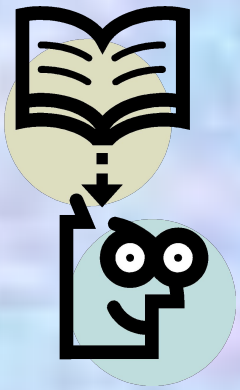
UML – УНИФИЦИРОВАННЫЙ ЯЗЫК ОБЪЕКТНО-ОРИЕНТИРОВАННОГО МОДЕЛИРОВАНИЯ И ДОКУМЕНТИРОВАНИЯ СЛОЖНЫХ СИСТЕМ

UML (Unified Modeling Language)

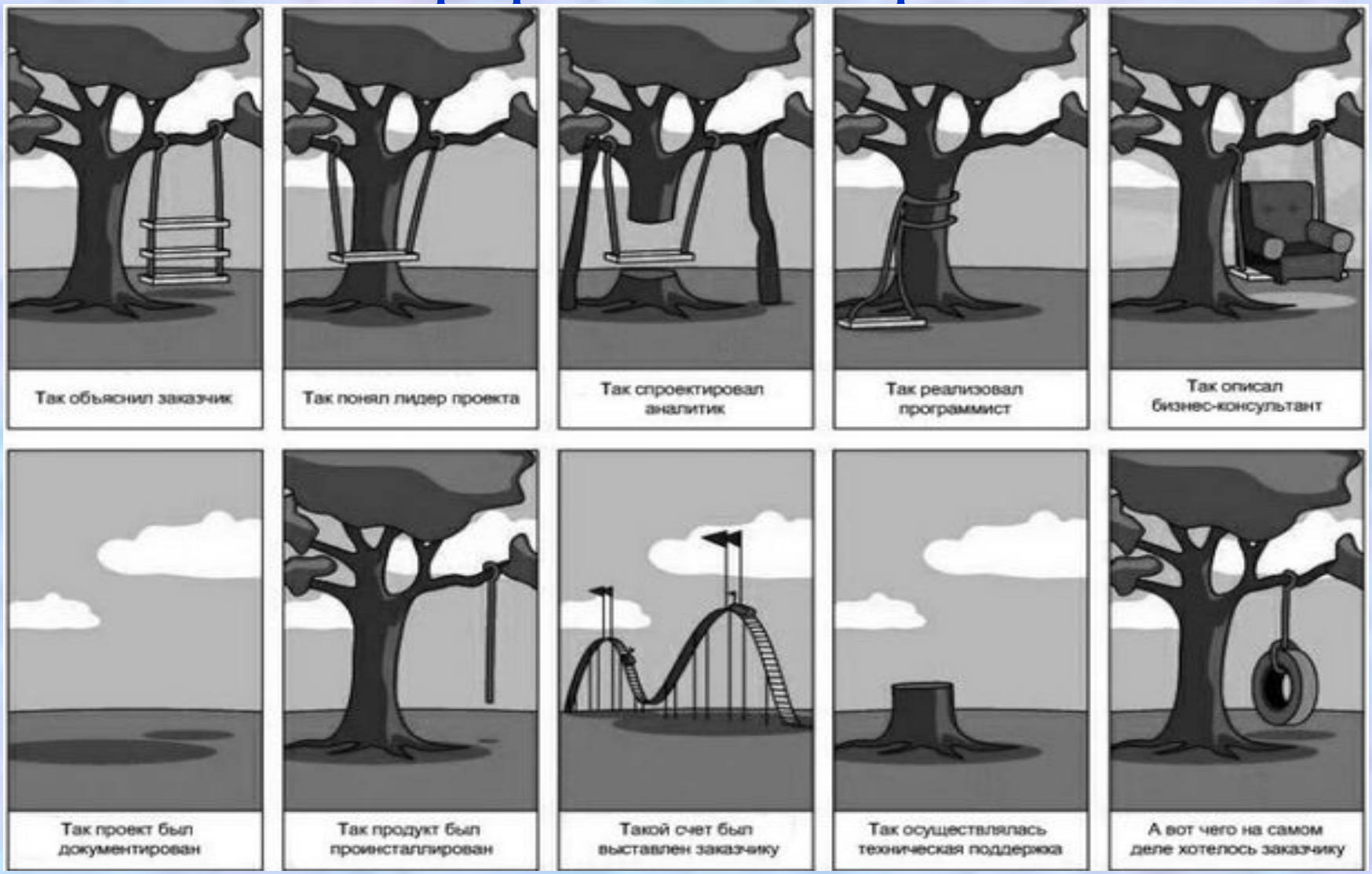
Язык визуального моделирования, разработанный для **спецификации**, **визуализации**, **проектирования**, **документирования** компонентов программного обеспечения, бизнес-процессов и других программных систем.

Назначение языка UML



Предоставить в распоряжение пользователей легко воспринимаемый и выразительный язык визуального моделирования, специально предназначенный для разработки и документирования моделей сложных систем самого различного целевого назначения.

