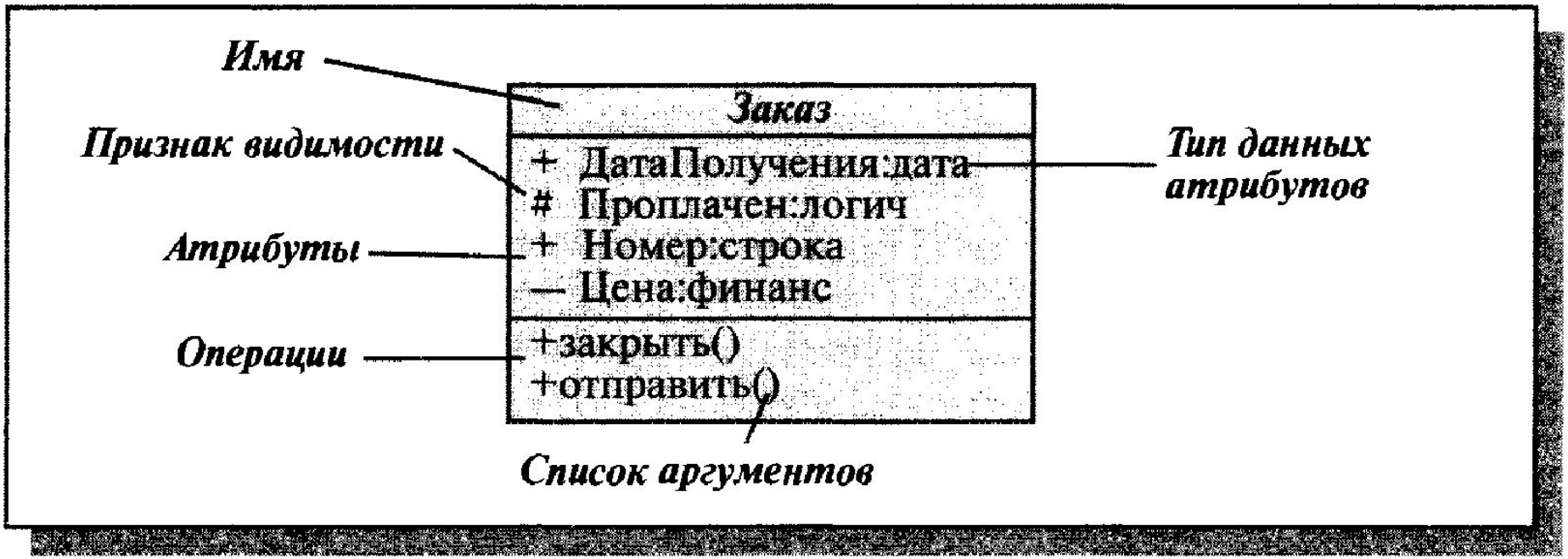
Диаграммы UML

Диаграммы классов

Клевцов С.И. кафедра МПС

Диаграммы классов

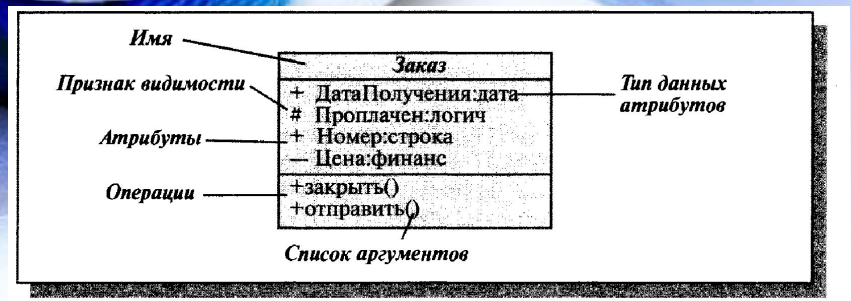
Класс



***Классы** представляют собой описание совокупностей однородных объектов с присущими им свойствами – атрибутами, операциями, отношениями и семантикой.*

Диаграммы классов

Класс



Запись атрибута:

<Признак видимости> <имя атрибута> : <тип данных> = <значение по умолчанию>

Запись операции:

<Признак видимости> <имя операции> <(список аргументов)

Три уровня видимости:

+ общий (public)

 # защищенный (protected)

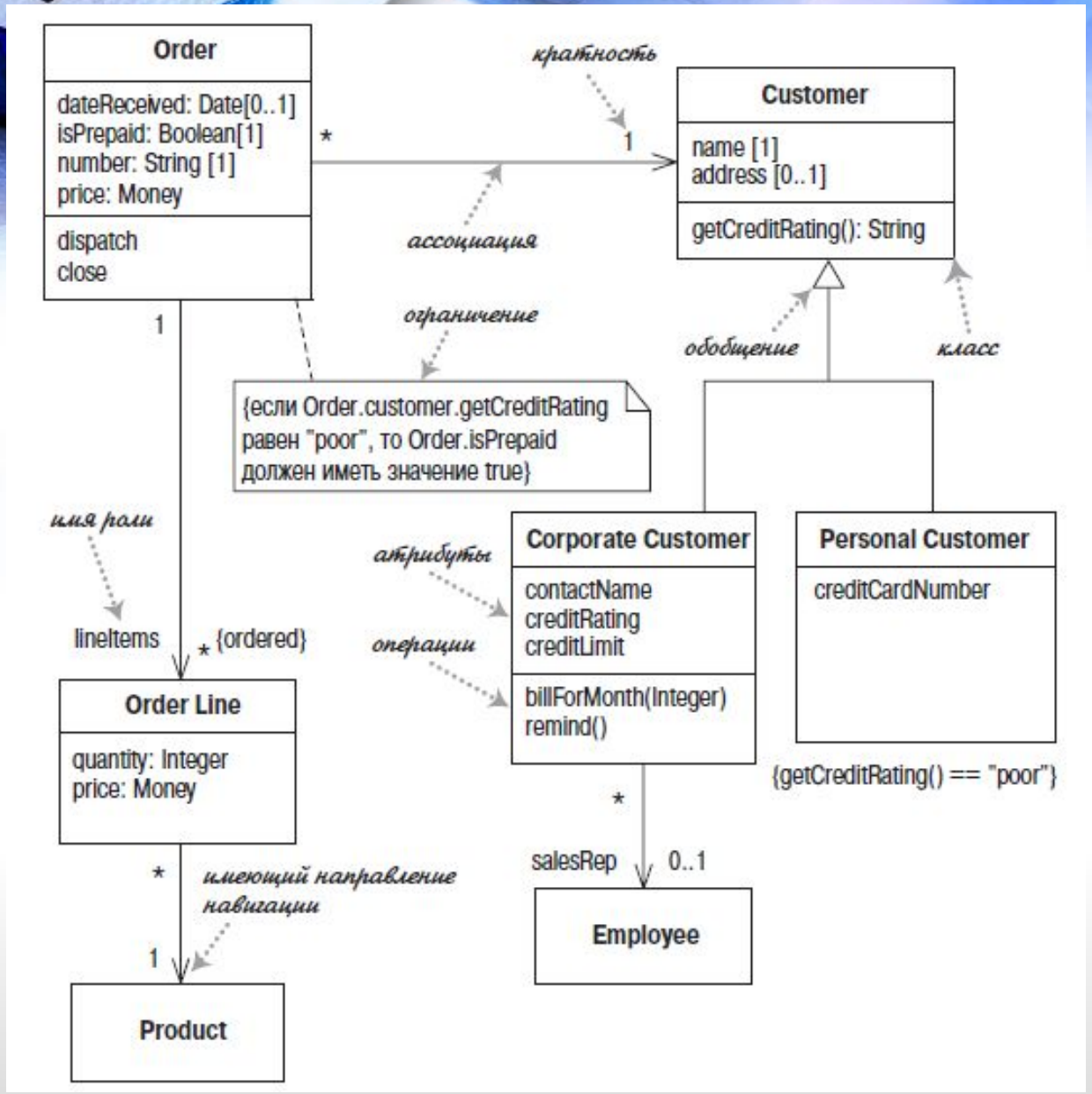
- закрытый (private)

Область действия:

Экземпляр (instance)

Классификатор (classifier)

Диаграммы классов



Диаграммы классов

Атрибуты:

Атрибут описывает свойство в виде строки текста внутри прямоугольника класса.

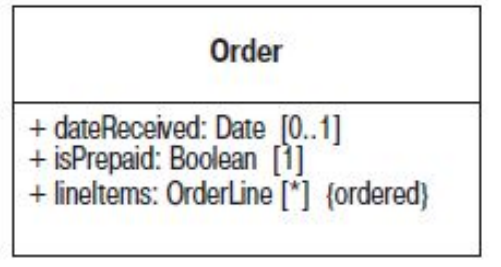
Полная форма атрибута:

видимость имя: тип кратность = значение по умолчанию {строка свойств}

Например:

- ST: String [1] = "Без имени" {readOnly}

Представление свойств в виде атрибутов

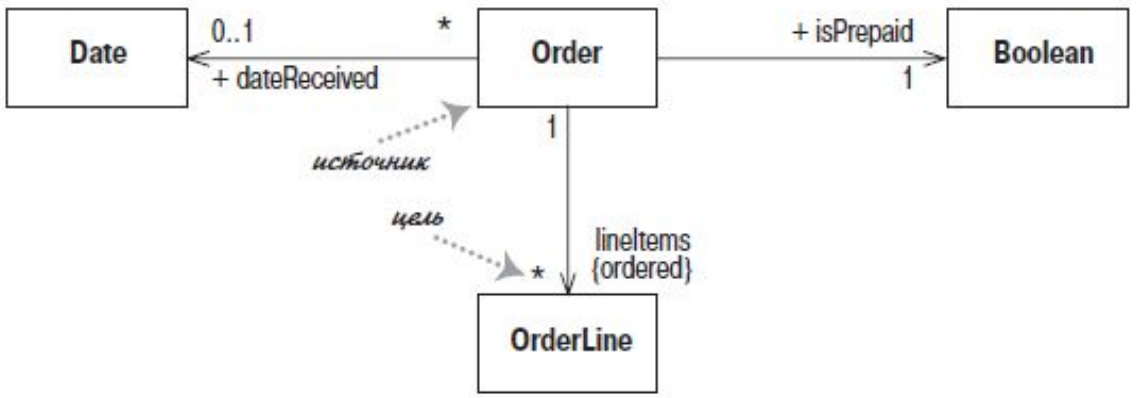


Диаграммы классов

Ассоциации

Ассоциация – это непрерывная линия между двумя классами, направленная от исходного класса к целевому классу.
Имя свойства (вместе с кратностью) располагается на целевом конце ассоциации.
Целевой конец ассоциации указывает на класс, который является типом свойства.

Представление свойств в виде ассоциаций



Диаграммы классов

Ассоциации

Кратность:

1 - только 1
0..1 - 0 или 1
*** - ноль или более**

При рассмотрении атрибутов могут встретиться термины, имеющие отношение к кратности.

