

Объектно-ориентированные CASE-технологии Язык UML 2.5.

Владимир Юрьевич Романов,
Московский Государственный Университет им. М.В.Ломоносова
Факультет Вычислительной Математики и Кибернетики
vromanov@cs.msu.su,
romanov.rvy@yandex.ru
<http://master.cmc.msu.ru>

UML – Unified Modeling Language. Унифицированный язык моделирования

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- Стандарт на язык моделирования разработанный консорциумом фирм Object Management Group:

<http://www.omg.org>

- Стандартизация языка UML консорциумом OMG:

<http://www.omg.org/uml>

<http://www.uml.org/>

- Текущие версии стандарта доступные для свободного скачивания:

<http://www.omg.org/spec/>

UML – Unified Modeling Language.

Стандарты связанные с языком UML

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

MODELING AND METADATA SPECIFICATIONS

UML, MOF, CWM, XMI Specifications

SPECIFICATION	acronym	topical area / domain	Document #
Action Language for Foundational UML	ALF	modeling	ptc/2012-08-43
Common Terminology Services 2	cts2	modeling	formal/2012-11-01
Common Warehouse Metamodel	CWM	data warehousing, modeling	formal/2003-03-02
CWM Metadata Interchange Patterns	MIPS	data warehousing, modeling	formal/2004-03-25
Diagram Definition	DD	modeling	formal/12-07-01
Essence - Kernel and Language for Software Engineering Methods	Essence	modeling	ptc/2013-06-08
Interaction Flow Modeling Language	IFML	modeling	ptc/2013-03-08
Meta Object Facility Core	MOF	modeling	formal/2011-08-07
Model Driven Message Interoperability	MDMI	modeling	formal/2010-03-01
Model-level Testing and Debugging	MLTD	modeling	ptc/2007-05-14
MOF Model to Text Transformation Language	MOFM2T	modeling	formal/2008-01-16
MOF Query / View / Transformation	QVT	modeling	formal/2011-01-01
MOF Support for Semantic Structures	SMOF	modeling	formal/2013-04-02
MOF 2 Facility and Object Lifecycle	MOFFOL	modeling	formal/2010-03-04
MOF 2 Versioning and Development Lifecycle	MOFVD	modeling	formal/2007-05-01
Object Constraint Language	OCL	modeling	formal/2012-01-01
OMG Systems Modeling Language	SysML	modeling	formal/2012-06-01
Ontology Definition Metamodel	ODM	modeling	formal/2009-05-01
Reusable Asset Specification	RAS	modeling	formal/2005-11-02
Semantics of a Foundational Subset for Executable UML Models	FUML	modeling	formal/2011-02-01
Service oriented architecture Modeling Language	SoaML	modeling	formal/2012-05-01
Software Process Engineering Metamodel	SPEM	modeling	formal/2008-04-01
Unified Modeling Language	UML	modeling	formal/2011-08-05, formal/2011-08-06
UML Simplified (UML 2.5)	UML	modeling	ptc/2012-10-24
UML Human-Usable Textual Notation	HUTN	modeling	formal/2004-08-01
MOF 2 XMI Mapping	XMI	modeling	formal/2011-08-09

UML – Unified Modeling Language.

Инфраструктура и суперструктура языка UML

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- Текущая версия языка **UML**:
<http://www.omg.org/spec/UML/2.5/PDF/>
- **Object Constraint Language™ (OCL™)**
<http://www.omg.org/spec/OCL/2.4/PDF/>
- **Diagram Definition (DD)**
<http://www.omg.org/spec/DD/1.1>
- **XML Metadata Interchange (XMI)** формат для обмена моделями инструментами:
<http://www.omg.org/spec/XMI/2.5.1/>
- Спецификация **UML** в формате **XMI**
<http://www.omg.org/spec/UML/20161101/UML.xmi>

UML – Unified Modeling Language. Объектный язык ограничений (OCL)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

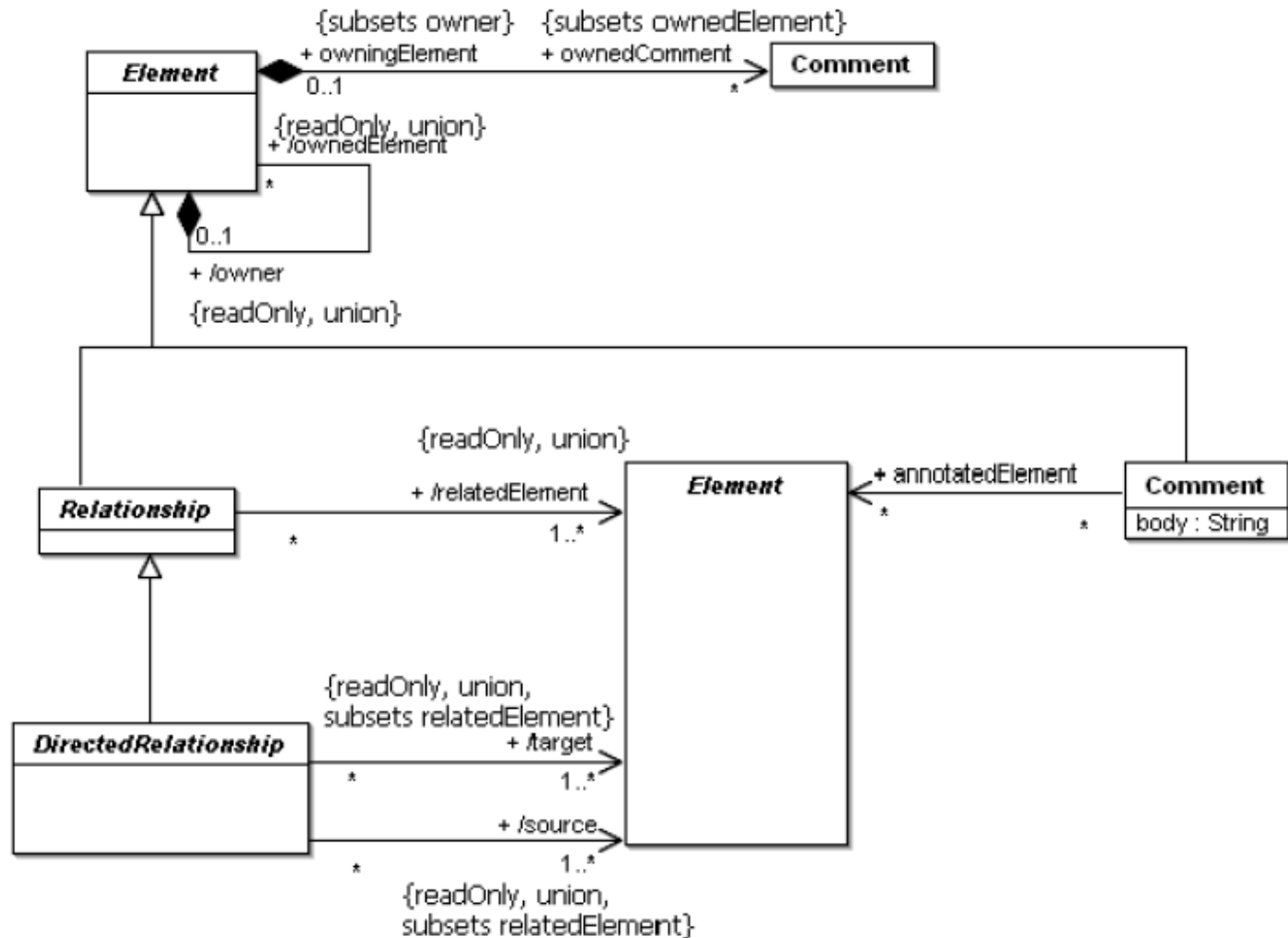
- **Object Constraint Language (OCL)**
<https://www.omg.org/spec/OCL/2.4/>
- Спецификация языка OCL на языке XML
<http://www.omg.org/spec/OCL/20090501/OCL.cmo>

UML – Unified Modeling Language.

Фрагмент спецификации метамодели UML

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



UML – Unified Modeling Language.

Фрагмент спецификации метамодели OCL

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- **if** memberEnd->size() > 2 **then** ownedEnd->*includesAll*(memberEnd)
- parents()->*select*(oclIsKindOf(Association)).oclAsType(Association)->*forAll*(p | p.memberEnd->size() = **self**.memberEnd->size())
- *Sequence*{1..**self**.memberEnd->size()}->
 forAll(i |
 self.general->*select*(oclIsKindOf(Association)).oclAsType(Association)
 ->
 forAll(ga |
 self.memberEnd->
 at(i).type.conformsTo(ga.memberEnd->at(i).type)))

UML – Unified Modeling Language. Meta Object Facility (MOF)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- **Meta Object Facility (MOF)**
<http://www.omg.org/spec/MOF/2.5.1/>
- Спецификация MOF на языке XMI
<http://www.omg.org/spec/MOF/20131001/MOF.xmi>

UML – Unified Modeling Language.

Практические занятия по курсу

- **Решение задач**

- Преобразование UML диаграмм в текст программы на языке программирования (Java | C# | C++)
- Преобразование текста программы на языке программирования (Java | C# | C++) в диаграмму UML

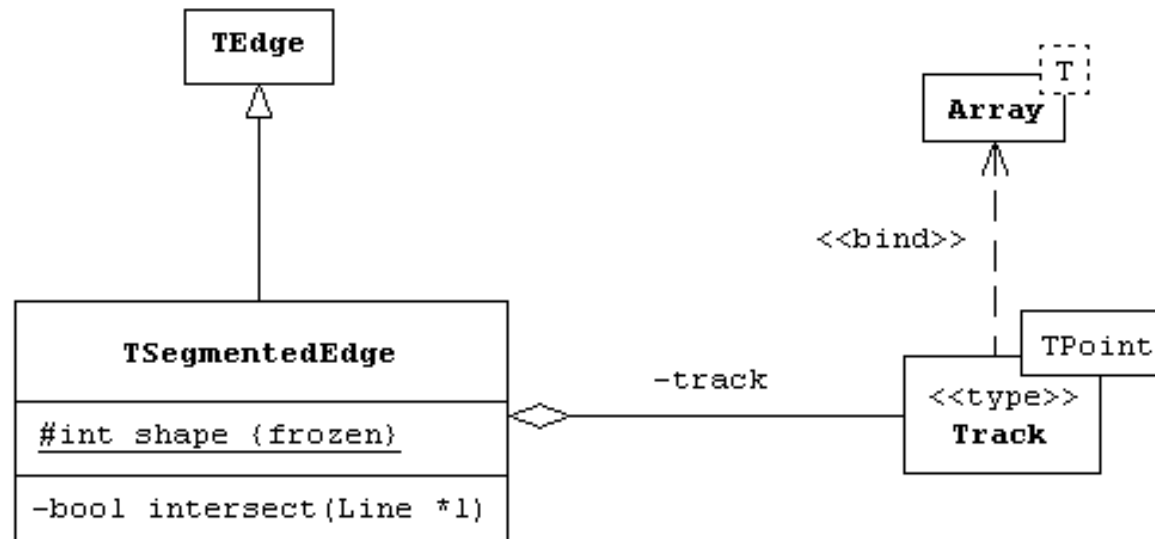
UML – Unified Modeling Language.

Практические занятия по курсу

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- **Задача 1.** Представить следующую диаграмму на языке UML в виде программы на языке C++.



UML – Unified Modeling Language.

Практические занятия по курсу

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

```
class Graph {  
    public Nodes nodes;  
    public Edges edges;  
    private Diagram d;  
    public Graph() {  
        nodes = new Nodes();  
        edges = new Edges();  
    }  
    public void draw(Diagram d) {  
        geometry.Rect R = d.getArea();  
        nodes.draw(d);  
        edges.draw(d);  
        r.draw(d);  
    }  
    static public void draw(Diagram d) {  
        d = new Diagram();  
        Graph g = new Graph();  
        g.draw(d);  
    }  
} // class Graph
```

UML – Unified Modeling Language. Совместные проекты по курсу (1)

Совместный проект 1

- Используя механизм рефлексии языка Java разработать программу строящую UML-модель исполняемого файла языка Java, а затем используя эту модель, строящую UML-диаграмму наследования классов в модели. Для визуализации UML-диаграммы использовать векторную графику формата HTML5 и SVG.

UML – Unified Modeling Language. Совместные проекты по курсу (2)

Совместный проект 2

- Используя механизм JDBC языка Java для анализа метаданных базы данных построить UML модель структуры базы данных и визуализировать ее с помощью графической нотации языка UML в форматах HTML5 и SVG

Литература (1)

1. Object Management Group,
UML 2.5 Superstructure Specification,
OMG document ptc-06-04-02.pdf
www.omg.org/uml
2. Буч Г., Якобсон А., Рамбо Дж., Орлов С.А.
UML. Классика CS. 2-е изд. 2005 год.
ISBN 5-469-00599-2
3. International Standard ISO/IEC 14482.
Programming Languages – C++.
4. Бьерн Страуструп. Язык программирования C++.
Издательство Бином. Москва. 1999.
ISBN 5-7989-0127-0.

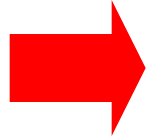
Литература (2)

5. James Gosling, Bill Joy, Guy Steele, Gilad Bracha. The Java™ Language Specification. Third Edition. ISBN 0-321-24678-0
6. Брюс Эккель. Философия Java. 3-е издание. Издательство «Питер». Петербург 2003. ISBN 5-88782-105-1
7. Standard ECMA-334 3rd Edition / June 2006
C# Language Specification.
www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf
8. Эндрю Троелсен. C# и платформа .NET. Издательство «Питер». Петербург 2002. ISBN 5-318-00750-3

Роль графической нотации языка UML

- Нотация языка UML – стандартная нотация современных CASE-инструментов.
- Семантика языка UML (метамодель языка UML)– описывается с помощью нотации языка UML.
- Нотация языка UML широко используется в литературе для описания структуры и поведения программного обеспечения
- В данном курсе нотации UML используется для сравнительного изучения понятий языка моделирования UML и языков программирования C#, Java, C++.

Разновидности UML-диаграмм. Обзор назначения диаграмм (1)



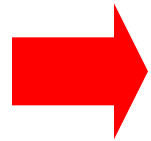
- Диаграмма статической структуры
- Диаграмма прецедентов.
- Диаграмма коммуникации объектов.
- Диаграмма последовательности взаимодействия объектов.

Разновидности UML-диаграмм. Обзор назначения диаграмм (2)

- Диаграмма состояний и переходов.
- Диаграмма деятельности.
- Диаграмма модулей и компонентов.
- Диаграмма внедрения.

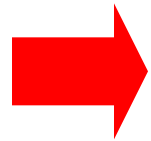
Диаграммы статической структуры

1. Диаграмма статической структуры. Классификаторы и пакеты



- Классификаторы и пакеты на диаграмме статической структуры
- Отношения между классификаторами и пакетами на диаграмме статической структуры

1.1 Классификаторы

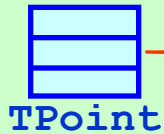


- Классификаторы
- Свойства классификаторов
- Пакеты и их свойства
- Шаблоны для классификаторов
и функций

Варианты изображения классов в графической нотации UML

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



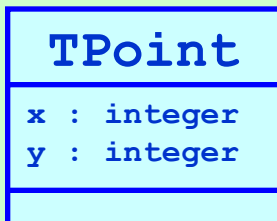
Изображение *класса* в виде пиктограммы



Сжатое изображение *класса*



Изображение *класса* с секциями

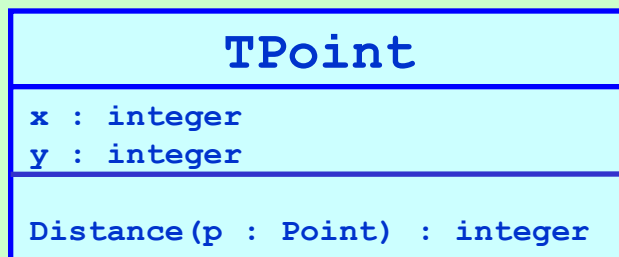


Изображение *класса* с раскрытой секцией атрибутов

Варианты изображения классов в графической нотации UML

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Классы в нотации языков UML, C++, Java и C#

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

```
case::geometry::TPoint
```

Квалифицированное
имя класса

```
// C#  
namespace case.geometry {  
    public class TPoint {  
        int x, y;  
    }  
}
```

```
// Java  
package case.geometry;  
  
public class TPoint {  
    int x, y;  
}
```

```
// C++  
namespace case {  
    namespace geometry {  
        class TPoint {  
            int x, y;  
        };  
    }  
}
```

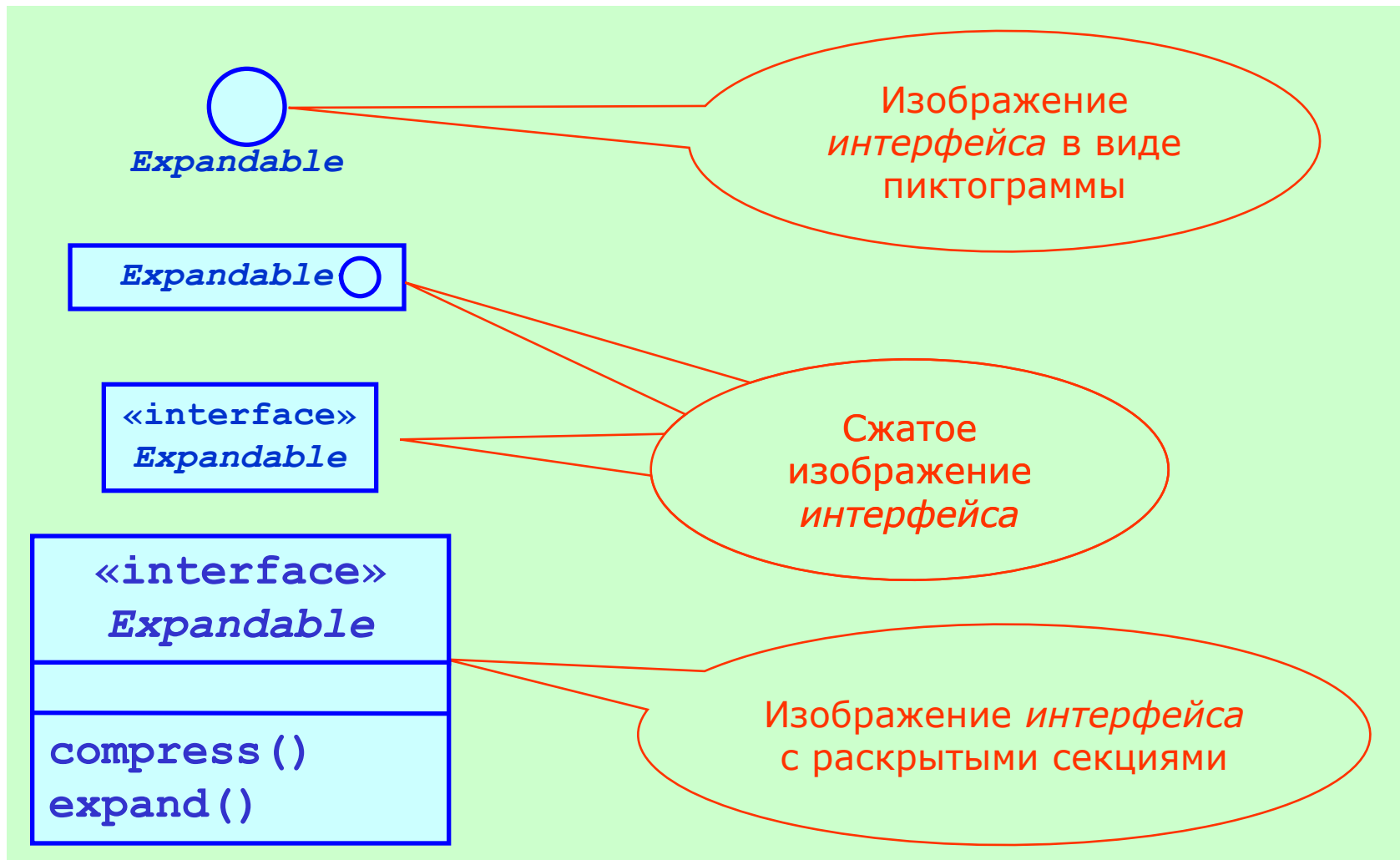

Назначение интерфейсов

- Интерфейс – предоставляет спецификацию множества согласованных операций. Метафора – *«что умеет делать»* класс реализующий данный интерфейс.
- Позволяет отделить спецификацию операций от их реализации. Спецификация одна, реализаций может быть много.
- Позволяет строить независимые иерархии наследования для интерфейсов и классов.
- Позволяет реализовывать более универсальные алгоритмы, работающие с интерфейсами. Алгоритмы применимы к множеству классов реализующих интерфейсы. Эти алгоритмы могут быть унаследованы в классах-потомках.

Варианты изображения интерфейсов в графической нотации UML

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Интерфейсы в нотации языков UML, C++, Java и C#

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

«interface»
Expandable

compress()
expand()

```
// C#  
public interface Expandable {  
    void compress();  
    void expand();  
}
```

```
// Java  
public interface Expandable {  
    public void compress();  
    public void expand();  
}
```

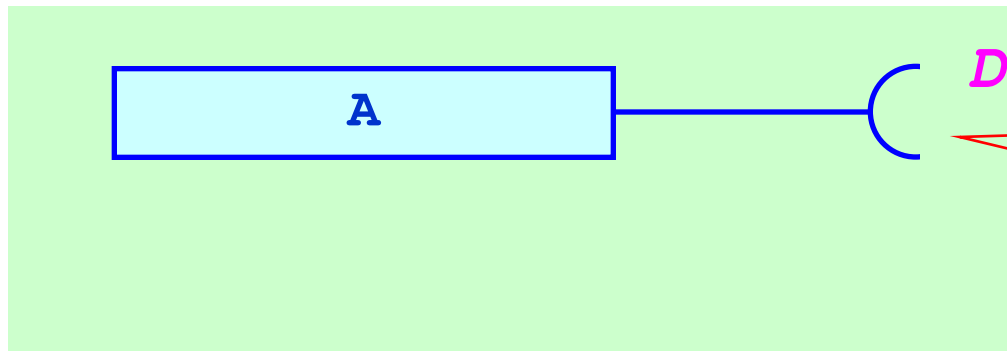
```
// C++  
class Expandable {  
public:  
    virtual void compress() = 0;  
    virtual void expand() = 0;  
};
```

Чистые
виртуальные
функции

Нотация «использование интерфейса» в языке UML 2.1

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Класс **A**
использует
интерфейс **D**

```
// Java
import D;

public class A {
    public D getD() {
        // ...
    }
    public void modify(D d) {
        // ...
    }
}
```

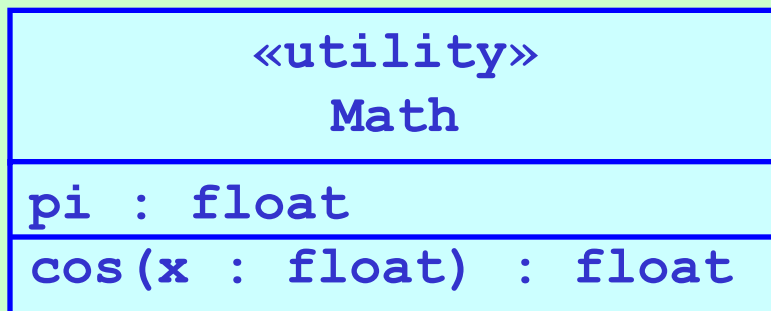
Пример на языке Java:
Класс **A** использует
интерфейс **D**

```
// Java
public interface D {
}
```

Утилиты в графической нотации UML. Изображение и назначение

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



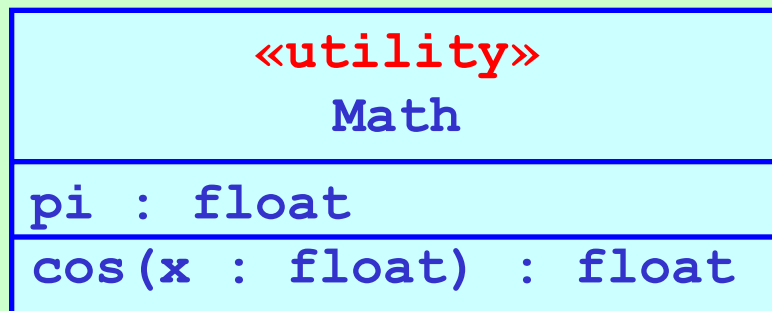
Изображение утилиты
с раскрытыми
секциями

- Назначение - группирование и именование части используемых констант и операций общих для множества классификаторов.
- Упрощается доступ к константам и операциям через имя утилиты.
- Не существует экземпляров утилиты.
- Реализуется с помощью статических атрибутов и операций

Утилиты в графической нотации UML. Реализация утилит

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



```
// Java
public class Math {
    final public static float pi = 3.14;
    public static float sin(float x) { ... }
}
```

```
// C++
class Math {
public:
    static const float pi = 3.14;
    static float sin(float x) { ... };
};
```

Синглетон в графической нотации UML. Изображение, назначение и реализация

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



«singleton»

A

Этот стереотип означает, что существует не более одного экземпляра класса **A**

```
// Java
public class A {
    static private A instance;

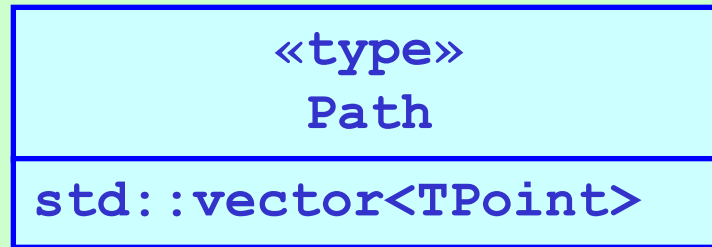
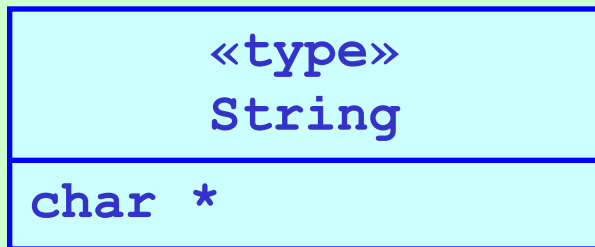
    private A() {}

    static public A getA() {
        if (instance = null)
            instance = new A();
        return instance;
    }
}
```

Типы в графической нотации UML. Изображение, назначение и реализация

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



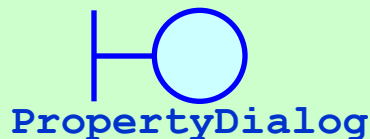
```
// C++  
typedef char* String;  
  
typedef std::vector<TPoint> Path;
```

- Назначение описания типов в языке C++ – локализация и упрощение модификации определений типов.

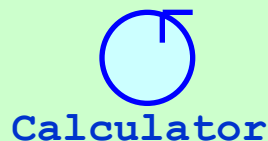
Классификаторы стадии анализа требований к системе

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



<<boundary>> Граничный классификатор.
Через граничные классификаторы система взаимодействует с внешней средой.



<<control>> Управляющий классификатор.
Управляющие классификаторы реализуют логику работы системы.

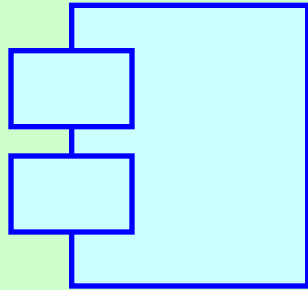


<<entity>> Классификатор - Сущность.
Сущности локализуют зависимость системы от способов хранения информации.

Назначение классификаторов стадии анализа:
построение 3-х уровневой модели системы устойчивой к изменениям:

- в способах взаимодействия системы с внешней средой
- в способах хранения информации.

Классификаторы – Компоненты



King.java

Компонент – представляет

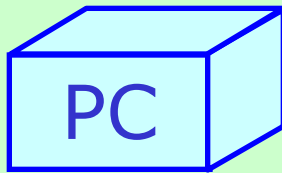
- тексты программ
- исполняемые файлы на интерпретируемых языках
- исполняемые двоичные файлы
- файлы с двоичными данными
- таблицы баз данных

- Назначение компонент – моделирование физической реализации системы. Далее компоненты будут рассмотрены на UML-диаграммах реализации (Implementation Diagrams).

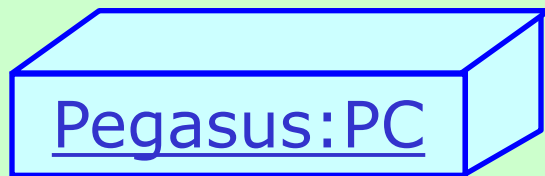
Классификаторы – Вычислительные Узлы

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Вычислительный узел – классы вычислительных узлов моделируемой системы



Экземпляр вычислительного узла – элемент для построения вычислительных сетей моделируемой системы

- Назначение *вычислительных узлов* – моделирование физической реализации системы. Далее вычислительные узлы будут рассмотрены на *UML-диаграммах реализации*.

Классификаторы – Прецеденты

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Make Move

Прецедент – представляет функциональность системы. Например, перемещение элемента диаграммы.

by Mouse:
Make Move

by Keyboard:
Make Move

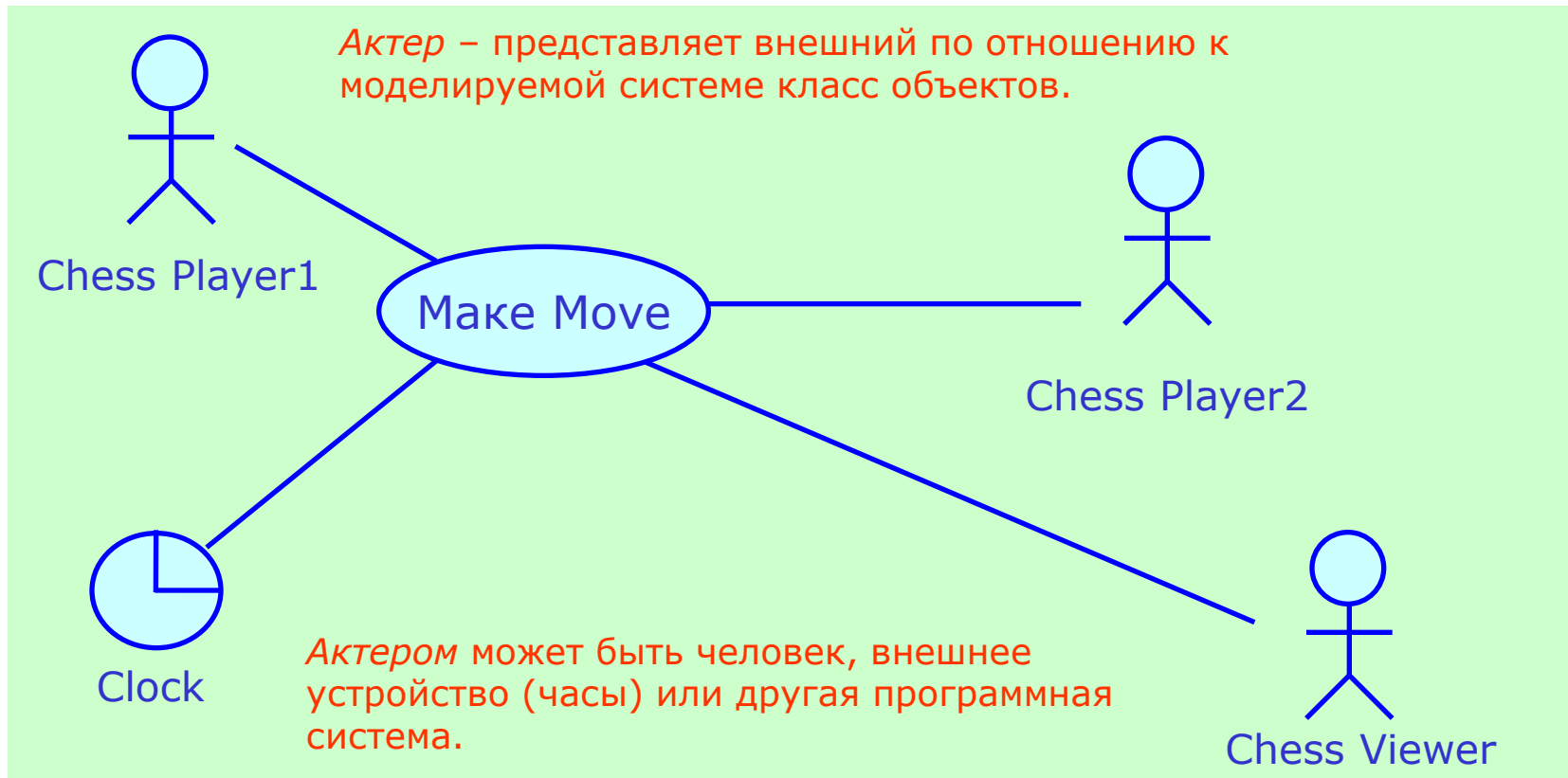
Экземпляр прецедента – один из способов реализации функциональности моделируемой системы

- Назначение *прецедентов* (использования системы): моделирование взаимодействия системы с внешней средой. В дальнейшем прецеденты будут рассмотрены на *UML-диаграммах прецедентов (Use Case diagrams)*.

