Лекция 4

БАЗИС ЯЗЫКА ВИЗУАЛЬНОГО МОДЕЛИРОВАНИЯ

языки визуального моделирования

- Проектирование ПО – это процесс разработки, следующий за этапом анализа и формирования требований. Задачей проектирования является преобразование требований к системе в проектные решения, требования к ПО и построение архитектуры системы.
- Для создания моделей анализа и проектирования объектно-ориентированных программных систем используют языки визуального моделирования. Появившись сравнительно недавно, в период с 1989 по 1997 год, эти языки уже имеют представительную историю развития.
- В настоящее время различают три поколения языков визуального моделирования. И если первое поколение образовали 10 языков, то численность второго поколения уже превысила 50 языков. Среди наиболее популярных языков 2-го поколения можно выделить: язык Буча (G. Booch), язык Рамбо (J. Rumbaugh), язык Джекобсона (I. Jacobson), язык Коада-Йордона (Coad-Yourdon), язык Шлеера-Меллора (Shlaer-Mellor) и т. д [41], [64], [69].

языки визуального моделирования

- Каждый язык вводил свои выразительные средства,— претендовал на роль единственного и неповторимого языка. В результате разработчики (и пользователи этих языков) перестали понимать друг друга. Возникла острая необходимость унификации языков.
- Идея унификации привела к появлению языков 3-го поколения.
- В качестве стандартного языка третьего поколения был принят Unified Modeling Language (UML), создававшийся в 1994-1997 годах (основные разработчики — три «amigos» Г. Буч, Дж. Рамбо, И. Джекобсон).

Унифицированный язык моделирования

- **UML** — стандартный язык для написания моделей анализа, проектирования и реализации объектно-ориентированных программных систем. UML может использоваться для визуализации, спецификации, конструирования и документирования результатов программных проектов.
- UML — это не визуальный язык программирования, но его модели прямо транслируются в текст на языках программирования (Java, C++, Visual Basic, Ada 95, Object Pascal) и даже в таблицы для реляционной БД.
- **Словарь UML** образуют три разновидности строительных блоков: предметы, отношения, диаграммы.
- **Предметы** — это абстракции, которые являются основными элементами в модели, отношения связывают эти предметы, диаграммы группируют коллекции предметов.

Предметы в UML

- В UML имеются четыре разновидности предметов:
- структурные предметы;
- предметы поведения;
- группирующие предметы;
- поясняющие предметы.

  Эти предметы являются базовыми объектно-ориентированными строительными блоками. Они используются для написания моделей.
- ***Структурные предметы*** являются существительными в UML-моделях. Они представляют статические части модели — понятийные или физические элементы. Перечислим восемь разновидностей структурных предметов.

Класс

1. ***Класс*** — описание множества объектов, которые разделяют одинаковые свойства, операции, отношения и семантику (смысл). Класс реализует один или несколько интерфейсов. Как показано на рис. 10.1, графически класс отображается в виде прямоугольника, обычно включающего секции с именем, свойствами (атрибутами) и операциями.

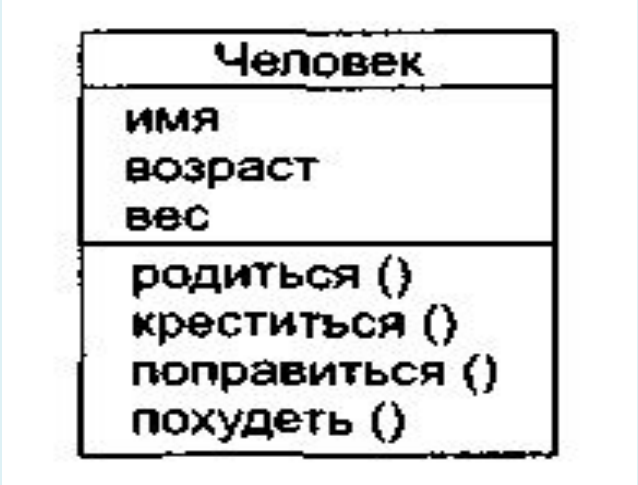


Рис. 10.1. Классы

Интерфейс

2. ***Интерфейс*** — набор операций, которые определяют услуги класса или компонента.

Интерфейс описывает поведение элемента, видимое извне. Интерфейс может представлять полные услуги класса или компонента или часть таких услуг. Интерфейс определяет набор спецификаций операций, а не набор реализаций операций. Графически интерфейс изображается в виде кружка с именем, как показано на рис.

Имя интерфейса обычно начинается с буквы «I». Интерфейс редко показывают самостоятельно. Обычно его присоединяют к классу или компоненту, который реализует интерфейс.

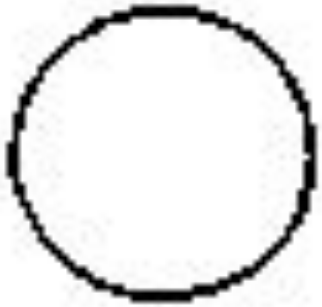
Кооперация

3. Кооперация (сотрудничество) определяет взаимодействие и является совокупностью ролей и других элементов, которые работают вместе для обеспечения коллективного поведения более сложного, чем простая сумма всех элементов.

Таким образом, кооперации имеют как структурное, так и поведенческое измерения. Конкретный класс может участвовать в нескольких кооперациях.

Графически кооперация изображается как пунктирный эллипс, в который вписывается ее имя.

Графическия кооперация



Учеба

Рис. 10.2. Интерфейсы

Обслуживание клиента

Актер

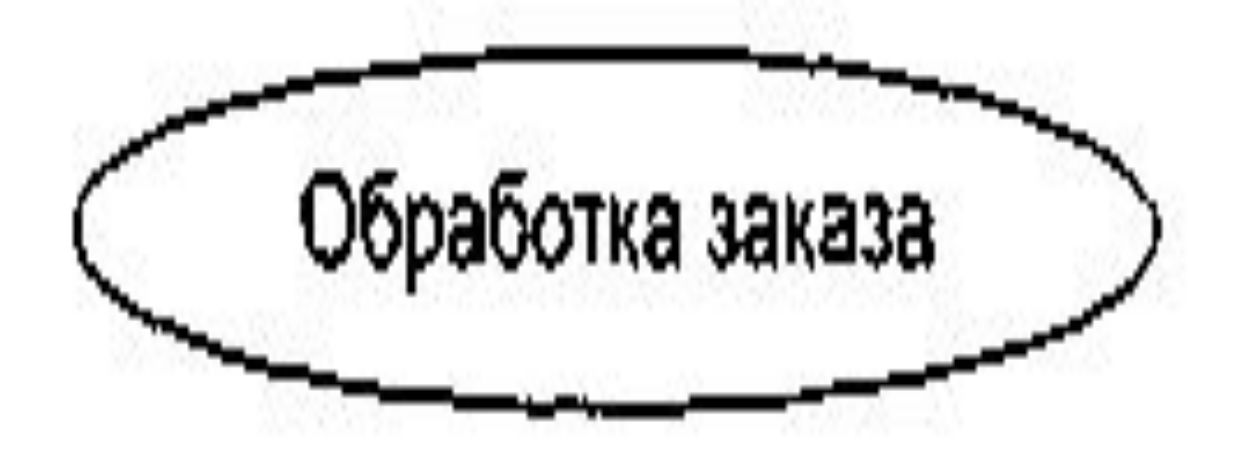
4. ***Актер*** — набор согласованных ролей, которые могут играть пользователи при взаимодействии с системой (ее элементами Use Case). Каждая роль требует от системы определенного поведения.

- Как показано на рис., актер изображается как проволочный человечек с именем.