- **Optional** – необязательный - предполагает нулевую нижнюю границу.
- **Mandatory** – обязательный - подразумевает, что нижняя граница равна или больше 1.
- **Singlevalued** – однозначный – для такого атрибута верхняя граница равна 1.
- **Multivalued** – многозначный - имеется в виду, что верхняя граница больше 1; обычно *.

Диаграммы классов

Ассоциации

Двунаправленные ассоциации

Двунаправленная ассоциация – это пара свойств, связанных в противоположных направлениях

Использование глагола в имени ассоциации

Диаграммы классов

Операции

Операции (operations) представляют собой действия, реализуемые некоторым классом. Существует соответствие между операциями и методами класса.

Полный синтаксис операций в языке UML выглядит следующим образом:

видимость имя (список параметров) : возвращаемый тип {строка свойств}

Параметры в списке параметров обозначаются таким же образом, что и для атрибутов. Они имеют вид:

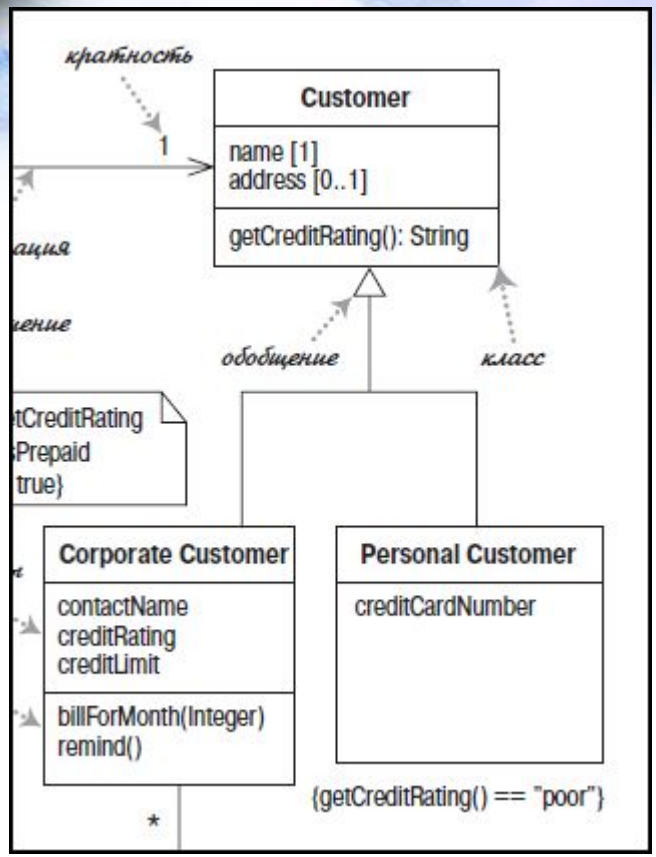
направление имя: тип = значение по умолчанию

Пример операции:

+ balanceOn (date: Date) : Money

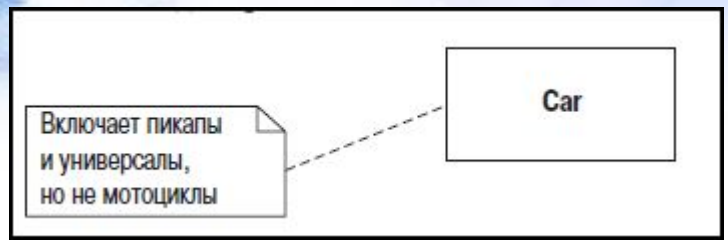
Диаграммы классов

Обобщение



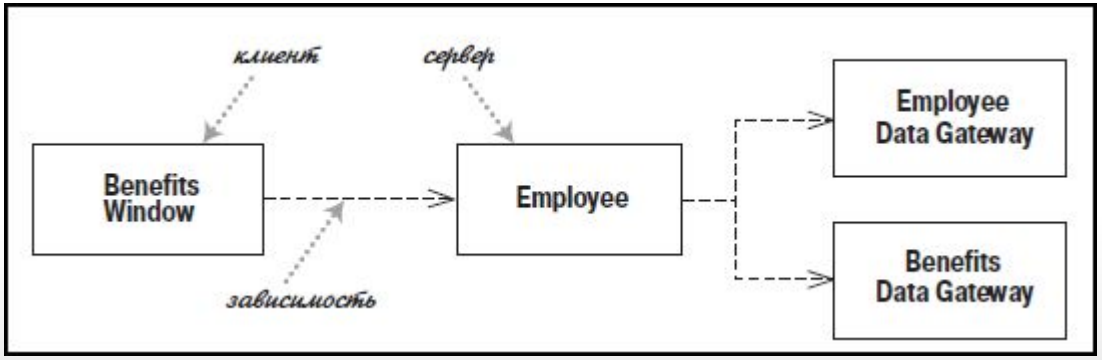
Диаграммы классов

Примечания и комментарии



Зависимость

Считается, что между двумя элементами существует **зависимость** (dependency), если изменения в определении одного элемента (**сервера**) могут вызвать изменения в другом элементе (**клиенте**).



Диаграммы классов

Зависимость

Ключевое слово	Значение
«call» (вызывать)	Источник вызывает операцию в цели
«create» (создавать)	Источник создает экземпляр цели
«derive» (производить)	Источник представляет собой производное цели
«instantiate» (создать экземпляр)	Источник является экземпляром цели. (Обратите внимание, что если источник является классом, то сам класс является экземпляром класса класс; то есть целевой класс – это метакласс)
«permit» (разрешать)	Цель разрешает источнику доступ к ее закрытой функциональности
«realize» (реализовать)	Источник является реализацией спецификации или интерфейса, определенного целью (*стр. 96*)
«refine» (уточнить)	Уточнение означает отношение между различными семантическими уровнями; например, источник может быть классом разработки, а цель – соответствующим классом анализа
«substitute» (заместить)	Источник может быть заменен целью (*стр. 72*)
«trace» (проследить)	Используется, чтобы отследить такие моменты, как требования к классам или как изменения одной ссылки модели влияют на все остальное
«use» (использовать)	Для реализации источника требуется цель

Избранные ключевые слова зависимостей

Диаграммы последовательностей

Диаграммы взаимодействия (interaction diagrams) описывают взаимодействие групп объектов в различных условиях их поведения.

UML определяет диаграммы взаимодействия нескольких типов, из которых наиболее употребительными являются **диаграммы последовательности.**

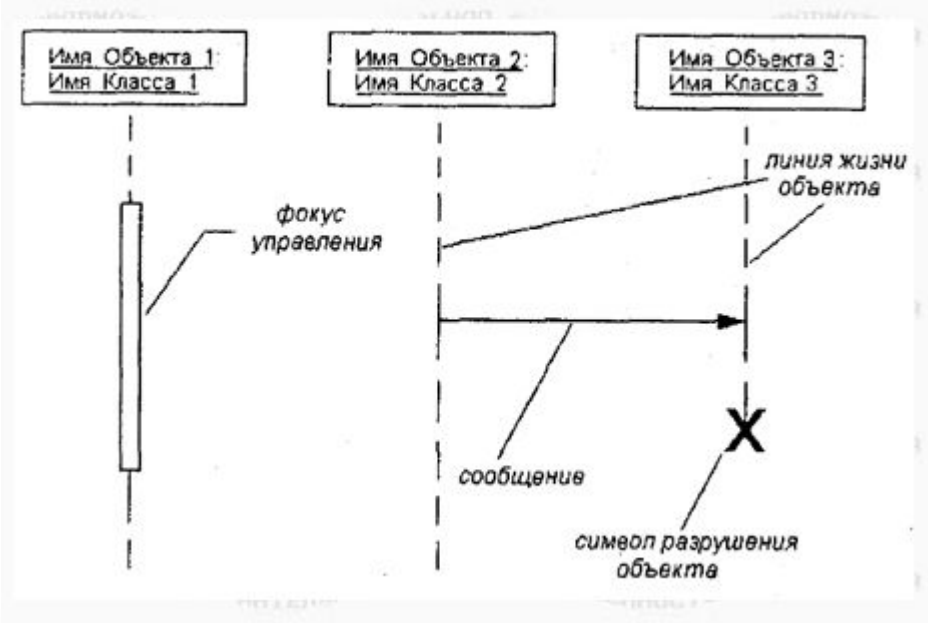
Обычно диаграмма последовательности описывает один сценарий.

На диаграмме показываются экземпляры объектов и сообщения, которыми обмениваются объекты в рамках одного прецедента (use case).

Диаграммы последовательностей

Элементы Диаграммы взаимодействия:

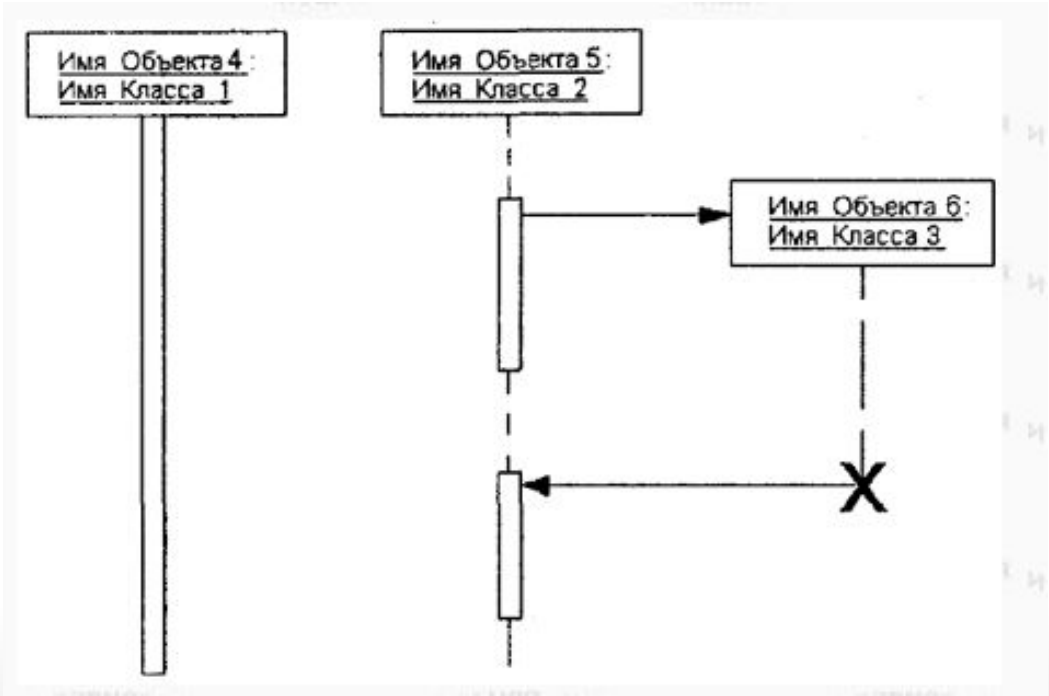
Объекты



Диаграммы последовательностей

Элементы Диаграммы взаимодействия:

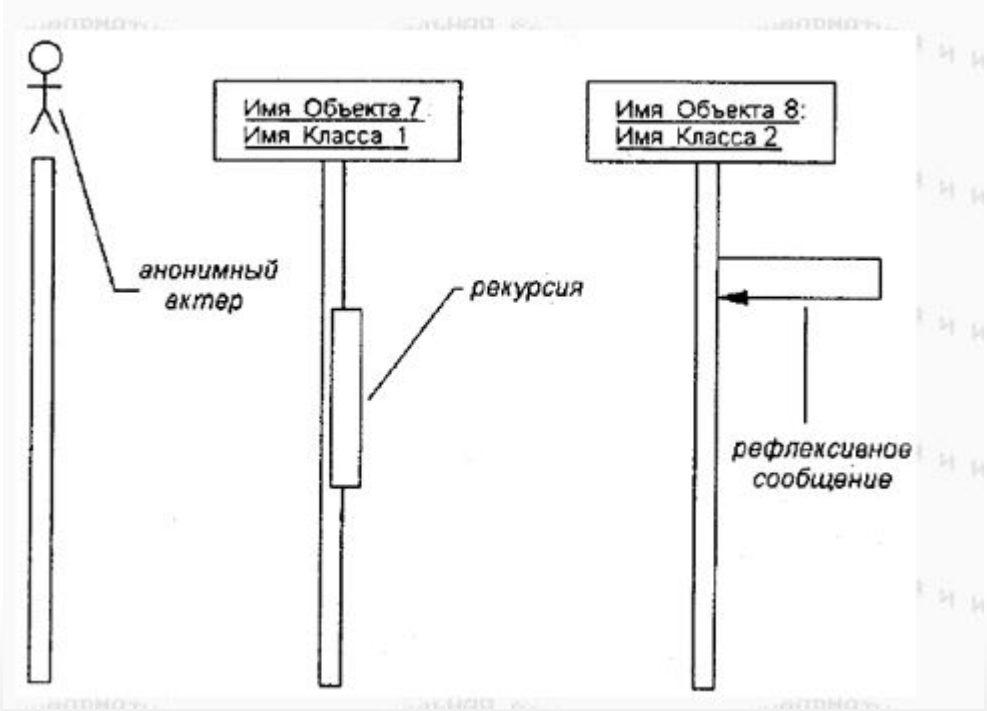
Линия жизни объекта



Диаграммы последовательностей

Элементы Диаграммы взаимодействия:

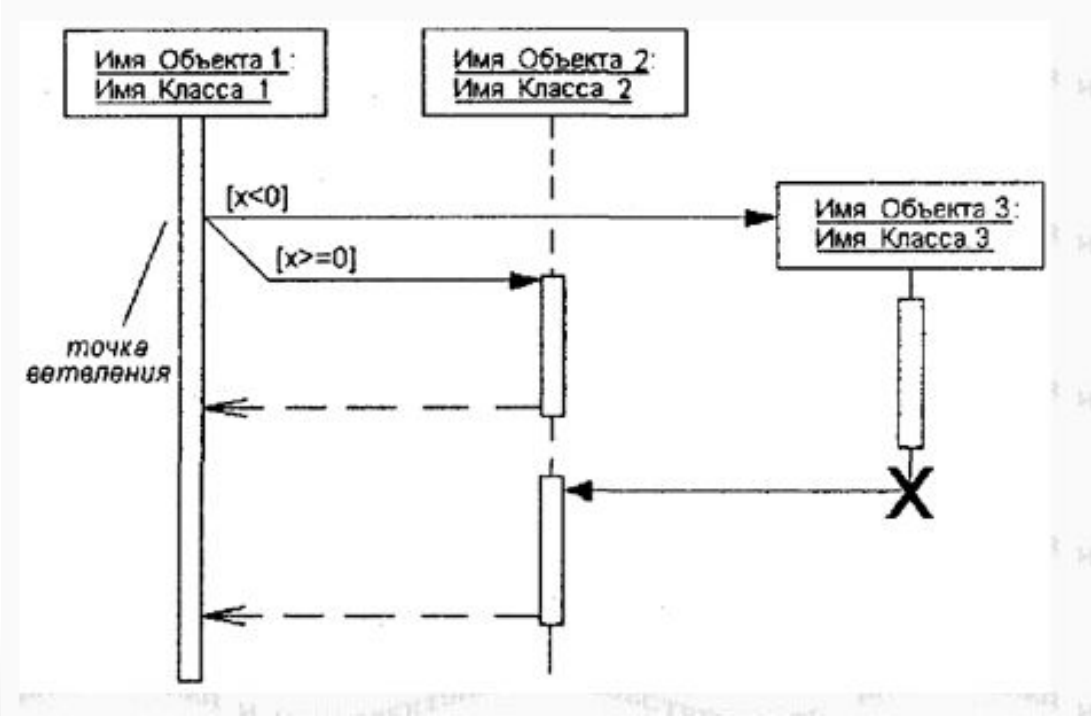
Фокус управления



Диаграммы последовательностей

Элементы Диаграммы взаимодействия:

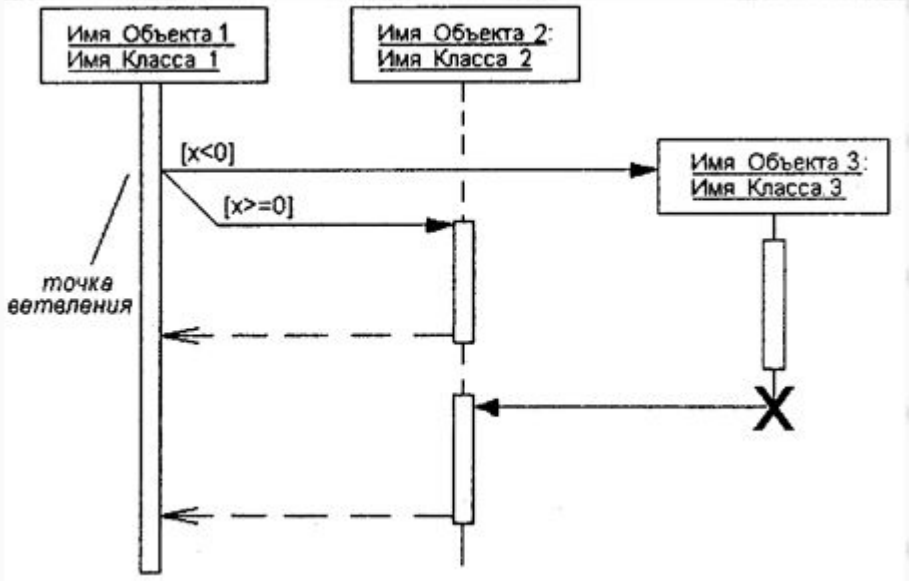
Сообщения

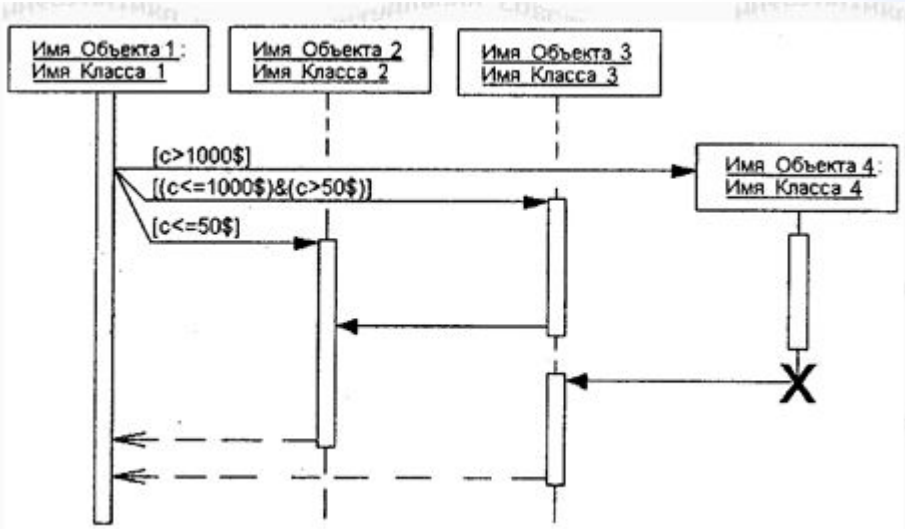


Диаграммы последовательностей

Элементы Диаграммы взаимодействия:

Ветвление потока управления

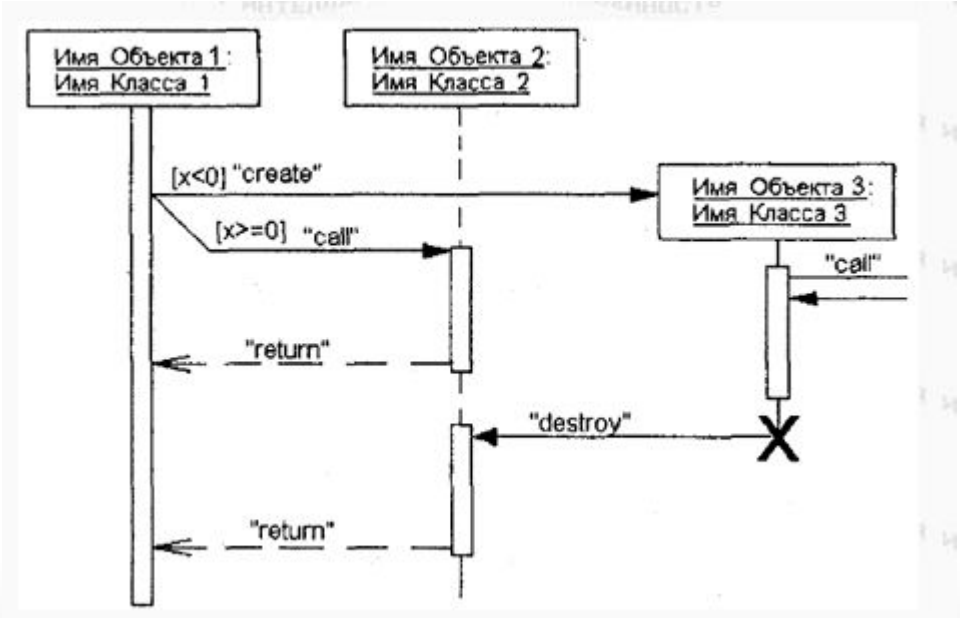




Диаграммы последовательностей

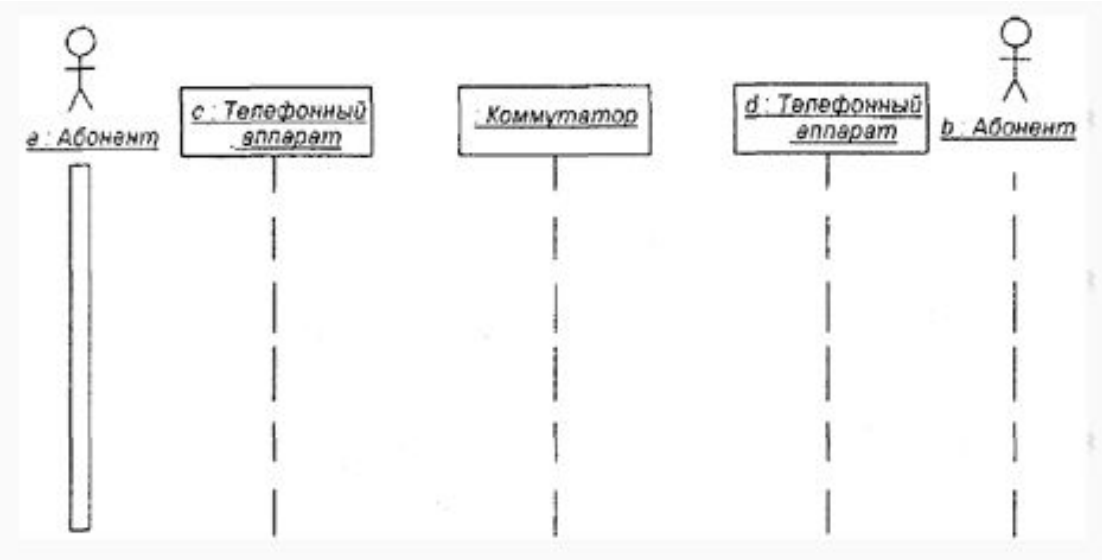
Элементы Диаграммы взаимодействия:

Стереотипы сообщений



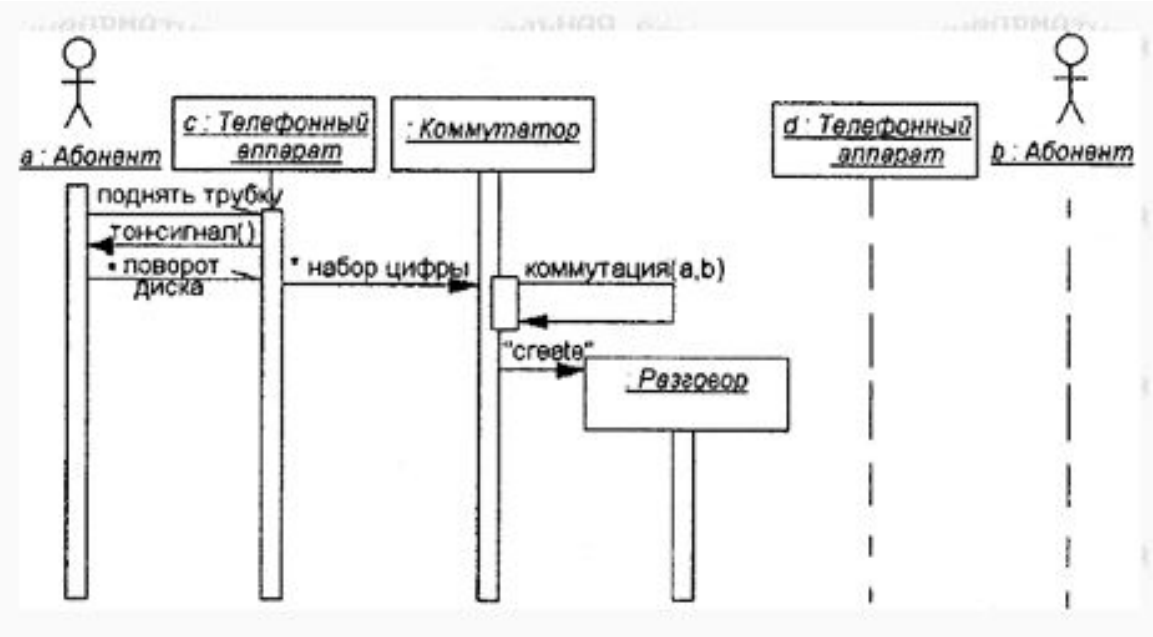
Диаграммы последовательностей

Пример построения Диаграммы взаимодействия: Этап 1



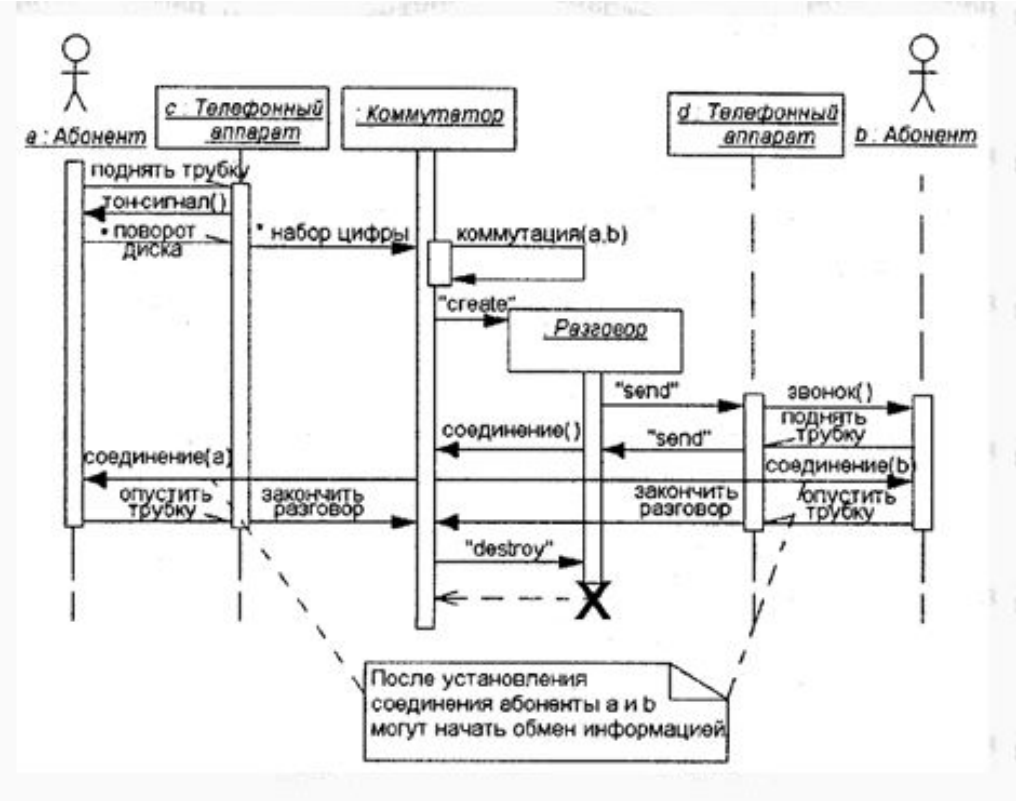
Диаграммы последовательностей

Пример построения Диаграммы взаимодействия: Этап 2



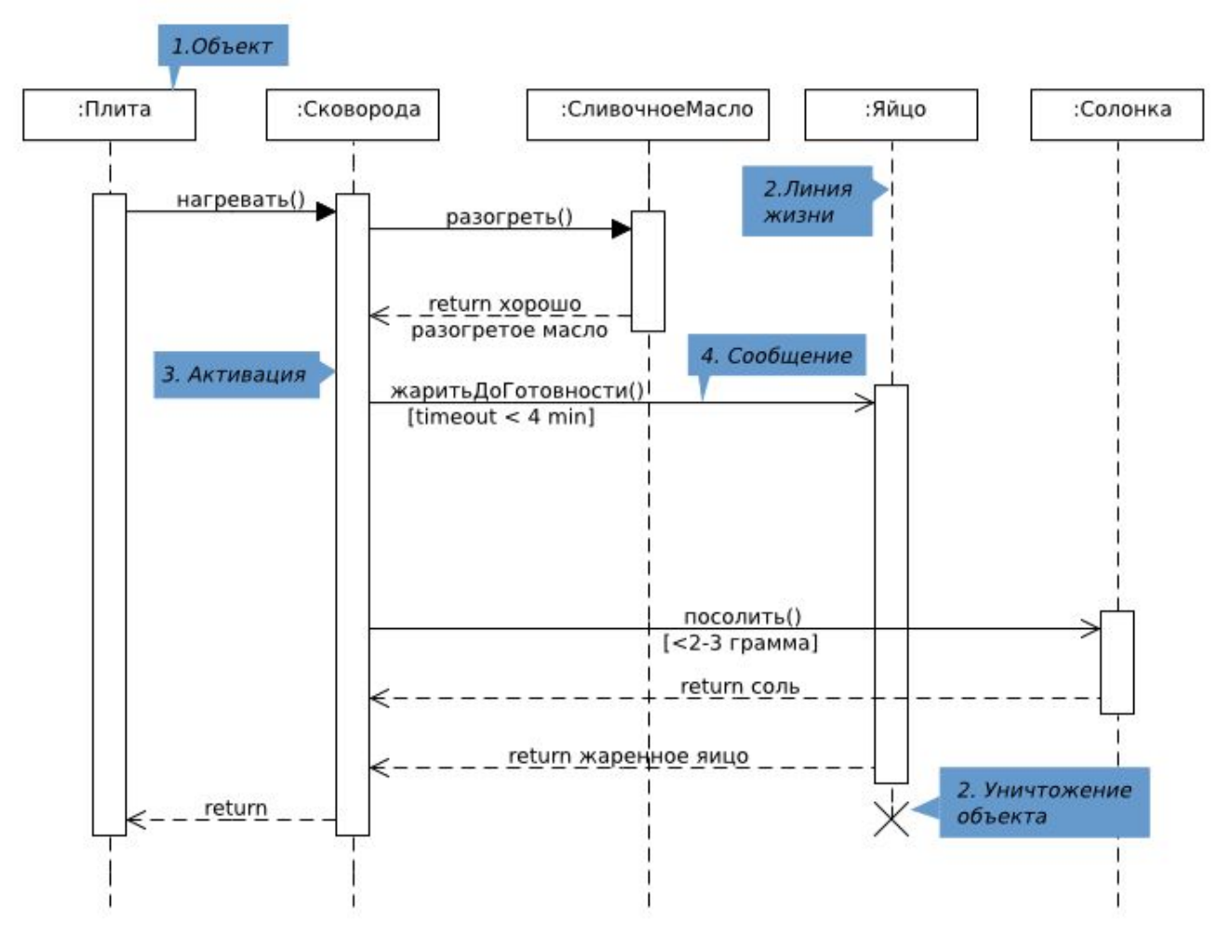