Проблемы с коммуникацией и пониманием задачи в программной инженерии



История создания

1996 г.-первая версия 0.9

1997 г. - версии языка UML 1.0 и 1.1,
*Принят **первый стандарт** OMG.*

1998 г - версия UML 1.2

1999 г - версия UML 1.3

2000 г - версия UML 1.4

*2005- UML 2.0 **Второй стандарт***

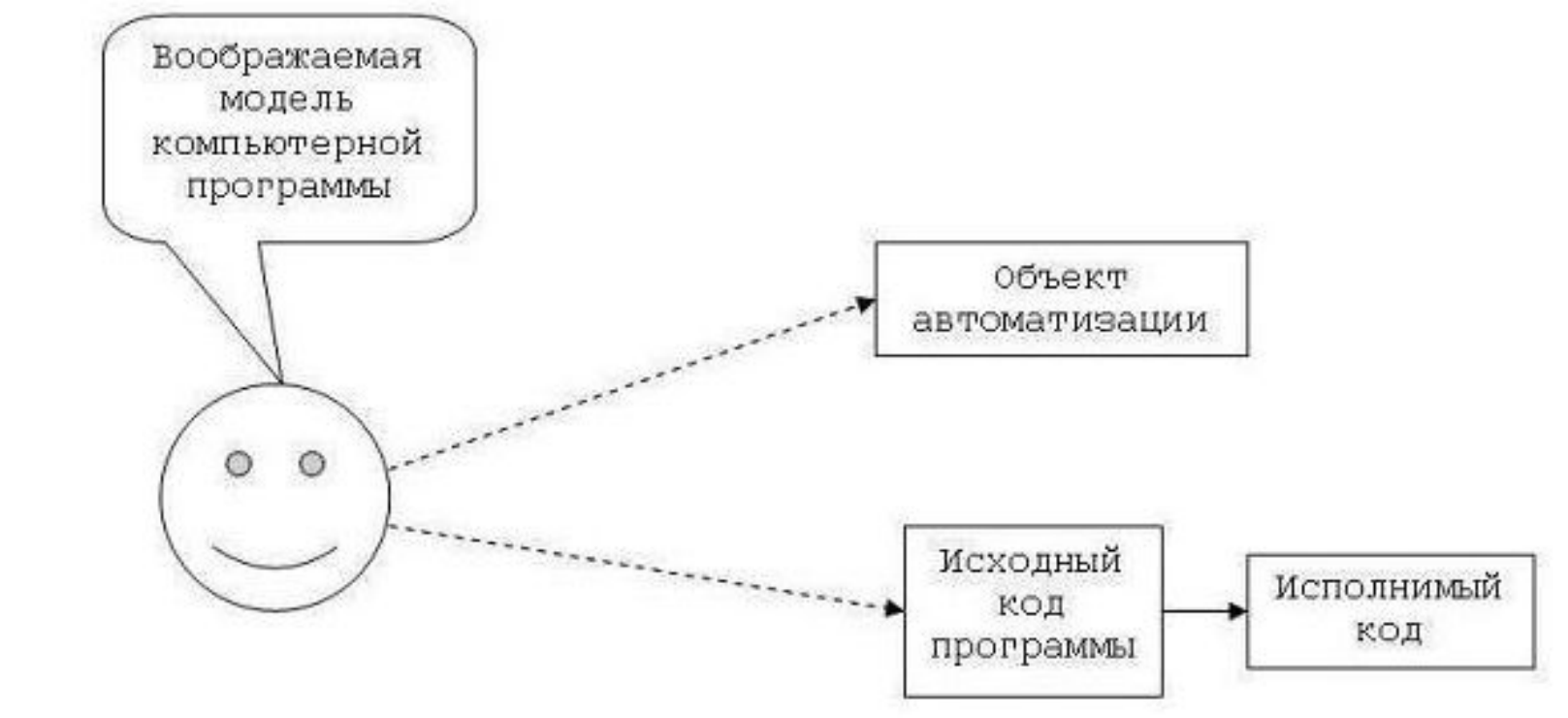
Группа OMG продолжает работы по созданию новых версий языка UML

UML – это стандартная нотация визуального моделирования программных систем, принятая консорциумом Object Management Group (OMG)

Создатели языка:

Гради Буч,
Джим Рамбо
Айвар Якобсон

Процесс проектирования ПО до появления UML



объединение текста программы (ее исходного кода) с характеристиками объекта автоматизации осуществляется только в сознании программиста, а документальная связь между ними *отсутствует*.

Процесс проектирования ПО с использованием UML

Диаграммы и спецификации языка UML *связали* исходный текст программы с характеристиками объекта автоматизации. При этом UML диаграммы опираются на теоретический фундамент в виде теории множеств и теории графов, что позволяет выполнить преобразование UML диаграмм в исходный код программы.

Строительные блоки UML

Словарь языка UML включает три вида строительных блоков:

- сущности;

- отношения;

- диаграммы

Модель представляется в виде сущностей и отношений между ними, которые показываются на ***диаграммах.***

Диаграмма - это графическое представление множества элементов. Обычно изображается в виде графа с вершинами (сущностями) и ребрами (отношениями).

Сущности - это абстракции, являющиеся основными элементами моделей. Имеется четыре типа сущностей - **структурные** (класс, интерфейс, компонент, вариант использования, кооперация, узел), **поведенческие** (взаимодействие, состояние), **группирующие** (пакеты) и **аннотационные** (комментарии). Каждый вид сущностей имеет свое графическое представление.

ГРУППЫ ДИАГРАММ

- 1. Статические
- 2. Поведенческие
- 3. Реализации

Диаграмма прецедентов (use case diagram, вариантов использования)

Представляет динамические или поведенческие аспекты системы.

- **Базовыми элементами диаграммы вариантов использования являются *вариант использования* и *эктор* (актер)**

- ***Вариант использования* (use case) — внешняя спецификация последовательности действий, которые система или другая сущность могут выполнять в процессе взаимодействия с актерами.**
- ***Актер (actor)* — согласованное множество ролей, которые играют внешние сущности по отношению к вариантам использования при взаимодействии с ними.**

Графические обозначения на диаграмме прецедентов

Эктор

Вариант использования

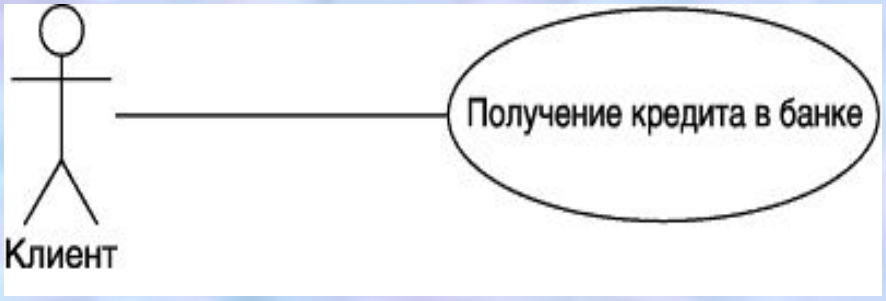
Отношения на диаграмме вариантов использования

Виды отношений между актерами и вариантами использования:

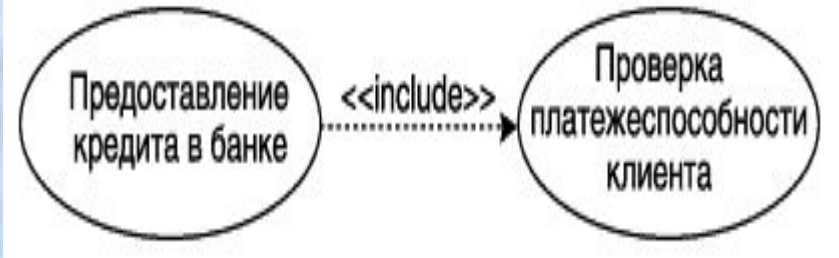
- **ассоциации** (association relationship)
- **включения** (include relationship)
- расширения (extend relationship)
- обобщения (generalization relationship)

Отношения на диаграмме вариантов использования

Ассоциация **служит для обозначения специфической роли актера при его взаимодействии с отдельным вариантом использования**