Элемент Use Case

5. ***Элемент Use Case*** (Прецедент) — описание последовательности действий (или нескольких последовательностей), выполняемых системой в интересах отдельного актера и производящих видимый для актера результат.

- В модели элемент Use Case применяется для структурирования предметов поведения. Элемент Use Case реализуется кооперацией. Как показано на рис., элемент Use Case изображается как эллипс, в который вписывается его имя.



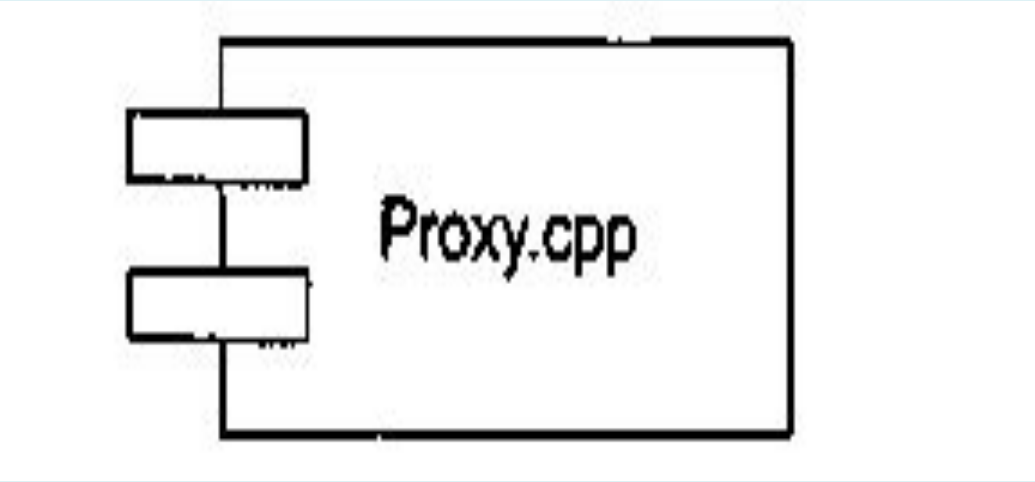
Активный класс

6. ***Активный класс*** — класс, чьи объекты имеют один или несколько процессов (или потоков) и поэтому могут инициировать управляющую деятельность. Активный класс похож на обычный класс за исключением того, что его объекты действуют одновременно с объектами других классов. Как показано на рис. активный класс изображается как утолщенный прямоугольник, обычно включающий имя, свойства (атрибуты) и операции.

Компонент

7. ***Компонент***— физическая и заменяемая часть системы, которая соответствует набору интерфейсов и обеспечивает реализацию этого набора интерфейсов.

   Обычно компонент — это физическая упаковка различных логических элементов (классов, интерфейсов и сотрудничеств). Как показано на рис., компонент изображается как прямоугольник с вкладками, обычно включающий имя.

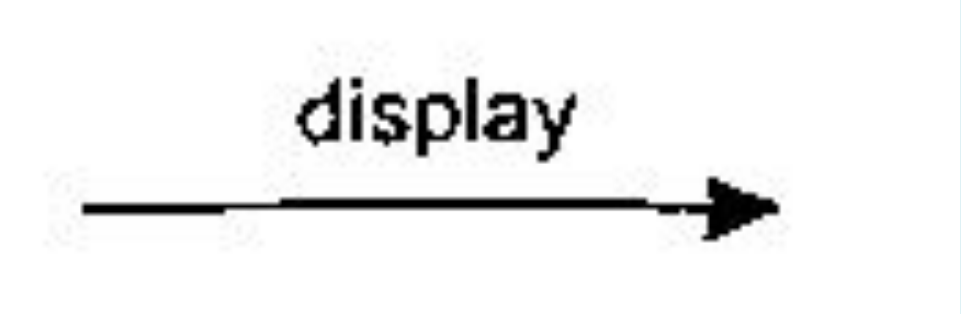


Узел

8. ***Узел*** — физический элемент, который существует в период работы системы и представляет ресурс, обычно имеющий память и возможности обработки. В узле размещается набор компонентов, который может перемещаться от узла к узлу. Как показано на рис. 10.8, узел изображается как куб с именем.

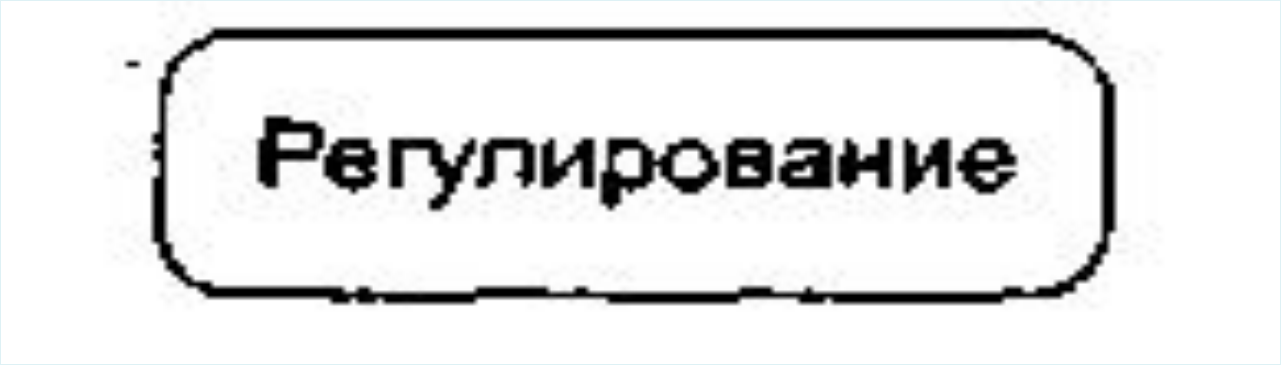
Предметы поведения

- ***Предметы поведения*** — динамические части UML-моделей. Они являются глаголами моделей, представлением поведения во времени и пространстве.
- ***Взаимодействие*** — поведение, заключающее в себе набор сообщений, которыми обменивается набор объектов в конкретном контексте для достижения определенной цели. Взаимодействие может определять динамику как совокупности объектов, так и отдельной операции. Элементами взаимодействия являются сообщения, последовательность действий (поведение, вызываемое сообщением) и связи (соединения между объектами). Как показано на рис. сообщение изображается в виде направленной линии с именем ее операции.



Конечный автомат

- ***Конечный автомат*** — поведение, которое определяет последовательность состояний объекта или взаимодействия, выполняемые в ходе его существования в ответ на события (и с учетом обязанностей по этим событиям). С помощью конечного автомата может определяться поведение индивидуального класса или кооперации классов. Элементами конечного автомата являются состояния, переходы (от состояния к состоянию), события (предметы, вызывающие переходы) и действия (реакции на переход). Как показано на рис., состояние изображается как закругленный прямоугольник, обычно включающий его имя и его подсостояния (если они есть).



Группирующие предметы

- *Группирующие предметы* — организационные части UML-моделей. Это ящики, по которым может быть разложена модель. Предусмотрена одна разновидность группирующего предмета — пакет.
- *Пакет* — общий механизм для распределения элементов по группам. В пакет могут помещаться структурные предметы, предметы поведения и даже другие группировки предметов. В отличие от компонента (который существует в период выполнения), пакет — чисто концептуальное понятие. Это означает, что пакет существует только в период разработки. Как показано на рис., пакет изображается как папка с закладкой, на которой обозначено его имя и, иногда, его содержание.