Классификаторы – Актеры

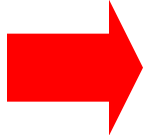
МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



- Назначение *актеров* – моделирование взаимодействия системы с внешней средой. Актеры взаимодействуют с *прецедентами* использования системы.

1.2 Свойства классификаторов

- Классификаторы
-  • Свойства классификаторов
- Пакеты и их свойства
- Шаблоны для классификаторов
и утилит

Графические обозначения свойств классификаторов (курсив)

МГУ им. М.В.Ломоносова. Факультет ВМК.

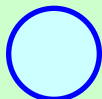
Романов Владимир Юрьевич ©2024

Абстрактность классификатора означает невозможность создания экземпляров данного классификатора.

Абстрактность на диаграмме обозначается курсивом.

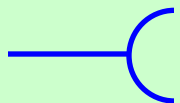


Абстрактный класс



Expandable

Интерфейс



*Message
Handler*

Использование интерфейса

Графические обозначения свойств классификаторов (толщина линии)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Активный класс – класс, экземпляры которого выполняются на отдельном потоке управления. В качестве такого потока управления могут использоваться *процесс (process)* или *нить (thread)*.

Активные классы обозначаются толстой рамкой.

В языке Java *активные классы* - это классы-потомки класса `Thread` или классы реализующие интерфейс `Runnable`.



Fish

Активный класс



Fish

Альтернативное обозначение
активного класса (нотация UML 2.x)

Обозначения свойств классификатора с помощью теговых значений (1)

- **Теговое значение (tagged value)** - показывается на диаграмме с помощью текста в фигурных скобках:

{ имя_тега = значение_тега }

- **Теговые значения** – один из трех универсальных механизмов расширения языка UML. Этот механизм расширения применим к любому элементу модели, а не только к классификаторам.
- Количество теговых значений, применимых к элементу модели, неограниченно. На диаграмме могут быть показаны не все теговые значения
- Теги для элемента могут быть заданы:
 - ✓ Определены в стандарте на язык UML
 - ✓ Заданы пользователем при создании модели
 - ✓ Определены используемым CASE-инструментом

Обозначения свойств с помощью теговых значений (2)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Для тегов принимающих логические значения (**true**, **false**) значение тега можно не указывать. Присутствие имени тега означает логическое значение **true**. Отсутствие имени тега означает логическое значение **false**.

Пример: представление свойства абстрактности не курсивом, а теговым значением.

View

{abstract}

Абстрактный класс **View**

Node

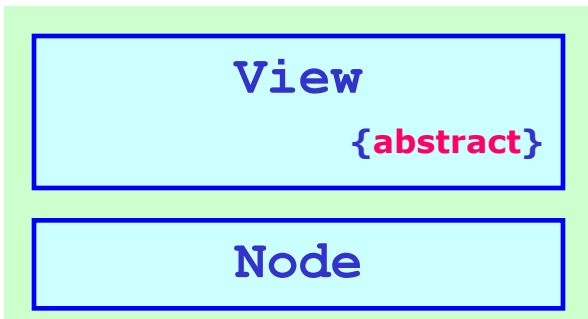
Конкретный класс **Node**

Реализация стандартных свойств языка UML: *abstract* и *leaf*

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

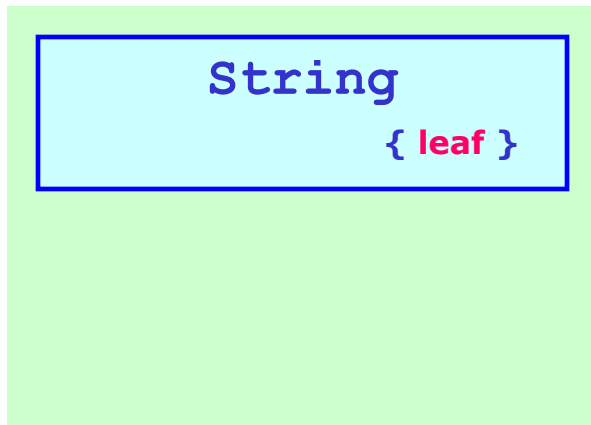
Реализация абстрактности класса (отсутствия экземпляров у класса).



```
// Java, C#
abstract class View {
}

class Node {
}
```

Реализация завершенности класса (**leaf** – лист в дереве наследования)
У завершенного класса не может быть классов-потомков.



```
// Java
final class String {
}
```

```
// C#
sealed class String {
}
```

Стереотипы классификаторов

- *Стереотип* – новый вид элемента модели, созданный на основе существующего. Стереотипы расширяют семантику модели.
- *Стереотип* – один из трех механизмов расширения языка UML
- Количество стереотипов у элемента модели (и у классификатора, в частности) неограниченно
- Стереотип характеризуется:
 - ✓ Расширяемым классификатором
 - ✓ Набором теговых значений
 - ✓ Набором ограничений (constraints)

Теги с произвольным типом значений

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Edge

```
{author="Romanov"}
```

Тег **author** представляющий автора разработки класса в UML-модели и текстах реализации класса

Edge

```
{version="1.10.623"}
```

Тег **version** от системы управления версиями UML-модели

Edge

```
{file="Edge.java",  
line=2,  
column=8 }
```

Теги для связи UML-модели и текстов на языке программирования

```
/** Java  
 * @author Romanov  
 */  
public class Edge {  
}
```

```
/// C#  
/// <author>Romanov</author>  
///  
public class Edge {  
}
```

Изображение стереотипов классификаторов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Изображение стереотипа *boundary*

«boundary»
Edge

Текстовое изображение стереотипа для узла графа

Edge 

Графическое изображение стереотипа для узла графа


Edge

Графическое изображение стереотипа для узла графа сжатого в пиктограмму

«control, singleton»
Sorter

Множество стереотипов класса *Sorter*

Ограничения накладываемые на UML-модель

МГУ им. М.В.Ломоносова. Факультет ВМК.

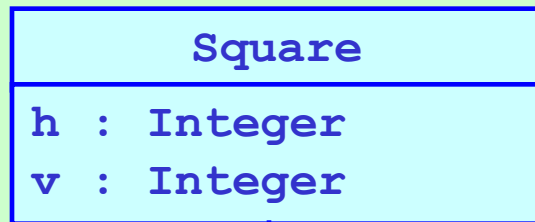
Романов Владимир Юрьевич ©2024

- *Ограничение (constraint)* – логическое выражение описанное с использованием атрибутов и отношений ассоциации классификаторов
- Истинно все время жизни элементов UML-модели
- Не изменяет состояние элементов UML-модели
- Может быть применено к любому элементу UML-модели
- Чаще всего ограничения описываются на языке **OCL** (**O**bject **C**onstraint **L**anguage)

Изображение ограничений на UML-диаграммах

МГУ им. М.В.Ломоносова. Факультет ВМК.

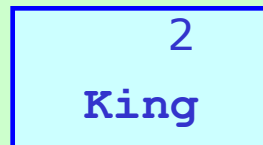
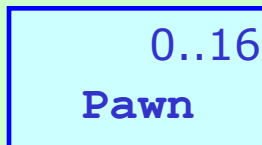
Романов Владимир Юрьевич ©2024



Элемент модели, на который накладывается ограничение



Ограничение записанное на языке OCL и использующее атрибуты класса



Ограничение на количество экземпляров класса

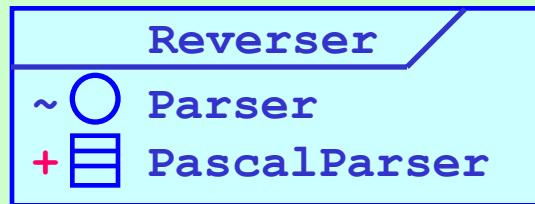
Видимость классификаторов, их атрибутов и операций

- + *публичная (public)* – элемент модели видим вне своего пространства имен
- # *защищенная (protected)* – элемент модели вне своего пространства имен видят только его потомки
- *скрытая (private)* – элемент модели невидим вне своего пространства
- ~ *пакетная (package)* – элемент модели невидим вне пакета содержащего классификатор владеющий данным элементом модели

Пример изображения видимости классификаторов (1)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



~Reverser::Parser○

+Reverser::PascalReverser

```
// Java
package Reverser;
interface Parser {}
```

```
// Java
package Reverser;
public class PascalParser {}
```

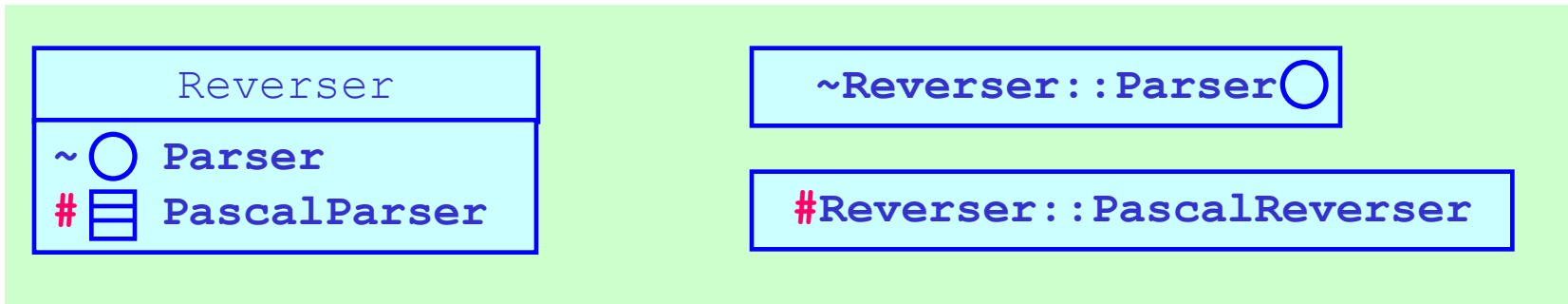
```
// C#
namespace Reverser {
    interface Parser {
    }
}
```

```
// C#
namespace Reverser {
    public class PascalParser {
    }
}
```

Пример изображения видимости классификаторов (2)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



```
// Java
class Reverser {
    interface Parser {
    }
}
```

```
// Java
class Reverser {
    protected class PascalParser {
    }
}
```

```
// C#
class Reverser {
    interface Parser {
    }
}
```

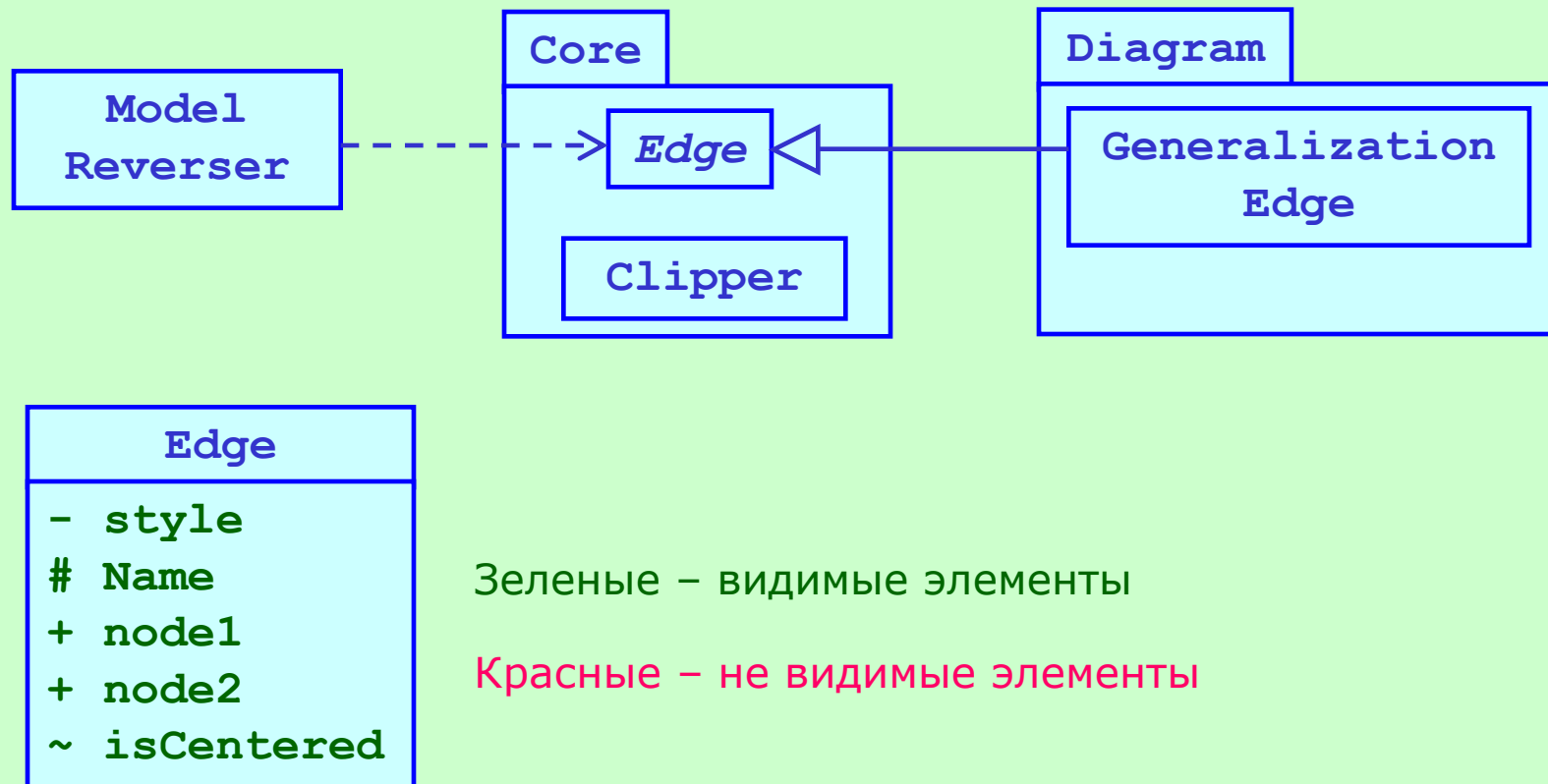
```
// C#
class Reverser {
    protected class PascalParser {
    }
}
```

Пример изображения видимости атрибутов и операций (1)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Атрибуты класса Edge видимые для класса Edge

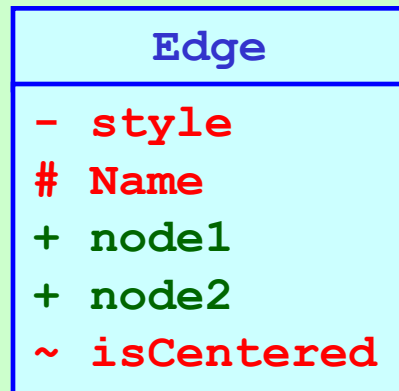
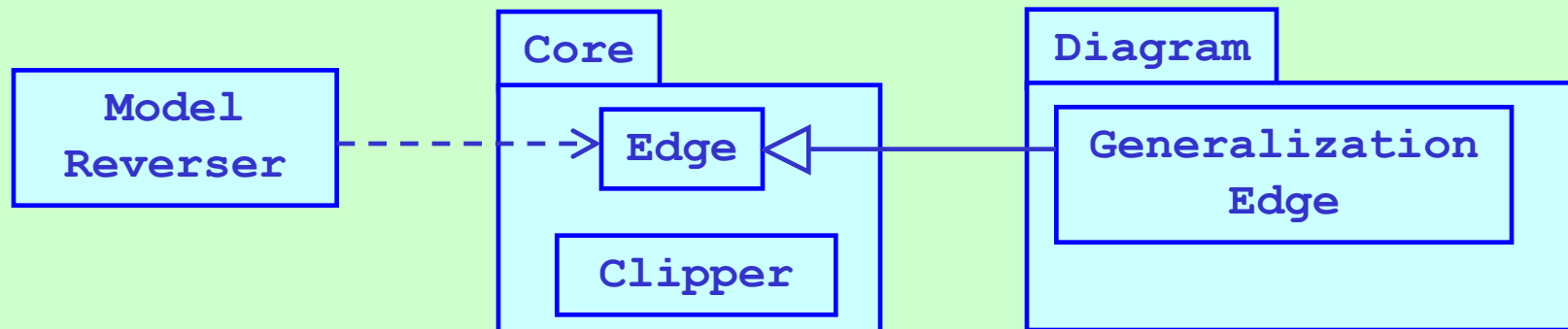


Пример изображения видимости атрибутов и операций (2)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Атрибуты класса Edge видимые для класса ModelReverser



Зеленые – видимые элементы

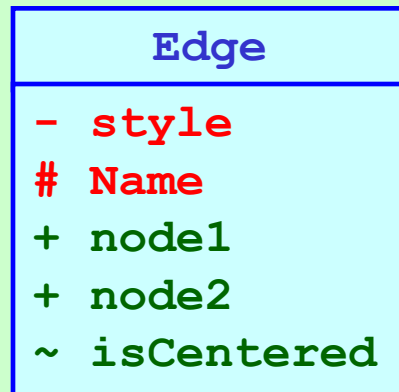
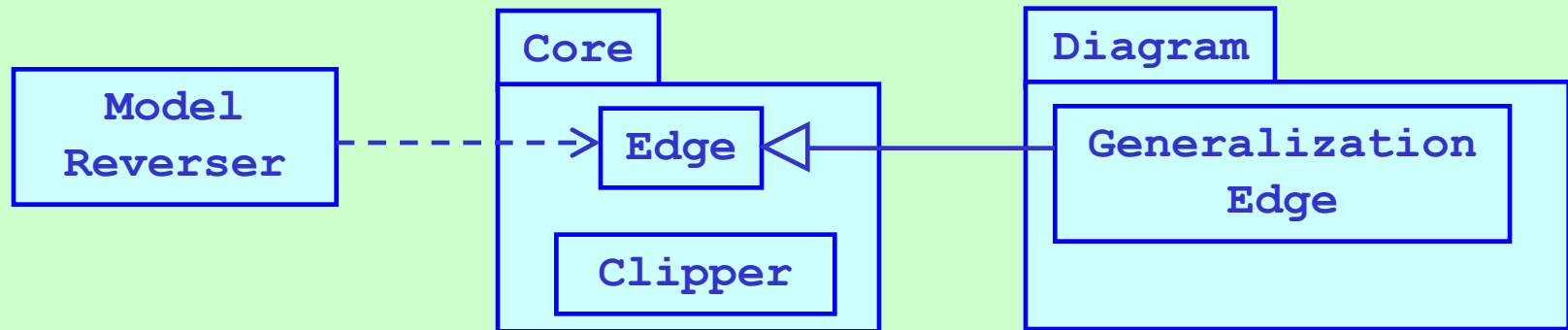
Красные – не видимые элементы

Пример изображения видимости атрибутов и операций (3)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Атрибуты класса Edge видимые для класса Clipper



Зеленые – видимые элементы

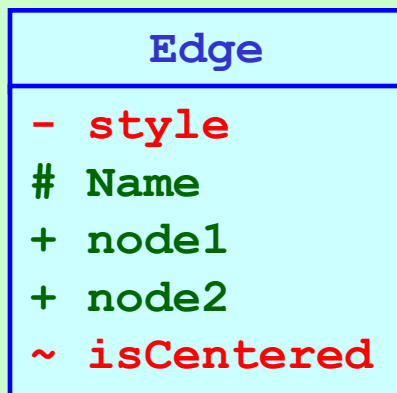
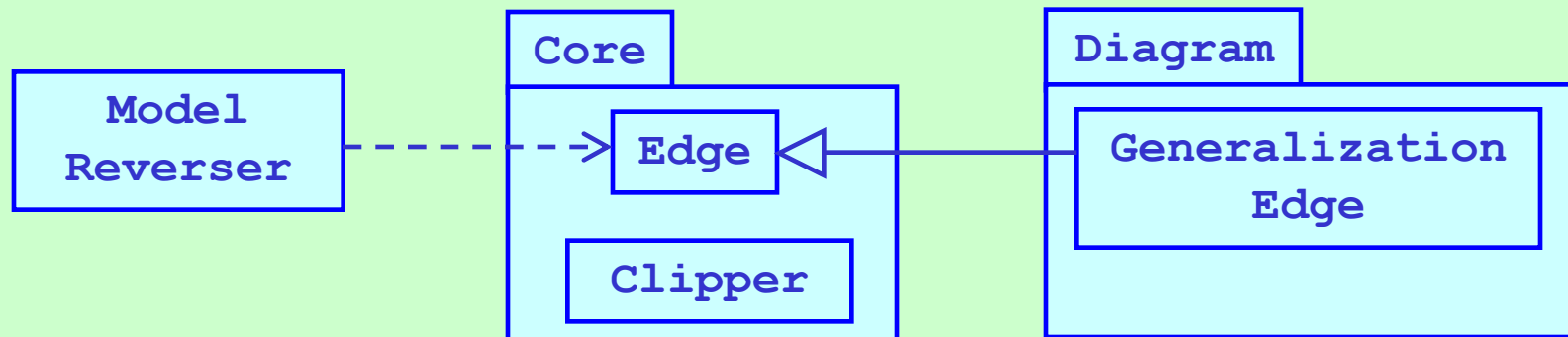
Красные – не видимые элементы

Пример изображения видимости атрибутов и операций (4)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Атрибуты класса Edge видимые для класса GeneralizationEdge



Зеленые – видимые элементы

Красные – не видимые элементы

Область действия атрибутов и операций класса (1)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- Область действия – *класс*. Атрибуты и операции с такой областью действия называются *статическими*. Они доступны всем экземплярам класса
- Область действия – *экземпляр класса*. Атрибуты и операции с такой областью только этому экземпляру класса

Node
+ frameColor : Color
+ <u>defaultFrameColor: Color</u>

Статические элементы класса
показываются подчеркиванием

Область действия атрибутов и операций класса (2)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Node

```
+ frameColor : Color  
+ defaultFrameColor: Color
```

Статические элементы класса
показываются подчеркиванием

```
// Java, C#  
class Node {  
    static public Color defaultFrameColor;  
    public Color frameColor;  
}
```

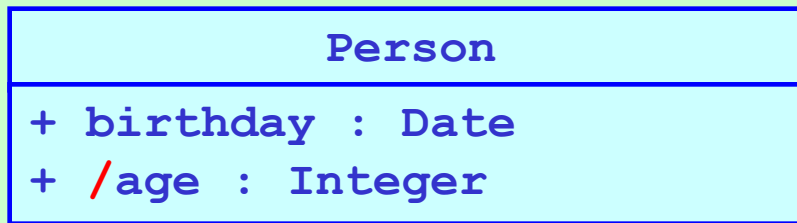
```
// C++  
class Node {  
public:  
    static Color defaultFrameColor;  
    Color frameColor;  
};
```

Свойства атрибутов. Порожденные атрибуты (1)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Значения порожденных атрибутов не хранятся,
а вычисляются в момент запроса.



На диаграмме перед именем порожденного атрибута стоит символ /

```
// Java
class Person {
public Date birthday;

public int getAge() {
    return CurrentDate - birthday;
}
}
```

```
// C#
class Person {
public Date birthday;

public int age {
    get { return CurrentDate -
        birthday; }
}
}
```

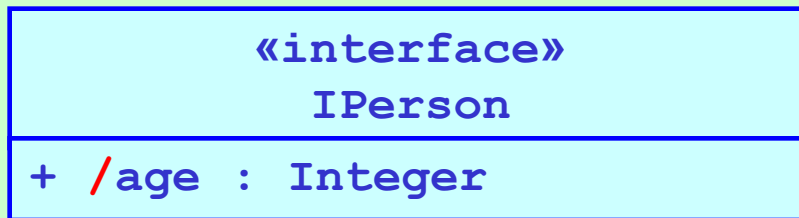
Свойства атрибутов.

Порожденные атрибуты (2)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Значения порожденных атрибутов не хранятся,
а вычисляются в момент запроса.



На диаграмме перед именем порожденного атрибута стоит символ /

```
// Java
interface Person {
    public int getAge();
}
```

```
// C#
interface Person {
    public int age { get; }
}
```

Свойства атрибутов. Неизменяемые атрибуты

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

«utility»

Math

+ pi : Double { **frozen** }

Атрибуты *frozen* вычисляются
на этапе компиляции.

```
// Java
class Math {
public final double pi = 3.14;
}
```

```
// C#
class Math {
public const double pi = 3.14;
}
```

```
// C++
class Math {
public:
    const pi = 3.14;
}
```

Свойства атрибутов. Атрибуты только для чтения

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

History

```
+ events : List { readonly }
```

Атрибуты *readonly*
неизменяемы вне класса.
Внутри класса возможны
чтение и запись.

```
// C#  
class Math {  
    public readonly List events;  
}
```

Свойства методов. Методы - запросы

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Node

```
+ paint(g : Graphics) { query }
```

```
// C++  
class Node {  
public:  
    void paint(Graphics g) const;  
}
```

1. Метод-запрос не может изменять унаследованные и собственные атрибуты
2. Метод-запрос не может не может вызывать собственные и унаследованные методы, которые не являются методами-запросами.

Свойства методов.

Не переопределяемые методы

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Account

```
+ Withdraw(n : long) {leaf}
```

leaf – запрет переопределения
метода в классах-потомках.

```
// C#  
class Account {  
    leaf  
    public void Withdraw(long n)  
    {  
        ...  
    }  
}
```

```
// Java  
class Account {  
    final  
    public void Withdraw(long n)  
    {  
        ...  
    }  
}
```

Свойства методов.

Не переопределяемые методы

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Account

```
+ Withdraw(n : long) {leaf}
```

leaf – запрет переопределения
метода в классах-потомках.

```
// C#  
class Account {  
    leaf  
    public void Withdraw(long n);  
}
```

```
// Java  
class Account {  
    final  
    public void Withdraw(long n);  
}
```


Свойства методов.

Абстрактность методов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

View

+ *paint*(*g* : *Graphics*) {abstract**}**

abstract – отсутствие реализации у метода.

Абстрактность может показываться также курсивом.

```
// C#, Java
abstract class View {
    abstract public void paint(Graphics g);
}
```

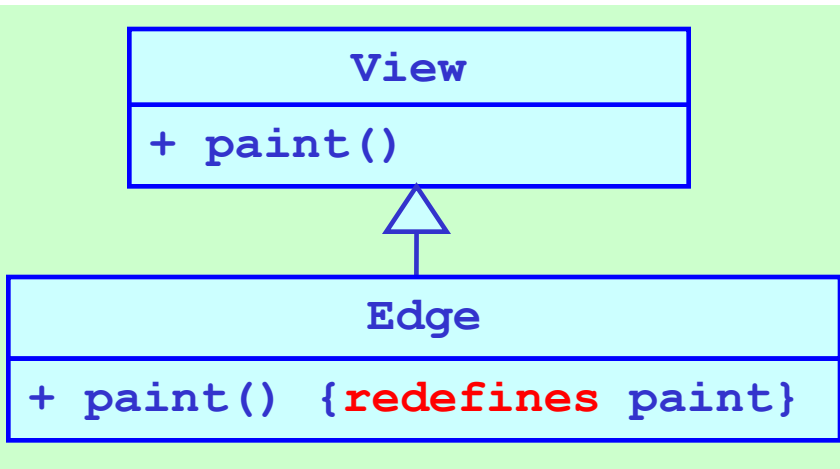
```
// C++
class View {
public:
    virtual void paint(Graphics g) = 0;
}
```

Свойства методов.

Переопределение методов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



```

// Java
class View {
    public void paint();
}

class Edge extends View {
    public void paint();
}
  
```

```

// C#
class View {
    virtual public void paint();
}

class Edge : View {
    override public void paint();
}
  
```

```

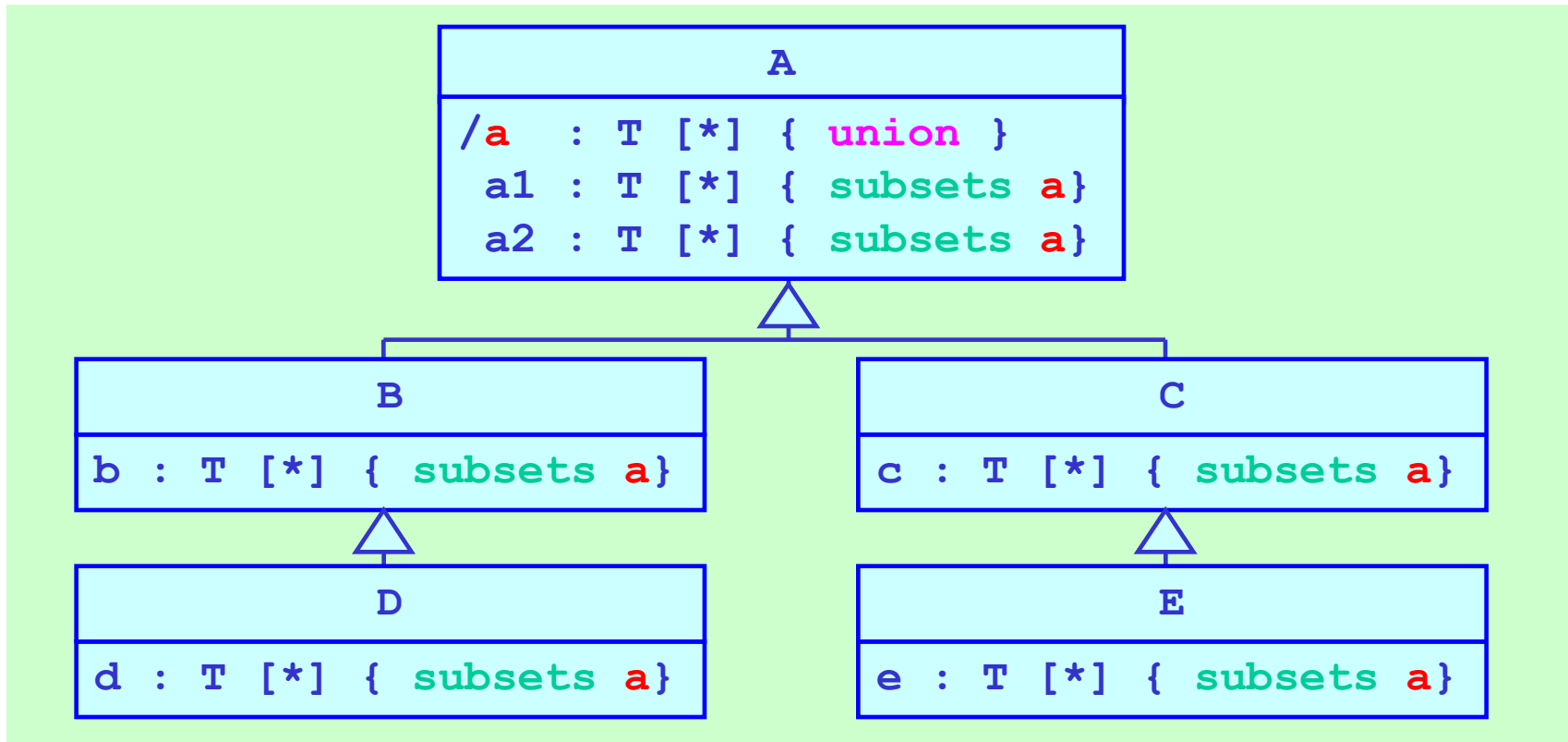
// C++
class View {
public:
    virtual void paint();
};

class Edge : View {
public:
    void paint();
};
  
```

Объединения и подмножества атрибутов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



```
// Java
D x = new D();
T[] y = x.getA();
// y = a1 + a2 + b + d
```

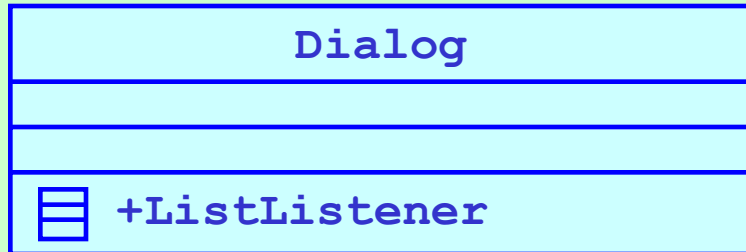
```
// Java
B x = new B();
T[] y = x.getA();
// y = a1 + a2 + b
```

```
// Java
A x = new A();
T[] y = x.getA();
// y = a1 + a2
```

Вложенность классификаторов

МГУ им. М.В.Ломоносова. Факультет ВМК.

