

The logo for UML (Unified Modeling Language) consists of the letters 'U', 'M', and 'L' in a bold, white, sans-serif font. The letters are spaced out and centered on a blue background with wavy lines at the top.

Unified Modeling Language

# Что такое UML?

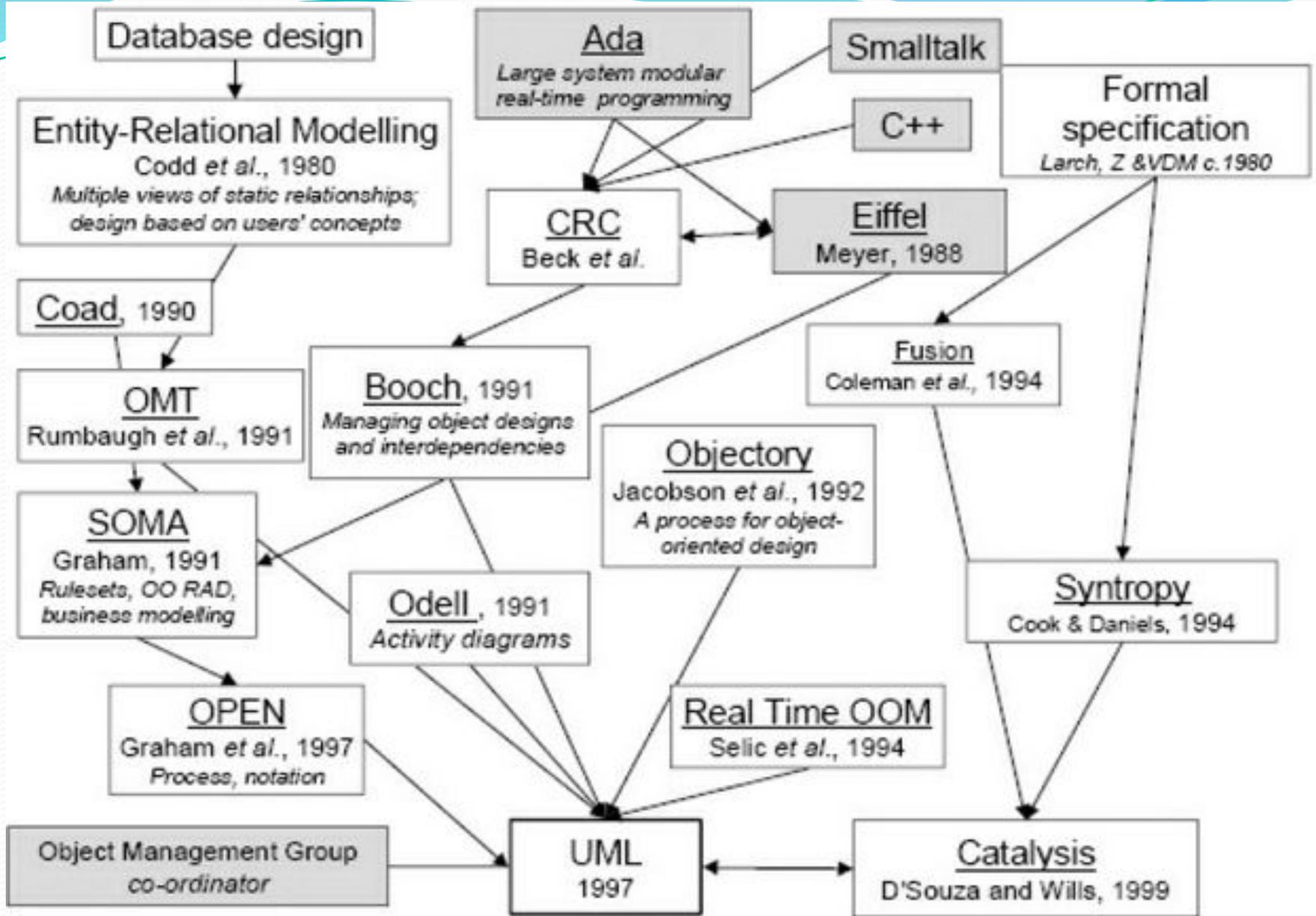
- Одна из **задач UML** - служить средством коммуникации внутри команды и при общении с заказчиком.
- UML – **Unified Modeling Language** - унифицированный язык моделирования.
- UML – это язык
  - Формальный (есть правила его употребления)
  - Искусственный (есть авторы)
  - Графический

# Что такое UML?

- Modeling подразумевает создание модели, описывающей объект.
- Unified – единый, объединенный (англ.)
- UML стал единым универсальным стандартом для объектно-ориентированного моделирования.

# История появления

- Начало 80-х – старт объектно-ориентированной эры.
- 80-е годы – «война методов». Один и тот же символ мог означать в разных нотациях абсолютно разные вещи!
- 1991 год - начало создания UML.  
Создатели: Грэйди Буч (Grady Booch), Джеймс Рэмбо (James Rumbaugh). Чуть позже к ним присоединился Якобсон (Jacobson).
- январь 1997 года - Первая версия UML была принята консорциумом OMG (Object Management Group).