- Пакеты

Поясняющие предметы

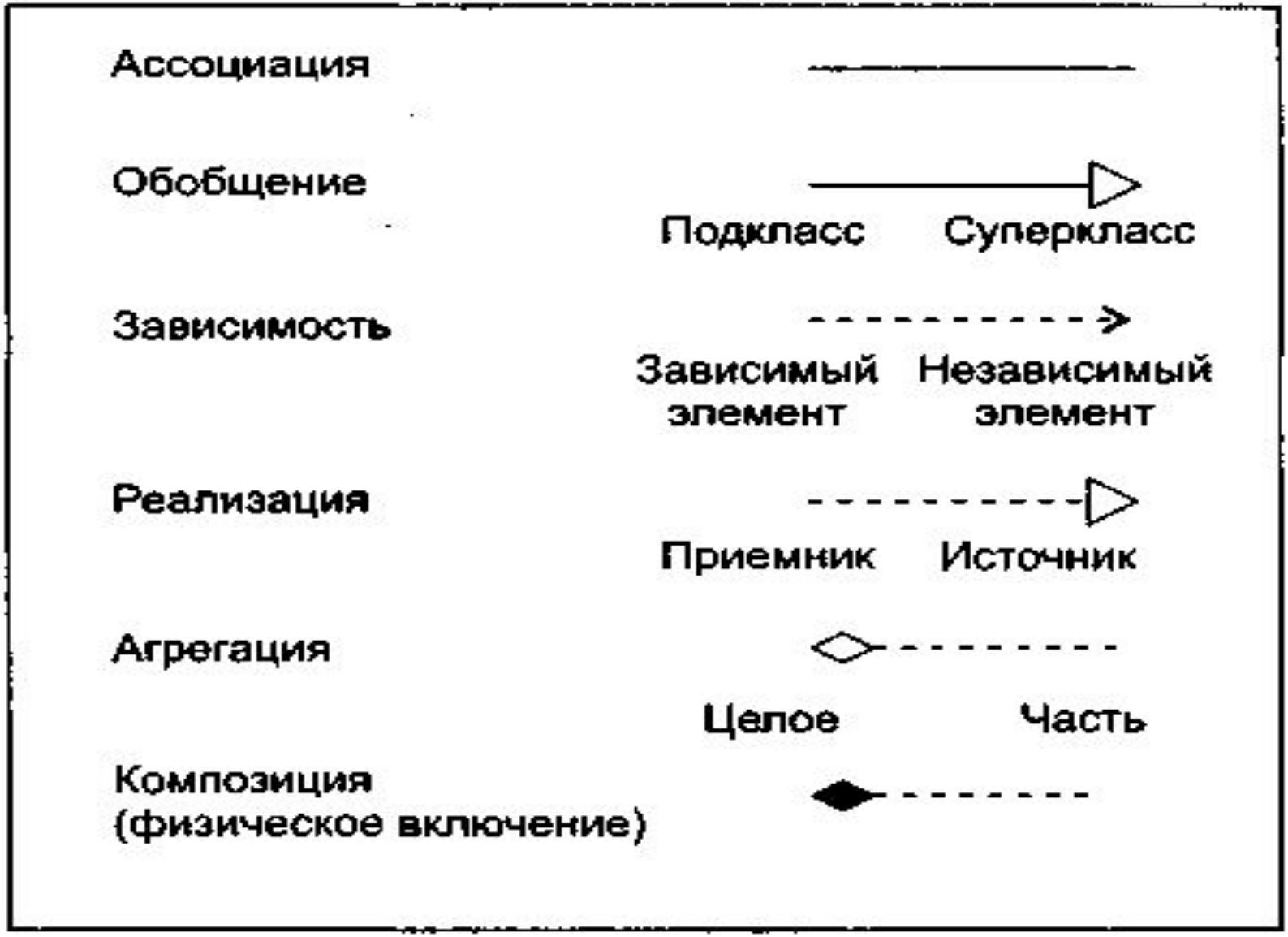
- *Поясняющие предметы* — разъясняющие части UML-моделей. Они являются замечаниями, которые можно применить для описания, объяснения и комментирования любого элемента модели. Предусмотрена одна разновидность поясняющего предмета — примечание.

УСТАЛИ!

Отношения в UML

- В UML имеются четыре разновидности отношений:
- 1) зависимость;
- 2) ассоциация;
- 3) обобщение;
- 4) реализация.
- Эти отношения являются базовыми строительными блоками отношений. Они используются при написании моделей.

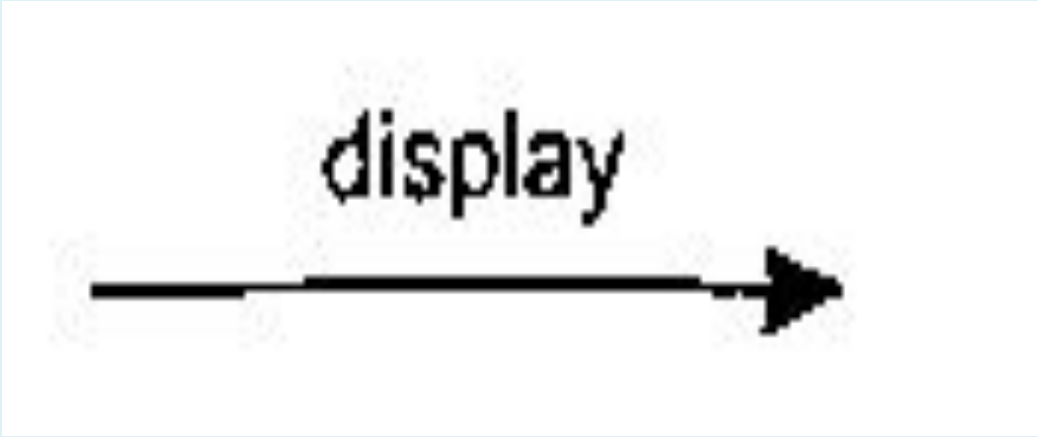
Отношения в диаграммах классов



- ***Ассоциации*** отображают структурные отношения между экземплярами классов, то есть соединения между объектами.
- ***Обобщение*** — отношение между общим предметом (суперклассом) и специализированной разновидностью этого предмета (подклассом). Подкласс может иметь одного родителя (один суперкласс) или несколько родителей (несколько суперклассов). Во втором случае говорят о множественном наследовании.
- ***Зависимость*** является отношением использования между клиентом (зависимым элементом) и поставщиком (независимым элементом).
- ***Реализация*** — семантическое отношение между классами, в котором класс-приемник выполняет реализацию операций интерфейса класса-источника.
- ***Агрегация*** -процесс объединения элементов в одну систему
- ***Композиция*** (агрегирование) — методика создания нового класса в объектно-ориентированном программировании из уже существующих путём включения

