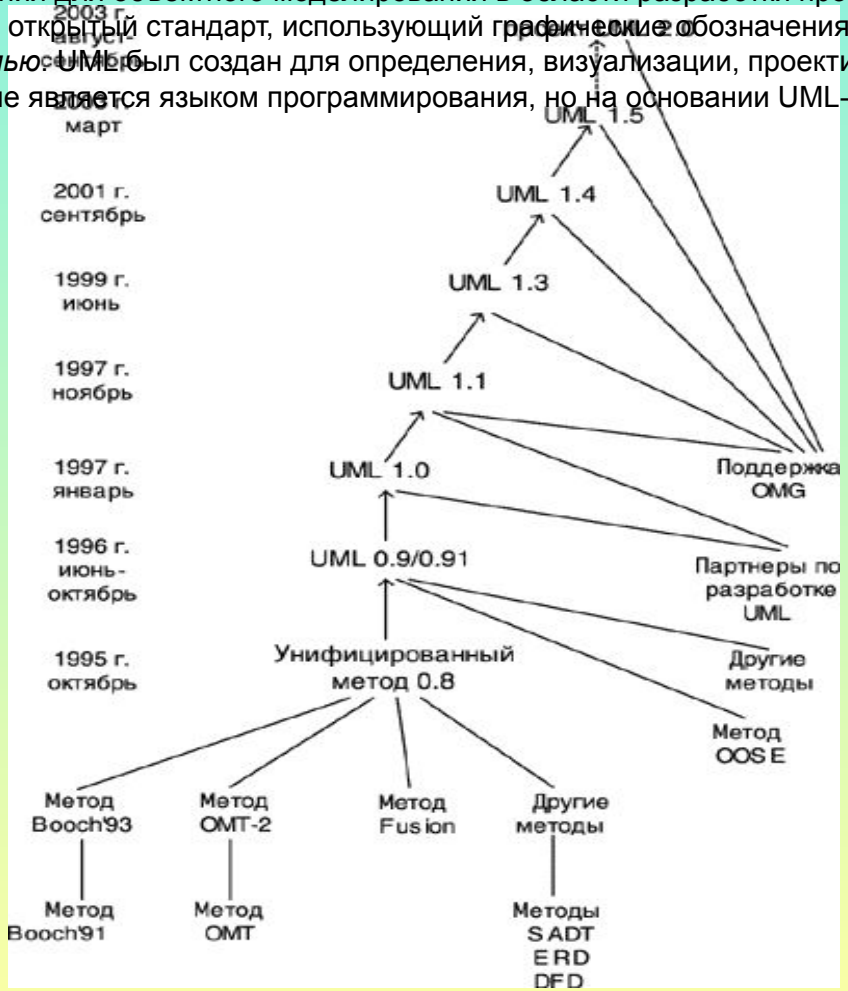
1. Характеристика мови UML. Канонічні діаграми.

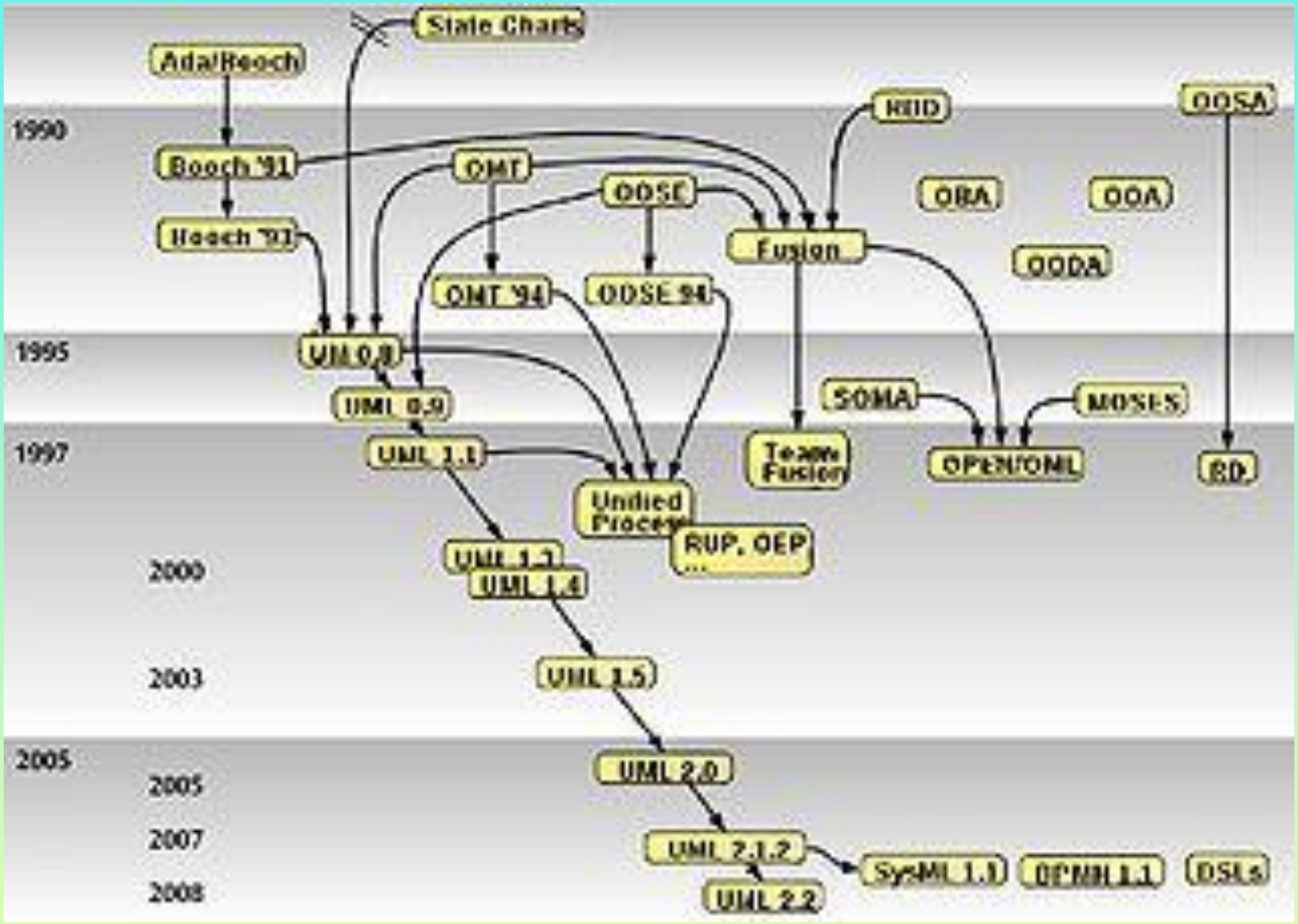
UML (англ. *Unified Modeling Language* — унифицированный язык моделирования) — язык — унифицированный язык моделирования) — язык графического — унифицированный язык моделирования) — язык графического описания для объектного моделирования — унифицированный язык моделирования) — язык графического описания для объектного моделирования в области разработки программного обеспечения — унифицированный язык моделирования) — язык графического описания для объектного моделирования в области разработки программного обеспечения. UML является языком широкого профиля, это — открытый стандарт — унифицированный язык моделирования) — язык графического описания для объектного моделирования в области разработки программного обеспечения. UML является языком широкого профиля, это — открытый стандарт, использующий графические обозначения для создания абстрактной модели — унифицированный язык моделирования) — язык графического описания для объектного моделирования в области разработки программного обеспечения. UML является языком широкого профиля, это — открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой *UML-моделью*. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML-моделей возможна генерация кода.