Диаграммы последовательностей

Пример построения Диаграммы взаимодействия: Этап 3



Диаграммы последовательностей

Пример построения Диаграммы взаимодействия



Диаграммы последовательностей

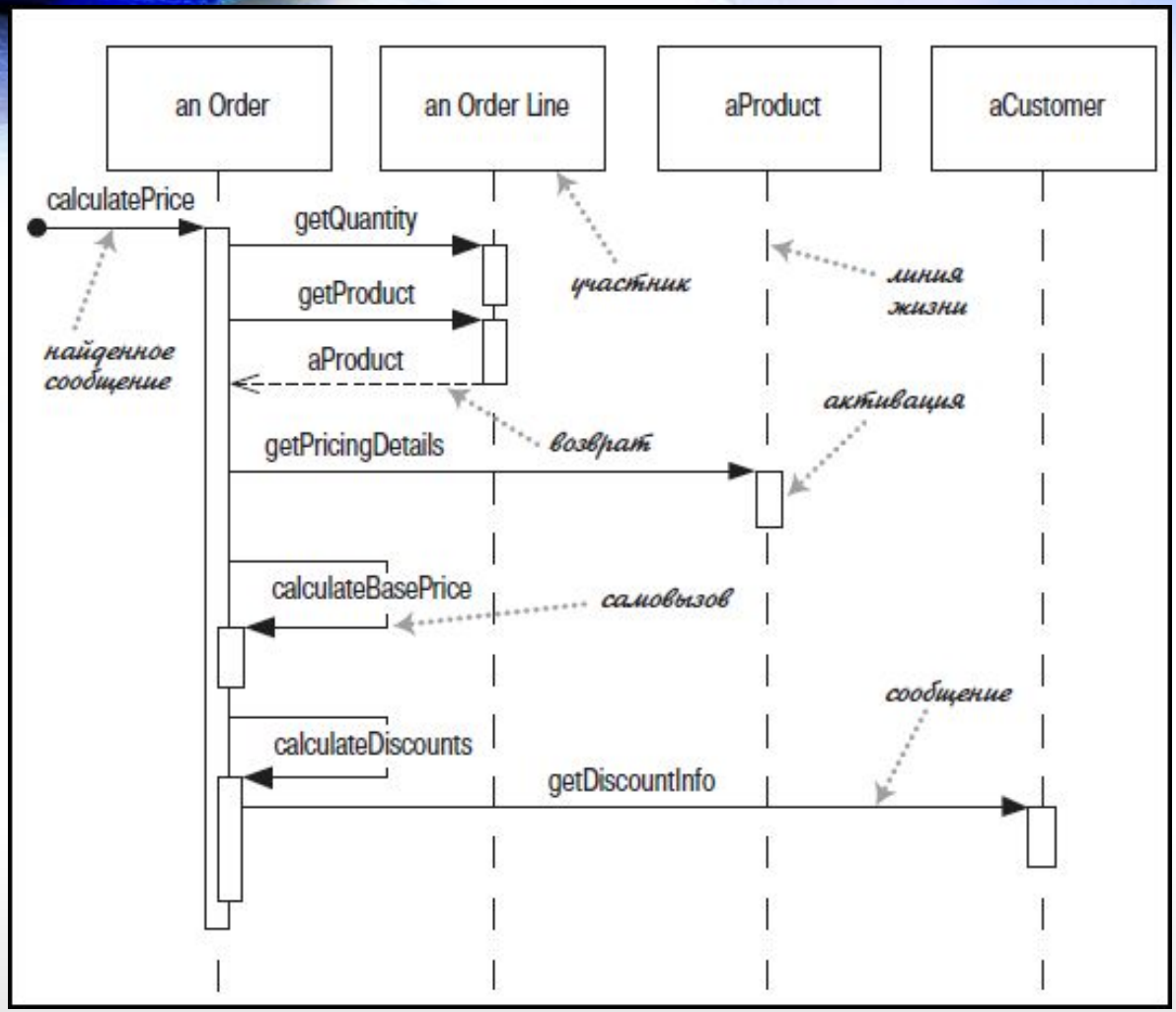
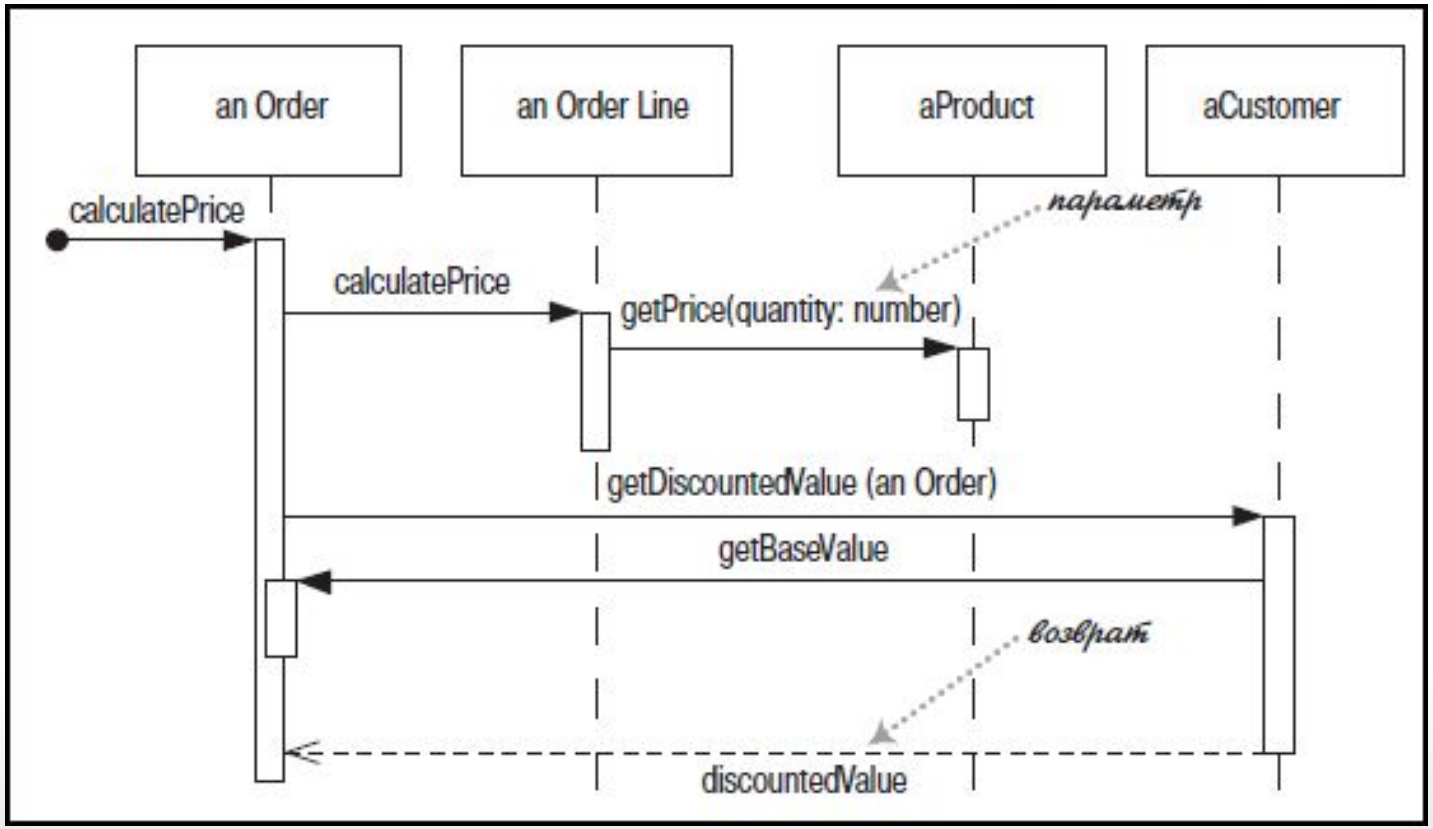