Так объяснил заказчик



Так понял лидер проекта



Так спроектировал аналитик



Так реализовал программист



Так описал бизнес-консультант



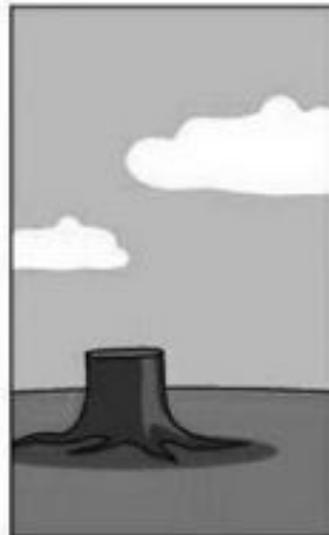
Так проект был документирован



Так продукт был проинсталлирован



Такой счет был выставлен заказчику



Так осуществлялась техническая поддержка



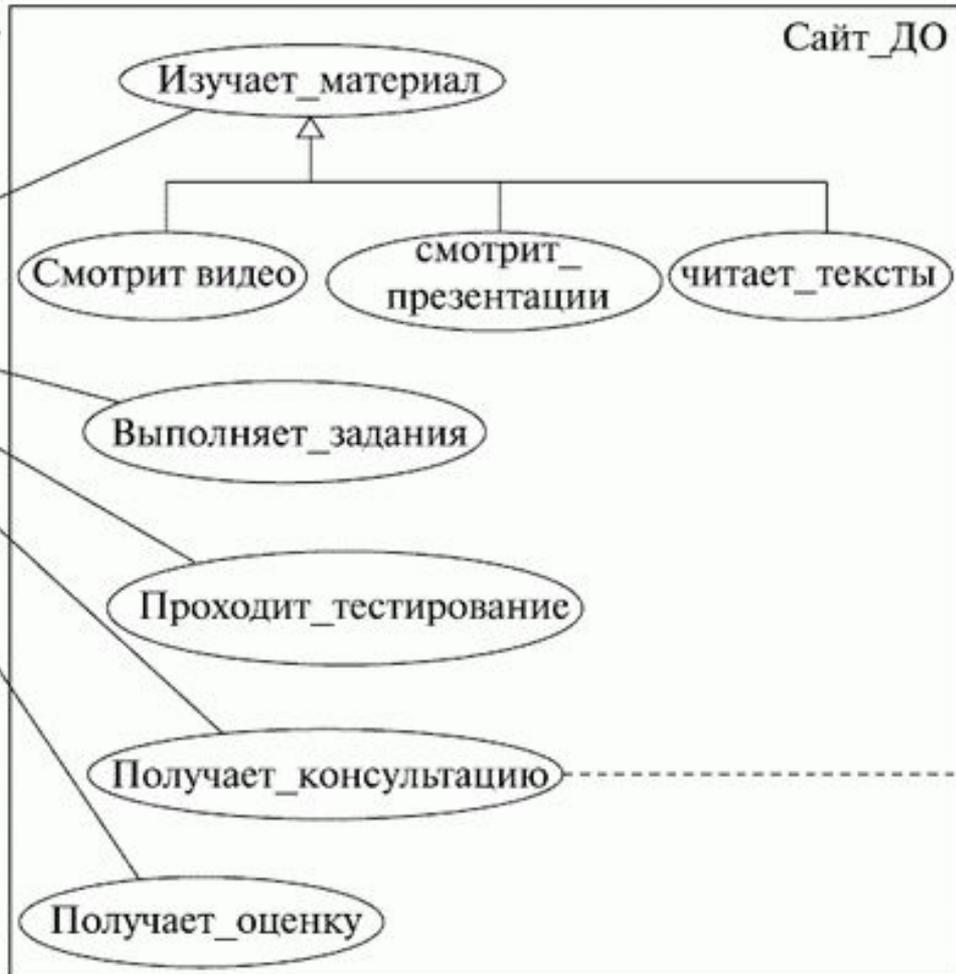
А вот чего на самом деле хотелось заказчику

# Что такое UML?

- Авторы UML представляют его как графический язык моделирования **общего** назначения для
  - спецификации,
  - визуализации,
  - проектирования
  - документированиявсех артефактов, создаваемых в ходе разработки.
- **Спецификация** - подробное описание системы, которое полностью определяет ее цель и функциональные возможности.

ПрецедентыСтудент

Сайт\_ДО



через ICQ

# Что такое UML?

- По моделям может производиться **генерация каркасного кода** проектируемых приложений.
- **Реверс-инжиниринг** - создание UML-модели из существующего кода приложения
- UML-модели являются документами (кроме того, модели UML являются XML-документами).
- Можно извлекать текстовую информацию из моделей и генерировать относительно удобочитаемые тексты.

# Что такое UML?

- UML не является языком программирования, это средство создания моделей.
- UML не зависит от языков программирования и может поддерживать любой объектно-ориентированный язык программирования.
- UML является **открытым** и позволяет расширять ядро.

# Нотация UML

- "Нотация" - это то, что в других языках называют "синтаксисом".
- 4 вида элементов нотации:
  - Фигуры - прямоугольники, эллипсы, ромбы и т. д. ,
  - Линии - сплошная и пунктирная,
  - значки,
  - надписи.

# CASE-средства

- **CASE-средства** реализуют UML-моделирование.  
CASE = Computer-Aided Software Engineering.
- Примеры:
  - Visual Paradigm;
  - IBM Rational Rose;
  - Borland Together
  - и др.

# Определения

- **Система** - набор более простых сущностей, которые относительно самостоятельны.
- **Модель** - это некий (материальный или нет) объект, отображающий лишь наиболее значимые для данной задачи характеристики системы.
- **Диаграмма** - это графическое представление множества элементов. Обычно изображается в виде графа с вершинами (сущностями) и ребрами (отношениями).

