



**UML - унифицированный язык
моделирования**

Этапы жизненного цикла программной системы

- Анализ предметной области и определение требований к системе
- Проектирование структуры системы
- Реализация системы в кодах
- Внедрение системы
- Сопровождение системы
- Отказ от использования системы

Методология объектно-ориентированного анализа и проектирования (ООАП)

Основные принципы ООАП

- Анализ требований к системе
- Объектно-ориентированный анализ предметной области
- Объектно-ориентированное проектирование

Сначала производится анализ требований, во время которого выделяются основные процессы, происходящие в предметной области и их формулировка в виде **прецедентов**

Прецедент (precedent) - это текстовое описание процессов, происходящих в предметной области

В процессе объектно-ориентированного анализа основное внимание уделяется определению и описанию объектов (понятий) в терминах предметной области

В процессе объектно-ориентированного проектирования разрабатывается структура программной системы

Это детализированная схема, на которой указываются классы, их свойства и методы, а также связи между классами

Именно данная схема служит исходной информацией для написания кода

Главная задача анализа предметной области – выработка точной, четкой, доступной для понимания модели реальной системы

Модель – это абстракция, которая создается для того, чтобы лучше понять задачу, перед тем как приступить к её решению

Моделирование - процесс построения и последующего применения моделей для получения информации о системе

Для фиксации результатов
моделирования системы и их
документирования естественный язык не
подходит из-за неоднозначности и
неопределенности

Типичный процесс создания продукта

Так объяснил заказчик



Так понял лидер проекта



Так спроектировал аналитик



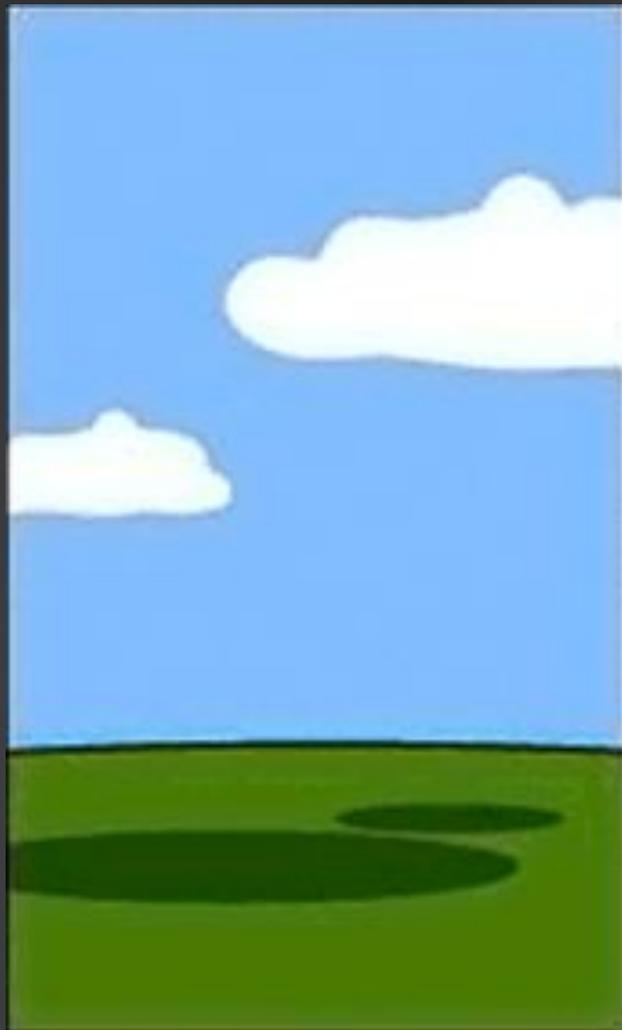
Так реализовал программист



Так описал бизнес-консультант



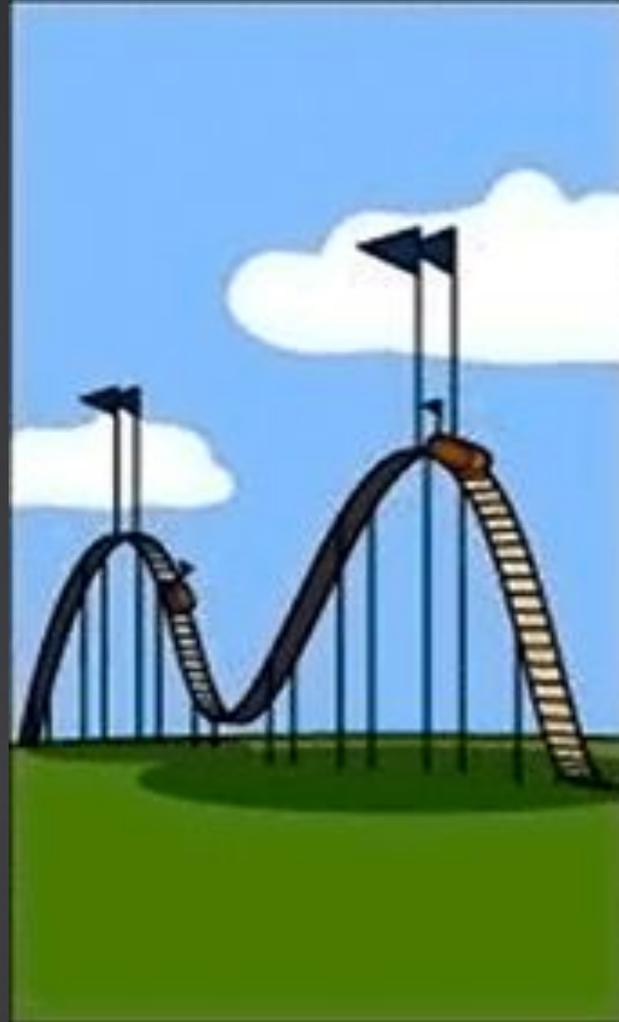
Так проект был документирован



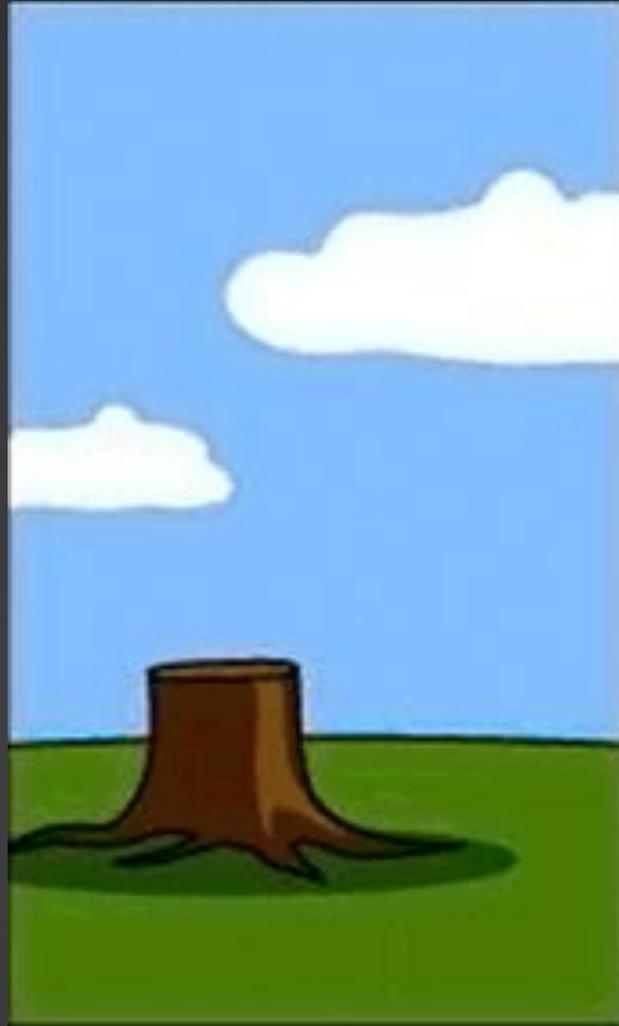
Так продукт был проинсталлирован



Такой счёт был выставлен заказчику



Так осуществлялась техническая поддержка



А вот что на самом деле хотел заказчик



Очевидны проблемы с коммуникацией и пониманием, вызванные отсутствием четкой спецификации создаваемого продукта

