Учебный курс

# МЕТОДЫ И СРЕДСТВА ПРОЕКТИРОВАНИЯ ПО

Лекция 7

## Краткая характеристика диаграмм UML

# Назначение основных видов диаграмм:

- прецедентов;
- классов;
- объектов;
- последовательностей;
- взаимодействия;
- состояний;
- **АКТИВНОСТИ**;
- развертывания.

# Диаграмма прецедентов

# (use case diagram)

# Диаграмма прецедентов. Эктор

- **Эктор (actor)** - это множество логически связанных ролей, исполняемых при взаимодействии с прецедентами или сущностями (система, подсистема или класс). Эктором может быть человек или другая система, подсистема или класс, которые представляют нечто вне сущности.
- Графически эктор изображается либо "*человечком* ", либо *символом класса с соответствующим стереотипом*, как показано на рисунке

# Прецедент

- ***Прецедент (use-case)*** *- описание отдельного аспекта поведения системы с точки зрения пользователя* (Буч).
- Определение вполне понятное и исчерпывающее, но его можно еще немного уточнить, воспользовавшись тем же *Zicom Mentor* 'ом:
- **Прецедент (use case)** - описание множества последовательных событий (включая варианты), выполняемых системой, которые приводят к наблюдаемому эктором результату. Прецедент представляет поведение сущности, описывая взаимодействие между экторами и системой. Прецедент не показывает, "как" достигается некоторый результат, а только "что" именно выполняется.
- Прецеденты обозначаются очень простым образом - в виде эллипса, внутри которого указано его название. *Прецеденты и экторы соединяются с помощью линий*. Часто на одном из концов линии изображают *стрелку*, причем *направлена она к тому, у кого запрашивают сервис*, другими словами, чьими услугами пользуются. Это простое объяснение иллюстрирует *понимание прецедентов как сервисов*, пропагандируемое компанией IBM.

# Диаграмма прецедентов для описания использования библиотек

# Диаграмма прецедентов для описания процесса обучения студента

# Диаграммы прецедентов. Итоги.

- *диаграммы прецедентов* относятся к той группе диаграмм, которые представляют статические и поведенческие аспекты системы. Это отличное *средство для достижения взаимопонимания между разработчиками, экспертами и конечными пользователями* продукта.
- просты для понимания и восприятия
- могут обсуждаться людьми, не являющимися специалистами в области разработки ПО.

# 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.

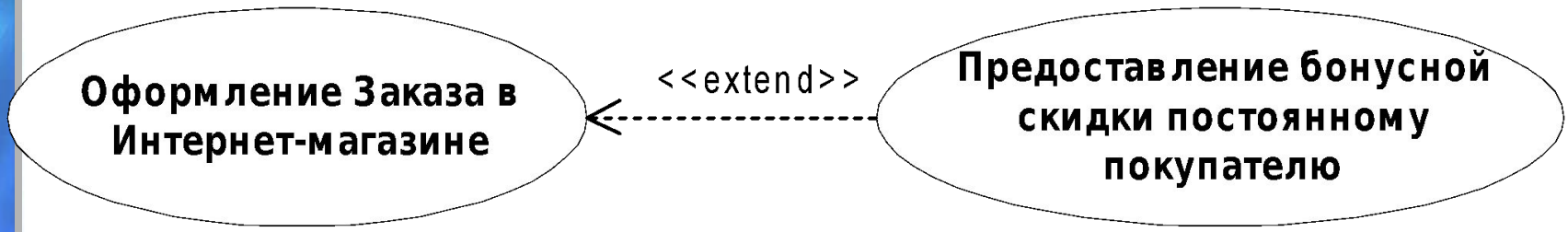
# Диаграмма прецедентов
# (use case diagram)
# Отношение зависимости

расширяемый

источник зависимости
(независимый элемент)

расширяющий

клиент зависимости
(зависимый элемент)




вариант использования *А*

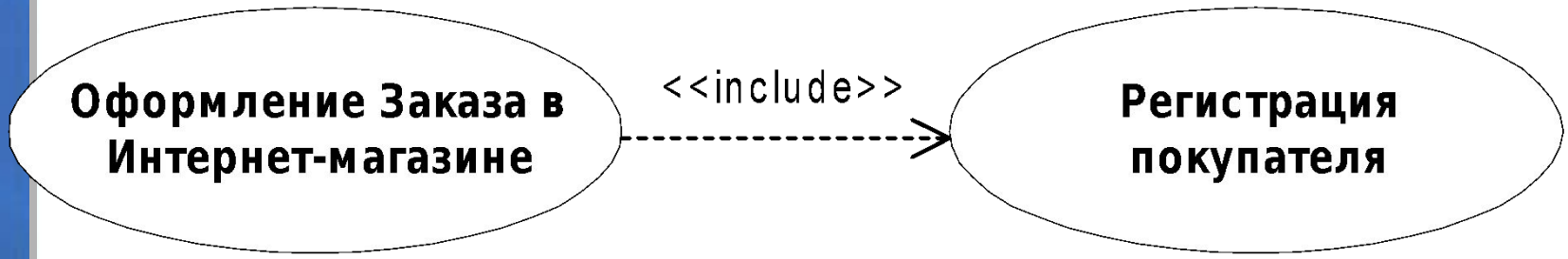
вариант использования *Б*

включающий

клиент зависимости
(зависимый элемент)

включаемый

источник зависимости
(независимый элемент)



# Диаграмма классов

(class diagram)

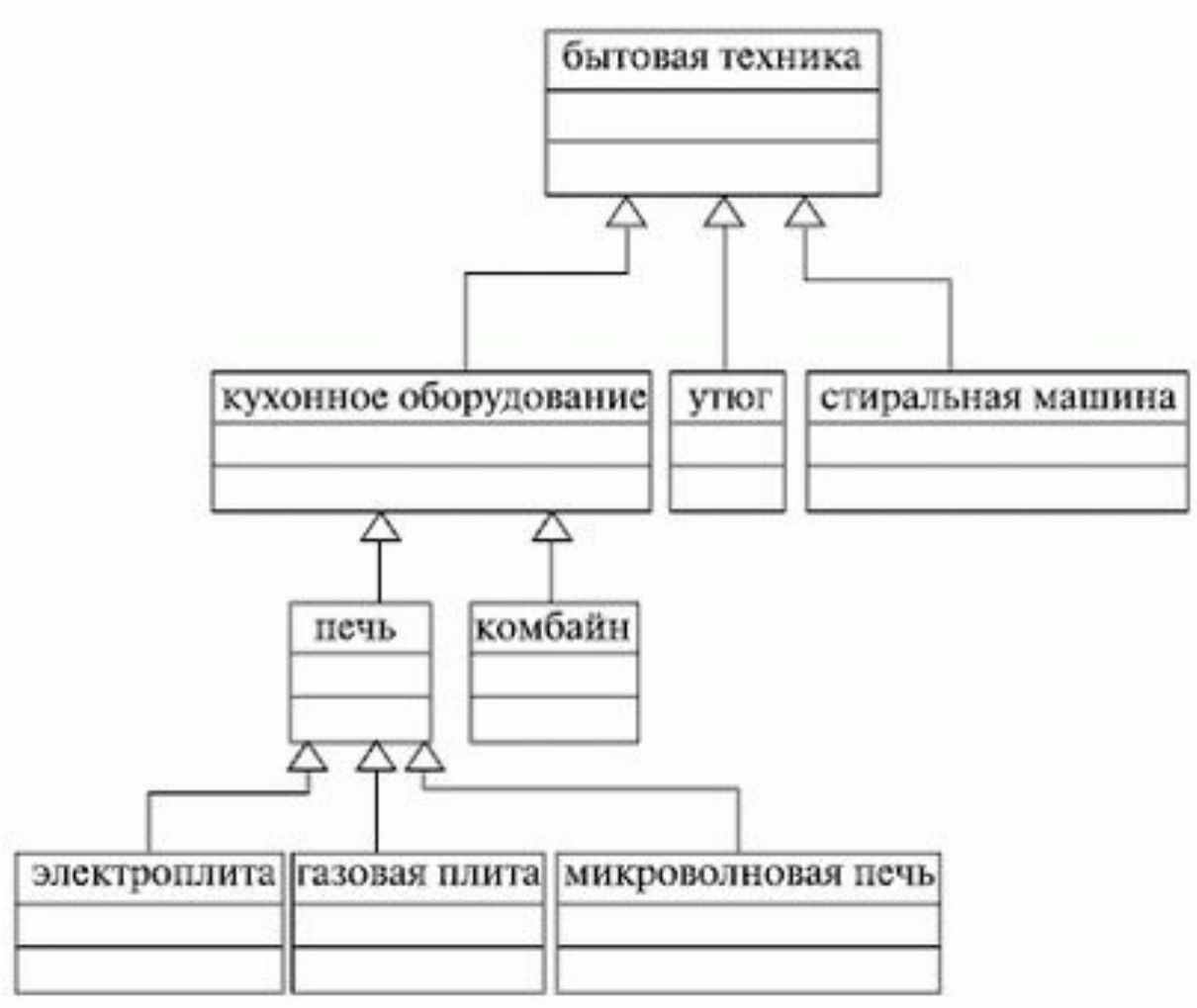
# Диаграмма классов (class diagram)

- ***Класс (class)*** *- категория вещей, которые имеют общие атрибуты и операции.*(Буч)
- Классы - *это строительные блоки любой объектно-ориентированной системы*. Они представляют собой описание совокупности объектов с общими атрибутами, операциями, отношениями и семантикой. При проектировании объектно-ориентированных систем диаграммы классов обязательны. Классы используются в процессе анализа предметной области для составления *словаря предметной области* разрабатываемой системы. Это могут быть как абстрактные понятия предметной области, так и классы, на которые опирается разработка и которые описывают программные или аппаратные сущности.

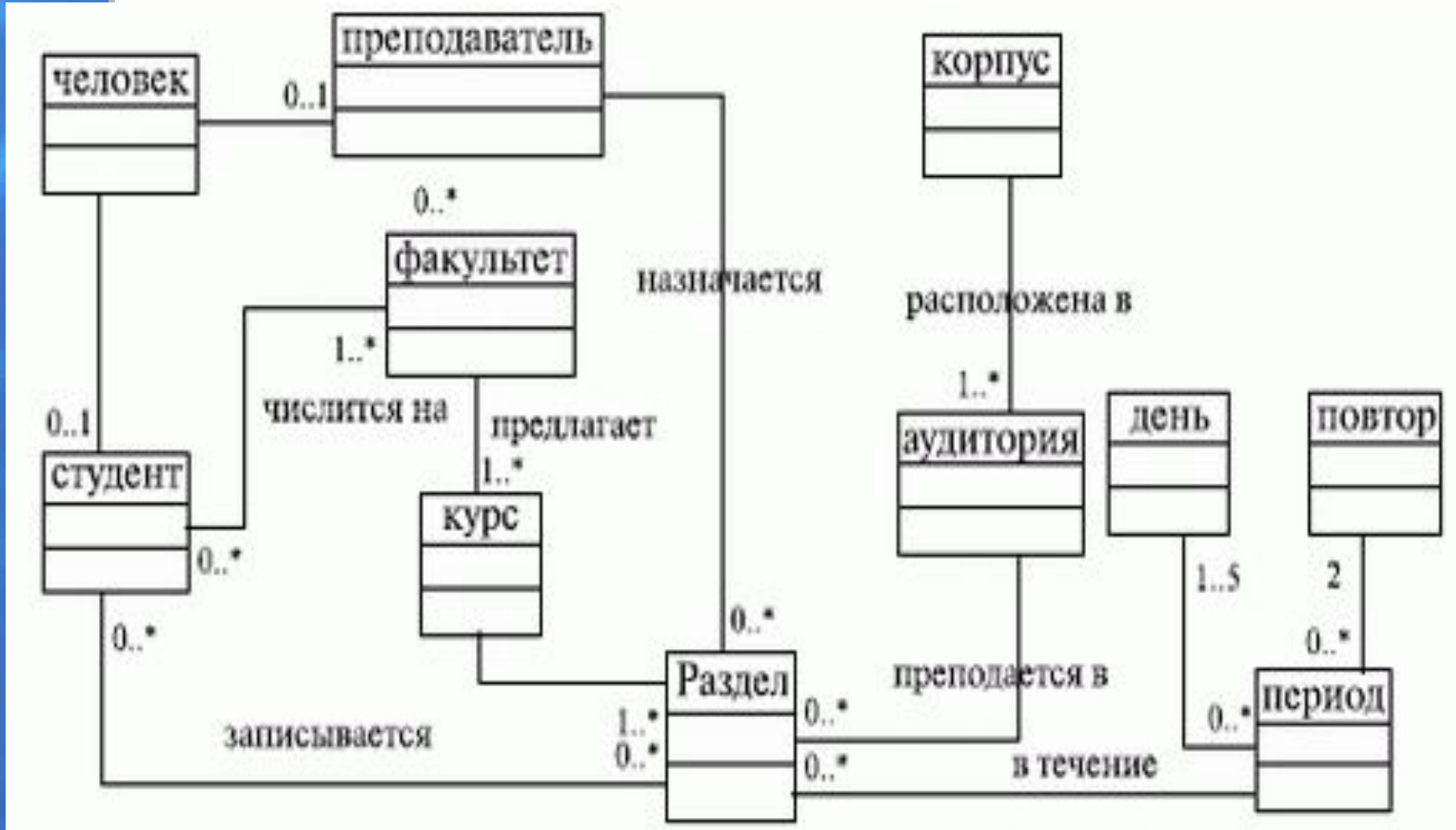
# Диаграмма классов (class diagram)

- Диаграмма классов - это *набор статических*, декларативных *элементов модели*. Диаграммы классов могут применяться и при прямом проектировании, то есть в процессе разработки новой системы, и при *обратном проектировании* - описании существующих и используемых систем. Информация с диаграммы классов напрямую отображается в исходный код приложения - в большинстве существующих инструментов UML-моделирования возможна *кодогенерация* для определенного языка программирования (обычно Java или C++).
- Таким образом, диаграмма классов - конечный результат проектирования и отправная точка процесса разработки.