# Виды диаграмм

- В языке UML 12 типов диаграмм:
  - 4 типа диаграмм представляют **статическую** структуру приложения;
  - 5 представляют **поведенческие аспекты** системы;
  - 3 представляют **физические аспекты** функционирования системы (диаграммы реализации).

# Виды диаграмм

- диаграмма прецедентов (Use-case diagram);
- диаграмма классов (Class diagram);
- диаграмма объектов (Object diagram);
- диаграмма последовательностей (Sequence d.);
- диаграмма взаимодействия (Collaboration d.);
- диаграмма состояний (Statechart diagram);
- диаграмма активности (Activity diagram);
- диаграмма развертывания (Deployment diagram)
- и другие...

# USE-case diagram

- **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.



# USE-case diagram

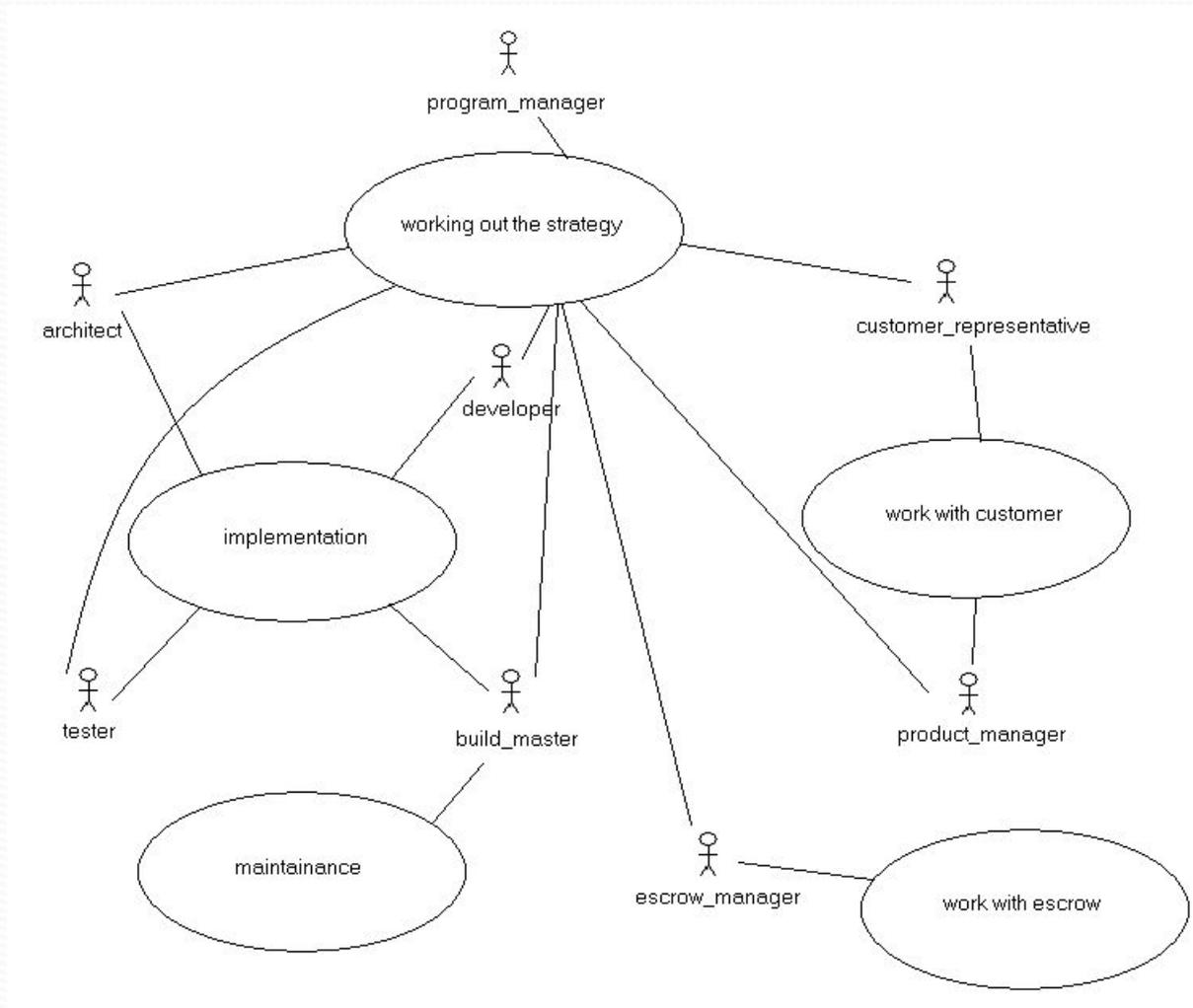
- **Прецедент (use-case)** - описание отдельного аспекта поведения системы с точки зрения пользователя.
- Прецедент представляет поведение сущности, описывая взаимодействие между акторами и системой.
- Прецедент не показывает, "как" достигается некоторый результат, а только "что" именно выполняется.