История развития языка UML



Поддержка разработки языка **UML** осуществляется консорциумом **OMG** (Object Management Group), который был образован в 1989 году с целью разработки предложений по стандартизации ***объектных*** и ***компонентных*** технологий.

Последняя версия UML 2.5 опубликована в июне 2015 года. UML 2.5 принят в качестве международного стандарта ISO/IEC 19505-1, 19505-2.


State Charts
Ada/Booch
1990
Booch '91
OMT
ODSE
HRID
OOSA
ORA
OOA
Booch '93
Fusion
OODA
OMT '94
ODSE 94
1995
UML 0.8
UML 0.9
SOMA
MOSES
1997
UML 1.1
Unified Process
RUP, OEP
...
OPENTOOL
RD
2000
UML 1.4
2003
UML 1.5
2005
UML 2.0
2005
2007
UML 2.1.2
SysML 1.1
DSLs
2008
UML 2.2

Версия	Дата принятия
1.1	*ноябрь 1997*[1]
1.3	*март 2000*[2]
1.4	*сентябрь 2001*[3]
1.4.2.	*июль 2004*[2]
1.5	*март 2003*[4]
2.0	*июль 2005*[5]
2.1	*формально не была принята*[2]
2.1.1	*август 2007*[6]
2.1.2	*ноябрь 2007*[7]
2.2	*февраль 2009*[8]
2.3	*май 2010*[9]
2.5	*июнь 2015*

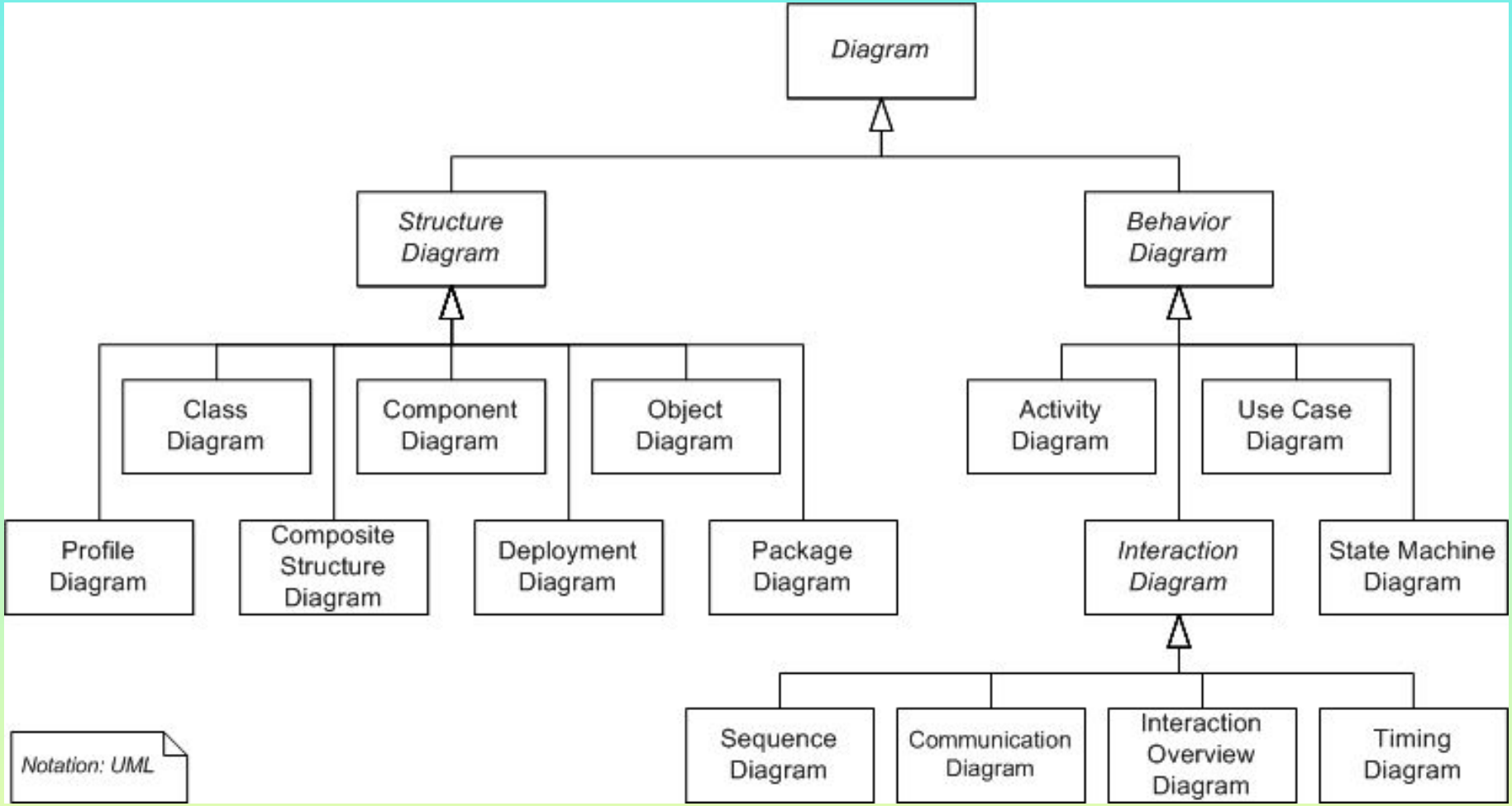


Diagram
Structure Diagram
Behavior Diagram
Class Diagram
Component Diagram
Object Diagram
Activity Diagram
Use Case Diagram
Profile Diagram
Composite Structure Diagram
Deployment Diagram
Package Diagram
Interaction Diagram
State Machine Diagram
Sequence Diagram
Communication Diagram
Interaction Overview Diagram
Timing Diagram
Notation: UML