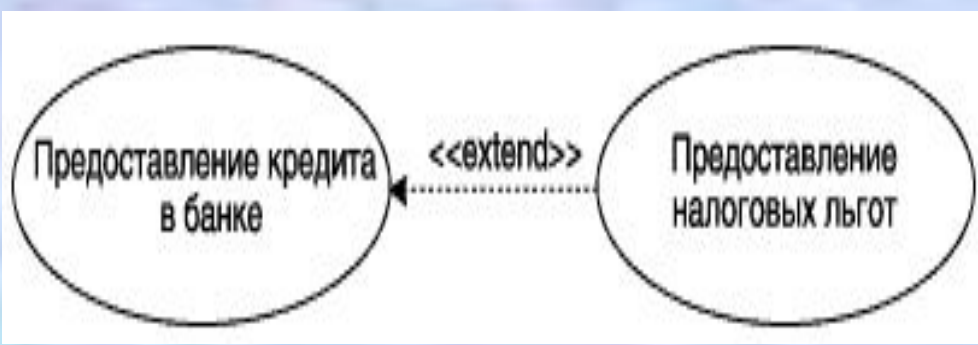


Включение **(include) — это разновидность отношения зависимости между базовым вариантом использования и его специальным случаем. При этом отношением зависимости (dependency) является такое отношение между двумя элементами модели, при котором изменение одного элемента (независимого) приводит к изменению другого элемента (зависимого).**

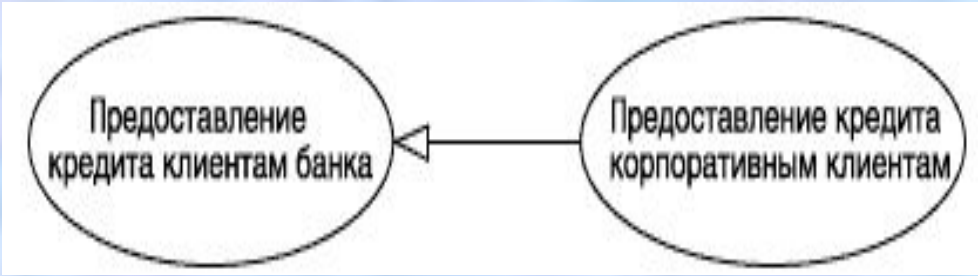


Отношения на диаграмме вариантов использования

Расширение **(extend) определяет взаимосвязь базового варианта использования с другим вариантом использования, функциональное поведение которого задействуется базовым не всегда, а только при выполнении дополнительных условий.**

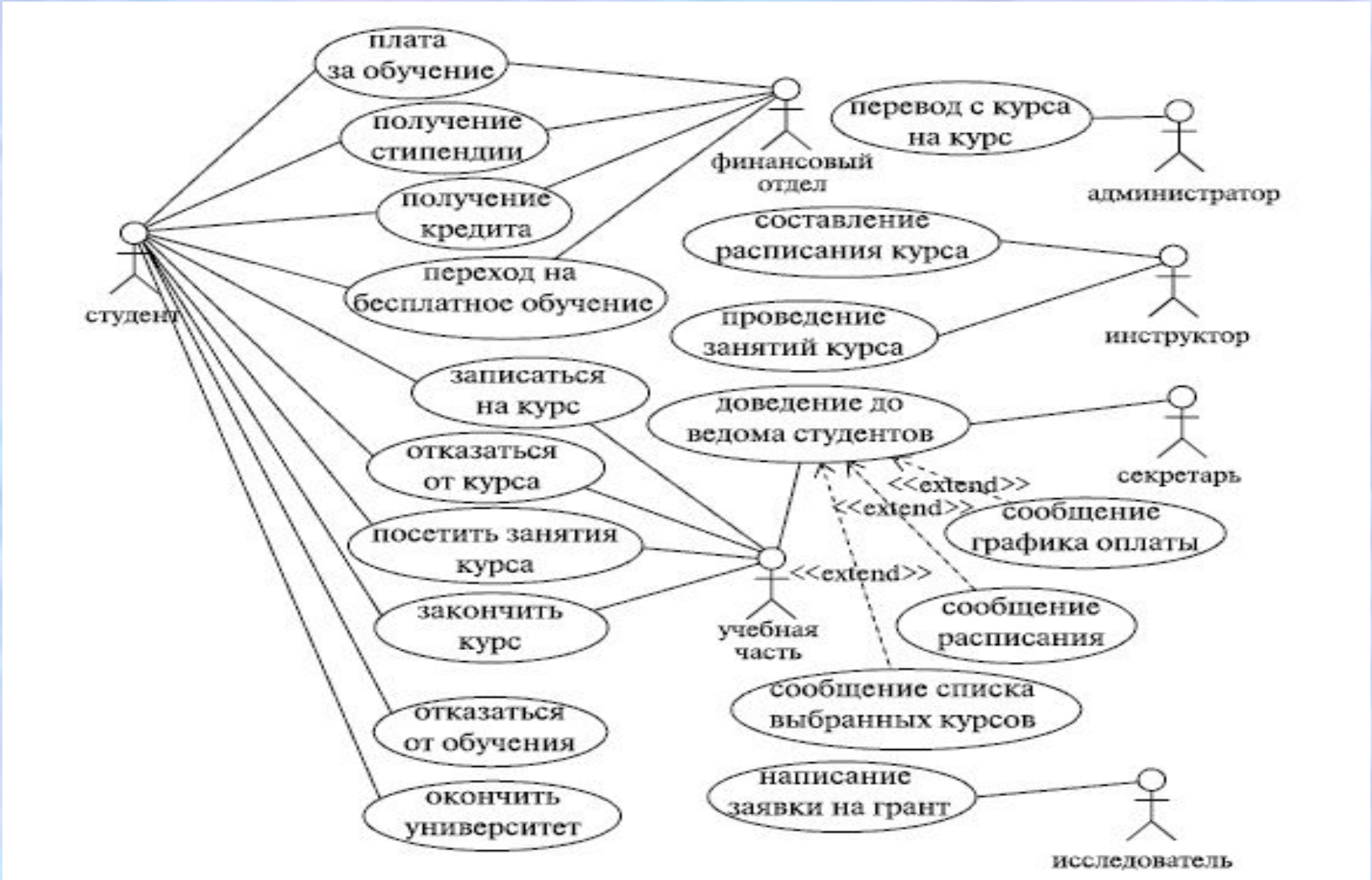


Два и более актера могут иметь общие свойства, т. е. взаимодействовать с одним и тем же множеством вариантов использования одинаковым образом. Такая общность свойств и поведения представляется в виде отношения ***обобщения*** **с другим, возможно, абстрактным актером, который моделирует соответствующую общность ролей.**



Пример диаграммы прецедентов

Пример диаграммы прецедентов



Цели создания диаграмм прецедентов

- определение границы и контекста моделируемой предметной области на ранних этапах проектирования;
- формирование общих требований к поведению проектируемой системы;
- разработка концептуальной модели системы для ее последующей детализации;
- подготовка документации для взаимодействия с заказчиками и пользователями системы

Диаграмма классов (class diagram)

- Диаграммы классов дают ***статический вид*** системы. Они представляют собой взгляды разработчиков на статические состояния проектируемых систем.
- На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами.
- Информация с диаграммы классов может напрямую отображается в исходный код приложения - в большинстве существующих инструментов UML-моделирования возможна *кодогенерация* для определенного языка программирования (обычно Java или C++).