Необходим унифицированный язык
моделирования, предоставляющий
изобразительные средства
(графическую нотацию)

UML (Unified Modeling Language)
использует графические обозначения
для создания абстрактной
модели системы

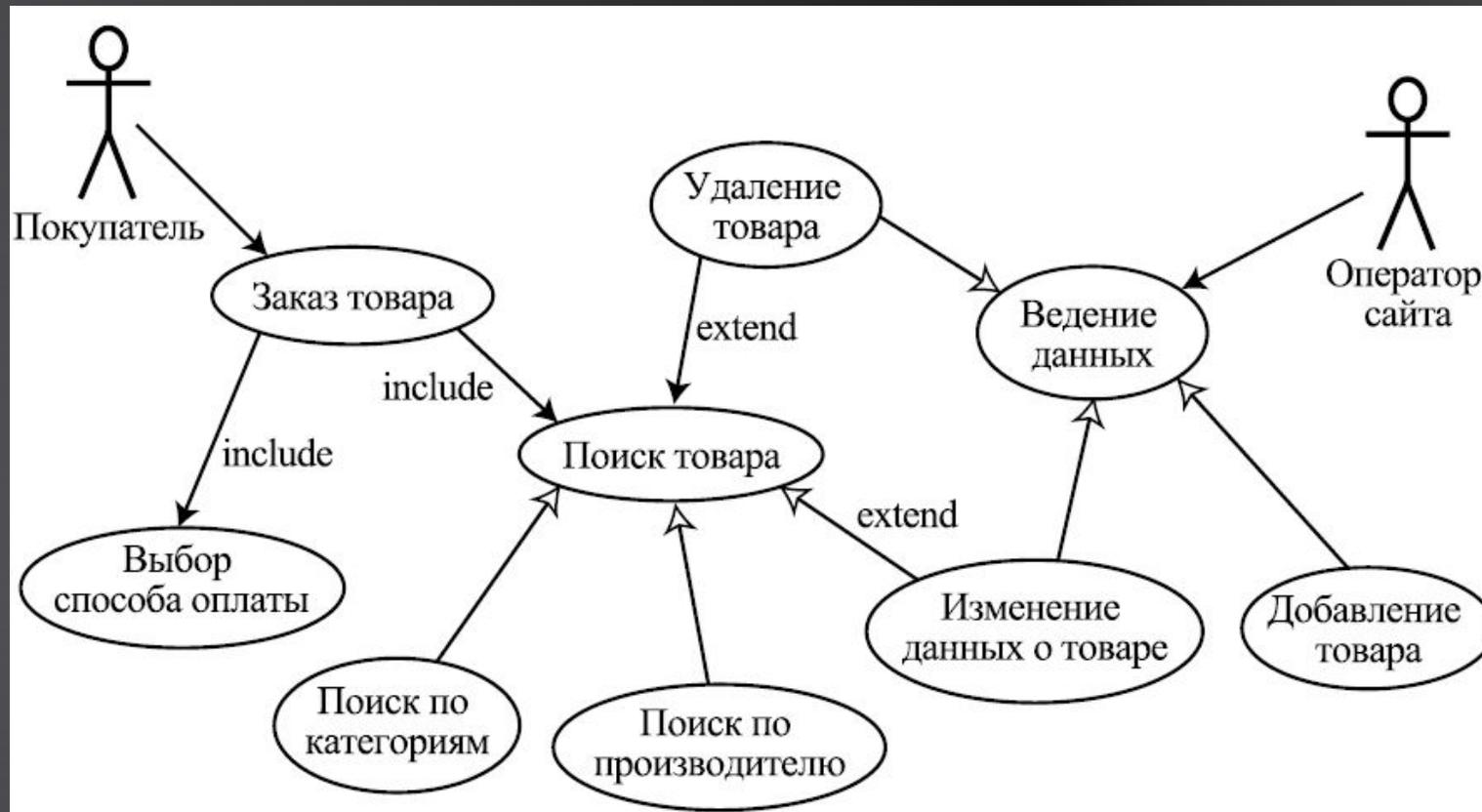
Основные UML-диаграммы



Основная цель

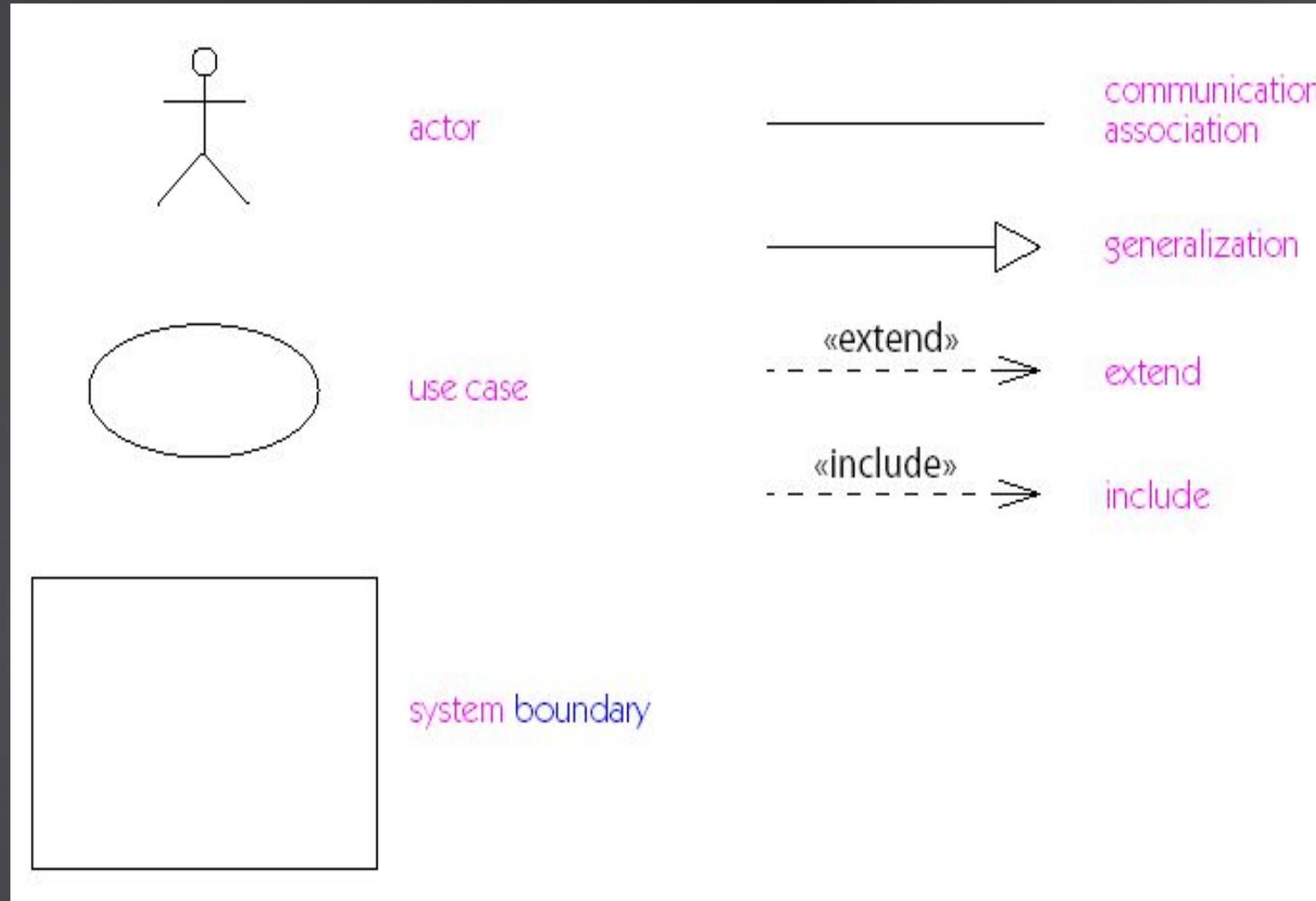
UML-диаграмм – описать программную
систему с разных сторон

Диаграмма прецедентов (вариантов использования)