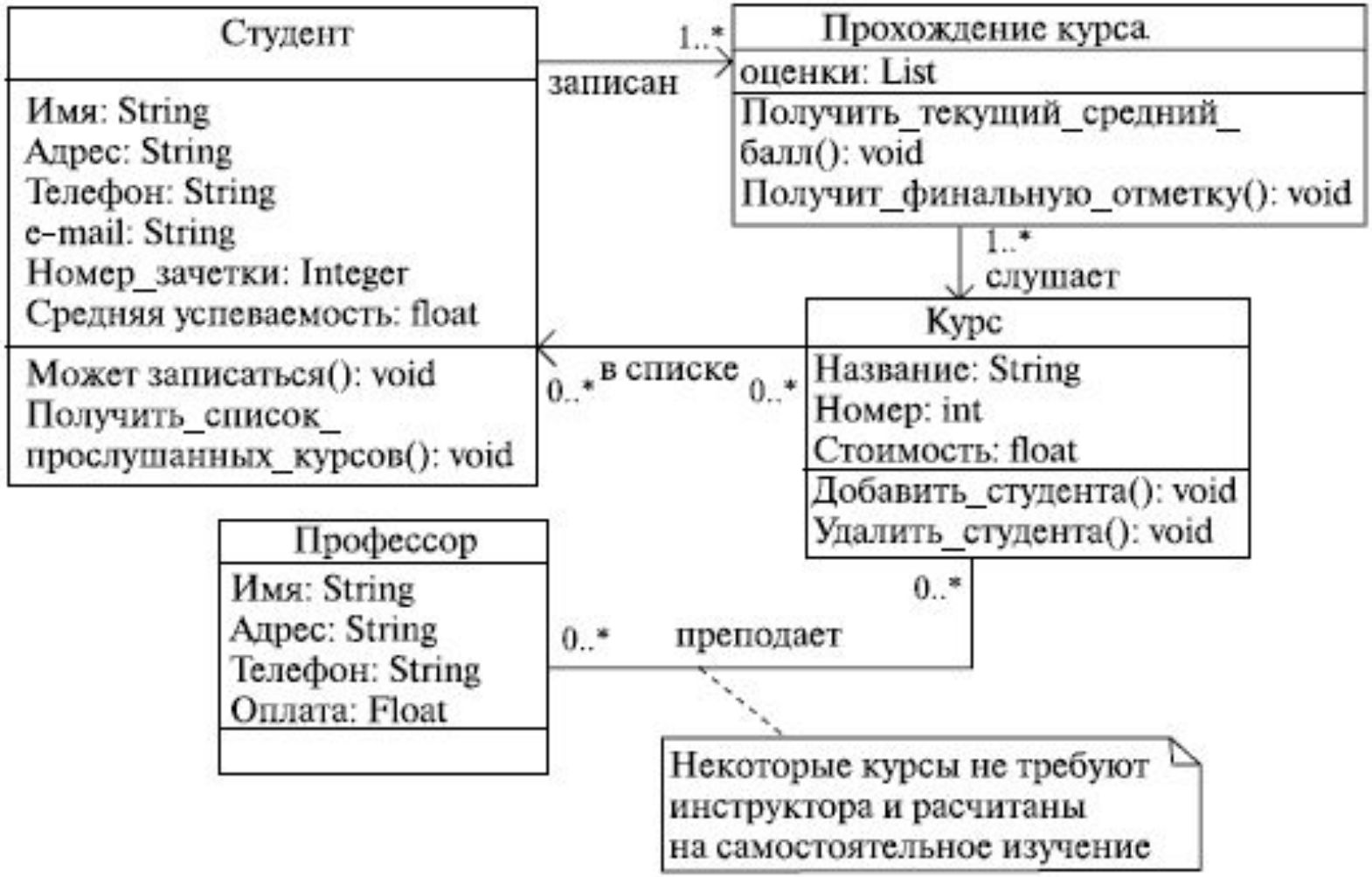
# Пример "генеалогического древа" бытовой техники на диаграмме классов.



# Пример автоматизации работы учебного центра на диаграмме классов.



# Пример распределения студентов по курсам ВУЗа на диаграмме классов.



# Диаграмма объектов (object diagram)

# Диаграмма объектов (object diagram)

1) ***Объект (object)*** - *экземпляр класса* (Буч)

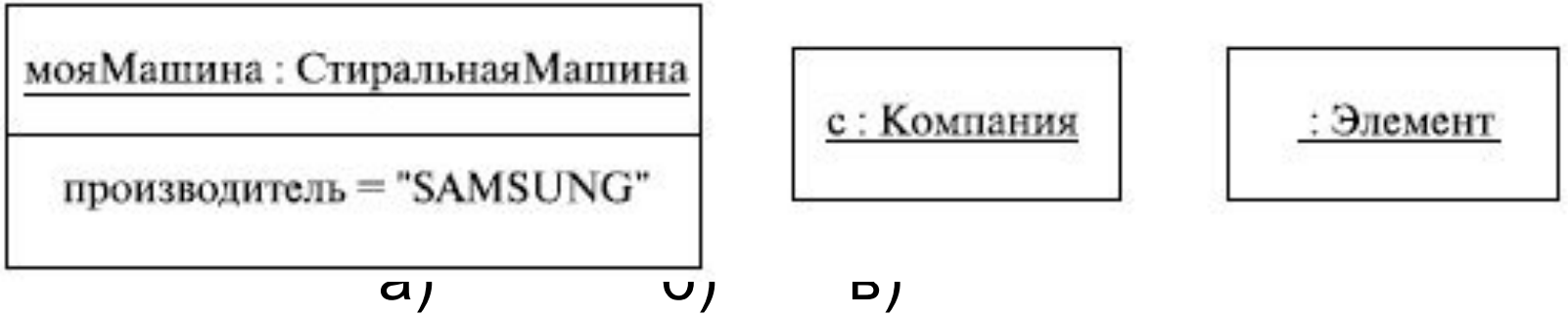
2) **Объект (object)** (Zicom Mentor):

- конкретная материализация абстракции;
- сущность с хорошо определенными границами, в которой инкапсулированы состояние и поведение;
- экземпляр класса (вернее, классификатора - эктор, класс или интерфейс). Объект уникально идентифицируется значениями атрибутов, определяющими его состояние в данный момент времени.

"Второе" определение расширяет "Бучевское".

# Диаграмма объектов

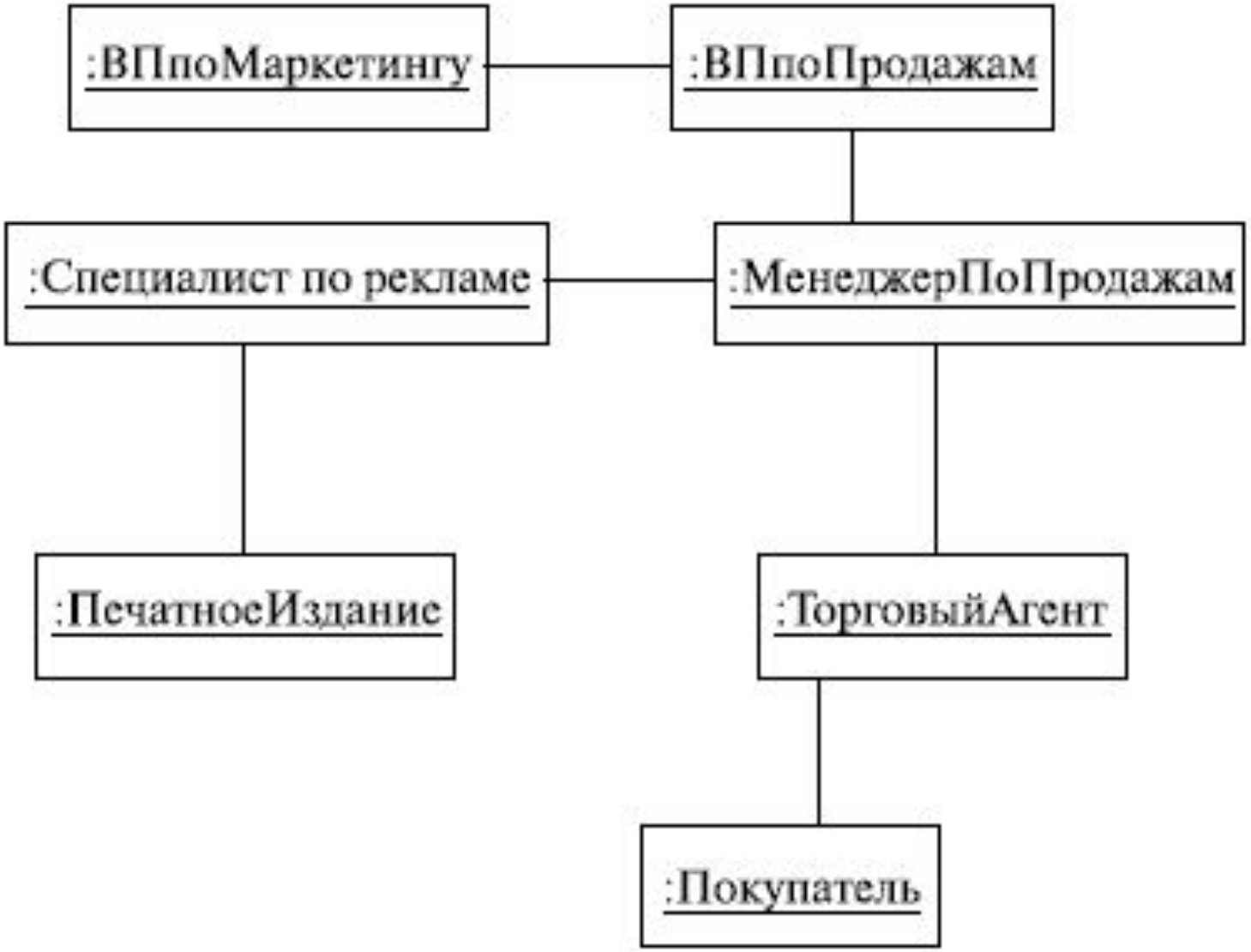
- Объект, как и класс, обозначается прямоугольником, но его имя подчеркивается (б).
- Имя - это название объекта и наименование его класса, разделенные двоеточием. Для указания значений атрибутов объекта в его обозначении может быть предусмотрена специальная секция (а). Объект может быть анонимным (в): это нужно в том случае, если в данный момент не важно, какой именно объект данного класса принимает участие во взаимодействии. Примеры



# Для чего нужны диаграммы объектов?

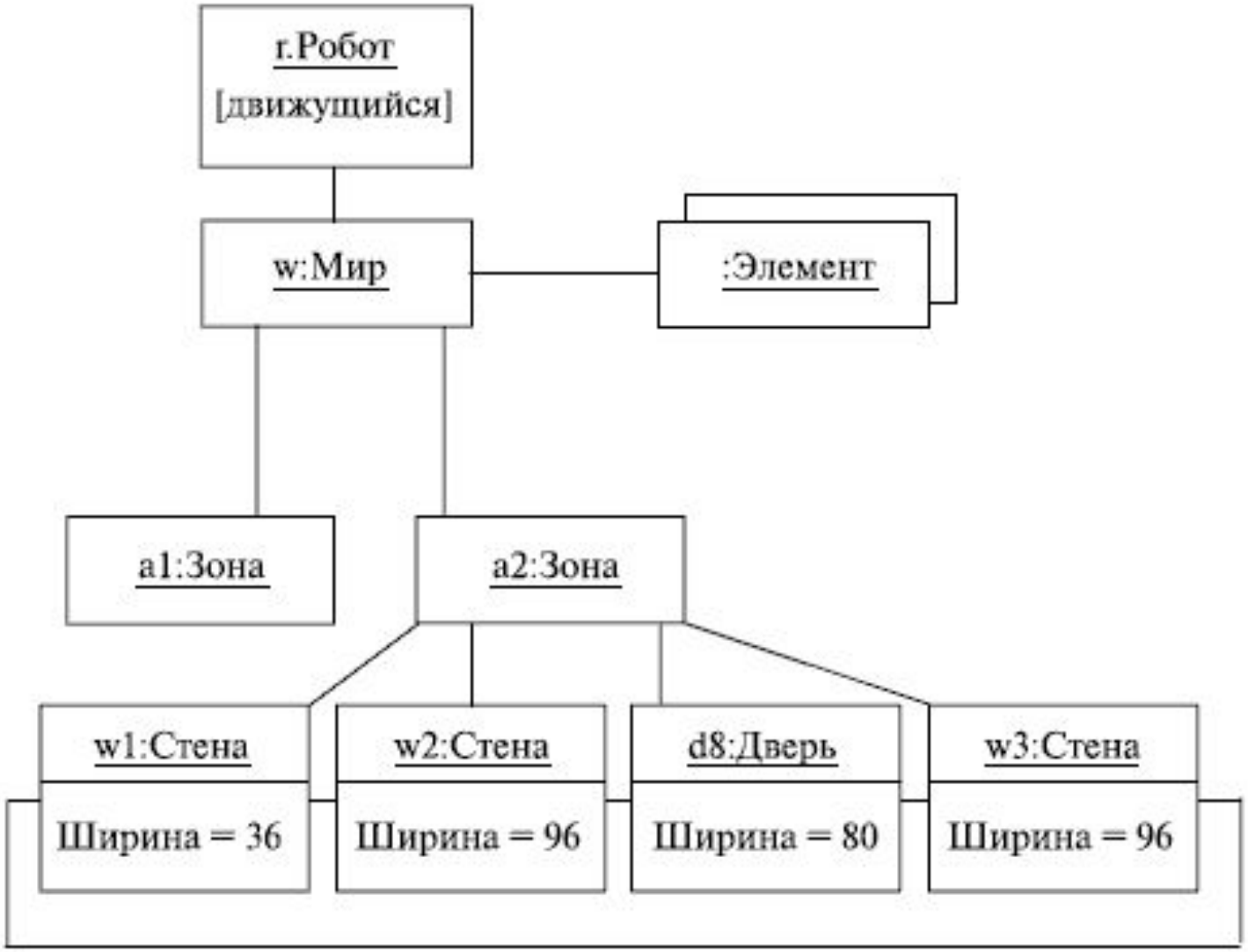
- Они показывают множество объектов - экземпляров классов (изображенных на диаграмме классов) и отношений между ними в некоторый момент времени. То есть *диаграмма объектов - это своего рода снимок состояния системы в определенный момент времени*, показывающий множество объектов, их состояния и отношения между ними в данный момент.
- Таким образом, диаграммы объектов представляют *статический вид системы с точки зрения проектирования и процессов*, являясь основой для сценариев, описываемых диаграммами взаимодействия. Говоря другими словами, *диаграмма объектов используется для пояснения и детализации диаграмм взаимодействия*, например, диаграмм последовательностей.