Диаграмма последовательности централизованного управления

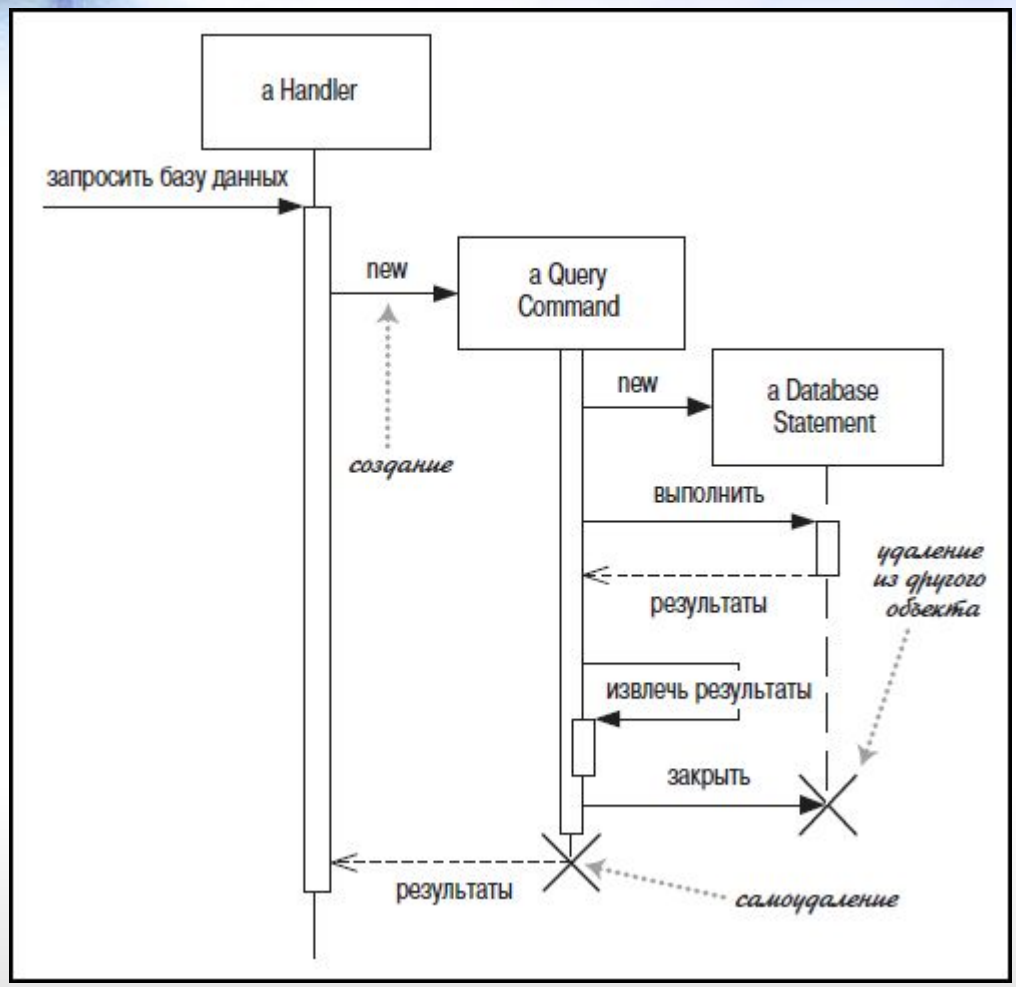
Диаграммы последовательностей

Диаграмма последовательности для распределенного управления



Диаграммы последовательностей

Создание и удаление участников



Диаграммы последовательностей

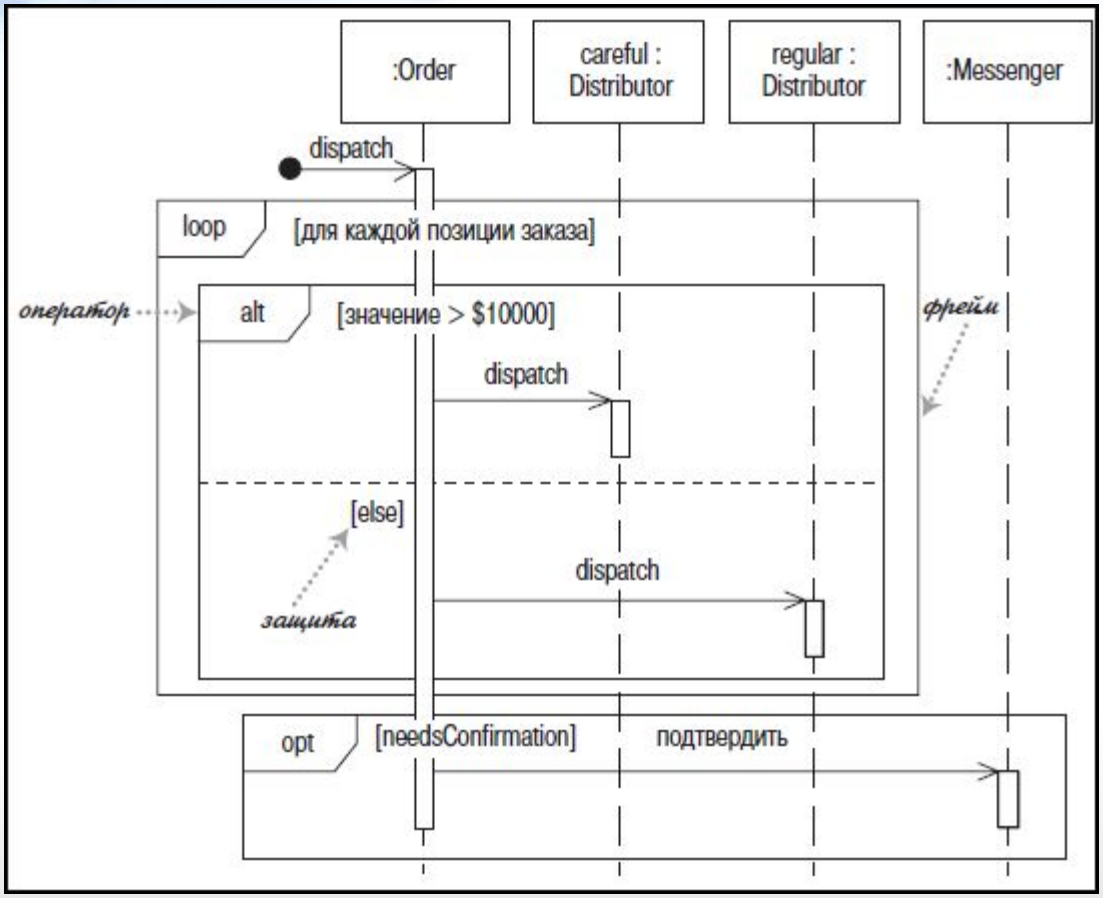
Циклы и условия

Общая проблема диаграмм последовательности заключается в том, как отображать циклы и условные конструкции.
Прежде всего надо знать, что диаграммы последовательности для этого не совсем предназначены.
Подобные управляющие структуры лучше показывать с помощью диаграммы деятельности.
Диаграммы последовательности применяются для визуализации процесса взаимодействия объектов, а не как средство моделирования алгоритма управления.

Диаграммы последовательностей

Циклы и условия

Фреймы взаимодействия



Диаграммы последовательностей

Фреймы взаимодействия

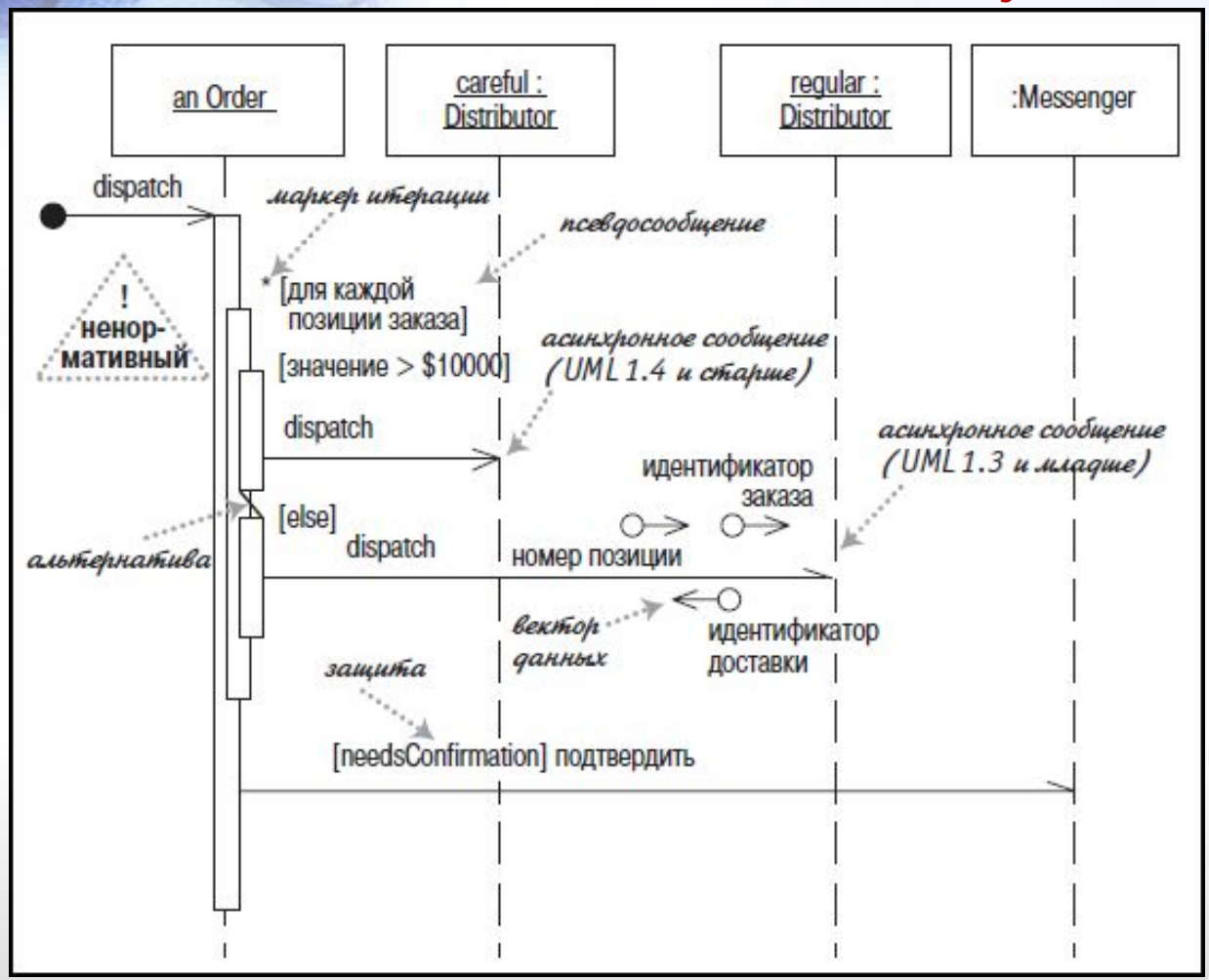
Оператор	Значение
alt	Несколько альтернативных фрагментов (alternative); выполняется только тот фрагмент, условие которого истинно (рис. 4.4)
opt	Необязательный (optional) фрагмент; выполняется, только если условие истинно. Эквивалентно alt с одной веткой (рис. 4.4)
par	Параллельный (parallel); все фрагменты выполняются параллельно
loop	Цикл (loop); фрагмент может выполняться несколько раз, а защита обозначает тело итерации (рис. 4.4)
region	Критическая область (critical region); фрагмент может иметь только один поток, выполняющийся за один прием
neg	Отрицательный (negative) фрагмент; обозначает неверное взаимодействие
ref	Ссылка (reference); ссылается на взаимодействие, определенное на другой диаграмме. Фрейм рисуется, чтобы охватить линии жизни, вовлеченные во взаимодействие. Можно определять параметры и возвращать значение
sd	Диаграмма последовательности (sequence diagram); используется для очерчивания всей диаграммы последовательности, если это необходимо

Общепринятые операторы для фреймов взаимодействия

Диаграммы последовательностей

Циклы и условия

Вместо Фреймов взаимодействия – Старые соглашения для условной логики



В UML 1 использовались маркеры итераций и защиты. В качестве маркера итерации (iteration marker) выступал символ *, добавленный к имени сообщения.

Для обозначения тела итерации можно добавить текст в квадратных скобках.

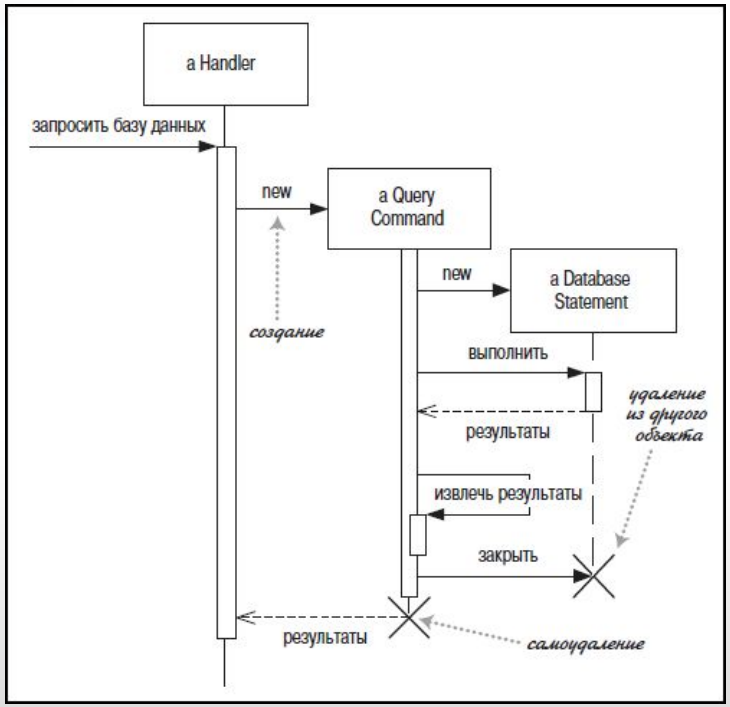
Защита (guard) – это условное выражение, размещенное в квадратных скобках и означающее, что сообщение посылается, только когда защита принимает истинное значение.