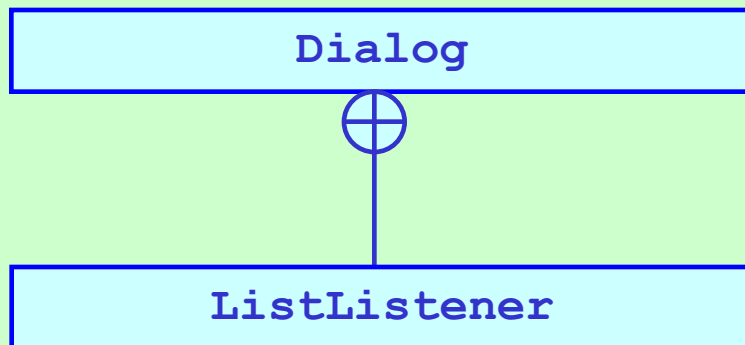
Романов Владимир Юрьевич ©2024



Вложенность классификатора *ListListener* показывается в отдельной секции классификатора *Dialog*.

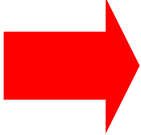


Вложенность классификатора *ListListener* показывается с помощью квалифицированного имени.



Вложенность классификатора *ListListener* показывается с помощью отношения вложенности между узлами графа *ListListener* и *Dialog*.

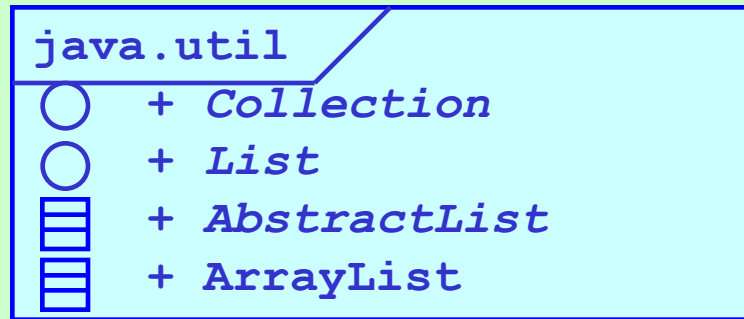
1.3 Классификаторы и пакеты в нотации языка UML

- Классификаторы
- Свойства классификаторов
-  • Структурирование модели с помощью пакетов
- Шаблоны для классификаторов и функций

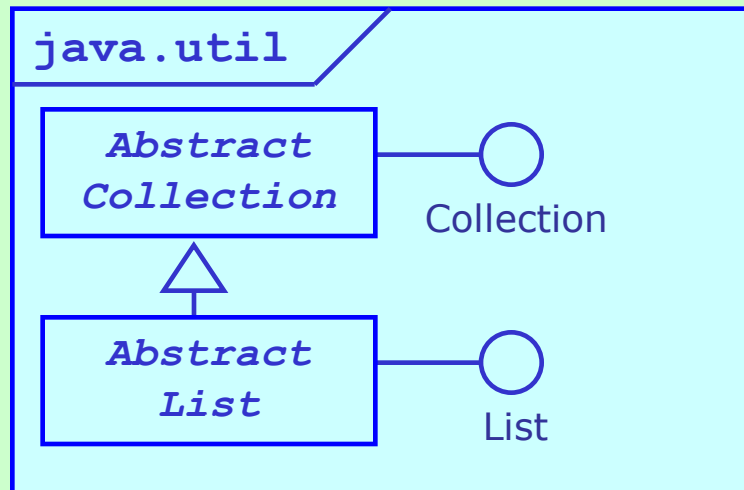
Изображение пакетов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Изображение «раскрытого» пакета `java.util` с вложенными в пакет классами и интерфейсами



Изображение «раскрытого» пакета `java.util` с вложенной в пакет UML-диаграммой, показывающей отношения между вложенными в пакет классами.

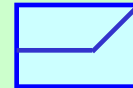
Представление пакетов в языках C++, Java и C#

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

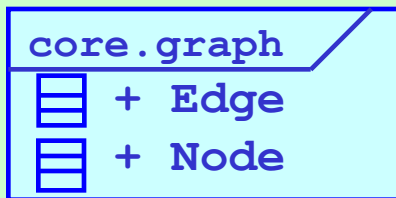


Изображение «сжатого»
пакета java.util



java.util

Изображение-пиктограмма
пакета java.util



```
// C#
namespace core.graph {
    class Edge {};
    class Node {};
}
```

```
// Java
package core.graph;

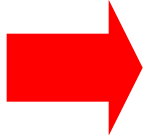
public class Edge {
}
```

```
// C++
namespace core {
    namespace graph {
        class Edge {};
        class Node {};
    }
}
```

```
// Java
package core.graph;

public class Node {
}
```

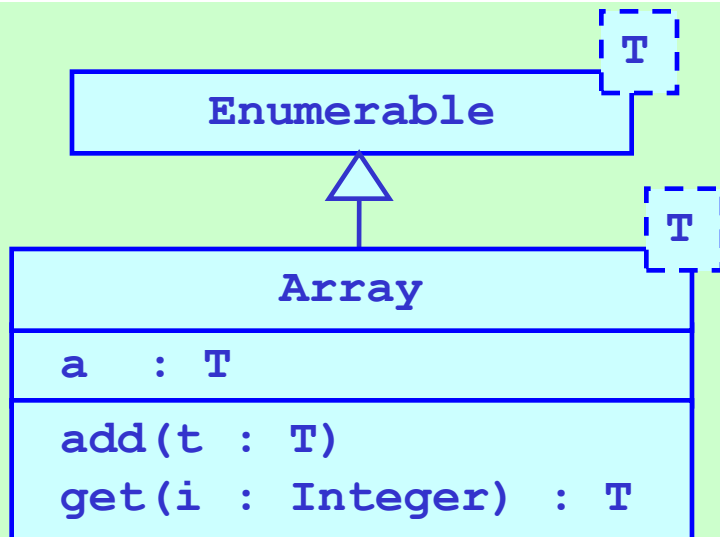
1.4 Шаблоны для классификаторов и функций

- Классификаторы
- Свойства классификаторов
- Пакеты и их свойства
-  • Шаблоны для классификаторов и функций

Шаблоны для классификаторов и функций. Шаблон класса C#

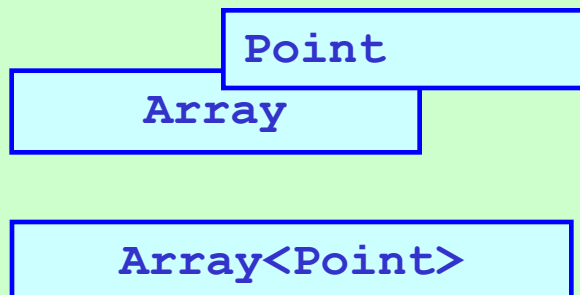
МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Объявление шаблона класса

```
// C# 2.0
class Array<T> : Enumerable<T> {
    public T a;
    public void add(T t) {}
    public T get(int i) {}
}
```



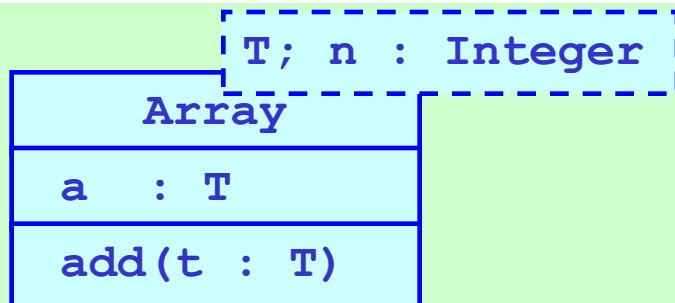
Настройка шаблона класса

```
// C# 2.0
Array<Point> path;
```

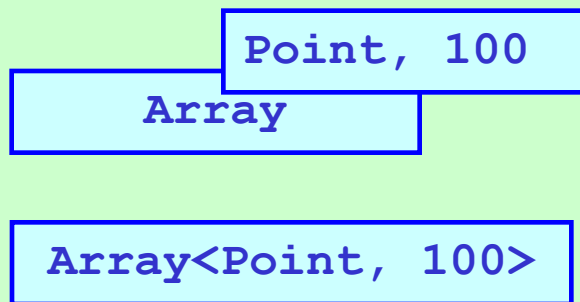
Шаблоны для классификаторов и функций. Шаблон класса C++

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



```
// C++
template <class T, int n>
class Array {
public:
    T a;
    void add(T t);
};
```



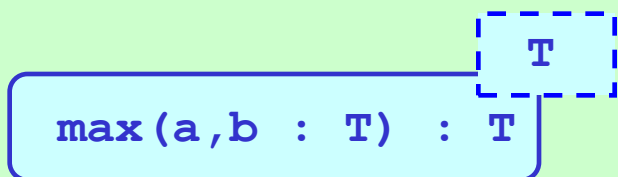
Настройка шаблона класса

```
// C++
Array<Point, 100> path;
```

Шаблоны для классификаторов и функций. Шаблон функции C++

МГУ им. М.В.Ломоносова. Факультет ВМК.

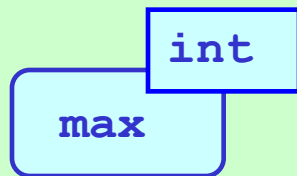
Романов Владимир Юрьевич ©2024



```
max(a, b : T) : T
```

Объявление шаблона функции

```
// C++  
template <class T>  
T max(T a, T b)  
    { return a > b ? a : b; }
```

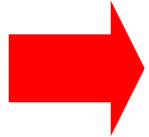


```
max
```

Настройка шаблона функции

```
// C++  
int x = max<int>(y, 2);
```

2.1 Отношение обобщения



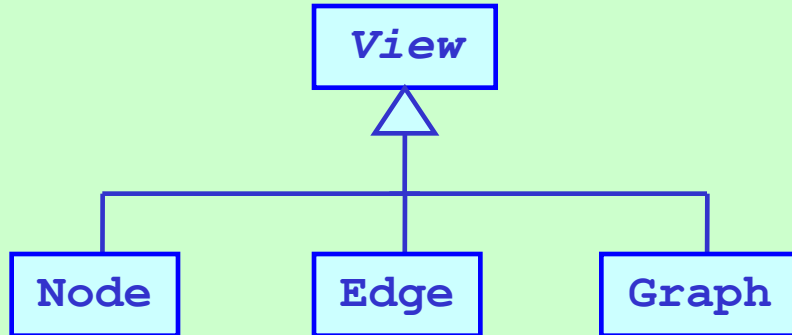
- Отношения обобщения
- Отношения реализации
- Отношение ассоциации
- Отношение зависимости

Отношения обобщения.

Особенности языков C++, Java и C#

МГУ им. М.В.Ломоносова. Факультет ВМК.

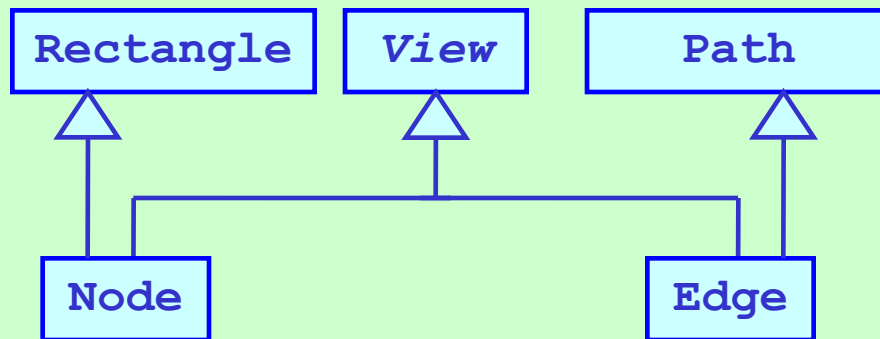
Романов Владимир Юрьевич ©2024



Одиночное наследование в Java и C#

```
// Java
class Node extends View {
}
```

```
// C#
class Node : View {
}
```



Множественное наследование в C++

```
// C++
class Node : public View,
            public Rectangle
{
};

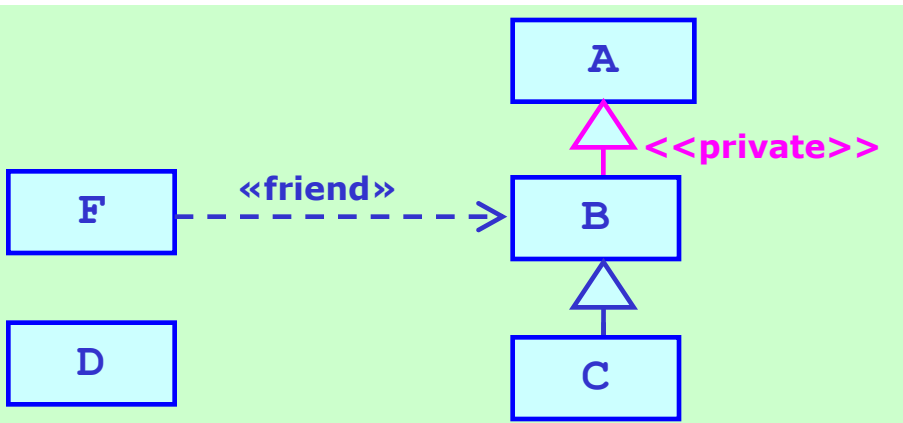
class Edge : public View,
            public Path
{
};
```

Отношения обобщения.

Скрытое наследование в языке C++

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



```

class C : public B {
    void aMember() {
        x = 1; // не допустимо
        y = 1; // не допустимо
        z = 1; // не допустимо
    }
};

```

```

class D {
    A a;
    void aMember() {
        a.x = 1; // не допустимо
        a.y = 1; // не допустимо
        a.z = 1; // допустимо
    }
};

```

```

// C++
class A {
    private:    int x;
    protected: int y;
    public:    int z;
};

```

```

class B : private A {
    friend F;

    void aMember() {
        x = 1; // не допустимо
        y = 1; // допустимо
        z = 1; // допустимо
    }
};

```

```

class F { //
    A a;
    void aMember() {
        a.x = 1; // не допустимо
        a.y = 1; // не допустимо
        a.z = 1; // допустимо
    }
};

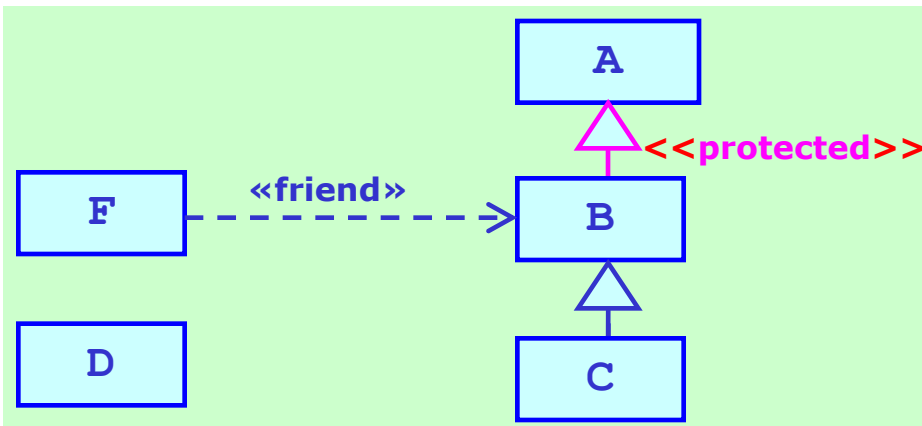
```

Отношения обобщения.

Защищенное наследование в языке C++

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



```
// C++
class A {
    private:    int x;
    protected: int y;
    public:    int z;
};
```

```
class B : protected A {
    friend F;

    void aMember() {
        x = 1; // недопустимо
        y = 1; // допустимо
        z = 1; // допустимо
    };
```

```
class F { //
    void aMember() {
        x = 1; // недопустимо
        y = 1; // допустимо
        z = 1; // допустимо
    };
```

```
class C : public B {
    void aMember() {
        x = 1; // недопустимо
        y = 1; // допустимо
        z = 1; // допустимо
    };
};
```

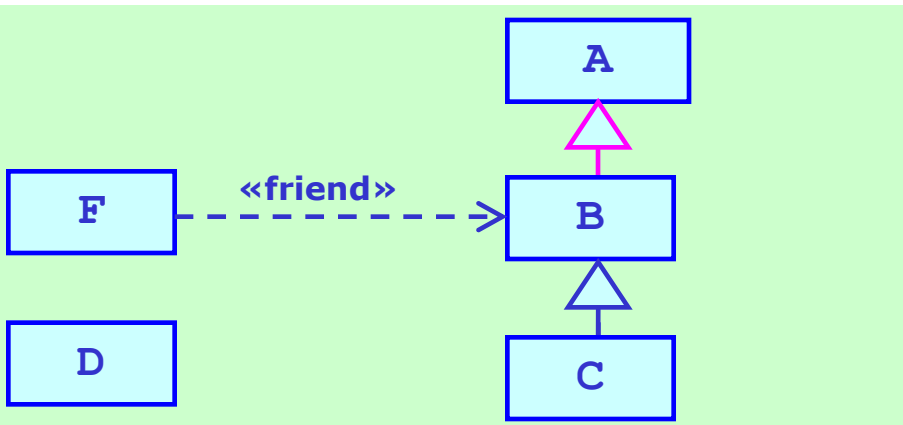
```
class D {
    void aMember() {
        x = 1; // недопустимо
        y = 1; // допустимо
        z = 1; // допустимо
    };
};
```

Отношения обобщения.

Открытое наследование в языке C++

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



```

class C : public B {
    void aMember() {
        x = 1; // недопустимо
        y = 1; // допустимо
        z = 1; // допустимо
    }
};

```

```

class D {
    void aMember() {
        x = 1; // недопустимо
        y = 1; // допустимо
        z = 1; // допустимо
    }
};

```

```

// C++
class A {
    private:    int x;
    protected: int y;
    public:    int z;
};

```

```

class B : public A {
    friend F;

    void aMember() {
        x = 1; // недопустимо
        y = 1; // допустимо
        z = 1; // допустимо
    }
};

```

```

class F { //
    void aMember() {
        x = 1; // недопустимо
        y = 1; // допустимо
        z = 1; // допустимо
    }
};

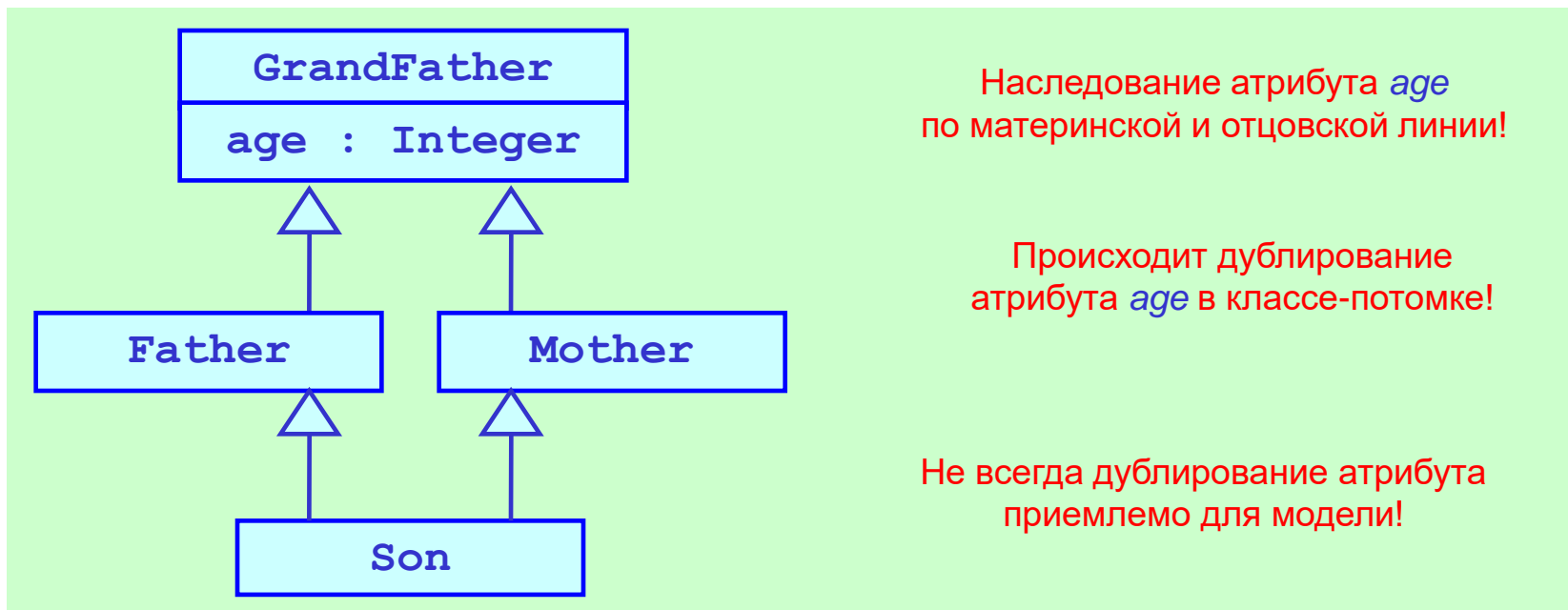
```


Отношения обобщения.

Множественное наследование в языке C++

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



```

class Son :
  public Mother, public Father {
  void aMember() {
    age = 1; // недопустимо
  }
};
  
```

```

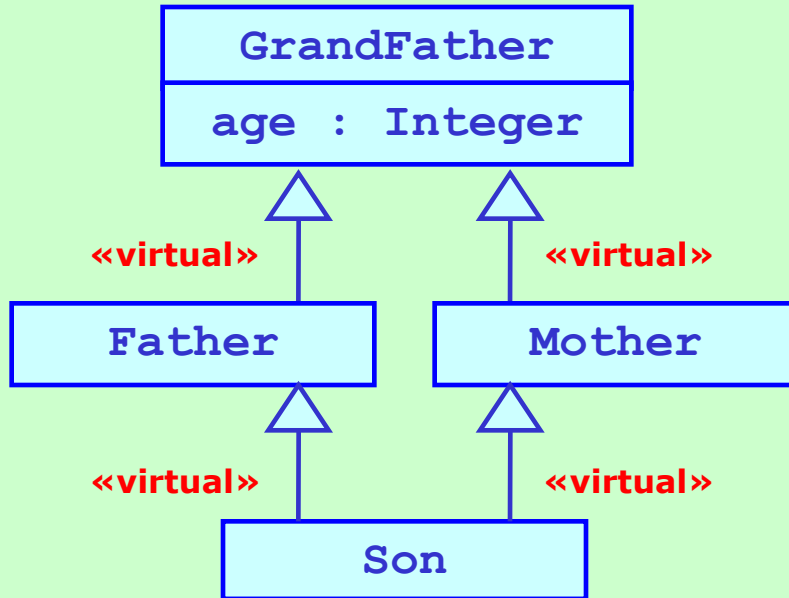
class Son :
  public Mother, public Father
  void aMember() {
    Father::age = 1; // Допустимо
    Mather::age = 1; // Допустимо
  }
};
  
```

Отношения обобщения.

Виртуальное наследование в языке C++

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



При виртуальном наследовании дублирования атрибута *age* не происходит!

```

class Son :
    virtual public Father,
    virtual public Mother {
void aMember() {
    age = 1; // Допустимо
}
};
  
```

```

class Father : virtual public GrandFather {
};
  
```

```

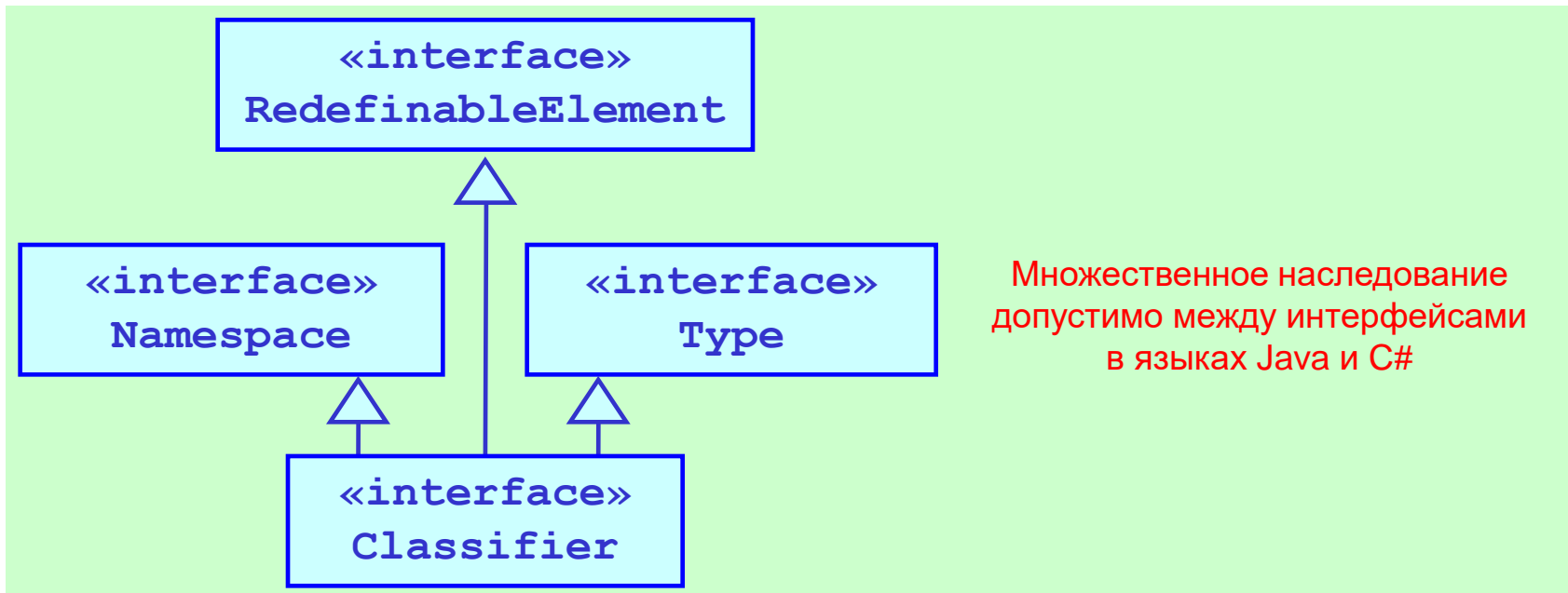
class Mother : virtual public GrandFather {
};
  
```

Отношения обобщения.

Наследование между интерфейсами

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

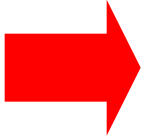


```
// Java
interface Classifier
extends Namespace,
    RedefinableElement,
    Type
{
}
```

```
// C#
interface Classifier
: Namespace,
    RedefinableElement,
    Type
{
}
```

2.2. Отношение реализации

- Отношения обобщения
- Отношения реализации
- Отношение ассоциации
- Отношение зависимости

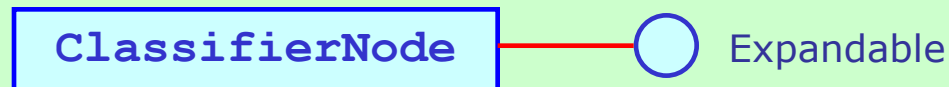


Отношения реализации.

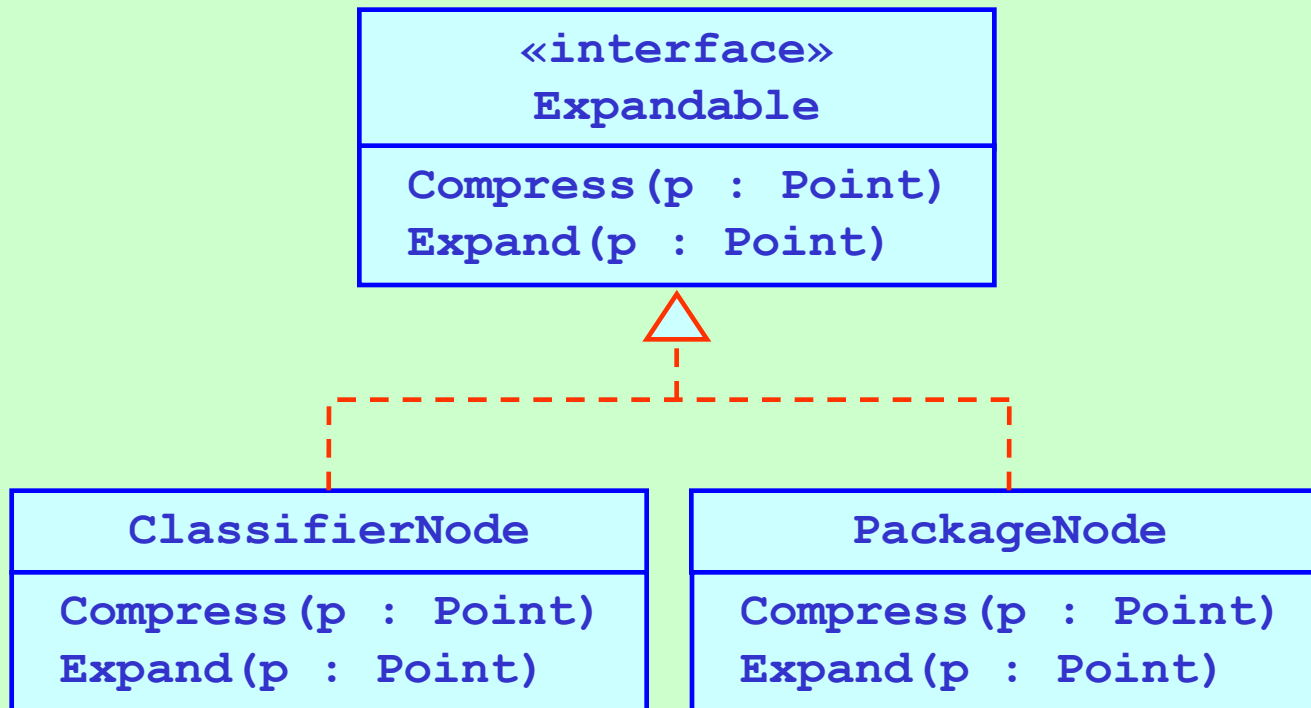
Варианты изображения отношения

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Отношение реализации интерфейса показанного в сжатом состоянии



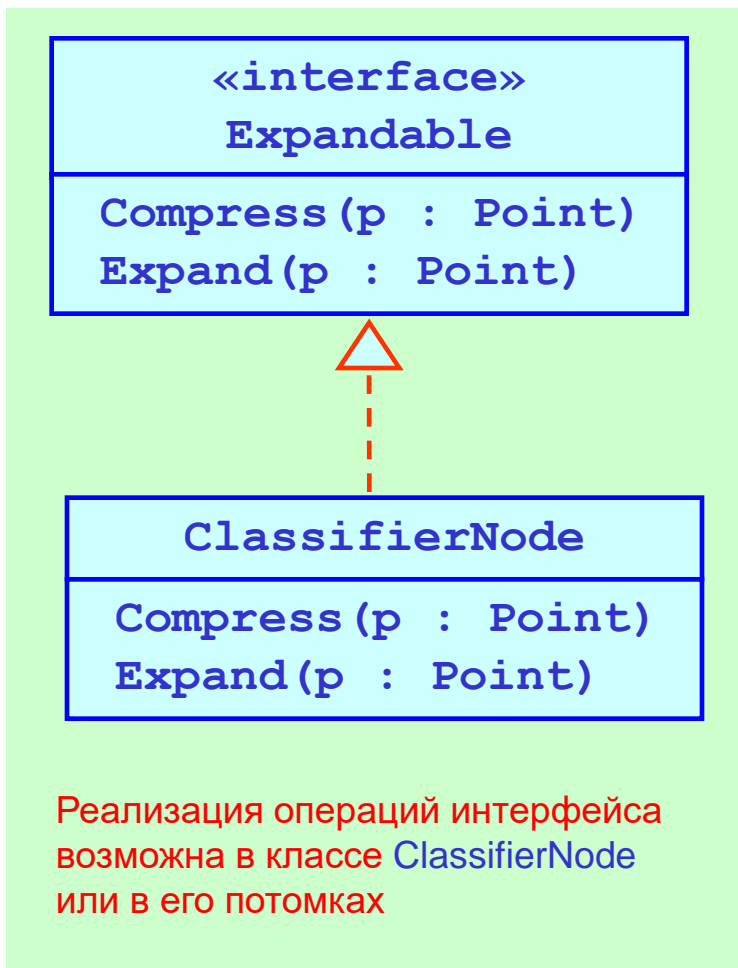
Отношение реализации интерфейса показанного в раскрытом состоянии

Отношения реализации.

Реализация интерфейса в языках Java и C#

МГУ им. М.В.Ломоносова. Факультет ВМК.