# Пример диаграммы объектов при "раскручивании" нового товара



# Пример диаграммы объектов организационных единиц в некоторой компании

# Пример диаграммы объектов учебной среды "Робот" для Turbo Pascal



# Диаграмма последовательностей (sequence diagram)

# Диаграмма последовательностей (sequence diagram)

- **Диаграмма последовательностей** *отображает взаимодействие объектов в динамике*.
- В UML взаимодействие объектов понимается как обмен информацией между ними. При этом информация принимает вид сообщений. *Сообщение несет какую-то информацию и некоторым образом также влияет на получателя*. *UML полностью соответствует основным принципам ООП*, в соответствии с которыми информационное взаимодействие между объектами сводится к отправке и приему сообщений.
- *Диаграмма последовательностей* относится к диаграммам взаимодействия UML, описывающим поведенческие аспекты системы, но *рассматривает взаимодействие объектов во времени*. Другими словами, *диаграмма последовательностей отображает временные особенности передачи и приема сообщений объектами*.

# Диаграмма последовательностей

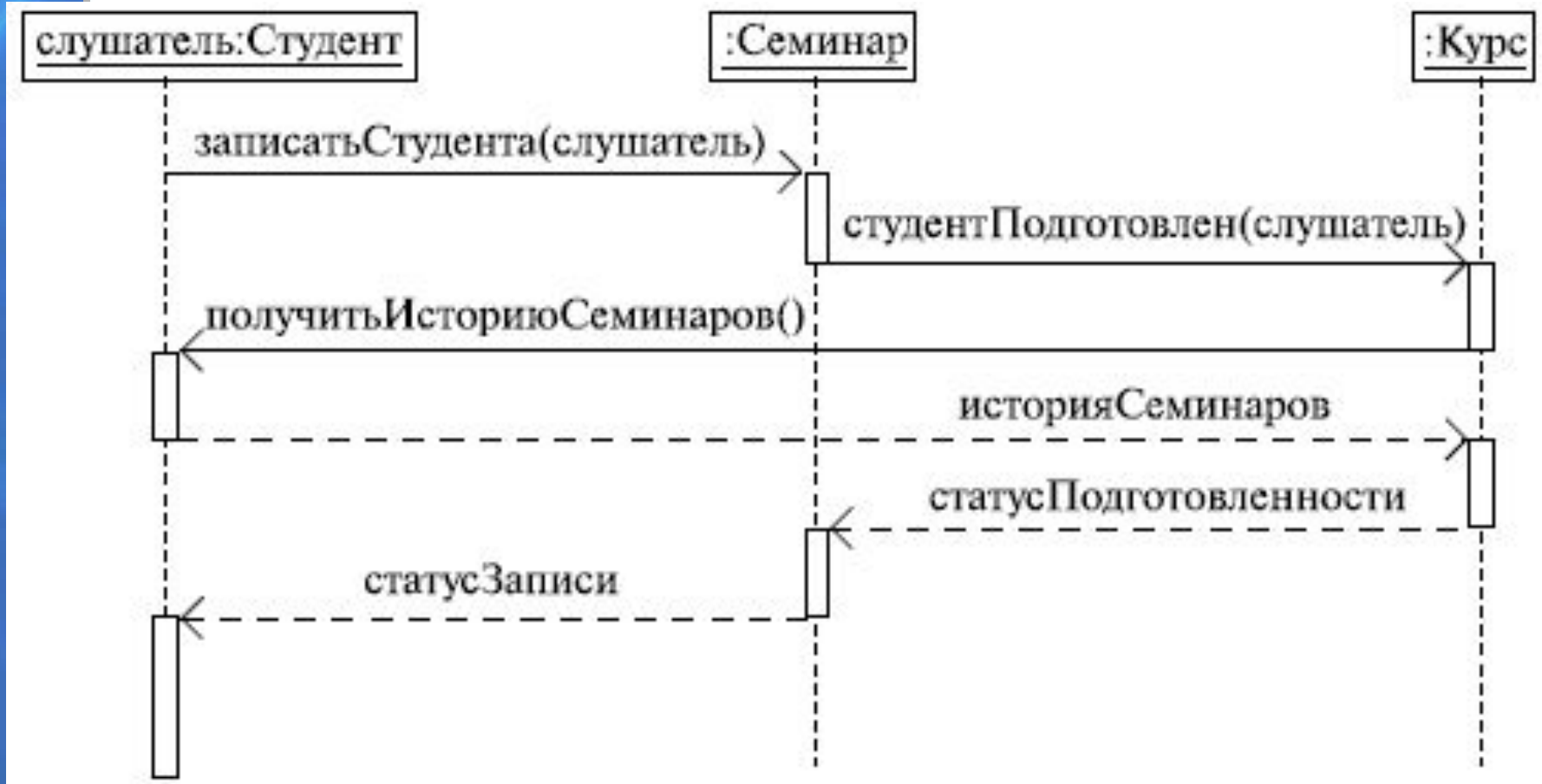
- *Диаграммы последовательностей можно (и нужно!) использовать для уточнения диаграмм прецедентов*, более детального описания логики сценариев использования. Это отличное средство документирования проекта с точки зрения сценариев использования. Диаграммы последовательностей обычно содержат
- *объекты*, которые взаимодействуют в рамках сценария,
- *сообщения*, которыми они обмениваются,
- *возвращаемые результаты*, связанные с сообщениями (их обозначают лишь в том случае, если это не очевидно из контекста).

# Обозначения на диаграмме последовательностей

- Объекты обозначаются прямоугольниками с подчеркнутыми именами (чтобы отличить их от классов),
- сообщения (вызовы методов) - линиями со стрелками,
- возвращаемые результаты - пунктирными линиями со стрелками.

Прямоугольники на вертикальных линиях под каждым из объектов показывают "время жизни" (фокус) объектов. Впрочем, довольно часто их не изображают на диаграмме, все это зависит от индивидуального стиля проектирования.

# Пример диаграммы последовательностей записи студент на семинар, предлагаемый в рамках некоторого учебного курса.



# Пример диаграммы последовательностей работы домового лифта

# Пример диаграммы последовательностей работы мобильного телефона

# Диаграмма взаимодействия

(collaboration diagram)

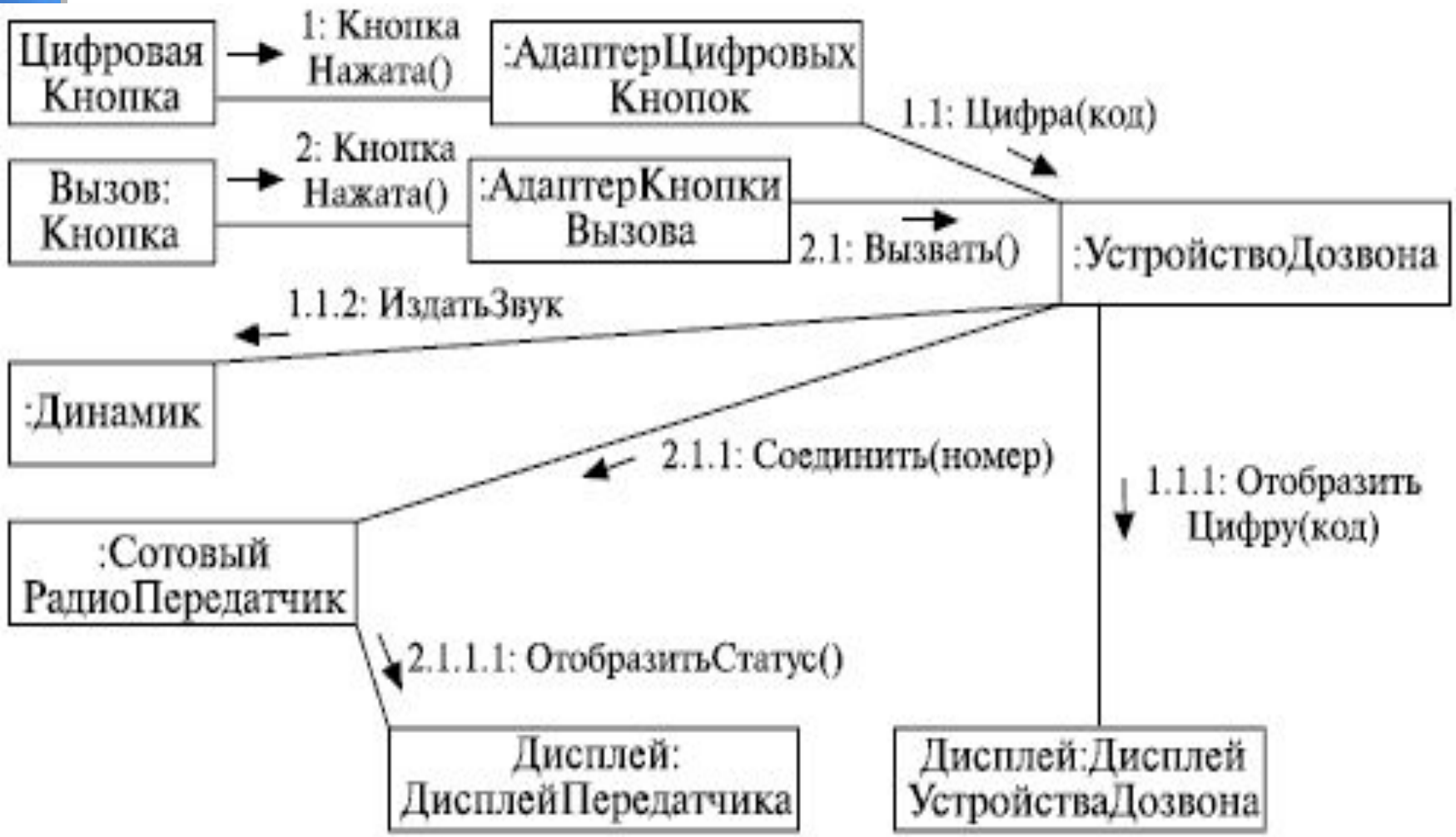
# Диаграмма взаимодействия (кооперации, collaboration diagram)

- Диаграмма взаимодействия показывает поток сообщений между объектами системы и основные ассоциации между ними и по сути, как уже было сказано выше, является альтернативой диаграммы последовательностей.
- Объекты обозначаются прямоугольниками с подчеркнутыми именами (чтобы отличить их от классов), ассоциации между объектами указываются в виде соединяющих их линий, над ними может быть изображена стрелка с указанием названия сообщения и его порядкового номера. Последовательность передачи сообщений можно указать только с помощью их нумерации. В этом и состоит вероятная причина пренебрежения этим видом диаграмм многими проектировщиками.

# Пример диаграммы взаимодействия работы персонала библиотеки по обслуживанию клиентов

# Пример диаграммы взаимодействия в процессе управления учебными курсами ВУЗа

# Пример диаграммы взаимодействия описания работы мобильного телефона



# Диаграмма состояний

(statechart diagram)

# Диаграмма состояний (statechart diagram)

- *Объекты характеризуются поведением и состоянием*, в котором находятся. Например, человек может быть новорожденным, младенцем, ребенком, подростком или взрослым. Другими словами, объекты что-то делают и что-то "знают". **Диаграммы состояний** *применяются для того, чтобы объяснить, каким образом работают сложные объекты*. Несмотря на то что смысл понятия "состояние" интуитивно понятен, все же приведем его определение в таком виде, в каком его дают классики и Zicom Mentor:
- **Состояние (state)** - ситуация в жизненном цикле объекта, во время которой он удовлетворяет некоторому условию, выполняет определенную деятельность или ожидает какого-то события. Состояние объекта определяется значениями некоторых его атрибутов и присутствием или отсутствием связей с другими объектами.

# Диаграмма состояний

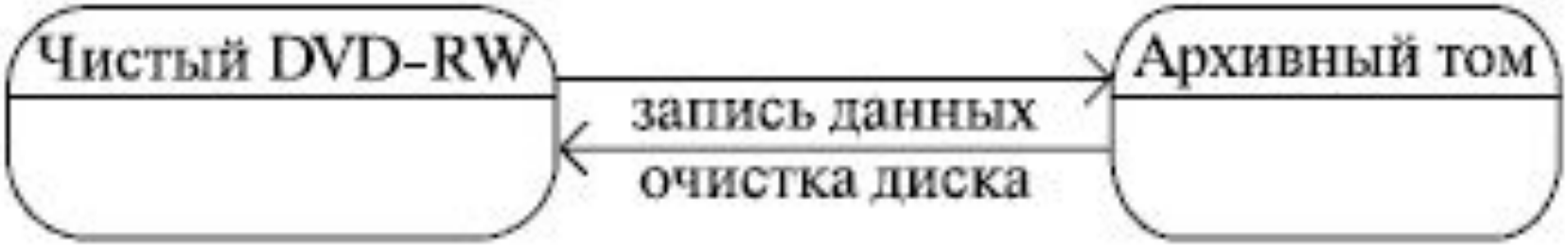
- показывает, как объект переходит из одного состояния в другое.
- служит для моделирования динамических аспектов системы (как и диаграммы последовательностей, кооперации, прецедентов и деятельности).
- часто можно услышать, что *она показывает автомат*.
- полезна при моделировании жизненного цикла объекта (как и ее частная разновидность - диаграмма деятельности).

# Диаграмма состояний

- От других диаграмм она отличается тем, что описывает процесс изменения состояний только одного экземпляра определенного класса - одного объекта, причем объекта *реактивного*, то есть объекта, поведение которого характеризуется его реакцией на внешние события. Понятие жизненного цикла применимо как раз к реактивным объектам, настоящее состояние (и поведение) которых обусловлено их прошлым состоянием.
- Но диаграммы состояний важны не только для описания динамики отдельного объекта. Они могут использоваться для *конструирования исполняемых систем* путем прямого и обратного проектирования.