**Диаграмма прецедентов
предназначена для анализа
функциональности программной
системы**

Основные обозначения на диаграмме вариантов использования



Система автоматизации заказов обедов в офис

- Секретарь размещает на сервере меню обеденных блюд на неделю
- Сотрудники должны иметь возможность ознакомиться с меню и сделать заказ, выбрав блюда на каждый день следующей недели

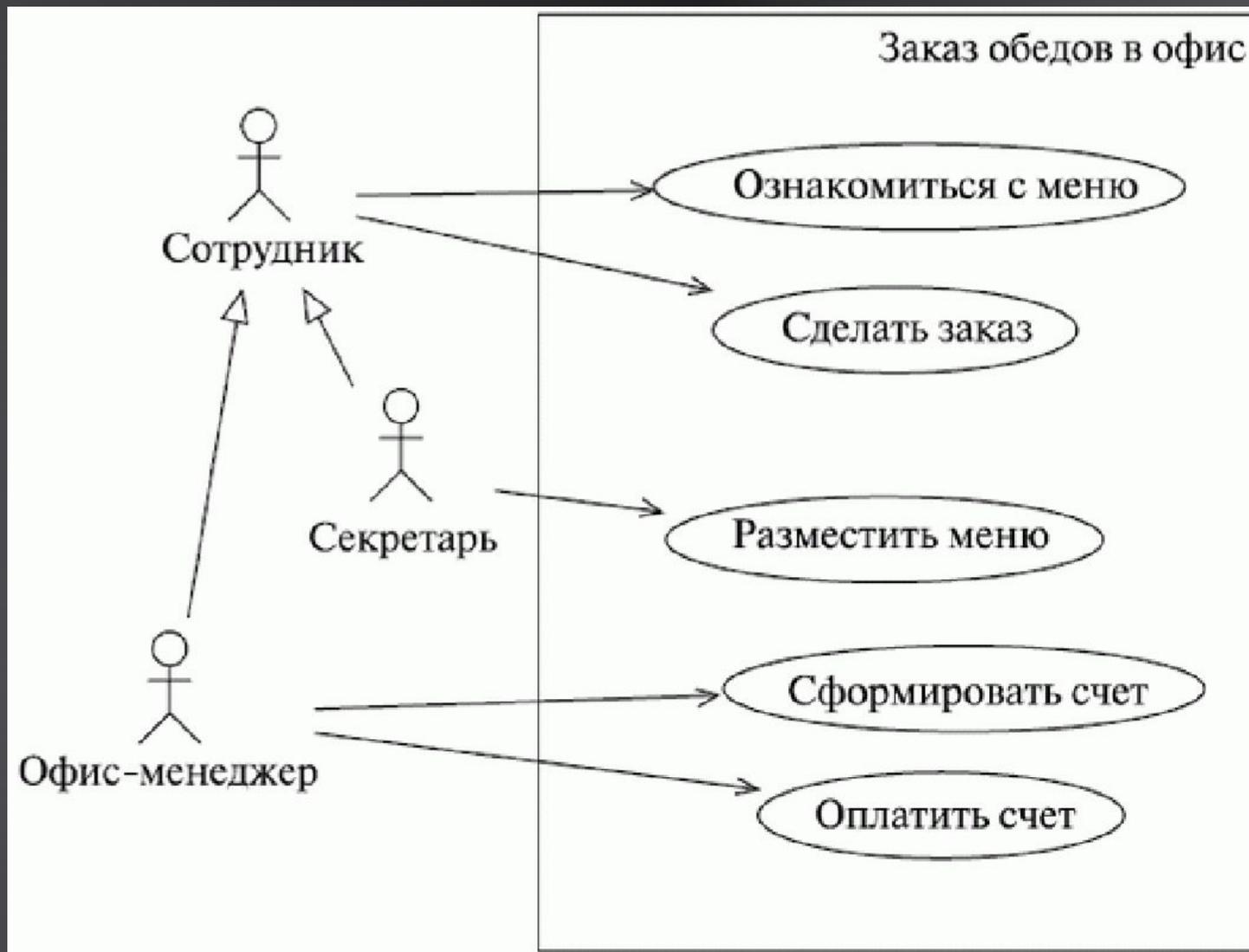
Система автоматизации заказов обедов в офис

- Офис-менеджер должен иметь возможность сформировать счет и оплатить его
- Система должна быть написана на ASP.NET

Таблица с описанием требований

Прецедент	Действующее лицо
разместить меню	секретарь
ознакомиться с меню	сотрудник, секретарь, офис-менеджер
сделать заказ	сотрудник, секретарь, офис-менеджер
сформировать счет	офис-менеджер
оплатить счет	офис-менеджер

Отношение ассоциации



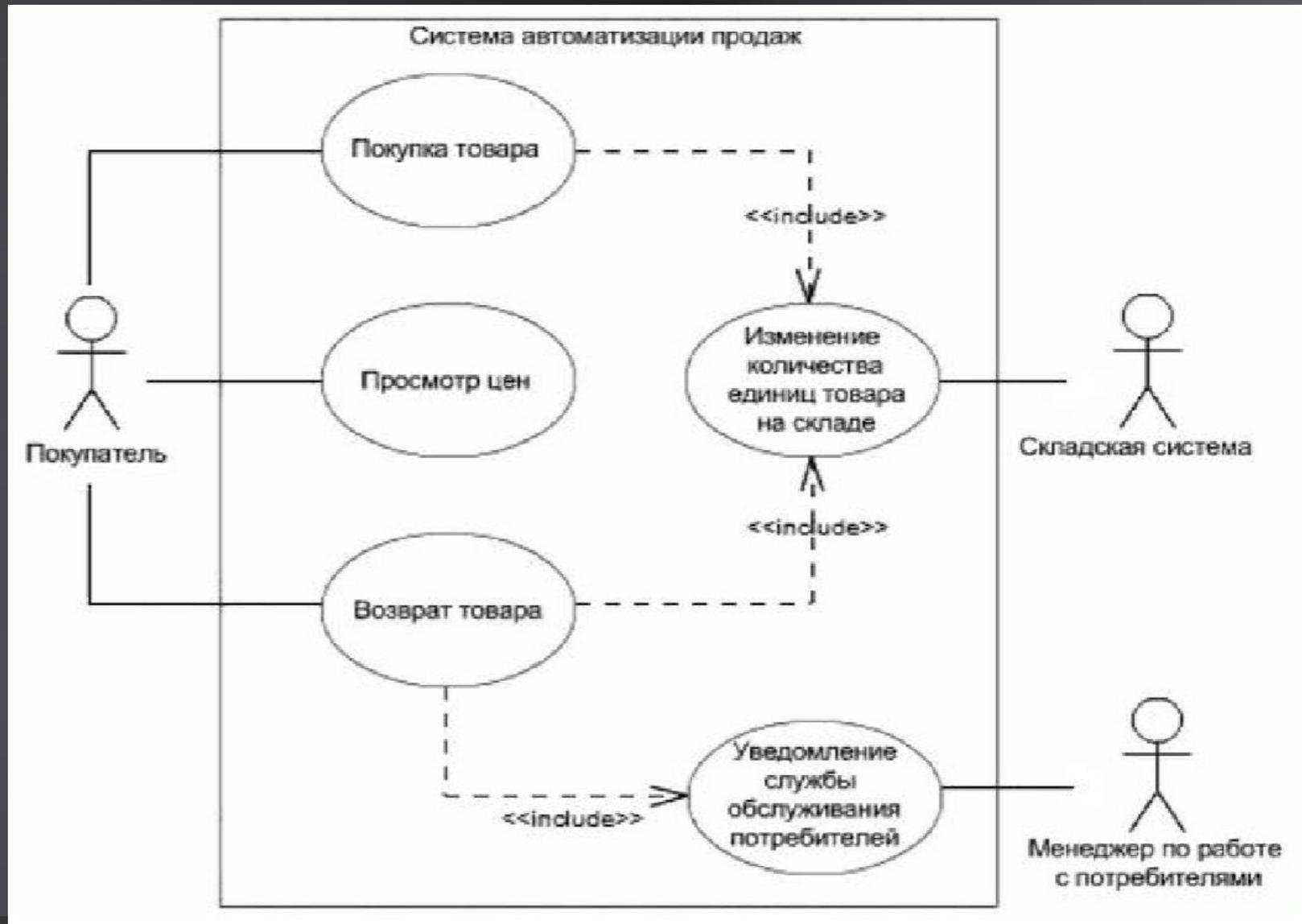
Применительно к диаграммам вариантов использования отношение ассоциации (*association*) может служить только для обозначения взаимодействия актера с вариантом использования