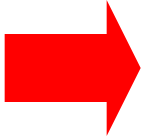
Романов Владимир Юрьевич ©2024



```
// Java
interface Expandable {
    void Compress(Point p);
    void Expand(Point p);
}
class ClassifierNode
implements Expandable {
    void Compress(Point p) {...}
    void Expand(Point p) {...}
}
```

```
// C#
interface Expandable {
    void Compress(Point p);
    void Expand(Point p);
}
class ClassifierNode
: Expandable {
    void Compress(Point p) {...}
    void Expand(Point p) {...}
}
```

2.3. Отношение ассоциации

- Отношения обобщения
- Отношения реализации
-  Отношение ассоциации
- Отношение зависимости

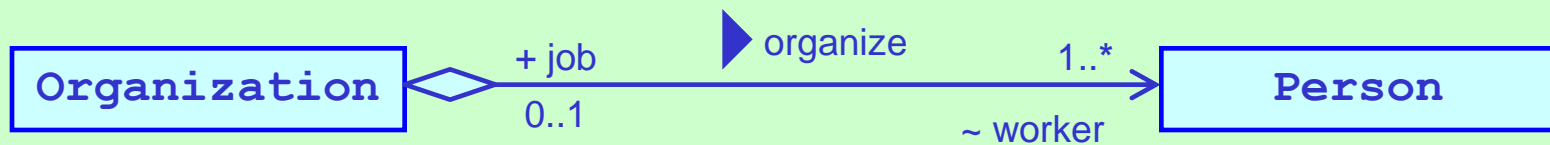
Отношение ассоциации.

Изображение ассоциации и ее свойств

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Отношение ассоциации описывает взаимосвязь между экземплярами классификаторов



organize

Имя ассоциации



Направление чтения ассоциации

job

Имя окончания ассоциации

worker

0..1

Множественность окончания ассоциации

1..*

+

Видимость окончания ассоциации

~



Свойство агрегации окончания ассоциации

Свойство навигации окончания ассоциации

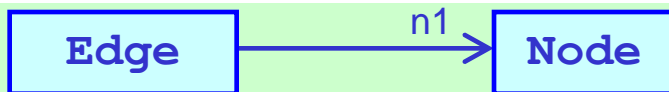


Отношение ассоциации.

Свойство навигации у окончания ассоциации

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Навигация означает возможность одного экземпляра класса ссылаться на другой экземпляр класса.

Навигация показывается стрелкой на окончании ассоциации.

```

// C++
class Node {
};

class Edge {
public:
    Node *n1;
};
  
```



Отсутствие стрелок означает возможность навигации по обоим направлениям.

```

// C++
class Husband {
public:
    Wife *wife;
};

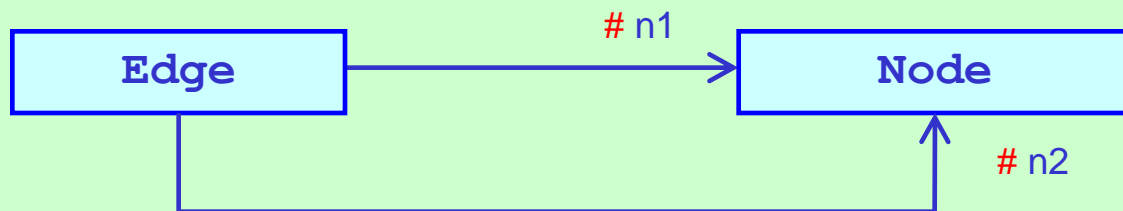
class Wife {
public:
    Husband *husband;
};
  
```

Отношение ассоциации.

Свойство видимости окончания ассоциации

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Разновидности видимости

+ открытая

- скрытая

защищенная

~ пакетная

```
// Java
public class Node {
}
```

```
// C#
public class Node {
}
```

```
// C++
class Node {
};
```

```
// Java
public class Edge {
    protected Node n1;
    protected Node n2;
}
```

```
public class Edge {
    protected Node n1;
    protected Node n2;
}
```

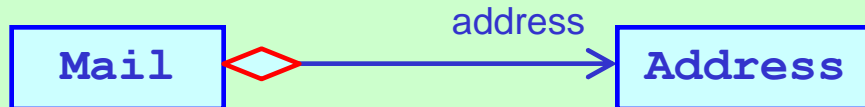
```
class Edge {
    protected:
        Node *n1;
        Node *n2;
};
```

Отношение ассоциации.

Свойство агрегации и композиции

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



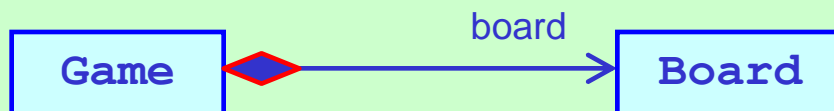
Агрегация описывает отношение часть-целое.

Время жизни части и целого могут не совпадать.

```

// C#
public class Address {
}

public class Mail {
    public Address address;
}
  
```



Композиция означает владение одного экземпляра класса другим экземпляром класса.

Время жизни собственника и собственности совпадают.

```

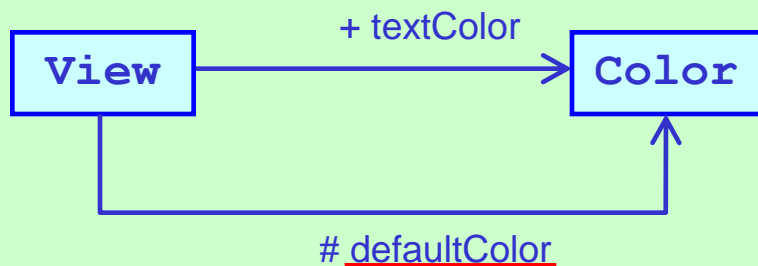
// C++
class Game {
public:
    Board *board;
    Game ()
    { board = new Board(); }
    ~Game ()
    { delete board; }
};
  
```

Отношение ассоциации.

Область действия (scope) окончания ассоциации

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Если область действия окончания ассоциации – класс, то оно доступно всем экземплярам класса.

Окончания с областью действия – класс подчеркнуты.

```

// C#
public class View {
    public Color textColor;

    static
    protected Color defaultColor;
}
  
```

```

// Java
public class View {
    public Color textColor;

    static
    protected Color defaultColor;
}
  
```

```

// C++
class View {
public:
    Color *textColor;

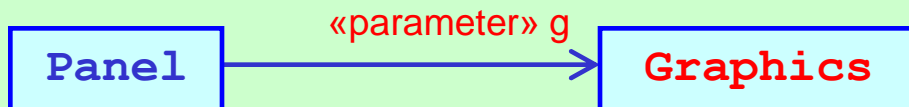
protected:
    static
    Color *defaultColor;
};
  
```

Отношение ассоциации.

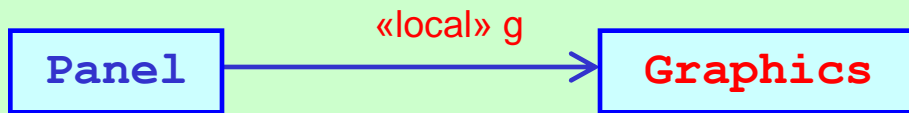
Стереотипы окончания ассоциации

МГУ им. М.В.Ломоносова. Факультет ВМК.

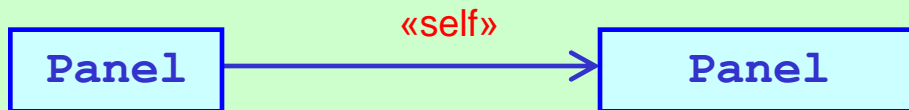
Романов Владимир Юрьевич ©2024



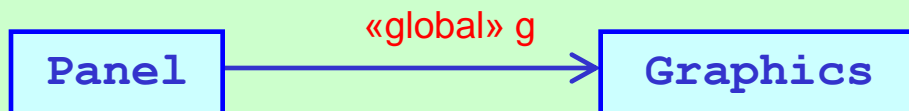
```
// C#
public class Panel {
    public void draw(Graphics g)
    {...}
}
```



```
// C#
public class Panel {
    public void draw()
    { Graphics g; }
}
```



```
// C#
public class Panel {
    public void draw()
    { this.expand(); }
}
```



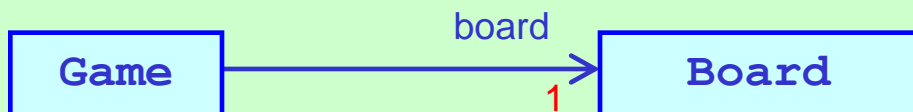
```
// C++
Graphics g;
class Panel {
    public void draw()
    { g.update(); }
};
```

Отношение ассоциации.

Множественность окончания ассоциации (1)

МГУ им. М.В.Ломоносова. Факультет ВМК.

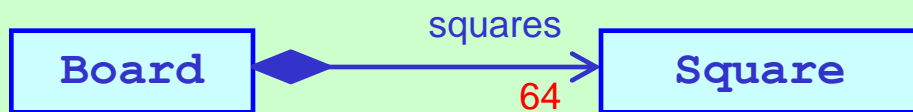
Романов Владимир Юрьевич ©2024



```
// C#
public class Game {
    public Board board;
    public Game(Board b)
        { board = b; }
}
```



```
// C#
public class Square {
    public Piece piece;
    public Square()
        { }
}
```



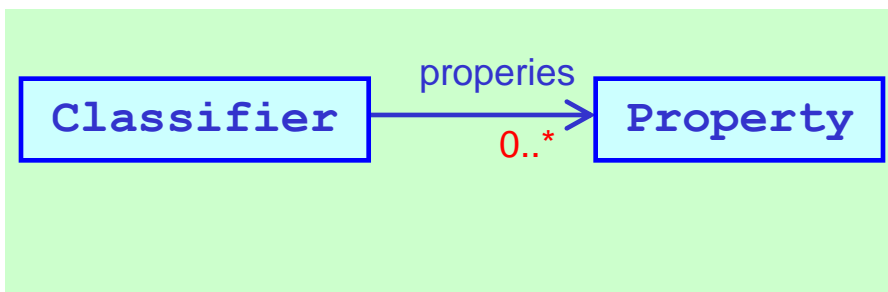
```
// C#
public class Board {
    public Square[] squares;
    public Board()
        { squares = new Square[64]; }
}
```

Отношение ассоциации.

Множественность окончания ассоциации (2)

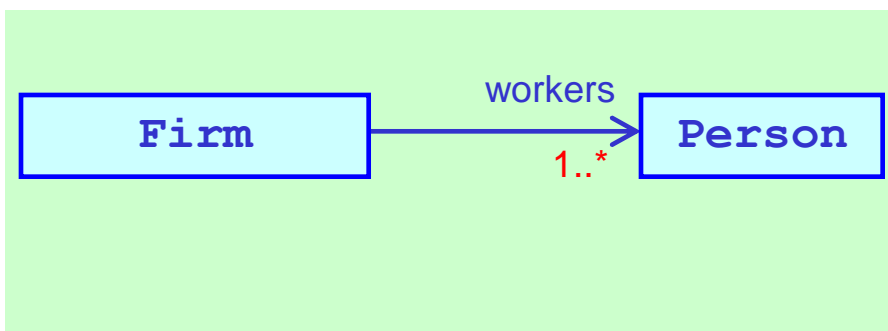
МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



```

// C#
using System.Collections.Generic;
public class Classifier {
    public ISet<Property> properties;
    public Classifier() {
        properties = new HashSet<Property>();
    }
}
  
```



```

// C#
using System.Collections.Generic;
public class Firm {
    public ISet<Person> workers;
    public Firm(Person chief) {
        workers = new HashSet<Person>();
        workers.Add(chief);
    }
}
  
```

```

// C++
#include <set>
using std::set;

class Firm {
public:
    set<Person*> workers;
    Firm(Person *chief) {
        workers.insert(chief);
    }
};
  
```

```

// Java
import java.util.*;

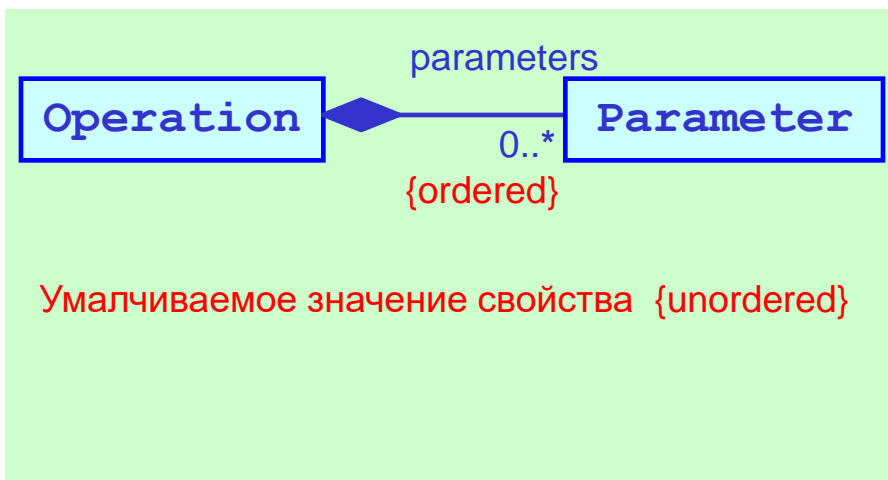
public class Firm {
    public Set<Person> workers;
    public Firm(Person chief) {
        workers = new HashSet<Person>();
        workers.add(chief);
    }
}
  
```

Отношение ассоциации.

Упорядоченность окончания ассоциации

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



```

// Java
public class Operation {
    public List<Parameter> parameters;

    public Operation() {
        parameters =
            new ArrayList<Parameter>();
    }

    public void addParameter(Parameter p) {
        if (parameters.Contains(p)) return;
        parameters.Add(p);
    }
}
  
```

```

// C#
using System.Collections.Generic;
public class Operation {
    public IList<Parameter> parameters;
    public Operation() {
        parameters = new List<Parameter>();
    }
    public void addParameter(Parameter p) {
        if (parameters.Contains(p)) return;
        parameters.Add(p);
    }
}
  
```

```

// C++
#include <vector>
#include <algorithm>
using namespace std;

class Operation {
public:
    vector<Parameter> ps;
    void addParameter(Parameter *p);
};

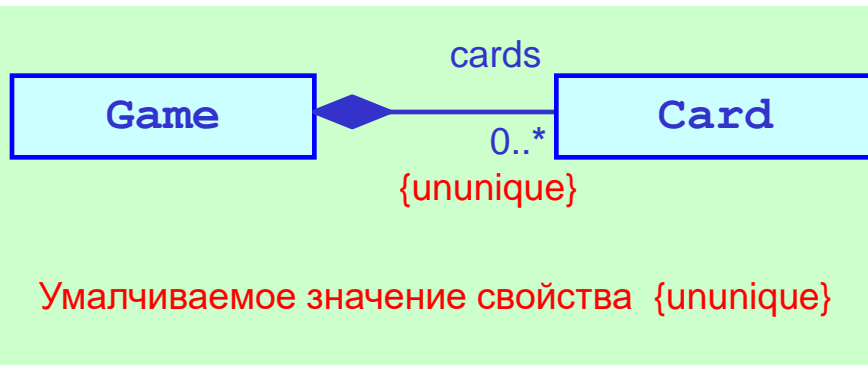
void Operation::addParameter(Parameter *p) {
    if (find(ps.begin(), ps.end(), p) != ps.end())
        return;
    ps.push_back(p);
}
  
```


Отношение ассоциации.

Уникальность окончания ассоциации

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



```

using System.Collections.Generic;

public class Game {
    public IList<Card> cards;
    public Game() {
        cards = new List<Card>();
    }
}

```

```

// Java
import java.util.*;

public class Game {
    public List<Card> cards;
    public Game() {
        parameters =
            new ArrayList<Card>();
    }
}

```

```

// C++
#include <vector>
using namespace std;

class Game {
public:
    vector<Card> cards;
};

```

Отношение ассоциации.

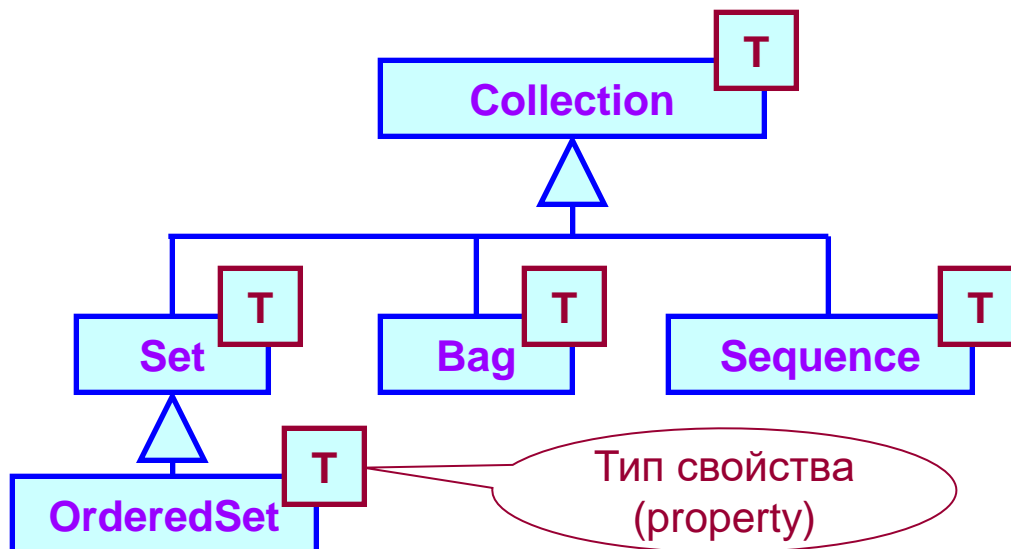
Уникальность, упорядоченность и язык OCL

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

OSL коллекции

	{unique}	{ununique}
{unordered}	Set	Bag
{ordered}	OrderedSet	Sequence



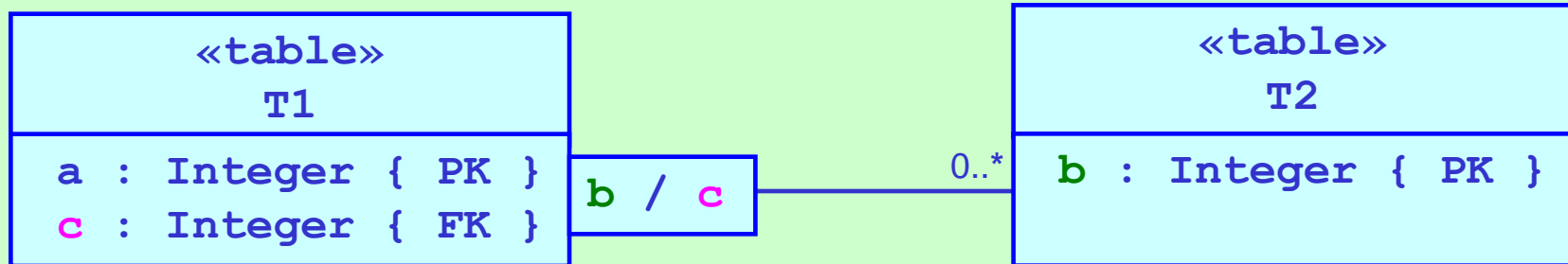
Отношение ассоциации.

Квалификатор окончания ассоциации

МГУ им. М.В.Ломоносова. Факультет ВМК.

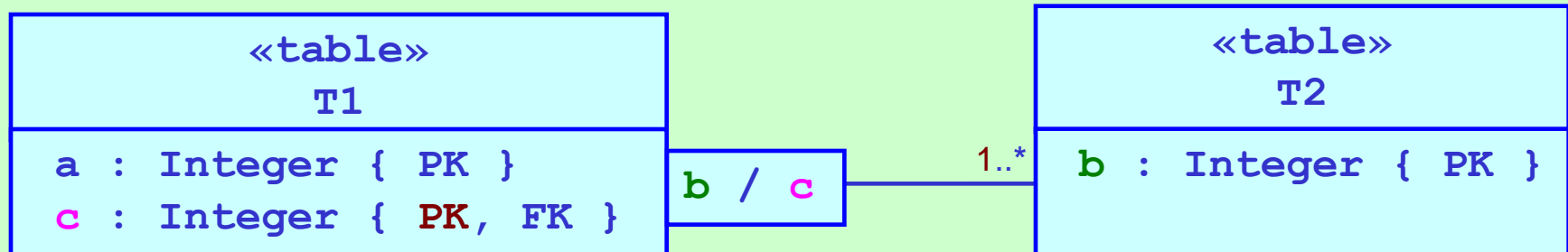
Романов Владимир Юрьевич ©2024

Описание миграции первичных ключей и превращения их во внешние ключи



`b` – первичный ключ {PK} таблицы T2 мигрирует в таблицу T1 и становится внешним ключом {FK} таблицы T1 с именем `c`.

Описание миграции первичных ключей и превращения их в первичные внешние ключи



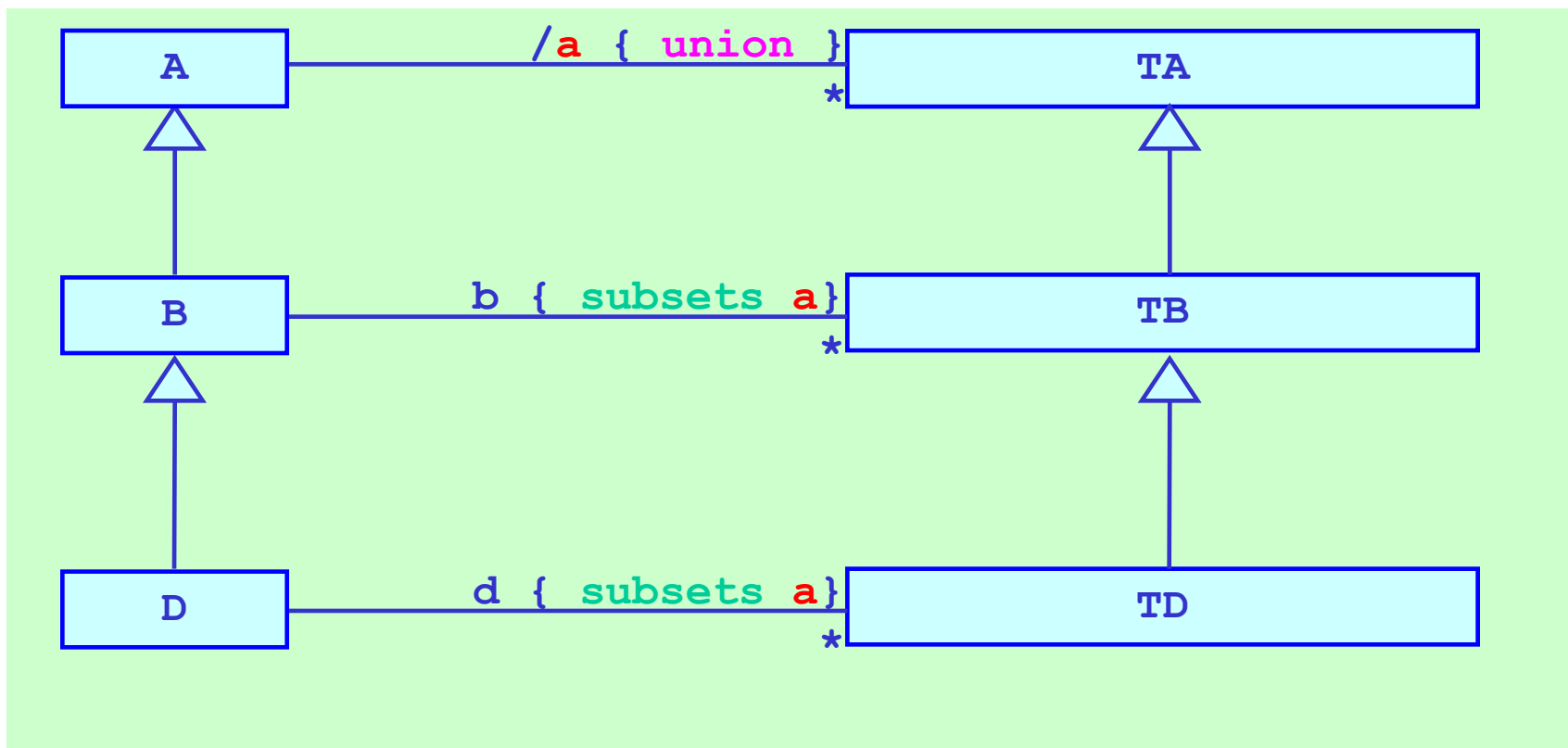
`b` – первичный ключ {PK} таблицы T2 мигрирует в таблицу T1 и становится первичным внешним ключом {FK} таблицы T1 с именем `c`.

Отношение ассоциации.

Объединения и подмножества в ассоциации

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



```

// Java
B x = new D();
T[] y = x.getA();
// y = b + d
  
```

```

// Java
B x = new B();
T[] y = x.getA();
// y = b
  
```

```

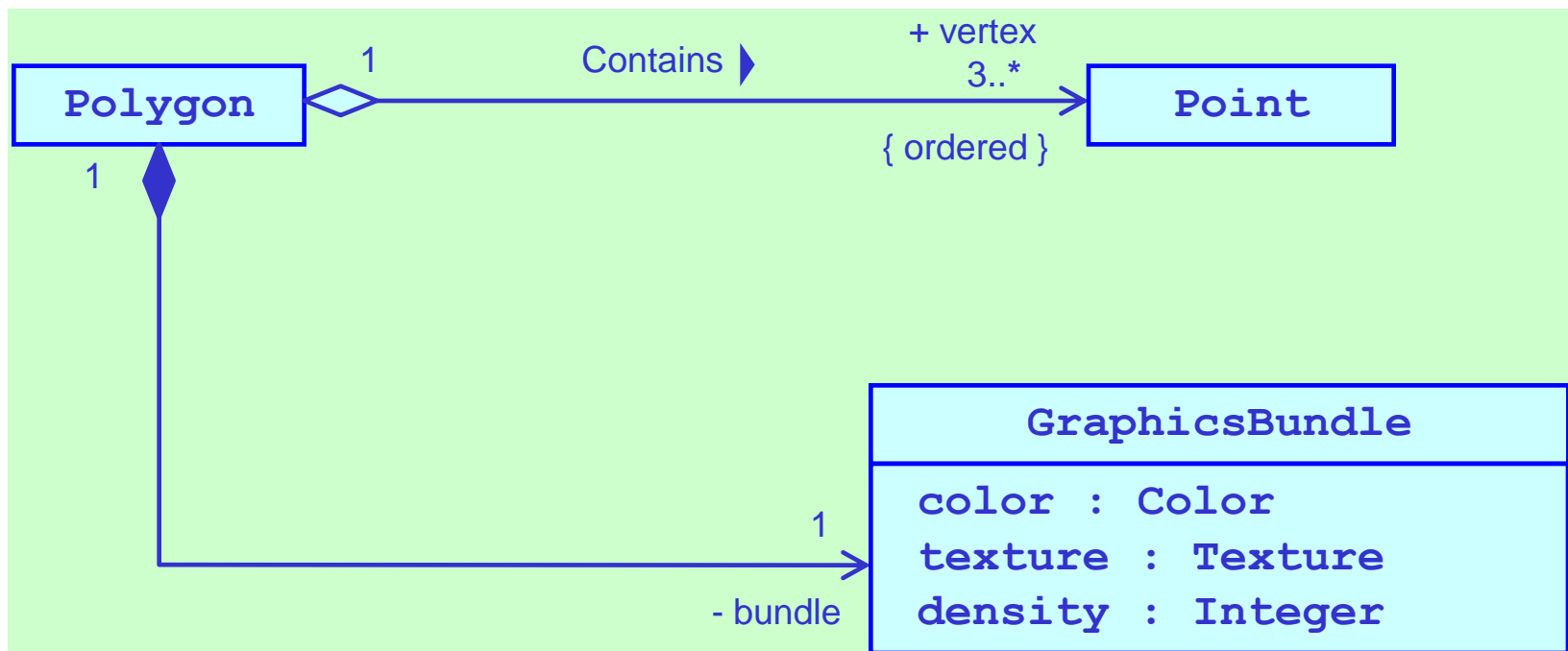
// Java
A x = new A();
T[] y = x.getA();
// y = {}
  
```

Отношение ассоциации.

Пример свойств окончания ассоциации

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

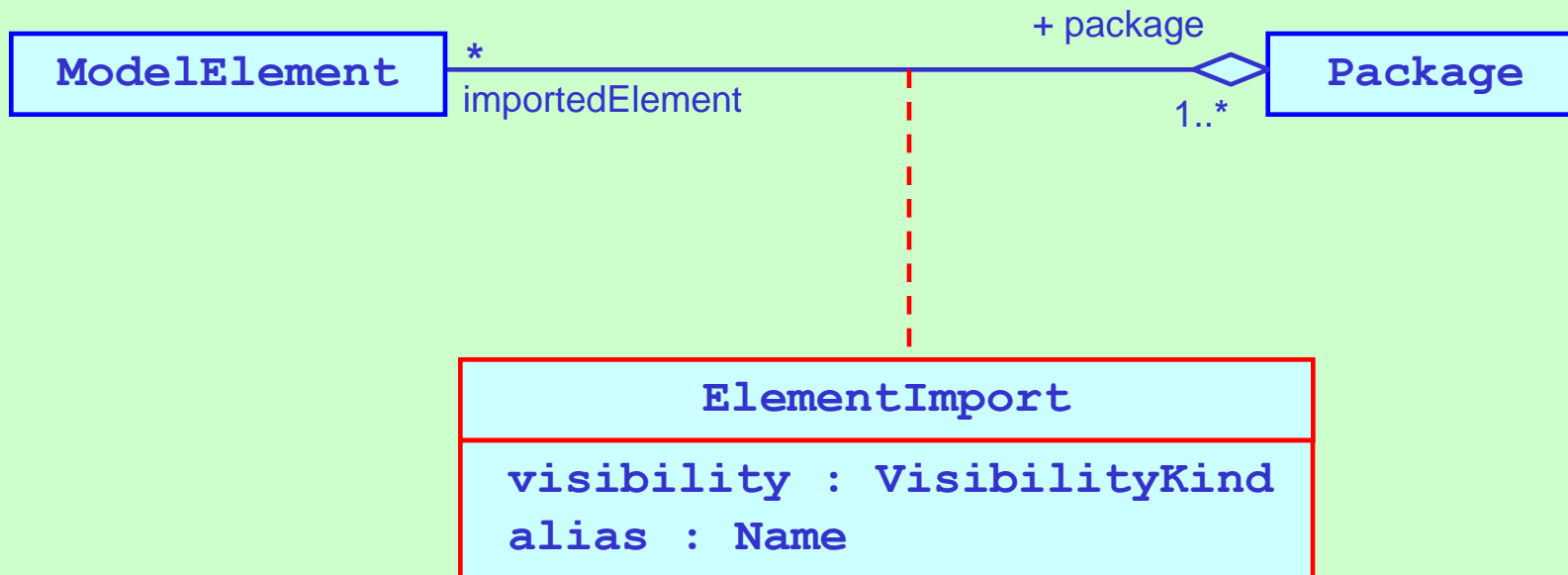


Отношение ассоциации.

Ассоциация - класс

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

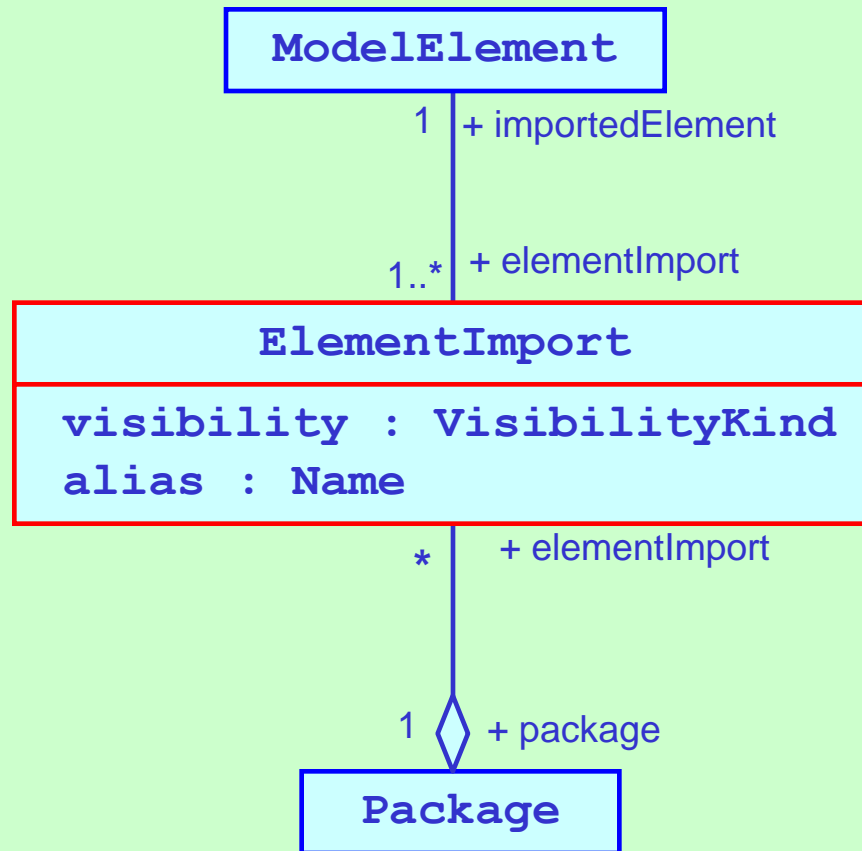


Отношение ассоциации.