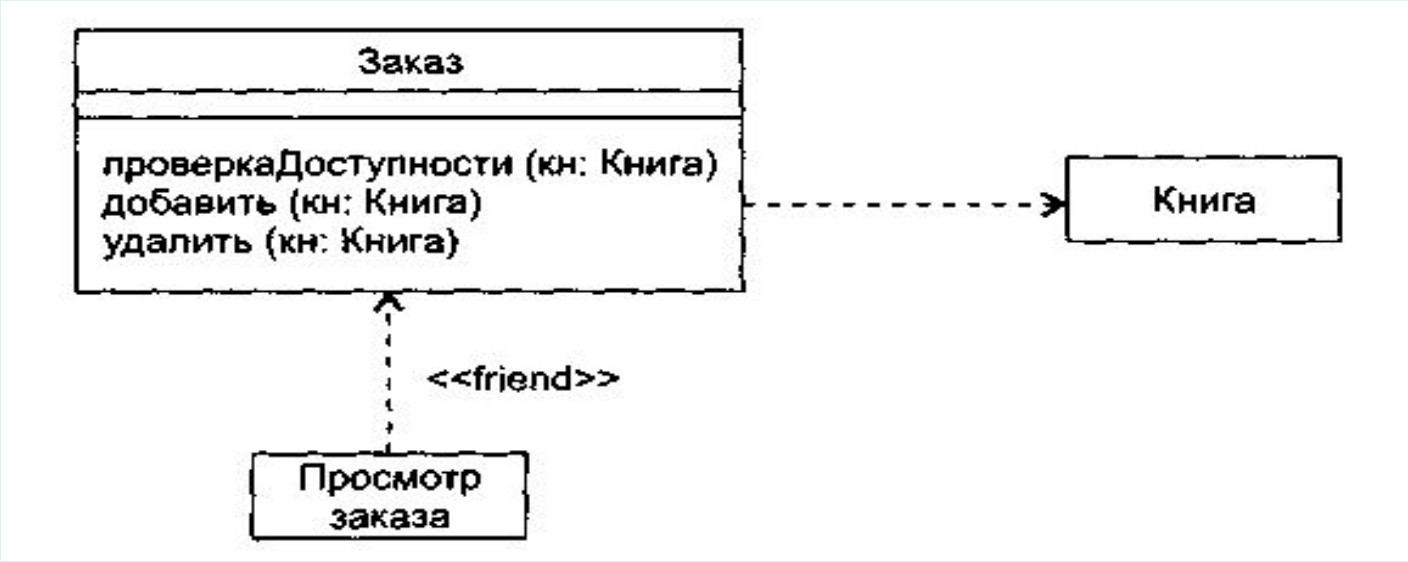
Зависимость

- ***Зависимость*** — семантическое отношение между двумя предметами, в котором изменение в одном предмете (независимом предмете) может влиять на семантику другого предмета (зависимого предмета). Как показано на рис., зависимость изображается в виде пунктирной линии, возможно направленной на независимый предмет и иногда имеющей метку.



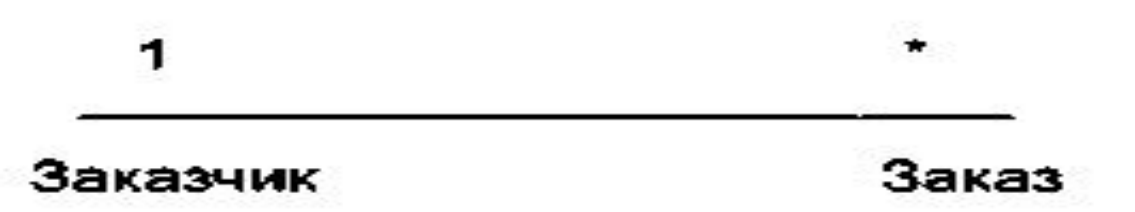
Зависимость

- ***Зависимость*** — семантическое отношение между двумя предметами, в котором изменение в одном предмете (независимом предмете) может влиять на семантику другого предмета (зависимого предмета). Как показано на рис., зависимость изображается в виде пунктирной линии, возможно направленной на независимый предмет и иногда имеющей метку.



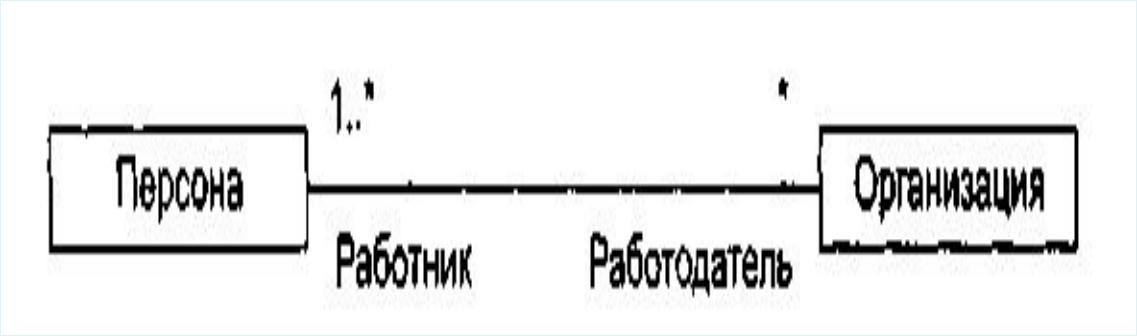
Ассоциация

- ***Ассоциация*** — структурное отношение, которое описывает набор связей, являющихся соединением между объектами, ассоциация изображается в виде сплошной линии, возможно направленной, иногда имеющей метку и часто включающей другие «украшения», такие как мощность и имена ролей.
- **Агрегация** — это специальная разновидность ассоциации, представляющая структурное отношение между целым и его частями. Как показано на рис



Один и тот же класс в разных ассоциациях может играть разные роли.

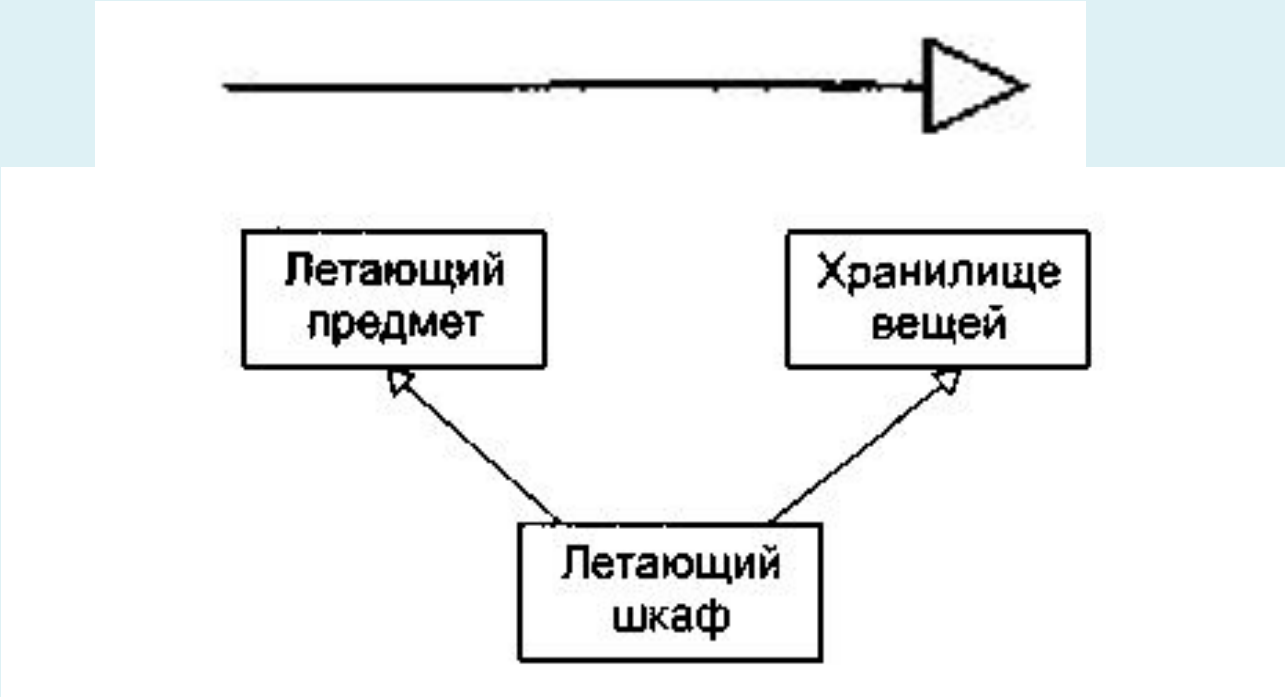
- Часто важно знать, как много объектов может соединяться через экземпляр ассоциации. Это количество называется ложностью роли в ассоциации, записывается в виде выражения, задающего диапазон величин или одну величину