Система автоматизации продаж

- В момент считывания кассиром штрих-кода обновляется состояние базы данных товаров, - количество наличных единиц купленного товара уменьшается
- Если купленный товар оказался бракованным, то также обновляется состояние базы данных товаров, - количество наличных единиц товара увеличивается

Оба вышеуказанных действия - и покупка,
и возврат - содержат (включают в себя)
такое действие, как обновление
содержимого БД

Отношение включения



Отношение включения (include)
специфицирует тот факт, что
некоторый вариант использования
содержит поведение, определенное в
другом варианте использования

Система оплаты товара

- Оплатить товар наличными, если сумма не превышает \$ 100
- Оплатить кредитной картой, если сумма находится в пределах от \$ 100 до \$ 1000
- Если же сумма превышает \$ 1000, то придется брать кредит

Эти случаи возникают только при строго определенных условиях: когда цена товара попадает в определенные рамки

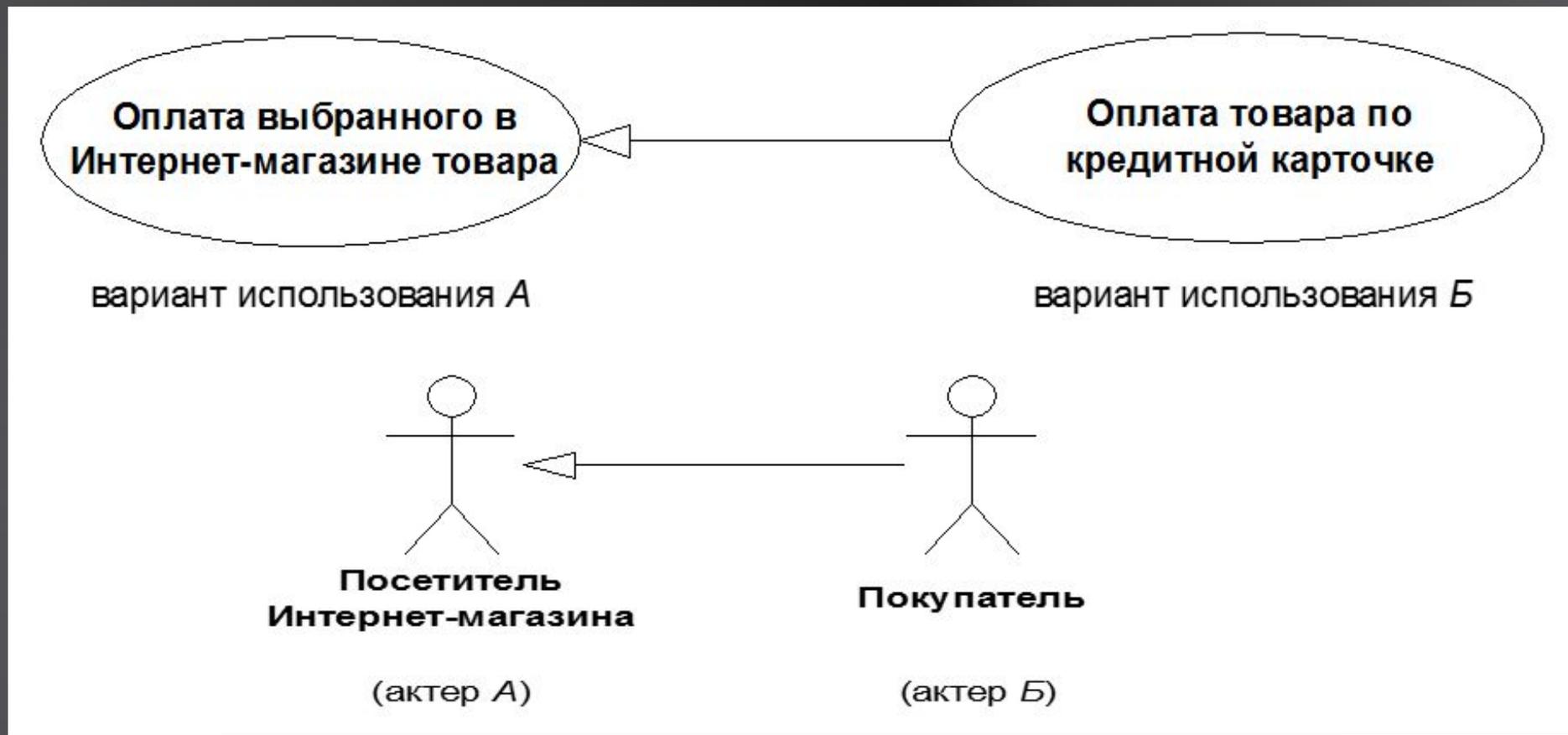
Отношение расширения



Отношение расширения (extend)

определяет взаимосвязь одного варианта использования с другим вариантом использования, функциональность которого задействуется первым не всегда, а только при выполнении некоторых дополнительных условий