Упрощение ассоциации - класса

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

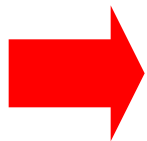


2.4 Отношение зависимости

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- Отношения обобщения
- Отношения реализации
- Отношение ассоциации
- Отношение зависимости

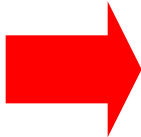


Отношение зависимости.

Категория зависимости “связывание”

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

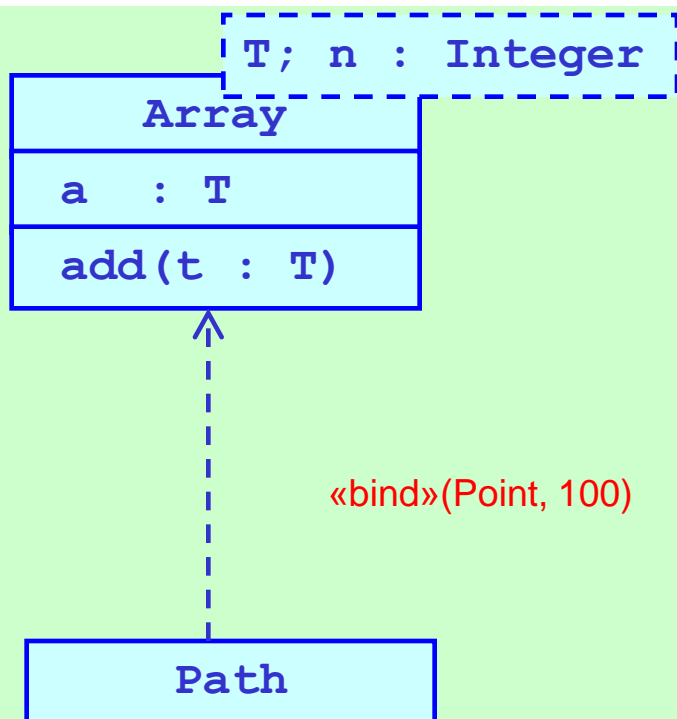
- 
- Категория зависимости связывание
 - Категория зависимости абстракция
 - Категория зависимости использование
 - Категория зависимости разрешение

Отношение зависимости.

Изображение зависимости связывание

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Отношение зависимости связывание

```

// C++
template <class T, int n>
class Array {
public:
    T a;
    void add(T t);
};
  
```

```

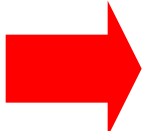
// C++
Array<Point, 100> path;
  
```

Отношение зависимости.

Категория зависимости “абстракция”

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- Категория зависимости связывание
-  • Категория зависимости абстракция
- Категория зависимости использование
- Категория зависимости разрешение

Отношение зависимости.

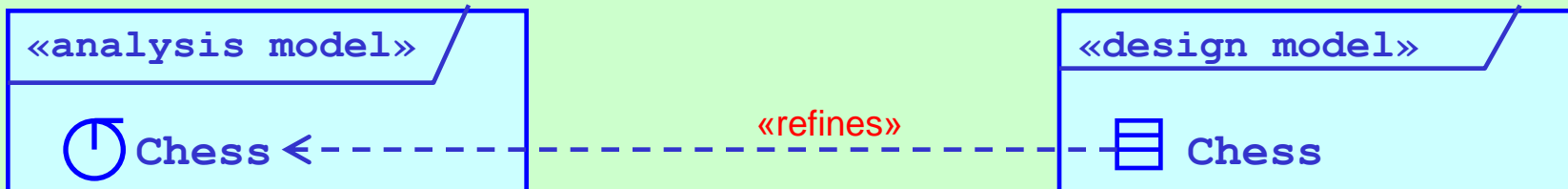
Изображение зависимостей абстракции

МГУ им. М.В.Ломоносова. Факультет ВМК.

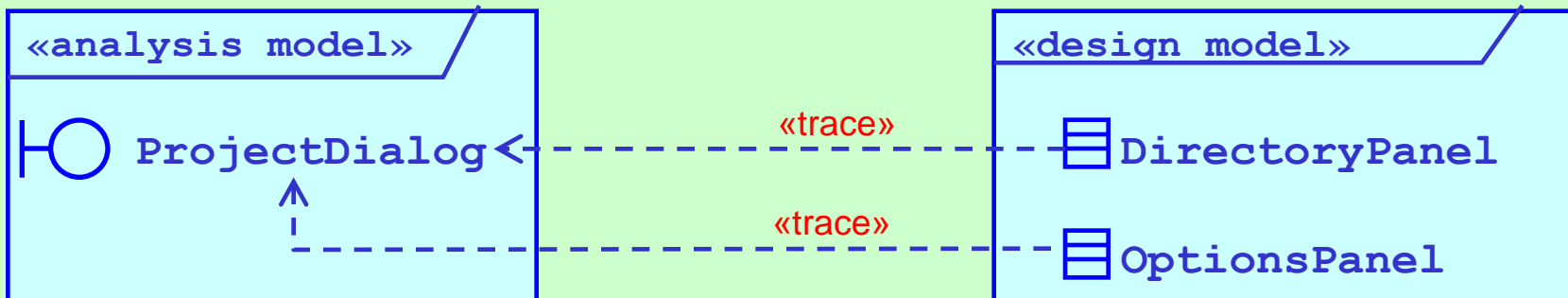
Романов Владимир Юрьевич ©2024



Класс Chess из зависящей от языка реализации модели проектирования уточняет (refines) класс Chess из независящей от языка реализации модели анализа



Классы DirectoryPanel и OptionsPanel были созданы из модели проектирования были созданы из спецификации класса ProjectDialog модели анализа

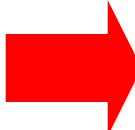


Отношение зависимости.

Категория зависимости “использование”

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- Категория зависимости связывание
- Категория зависимости абстракция
-  • Категория зависимости использование
- Категория зависимости разрешение

Отношение зависимости.

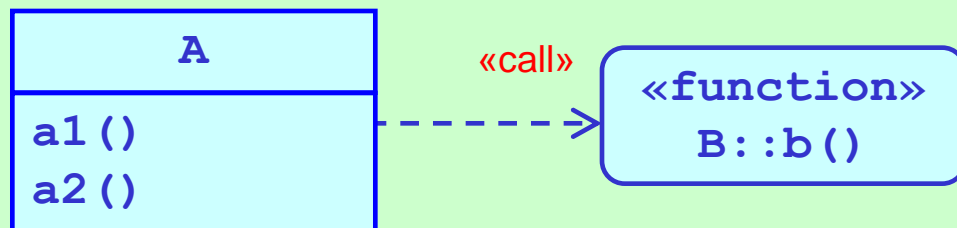
Изображение зависимости “вызов функции”

МГУ им. М.В.Ломоносова. Факультет ВМК.

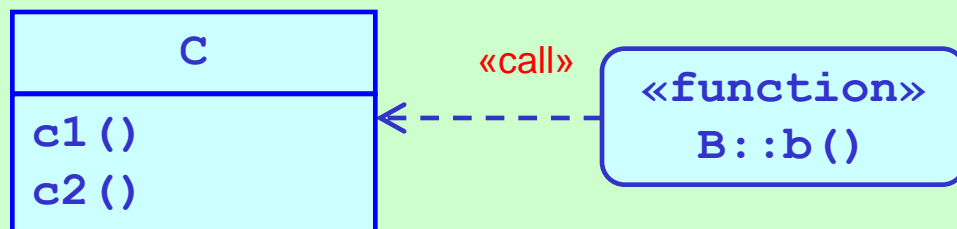
Романов Владимир Юрьевич ©2024



Представление графа вызова с помощью отношения зависимости. Порядок вызова не важен.



Представление всех функций классов вызывающих функцию B::g().



Представление всех функций классов вызываемых функцией B::g().

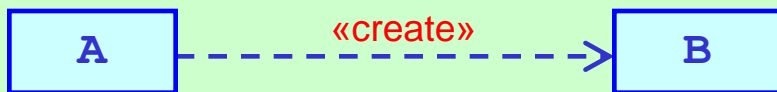
Изображения зависимости вызова могут быть полезны, например, как комментарии поясняющие вычисленные объектно-ориентированные метрики для функций программы.

Отношение зависимости.

Изображение зависимости "создание экземпляра"

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

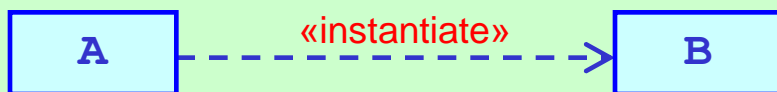


Создание экземпляра класса A приводит к созданию экземпляра класса B

```
// C#
public class A : B {
    public A(string s) : base(s) {}
}
```

```
// Java
public class A extends B {
    public A(string s) { base(s); }
}
```

```
// C++
class A : public B {
public:
    A(std::string s) : B(s) {}
}
```



Вызов методов класса A приводит к созданию экземпляра класса B

```
// C#
public class A : B {
    public B f(string s)
        { return new B(s); }
}
```

```
// Java
public class A extends B {
    public B f(string s)
        { return new B(s); }
}
```

```
// C++
class A : public B {
public:
    B f(std::string s)
        { return new B(s); }
}
```

Отношение зависимости.

Изображение зависимости "посылка сигнала"

МГУ им. М.В.Ломоносова. Факультет ВМК.

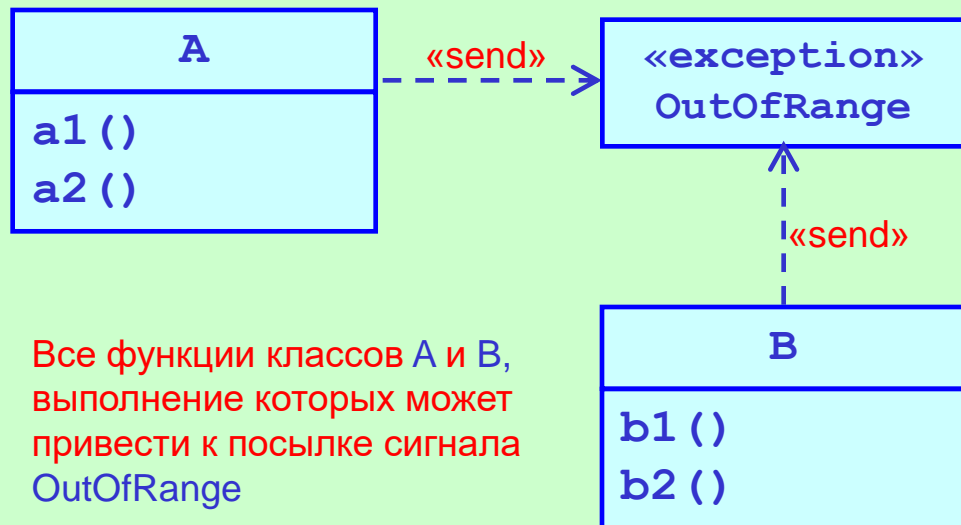
Романов Владимир Юрьевич ©2024



Выполнение функции A::f() может привести к посылке сигнала OutOfRange

```
// Java
class OutOfRange
extends Exception
{}

class A {
    void f() throws OutOfRange
    {}
}
```



Все функции классов A и B, выполнение которых может привести к посылке сигнала OutOfRange

```
// Java
class OutOfRange
extends Exception
{}

class A {
    void a1() throws OutOfRange {}
    void a2() throws OutOfRange {}
}

class B {
    void b1() throws OutOfRange {}
    void b2() throws OutOfRange {}
}
```

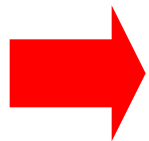

Отношение зависимости.

Категория зависимости “разрешение”

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- Категория зависимости связывание
- Категория зависимости абстракция
- Категория зависимости использование
- Категория зависимости разрешение



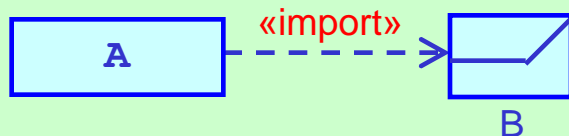
Отношение зависимости.

Изображение зависимости "импорт"

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Отношение между пространствами имен. Содержимое пространства имен поставщика полностью или частично копируется в пространство имен клиента.



Импорт всего пространства имен **B**.

```
//Java
import B.*;

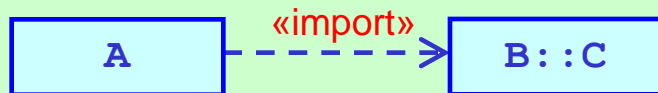
public class A {}
```

```
// C++
#include "B.h"
using namespace B;

class A {};
```

```
// C#
using B;

public class A {}
```



Импорт класса **C** из пространства имен **B**.

```
//Java
import B.C;

public class A {}
```

```
// C++
#include "B.h"
using B::C;

class A {};
```

```
// В C# импорт класса возможен
// только с новым именем-псевдонимом.
using C_from_B = B.C;

public class A {}
```

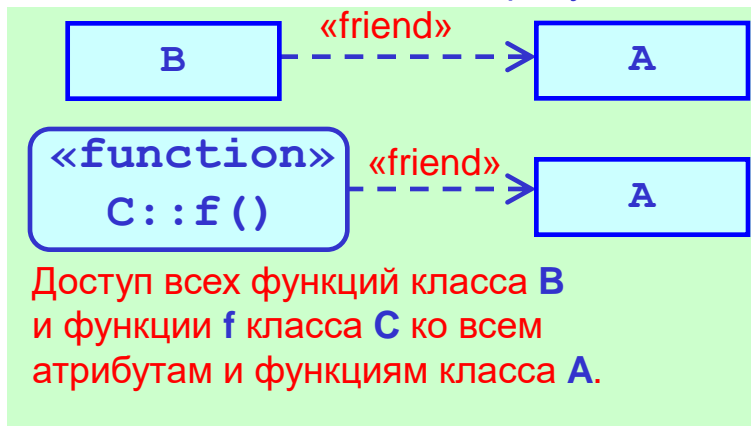
Отношение зависимости.

Изображение зависимости "друг"

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Клиенту отношения дается доступ ко всем атрибутам и методов класса-поставщика независимо от типа видимости атрибутов и методов.



```
// C++
class B, C;
class A {
    friend B;
    friend C.f;
private:    int x;
protected: int y;
};
```

```
class C {
    A a;

    void f() {
        a.x = 1; // Допустимо.
        a.y = 2; // Допустимо.
    }
    void g() {
        a.x = 1; // Не допустимо.
        a.y = 2; // Не допустимо.
    }
};
```

```
class B {
    A a;

    void f() {
        a.x = 1; // Допустимо.
        a.y = 2; // Допустимо.
    }
    void g() {
        a.x = 1; // Допустимо.
        a.y = 2; // Допустимо.
    }
};
```

Диаграммы коммуникации объектов

2. Диаграмма коммуникации. Назначение диаграммы

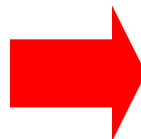
- Диаграмма коммуникации (Collaboration diagram) показывает экземпляры классов (объекты) и связи между ними.
- Отношение связи – экземпляр отношения ассоциации, представляемой на диаграмме статической структуры.
- Совокупность объектов и связей между ними – “Сеть” для обмена объектов сообщениями.
- Достоинство диаграммы коммуникации – показываются причины возникновения связей между объектами.
- Последовательность рассылки сообщений более наглядно показывают диаграммы последовательности взаимодействия (Sequence diagram)

Диаграмма коммуникации.

2.1 Изображение объектов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



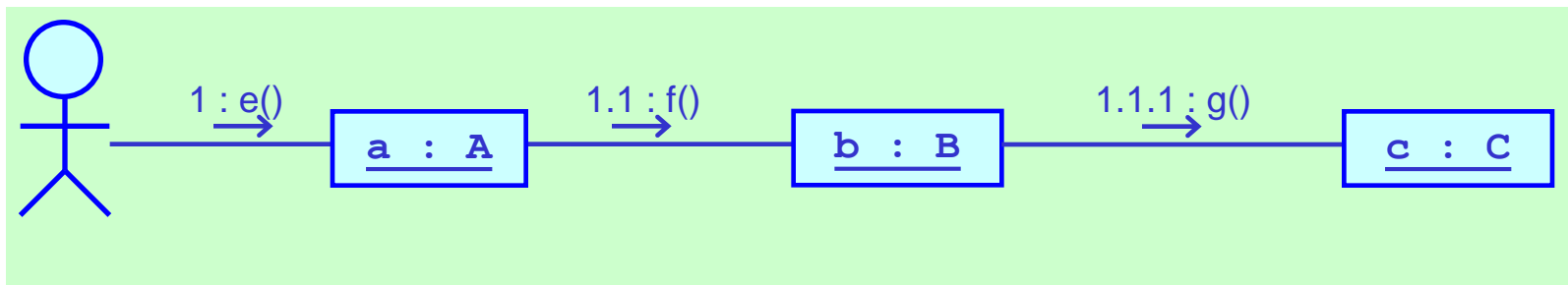
- Изображения объектов
- Отношения связи между объектами
- Рассылка сообщений объектами

Диаграмма коммуникации.

Простой пример диаграммы коммуникации

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



```
class A {
    public B b;

    public void e() {
        b.f();
    }
}
```

```
class B {
    public C c;

    public void f() {
        c.g();
    }
}
```

```
class C {
    public void g() {
    }
}
```

Изображение объектов.

Контекст и множественность объекта

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Имя объекта

Квалифицированное
имя типа объекта

theGraph : CASETool::Diagrams::Graph

Экземпляры классификаторов
показываются подчеркнутыми

theGraph : Diagrams::Graph

Изображение объекта theGraph
в контексте пакета CASETools

theGraph : Graph

Изображение объекта theGraph
в контексте пакета CASETools::Diagrams

p : Point

Изображение множества объектов

Изображение объектов.

Стереотипы, атрибуты и состояния объектов

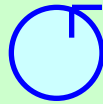
МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

s : Sorter

«control»

s : Sorter



s : Sorter

Способы изображения
стереотипов объектов.

worker : Person

name : String = "Ann"

age : Integer = 32

Изображение атрибутов объекта

movedNode : Node

[selected]

Изображение состояния объекта

movedNode : Node

[sourceSelection, targetSelection]

Изображение вложенных
состояний объекта

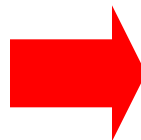
Диаграмма коммуникации.

2.2 Отношение связи между объектами

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- Изображения объектов



- Отношения связи между объектами
- Рассылка сообщений объектами

Отношение связи между объектами.

Стереотипы отношения связи (1)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

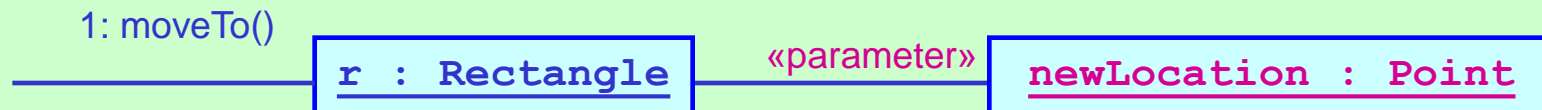
Умалчиваемый стереотип «association» используется для представления полей класса.



Имя умалчиваемого стереотипа «association» чаще всего отсутствует!!!

```
// Java, C#
class Rectangle {
    public Point origin;
}
```

Стереотип «parameter» используется для представления объектов-параметров



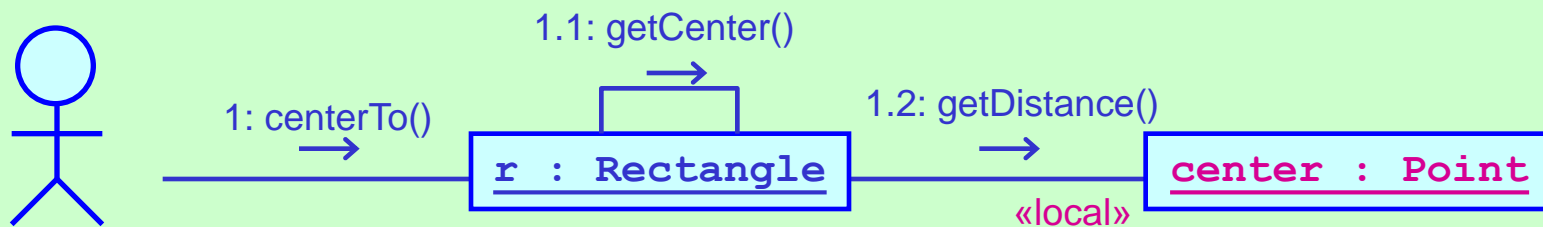
```
// Java, C#
class Rectangle {
    public void moveTo(Point newLocation) {
    }
}
```

Отношение связи между объектами.

Стереотипы отношения связи (2)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Умалчиваемый стереотип «local» используется для представления локальных переменных.

```
// Java, C#
class Rectangle {
    public void centerTo(Point p) {
        Point center = getCenter();
        int d = center.getDistance();
    }
}
```

Стереотип «global» используется для представления глобальных переменных.



```
// C++
Color default;

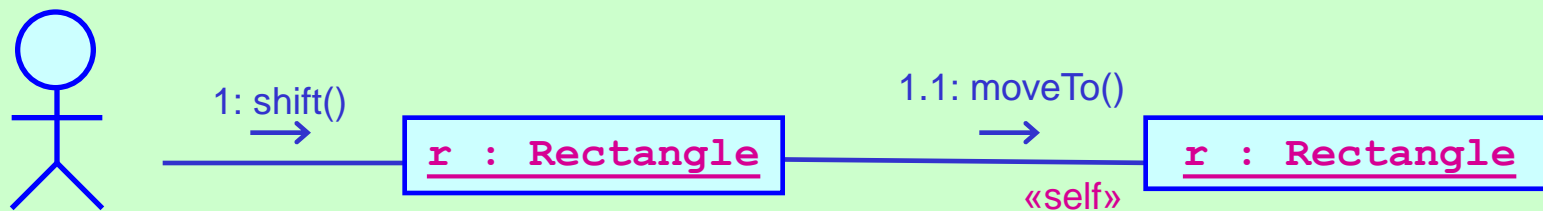
class Diagram {
};
```

Отношение связи между объектами.

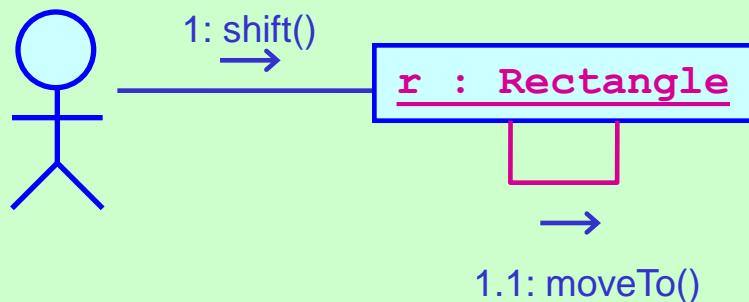
Стереотипы отношения связи (3)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Стереотип «self» используется при большом количестве сообщений, которые объект посылает самому себе.



Изображение посылки сообщения самому, показанное с помощью отношения связи объекта с самим собой.

```

// Java, C#
class Rectangle {
    public void shift() {
        moveTo();
    }
}
  
```

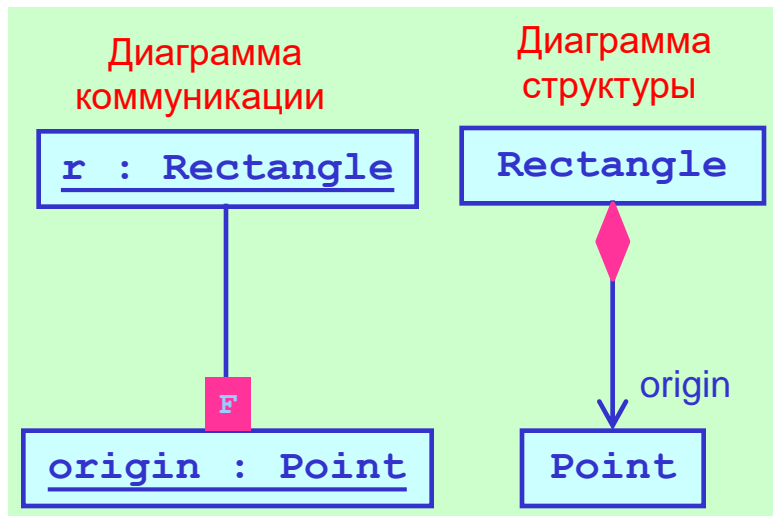
Отношение связи между объектами.

Стереотипы отношения связи (4)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

	Ссылка	Значение		Ссылка	Значение
<<association>> (field)	F	F	<<global>>	G	G
<<parameter>>	P	P	<<self>>	S	S
<<local>>	L	L			



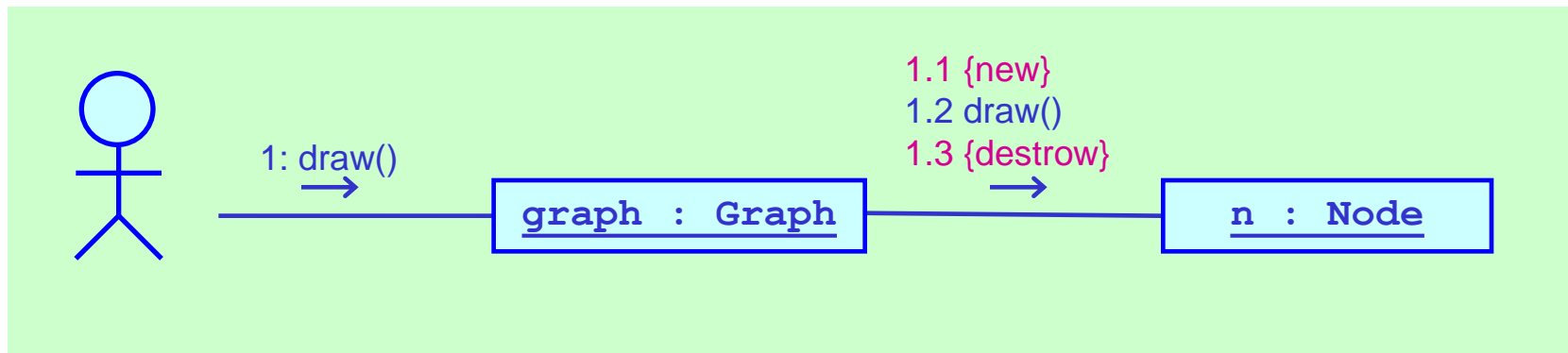
```
// C++
class Rectangle {
public:
    Point origin;
};
```

Переменная origin хранит значение

Отношение связи между объектами. Задание времени жизни объекта

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



```
// C++
class Graph {
public:
    void draw() {
        Node *n = new Node();
        n->draw();
        delete n;
    };
};
```

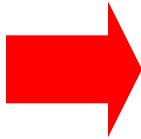
Диаграмма коммуникации.

2.3 Рассылка сообщений объектами

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- Изображения объектов
- Отношения связи между объектами
- Рассылка сообщений объектами



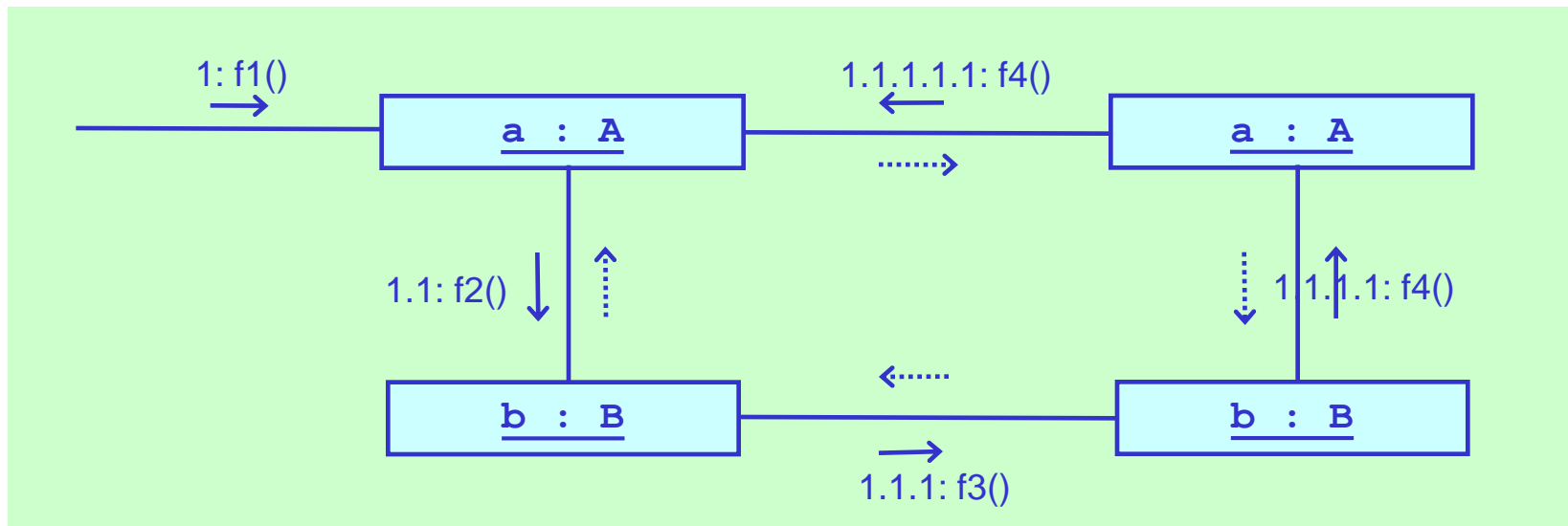
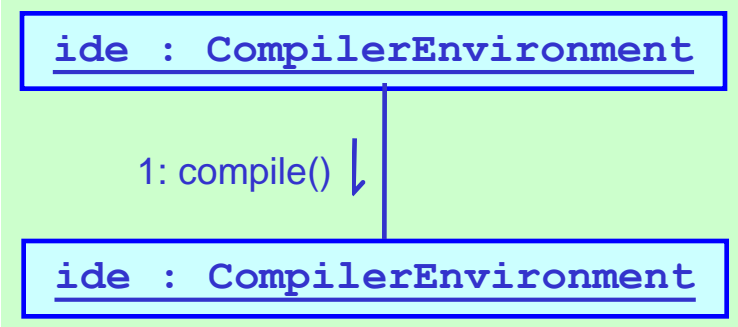
Рассылка сообщений объектами.

Синхронизация рассылаемых сообщений

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- **Процедурный вызов либо посылка сообщения с ожиданием завершения реакции на сообщение.**
- **Асинхронный вызов или посылка сообщения**
-→ **Возврат из процедуры. Может быть показан если это не ясно из контекста**



Рассылка сообщений объектами.

Синтаксис посылки сообщения (1)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Предшественник

Условие-выполнения

Выражение-последовательности

Возвращаемое-значение

:=

имя-сообщения

список-аргументов

Предшественник (predecessor)

Список номеров последовательности сообщений, заканчивающийся символом /
Сообщения из указанного списка должны быть завершены, прежде чем будет послано данное сообщение.

3, 5, 8 / 10 : f()
 →

Сообщение с номером на 1 меньше текущего может не указываться

Предшественники могут отсутствовать

С помощью предшественников на диаграммах коммуникации показывают “сложные переходы” синхронизирующие параллельно выполняющиеся сообщения

Рассылка сообщений объектами.

Синтаксис посылки сообщения (2)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Предшественник

Условие-выполнения

Выражение-последовательности

Возвращаемое-значение

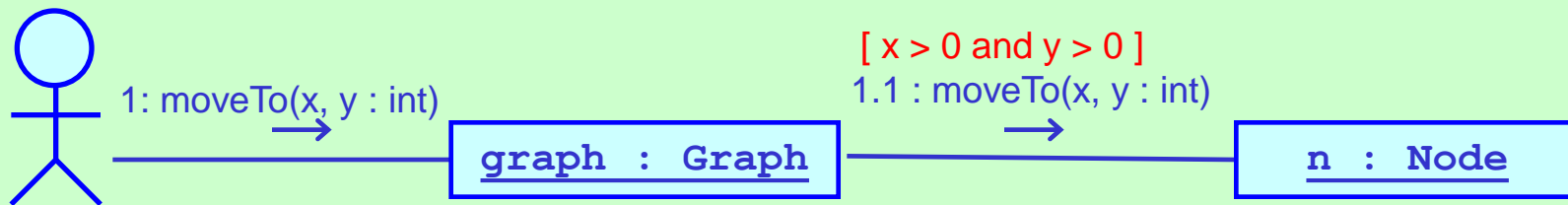
:=

имя-сообщения

список-аргументов

Условие выполнения (guard)

Логическое условие, которое должно быть истинным, прежде чем начнется выполнение посылки данного сообщения.