# USE-case diagram



# USE-case diagram



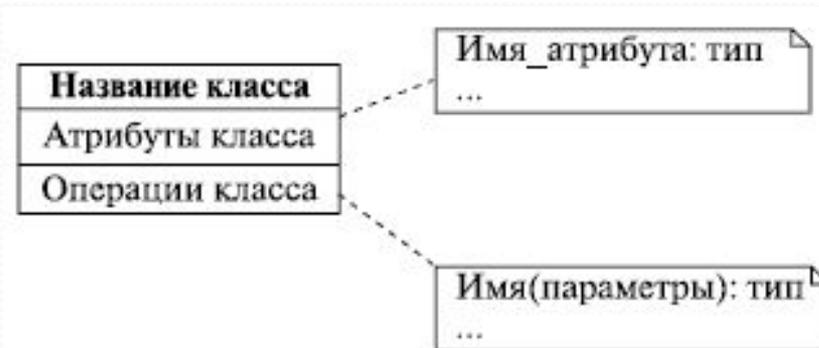
# USE-case diagram. Цели

## создания:

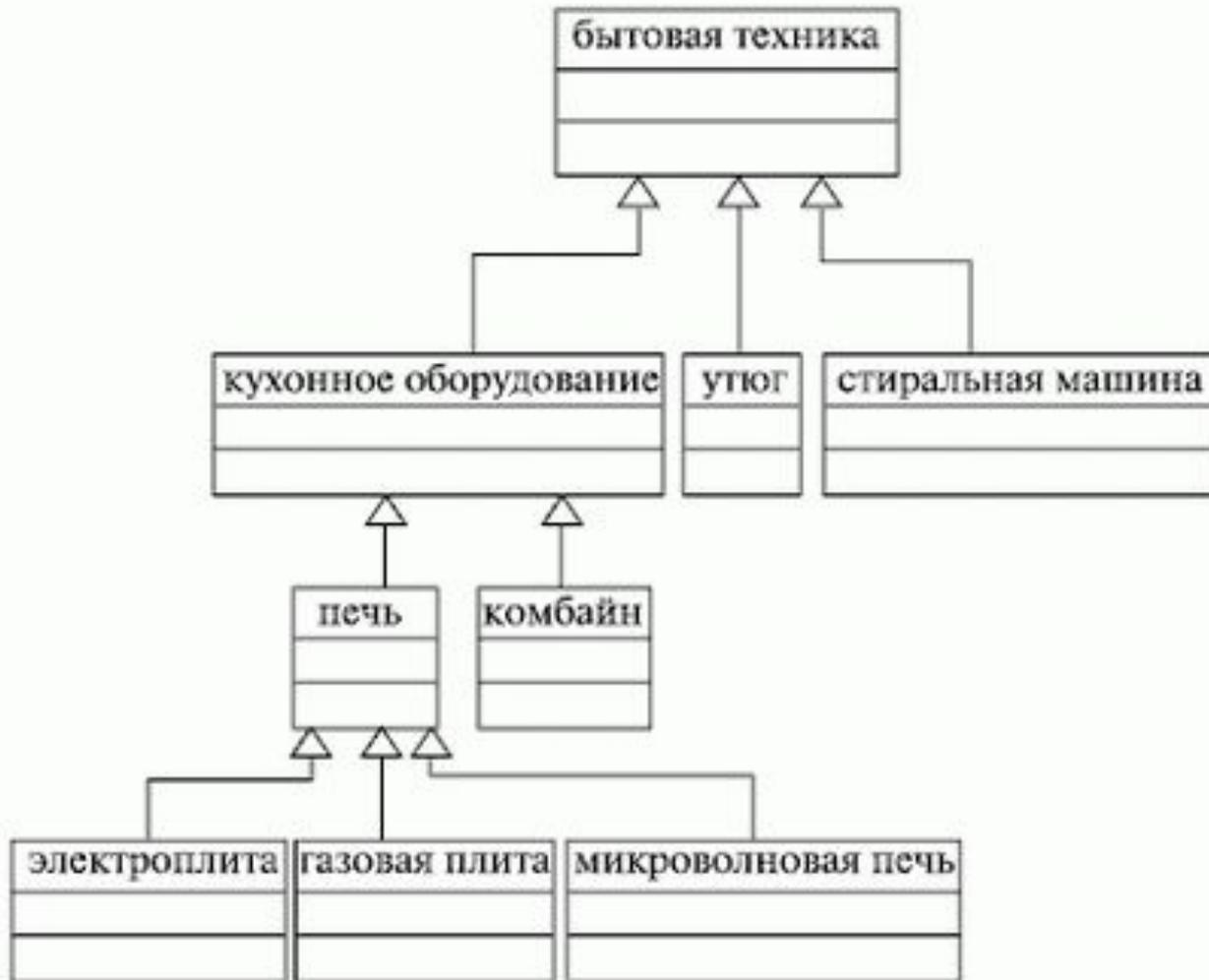
- определение границы и контекста моделируемой предметной области на ранних этапах проектирования;
- формирование общих требований к поведению проектируемой системы;
- разработка концептуальной модели системы для ее последующей детализации;
- подготовка документации для взаимодействия с заказчиками и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