. Изображение класса в UML

Обозначения признаков видимости:

+ public

 # protected

- private

Отношения между классами

- **Ассоциации** (association relationship)
- **Обобщения** (generalization relationship)
- **Агрегации** (aggregation relationship)
- **Композиции** (composition relationship)

Отношение ассоциации

Ассоциация (association) - семантическое отношение между двумя и более классами, которое специфицирует характер связи между соответствующими экземплярами этих классов

Ассоциация может быть:

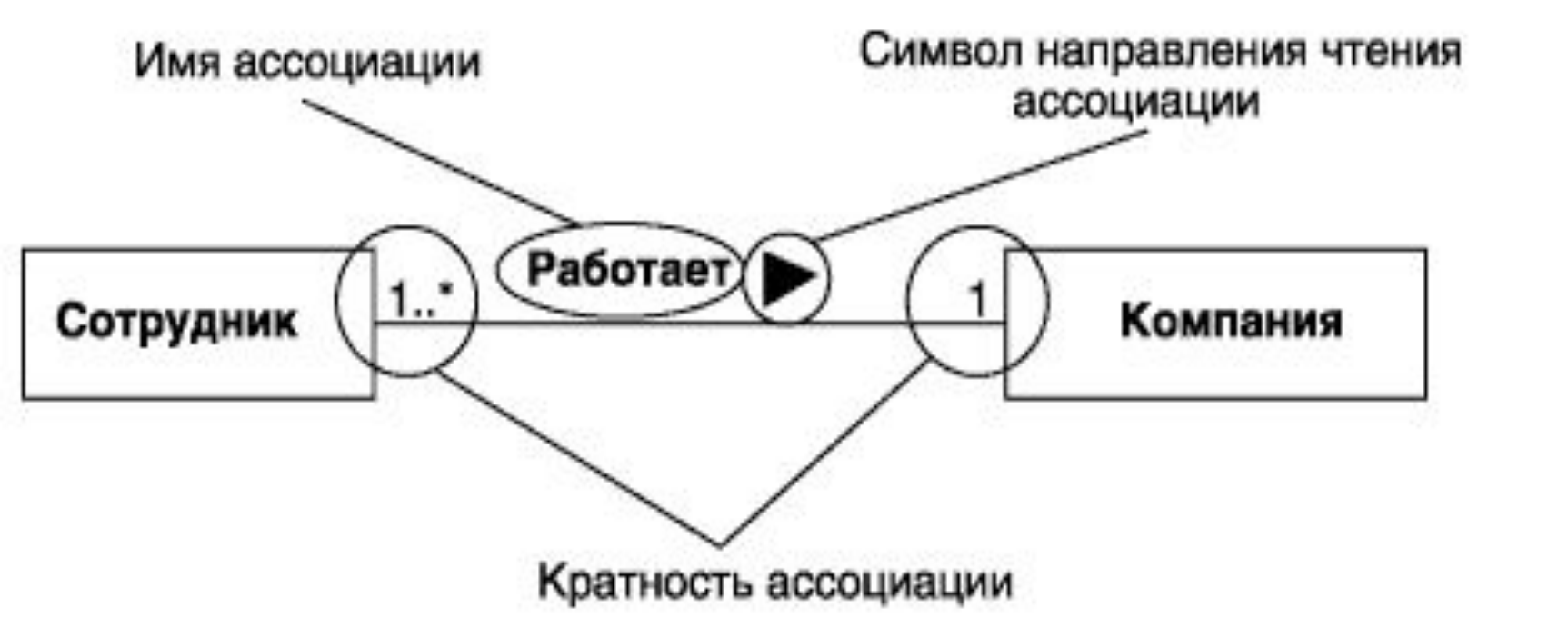
бинарной или n-арной

бинарная ассоциация (binary association), служит для представления произвольного отношения между двумя классами. Она связывает в точности два различных класса и может быть ненаправленным (симметричным) или направленным отношением

n-арная ассоциация (n-ary association) - ассоциация между тремя и большим числом классов

Пример ненаправленной бинарной ассоциации

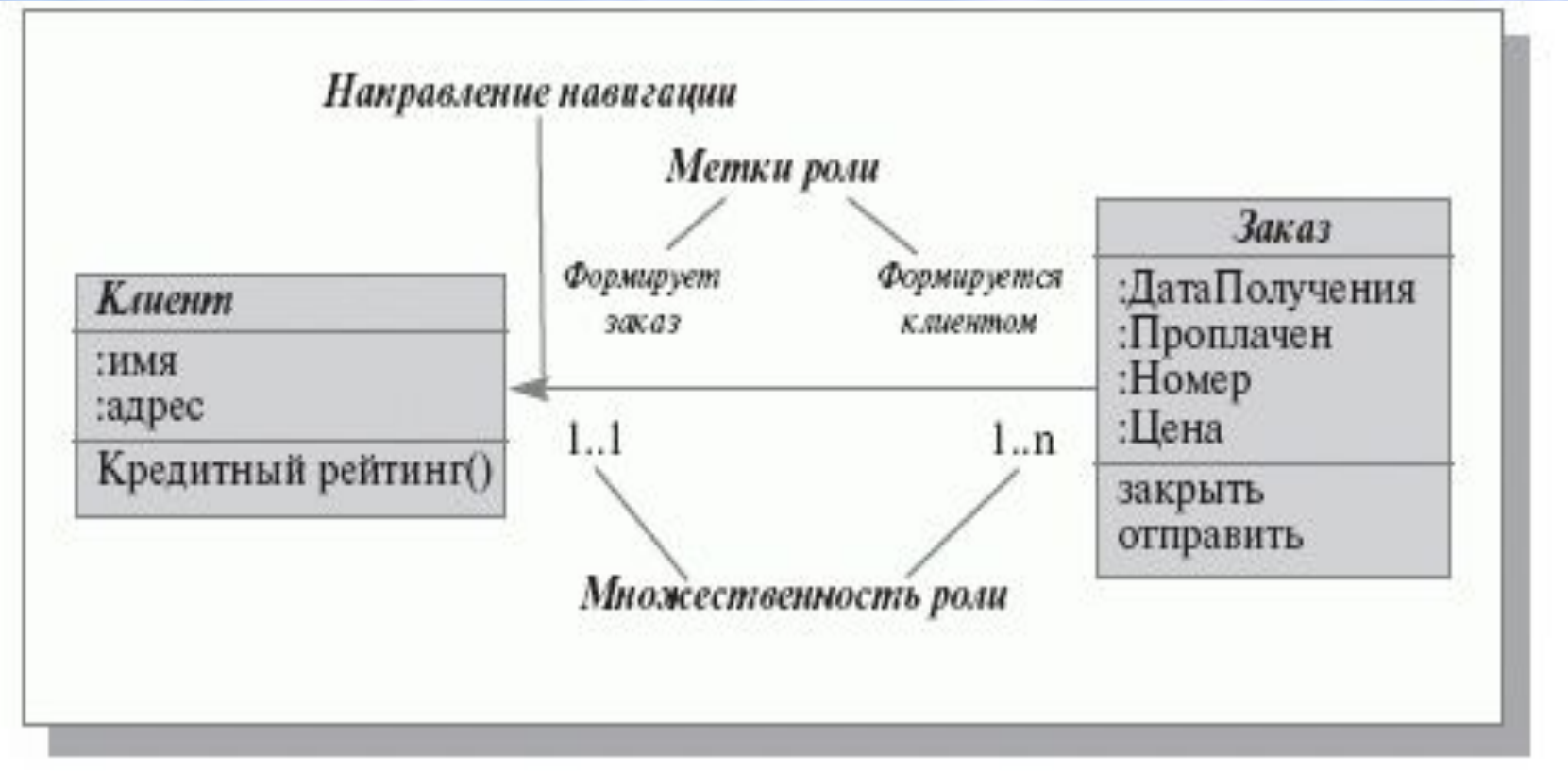
Ненаправленная бинарная ассоциация изображается линией без стрелки. Для нее на диаграмме может быть указан порядок чтения классов с использованием значка в форме треугольника рядом с именем данной ассоциации.