- Запись мощности на одном конце ассоциации определяет количество объектов, соединяемых с каждым объектом на противоположном конце ассоциации. Например, можно задать следующие варианты мощности:

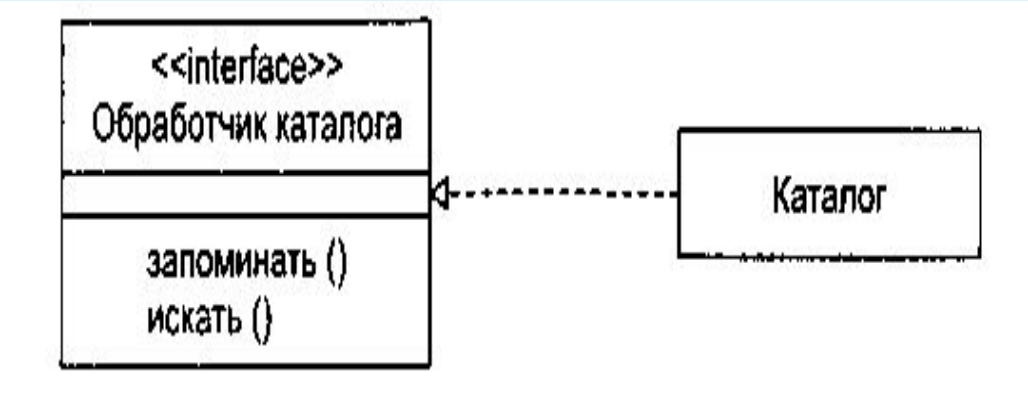
Обобщение

- ***Обобщение*** — отношение специализации/обобщения, в котором объекты специализированного элемента (потомка, ребенка) могут заменять объекты обобщенного элемента (предка, родителя). Иначе говоря, потомок разделяет структуру и поведение родителя. Как показано на рис., обобщение изображается в виде сплошной стрелки с полым наконечником, указывающим на родителя.



Реализация

- ***Реализация*** — семантическое отношение между классификаторами, где один классификатор определяет контракт, который другой классификатор обязуется выполнять (к классификаторам относят классы, интерфейсы, компоненты, элементы Use Case, кооперации). Отношения реализации применяют в двух случаях: между интерфейсами и классами (или компонентами), реализующими их; между элементами Use Case и кооперациями, которые реализуют их. Как показано на рис., реализация изображается как нечто среднее между обобщением и зависимостью.



Диаграммы в UML

- ***Диаграмма*** — графическое представление множества элементов, наиболее часто изображается как связный граф из вершин (предметов) и дуг (отношений).
- Диаграммы рисуются для визуализации системы с разных точек зрения, затем они отображаются в систему. Обычно диаграмма дает неполное представление элементов, которые составляют систему. Хотя один и тот же элемент может появляться во всех диаграммах, на практике он появляется только в некоторых диаграммах. Теоретически диаграмма может содержать любую комбинацию предметов и отношений, на практике ограничиваются малым количеством комбинаций, которые соответствуют пяти представлениям архитектуры ПС. По этой причине UML включает девять видов диаграмм:

Виды диаграмм

- 1) диаграммы классов;
- 2) диаграммы объектов;
- 3) диаграммы Use Case (диаграммы прецедентов);
- 4) диаграммы последовательности;
- 5) диаграммы сотрудничества (кооперации);
- 6) диаграммы схем состояний;
- 7) диаграммы деятельности;
- 8) компонентные диаграммы;
- 9) диаграммы размещения (развертывания).

Виды диаграмм

- ***Диаграмма классов*** показывает набор классов, интерфейсов, сотрудничеств и их отношений. При моделировании объектно-ориентированных систем диаграммы классов используются наиболее часто. Диаграммы классов обеспечивают статическое проектное представление системы. Диаграммы классов, включающие активные классы,
- ***Диаграмма объектов*** обеспечивают статическое представление процессов системы, показывает набор объектов и их отношения. Диаграмма объектов представляет статический «моментальный снимок» с экземпляров предметов, которые находятся в диаграммах классов. Как и диаграммы классов, эти диаграммы обеспечивают статическое проектное представление или статическое представление процессов системы (но с точки зрения реальных или фототипичных случаев).

Виды диаграмм

- ***Диаграмма Use Case*** **(диаграмма прецедентов)** показывает набор элементов Use Case, актеров и их отношений. С помощью диаграмм Use Case для системы создается статическое представление Use Case. Эти диаграммы особенно важны при организации и моделировании поведения системы, задании требований заказчика к системе.
- ***Диаграмма взаимодействия*** показывает взаимодействие, включающее набор объектов и их отношений, а также пересылаемые между объектами сообщения. Диаграммы взаимодействия обеспечивают динамическое представление системы. **Диаграммы последовательности и диаграммы сотрудничества** — это разновидности диаграмм взаимодействия.
- ***Диаграмма последовательности*** — это диаграмма взаимодействия, которая выделяет упорядочение

Виды диаграмм

- ***Диаграмма сотрудничества*** (диаграмма кооперации) — это диаграмма взаимодействия, которая выделяет структурную организацию объектов, посылающих и принимающих сообщения. Диаграммы последовательности и диаграммы сотрудничества изоморфны, что означает, что одну диаграмму можно трансформировать в другую диаграмму.
- ***Диаграмма схем состояний*** показывает конечный автомат, представляет состояния, переходы, события и действия. Диаграммы схем состояний обеспечивают динамическое представление системы. Они особенно важны при моделировании поведения интерфейса, класса или сотрудничества.
- ***Диаграмма деятельности*** — специальная разновидность диаграммы схем состояний, которая показывает поток от действия к действию внутри системы. Диаграммы деятельности обеспечивают динамическое представление системы. Они особенно важны при моделировании функциональности системы и выделяют поток управления между объектами.