Диаграммы последовательностей

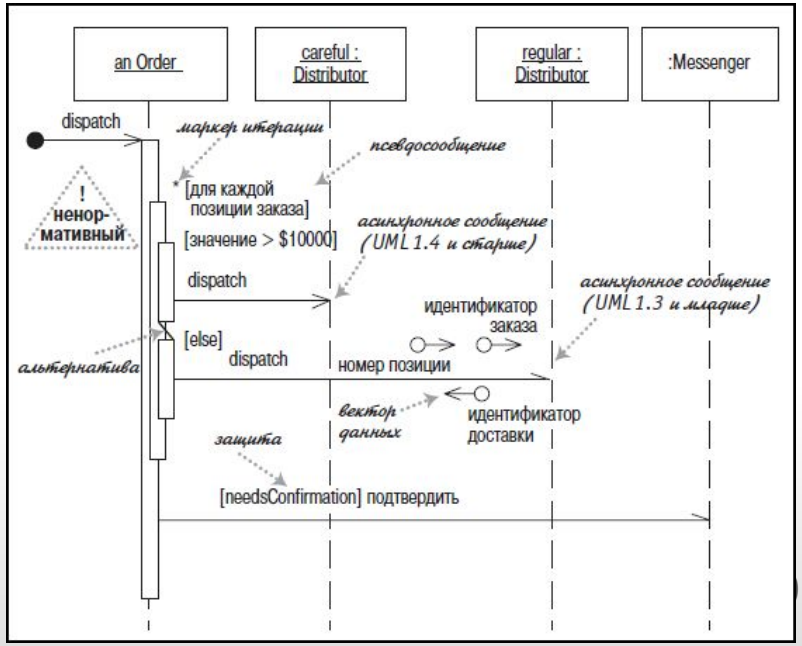
Синхронные и асинхронные вызовы

Если вызывающий объект посылает синхронное сообщение (synchronous message), то он должен ждать, пока обработка сообщения не будет закончена.
Если вызывающий объект посылает асинхронное сообщение (asynchronous message), то он может продолжать работу и не должен ждать ответа.

В UML2:
Закрашенные стрелки показывают синхронное сообщение.
Простые стрелки обозначают асинхронное сообщение.



В UML1:
Обычные стрелки показывают синхронное сообщение.
Половинные стрелки обозначают асинхронное сообщение.



Диаграммы объектов

***Диаграмма объектов (object diagram)** – это снимок объектов системы в какой-то момент времени.*
*Поскольку она показывает экземпляры, а не классы, то диаграмму объектов часто называют **диаграммой экземпляров.***

Каждое имя объекта представляется в виде: имя экземпляра : имя класса.

Элементы диаграммы объектов – это спецификации экземпляров, а не сами экземпляры.

Диаграмма классов, показывающая структуру класса

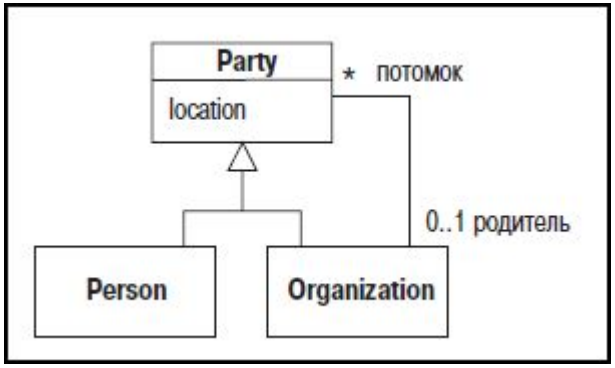
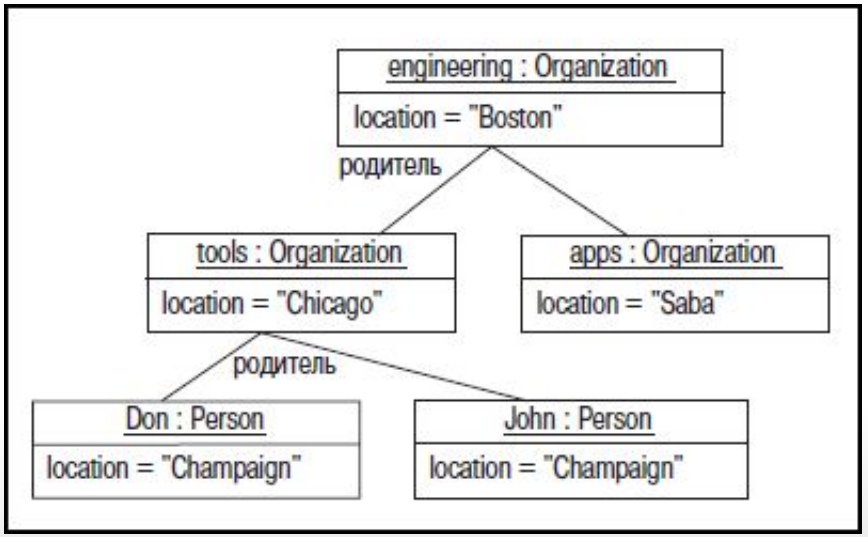
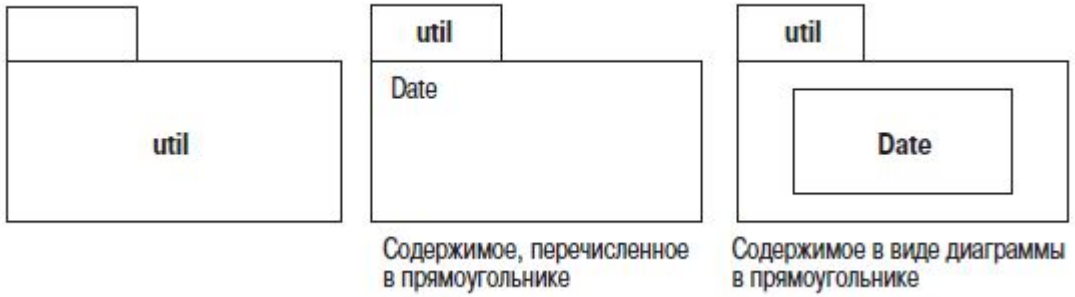


Диаграмма объектов с примером экземпляра класса

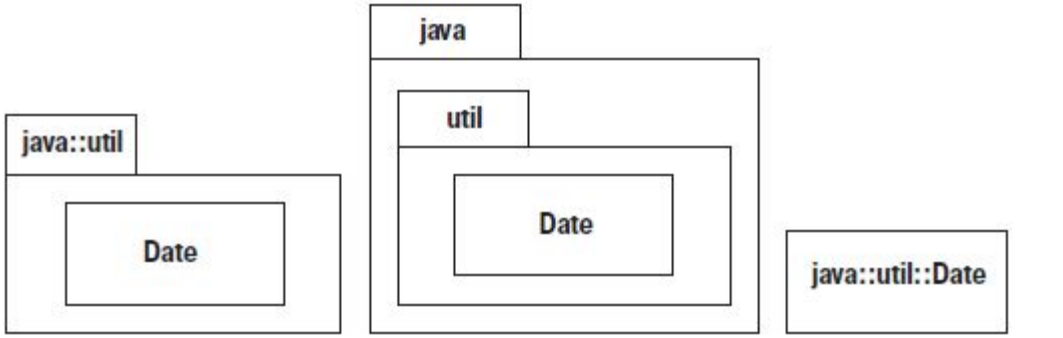


Диаграммы пакетов

Пакет (package) – это инструмент группирования, который позволяет взять любую конструкцию UML и объединить ее элементы в единицы высокого уровня



Способы изображения пакетов на диаграммах



Диаграммы пакетов

*Каждый пакет представляет **пространство имен (namespace).** Это означает, что каждый класс внутри собственного пакета должен иметь уникальное имя.*

Например, если создается пакет с именем Date, а класс Date уже существует в пакете System, то необходимо поместить его в отдельный пакет.

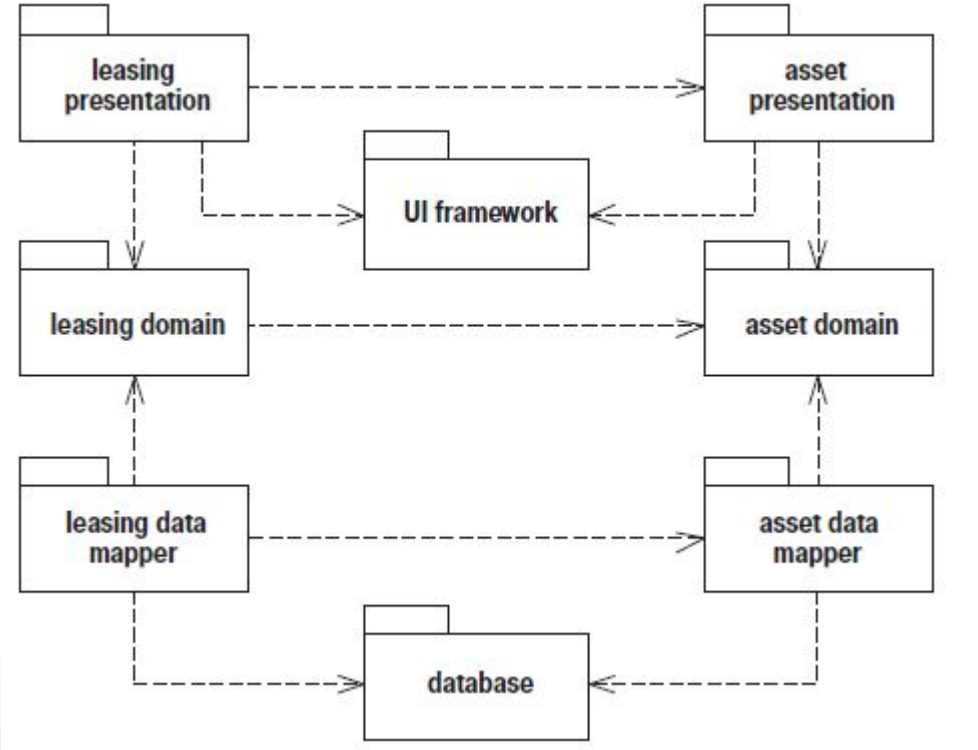
*Чтобы отличить один класс от другого, можно использовать **полностью определенное имя (fully qualified name),** то есть имя, которое указывает на структуру, владеющую пакетом.*

В языке UML в именах пакетов используются двойные двоеточия, поэтому классы могут иметь имена
System::Date** или **MartinFowler::Util::Date.

Диаграммы пакетов

Пакеты и зависимости

Диаграмма пакетов (package diagram) показывает пакеты и зависимости между ними.