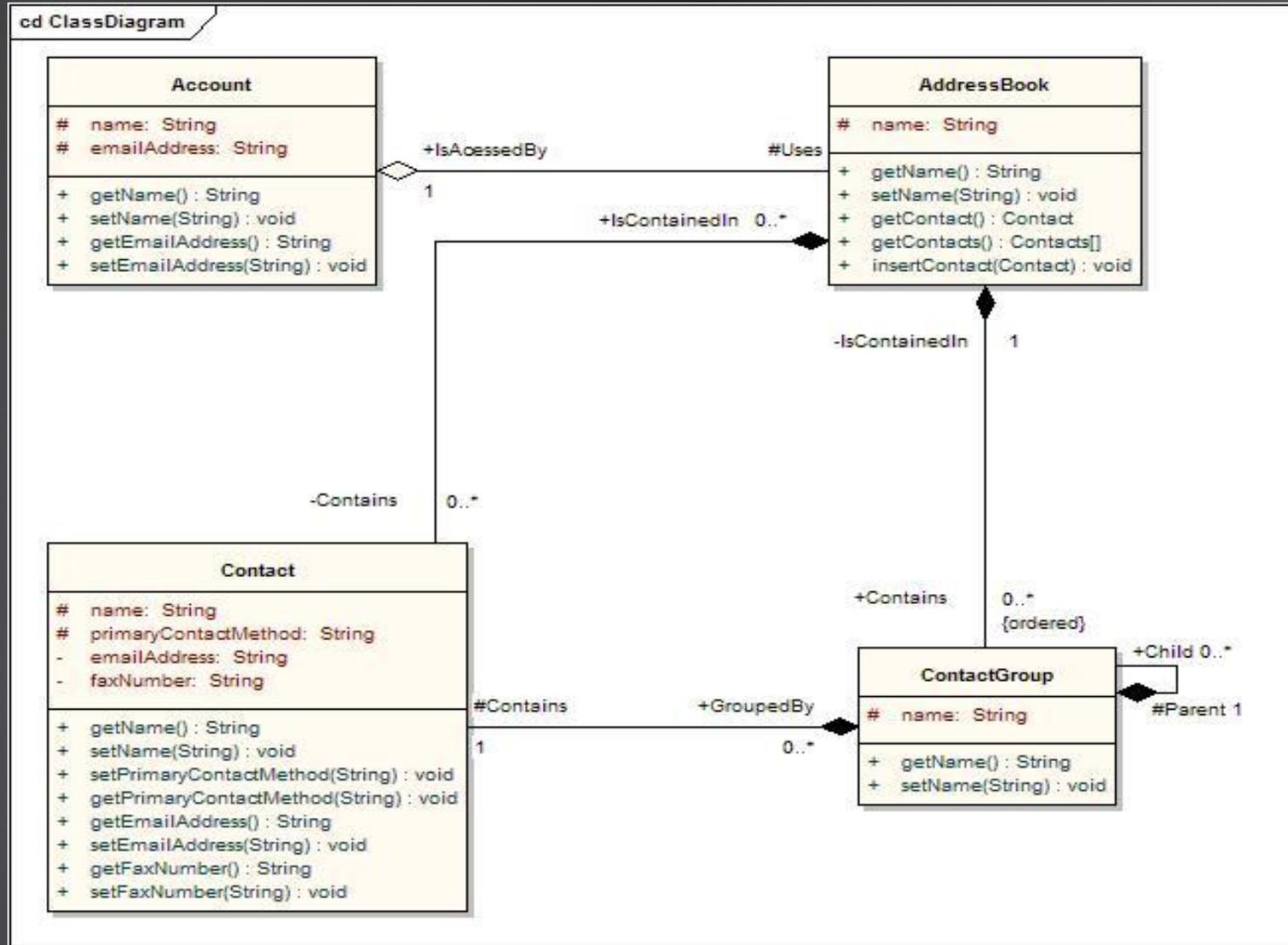
Отношение генерализации



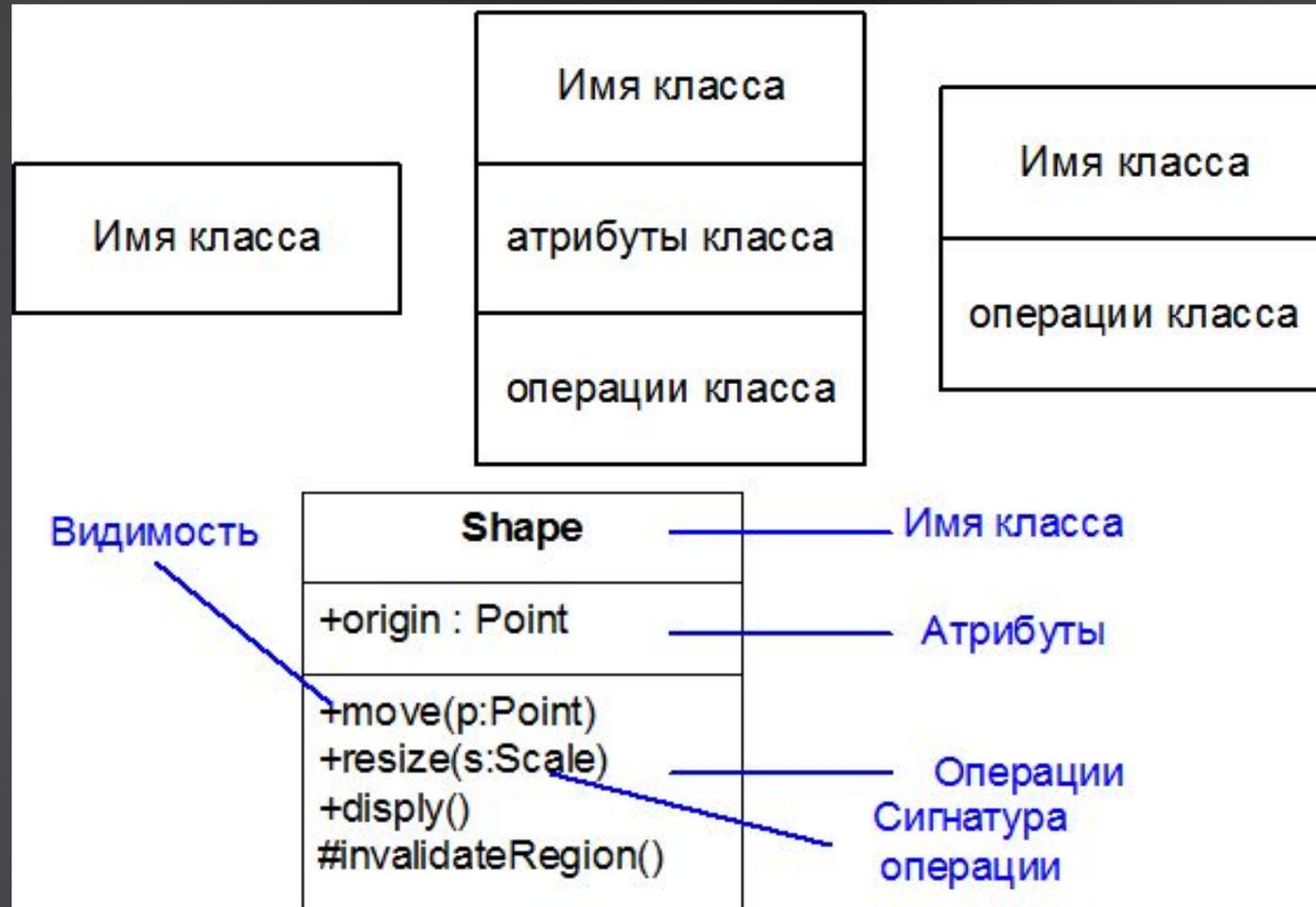
Отношение обобщения (generalization relationship) предназначено для спецификации того факта, что один элемент модели является частным случаем другого элемента модели

**Диаграмма классов предназначена для
описания структуры программной
системы**

Диаграмма классов



Варианты графического изображения класса на диаграмме классов



Вид видимости

- + public (общедоступный)
- - private (закрытый)
- # protected (защищенный)
- ~ package (пакет)

Примеры записи атрибутов

+ имяСотрудника : String

~ датаРождения : Data

возрастСотрудника : Integer

+ номерТелефона : Integer [1..*]

– заработнаяПлата : Currency = 500.00

Примеры записи операций

+добавить(in номерТелефона : Integer
[1..*])

+изменить(in
заработнаяПлата : Currency)

+создать() : Boolean

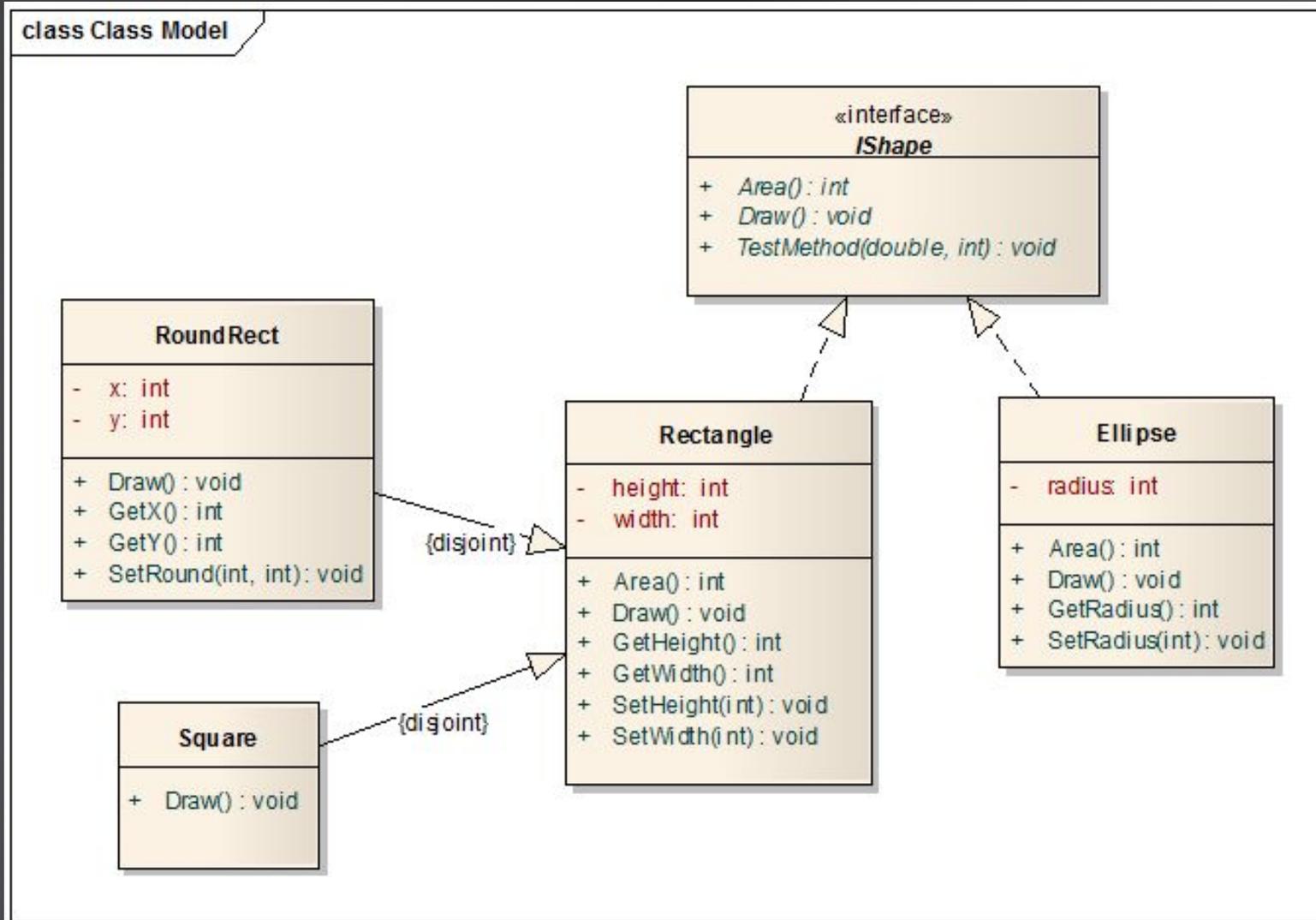
+toString(return : String)