Рассылка сообщений объектами.

Синтаксис посылки сообщения (3)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Предшественник

Условие-выполнения

Выражение-последовательность

Возвращаемое-значение

:=

имя-сообщения

список-аргументов

Выражение-последовательность

Список термов последовательности, разделенных символом точка и завершаемых символом двоеточие. Может использоваться для отображения вложенности вызовов и параллельности выполнения вызовов.

Выражение-последовательность =

Терм-последовательность ["." Терм-последовательность] ":" .

Терм-последовательность =

[целое | имя] [Повторяемость].

Повторяемость =

"." [Конструкция-итерации].

Рассылка сообщений объектами.

Синтаксис посылки сообщения (4)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Предшественник

Условие-выполнения

Выражение-последовательность

Возвращаемое-значение

:=

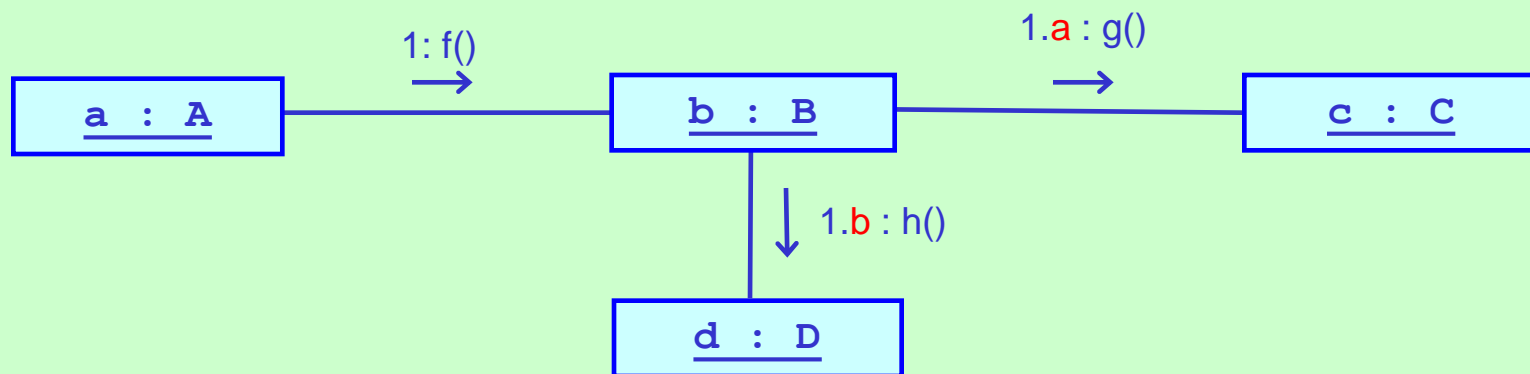
имя-сообщения

список-аргументов

Имя в списке последовательности имеют параллельно выполняющиеся сообщения.

Терм-последовательность =

[целое | **ИМЯ**] [Повторяемость].



Сообщения g и h выполняются параллельно.

Рассылка сообщений объектами. Синтаксис посылки сообщения (5)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

Предшественник

Условие-выполнения

Выражение-последовательность

Возвращаемое-значение

:=

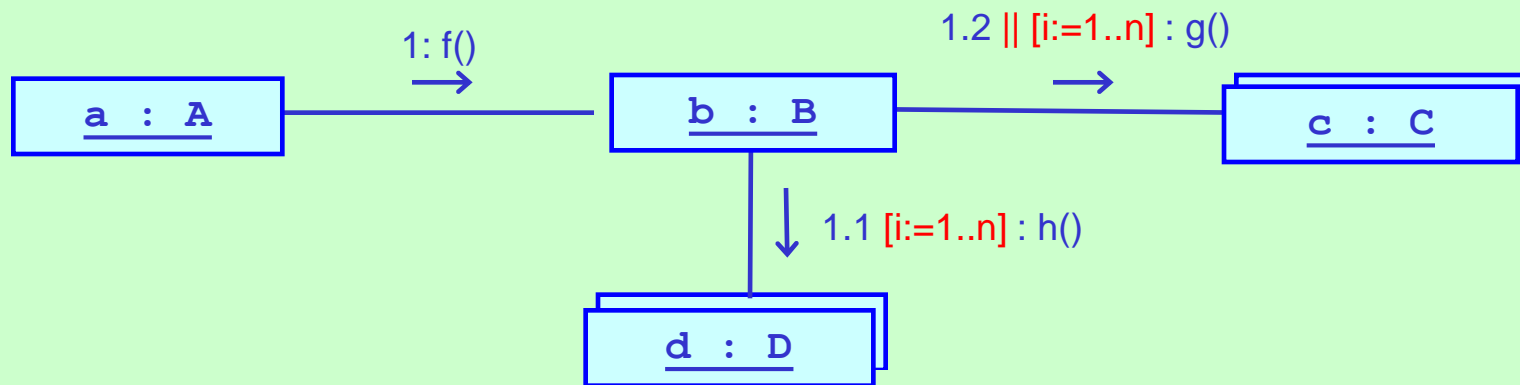
имя-сообщения

список-аргументов

Имя в списке последовательности имеют параллельно выполняющиеся сообщения.

Терм-последовательность =

[целое | имя] [Повторяемость].



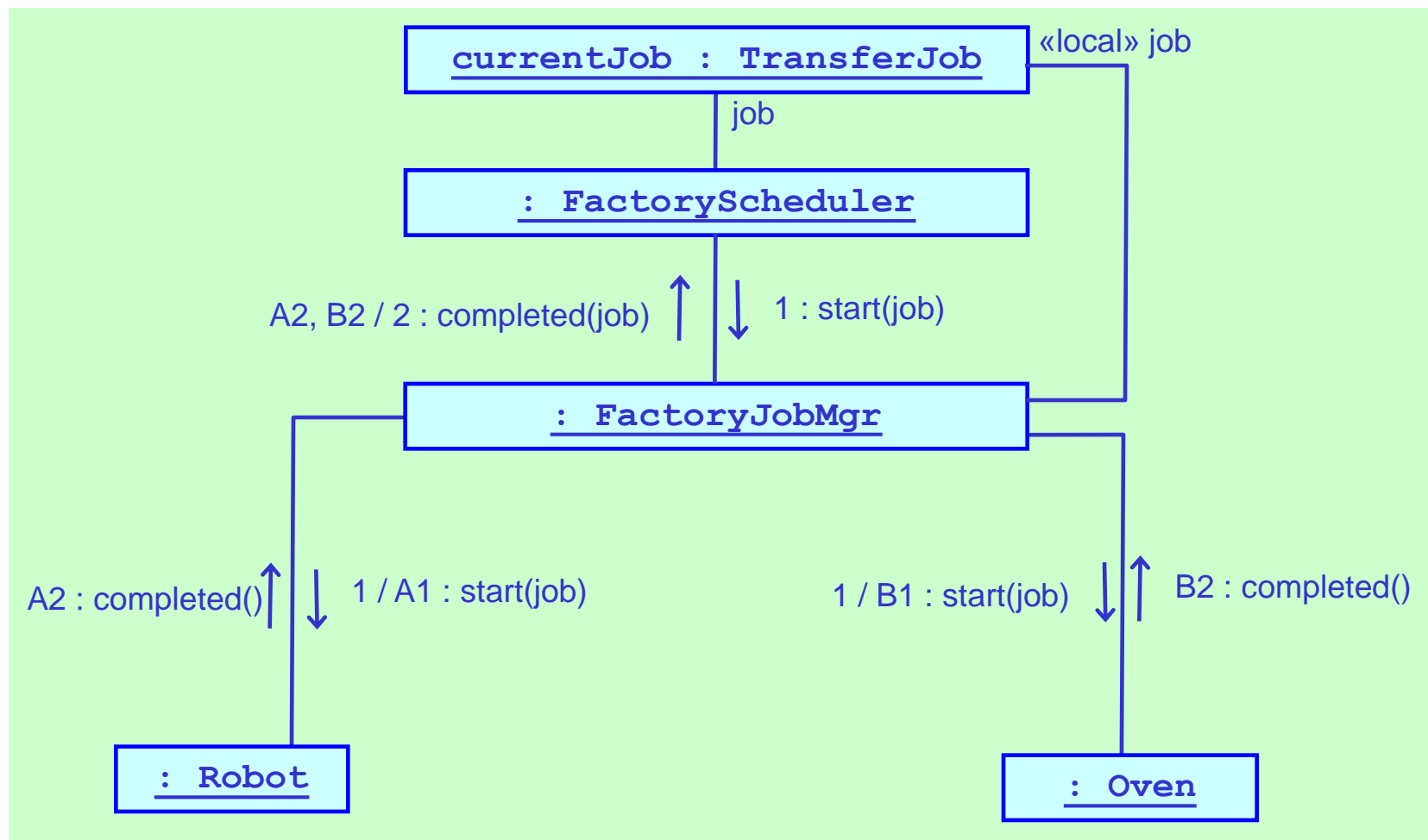
Сообщение h посылается n раз.

Сообщение g посылается n раз, и каждая посылка выполняется параллельно $||$.

Рассылка сообщений объектами. Пример параллельной рассылки сообщений

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Диаграммы последовательности взаимодействия объектов

3. Диаграмма последовательности.

Назначение диаграммы

- Диаграмма последовательности взаимодействия (Sequence diagram) показывает экземпляры классов (объекты) и обмен сообщениями между ними.
- Посылки сообщений упорядочены по оси Y. Это делает более ясным описание сложных сценариев взаимодействия.
- Объекты упорядочены по оси X и имеют линии жизни. Это позволяет более наглядно показать время существования объектов и области активации (стек) возникающие при посылке сообщений.
- Предоставляется графическая нотация для структурирования сложных сценариев взаимодействия. Это позволяет строить модель предложений объектно-ориентированных языков программирования.

Диаграмма последовательности.

Простой пример диаграммы последовательности

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

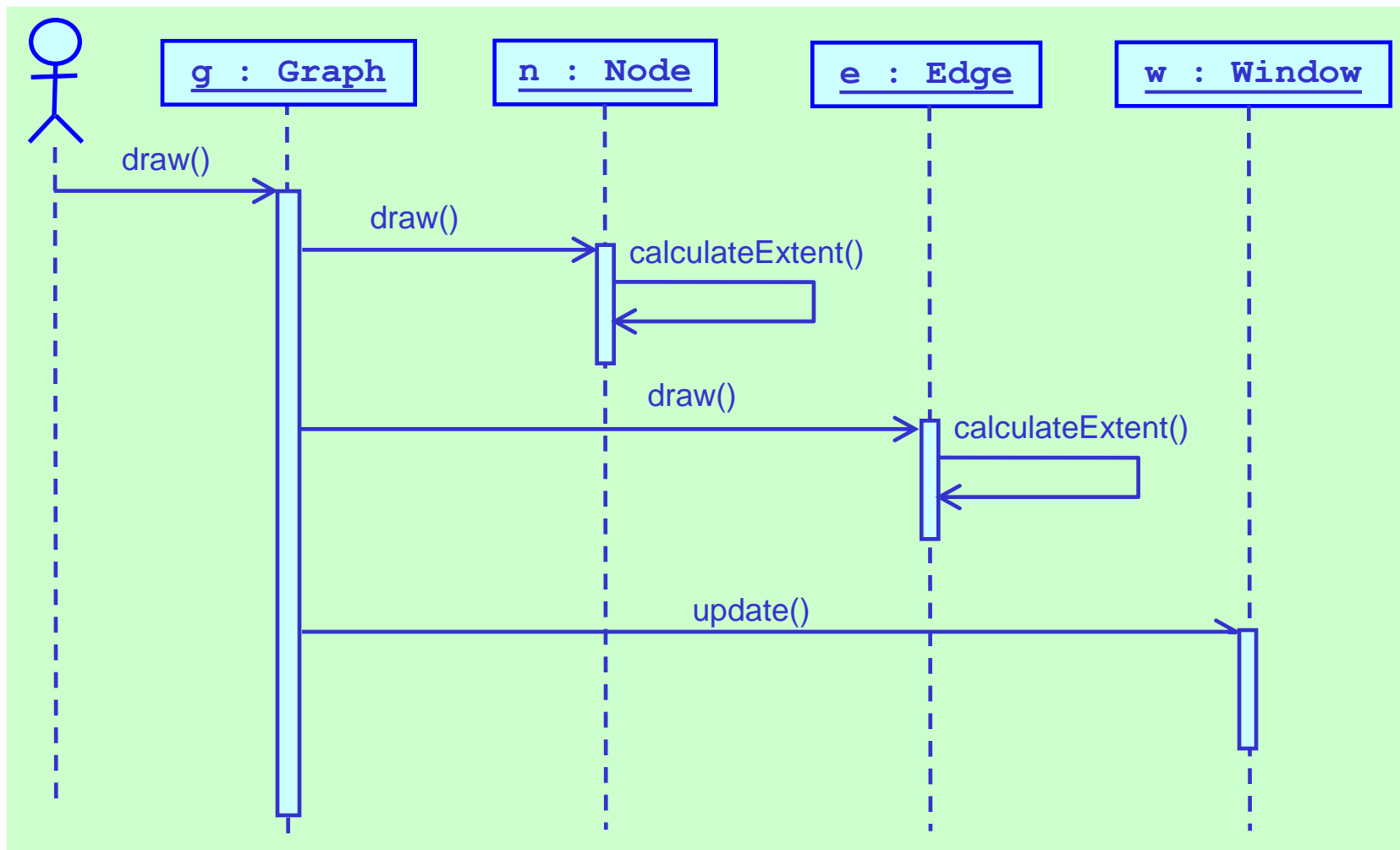
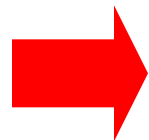


Диаграмма последовательности.

3.1 Изображение объектов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



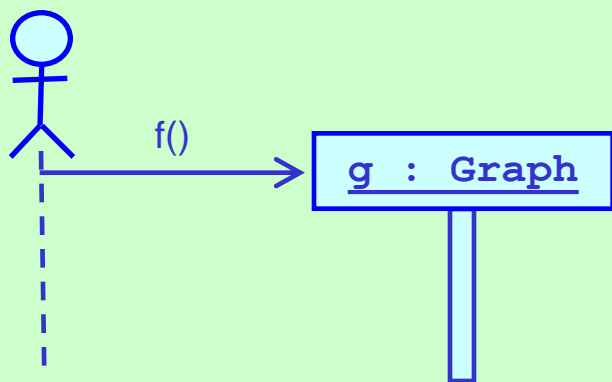
- Изображения объектов
- Рассылка сообщений объектами
- Нотация для структурирования диаграммы последовательности

Изображения объекта.

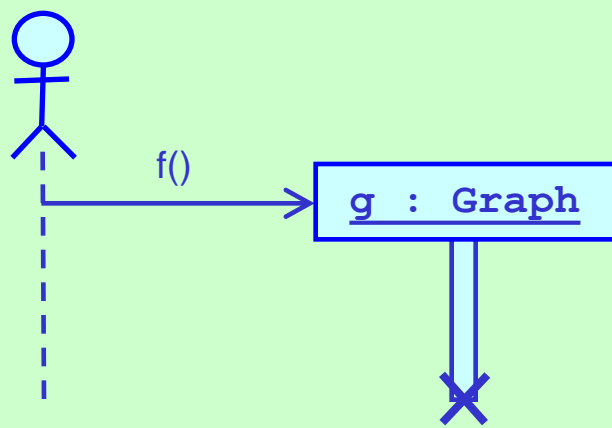
Время существования объекта

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Создание объекта в интервале времени описываемого объектом.



Уничтожение объекта в интервале времени описываемого объектом.

Изображения объекта.

Области активации объекта

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

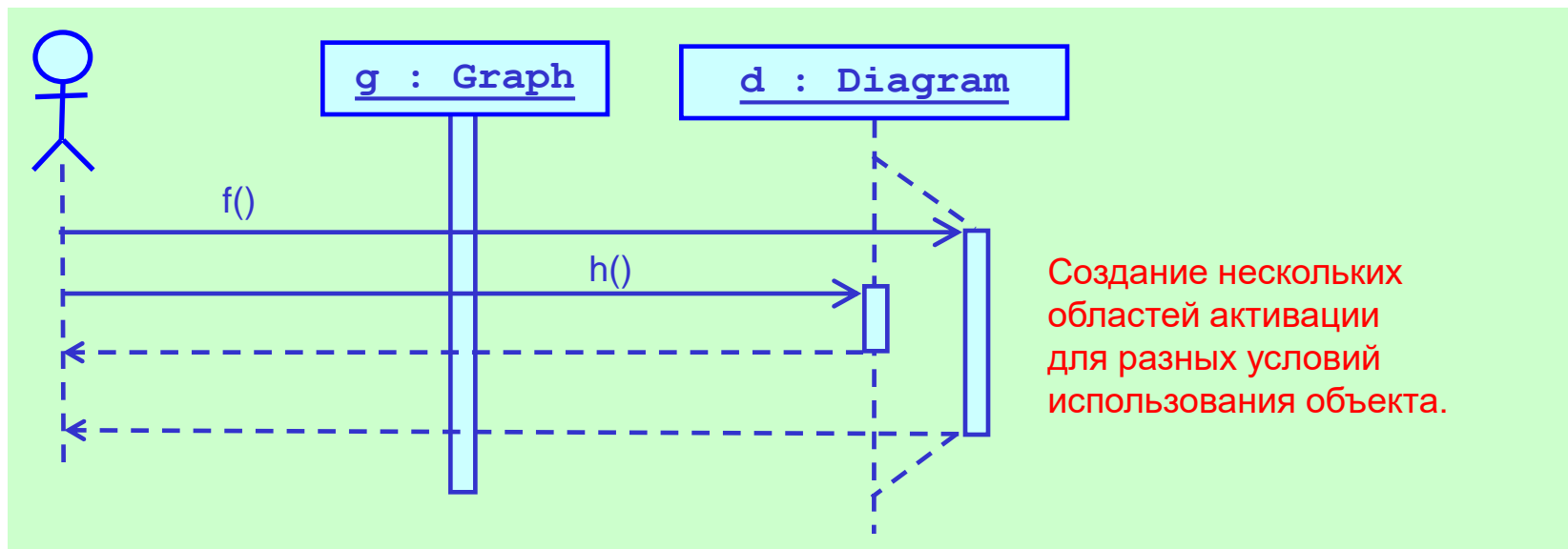
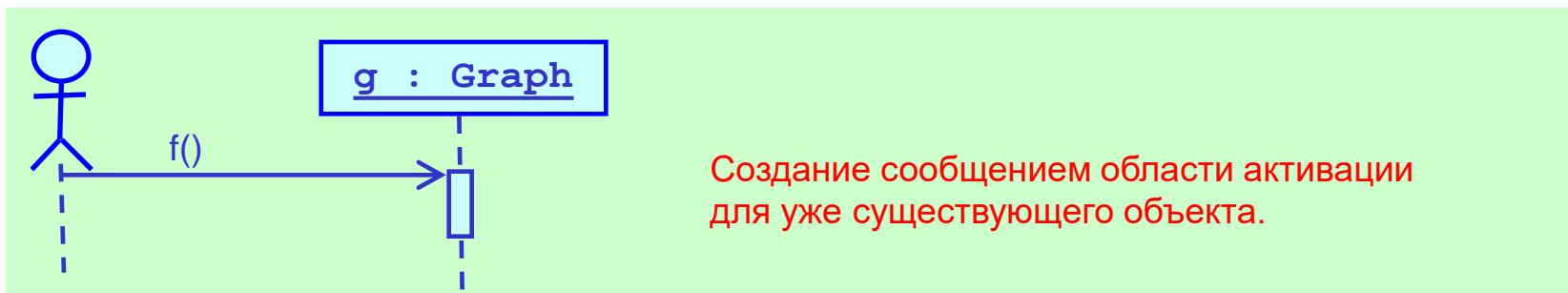
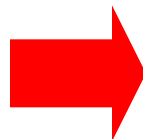


Диаграмма последовательности.

3.2 Рассылка сообщений объектами

- Изображения объектов
- Посылка сообщений объектами
- Нотация для структурирования диаграммы последовательности

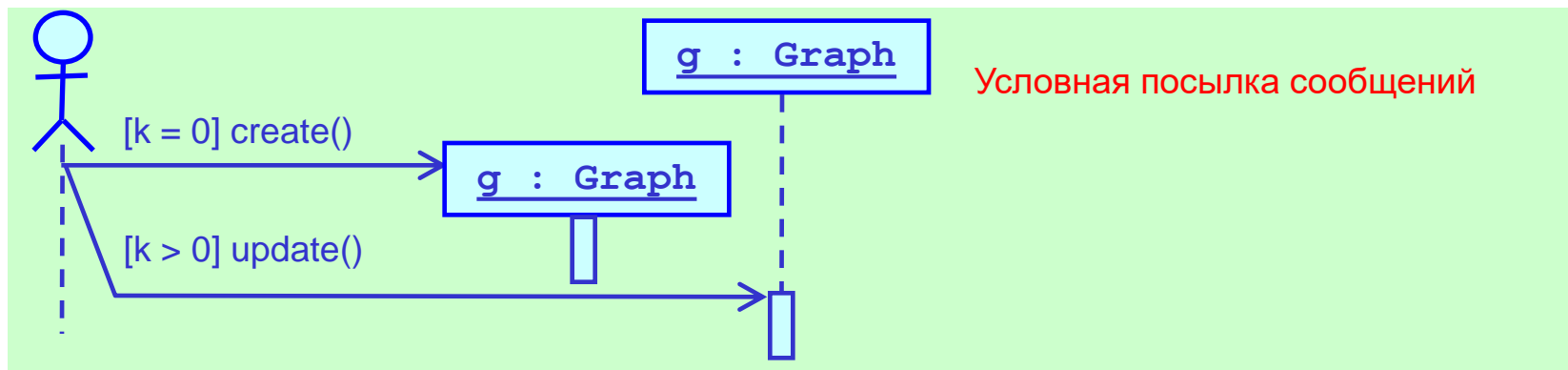
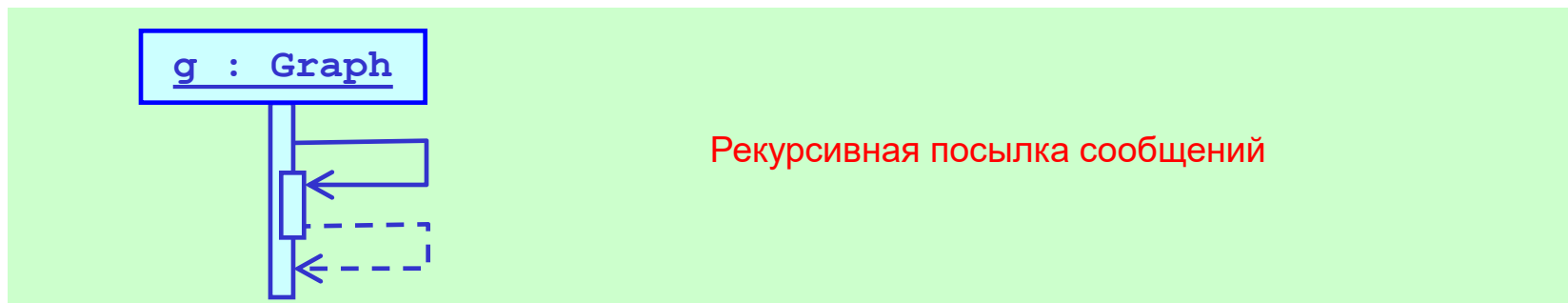


Посылка сообщений объектами.

Рекурсивная и условная посылка сообщений

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Посылка сообщений объектами.

Сообщения активных объектов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

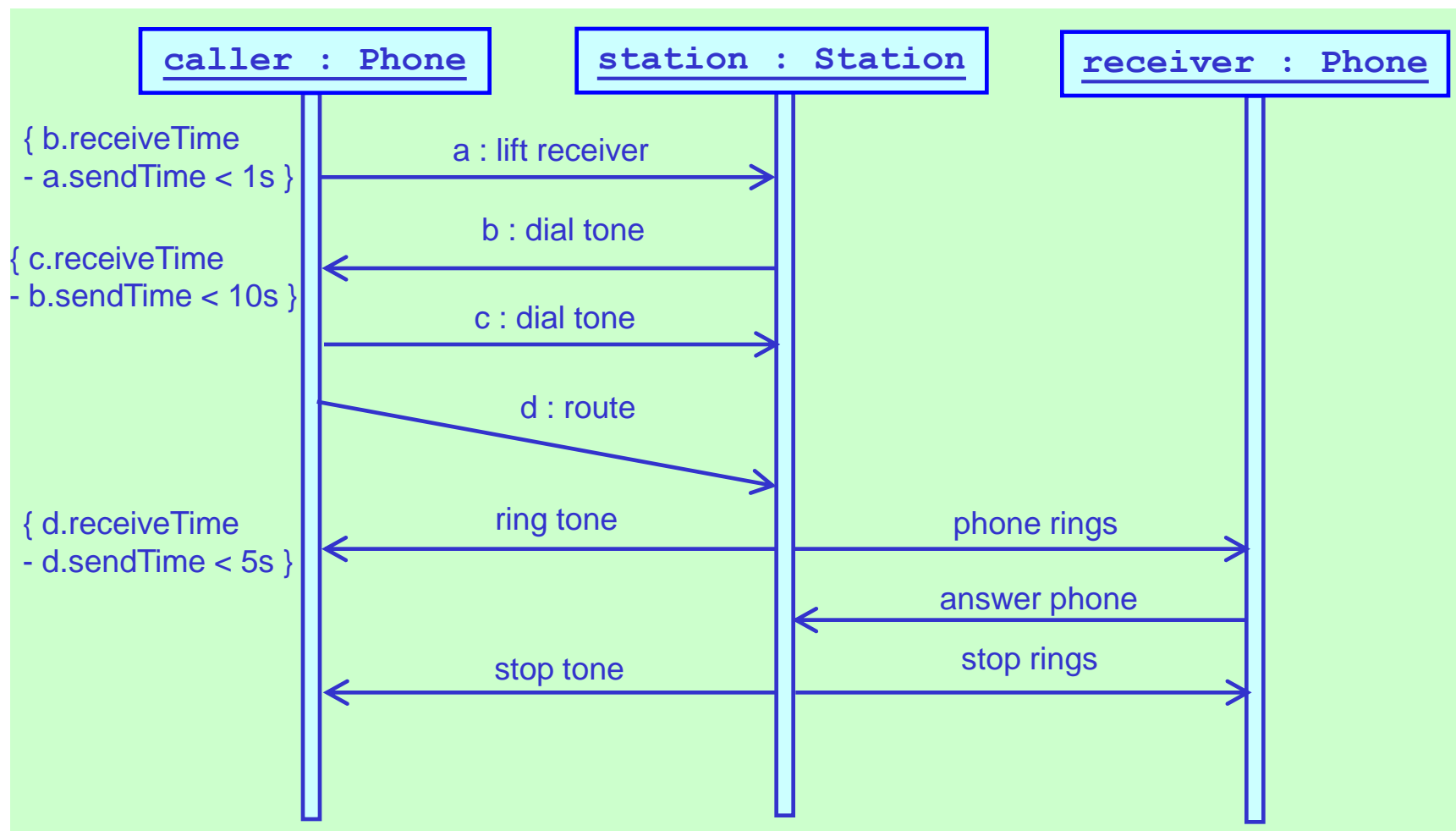
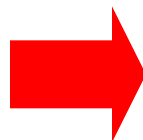


Диаграмма последовательности.

3.3 Структурирование диаграммы

- Изображения объектов
- Рассылка сообщений объектами
- Нотация для структурирования диаграммы последовательности

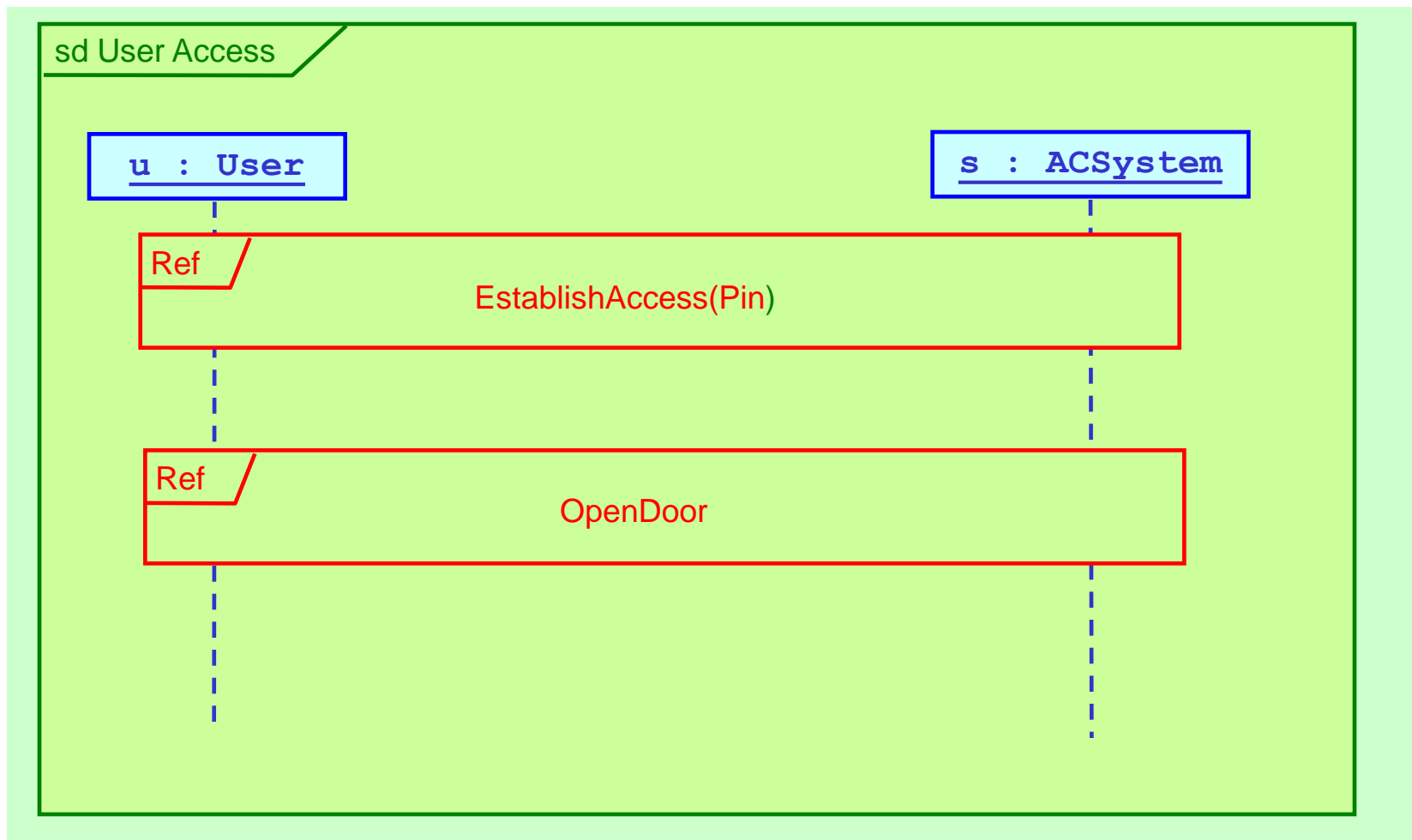


Структурирование диаграммы.