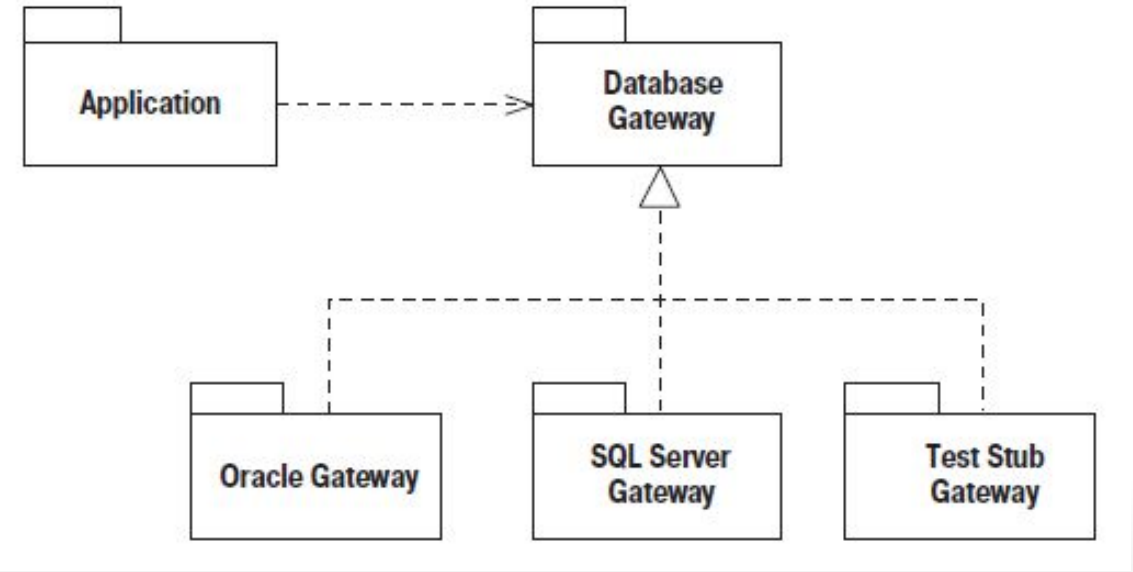


Диаграммы пакетов

Реализация пакетов

Отношение реализации означает:

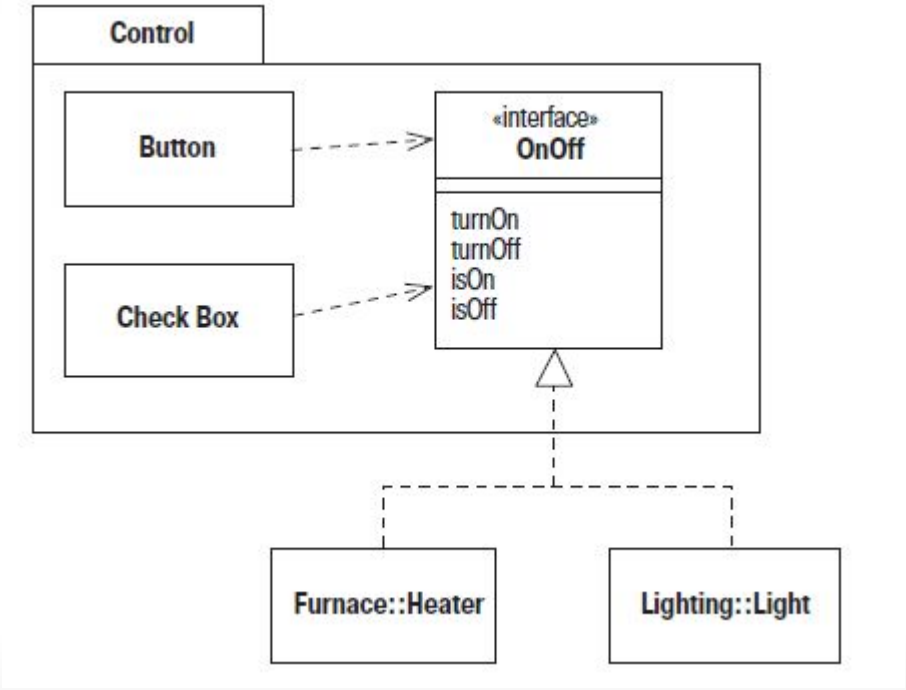
- ***шлюз базы данных (Database Gateway) определяет интерфейс,***
- ***а другие классы шлюзов обеспечивают реализацию.***



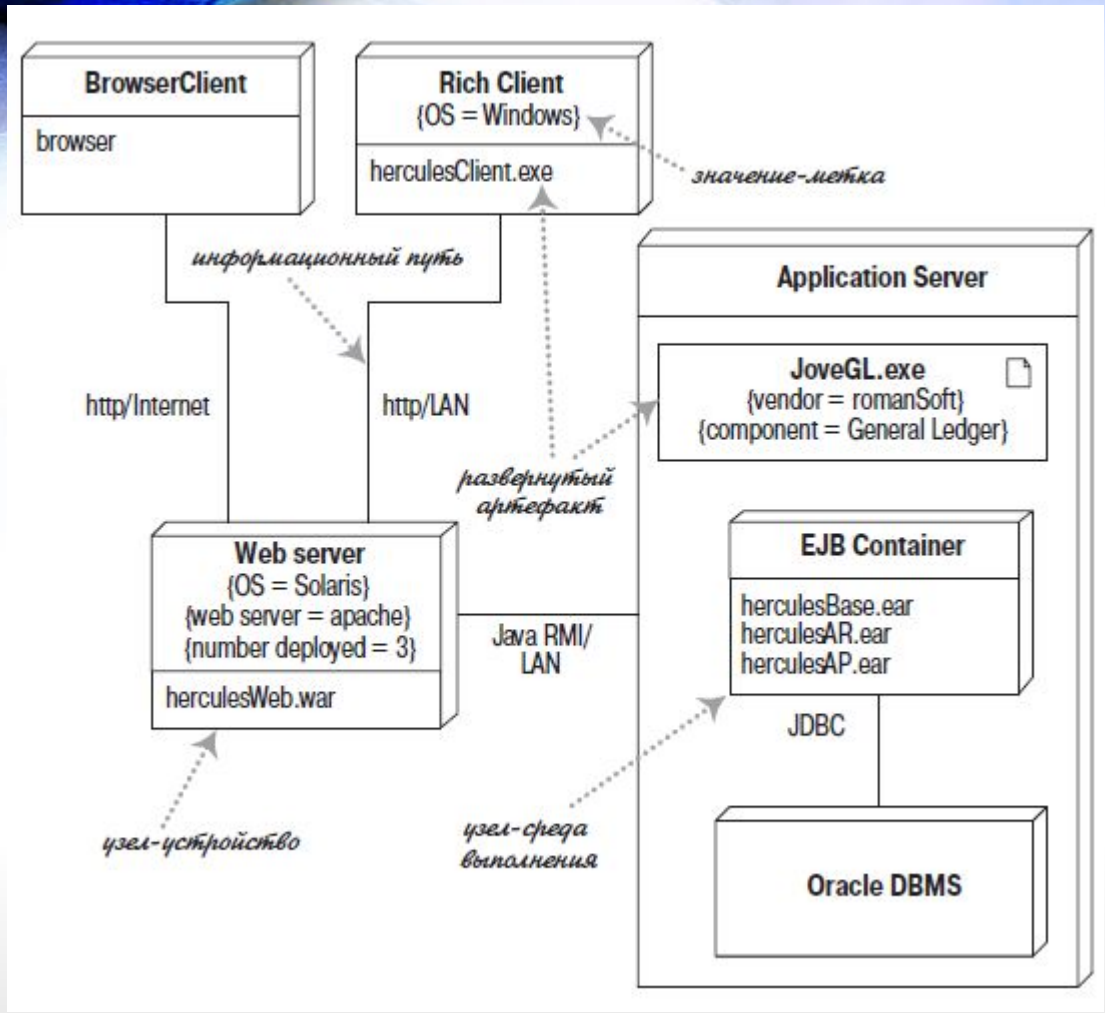
Диаграммы пакетов

Реализация пакетов

Общепринято размещать интерфейс и его реализацию в разных пакетах

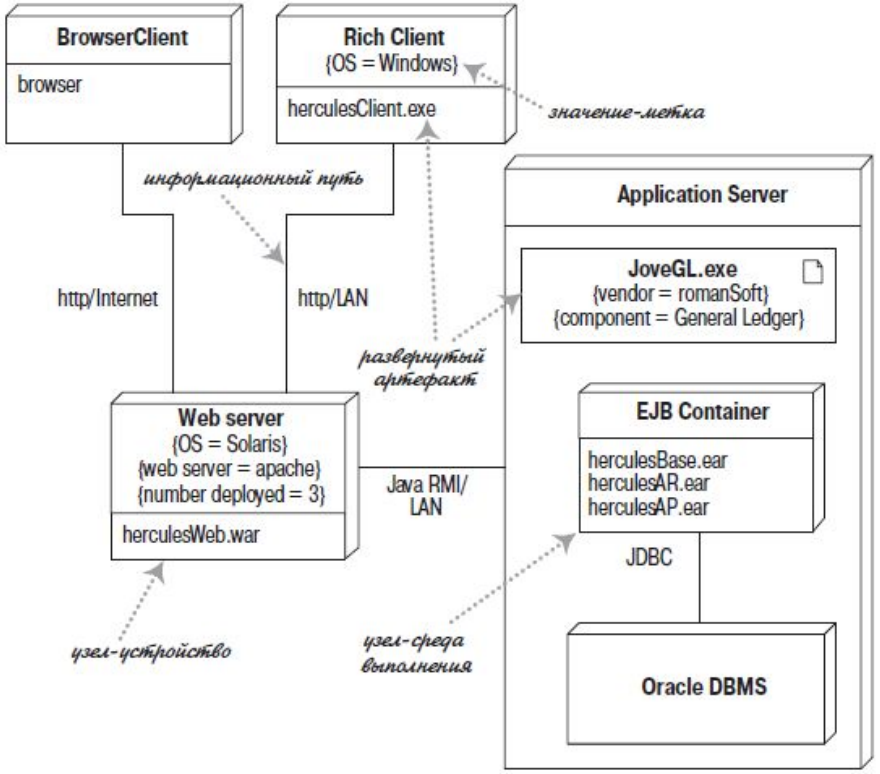


Диаграммы развертывания



Диаграммы развертывания представляют физическое расположение системы, показывая, на каком физическом оборудовании запускается та или иная составляющая программного обеспечения.

Диаграммы развертывания



Главными элементами диаграммы являются узлы, связанные информационными путями.

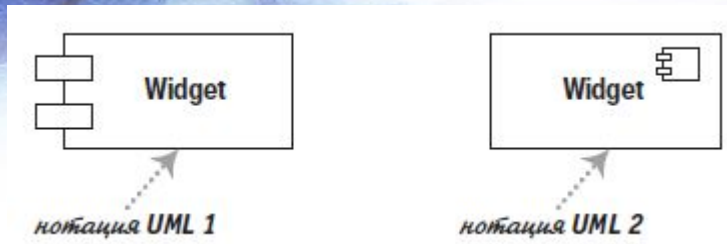
Узел (node) – это то, что может содержать программное обеспечение.

Узлы бывают двух типов:
Устройство (device) – это физическое оборудование: компьютер или устройство, связанное с системой.
Среда выполнения (execution environment) – это программное обеспечение, которое само может включать другое программное обеспечение, например операционную систему или процесс-контейнер.

Узлы могут содержать **артефакты (artifacts),** которые являются физическим олицетворением программного обеспечения; обычно это файлы.

Диаграммы компонентов

Нотация для компонентов



Пример диаграммы компонентов