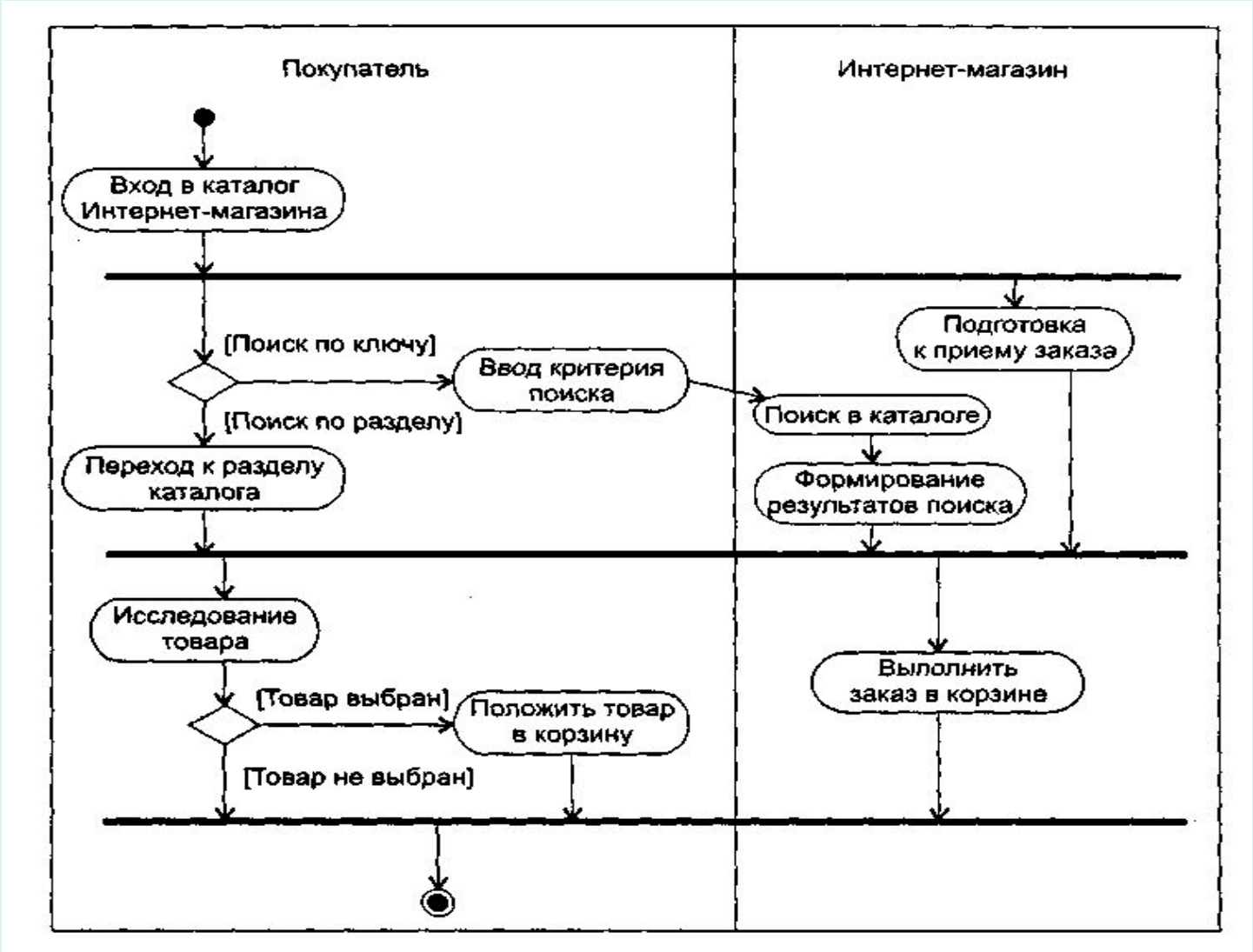
Виды диаграмм

- ***Компонентная диаграмма*** показывает организацию набора компонентов и зависимости между компонентами. Компонентные диаграммы обеспечивают статическое представление реализации системы. Они связаны с диаграммами классов в том смысле, что в компонент обычно отображается один или несколько классов, интерфейсов или кооперацией.
- ***Диаграмма размещения*** (диаграмма развертывания) показывает конфигурацию обрабатывающих узлов периода выполнения, а также компоненты, живущие в них. Диаграммы размещения обеспечивают статическое представление размещения системы. Они связаны с компонентными диаграммами в том смысле, что узел обычно включает один или несколько компонентов.

Диаграммы деятельности

- Диаграмма деятельности представляет особую форму конечного автомата, в которой показываются процесс вычислений и потоки работ.
- Основной вершиной в диаграмме деятельности является состояние действия, которое изображается как прямоугольник с закругленными боковыми сторонами.

Диаграмма деятельности покупателя в Интернет-магазине

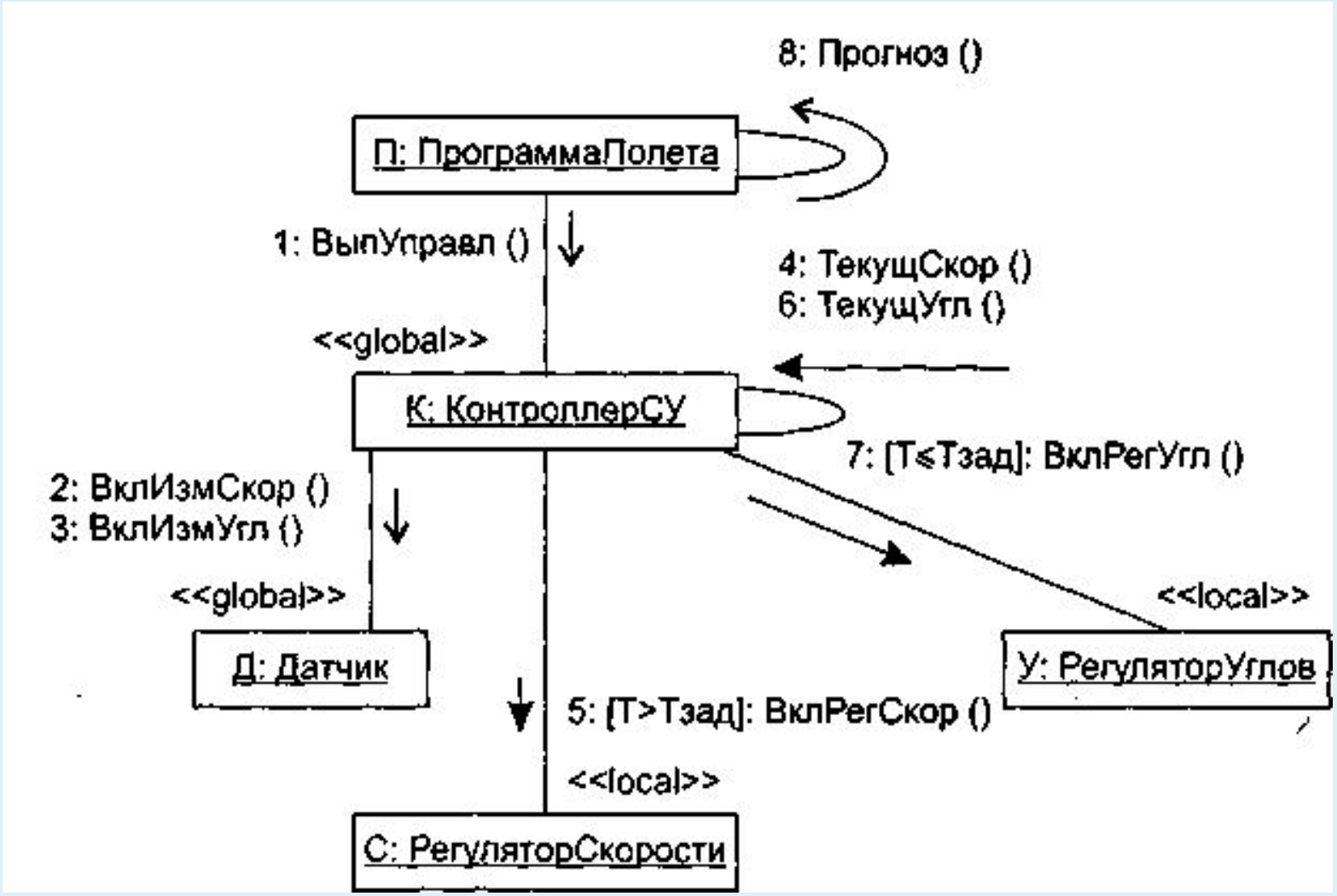


Диаграммы взаимодействия

- Диаграммы взаимодействия предназначены для моделирования динамических аспектов системы. Диаграмма взаимодействия показывает взаимодействие, включающее набор объектов и их отношений, а также пересылаемые между объектами сообщения. Существуют две разновидности диаграммы взаимодействия — диаграмма последовательности и диаграмма сотрудничества. Диаграмма последовательности — это диаграмма взаимодействия, которая выделяет упорядочение сообщений по времени. Диаграмма сотрудничества — это диаграмма взаимодействия, которая выделяет структурную организацию объектов, посылающих и принимающих сообщения. Элементами диаграмм взаимодействия являются участники взаимодействия — объекты, связи, сообщения.