Узлы – ссылки

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

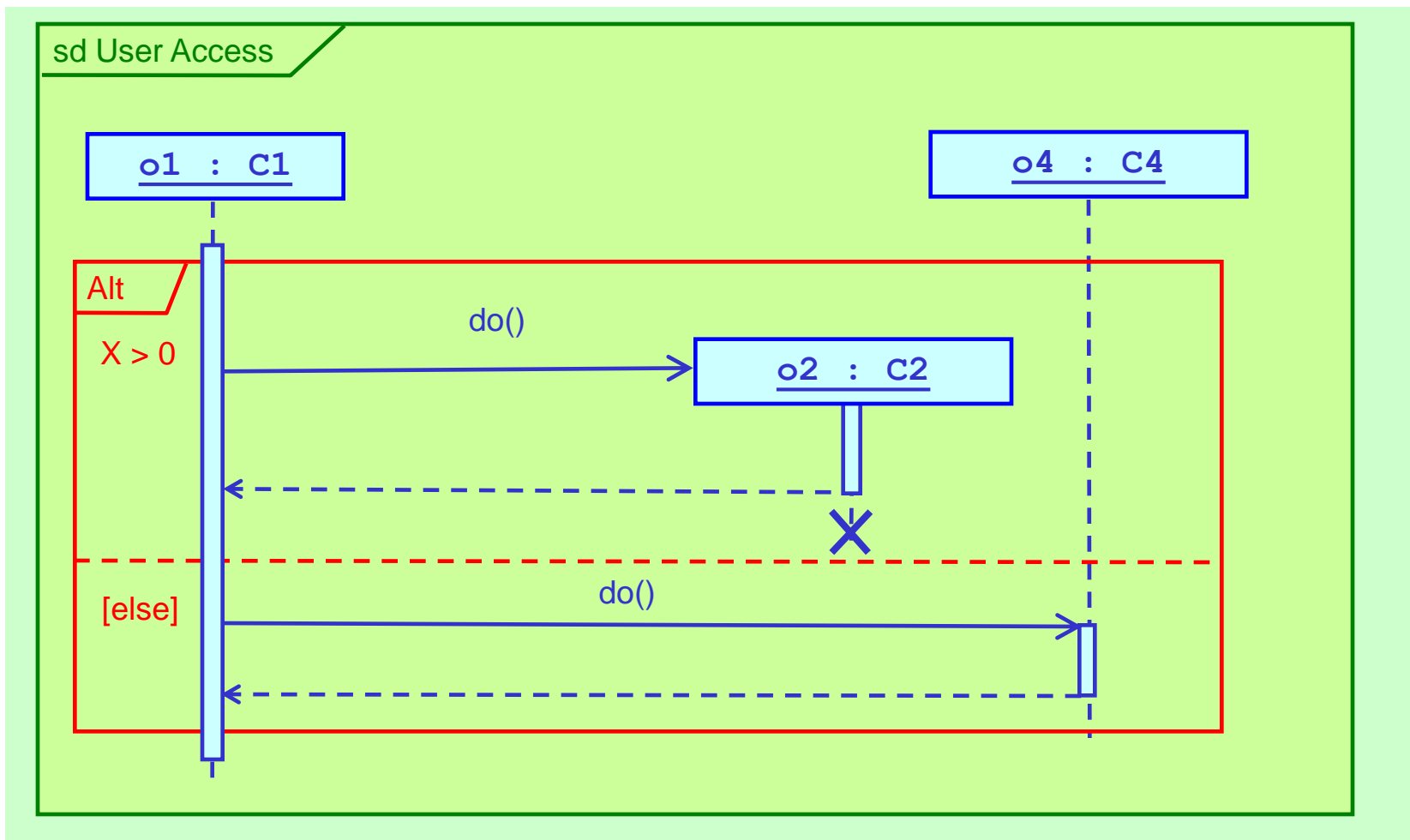


Структурирование диаграммы.

Узлы – альтернативные варианты выполнения

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



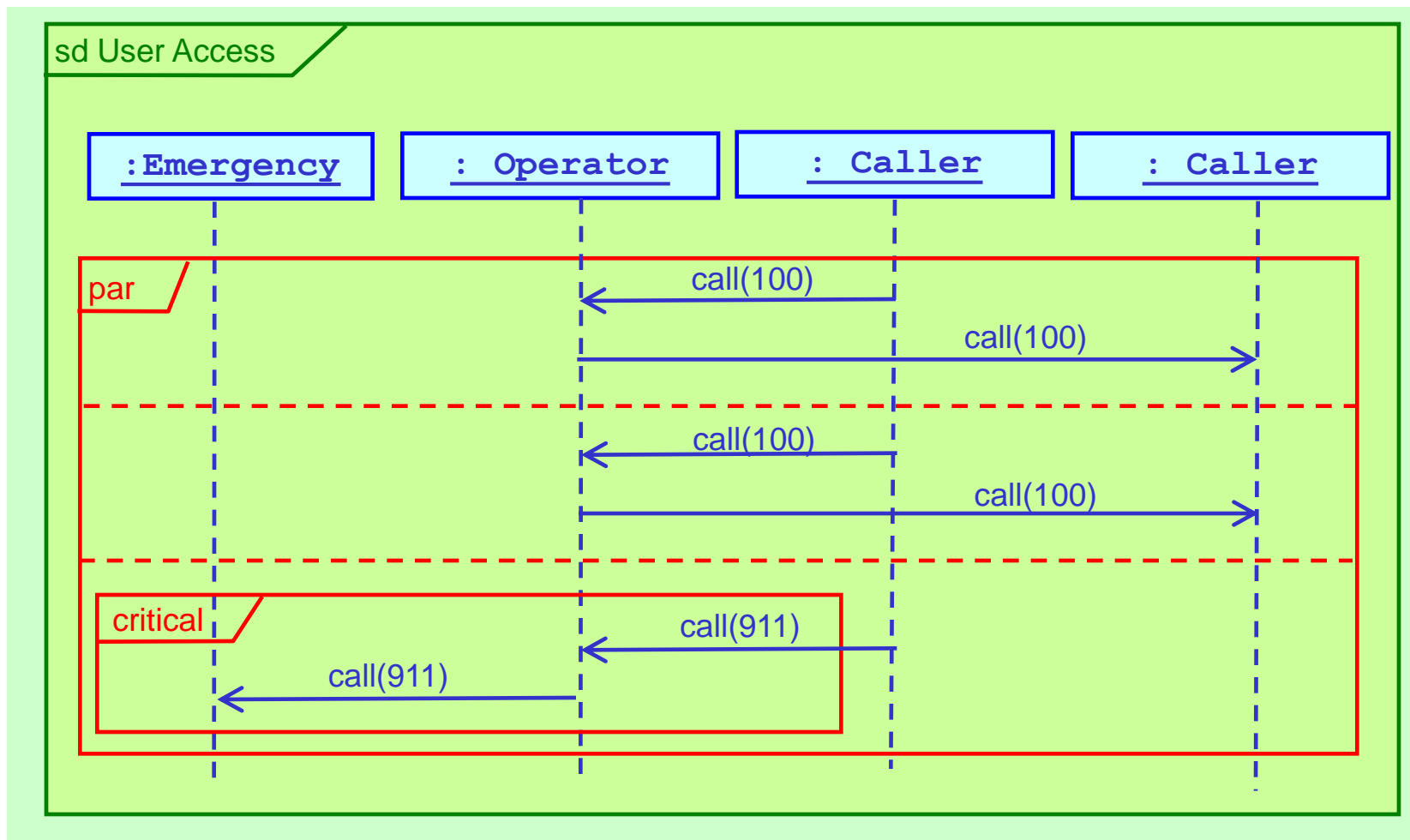
do()

Структурирование диаграммы.

Узлы – параллельные ветви исполнения

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

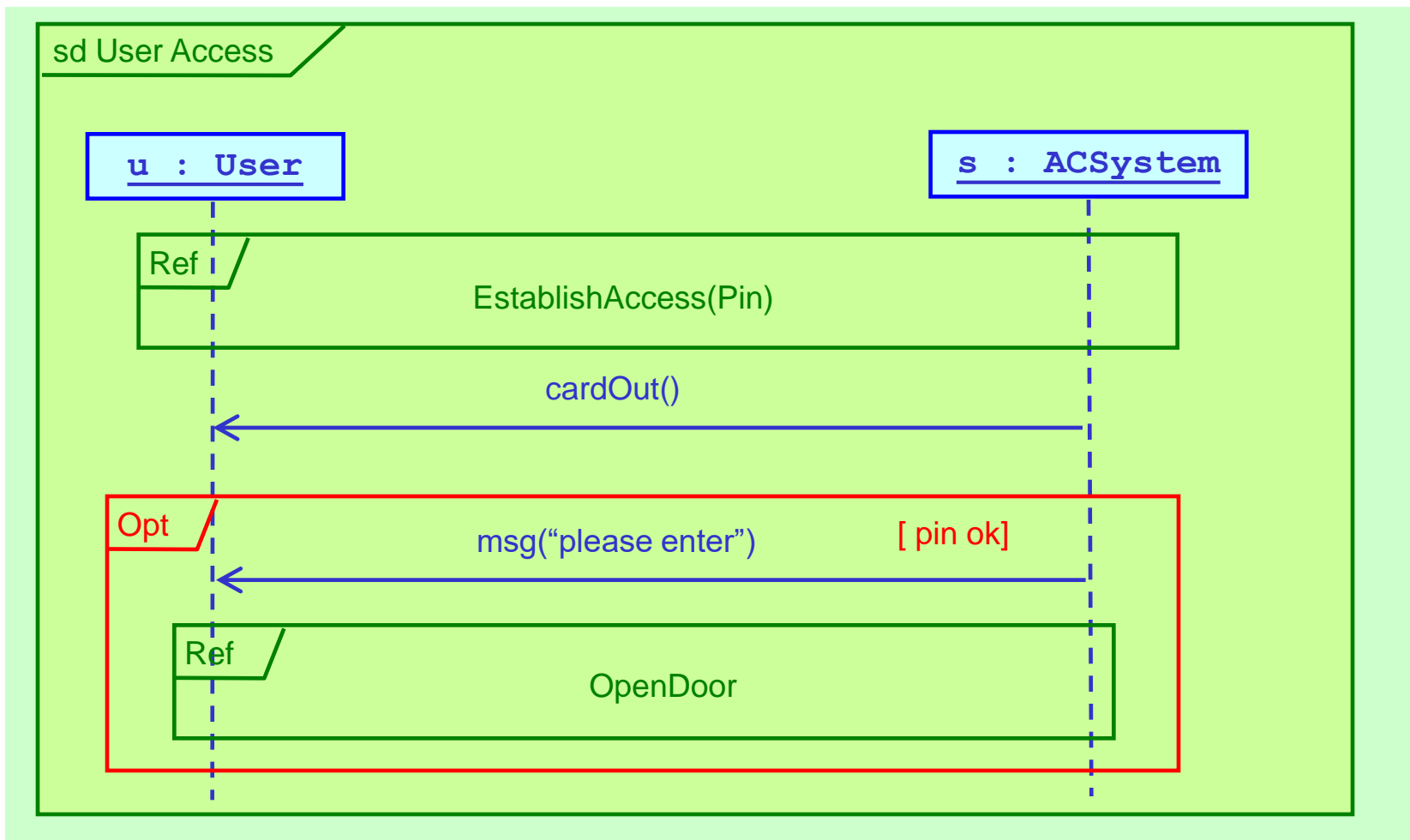


Структурирование диаграммы.

Узлы – условные ветви выполнения

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Диаграммы конечных автоматов

Диаграмма конечных автоматов

- Отражает жизненные циклы объектов
- Каждый объект реагирует на различные события (events)
- События – изменения вне объекта опознаваемые объектом:
 - получение вызова
 - передача сигнала
 - изменение значений
 - истечение промежутка времени

Диаграмма конечных автоматов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- С помощью конечных автоматов описывается:
 - Поведение классов
 - Динамическое поведение элементов Use Case
 - Коопераций объектов
 - Методов

Диаграммы конечных автоматов

Состояния

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- Состояние - множество значений объекта одного класса, при которых объект реагирует на события одинаковым образом
- Реакция на событие в состоянии (эффект)
 - Действие (action)
 - Деятельность (activity)
- В разных состояниях объекты могут реагировать на одно и то же событие по разному

Диаграммы конечных автоматов

Конечный автомат. События

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- **Конечный автомат** – граф состояний и переходов из одного состояния в другое
- **Событие** – явление имеющее положение во времени и пространстве
- События:
 - **Событие вызова** (call event)
 - **Событие сигнала** (signal event)
 - **Событие изменения** (change event)
 - **Событие времени** (time event)

Диаграммы конечных автоматов

События вызова и времени

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- **Событие вызова** - вызов операции объекта. Параметры операции – параметры события.

Переход конечного автомата, инициированный событием, может выполняться параллельно

Синтаксис:

op(a: T)

- **Событие времени** – истечение определенного промежутка времени.

Синтаксис:

***after**(time)*

Диаграммы конечных автоматов

События изменения и сигнала

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- **Событие изменения** – происходит когда становится истинным логическое выражение

Синтаксис:

when(логическое-выражение)

- **Событие сигнала** – получение сигнала-классификатора от объекта-отправителя приводит к возникновению события. Отправитель продолжает работу сразу после отправки сигнала без ожидания

Синтаксис:

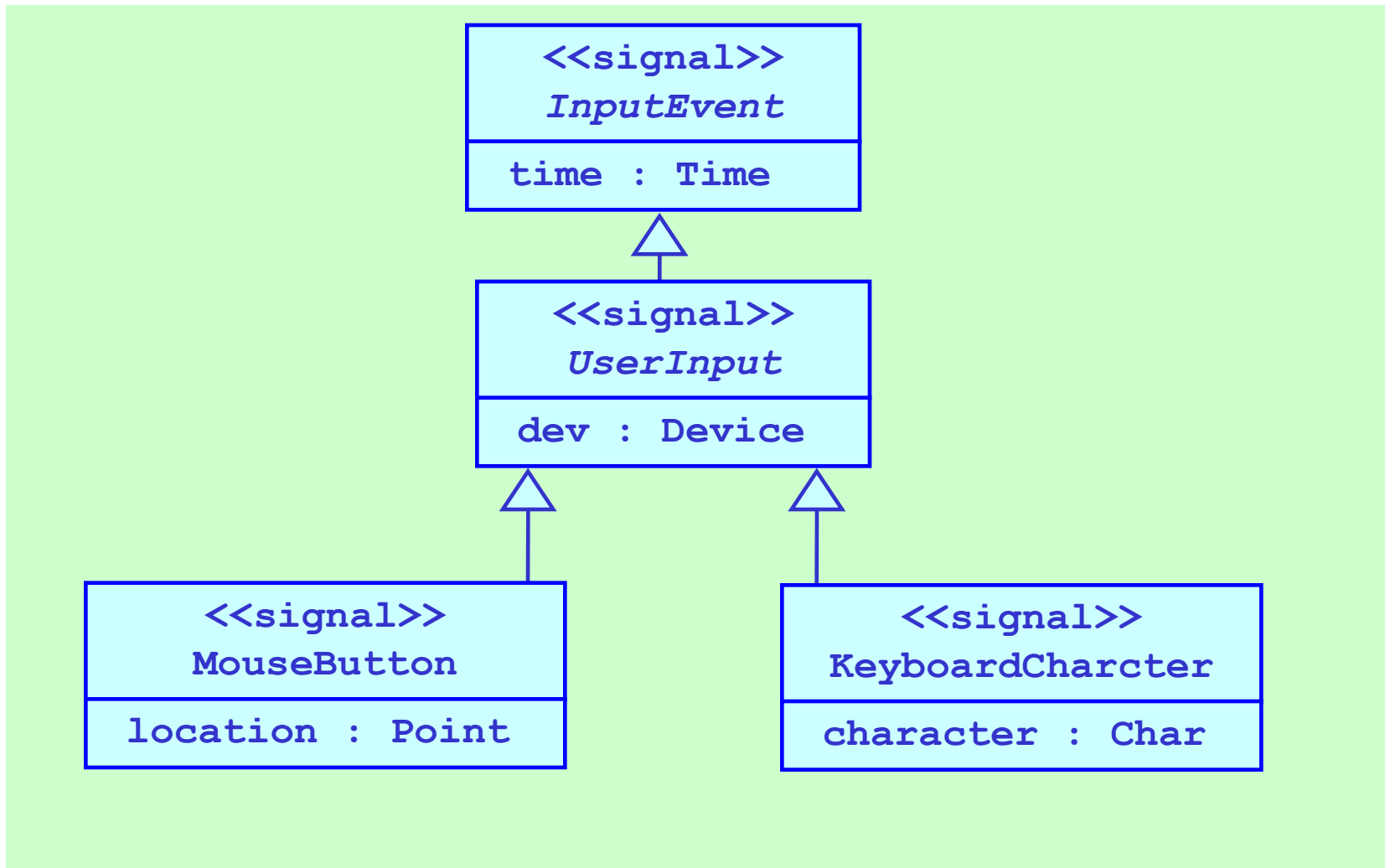
signal-name (*a : T*)

Диаграммы конечных автоматов

События изменения и сигнала

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Диаграммы конечных автоматов

Состояние

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

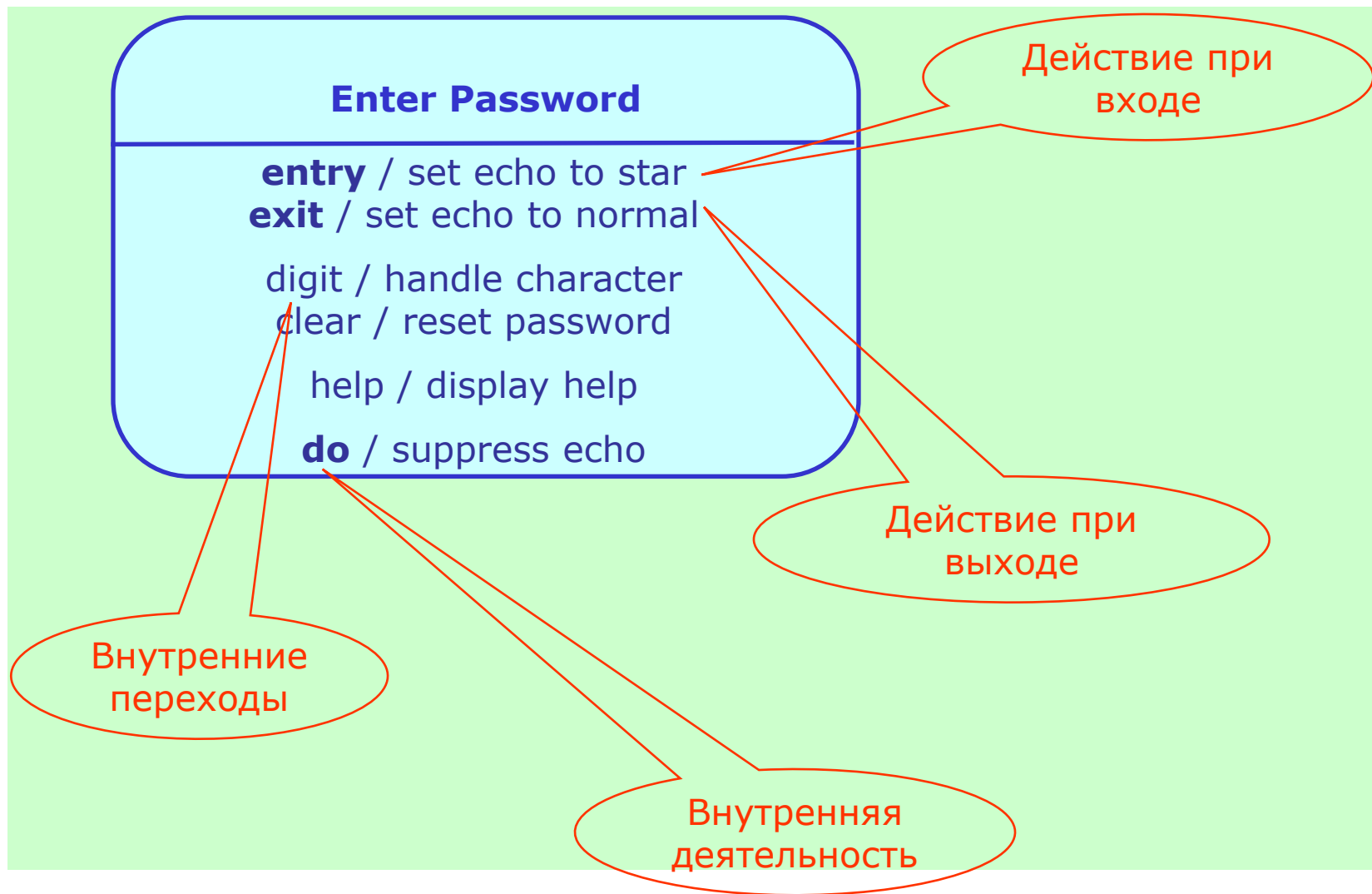
- Состояние – период времени в течение которого объект ожидает события
- Во время ожидания объект может выполнять некоторую деятельность
- Состояние может иметь имя. Все состояния без имени различны
- Состояния связаны между собой переходами.
- Переходы связывают два и более состояний.
- Более 2-х если есть сложные переходы с развилками и слияниями управления

Диаграммы конечных автоматов

Изображение состояния

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Диаграммы конечных автоматов

Действия по входу и выходу

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- **entry** – выполняется:
 - После действия прикрепленного на входе в состояние
 - До начала любой деятельности

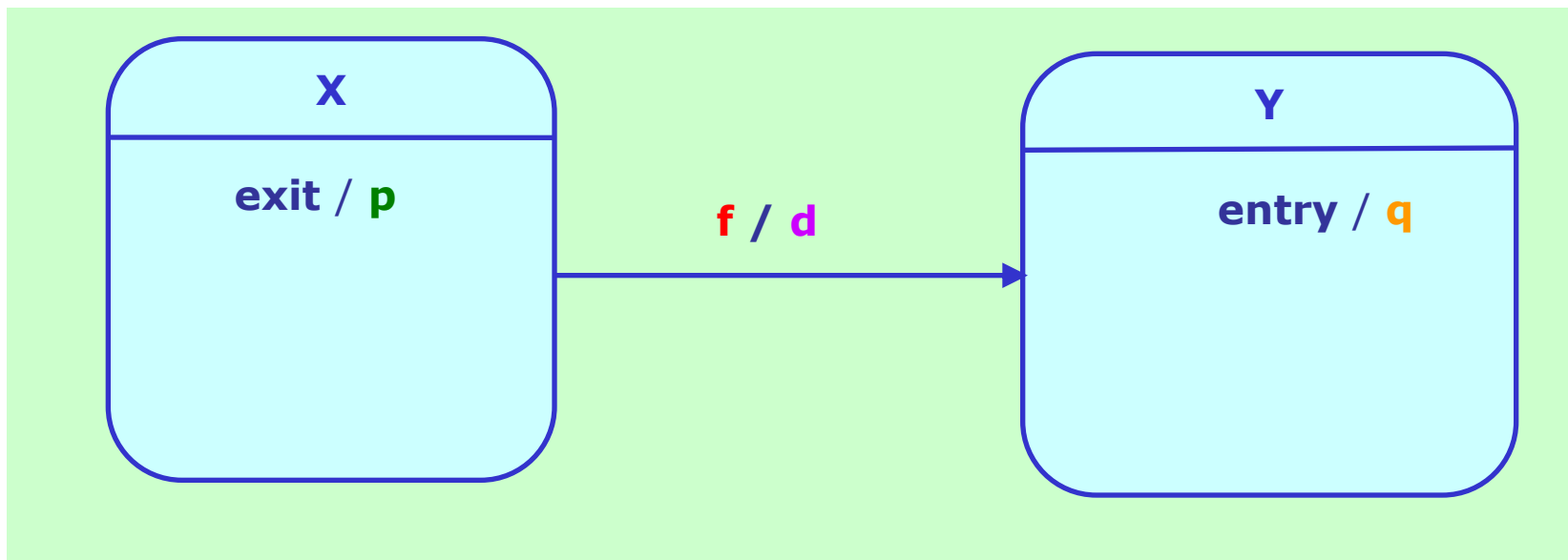
- **exit** – выполняется:
 - После завершения деятельности
 - До выполнения действия по переходу

Диаграммы конечных автоматов

Последовательность действий

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



- Последовательность действий:

f / p; d; q

Диаграммы конечных автоматов

Внутренняя деятельность

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- **do** выражение:
 - Неатомарное действие
 - Начинает выполняться сразу после завершения деятельности по входу
 - Завершение деятельности вызывает переход
 - Деятельность прерывается переходом в другое состояние

Диаграммы конечных автоматов

Внутренние переходы

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- Внутренний переход (без выхода из состояния)

Синтаксис:

метка-действия<список-параметров>

"[" спусковое-условие "]"

"/" выражение-действия

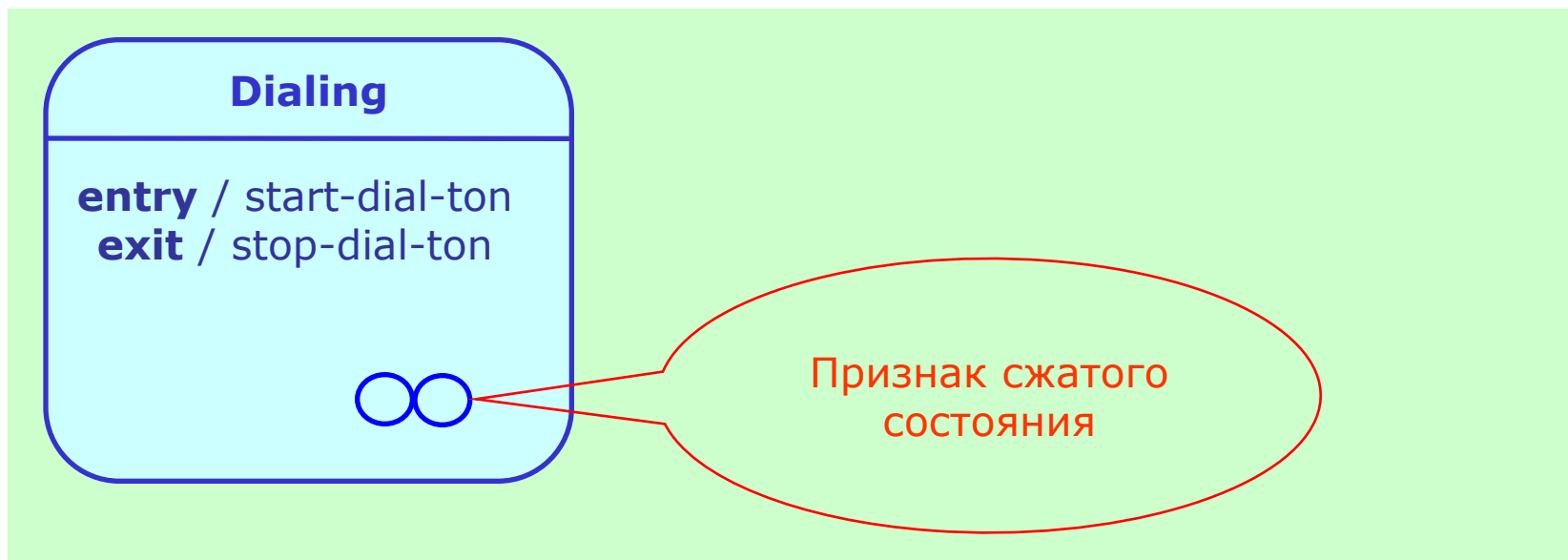
- спусковое-условие и выражение-действия могут использовать атрибуты и отношения объектов
- метка-действия может повторяться если спусковые-условия разные

Диаграммы конечных автоматов

Работа телефона (1)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

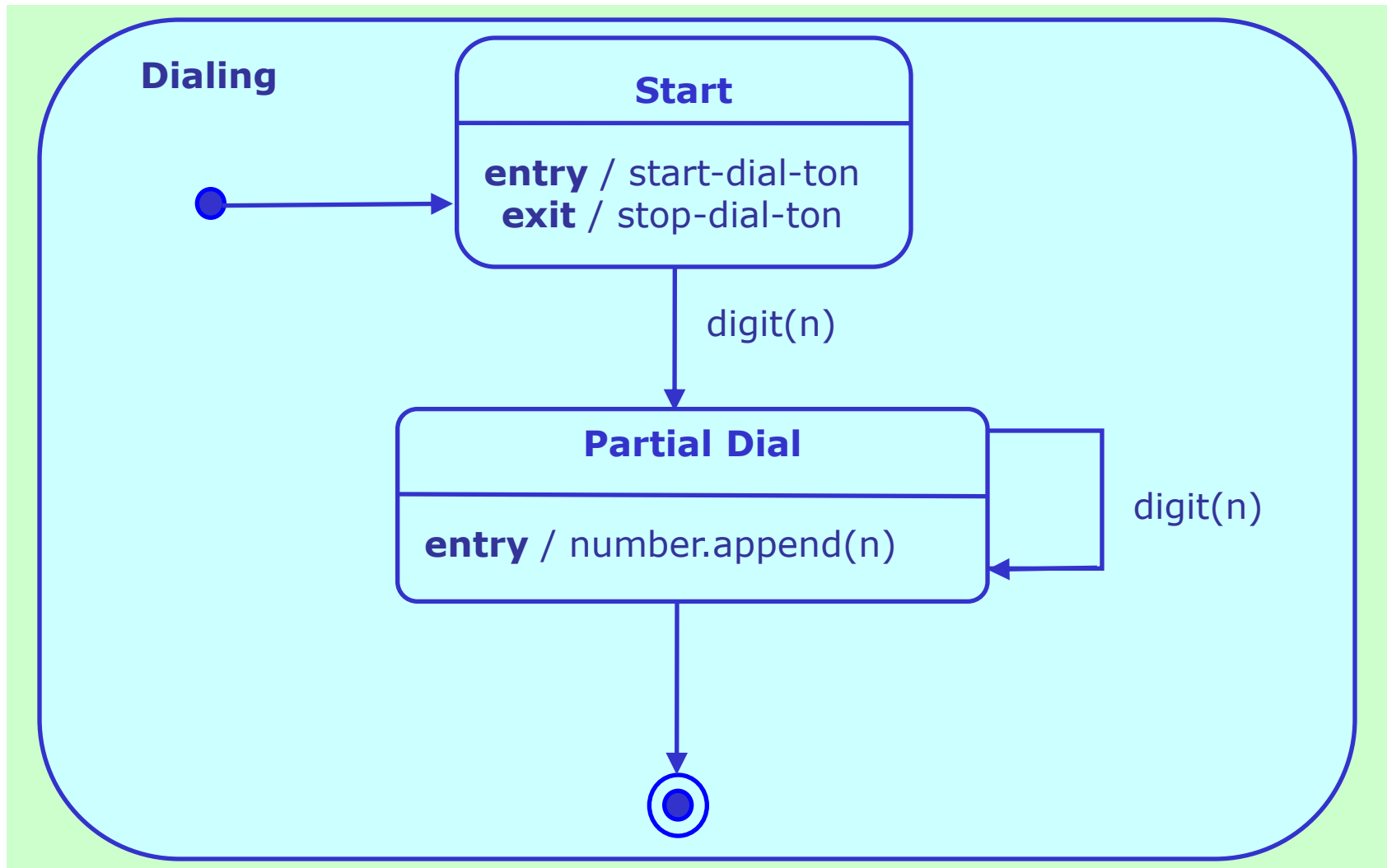


Диаграммы конечных автоматов

Вложенные состояния

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

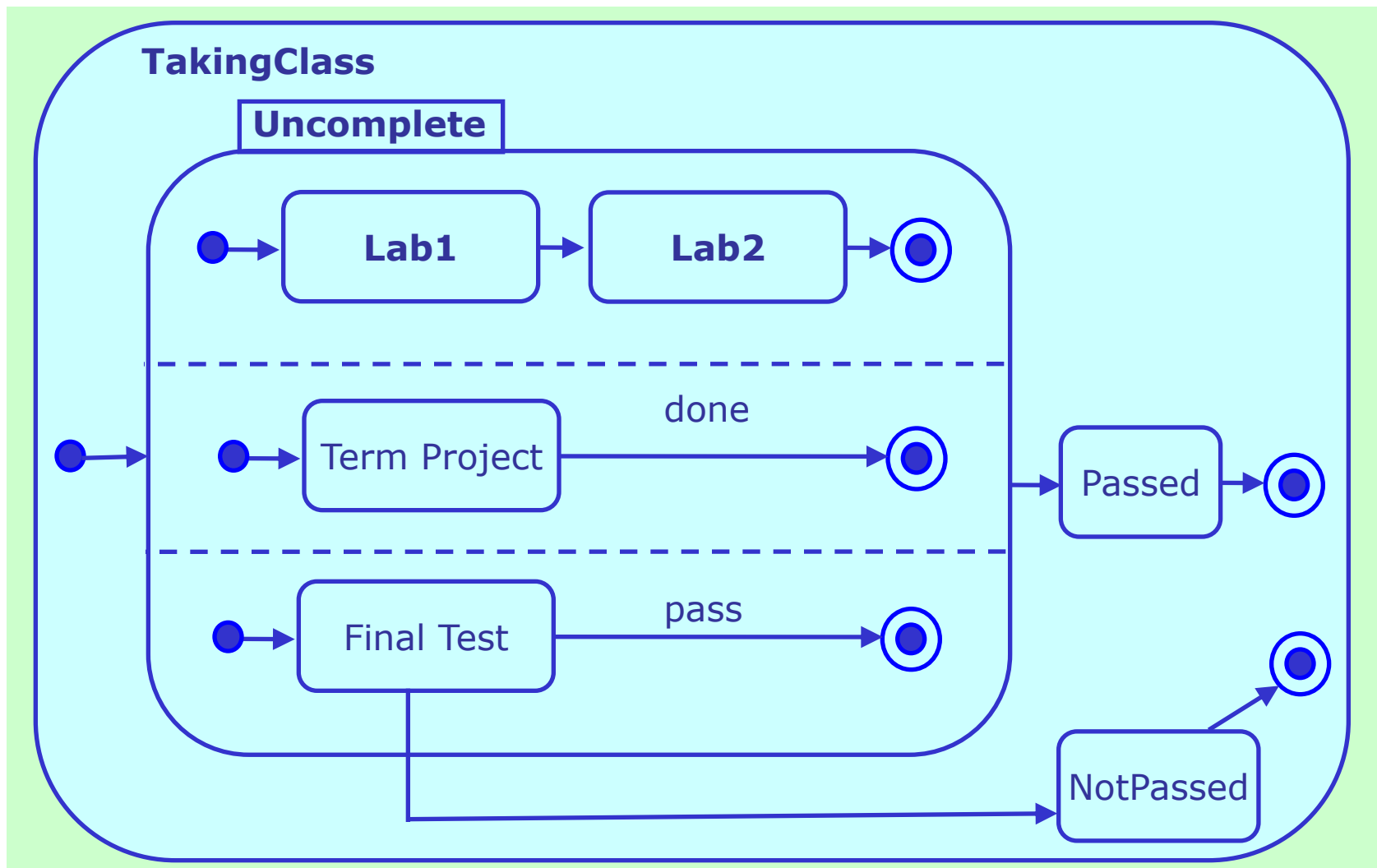


Диаграммы конечных автоматов

Параллельные вложенные состояния

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Диаграммы деятельности

Диаграммы деятельности.