+toString() : String

Виды отношений между классами

- Реализация
- Ассоциация
- Генерализация
- Агрегация
- Композиция

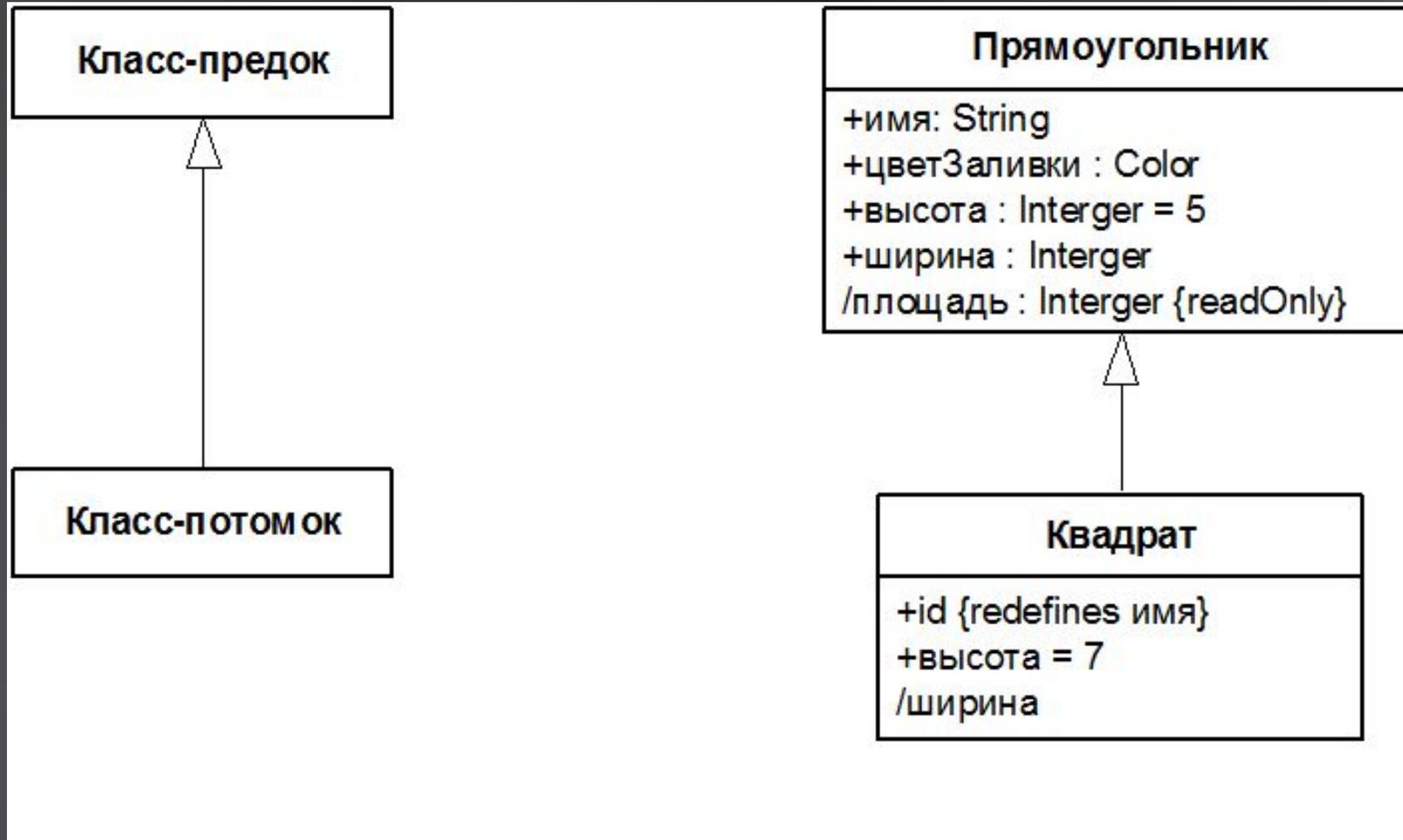
Отношение реализации



Отношение ассоциации



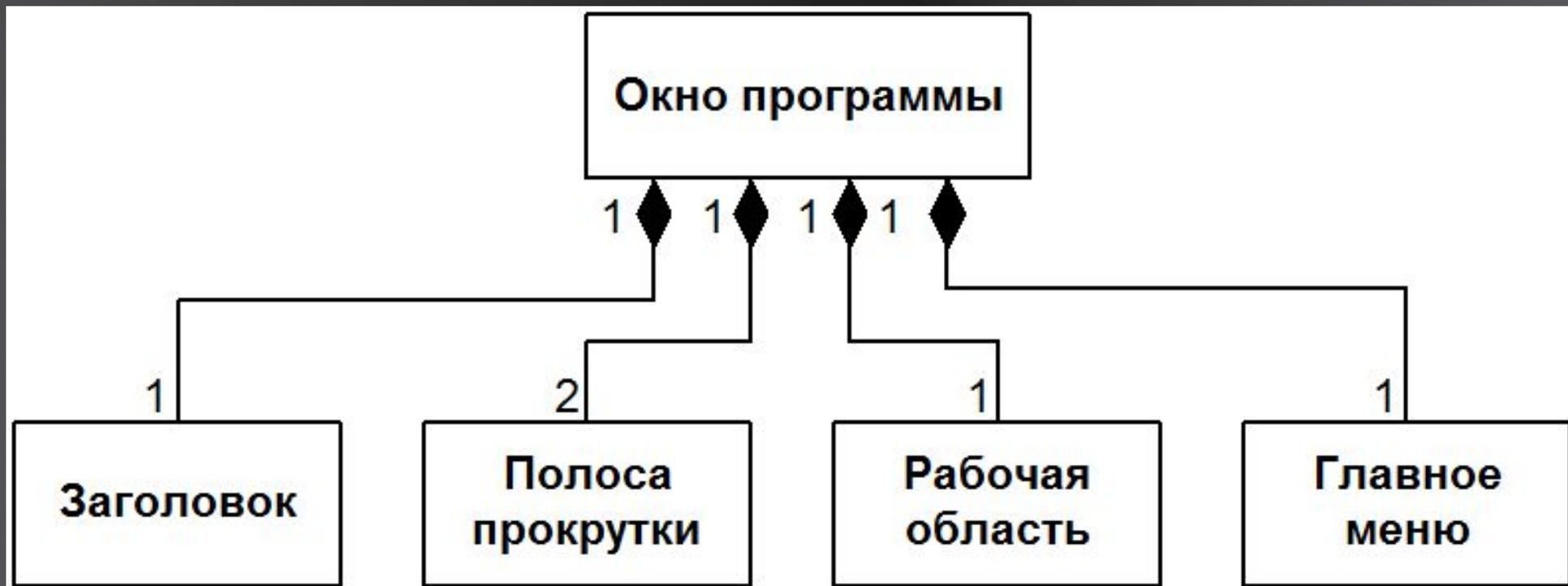
Отношение генерализации



Отношение агрегации

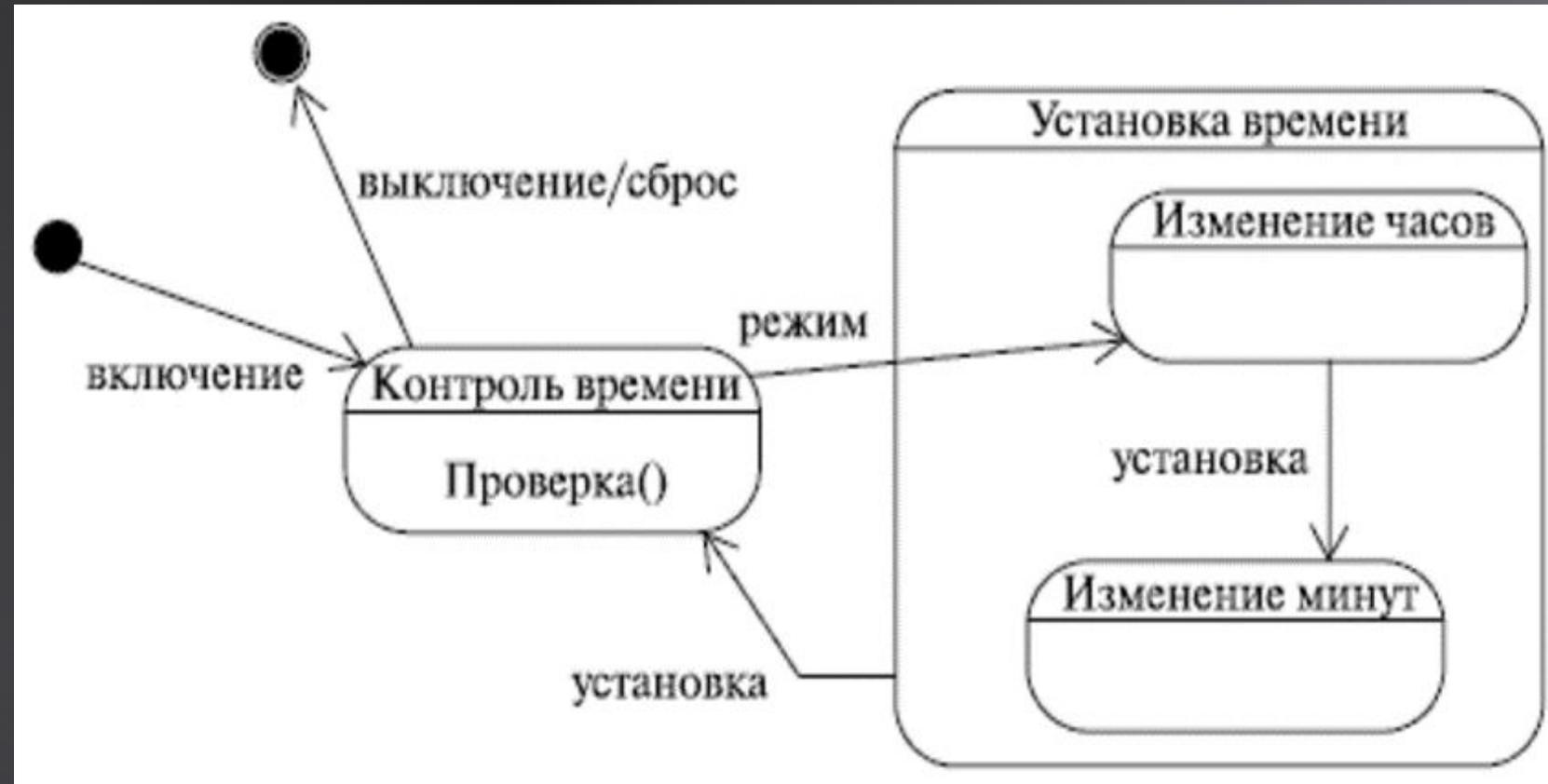
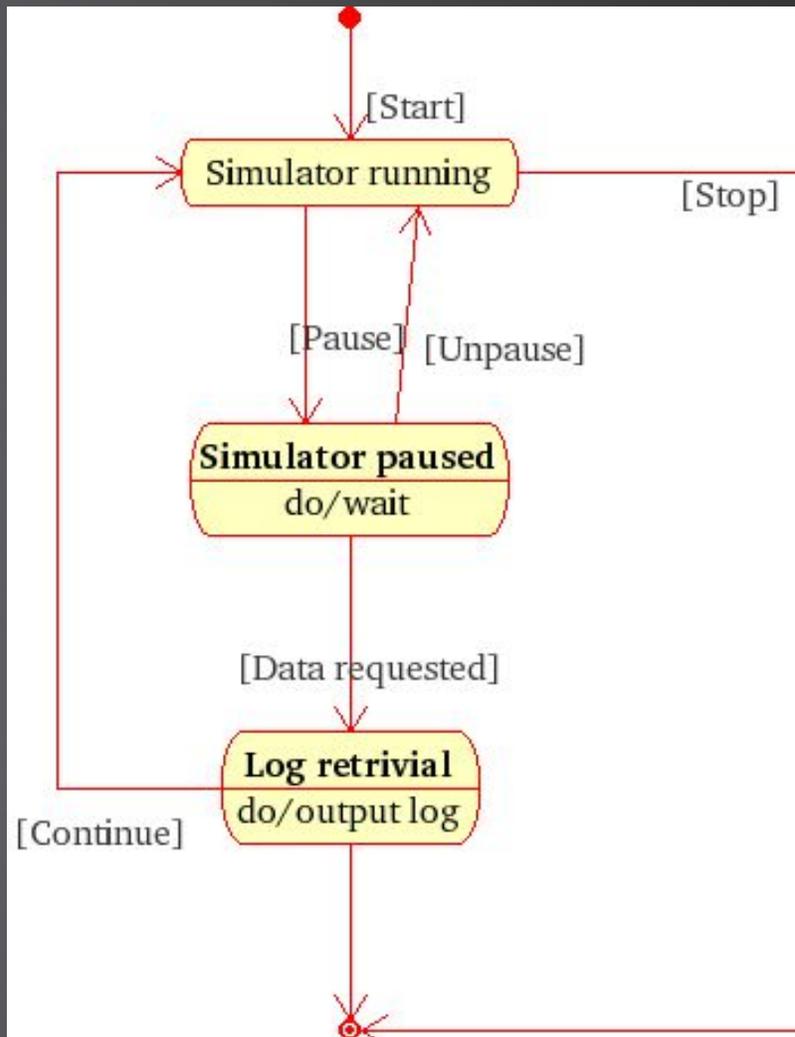


Отношение композиции