# Обозначения на диаграмме состояний

- Скругленные прямоугольники представляют состояния, через которые проходит объект в течение своего жизненного цикла. Стрелками показываются переходы между состояниями, которые вызваны выполнением методов описываемого диаграммой объекта. Пример простейшей диаграммы состояний:



# Обозначения на диаграмме состояний

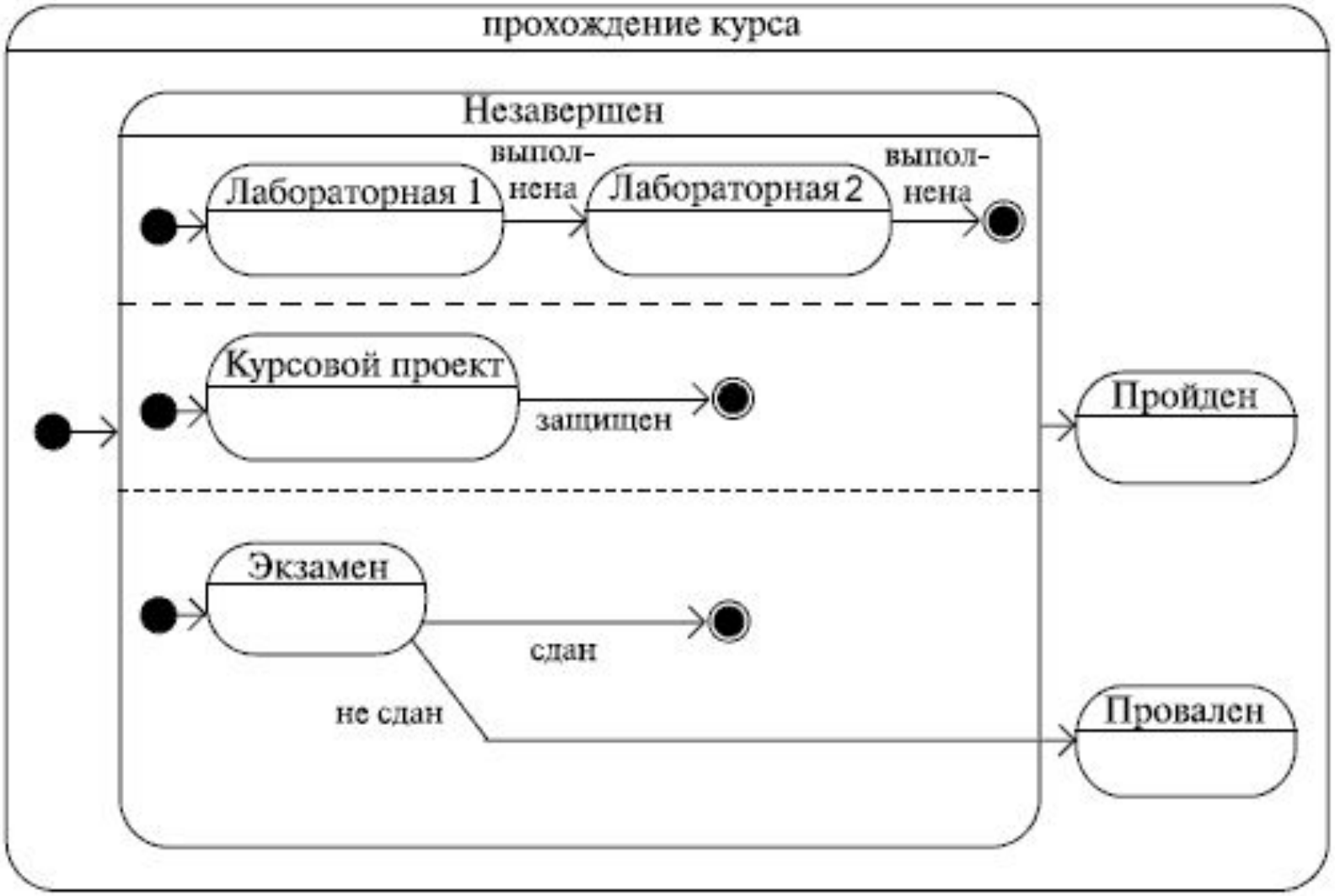
Существует *два вида псевдосостояний:*

- *начальное*, в котором находится объект сразу после его создания (обозначается сплошным кружком),
- и *конечное*, которое объект не может покинуть, если перешел в него (обозначается кружком, обведенным окружностью).

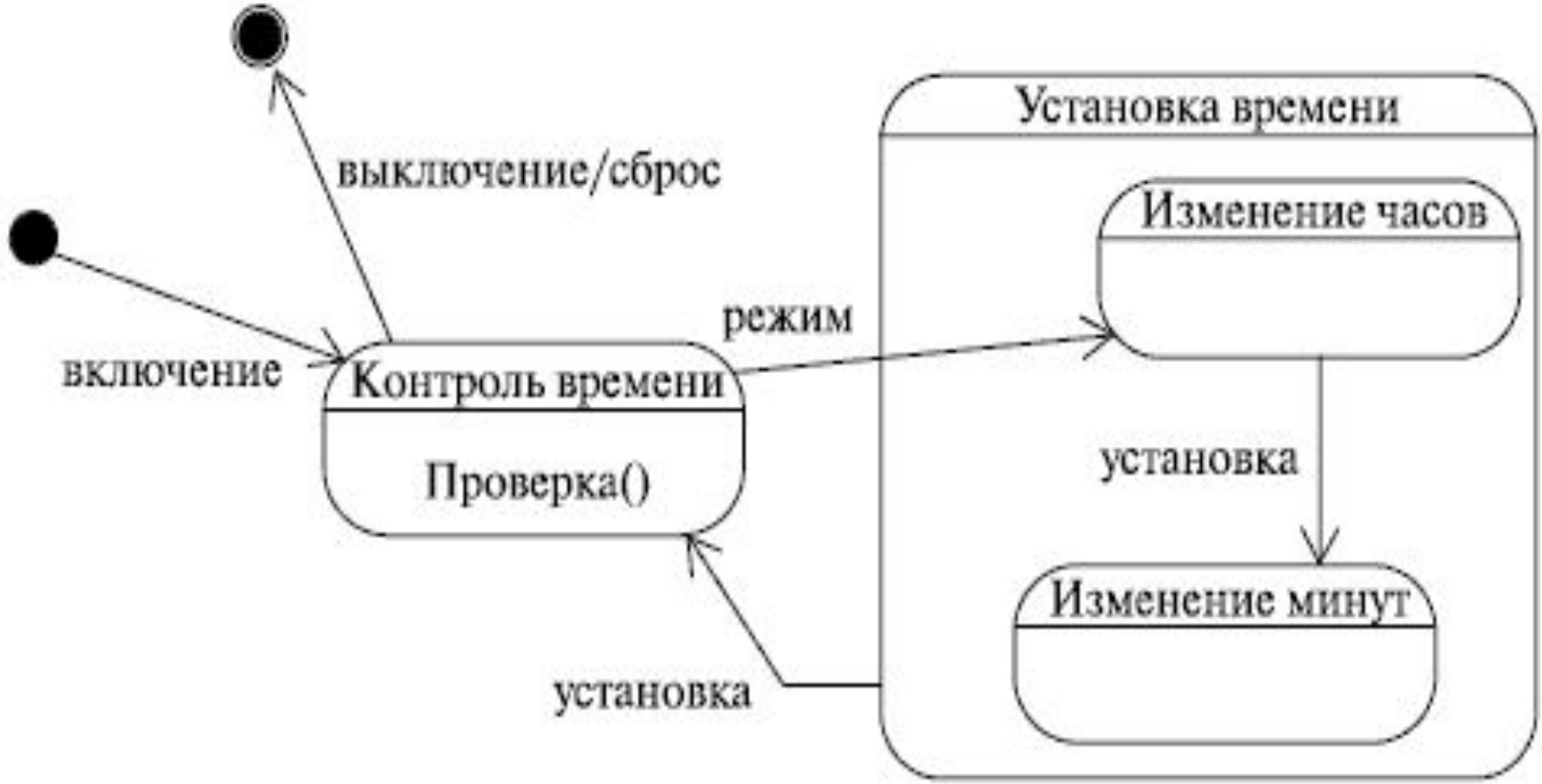
Начальное состояние

Конечное состояние

# Диаграмма прохождения академического курса студентом



# Диаграмма состояний таймера



# Диаграмма активности или деятельностей (activity diagram)

# Диаграмма активности (деятельности, activity diagram)

- Моделируя поведение проектируемой системы, часто недостаточно изобразить процесс смены ее состояний, а нужно также раскрыть детали алгоритмической реализации операций, выполняемых системой. Для этой цели традиционно использовались блок-схемы или структурные схемы алгоритмов.
- В UML для этого существуют **диаграммы деятельности**, являющиеся частным случаем диаграмм состояний. Диаграммы деятельности удобно применять для визуализации алгоритмов, по которым работают операции классов.

# Диаграмма деятельностей

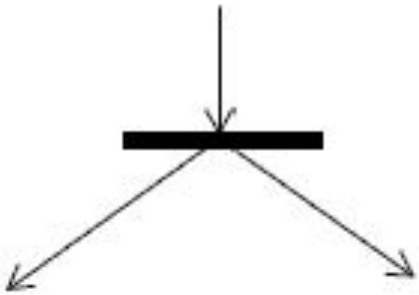
- Понятие деятельности (activity) - протяженное во времени составное (неатомарное) вычисление (действие, action) и переход как передача контроля
- В отличие от диаграмм взаимодействия (акцент делается на переходах потока управления *от объекта к объекту*) диаграммы деятельности описывают переход *от одной деятельности к другой*

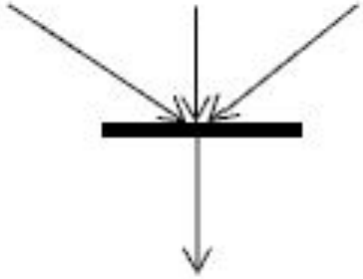
# Диаграмма деятельностей. Примитивы

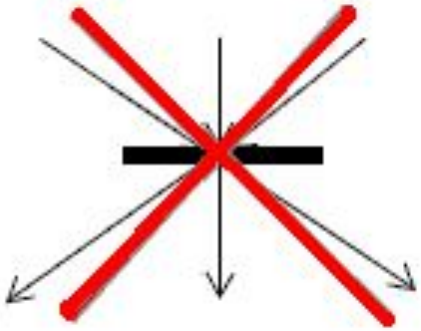
Начальное состояние

Конечное состояние

- *Синхронизация* потоков управления:

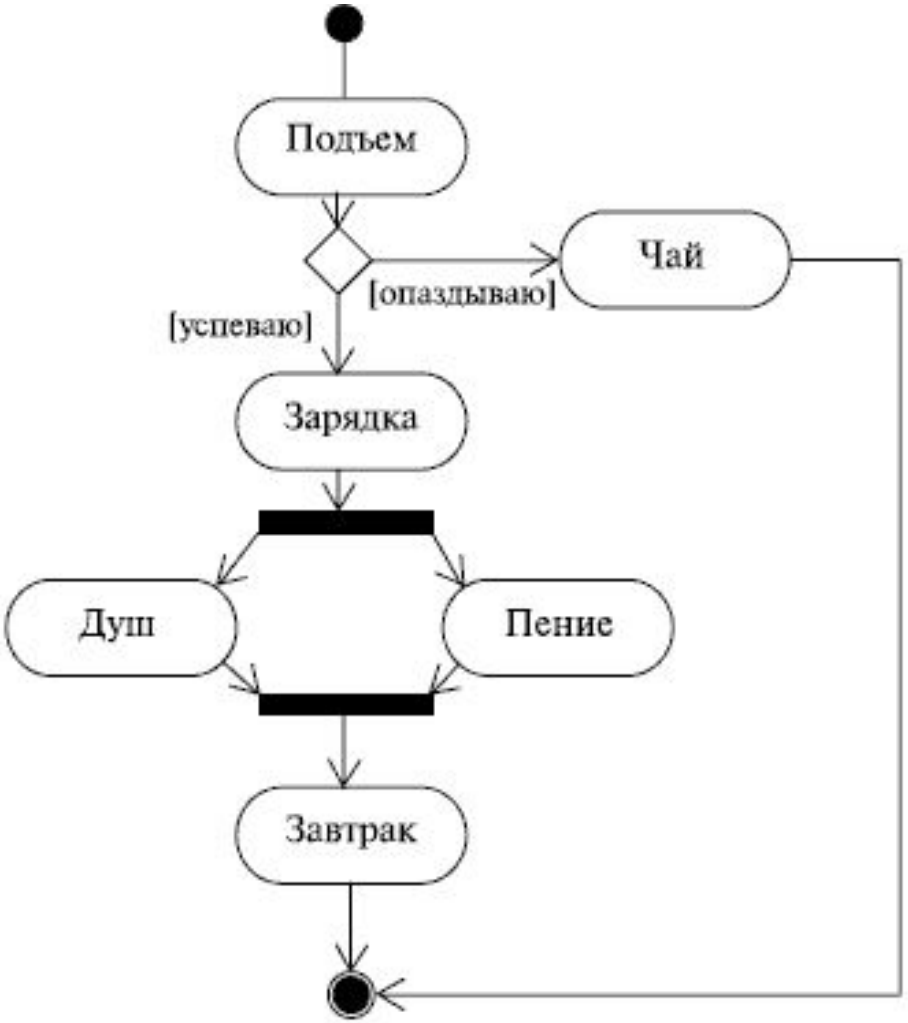






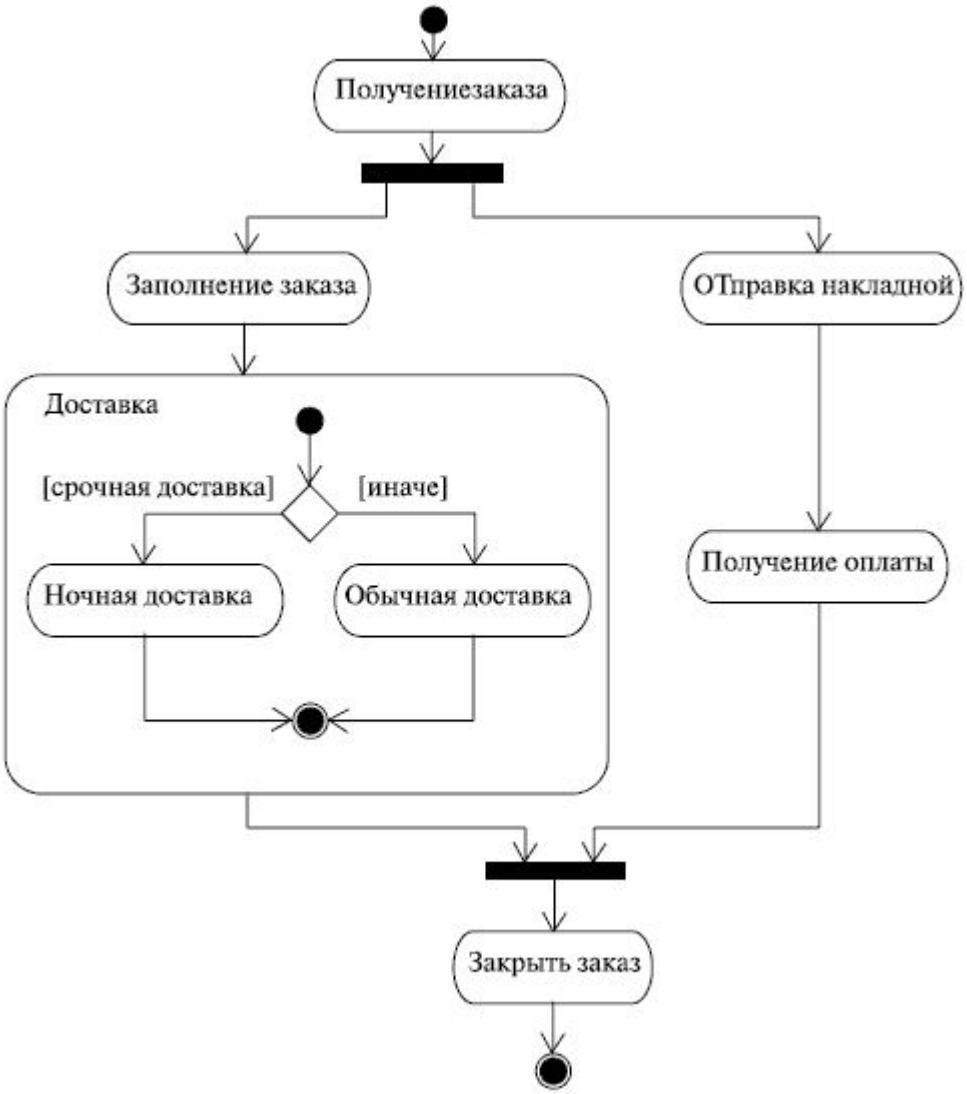
распараллеливание, а затем опять слияние

# Диаграмма активности. Пример.



# Пример оформления заказа в интернет-магазине

# Пример выполнения заказа

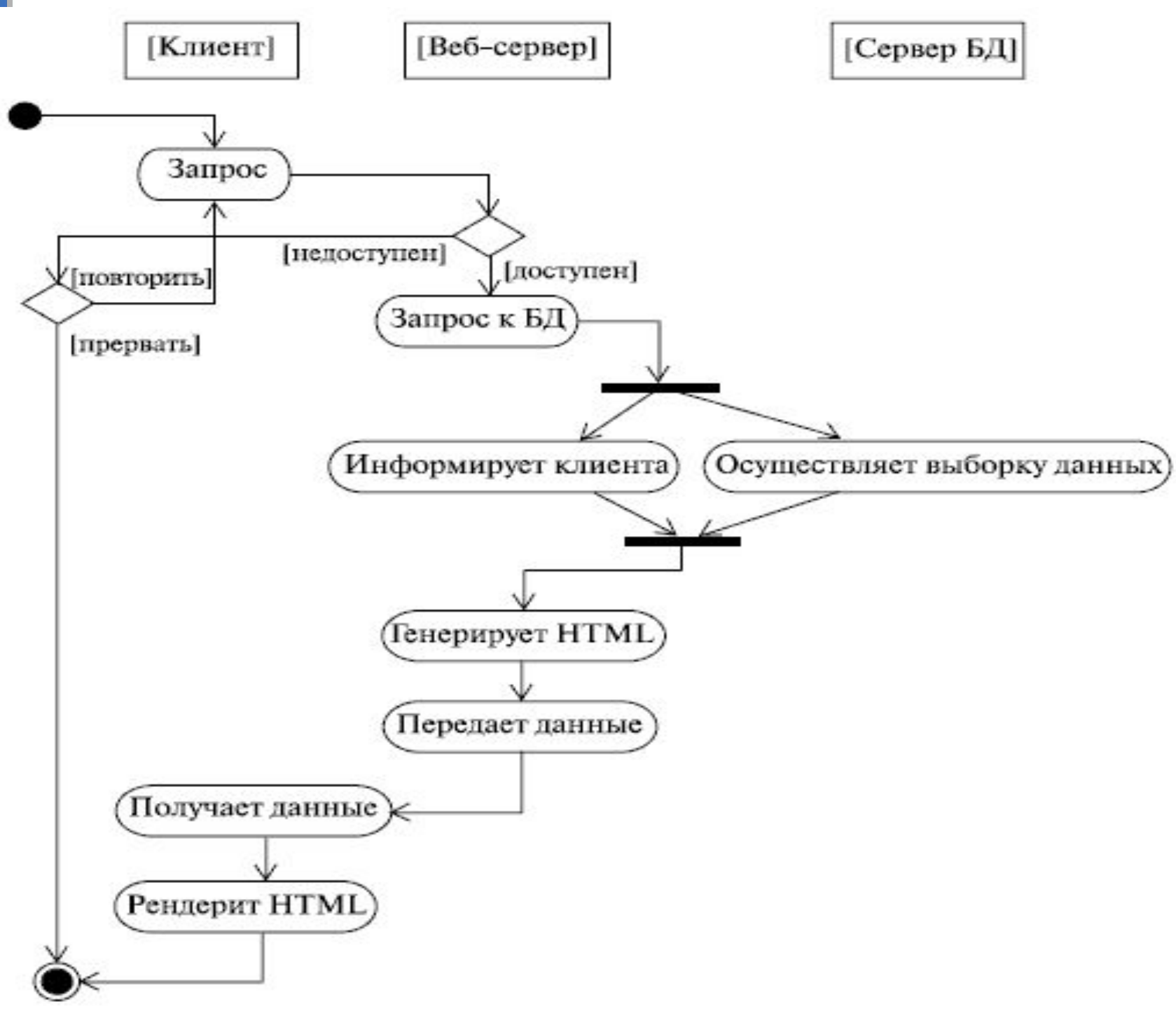


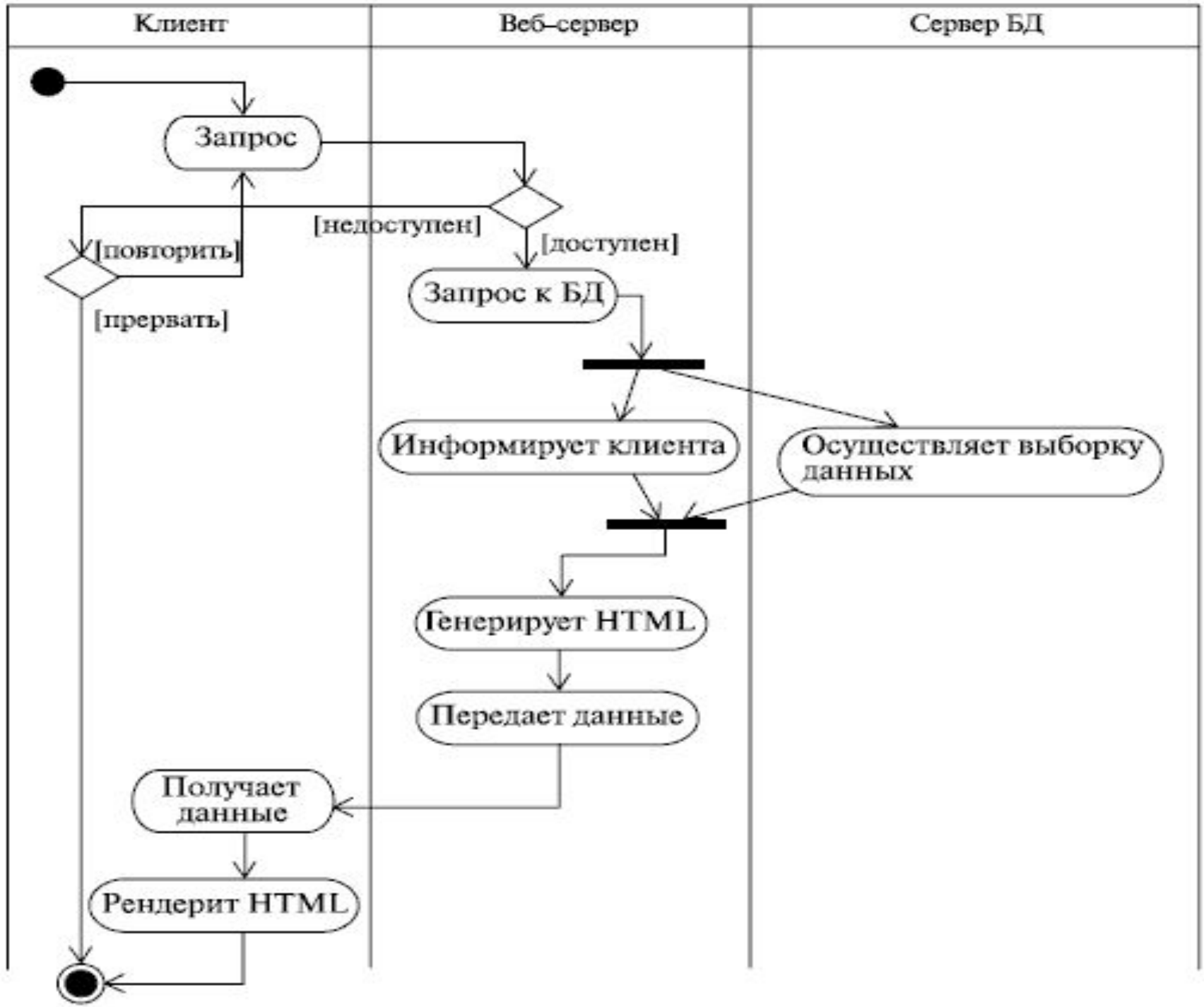
# Диаграмма деятельностей

На диаграмме деятельностей можно не только показать параллельно выполняемые действия, но и

- указать состояния объектов (так же, как и на представлениях конечных автоматов),
- также есть возможность показывать распределение ролей

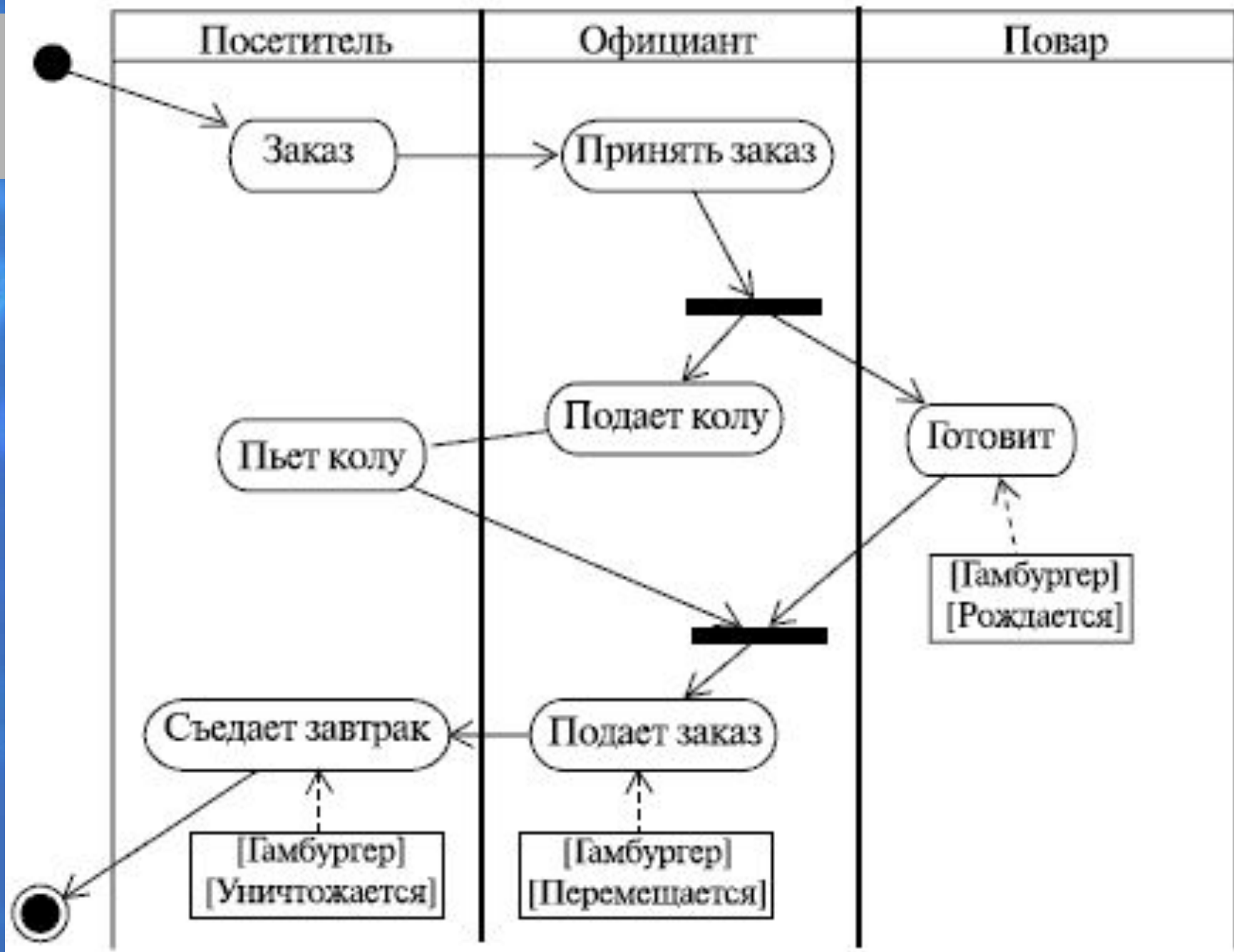
Для этого используется дорожка - часть области диаграммы деятельности, на которой отображаются только те деятельности, за которые отвечает конкретный объект. Поясним на примере.