Действие

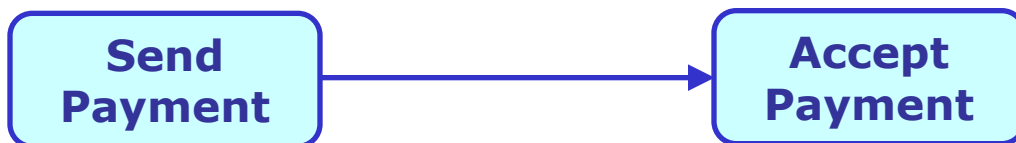
МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- **Action (действие)** представляет собой один шаг в деятельности, который не предполагает дальнейшей декомпозиции.



- *Пример:*



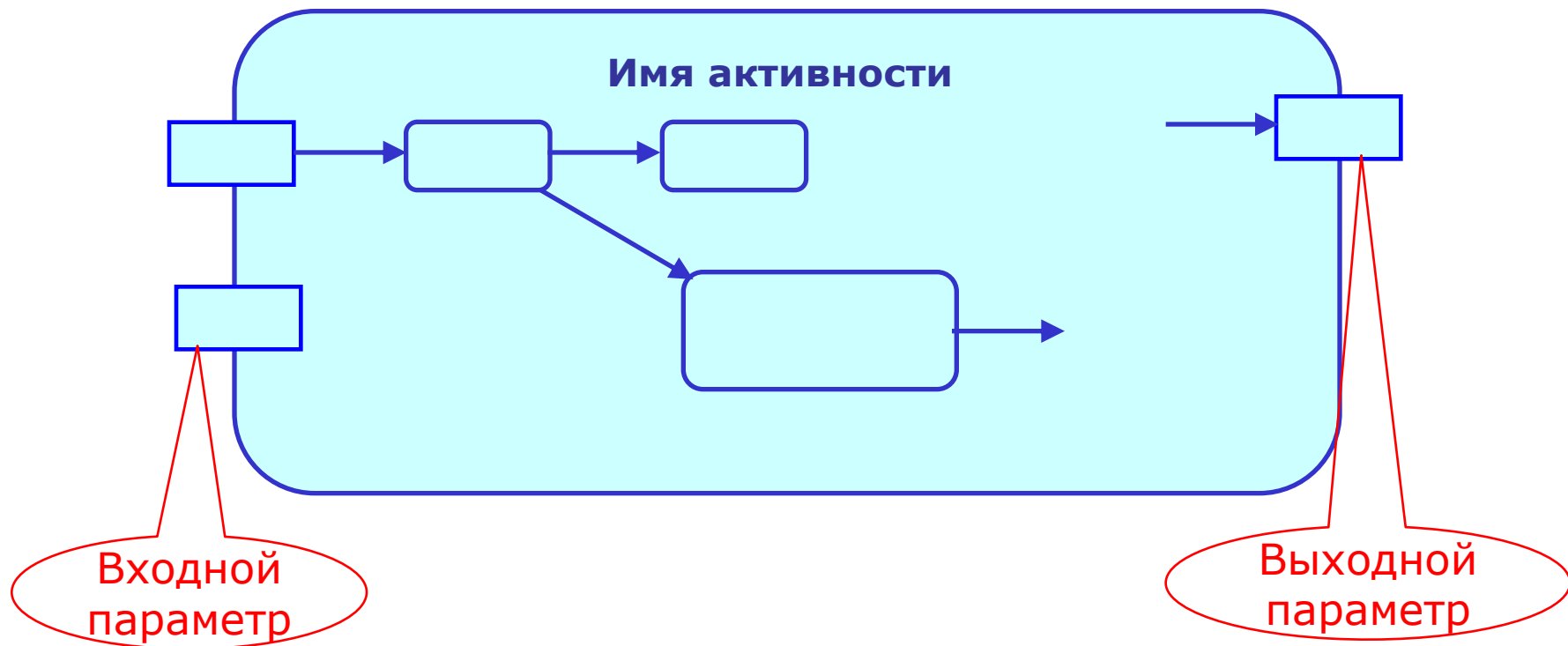
Диаграммы деятельности.

Деятельность

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- **Activity** (деятельность, активность) - параметризованная спецификация поведения как упорядоченная спецификация действий.



Диаграммы деятельности.

Деятельность - классификатор

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- **Activity** - это разновидность классификатора. Может иметь атрибуты и операции

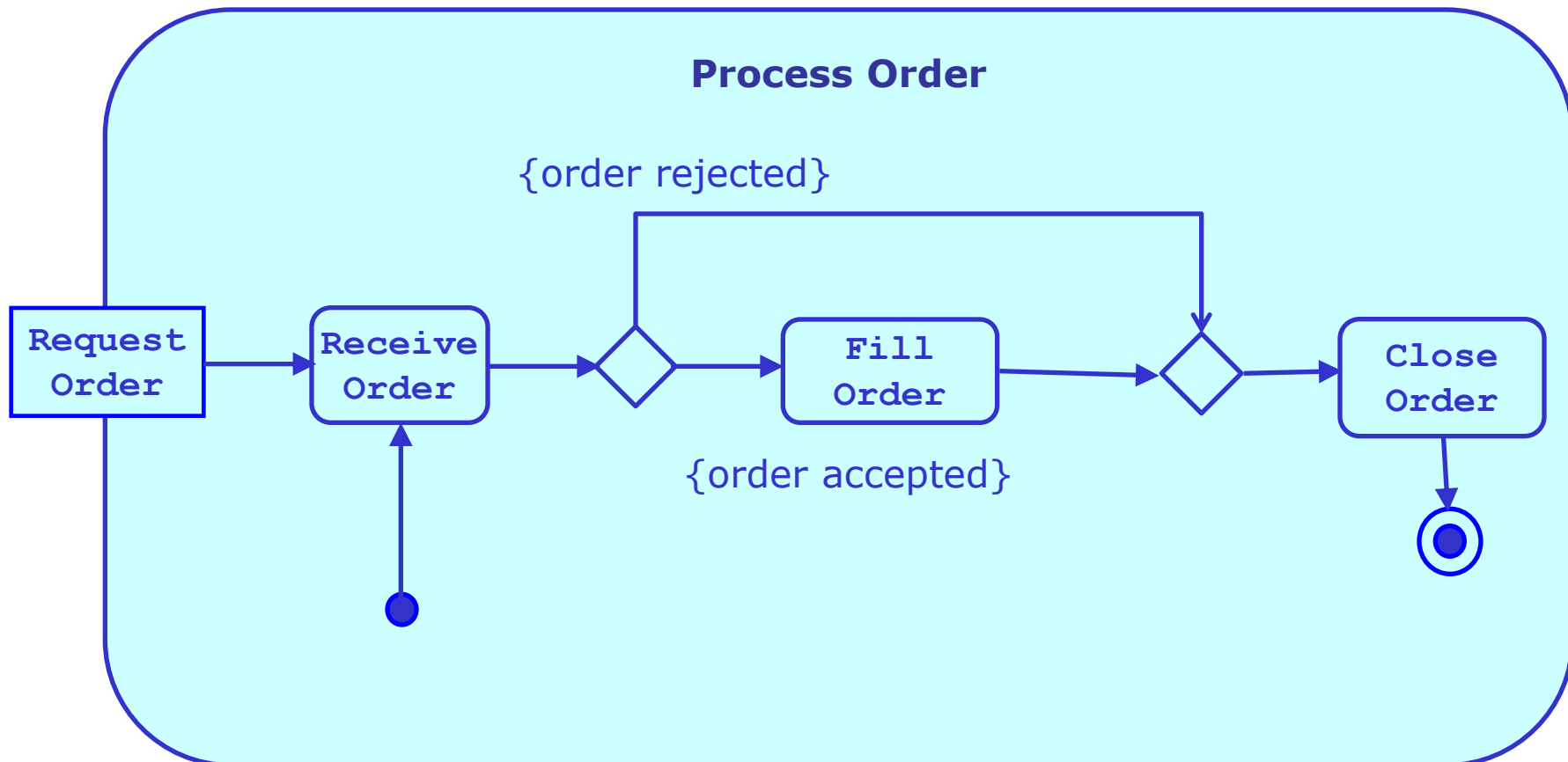
«activity» Имя деятельности
атрибут : тип
Операция (параметр: тип)

Диаграммы деятельности.

Пример деятельности

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Диаграммы деятельности.

Объекты на диаграммах деятельности

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



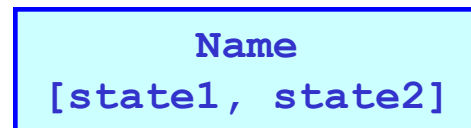
Единичный экземпляр объекта



Несколько экземпляров объекта



Объект имеющий тип Signal



Объект в состоянии state1



Множественный объект с заданным упорядочением:

{ordering = LIFO}

- Unordered
- Ordered
- LIFO
- FIFO

Диаграммы деятельности.

Поток объектов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

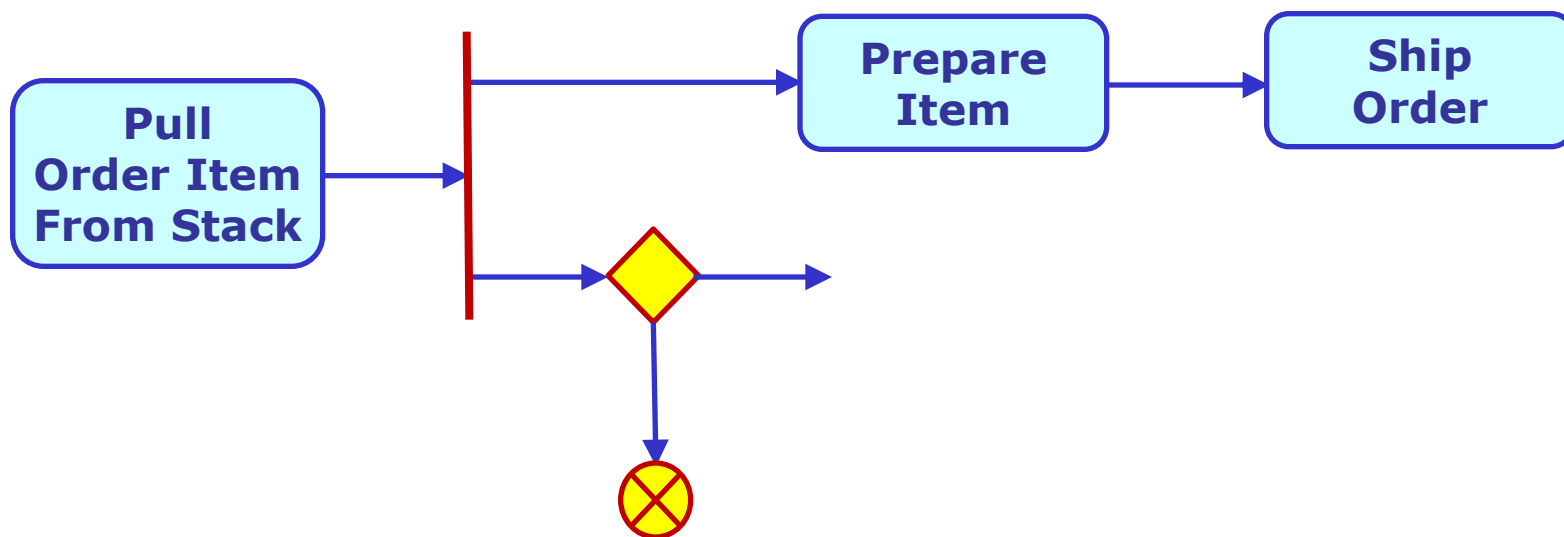


Диаграммы деятельности.

Управляющие узлы

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

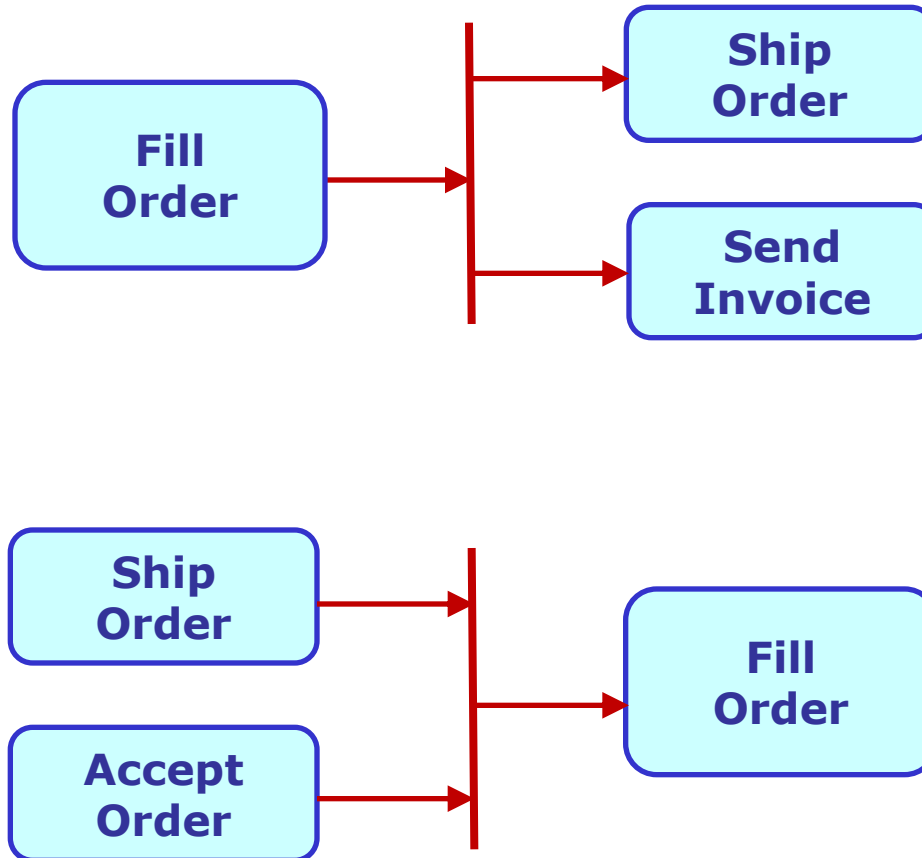


Диаграммы деятельности.

Узлы Fork (расщепить) и Join (слить)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

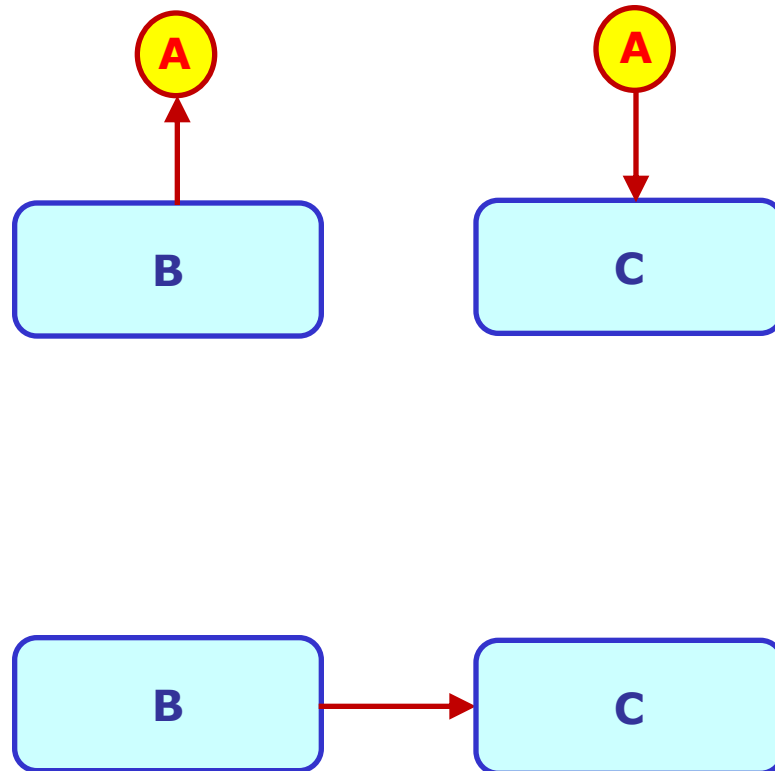


Диаграммы деятельности.

Узлы-соединители

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

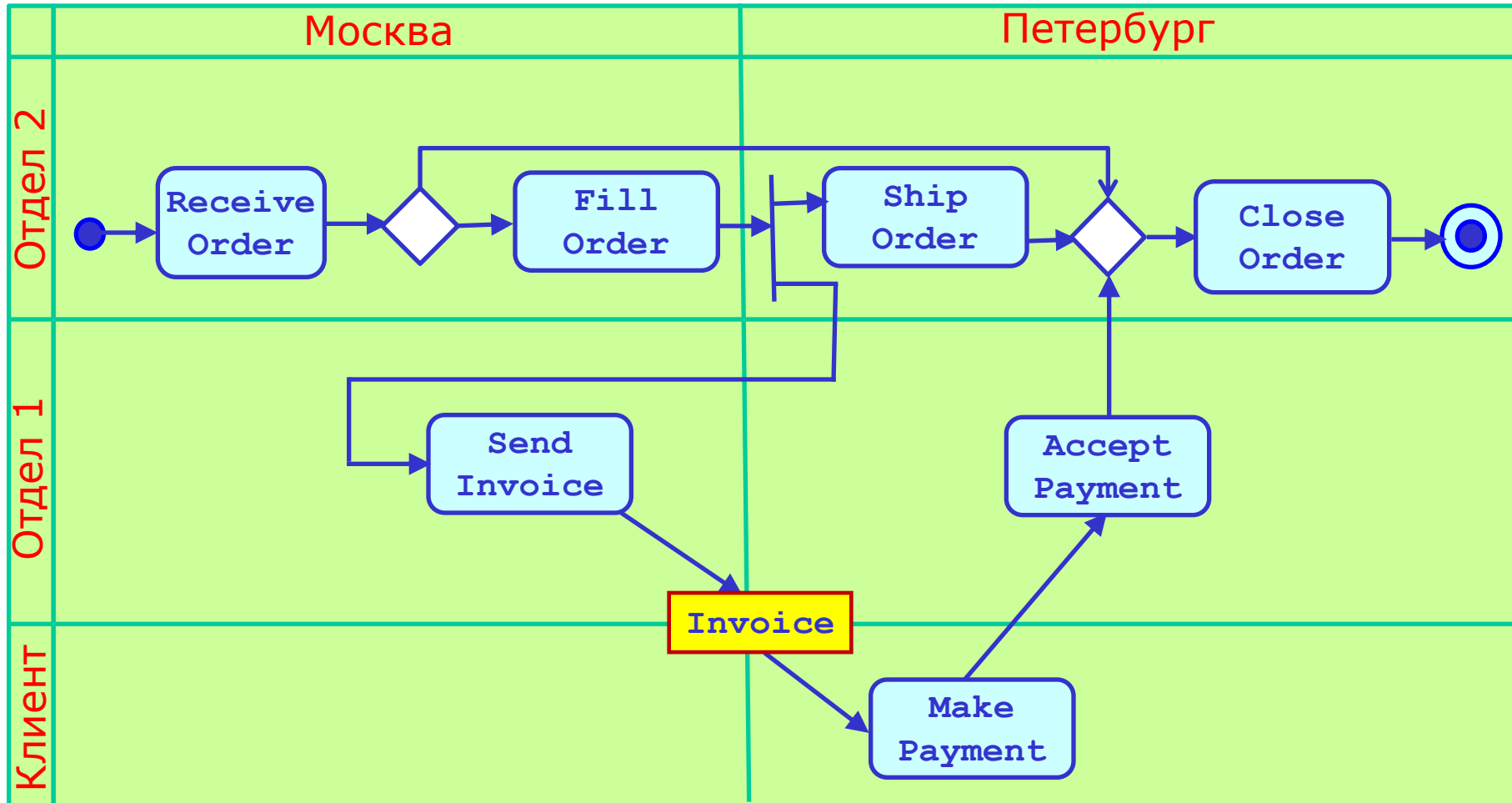


Диаграммы деятельности.

Дорожки заплыва и зоны ответственности

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

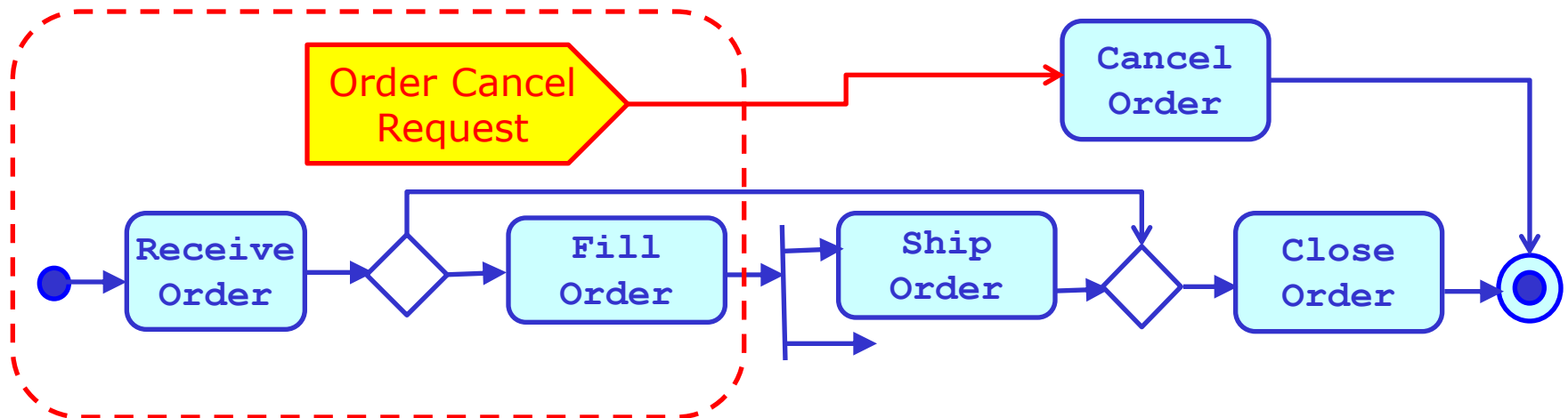


Диаграммы деятельности.

Области прерывания

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

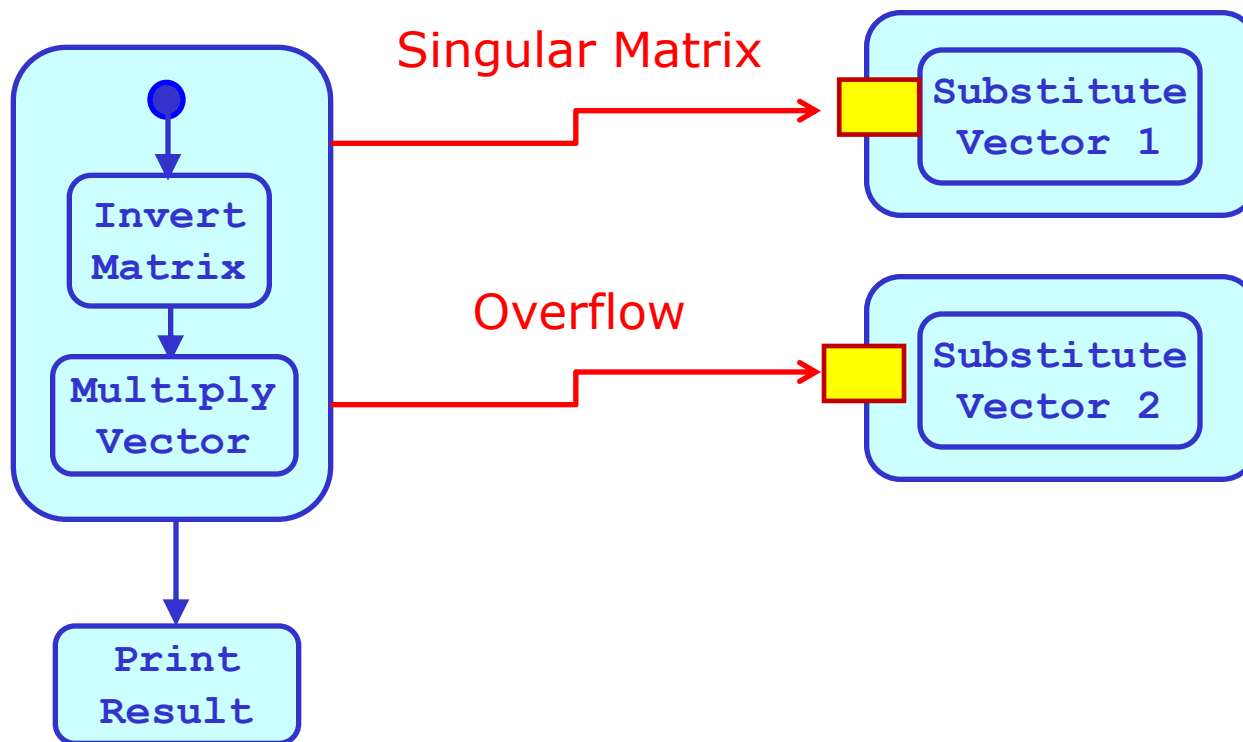


Диаграммы деятельности.

Обработка исключительных ситуаций

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Диаграммы прецедентов использования

Диаграммы прецедентов.

Назначение

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

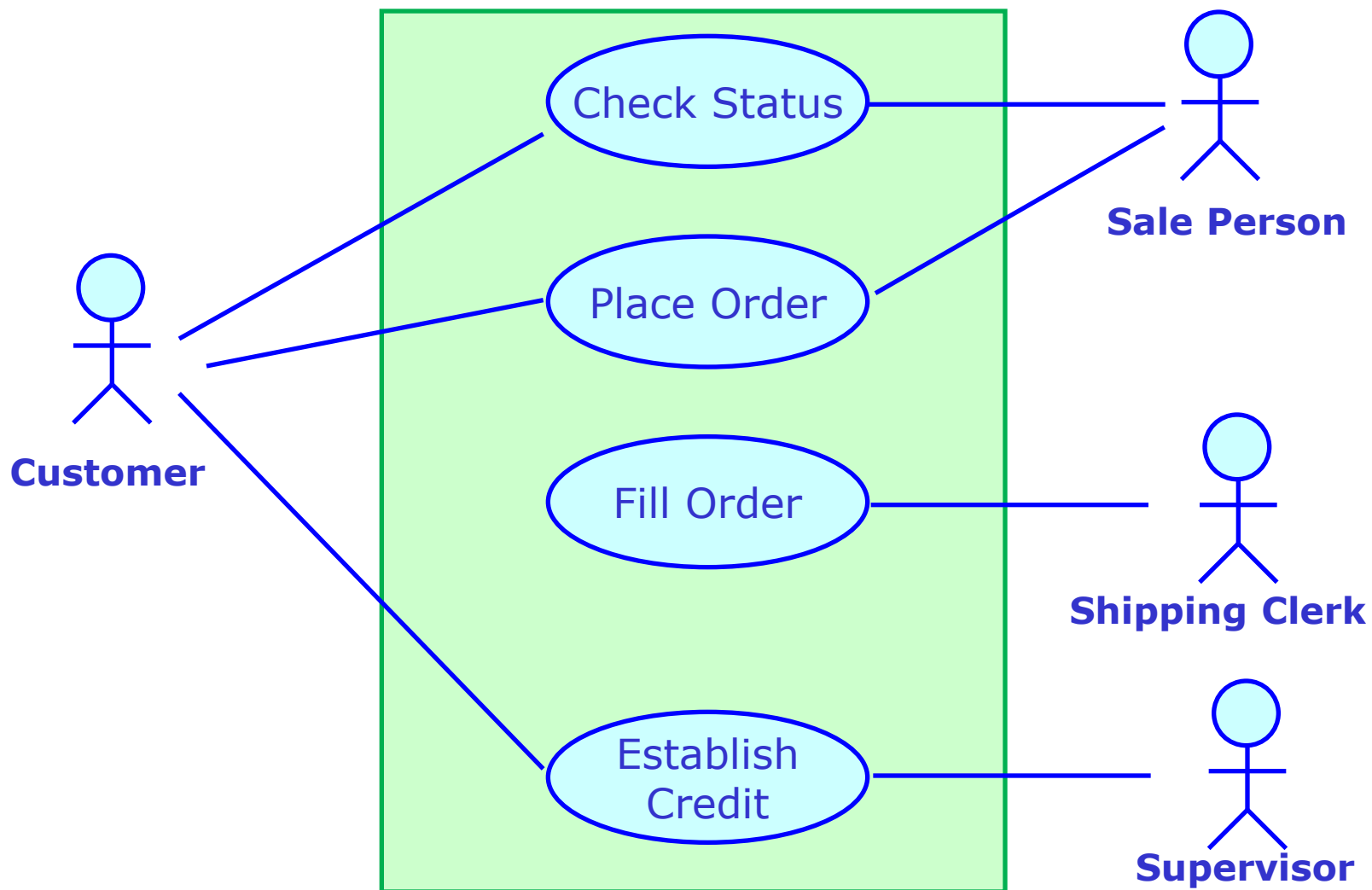
- Описание системы и классификаторов путем описания их взаимодействия с внешней средой.

- На диаграмме прецедентов показываются:
 - Актеры
 - Прецеденты
 - Интерфейсы
 - Отношения между перечисленными классификаторами

Пример диаграммы прецедентов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



Диаграммы прецедентов.

Элементы диаграммы

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

- Прямоугольной рамкой на диаграмме могут быть показаны границы используемой системы или классификатора
- Прецеденты располагаются внутри рамки
- Допустимые отношения:
 - Актеры: отношения обобщения
 - Актеры и прецеденты: отношения ассоциации
 - Прецеденты: отношения обобщения, отношения зависимости *включения* и *расширения*

Диаграммы прецедентов.

Актеры

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024

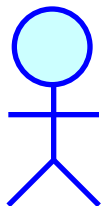
- **Актер** - согласованное множество ролей, которые могут играть пользователи моделируемой системы или классификатора
- *Актерами могут быть:*
 - *Люди*
 - *Физические устройства*
 - *Внешние по отношению к моделируемой системе классификаторы*

Диаграммы прецедентов.

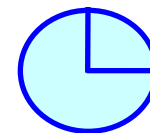
Графическое обозначение актеров

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2024



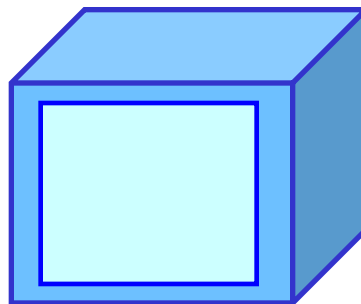
User



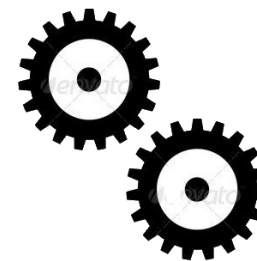
Clock



Phone



Monitor



Mechanism