Классы *Компания и Сотрудник* связаны между собой бинарной ассоциацией *Работает*, имя которой указано на рисунке рядом с *линией ассоциации*. Для данного отношения определен следующий порядок чтения следования классов - сотрудник работает в компании

Пример направленной бинарной ассоциации

Направленная бинарная ассоциация изображается сплошной линией с простой стрелкой на одной из ее концевых точек.



Каждый заказ может быть создан единственным клиентом (множественность роли 1.1). Каждый клиент может создать один и более заказов (множественность роли 1..n). Направление навигации показывает, что каждый заказ должен быть "привязан" к определенному клиенту

Пример тернарной ассоциации

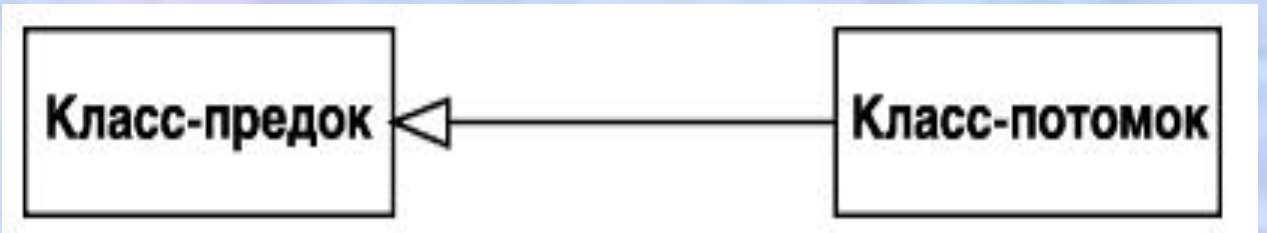
n-арная ассоциация обозначается ромбом, от которого ведут линии к символам классов данной ассоциации. Сам же ромб соединяется с символами классов сплошными линиями. Обычно линии проводятся от вершин ромба или от середины его сторон. Имя n-арной ассоциации записывается рядом с ромбом соответствующей ассоциации. Однако порядок классов в n-арной ассоциации, в отличие от порядка множеств в отношении, на диаграмме не фиксируется

Отношение обобщения

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

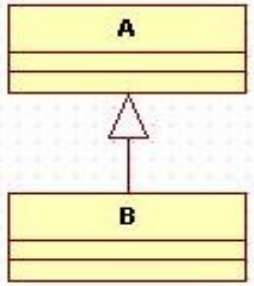
Согласно одному из главных принципов методологии ООП - наследованию, класс-потомок обладает всеми свойствами и поведением класса-предка, а также имеет собственные свойства и поведение, которые могут отсутствовать у класса-предка.

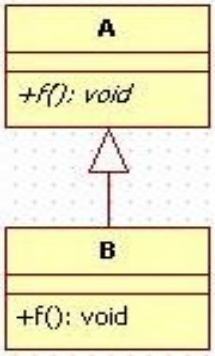
Родитель, предок (parent) - в отношении обобщения более общий элемент. Потомок (child) - специализация одного из элементов отношения обобщения, называемого в этом случае родителем.

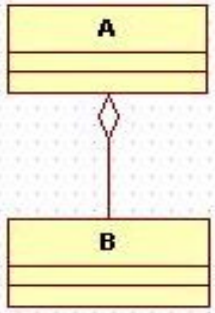


Отношения классов на языке UML

Описание	*UML*	*C#*

Производный класс		public class A
		{
		}
		
		public class B: A
		{
		}

Переопределение абстрактного метода в производном классе		public class A
		{
		public abstract void f();
		}
		
		public class B: A
		{
		override public void f(){}
		}

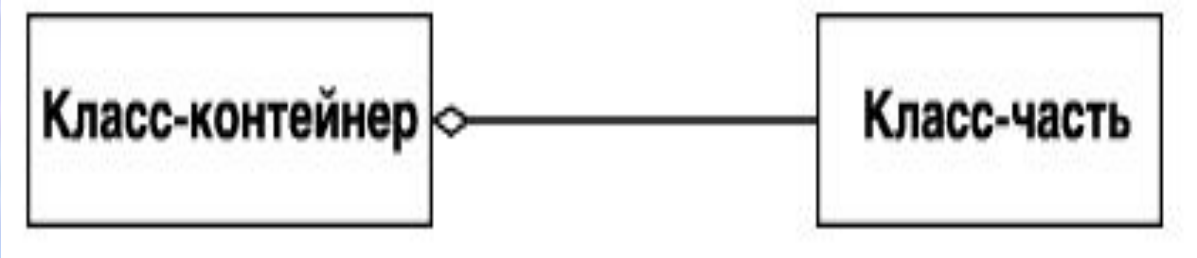
Класс A содержит ссылку на класс B		Например,
		
		public class A
		{}
		
		public class B
		{
		A a = new A();
		}

Пример отношения обобщения

Отношение агрегации

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

Графически отношение агрегации изображается сплошной линией, один из концов которой представляет собой не закрашенный внутри ромб. Этот ромб указывает на тот класс, который представляет собой "целое" или класс-контейнер. Остальные классы являются его "частями“