[Клиент]
[Веб-сервер]
[Сервер БД]
Запрос
[недоступен]
[доступен]
[повторить]
[прервать]
Запрос к БД
Информирует клиента
Осуществляет выборку данных
Генерирует HTML
Передает данные
Получает данные
Рендерит HTML


Клиент
Веб–сервер
Сервер БД
Запрос
[недоступен]
[доступен]
[повторить]
[прервать]
Запрос к БД
Информирует клиента
Осуществляет выборку данных
Генерирует HTML
Передает данные
Получает данные
Рендерит HTML

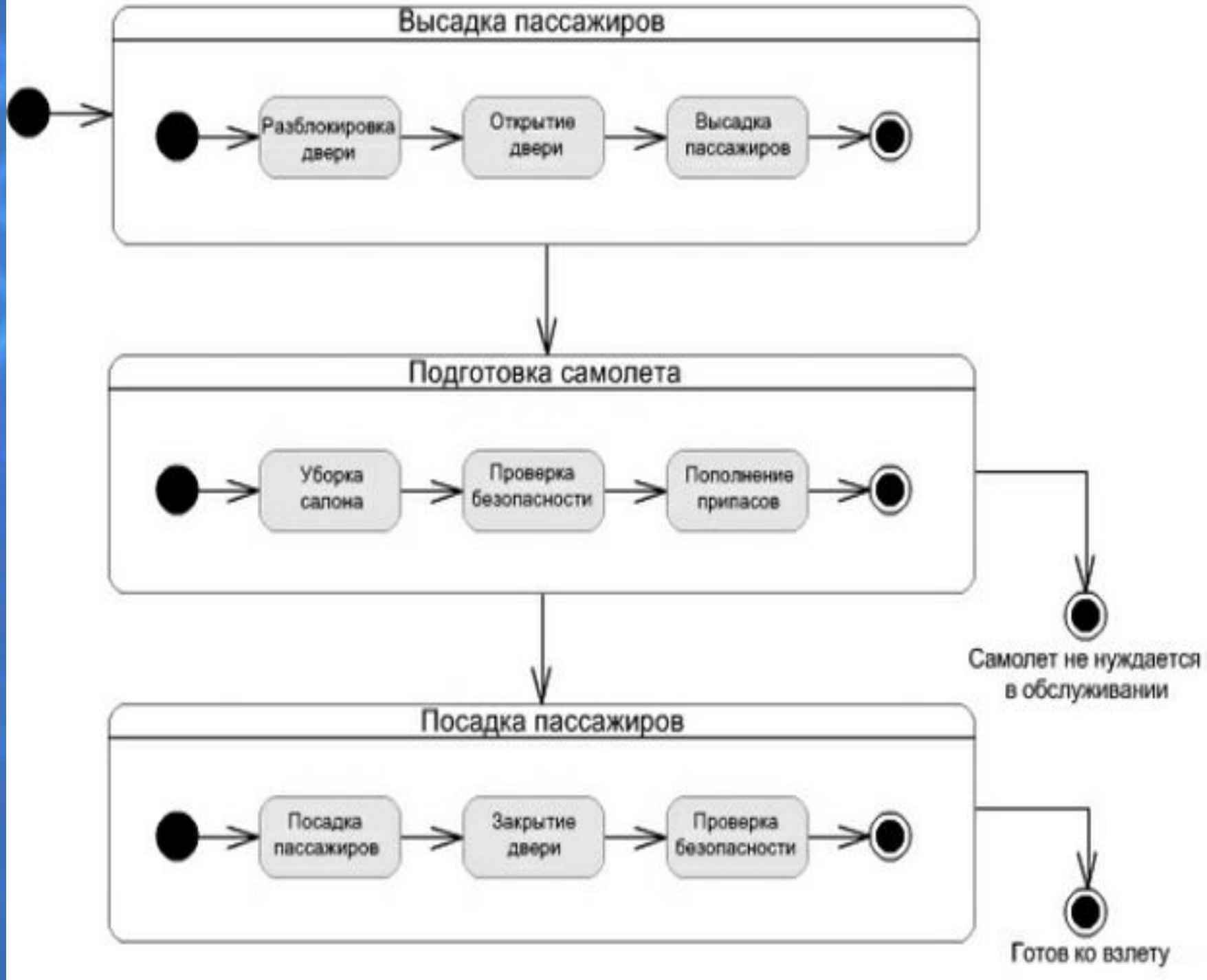
# Диаграмма деятельностей. *Траектория объекта*, или поток объекта

- Есть еще один нюанс нотации диаграмм активностей: это так называемая *траектория объекта*, или поток объекта (object flow). Суть его состоит в том, что на диаграмме деятельности можно изобразить и объекты, относящиеся к деятельности. С помощью символа зависимости (пунктирная стрелка) эти объекты можно соотнести с той деятельностью или переходом, где они создаются, изменяются или уничтожаются. Представим такую ситуацию из повседневной жизни: вы приходите в какой-нибудь фастфуд и заказываете гамбургер с колой. Во время приготовления завтрака повар создает новый объект - гамбургер. Пока вы нетерпеливо выпиваете колу, официант перемещает этот объект (подает ваш заказ). Естественно, во время завтрака вы уничтожаете этот объект. Вот как это выглядит на диаграмме.


Посетитель
Официант
Повар
Заказ
Принять заказ
Подает колу
Пьет колу
Готовит
[Гамбургер]
[Рождается]
Съедает завтрак
Подает заказ
[Гамбургер]
[Уничтожается]
[Гамбургер]
[Перемещается]

# Диаграмма деятельностей. "Вложенные" диаграммы

- Деятельность - это протяженное по времени составное действие. Составное, т.е.*составленное* из более простых действий. Вот эти простые (атомарные) действия, а вернее, последовательность их выполнения, частенько изображают внутри деятельности в виде маленькой диаграммы активностей. Это слегка напоминает матрешку - одна (а часто и не одна) диаграмма внутри другой. Покажем подобную возможность "вложенных" диаграмм на примере


Высадка пассажиров
Разблокировка двери
Открытие двери
Высадка пассажиров
Подготовка самолета
Уборка салона
Проверка безопасности
Пополнение припасов
Самолет не нуждается в обслуживании
Посадка пассажиров
Посадка пассажиров
Закрытие двери
Проверка безопасности
Готов ко взлету

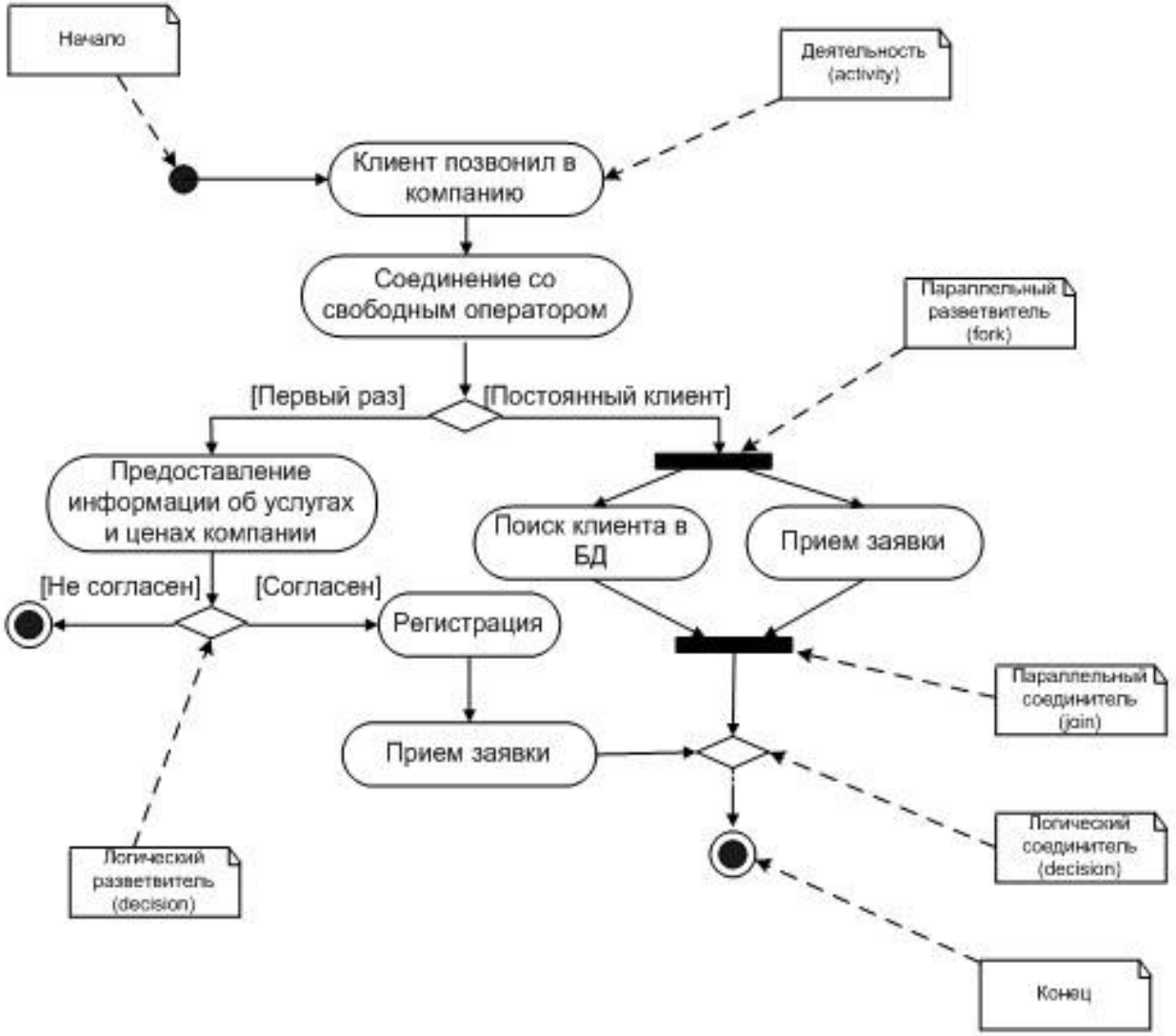
# Диаграмма деятельностей. Конечное состояние потока

- Диаграмма описывает высадку пассажиров самолета, достигших пункта назначения, и посадку новых пассажиров. Из диаграммы видно, что конечных состояний может быть больше одного.
- Кроме начального и конечного состояний есть еще конечное состояние потока (Flow final mode). Конечное состояние потока означает завершение одного потока управления, а конечное состояние говорит о завершении всех потоков управления внутри деятельности. Обозначается конечное состояние потока простым символом, напоминающим лампочку накаливания в схемах электрических цепей

# Диаграммы активностей (activity diagrams)

- С их помощью удобно изображать бизнес-процессы - алгоритмы, по которым работает компания. Именно в эти алгоритмы должна встроиться информационная система, автоматизировав некоторую их часть.
- Например, в компании должен быть создан новый бизнес-процесс по телефонной обработке заявок. Заказчик как-то себе представляет этот будущий процесс. Перед началом разработки системы необходимо уточнить алгоритм работы этой новой службы.

# Создание нового бизнес-процесса в организации по телефонной обработке заявок



# Диаграммы активностей (activity diagrams)

- Программистам полезно ясно представлять себе все бизнес-процессы компании, которые будут затронуты их новой системой. В данном случае у компании еще есть бизнес-процесс обработки заявок, который уже работает и есть у заказчика, и его также нужно понять. Иначе может оказаться, что упущена какая-то важная деталь, которая не позволяет новой системе полноценно выполнять свои функции. Например, может оказаться, что подсистема обработки заявок, с которой должна интегрироваться создаваемая система, реализована… на макросах к Word/Excel! Очевидно, интегрироваться с такой системой весьма затруднительно. На этот и подобные факты необходимо указать заказчику как можно раньше, так как иначе проект может закончиться неуспешно - заказчик потратит деньги и не получит нужных для своего бизнеса сервисов.

# Диаграммы активностей (activity diagrams)

- Главной сущностью этого типа диаграмм является активность (activity) - активное состояние системы, в котором она выполняет некоторую работу. После ее завершения происходит переход в другую активность. Возможны и более сложные случаи переходов между активностями. Например, переход по событию.
- На диаграмме должны присутствовать символы начала (start) и конца (finish).
- Далее, на диаграмме может использоваться параллельный разветвитель (fork), который запускает несколько одновременно работающих веток. Такие ветки могут объединяться (все или только часть) конструкцией под названием параллельный соединитель (join).
- Наконец, на диаграмме могут использоваться символы логического ветвления и логического соединения (decision). На ветках, идущих из логического ветвления, обозначаются условия перехода.

# Диаграммы активностей (activity diagrams)

- **Диаграмма деятельности (activity diagram)** - это методология объектно-ориентированного проектирования, предназначенная для детализации особенностей алгоритмической и логической организации системы. При этом каждое действие расчленяется на фундаментальные процессы.