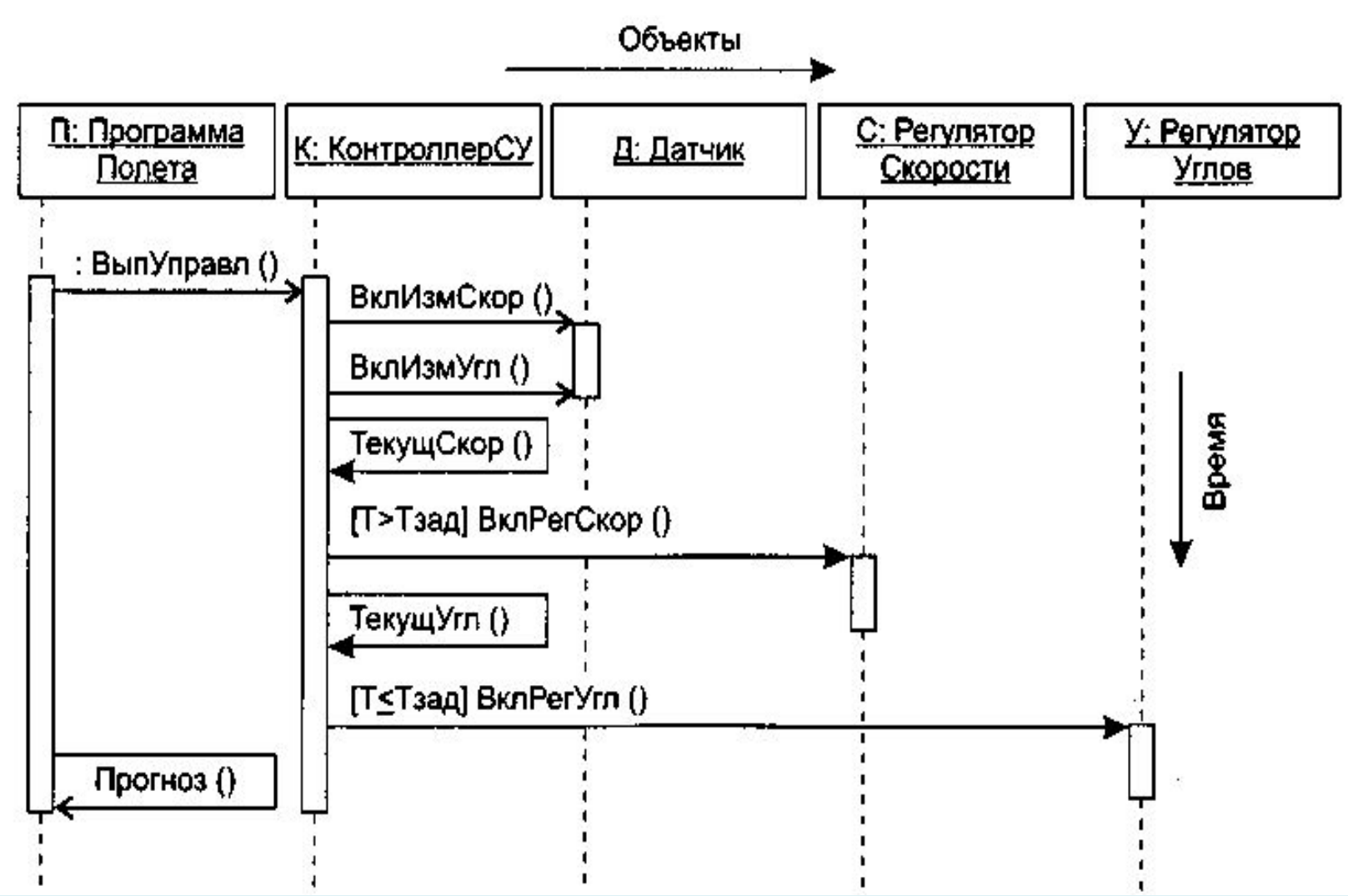
- Диаграммы сотрудничества отображают взаимодействие объектов в процессе функционирования системы. Такие диаграммы моделируют сценарии поведения системы. В русской литературе диаграммы сотрудничества часто называют диаграммами кооперации.

Диаграмма сотрудничества



- Диаграмма последовательности — вторая разновидность диаграмм взаимодействия. Отражая сценарий поведения в системе, эта диаграмма обеспечивает более наглядное представление порядка передачи сообщений. Правда, она не позволяет показать такие детали, которые видны на диаграмме сотрудничества (структурные характеристики объектов и связей).
- Графически диаграмма последовательности — разновидность таблицы, которая показывает объекты, размещенные вдоль оси *X,* и сообщения, упорядоченные по времени вдоль оси *Y.*

диаграмма последовательности

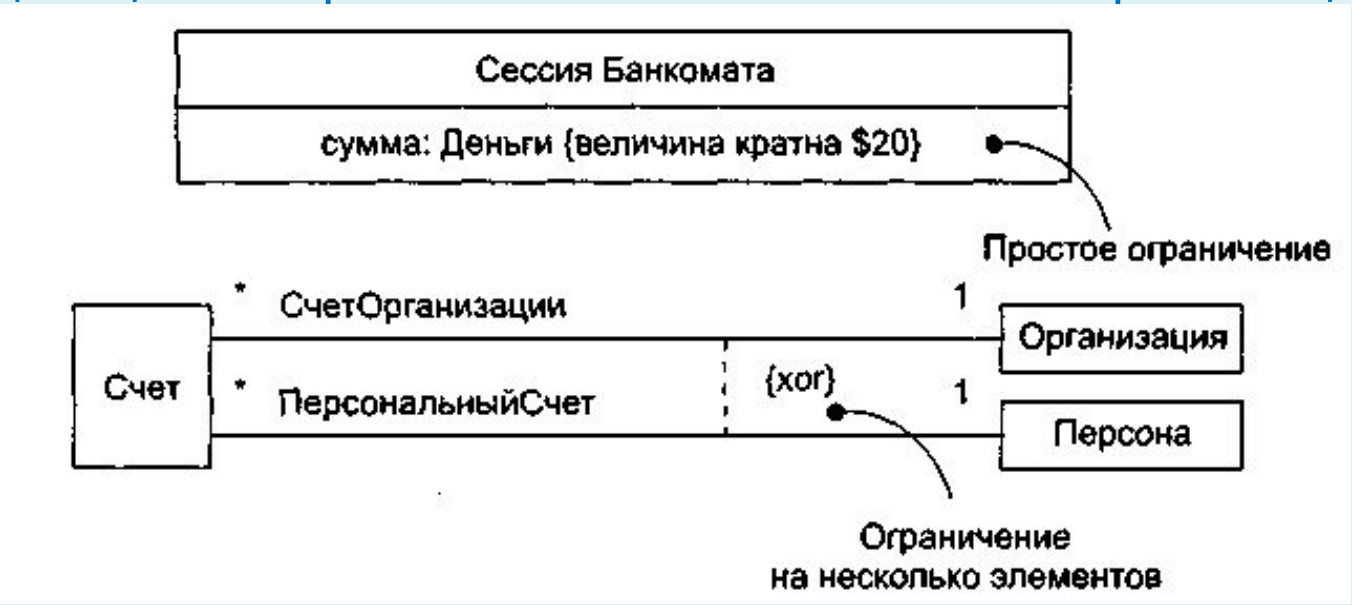


Механизмы расширения в UML

- UML — развитый язык, имеющий большие возможности, но даже он не может отразить все нюансы, которые могут возникнуть при создании различных моделей. Поэтому UML создавался как открытый язык, допускающий контролируемые расширения.
- Механизмами расширения в UML являются:
- ограничения;
- теговые величины;
- стереотипы.