**Диаграмма состояний - диаграмма,
которая представляет конечный
автомат**

Диаграмма состояний

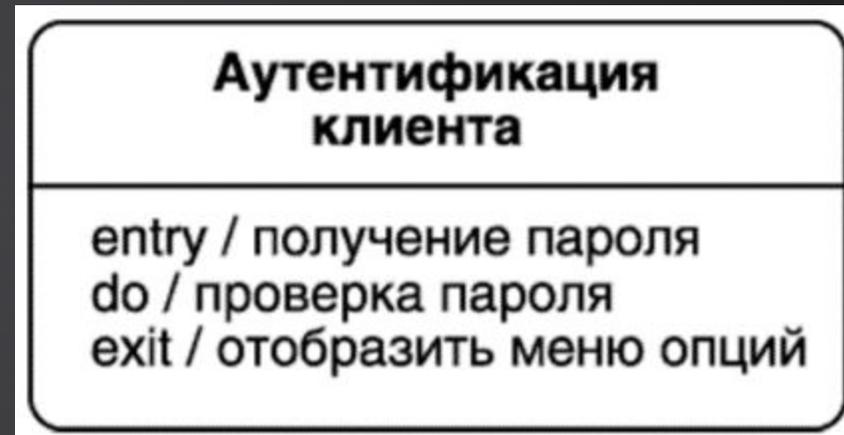


Вершинами графа конечного автомата являются **СОСТОЯНИЯ**

Дуги графа служат для обозначения
переходов из состояния в состояние

Диаграмма состояний позволяет описать
последовательности состояний и
переходов, которые в совокупности
характеризуют поведение моделируемой
системы в течение всего жизненного
цикла