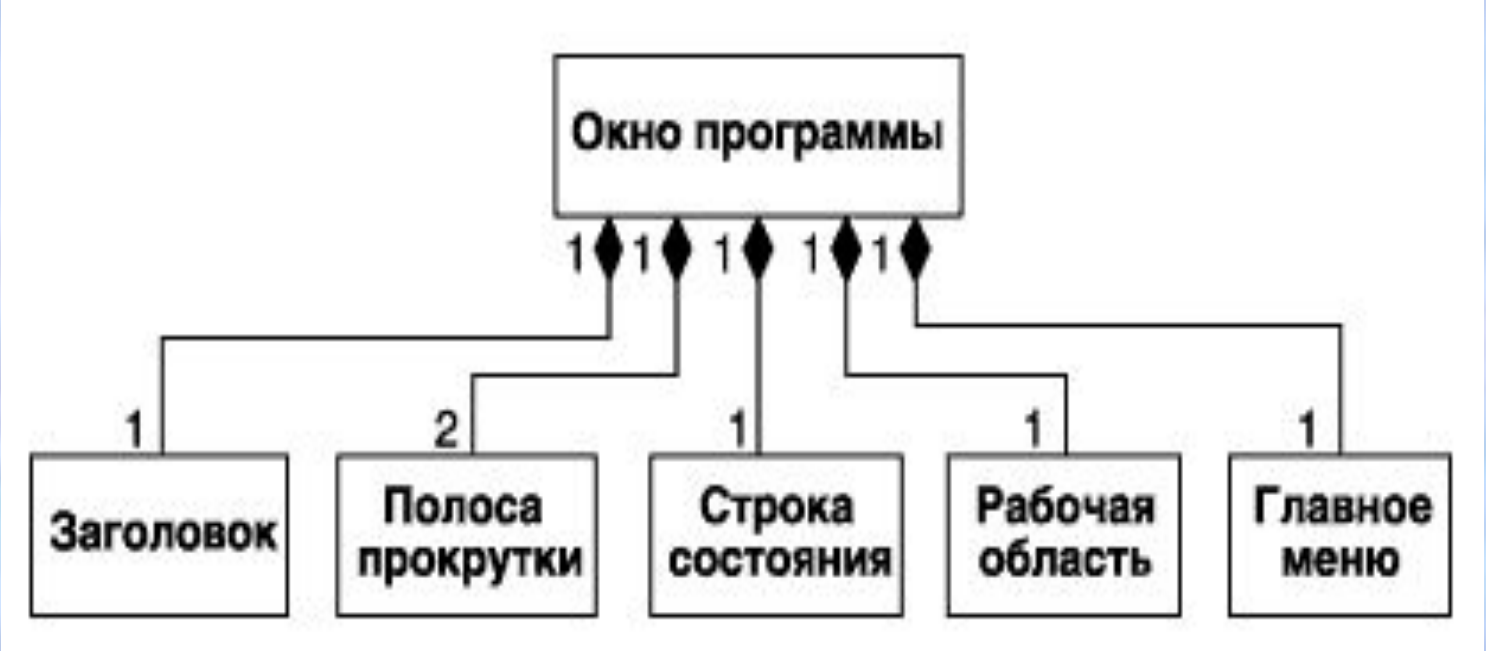
Пример отношения агрегации

Отношение композиции

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

Графически отношение композиции изображается сплошной линией, один из концов которой представляет собой закрашенный внутри ромб. Этот ромб указывает на тот класс, который представляет собой класс-композит. Остальные классы являются его "частями"

Пример отношения композиции



Для отношений композиции и агрегации могут использоваться дополнительные обозначения, применяемые для отношения ассоциации. А именно, могут указываться кратности отдельных классов, которые в общем случае не обязательны.