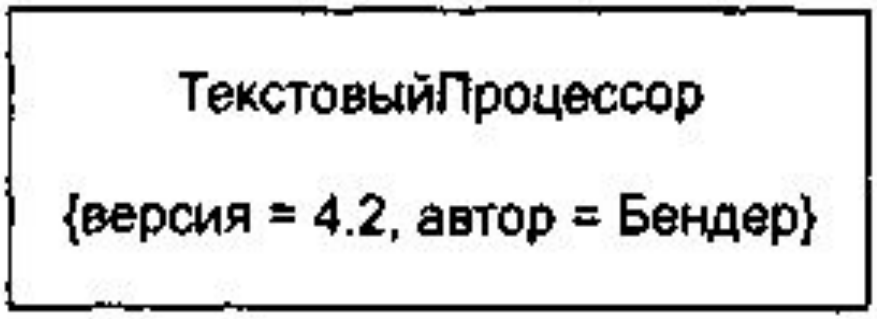
Ограничение (constraint)

- ***Ограничение*** (constraint) расширяет семантику строительного UML-блока, позволяя добавить новые правила или модифицировать существующие. Ограничение показывают как текстовую строку, заключенную в фигурные скобки {}. Например, на рис. введено простое ограничение на свойство *сумма* класса *Сессия Банкомата* — его значение должно быть кратно 20. Кроме того, здесь показано ограничение на два элемента (две ассоциации), оно располагается возле пунктирной линии, соединяющей элементы, и имеет следующий смысл — владельцем конкретного счета не может быть и организация, и пер



Теговая величина (tagged value)

- ***Теговая величина*** (tagged value) расширяет характеристики строительного UML-блока, позволяя создать новую информацию в спецификации конкретного элемента. Теговую величину показывают как строку в фигурных скобках {}. Строка имеет вид
- имя теговой величины = значение.
- Иногда (в случае предопределенных тегов) указывается только имя теговой величины.
- Отметим, что при работе с продуктом, имеющим много реализаций, полезно отслеживать версию и автора определенных блоков. Версия и автор не принадлежат к основным понятиям UML. Они могут быть добавлены к любому строительному блоку (например, к классу) введением в блок новых теговых величин. Например, на рис. 10.18 класс ТекстовыйПроцессо версии и автора.

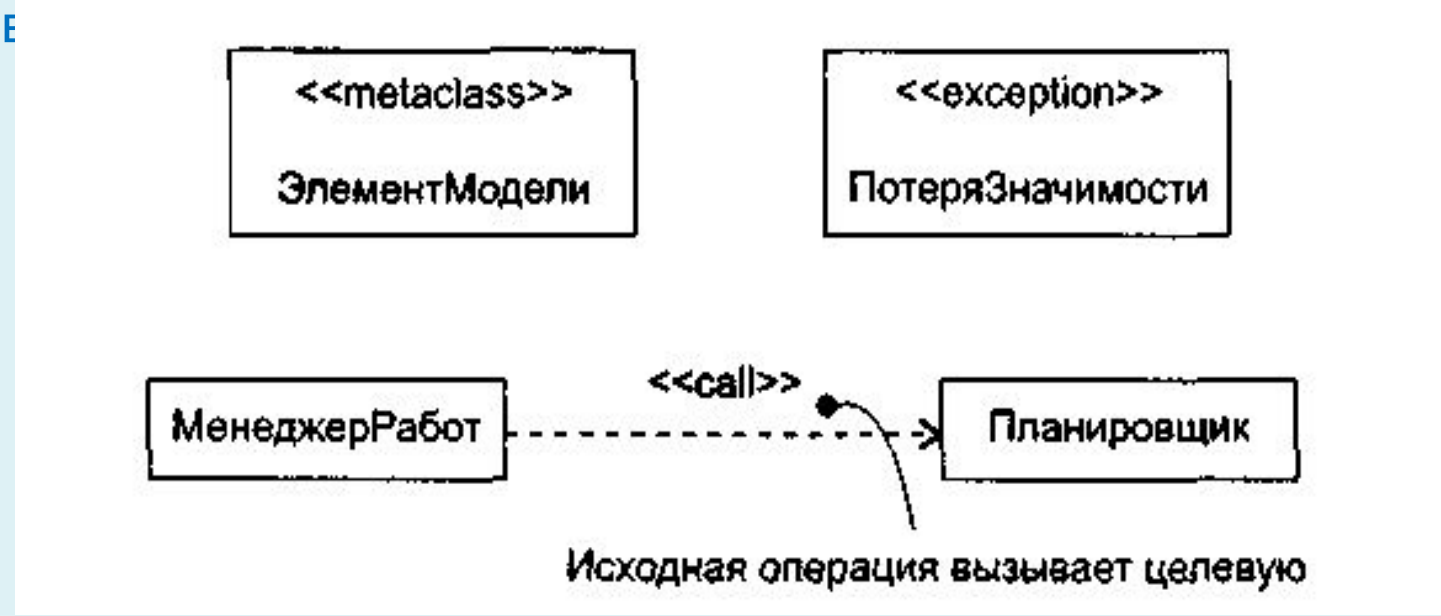


Стереотип

- ***Стереотип*** (stereotype) расширяет словарь языка, позволяет создавать новые виды строительных блоков, производные от существующих и учитывающие специфику новой проблемы.
- Элемент со стереотипом является вариацией существующего элемента, имеющей такую же форму, но отличающуюся по сути. У него могут быть дополнительные ограничения и теговые величины, а также другое визуальное представление. Он иначе обрабатывается при генерации программного кода. Отображают стереотип как имя, указываемое в двойных угловых скобках (или в угловых кавычках).

Стереотип

- Примеры элементов со стереотипами приведены на рис. 10.19. Стереотип «exception» говорит о том, что класс ПотеряЗначимости теперь рассматривается как специальный класс, которому, положим, разрешается только генерация и обработка сигналов исключений. Особые возможности метакласса получил класс ЭлементМодели. Кроме того, здесь показано применение стереотипа «call» к отношению зав



механизмы расширения

- Таким образом, механизмы расширения позволяют адаптировать UML под нужды конкретных проектов и под новые программные технологии. Возможно добавление новых строительных блоков, модификация спецификаций существующих блоков и даже изменение их семантики. Конечно, очень важно обеспечить контролируемое введение расширений.

Контрольные вопросы

- Сколько поколений языков визуального моделирования вы знаете?
- Назовите численность языков визуального моделирования 2-го поколения.
- Какая необходимость привела к созданию языка визуального моделирования третьего поколения?
- Поясните назначение UML.
- Какие строительные блоки образуют словарь UML? Охарактеризуйте их.
- Какие разновидности предметов UML вы знаете? Их назначение?
- Перечислите известные вам разновидности структурных предметов UML.
- Перечислите известные вам разновидности предметов поведения UML.
- Перечислите известные вам группирующие предметы UML.
- Перечислите известные вам поясняющие предметы UML.
- Какие разновидности отношений предусмотрены в UML? Охарактеризуйте каждое отношение.
- Дайте характеристику диаграммы классов.

Контрольные вопросы

- Дайте характеристику диаграммы объектов.
- Охарактеризуйте диаграмму Use Case.
- Охарактеризуйте диаграммы взаимодействия.
- Дайте характеристику диаграммы последовательности.
- Дайте характеристику диаграммы сотрудничества.