# Class diagram

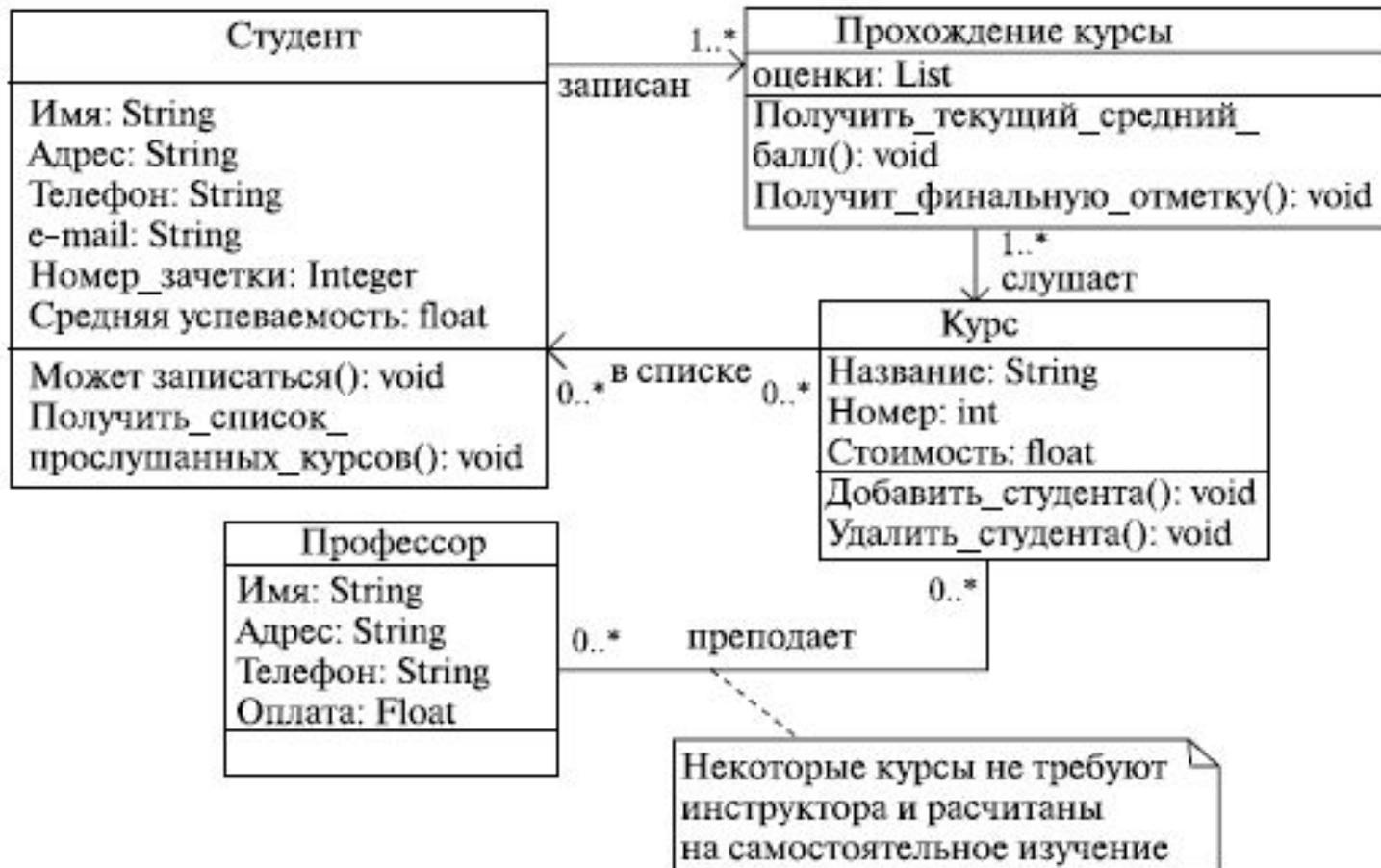
- **Класс (class)** - категория вещей, которые имеют общие атрибуты и операции.
- **Диаграмма классов** - это набор статических, декларативных элементов модели.