Пример диаграммы классов

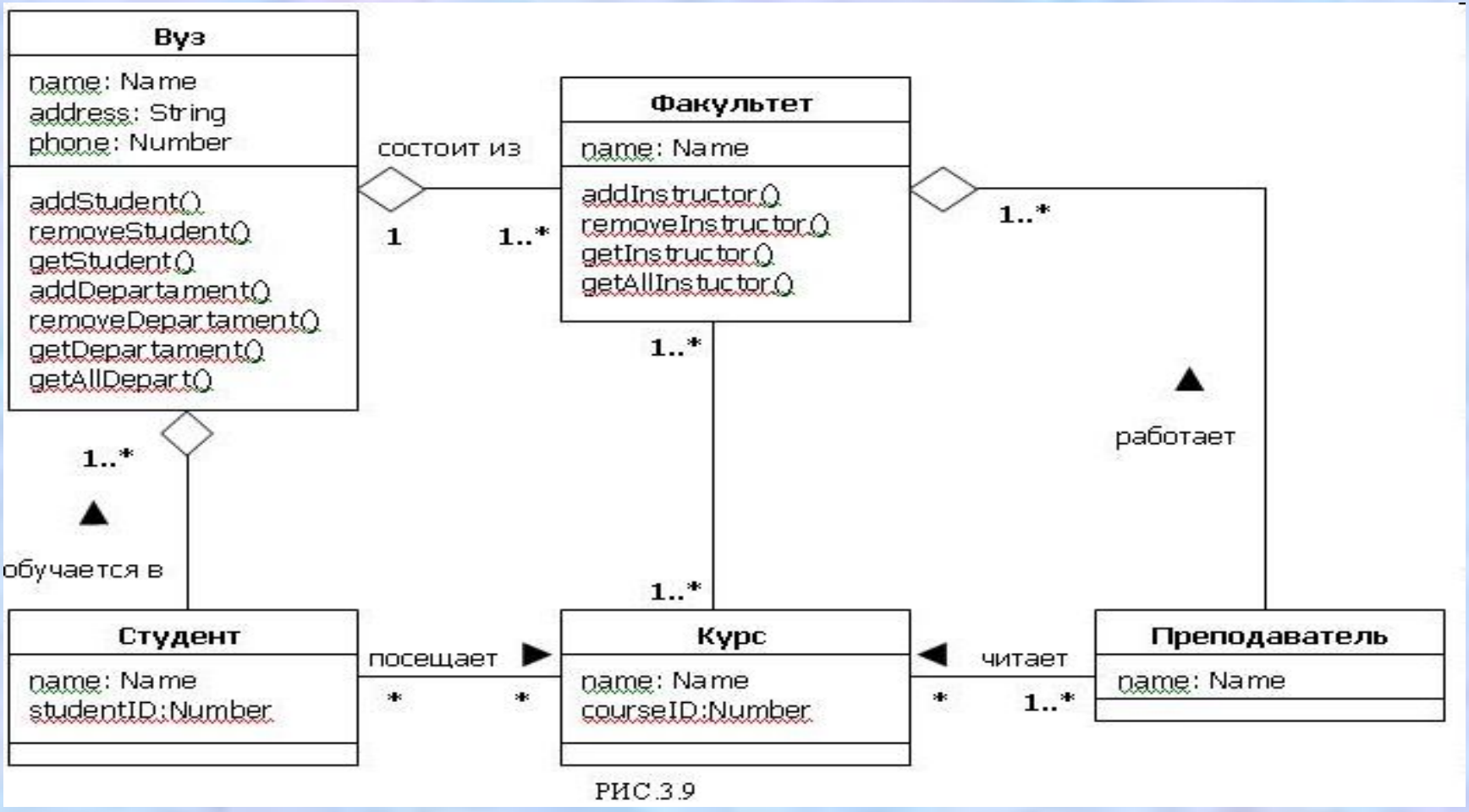


РИС.3.9

Диаграммы последовательности (Sequence Diagram)

Диаграммы последовательности отражают временную последовательность событий, происходящих в рамках варианта использования.

Диаграмма последовательностей

Диаграмма последовательностей **относится к диаграммам взаимодействия UML, описывающим поведенческие аспекты системы и** ***рассматривает взаимодействие объектов во времени*****. Другими словами,** ***диаграмма последовательностей отображает временные особенности передачи и приема сообщений объектами***.

Обозначения:

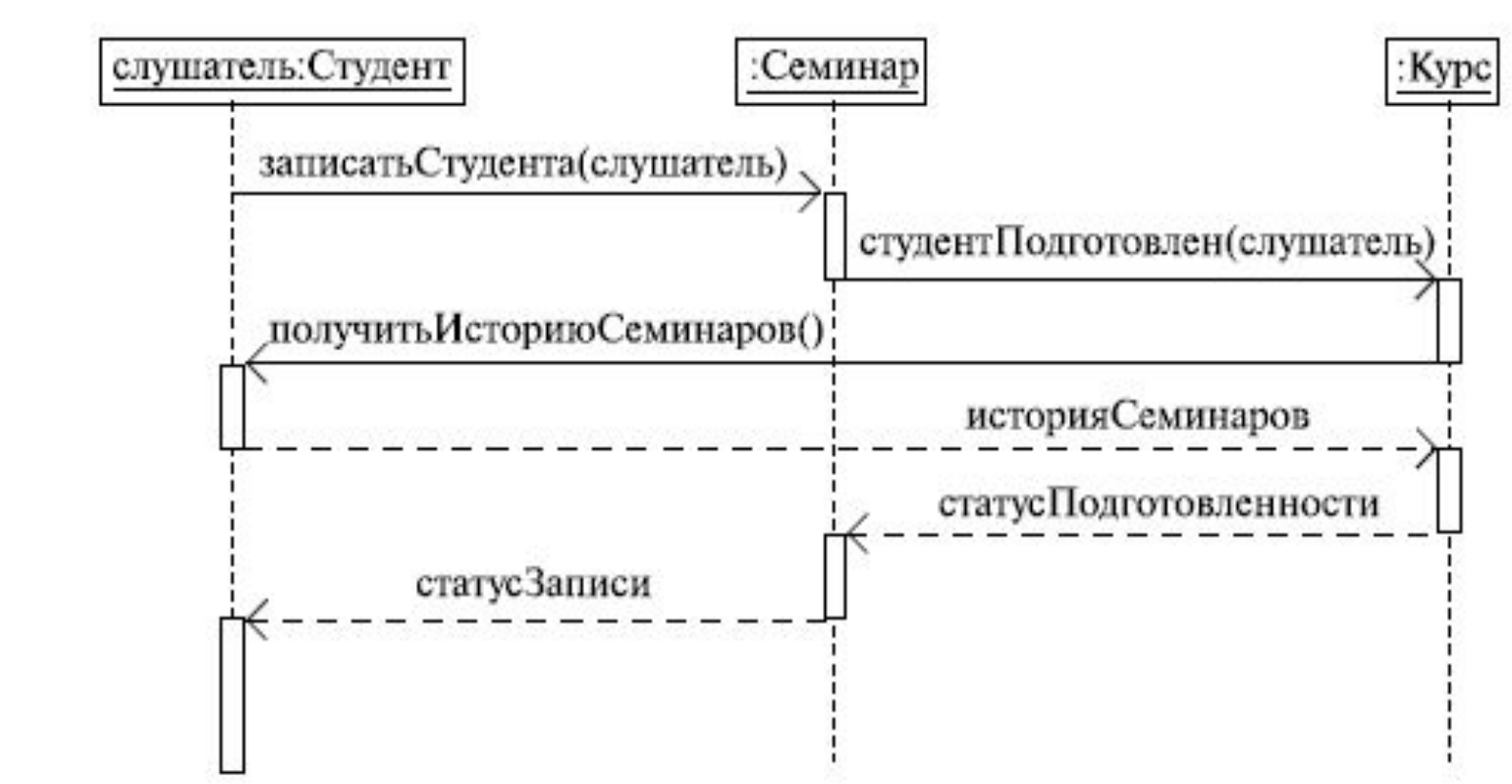
объекты - прямоугольники с подчеркнутыми именами (чтобы отличить их от классов),

сообщения (вызовы методов) – линии со стрелками,

возвращаемые результаты – пунктирные линии со стрелками.

Прямоугольники на вертикальных линиях под каждым из объектов показывают "время жизни" (фокус) объектов.