- На диаграмме деятельности управление осуществляется:

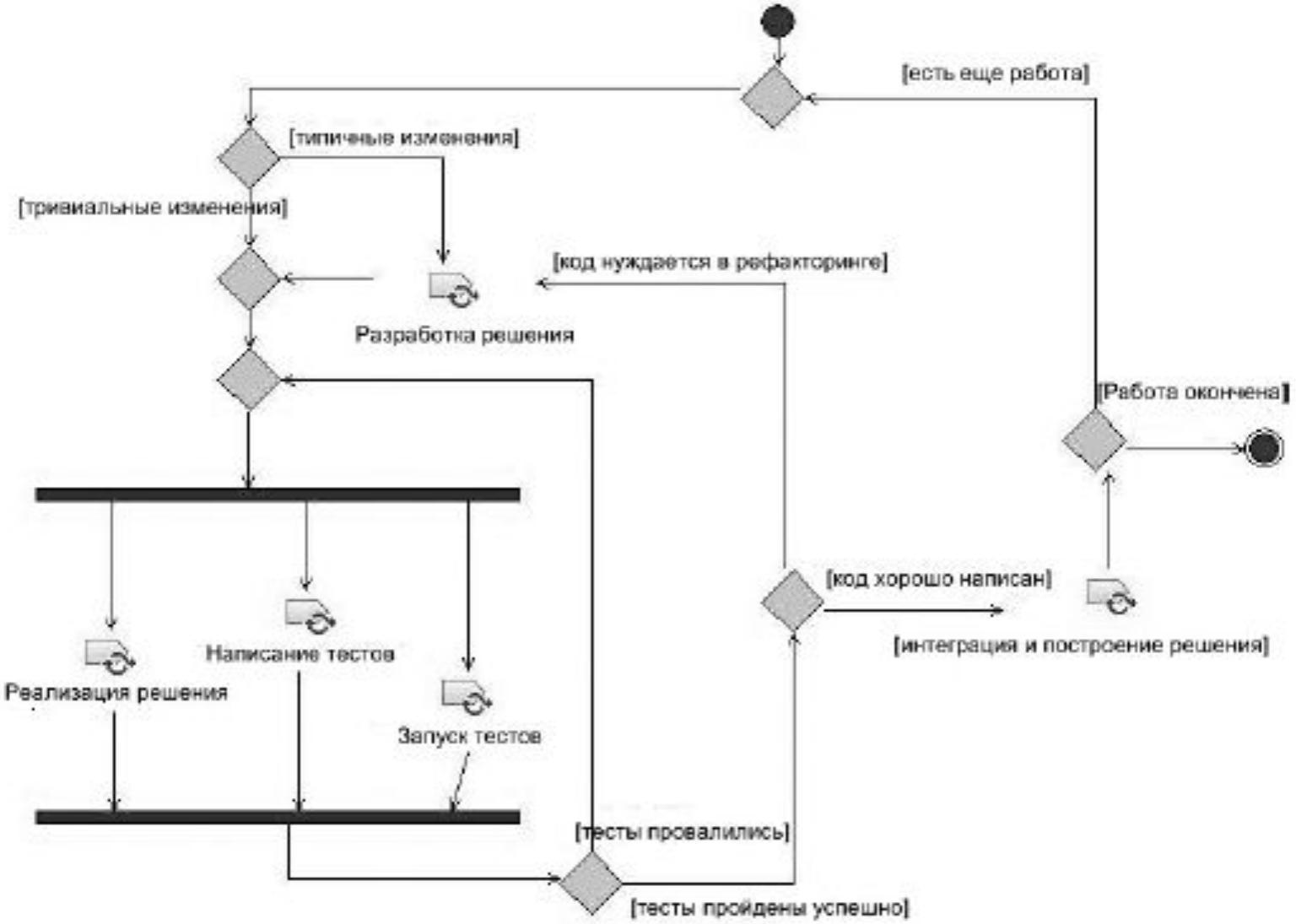
  - либо через потоки управления (явно);

  - либо через определяемые потоки данных (неявно).

# Два способа использования диаграмм деятельностей

- **Для моделирования процессов.**
- В этом случае внимание фокусируется на деятельности с точки зрения экторов, которые работают с системой (для описания бизнес-процессов). При этом используются *траектории объектов*. (Например на диаграмме завтрака (гамбургер с колой), изменив роли и деятельности, вместо гамбургера легко представить некий документ.)
- **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*).

# Пример использования диаграммы активностей для описания процесса разработки ПО в OpenUP



# Моделирования операций с помощью диаграмм активностей

- В этом случае диаграмма активностей превращается в "продвинутую" блок-схему, предоставляющую дополнительные возможности, например, отображение параллельно выполняющихся операций. Из-за соблазна выполнить кодогенерацию такой диаграммы или даже откомпилировать ее и сразу получить выполняемый файл были созданы пакеты для генерации приложений непосредственно из диаграмм UML. Некоторые даже оказались более-менее удачными - например, Rational Rose Real Time.

# Пример моделирования базовой алгоритмической конструкции: цикла с постусловием

# Советы по построению диаграмм активностей

Процесс построения диаграммы активностей можно описать в виде последовательности таких действий:

1. **Составление перечня деятельностей в системе.** Исходными данными для этого - список прецедентов (или список операций - см. два способа использования диаграмм деятельности). Она дополняет сценарии использования и даже описывает связь между ними.
2. **Принятие решения о необходимости построения диаграммы деятельностей.** Начав эту работу проверьте отказаться или продолжать построение диаграммы деятельностей.
3. **Определение зависимостей между деятельностями.** Для каждой активности нужно найти непосредственно предшествующие (и следующие за ней) активности, то есть активности, без выполнения которых поток управления не может перейти к данной деятельности.
4. **Выделение параллельных потоков деятельностей.** Выделите активности, имеющие общих предшественников.
5. **Определение условий переходов.** Сформулируйте выражения, которые могут принимать только два значения - "истинно" или "ложно", соответствующие альтернативным потокам управления.
6. **Уточните сложные деятельности**

# Пример моделирования пословицы "После драки кулаками не машут"

1. Выделяем деятельности: драться, махать кулаками.
2. Следует ли строить диаграмму в этом случае? {Вообще-то нет. Но ведь это пример)
3. Определяем зависимости между деятельностями: размахивание кулаками не происходит после драки.
4. Определяем параллельные деятельности: вроде бы тут таких не наблюдается...
5. Определяем условия переходов: драка состоялась? Если "нет", то машем кулаками, если "да", то нет.
6. Уточняем сложные деятельности: при драке машут не только кулаками, но и ногами. А еще можно пинаться головой и использовать подручные средства, мебель, например. Плюс можно выделить еще подготовительные деятельности (выбор места для нападения) и завершающие (вынос раненых).

# Диаграммы деятельностей. Выводы

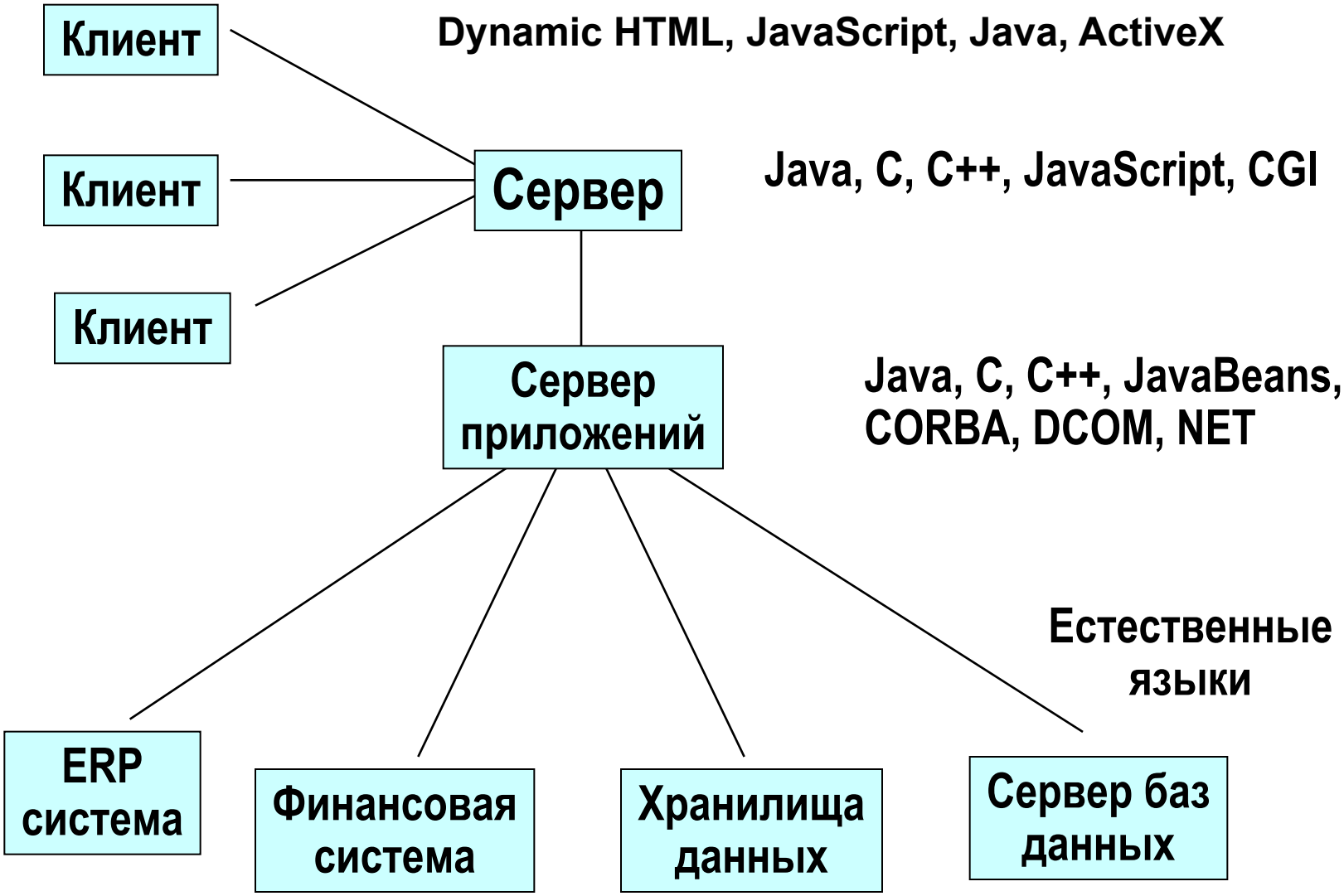
- Можно дополнить любой элемент модели, имеющий динамическое поведение.
- Являются частным случаем диаграммы состояний.
- В отличие от блок-схем могут отображать одновременно выполняемые действия.
- На них можно использовать плавательные дорожки, распределяющие деятельности в соответствии с ролями (объектами), их выполняющими.
- Траектория объекта позволяет показать объекты, относящиеся к деятельности, и моменты переходов этих объектов из одного состояния в другое.
- Сложные деятельности можно дополнительно детализировать, разбив на действия и изобразив "диаграмму в диаграмме".
- Можно использовать для проектирования процессов (например, бизнес-процессов) или операций (вычислений). Во втором случае UML выступает в роли визуального языка программирования.

# Диаграмма развертывания (deployment diagram)

# Диаграмма развертывания (deployment diagram)

- Является опционной и используется только тогда, когда разрабатываемое приложение имеет распределенный характер, т.е. отдельные компоненты системы могут быть инсталлированы на различных узлах компьютерной сети, и необходимо ответить на вопрос, какие компоненты должны быть размещены на каких узлах. Если речь идет о распределенных приложениях, то актуальность этой диаграммы несомненна.
- Второй аспект: показать инфраструктуру, являющуюся основой для функционирования приложения. Традиционно её изображают произвольно.

# Архитектура распределенных систем – исходное представление



# Диаграмма развертывания (deployment diagram)

- Когда мы пишем программу, мы пишем ее для того, чтобы запускать на компьютере, который имеет некоторую аппаратную конфигурацию и работает под управлением некоторой операционной системы. Корпоративные приложения часто требуют для своей работы некоторой *ИТ-инфраструктуры*, хранят информацию в базах данных, расположенных где-то на серверах компании, вызывают веб-сервисы, используют общие ресурсы и т. д. В таких случаях хорошо иметь *графическое представление инфраструктуры, на которую будет развернуто приложение*. Для этого и нужны **диаграммы развертывания**, которые иногда называют диаграммами размещения.
- Такие диаграммы есть смысл строить только для аппаратно-программных систем, тогда как UML позволяет строить модели любых систем, не обязательно компьютерных

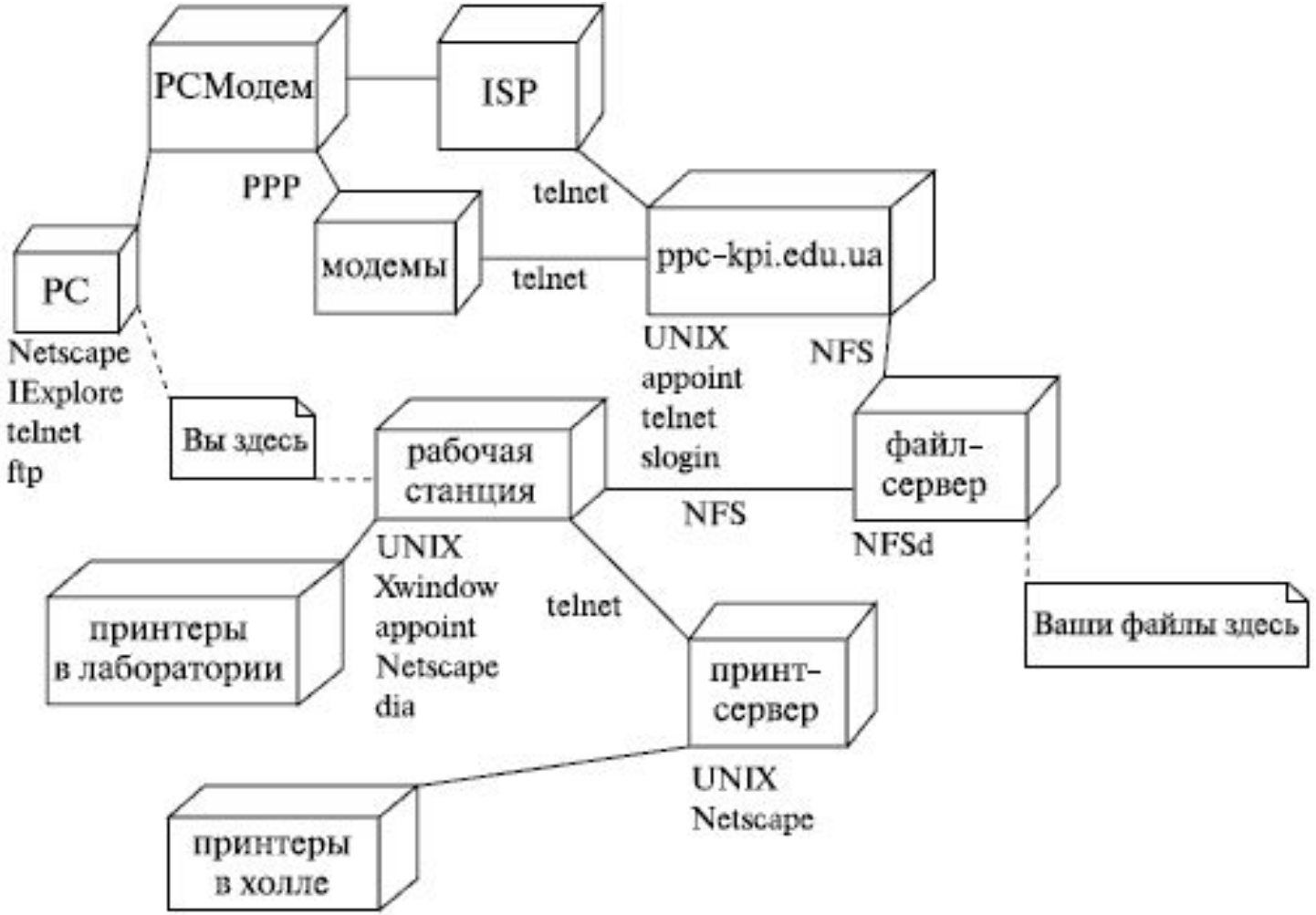
# Какую пользу можно извлечь из диаграмм развертывания?