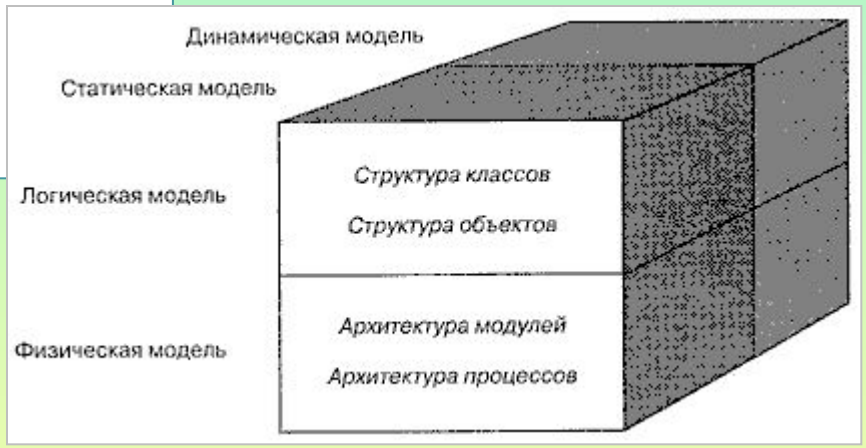


Рис.1. Общая схема взаимосвязей моделей и представлений сложной системы в процессе *объектно-ориентированного анализа* и *проектирования*

Канонические диаграммы языка UML

Диаграмма *(diagram)* — графическое представление совокупности элементов *модели* в форме связного графа, вершинам и ребрам (дугам) которого приписывается определенная семантика.

Нотация ***канонических диаграмм*** - основное средство разработки *моделей* на языке UML.

Канонические диаграммы:

- вариантов использования (use case diagram)
- классов (class diagram)
- кооперации (collaboration diagram, communication)
- последовательности (sequence diagram)
- состояний (statechart diagram)
- деятельности (activity diagram)
- компонентов (component diagram)
- развертывания (deployment diagram)

Канонические диаграммы языка UML (2)

Диаграмма **вариантов использования** представляет собой наиболее общую **концептуальную** *модель* сложной системы, которая является исходной для построения всех остальных *диаграмм*.

Диаграмма **классов** – это **логическая** *модель*, отражающая **статические** аспекты структурного построения сложной системы.

Диаграммы **кооперации** и **последовательностей** представляют собой разновидности **логической** *модели*, которые отражают **динамические** аспекты функционирования сложной системы.

Диаграммы **состояний** и **деятельности** предназначены для моделирования **поведения** системы. Аналогом диаграмм **деятельности** являются схемы алгоритмов по ГОСТ 19.701-90.

Диаграммы **компонентов** и **развертывания** служат для представления физических компонентов сложной системы и поэтому относятся к ее **физической** *модели*.

2. Система позначень уніфікованої мови моделювання

Геометрические фигуры (примитивы) на плоскости, играющие роль ***вершин графов*** соответствующих *диаграмм*. При этом сами геометрические фигуры выступают в роли графических примитивов языка UML, а форма этих фигур (прямоугольник, эллипс) должна строго соответствовать изображению отдельных элементов языка UML (класс, вариант использования, состояние, деятельность). Графические примитивы языка UML имеют фиксированную семантику, переопределять которую пользователям не допускается. Графические примитивы должны иметь собственные имена, а, возможно, и другой текст, который содержится внутри границ соответствующих геометрических фигур или, как исключение, вблизи этих фигур.

Графические взаимосвязи, которые представляются различными **линиями** на плоскости. Взаимосвязи в языке UML обобщают понятие **дуг** и **ребер** из теории графов, но имеют менее формальный характер и более развитую семантику.

Специальные графические символы, изображаемые вблизи от тех или иных визуальных элементов *диаграмм* и имеющие характер дополнительной спецификации.

Расширения

***Стереотип* (stereotype)** — элемент *модели*, который расширяет семантику метамодели. Некоторые *стереотипы* предопределены в языке UML, другие могут быть указаны разработчиком. На *диаграммах* изображаются в форме **текста, заключенного в угловые кавычки (<<interface>>)**.

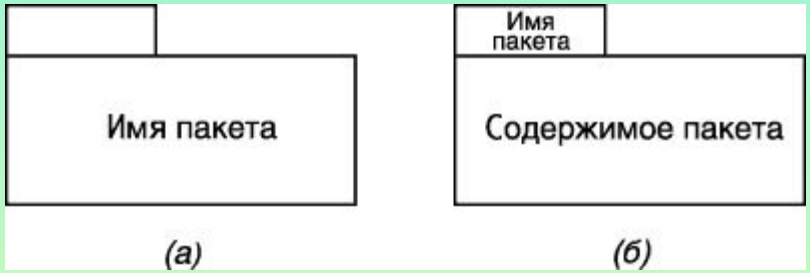
***Помеченное значение* (tagged value)** — **явное определение свойства** как пары. Имя называют **тегом** (tag). *Ф*ормат записи: **тег = значение**. Теги встречаются в нотации языка UML, но их определение не является строгим, поэтому теги могут быть указаны самим разработчиком.

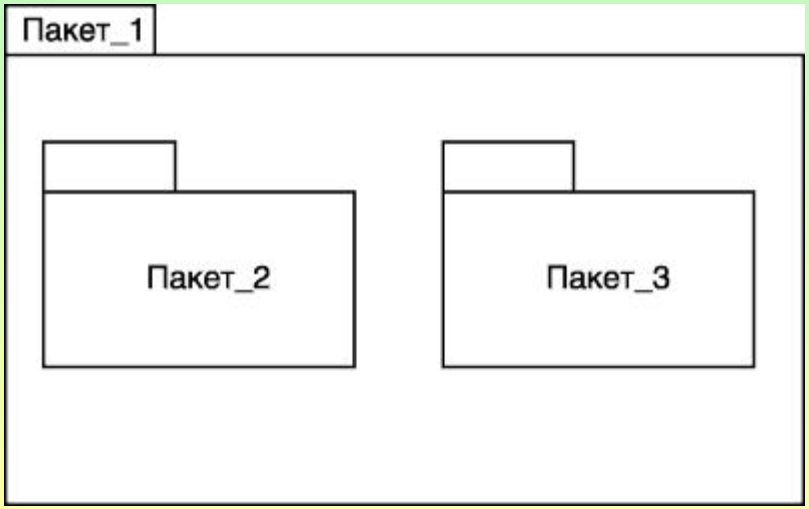
***Ограничение* (constraint)** — некоторое **логическое условие**, ограничивающее семантику выбранного элемента *модели*. Как правило, все *ограничения* специфицируются разработчиком. *Ограничения* на *диаграммах* изображаются в форме **строки текста, заключенного в фигурные скобки**.