Пример диаграммы последовательностей



Студент хочет записаться на некий семинар, предлагаемый в рамках некоторого учебного курса. С этой целью проводится проверка подготовленности студента, для чего запрашивается список (история) семинаров курса, уже пройденных студентом (перейти к следующему семинару можно, лишь проработав материал предыдущих занятий. После получения истории семинаров объект класса "Семинар" получает статус подготовленности, на основе которой студенту сообщается результат (статус) его попытки записи на семинар

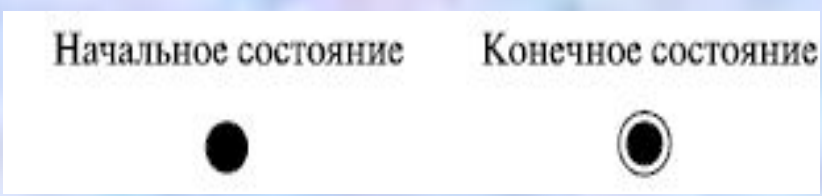
Диаграмма активности (деятельности, activity diagram)

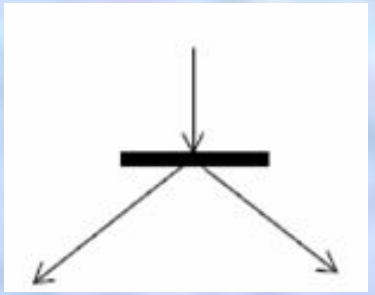
- Назначение: моделирование процесса выполнения операций в языке UML

- На диаграмме деятельности отображается логика или последовательность перехода от одной деятельности к другой, диаграмма фокусируется на потоке действий, вовлечённых в процесс и показывает как действия зависят друг от друга

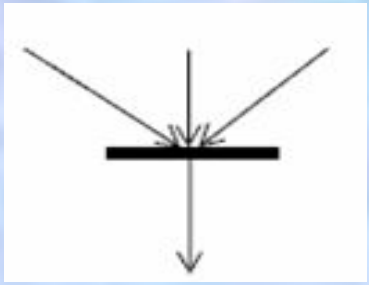
Пример диаграммы активности

Обозначения



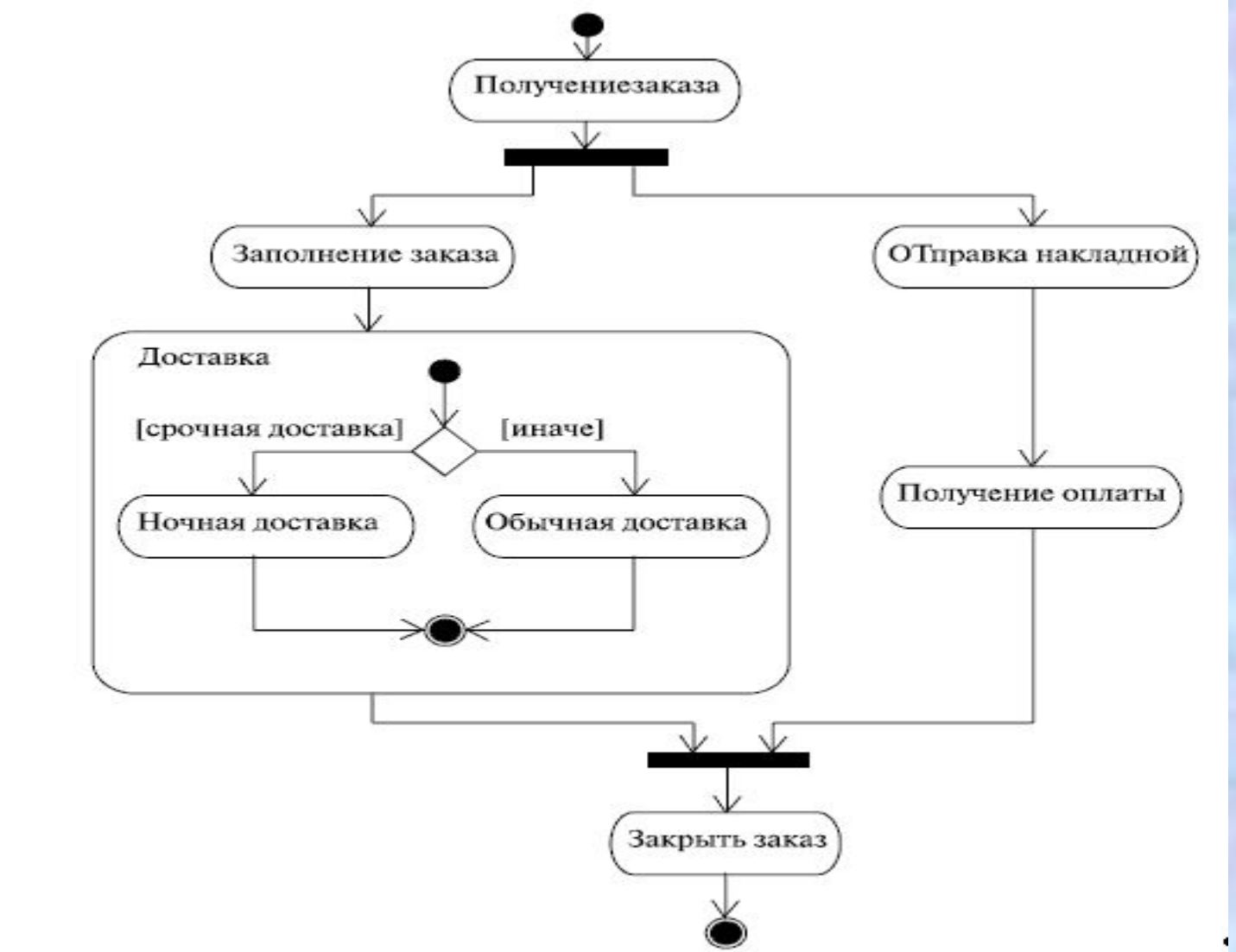


Разделение потоков деятельности



Слияние потоков деятельности

Пример диаграммы активности



Рекомендации по использованию диаграмм деятельности

- На практике диаграммы деятельности применяются в основном двумя способами:
- **Для моделирования процессов**
- В этом случае внимание фокусируется на деятельности с точки зрения экторов, которые работают с системой.

- **Для моделирования операций**
- В этом случае диаграммы деятельности играют роль "продвинутых" блок-схем и применяются для подробного моделирования вычислений. На первое место при таком использовании выходят конструкции принятия решения, а также разделения и слияния потоков управления (*синхронизации*).

ЗАКЛЮЧЕНИЕ

Язык UML предлагает набор изобразительных средств, позволяющих проводить всесторонний анализ сложных проектов как с технической точки зрения, так и с точки зрения потребностей бизнеса.

Данный язык упрощает процесс проектирования, снижает его стоимость и повышает эффективность.

Что обеспечивает UML

1. **иерархическое описание сложной системы путем выделения пакетов;**
2. **формализацию функциональных требований к системе с помощью аппарата вариантов использования;**
3. **детализацию требований к системе путем построения диаграмм деятельностей и сценариев;**
4. **выделение классов данных и построение концептуальной модели данных в виде диаграмм классов;**
5. **выделение классов, описывающих пользовательский интерфейс, и создание схемы навигации экранов;**
6. **описание процессов взаимодействия объектов при выполнении системных функций;**
7. **описание поведения объектов в виде диаграмм деятельностей и состояний;**
8. **описание программных компонент и их взаимодействия через интерфейсы;**
9. **описание физической архитектуры системы.**