- Во-первых, графическое представление ИТ-инфраструктуры может помочь *более рационально распределить компоненты системы по узлам сети*, от чего, как известно, зависит в том числе и производительность системы.
- Во-вторых, такая диаграмма может помочь *решить множество вспомогательных задач*, связанных, например, с обеспечением безопасности.

Диаграмма развертывания показывает топологию системы и распределение компонентов системы по ее узлам, а также соединения - маршруты передачи информации между аппаратными узлами. Это единственная диаграмма, на которой применяются "трехмерные" обозначения: узлы системы обозначаются кубиками. Все остальные обозначения в UML - плоские фигуры.

# Диаграмма развертывания. Пример

# Диаграмма инфраструктуры учебного заведения (включающая шлюз, файл-сервер, принт-сервер, принтеры в лабораториях и холле)

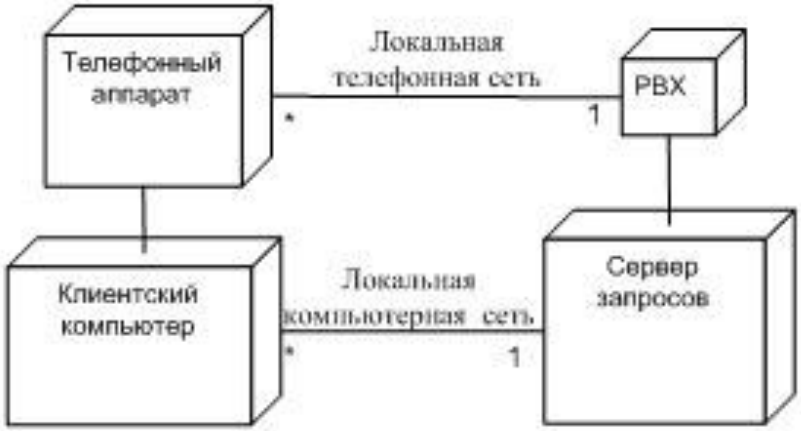


# Диаграммы развертывания (deployment diagrams)

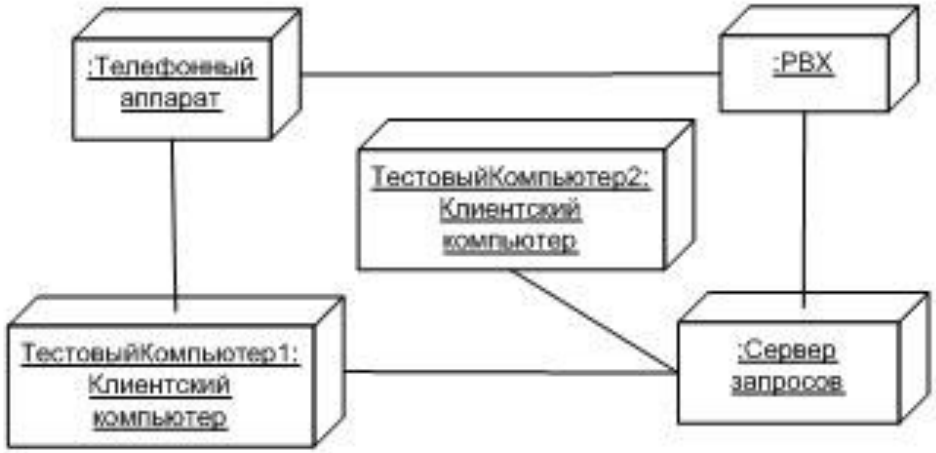
- Диаграммы развертывания предназначены для описания аппаратной части системы.
- Они первом приближении определяют будущую систему изнутри.
- Диаграмма развертывания может давать хороший обзор всей системы, доступный для непрограммистов (особенно в составе Power-Point презентации с устными пояснениями), поскольку содержит минимум программистских деталей.

# Описательный и экземплярный виды диаграмм развертывания

- Телефонная служба приема заявок будет состоять из офисной телефонной станции ( PBX - Public Branch Exchange), сервера, телефонных аппаратов и клиентских компьютеров. На диаграмма развертывания (в описательном виде) определены типы аппаратных узлов системы, а между ними - ассоциации с пометками множественности



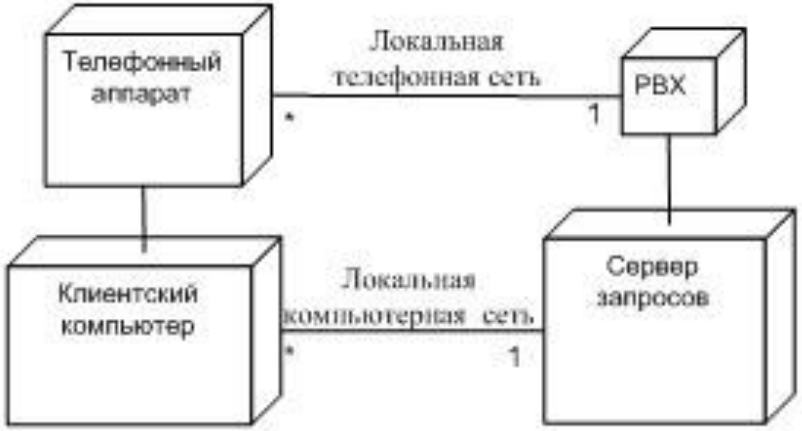

а). Диаграмма развертывания: описательный уровень



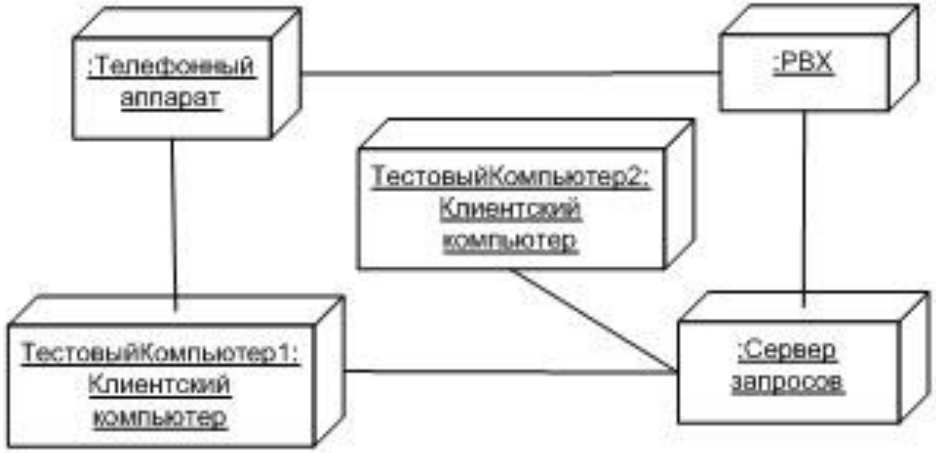

б). Диаграмма развертывания: экземплярный уровень

# Описательный и экземплярный виды диаграмм развертывания

- В экземплярном варианте показан тестовый вариант системы, который, кроме сервера и PBX, содержит один пользовательский компьютер для тестирования взаимодействия сервера и клиента и второй клиентский компьютер вместе с телефонным аппаратом для тестирования связи клиента с сервером и PBX. Два клиентских компьютера нужны, чтобы тестировать работу ПО в случае более чем одного клиента (при переходе от одного к двум начинают появляться многочисленные ситуации, которые не проявлялись ранее).




а). Диаграмма развертывания: описательный уровень




б). Диаграмма развертывания: экземплярный уровень

# Диаграммы развертывания (deployment diagrams)

- На диаграммах развертывания показываются узлы (nodes) - элементы аппаратуры, которые также входят в целевую систему, наравне с ПО. На части этих узлов и развертывается ПО системы. В рассмотренном примере такими узлами являются клиентский компьютер и сервер запросов. Кроме того, на диаграммах развертывания могут быть показаны и другие виды узлов - элементы аппаратуры, с которым ПО лишь взаимодействует, например, PBX или телефонный аппарат. Данный тип диаграмм не предназначен для подробного описания аппаратной части системы, а позволяет моделировать только ту часть оборудования, которая прямо или косвенно связана с ПО системы. Например, в данном случае в целевую систему может входить дополнительное сетевое оборудование - переключатели (так называемые "хабы") и т. д. Для подробной спецификации всего этого целесообразно использовать не UML, а средства классического инженерного проектирования.
- Построение инженерных чертежей на сегодняшний день также компьютеризировано. Самыми распространенными программными продуктами здесь являются пакеты AutoCAD, Microsoft Visio и др.

# Диаграммы развертывания (deployment diagrams)

- Диаграммы развертывания могут использоваться, например, как приложение к техническому заданию, а также при обсуждении цен на различные офисные АТС, телефонные аппараты и компьютеры. Такую диаграмму может нарисовать менеджер проекта перед тем, как начать обсуждение архитектуры системы с разработчиками. Эта диаграмма может лежать на столе во время первых таких обсуждений, пока не родилось ничего более конкретного, что также можно нарисовать. Такое "начало от аппаратуры" часто является хорошим стартом проекта, поскольку именно в терминах аппаратуры для многих программно-аппаратных систем формируется существенная часть их функциональных требований: ПО должно уметь управлять таким-то оборудованием в таких-то режимах и т. д.

# Итоги. Диаграммы UML

- Каждая из диаграмм детализирует и конкретизирует различные представления о модели сложной системы в терминах языка UML.
- При этом диаграмма вариантов использования (функционал, доступный пользователям системы) представляет собой наиболее общую концептуальную модель сложной системы, которая является исходной для построения всех остальных диаграмм.
- Диаграмма классов (логическая модель данных системы, сущностей, из которых будет строиться БД), по своей сути, логическая модель, отражающая статические аспекты структурного построения сложной системы.
- Диаграммы кооперации и последовательностей (описание одного сценария) представляют собой разновидности логической модели, которые отражают динамические аспекты функционирования сложной системы.
- Диаграммы состояний (отображение статусов объектов) и деятельности (блок-схема) предназначены для моделирования поведения системы.
- И, наконец, диаграммы компонентов и развертывания служат для представления физических компонентов сложной системы и поэтому относятся к ее физической модели.

# Диаграммы UML

- Каждая из диаграмм позволяет рассматривать бизнес-процессы под различным углом.
- Деловые пользователи при помощи данных диаграмм могут оценить основные положения бизнес-процесса и разобраться в том, кто за что отвечает. Разработчики же применяют диаграммы классов и объектов для получения точного представления о том, как встраивать данные компоненты в свой код.
- Диаграммы классов описывают статическое состояние элементов системы в каждый конкретный момент, показывают структуру объектов, их атрибуты и взаимные связи.
- Диаграммы деятельности отображают управляющие потоки, идущие от одного действия к другому, а диаграммы вариантов использования иллюстрируют элементы, находящиеся за пределами системы. (К примеру, внутренние операции новой платежной системы отображаются на диаграмме действий, тогда как работа внешних агентов, в частности отдела обработки заказов, представлена на диаграмме вариантов использования.)
- Диаграммы последовательности отражают интерактивные процессы: вы видите не только объекты и классы, но и сообщения, которыми они обмениваются. Таким образом, с помощью системы можно моделировать ситуации, применяя обычную в таких случаях технологию «что, если». Диаграммы состояния используются для описания динамических объектов.
- Но нужно четко понимать какую цель мы преследуем, когда хотим заняться описанием бизнес-процессов, и подбирать соответствующие средства для этого. Например, если цель сделать автоматизацию БП, в сфере где БП нам известны, то не факт, что имеет смысл делать полномасштабное описание бизнеса, а может только некоторые общие вещи или наоборот -- отдельные детали.

# Последовательность построения диаграмм

Можно дать множество рекомендаций относительно того, какие же именно диаграммы строить и как. Прежде всего, нужно ответить на вопросы:

- Какие именно виды диаграмм лучше всего отражают архитектуру системы и возможные технические риски, связанные с проектом?
- Какие из диаграмм удобнее всего превратить в инструмент контроля над процессом (и прогрессом) разработки системы?

# Рекомендуемая последовательность построения диаграмм:

| Модель | Назначение |
|---|---|
| Диаграммы вариантов использования | Описание функционального назначения системы |
| Диаграммы деятельности | Описание потоков работ в бизнес-процессах и\или внутри UC |
| Диаграммы последовательности | Описание поведения системы |
| Диаграммы классов | Описание предметной области |
| Диаграммы состояний | Описание событий, изменяющих состояния объектов |

# Другая рекомендуемая последовательность построения диаграмм:

- диаграмма прецедентов,
- диаграмма классов,
- диаграмма объектов,
- диаграмма последовательностей,
- диаграмма кооперации,
- диаграмма состояний,
- диаграмма активности,
- диаграмма развертывания.

# Выводы

- Диаграммы разных видов позволяют взглянуть на систему с разных точек зрения.
- UML содержит диаграммы трех типов - для моделирования статической структуры, поведенческих аспектов и подробностей реализации приложения.
- Недостаточно читать об UML - им надо пользоваться!

Для диаграммы UC модели банкомата построить

- диаграмму деятельностей
- диаграмму классов