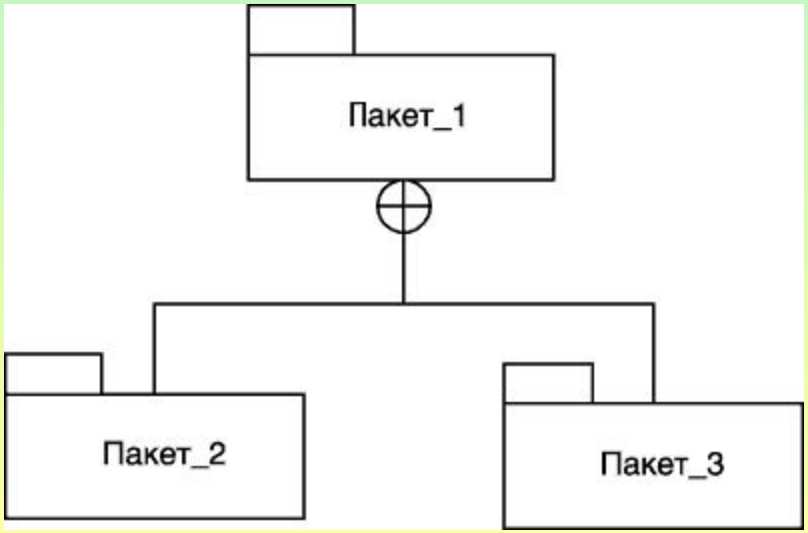
Пакеты в языке UML

Пакет (package) — общецелевой механизм для организации различных элементов *модели* в множество, реализующий системный принцип декомпозиции *модели* сложной системы и допускающий вложенность *пакетов* друг в друга. ***Пакет***, служит для **группировки** *моделей и элементов модели*.

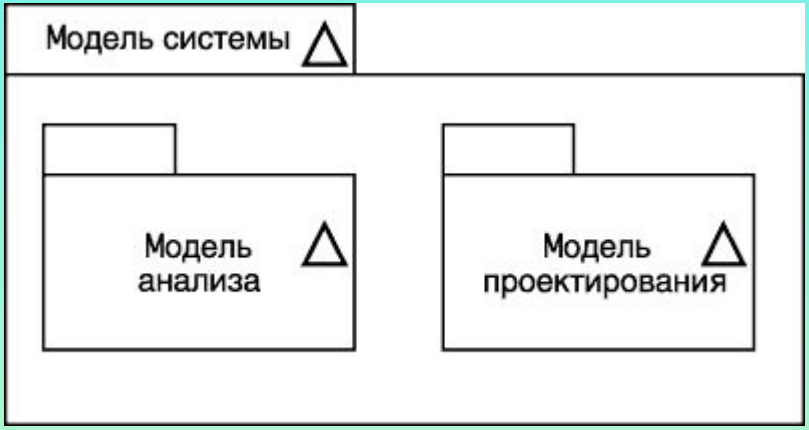
(*Пакет* может включать в себя несколько различных *моделей* одной и той же системы)







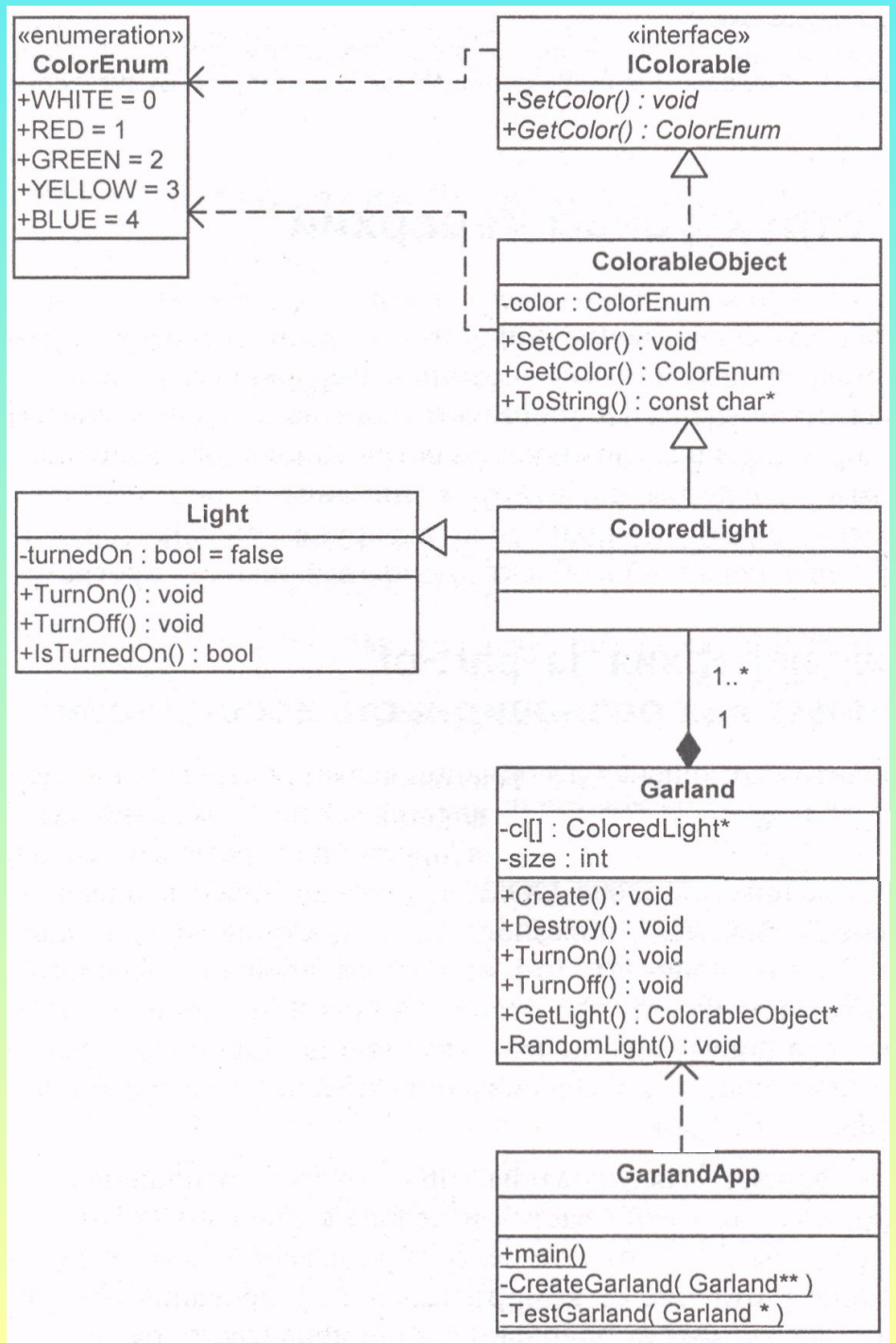
Пакеты в языке UML (2)



Модель является *подклассом пакета* и представляет собой абстракцию физической системы, которая предназначена для вполне определенной цели.