# Class diagram

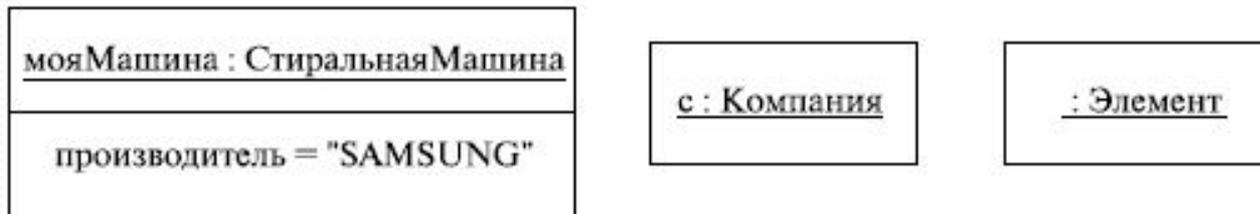


# Class diagram

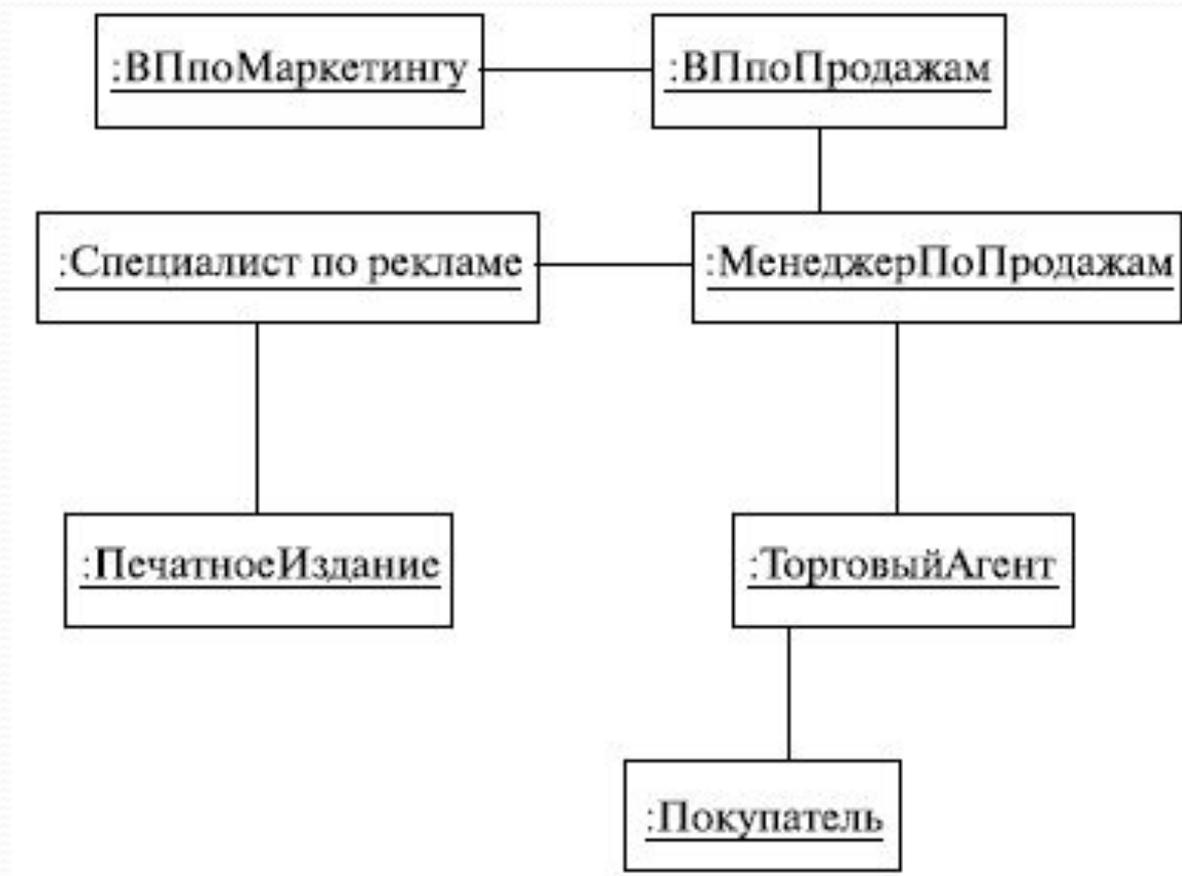


# Object diagram

- **Объект (object)** - экземпляр класса.
- **Диаграмма объектов** - это своего рода снимок состояния системы в определенный момент времени, показывающий множество объектов, их состояния и отношения между ними в данный момент.