Графическое изображение состояний на диаграмме состояний



**Входное действие (entry action) -
действие, которое выполняется в
момент перехода в данное состояние**

**Действие выхода (exit action) -
действие, производимое при
выходе из данного состояния**

Внутренняя деятельность (do activity) - выполнение объектом операций или процедур, которые требуют определенного времени

**Псевдосостояние (pseudo-state) -
вершина в конечном автомате, которая
имеет форму состояния, но не обладает
поведением**

Примерами псевдосостояний, которые определены в языке UML, являются начальное и конечное состояния

**Начальное состояние (start state) -
разновидность псевдосостояния,
обозначающая начало выполнения
процесса изменения состояний конечного
автомата**

Конечное состояние (final state) -
разновидность псевдосостояния,
обозначающая прекращение процесса
изменения состояний конечного автомата

Псевдосостояния



начальное состояние

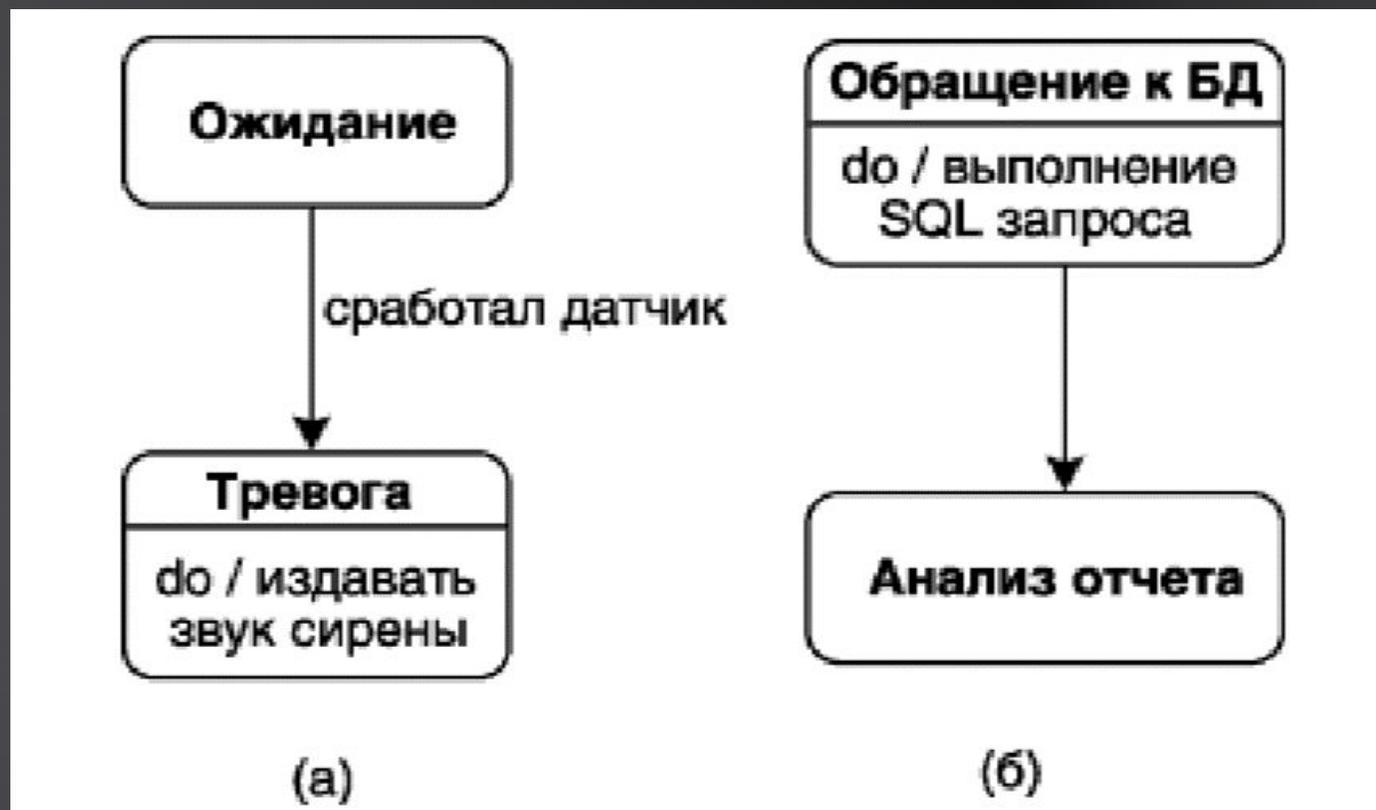
конечное состояние

Переход (transition) - отношение между двумя состояниями, которое указывает на то, что объект в первом состоянии должен выполнить определенные действия и перейти во второе состояние

Переход называется триггерным, если его специфицирует событие-триггер, связанное с внешними условиями по отношению к рассматриваемому состоянию

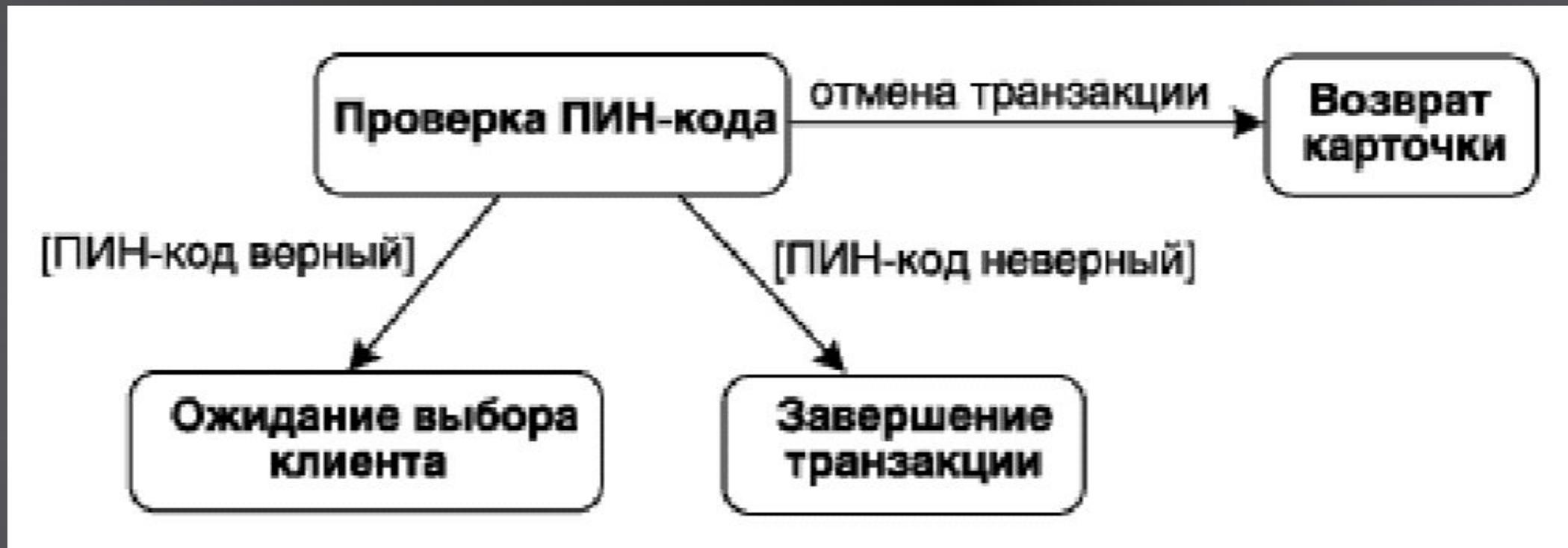
**Переход называется нетриггерным,
если он происходит по завершении
выполнения внутренней
деятельности в данном состоянии**

Графическое изображение триггерного (а) и нетриггерного (б) переходов на диаграмме состояний



Сторожевое условие (guard condition) - логическое условие, записанное в прямых скобках и представляющее собой булевское выражение

Триггерные и нетриггерные переходы на диаграмме состояний



Выражение действия перехода на диаграмме состояний