Графическое изображение подсистемы в языке UML

Ієрархія класів проекту "Гірлянда"



3. Діаграми варіантів застосування UML

Диаграмма ***вариантов использования*** - это исходное концептуальное представление или *концептуальная модель* системы в процессе ее проектирования и разработки.

Цели создания диаграммы *вариантов использования* :

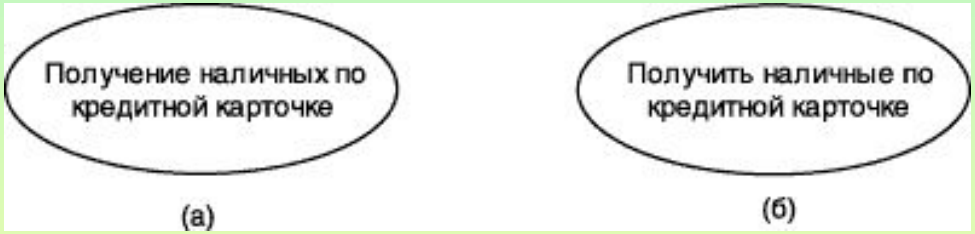
- **Определить общие границы и контекст моделируемой предметной области.**
- **Сформулировать общие требования к поведению проектируемой системы.**
- **Разработать исходную *концептуальную* модель системы для ее последующей детализации в форме *логических* и *физических* моделей.**
- **! ! ! Подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.**

Диаграмма вариантов использования (2)

Диаграмма ***вариантов использования*** (use case diagram) — диаграмма, на которой изображаются ***отношения*** между ***актерами*** и ***вариантами использования***

Актером или действующим лицом называется любой объект, субъект или система, взаимодействующая с моделируемой системой **извне.** (*Это может быть человек, техническое устройство, программа или любая другая система, которая служит источником воздействия на моделируемую систему так, как определит разработчик*)

Вариант использования служит для описания **сервисов**, которые система предоставляет *актеру*.

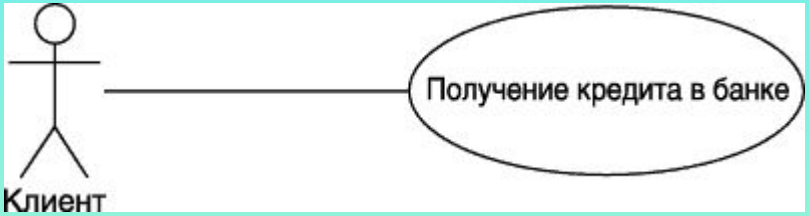


Графическое обозначение варианта использования

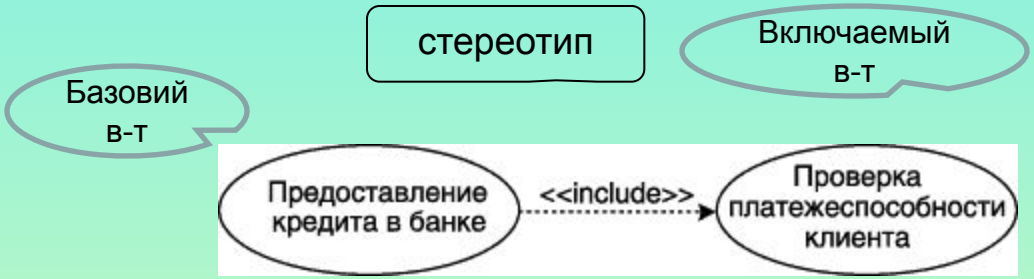
Графическое обозначение актера

Отношения на диаграмме вариантов использования

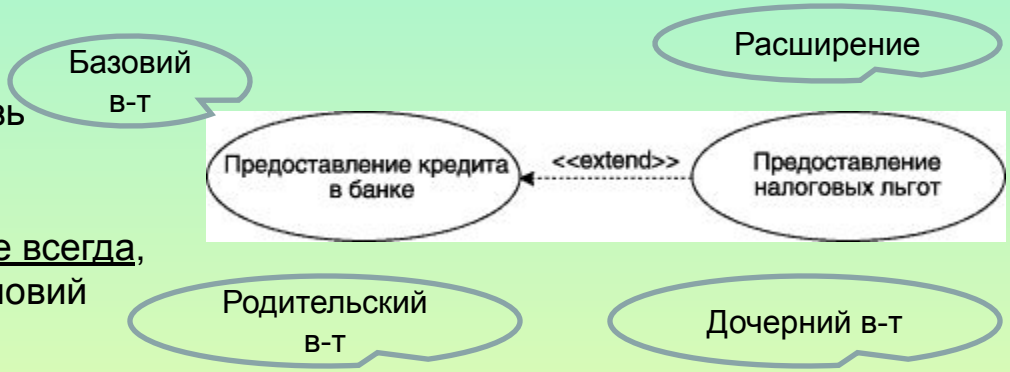
Ассоциация - обозначение специфической роли *актера* при его взаимодействии с отдельным *вариантом использования*



Включение (***include***) указывает на то, что заданное поведение для одного варианта использования <u>всегда</u> включается в качестве составного фрагмента в последовательность поведения другого варианта использования.



Расширения (extend) определяет взаимосвязь *базового варианта использования* с другим *вариантом использования*, функциональное поведение которого задействуется базовым <u>не всегда</u>, а только при выполнении дополнительных условий



Отношение ***обобщения*** применяется, когда необходимо отметить, что **дочерние** варианты использования обладают всеми особенностями поведения **родительских** вариантов.