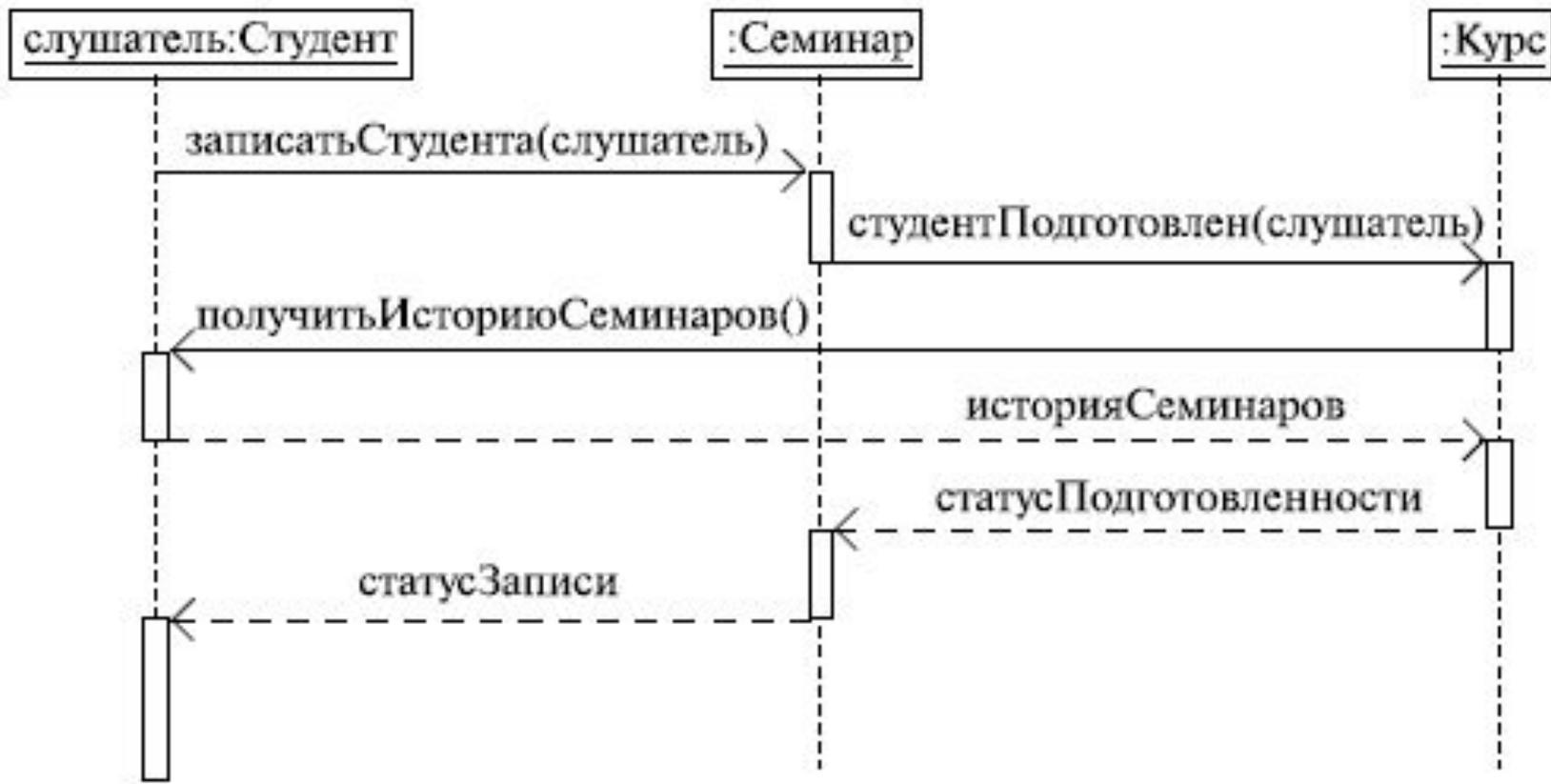
# Object diagram



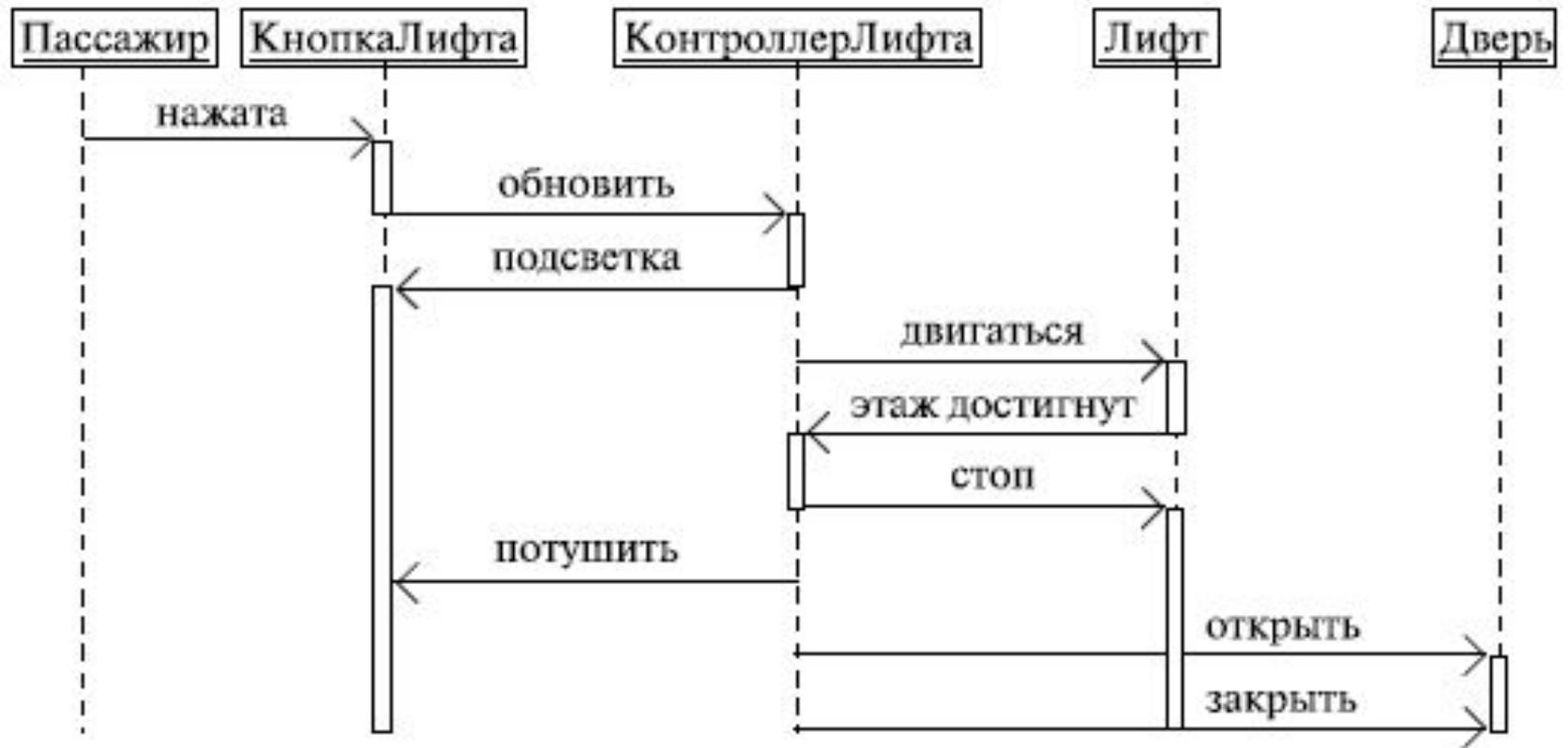
# Sequence diagram

- Взаимодействие объектов - обмен информацией между ними. Информация принимает вид сообщений.
- **Диаграмма последовательностей** отображает взаимодействие объектов в динамике, во времени.
- Отражает **поведенческие** аспекты системы
- Отображает временные особенности передачи и приема сообщений объектами.