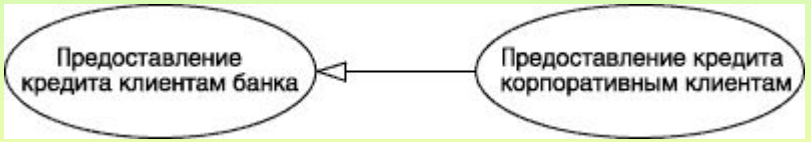
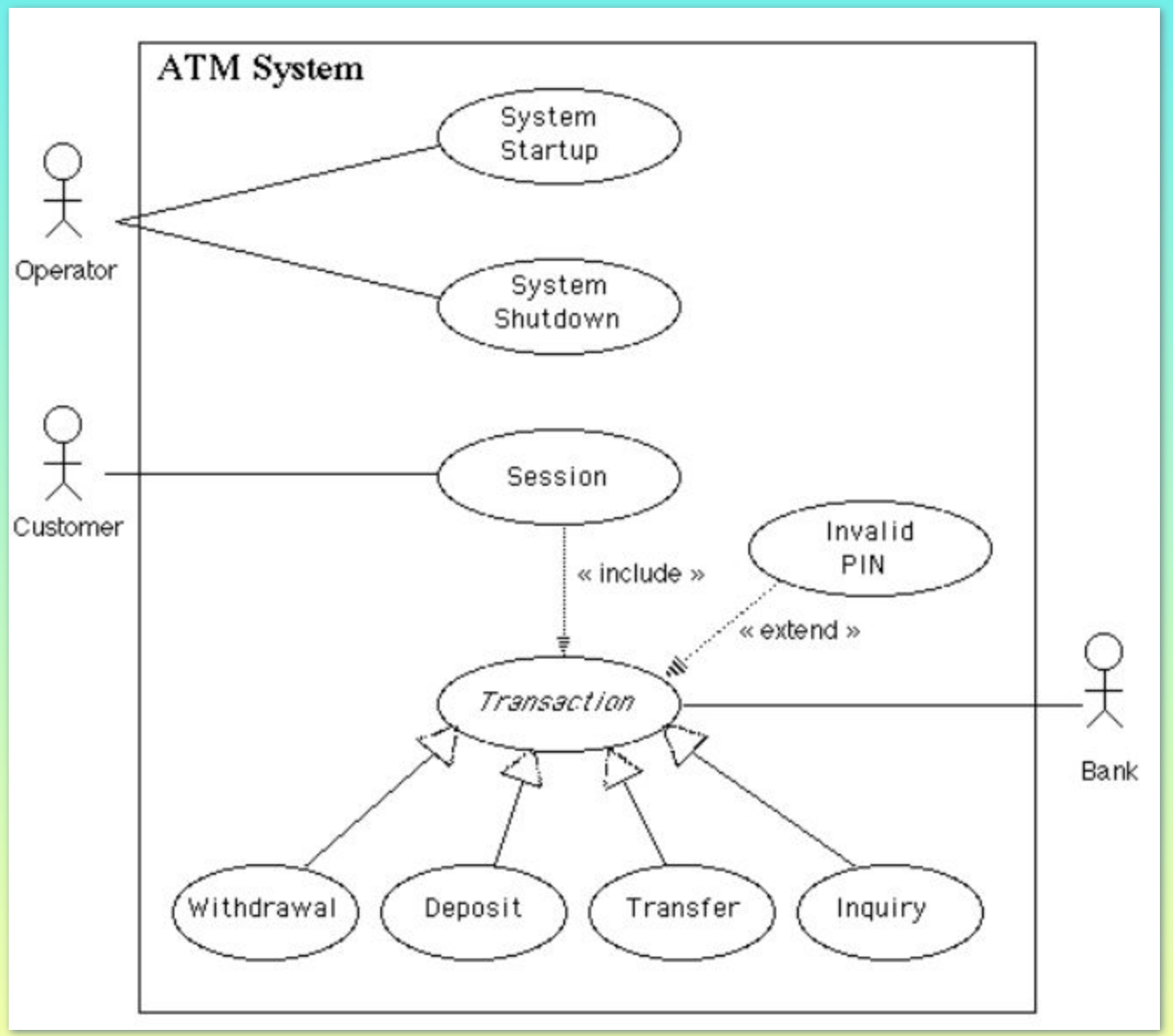


Диаграмма вариантов использования для системы продажи товаров по каталогу


ATM System
System Startup
System Shutdown
Operator
Customer
Session
Invalid PIN
« include »
« extend »
Transaction
Bank
Withdrawal
Deposit
Transfer
Inquiry

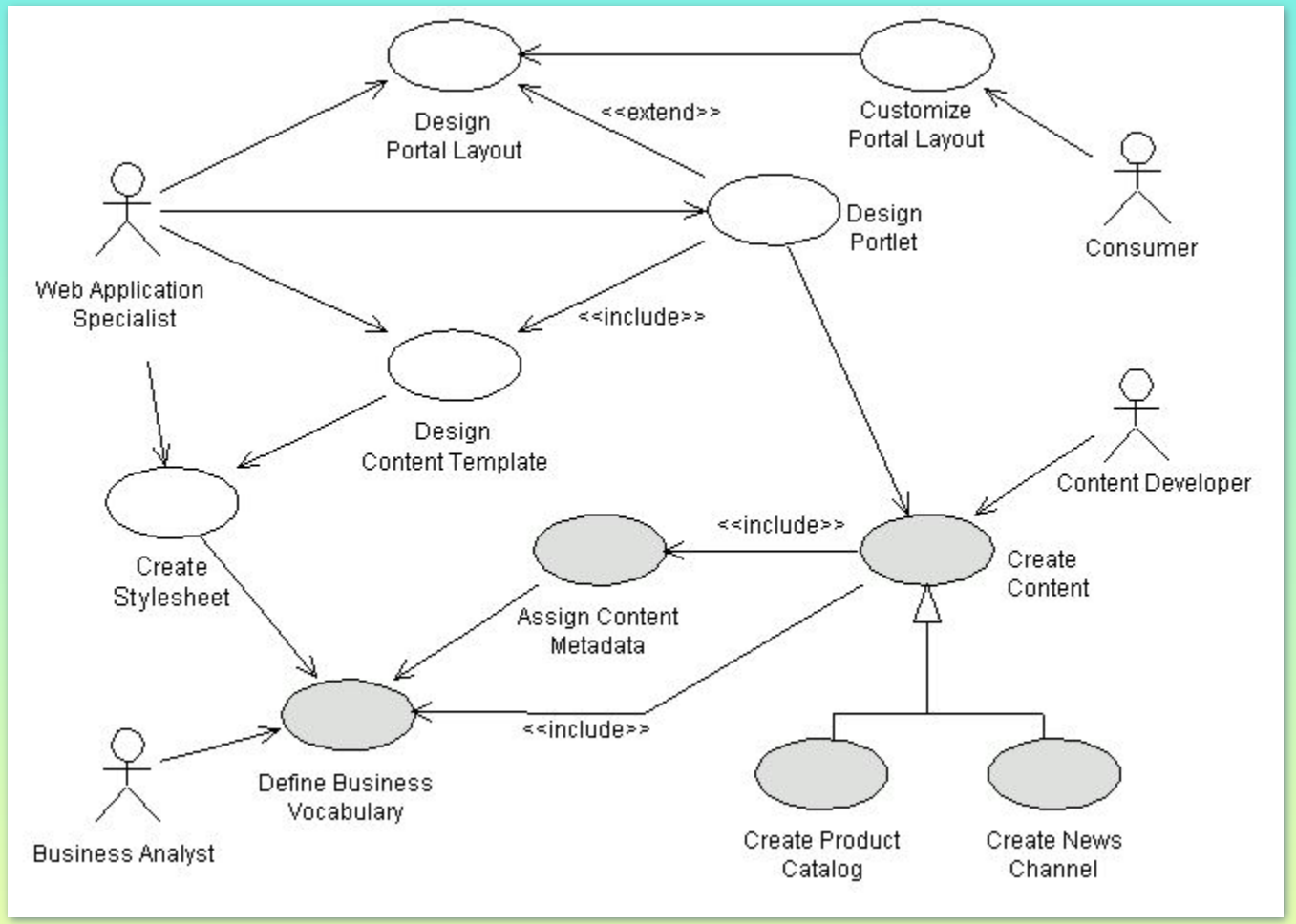
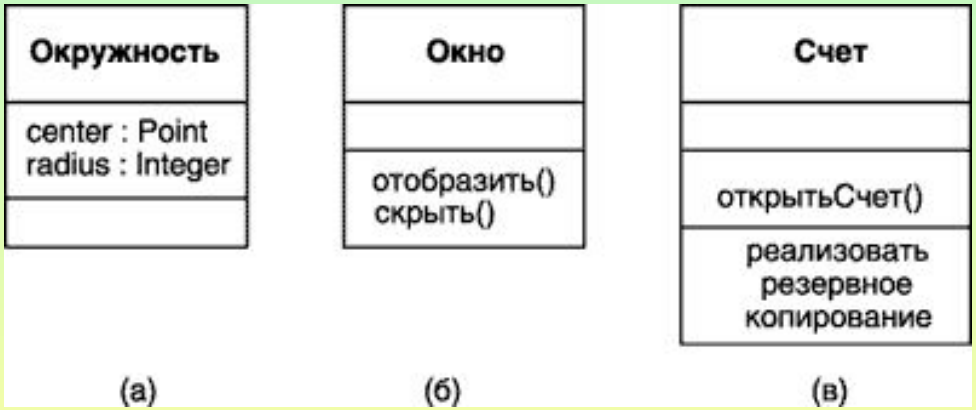

Design
Portal Layout
<<extend>>
Customize
Portal Layout
Design
Portlet
Consumer
Web Application
Specialist
<<include>>
Design
Content Template
Content Developer
Create
Stylesheet
<<include>>
Create
Content
Assign Content
Metadata
<<include>>
Define Business
Vocabulary
Business Analyst
Create Product
Catalog
Create News
Channel

Диаграмма классов

Диаграмма *классов* предназначена для представления статической структуры модели системы **в терминологии *классов* объектно-ориентированного программирования**.

Варианты графического изображения класса на диаграмме классов



Примеры графического изображения конкретных классов

Класс может иметь или не иметь экземпляров или объектов. В зависимости от этого в языке UML различают *конкретные* и *абстрактные классы*.

Конкретный класс (concrete class) — *класс*, на основе которого могут быть непосредственно созданы экземпляры или объекты.

Абстрактный класс (abstract class) — *класс*, который не имеет экземпляров или объектов.

Обозначения на диаграммах классов

Формат строки *имени класса*: **<Имя пакета>::<*Имя класса*>**

Формат записи *атрибута класса:* **<квантор видимости> <имя атрибута> [кратность] : <тип атрибута> = <исходное значение> {строка-свойство}.**

Формат записи *операции класса:*

<квантор видимости> <имя операции> (список параметров) : <выражение типа возвращаемого значения> {строка-свойство}

"+" – обозначает *атрибут* с областью видимости типа **общедоступный** (**public**). *Атрибут* с этой областью видимости доступен или виден из любого другого *класса* пакета, в котором определена диаграмма.

"#" – обозначает *атрибут* с областью видимости типа **защищенный** (**protected**). *Атрибут* с этой областью видимости недоступен или невиден для всех *классов*, за исключением подклассов данного *класса*.

"-" – обозначает *атрибут* с областью видимости типа **закрытый** (**private**). *Атрибут* с этой областью видимости недоступен или невиден для всех *классов* без исключения.

"~" - обозначает *атрибут* с областью видимости типа **пакетный** (**package**). *Атрибут* с этой областью видимости недоступен или невиден для всех *классов* за пределами пакета, в котором определен *класс*-владелец данного *атрибута*.

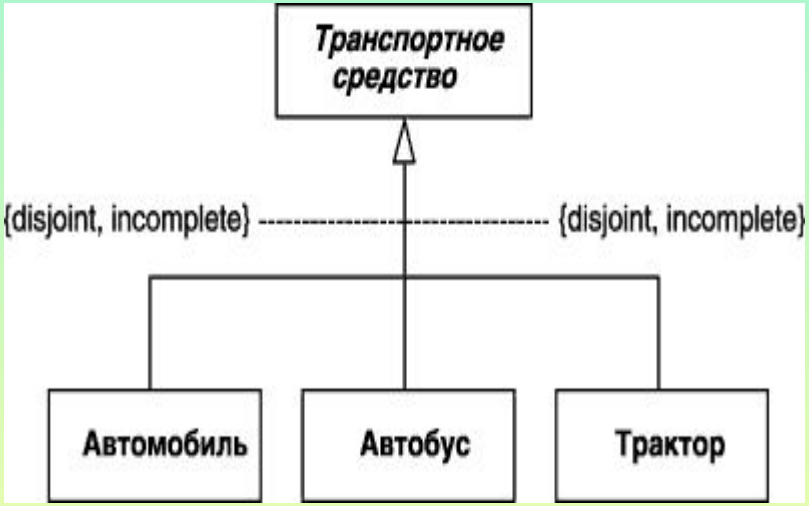
Формат записи параметров:

<направление параметра: *in, out, inout* > <имя параметра>: <выражение типа> = <значение параметра по умолчанию>

Диаграмма классов
Отношения классов

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

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



Ограничения

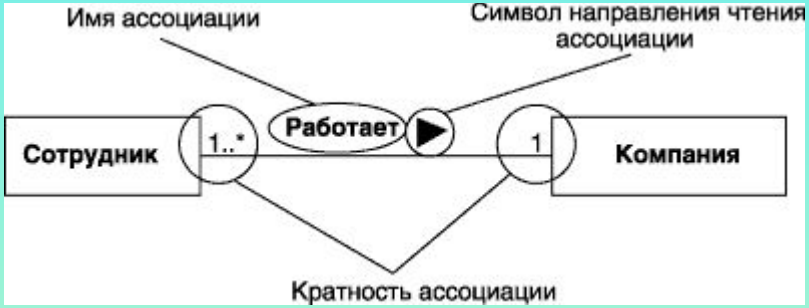
{**complete**} - в данном отношении *обобщения* специфицированы все классы-потомки, и других классов-потомков у данного класса-предка быть не может.

{**incomplete**} - означает, что на диаграмме указаны не все классы-потомки.

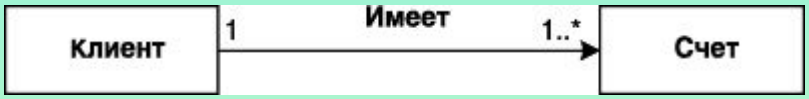
{**disjoint**} - означает, что классы-потомки не могут содержать объектов, одновременно являющихся экземплярами двух или более классов.