# Sequence diagram



# Sequence diagram



# Collaboration diagram

- Диаграмма взаимодействия показывает поток сообщений между объектами системы и основные ассоциации между ними.
- По сути является альтернативой диаграммы последовательностей.
- Время на диаграмме взаимодействия не показывается в виде отдельного измерения.

# Collaboration diagram

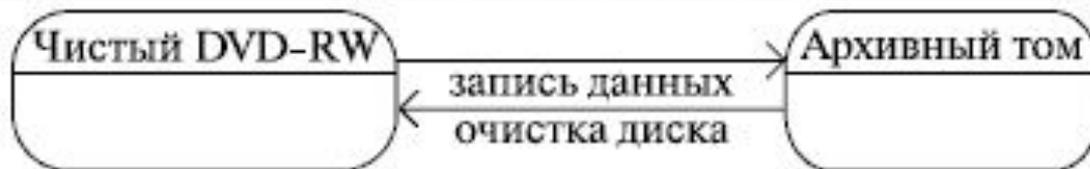


# Statechart diagram

- Объекты характеризуются поведением и состоянием, в котором находятся.
- **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.

# Statechart diagram

- **Диаграмма состояний** показывает, как объект переходит из одного состояния в другое.
- Диаграмма состояний полезна при моделировании жизненного цикла объекта.
- Диаграмма состояний описывает процесс изменения состояний **только одного** экземпляра определенного класса, поведение которого характеризуется его реакцией на внешние события.

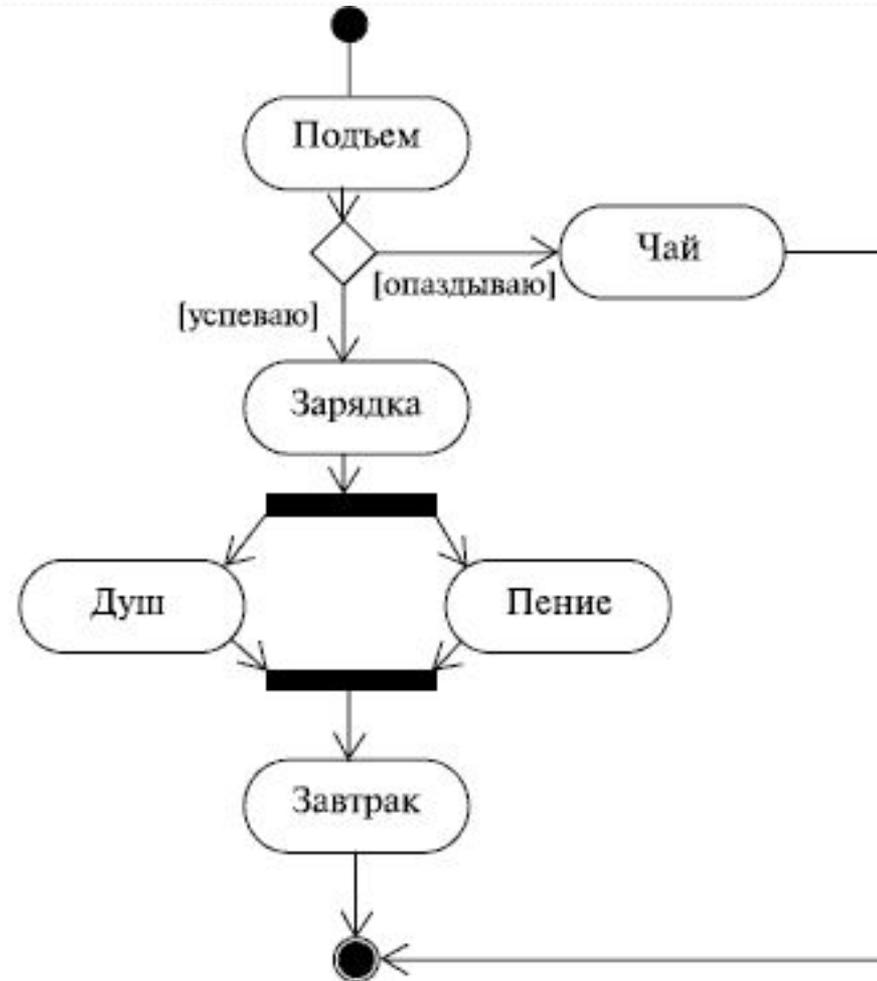




# Activity diagram

- Часто недостаточно изобразить процесс смены состояний системы, а нужно также раскрыть детали алгоритмической реализации операций ею выполняемых.
- **Диаграмма деятельности** - частный случай диаграмм состояний.
- **Алгоритм** - последовательность определенных действий или элементарных операций, выполнение которых приводит к получению желаемого результата.

# Activity diagram



# Activity diagram



# Deployment diagram

- Диаграммы развертывания есть смысл строить только для аппаратно-программных систем.
- **Диаграммы развертывания** - графическое представление инфраструктуры, на которую будет развернуто приложение.
- Это единственная диаграмма, на которой применяются **"трехмерные"** обозначения

# Deployment diagram



# Deployment diagram

- Диаграмма развертывания показывает
  - топологию системы
  - распределение компонентов системы по ее узлам,
  - соединения - маршруты передачи информации между аппаратными узлами.
- Цель:
  - рационально распределить компоненты системы по узлам сети, от чего зависит производительность.
  - решить множество вспомогательных задач, связанных, например, с обеспечением безопасности.