{**overlapping**} - предполагается, что отдельные экземпляры классов-потомков могут принадлежать одновременно нескольким классам.

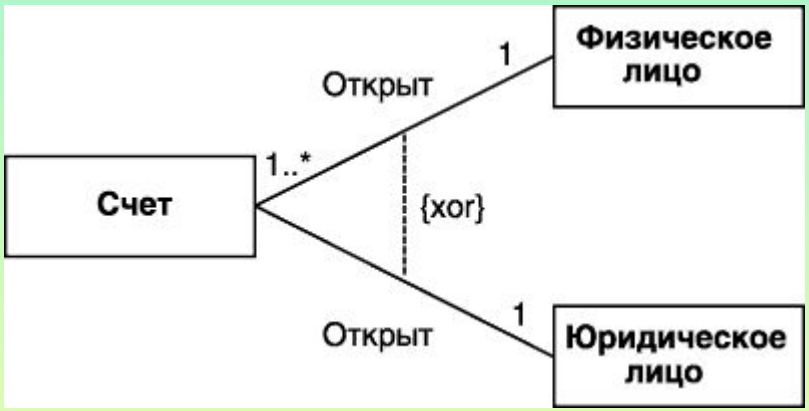
Диаграмма классов
Отношения классов



Графическое изображение ненаправленной бинарной ассоциации между классами



Графическое изображение направленной бинарной ассоциации между классами



Графическое изображение исключающей ассоциации между тремя классами

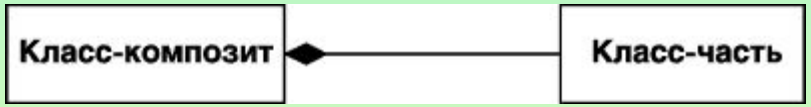
Графическое изображение тернарной ассоциации между тремя классами

Диаграмма классов
Отношения классов

Агрегация

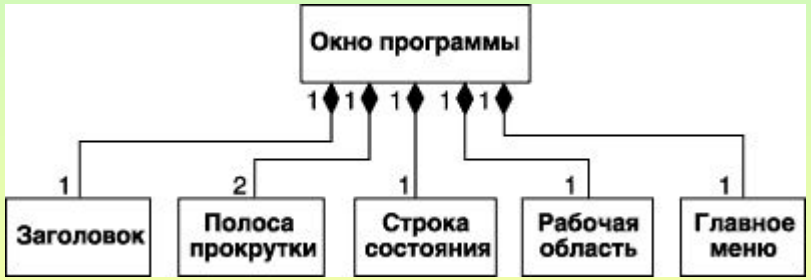
Агрегация (aggregation) - специальная форма **ассоциации**, которая служит для представления отношения типа "часть-целое" между агрегатом (целое) и его составной частью.

Отношение **агрегации** на примере системного блока ПК.

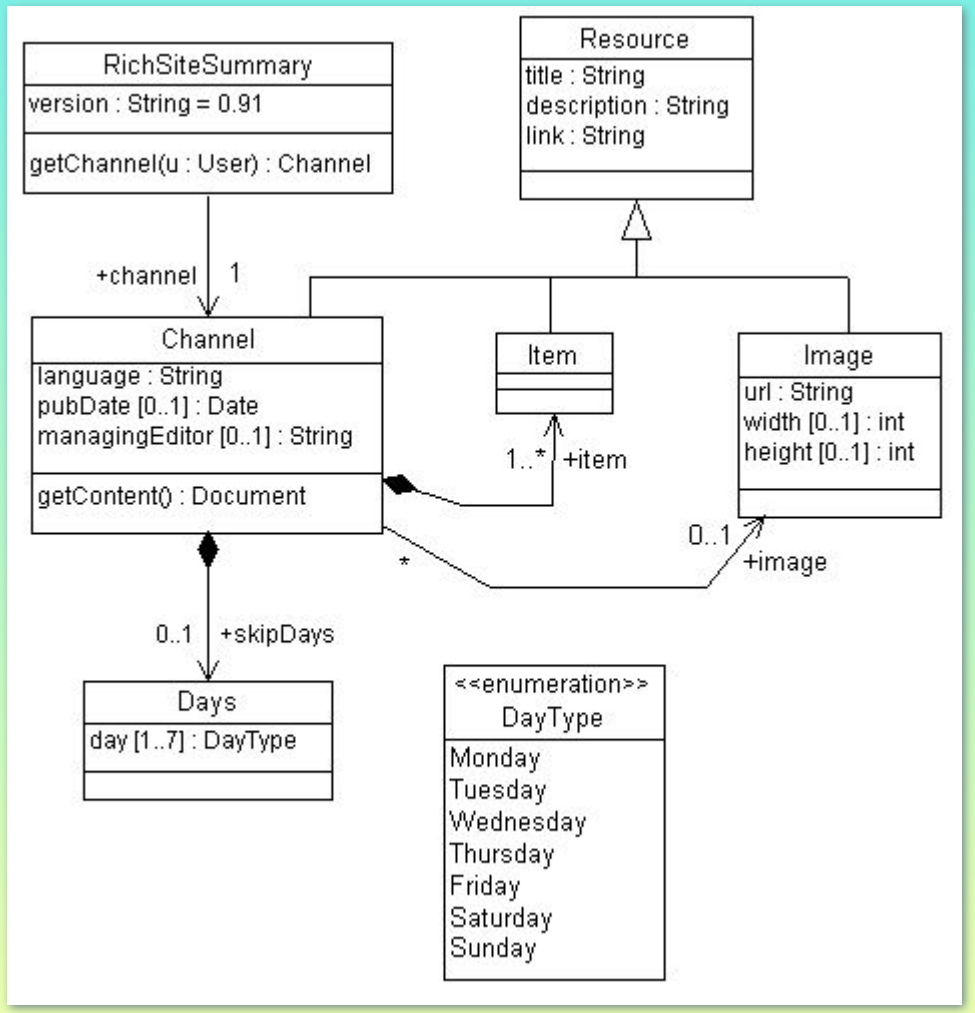


Композиция

Композиция (composition) - разновидность отношения **агрегации**, при которой составные части целого имеют такое же время жизни, что и само целое. Эти части уничтожаются вместе с уничтожением целого.



Отношение **композиции** на примере класса-композита Окно программы


RichSiteSummary
version : String = 0.91
getChannel(u : User) : Channel
Resource
title : String
description : String
link : String
+channel
1
Channel
language : String
pubDate [0..1] : Date
managingEditor [0..1] : String
getContent() : Document
Item
1..*
+item
Image
url : String
width [0..1] : int
height [0..1] : int
0..1
+image
*
0..1
+skipDays
Days
day [1..7] : DayType
<<enumeration>>
DayType
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday