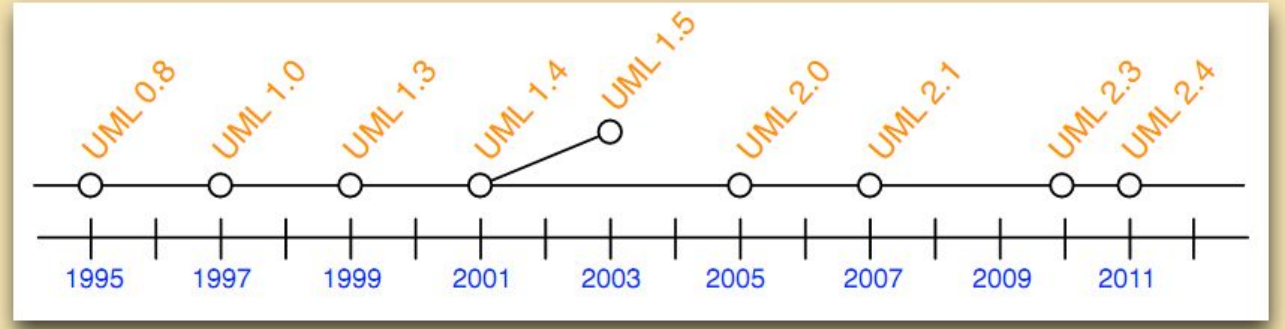
Применение UML при разработке программного обеспечения

Что такое UML (Unified Modeling Language)

- UML - это **язык**
  - знаковая система для хранения и передачи информации
- UML - это язык **моделирования**
  - Модель - это описание системы, предметной области, явления
- UML - это **унифицированный** язык моделирования



Сущность UML

Назначение - спецификация, визуализация, конструирование, документирование

Спецификация = описание (системы)

Средство описания -Как устроена и работает

взгляд заказчика ≠ взгляд разработчик

Визуализация

Средство коммуникации (наглядность)

<

<

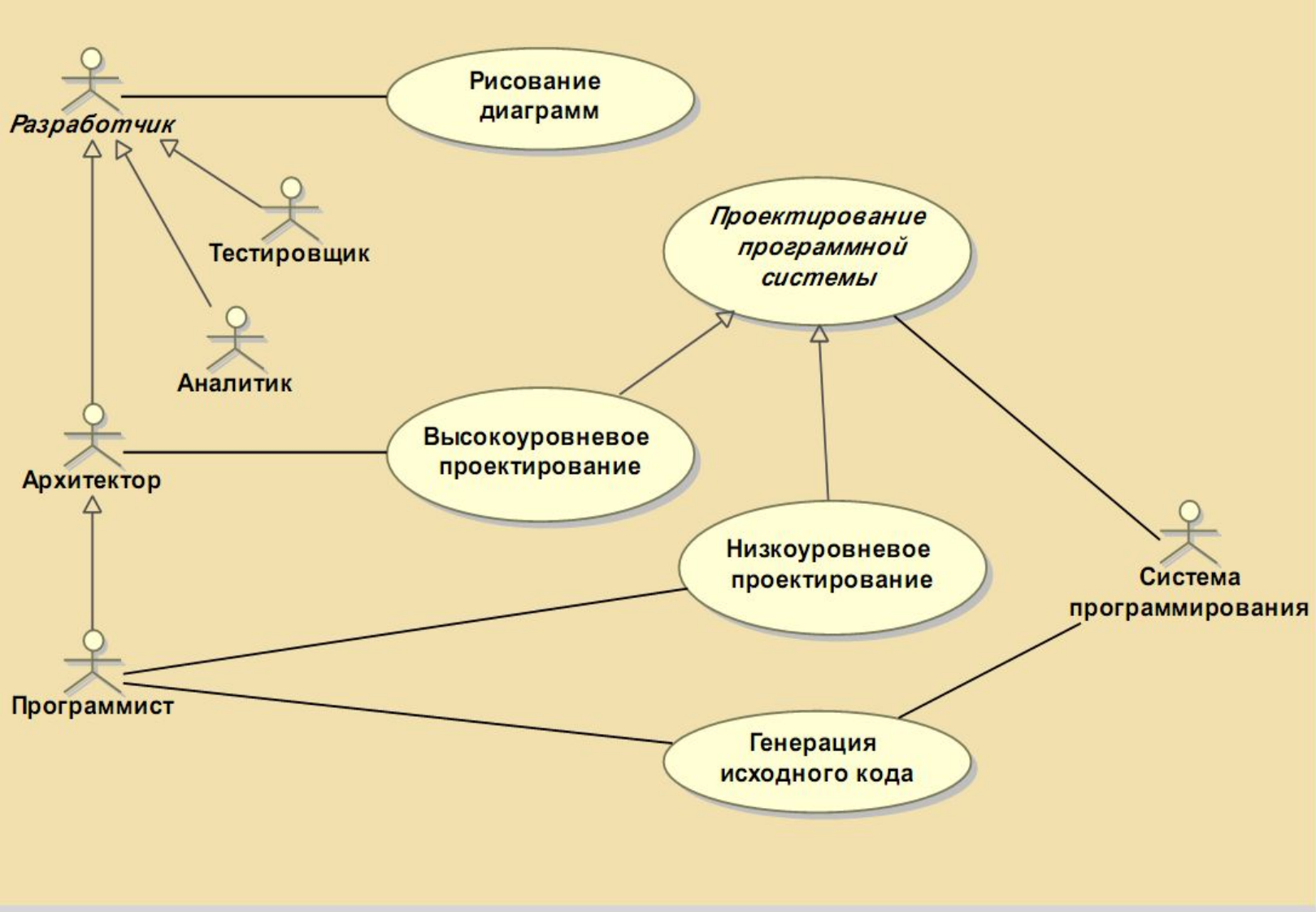
Конструирование = модель → артефакт (документ)

≈ автоматический синтез программ

Документирование

все элементы модели могут быть описаны

Использование UML



Модель процесса моделирования

Сущности представлений

Представление использования	• Действующие лица • Границы системы • Функциональные требования • Ответственность компонентов
Представление структуры	• Предметная область • Архитектура системы • Структура хранения • Детали реализации
Представление поведения	• Бизнес-процессы • Пользовательский интерфейс • Алгоритмы обработки • Жизненные циклы

Стандарт UML

OMG WE SET THE STANDARD

UNIFIED MODELING LANGUAGE™

www.uml.org

MOF Meta-Object Facility

Meta-Object Facility core specification, 07-08-2011, v 2.4.1

UNIFIED MODELING LANGUAGE

Infrastructure, 05-08-2011, v 2.4.1 — 230 стр.

Superstructure, 06-08-2011, v 2.4.1 — 748 стр.

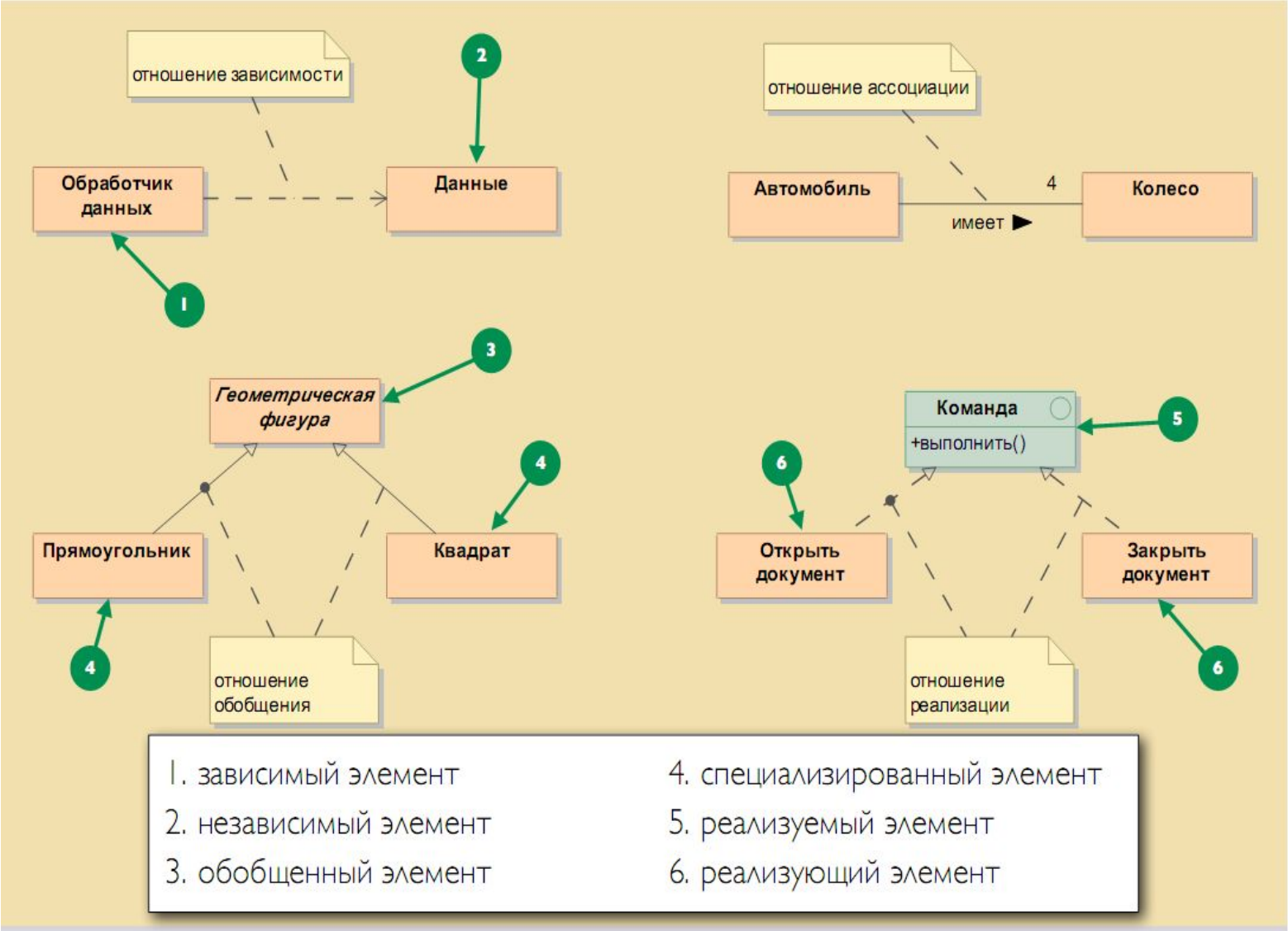
Object Constraint Language, 01-01-2012, v 2.3.1

UML Diagram Interchange specification, 04-2006, v 1.0

Сущности UML

Имя	Нотация	Имя	Нотация
Артефакт Artifact	«artifact» Requirement Specification	Компонент Component	DataBase
Вариант использования Use case	Make Order	Объект Object	: Rectangle
Действующее лицо Actor	Customer	Пакет Package	Analysis Model
Деятельность / действие Activity / action	Display menu	Комментарий Comment	It is a comment
Интерфейс Interface	«interface» IAudio	Состояние State	Logged entry / OpenLog exit / CloseLog
Класс Class	Product	Вычислительный узел Node	Server
Активный класс Active class	Dispatcher	Кооперация Collaboration	Visitor

Отношения UML



Диаграммы

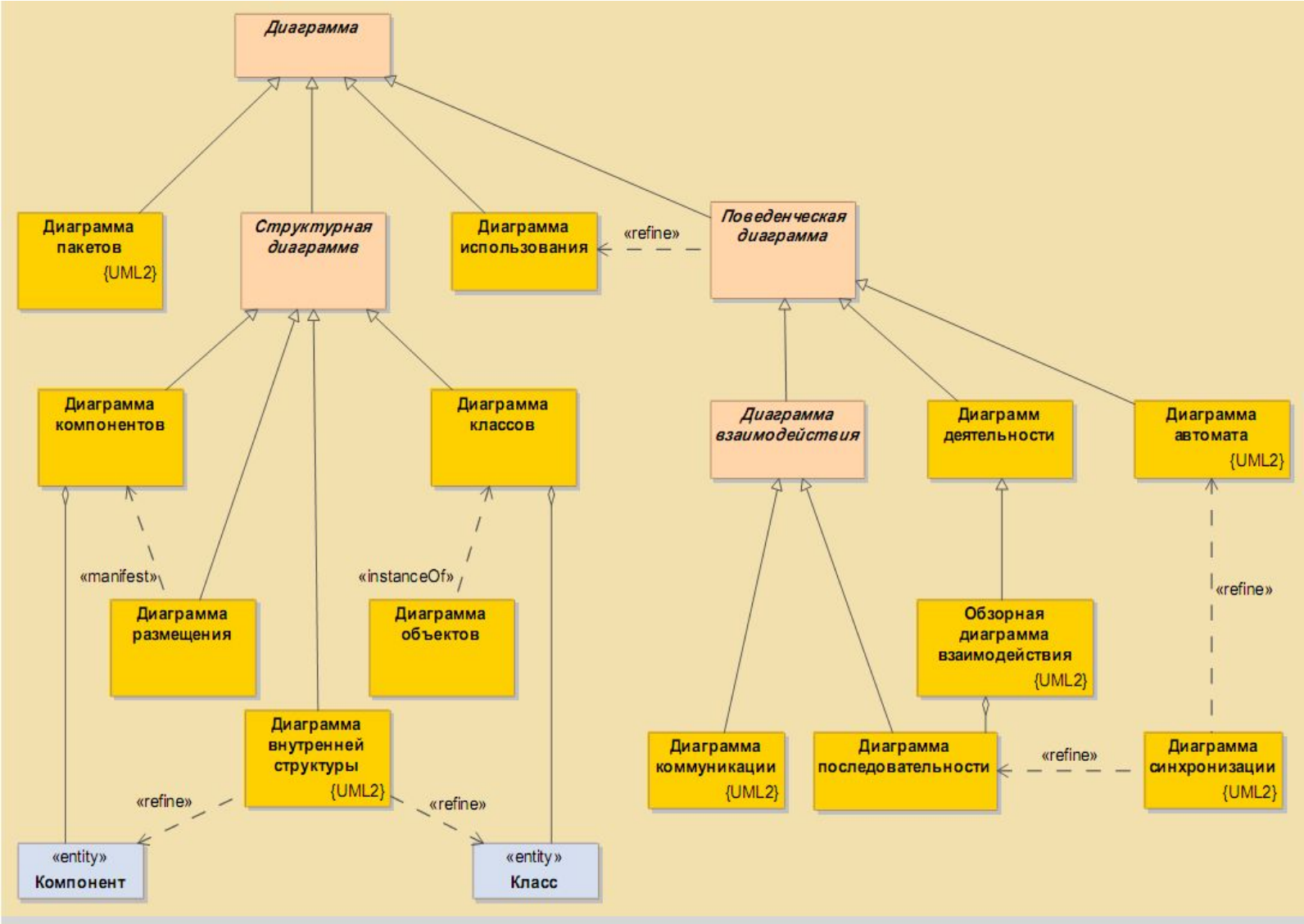


Диаграмма
Диаграмма пакетов {UML2}
Структурная диаграмма
Диаграмма использования
«refine»
Поведенческая диаграмма
Диаграмма компонентов
Диаграмма классов
Диаграмма взаимодействия
Диаграмм деятельности
Диаграмма автомата {UML2}
«manifest»
«instanceOf»
«refine»
Диаграмма размещения
Диаграмма объектов
Обзорная диаграмма взаимодействия {UML2}
Диаграмма внутренней структуры {UML2}
Диаграмма коммуникации {UML2}
Диаграмма последовательности
«refine»
Диаграмма синхронизации {UML2}
«refine»
«refine»
«entity» Компонент
«entity» Класс

Назначение диаграмм

Что делает система?

- Диаграмма использования / Use case diagram

Из чего состоит система?

- Диаграмма классов / Class diagram
- Диаграмма компонентов / Component diagram
- Диаграмма размещения / Deployment diagram
- Диаграмма объектов / Object diagram
- Диаграмма внутренней структуры / Composite structure diagram

Как работает система?

- Диаграмма автомата / State machine diagram
- Диаграмма деятельности / Activity diagram
- Диаграмма коммуникации / Communication diagram
- Диаграмма последовательности / Sequence diagram
- Обзорная диаграмма взаимодействия / Interaction overview diagram
- Диаграмма синхронизации / Timing diagram

Как уменьшить сложность модели?

- Диаграмма пакетов / Package diagram

Диаграммы использования (Use Case)

Диаграммы вариантов использования описывают функциональное назначение системы или то, что система должна делать. Разработка диаграммы преследует следующие цели:

- определить общие границы и контекст моделируемой предметной области;
- сформулировать общие требования к функциональному поведению проектируемой системы;
- разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
- подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

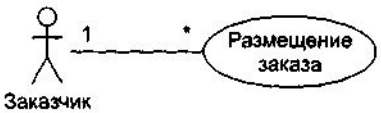
Отдельный ***вариант использования*** (прецедент) обозначается на диаграмме ***эллипсом***, внутри которого содержится его краткое название.

Актер представляет собой любую внешнюю по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей.

Стандартным графическим обозначением актера на диаграммах является ***фигурка человечка***, под которой записывается имя актера.

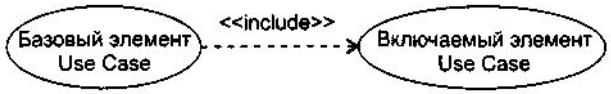
Отношения в диаграммах использования

Ассоциация



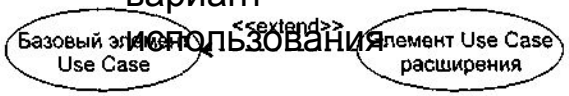
Включение

Включаемый элемент является составной частью базового элемента



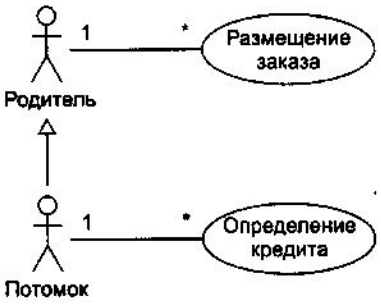
Расширения

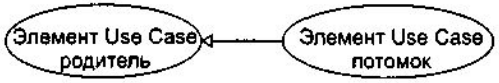
Частный вариант использования

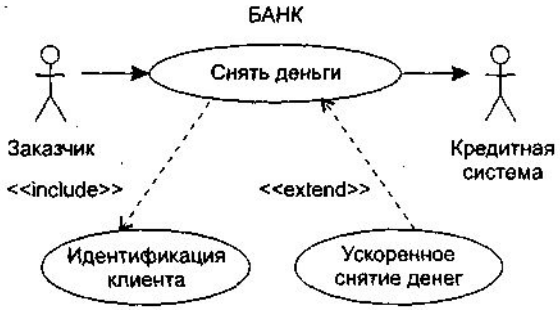


Обобщение

Потомок наследует поведение родителя







Пример диаграммы использования

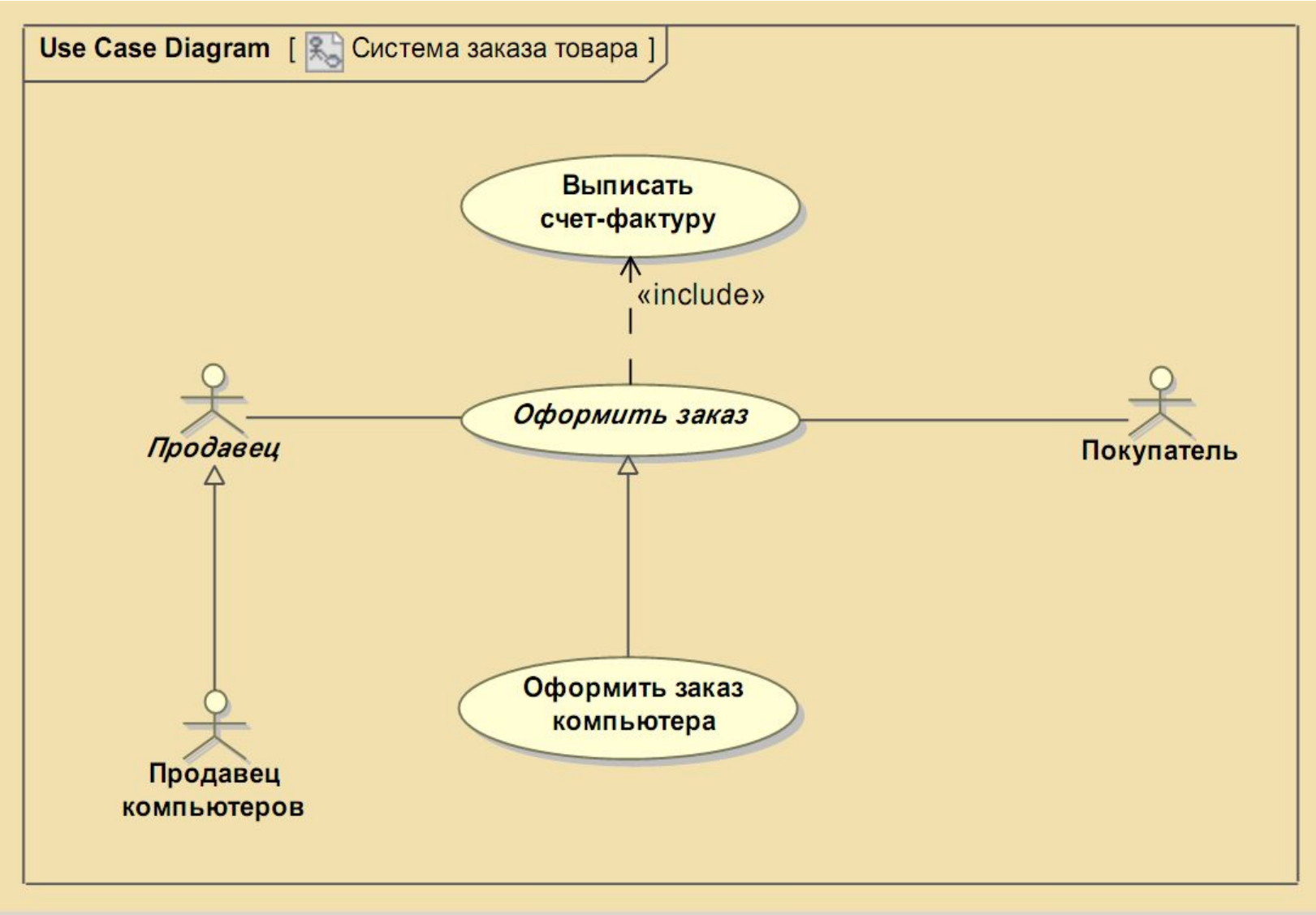


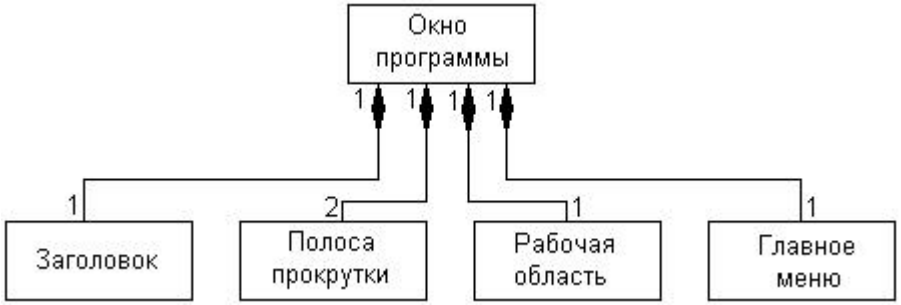
Диаграмма классов

Диаграмма классов представляет собой граф, вершинами которого являются элементы типа «классификатор», связанные различными типами структурных отношений.

Обобщение
(наследовани

Композиции

Сильная агрегация



Агрегациии

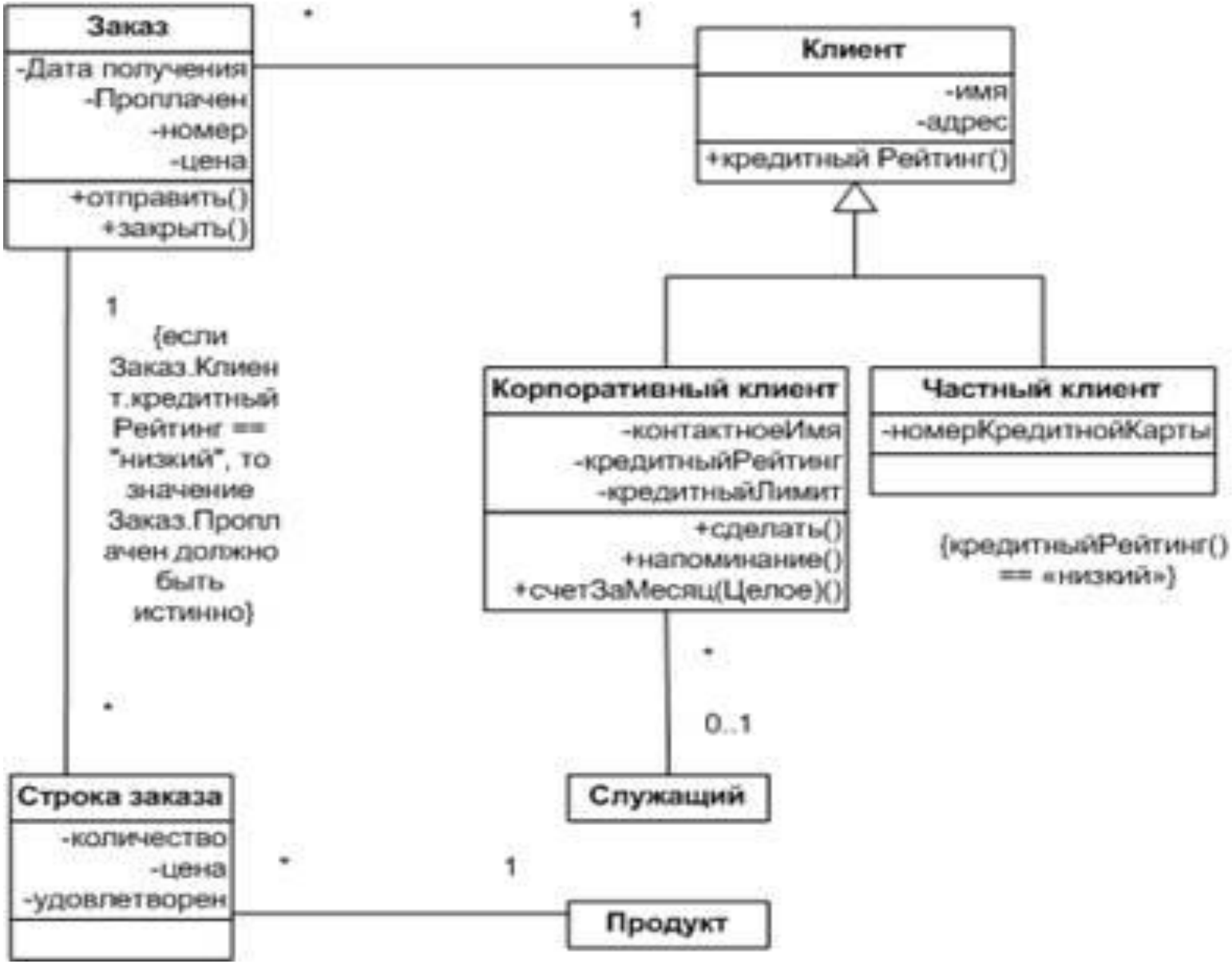
Ассоциаци
я

Отношения между экземплярами класса

Классификатор

Классификатор – это элемент, описывающий структурные и поведенческие свойства.

Пример диаграммы классов



Пример диаграммы классов

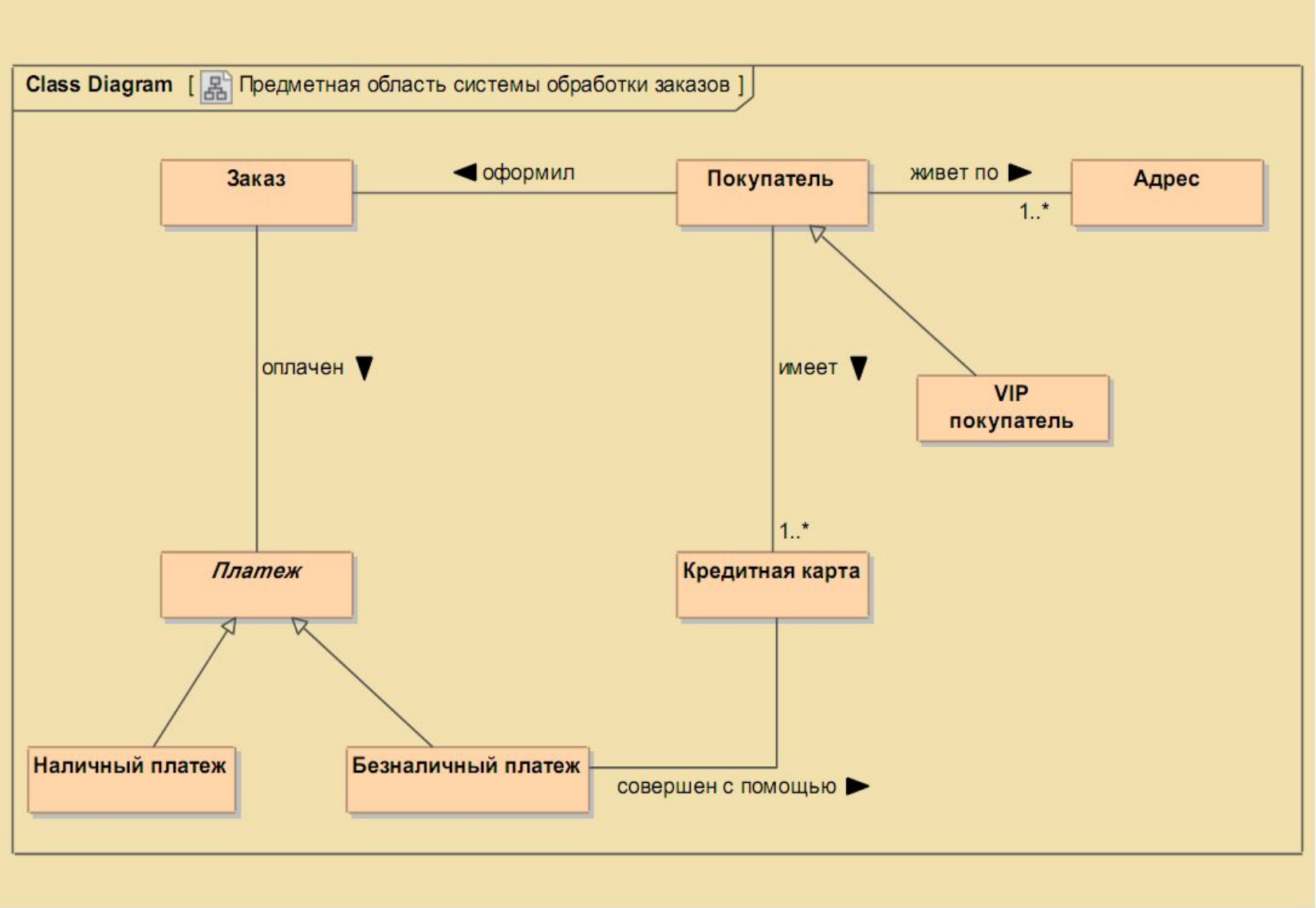


Диаграмма объектов

Диаграмма объектов представляет статический «моментальный снимок» с экземпляров предметов, которые находятся в диаграммах классов.

Диаграмма объектов характеризуется следующими свойствами:

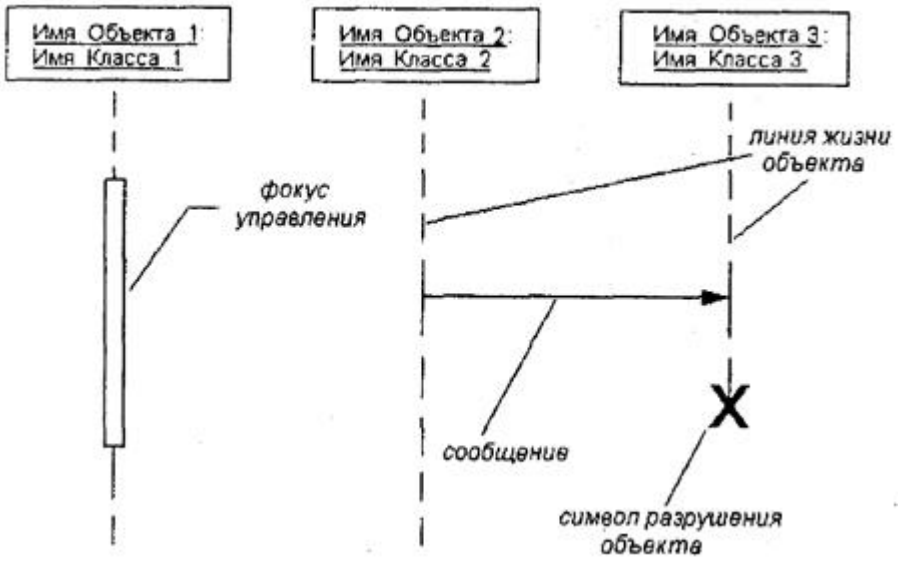
- акцентирует внимание на одном аспекте статического вида системы с точки зрения проектирования или процессов;
- представляет лишь один из кадров динамического сценария, показанного на диаграмме взаимодействия;
- содержит только существенные для понимания данного аспекта элементы;
- уровень ее детализации соответствует уровню абстракции системы. (Показывайте только те значения атрибутов и дополнения, которые существенны для понимания);

Пример диаграммы объектов

Диаграмма последовательности

Данный вид диаграмм отражает следующие аспекты проектируемой Системы:

- обмен сообщениями между объектами;
- ограничения, накладываемые на взаимодействие объектов;
- события, инициирующие взаимодействия объектов.



Элементы диаграммы последовательности

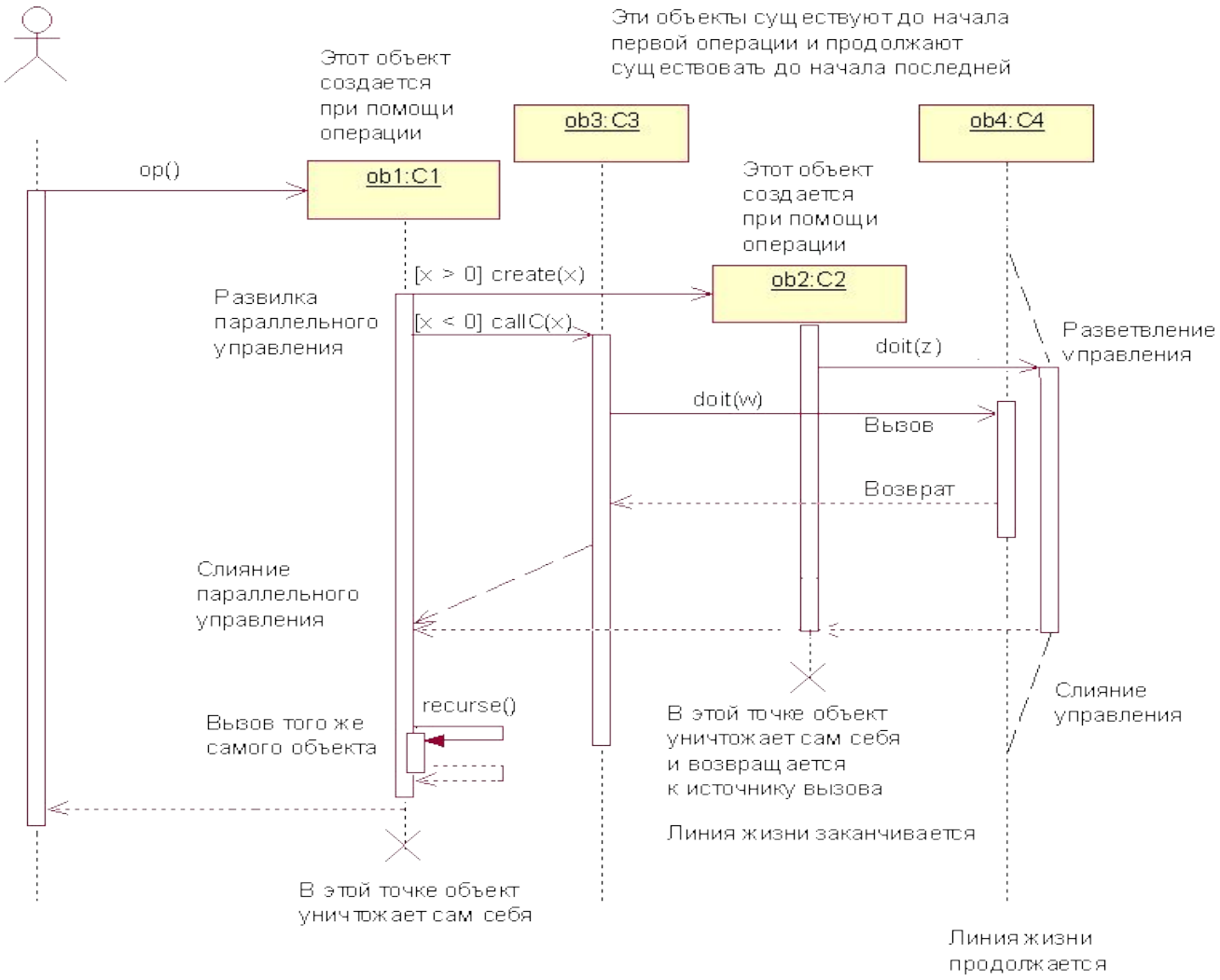


Рис. 162. Диаграмма последовательности с процедурным потоком управления

Пример диаграммы последовательности

Диаграмма коммуникаций

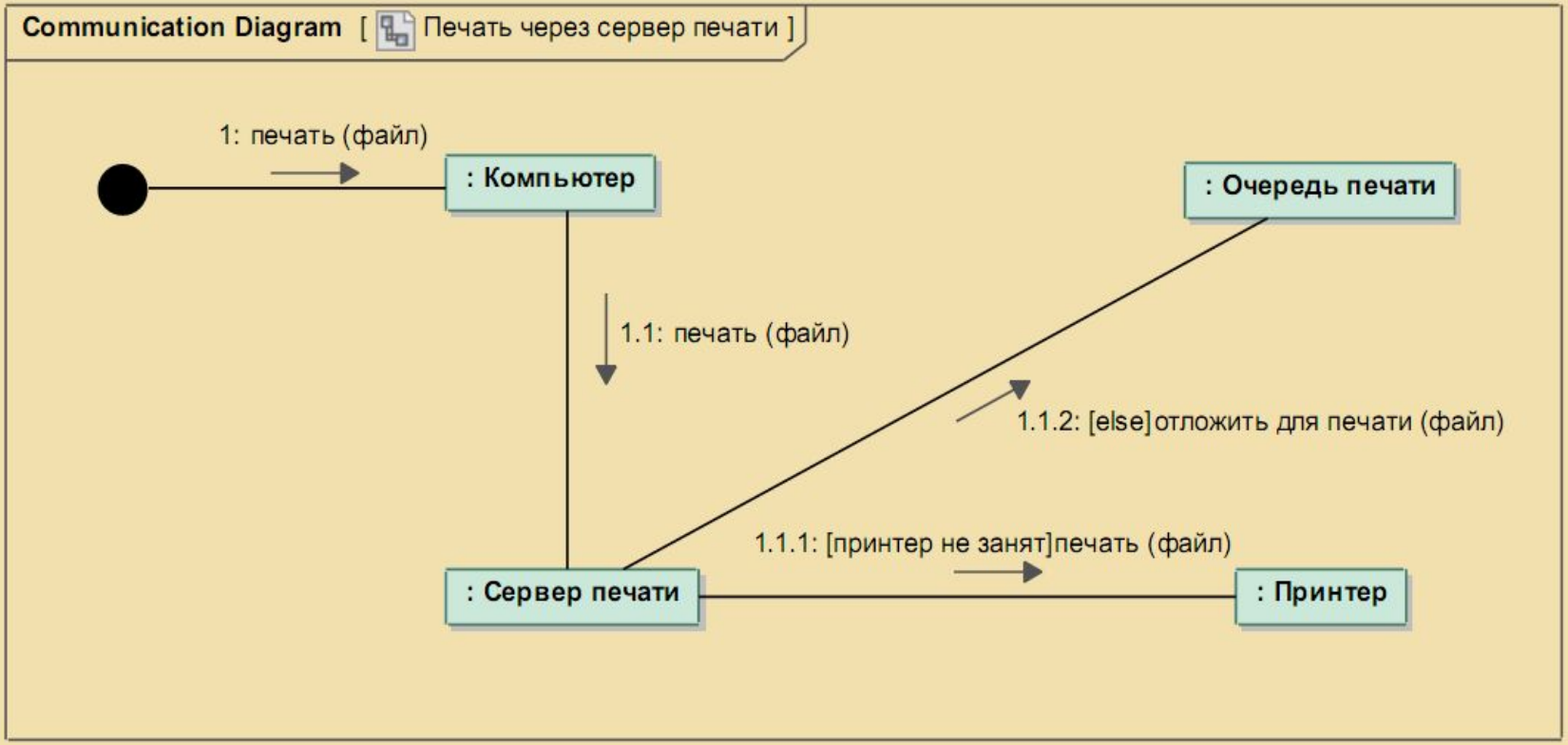


Диаграмма состояний

Диаграммы состояний показывают различные состояния объекта в течение его времени жизни

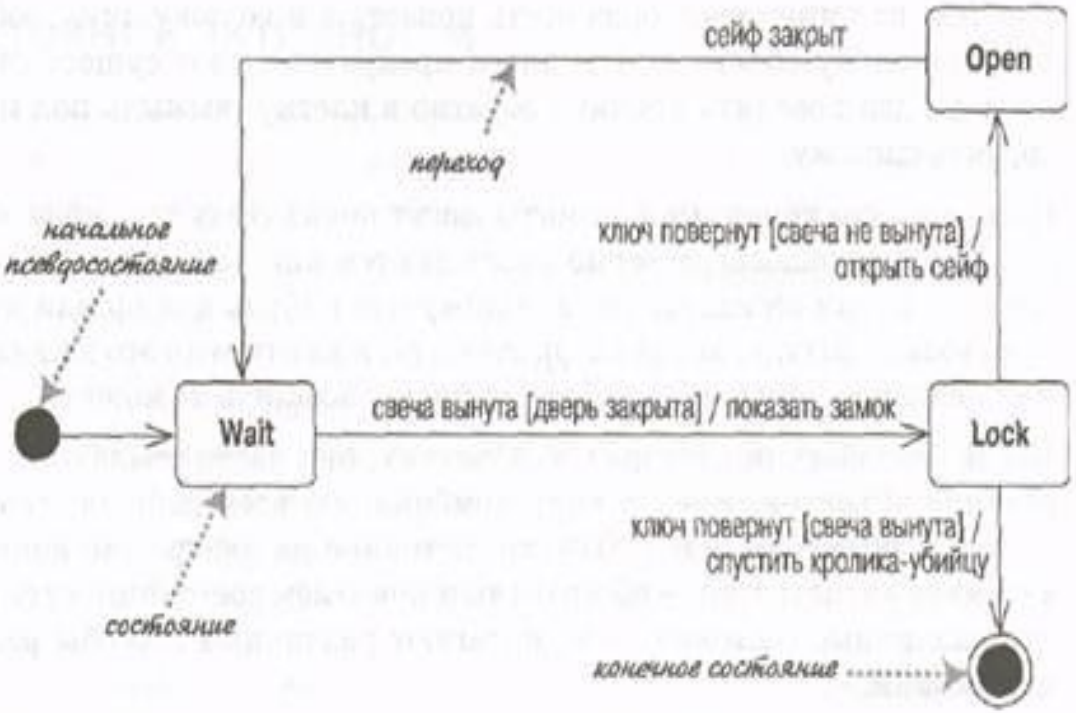


Рис. 10.1. Простая диаграмма состояний

Синхронизирующие состояния

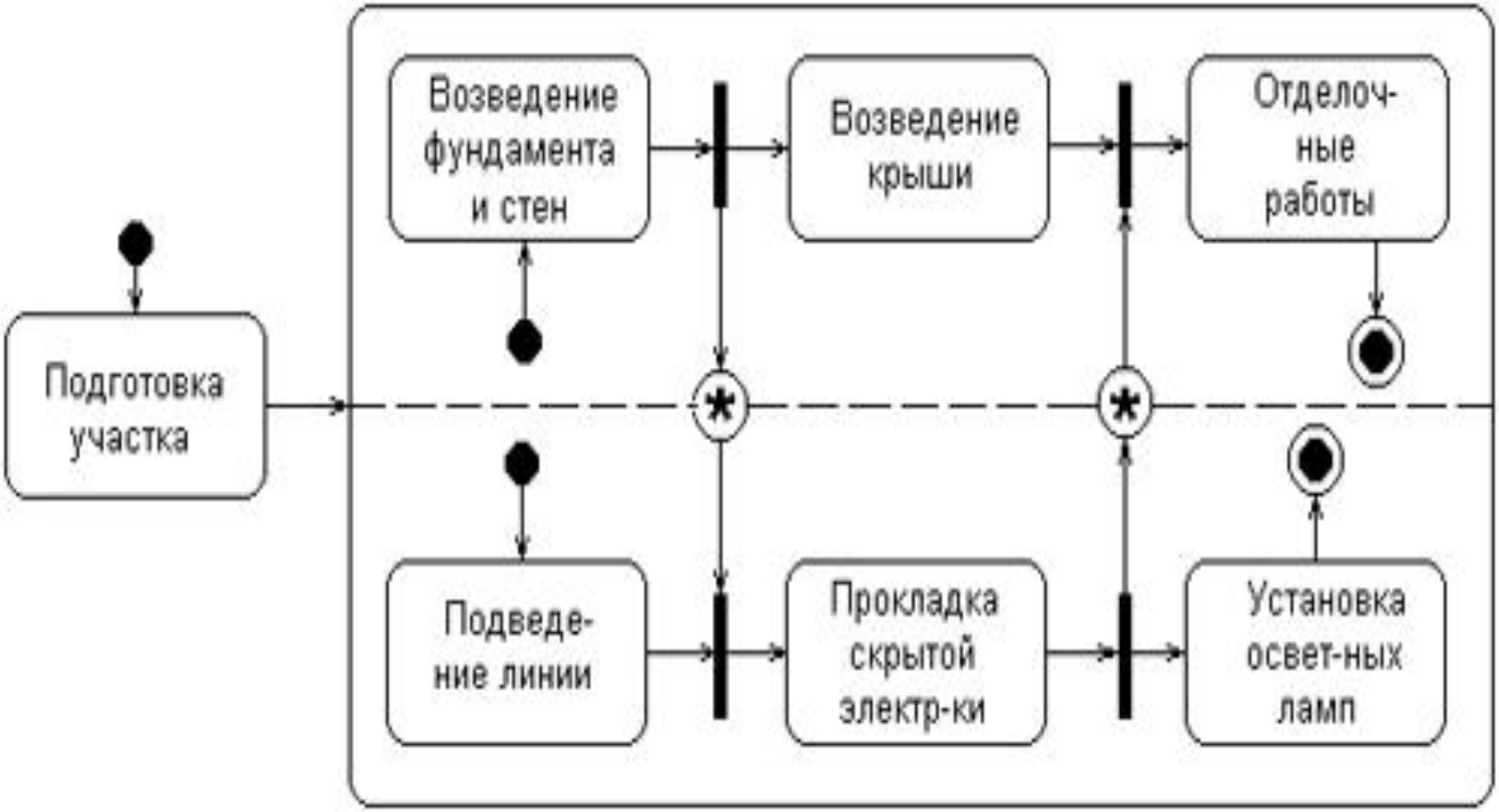
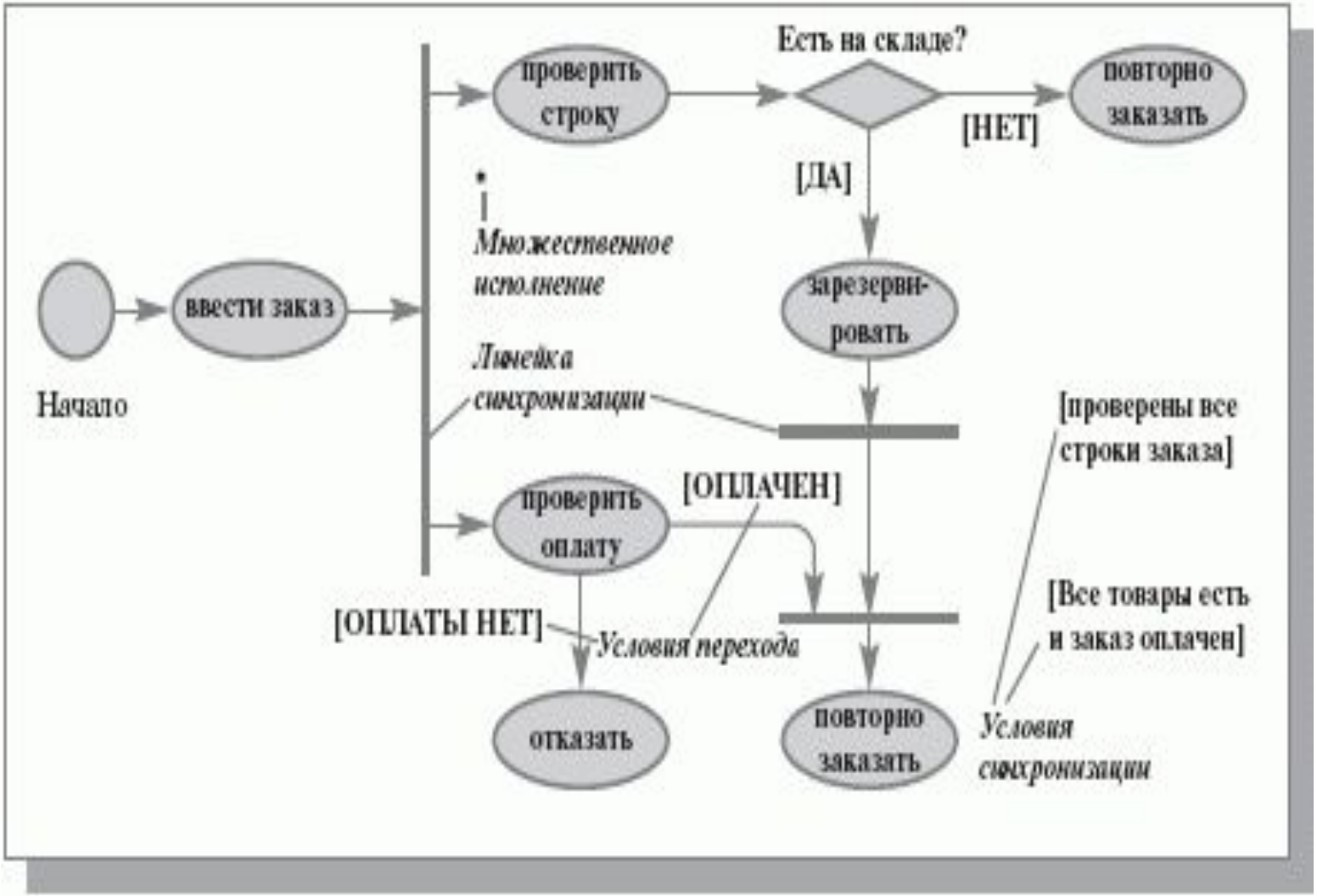


Диаграмма деятельности



Пример диаграммы деятельности

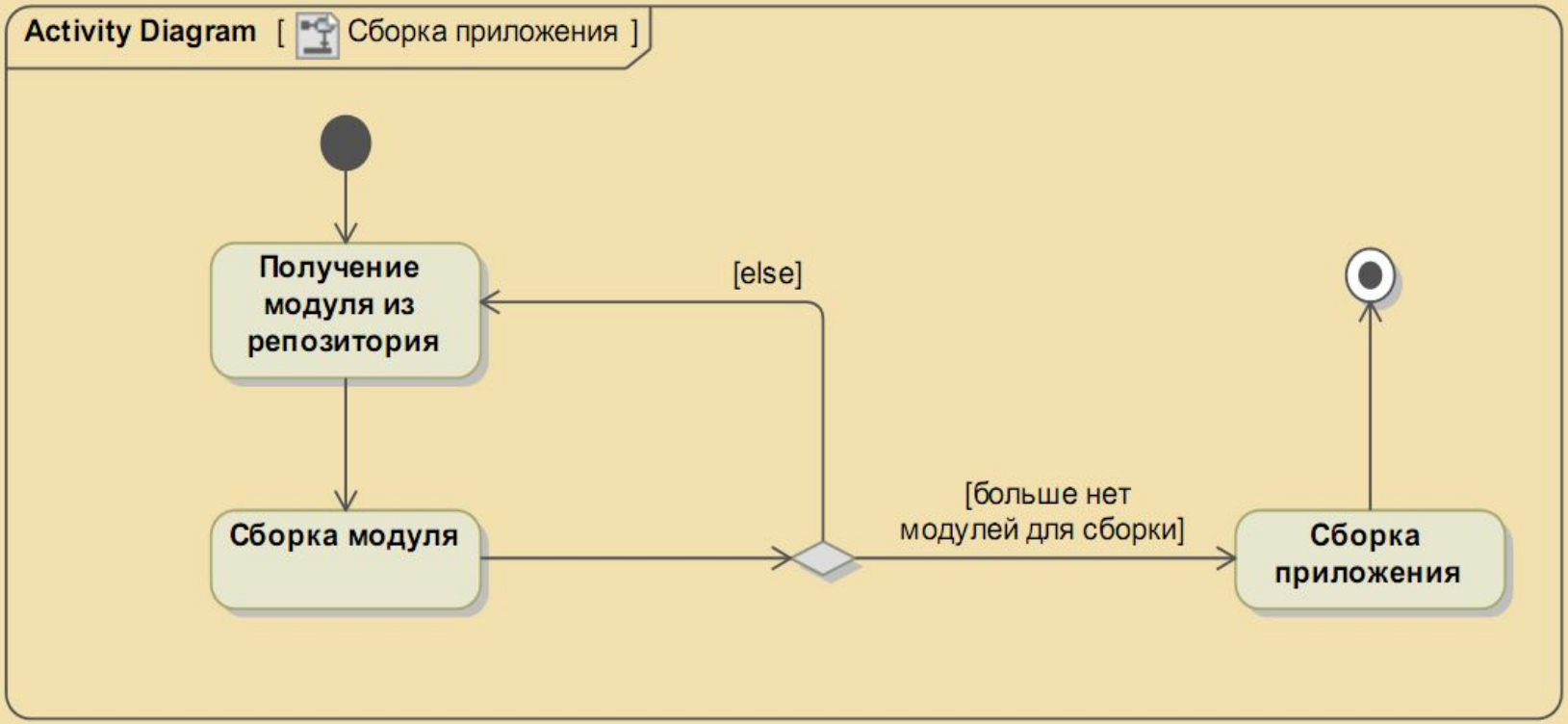


Диаграмма компонентов

Диаграмма компонентов описывает особенности физического представления системы.

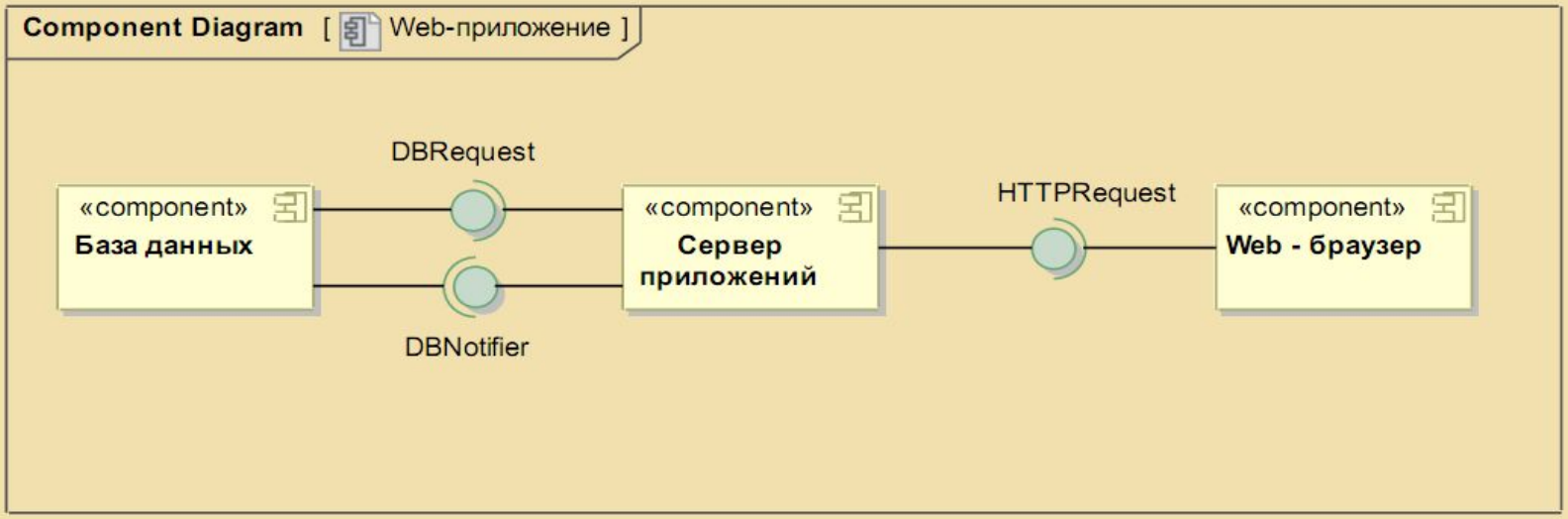


Диаграмма внутренней структуры

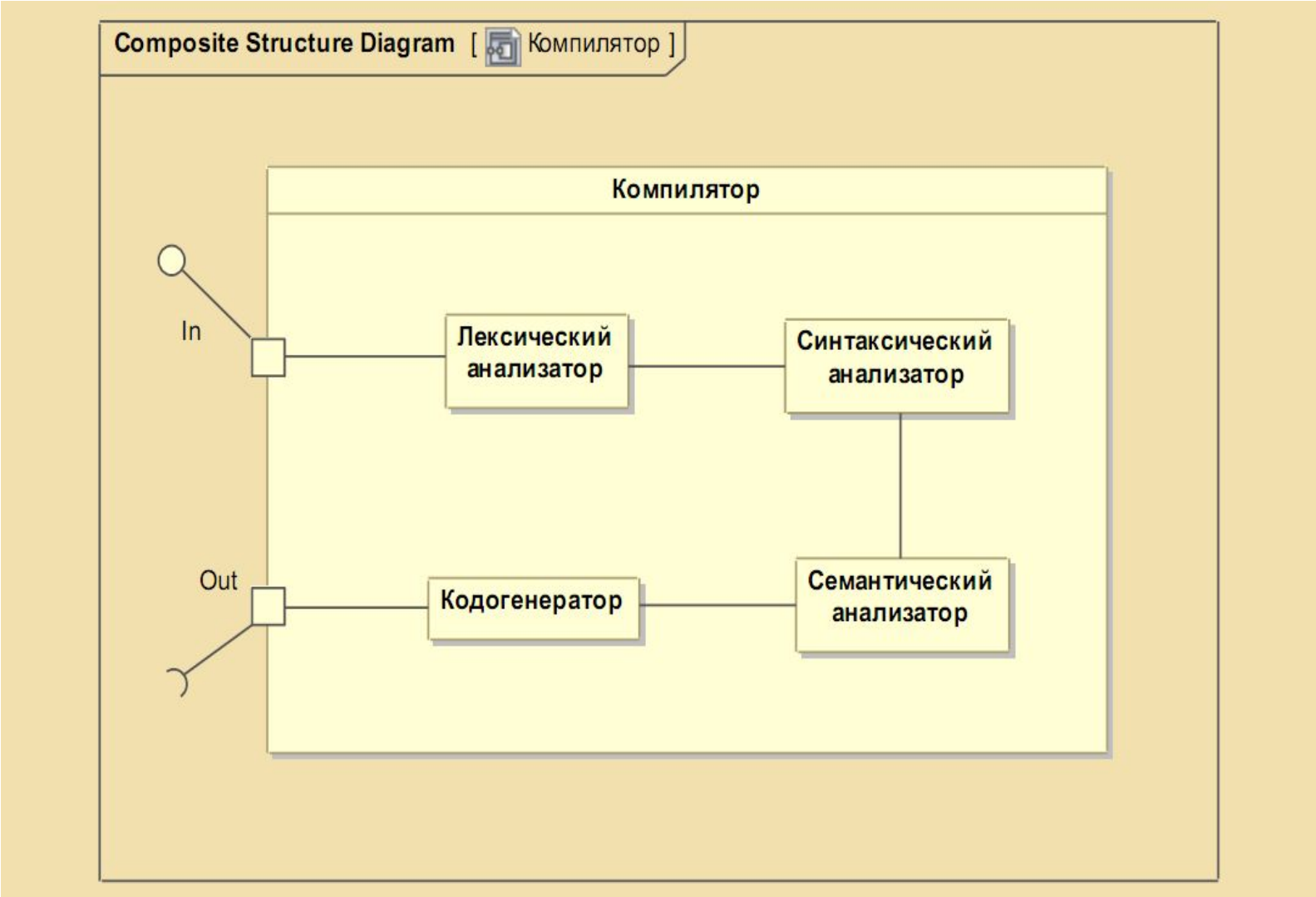


Диаграмма размещения

Диаграмма размещения наряду с отображением состава и связей элементов системы показывает, как они физически размещены на вычислительных ресурсах во время выполнения.

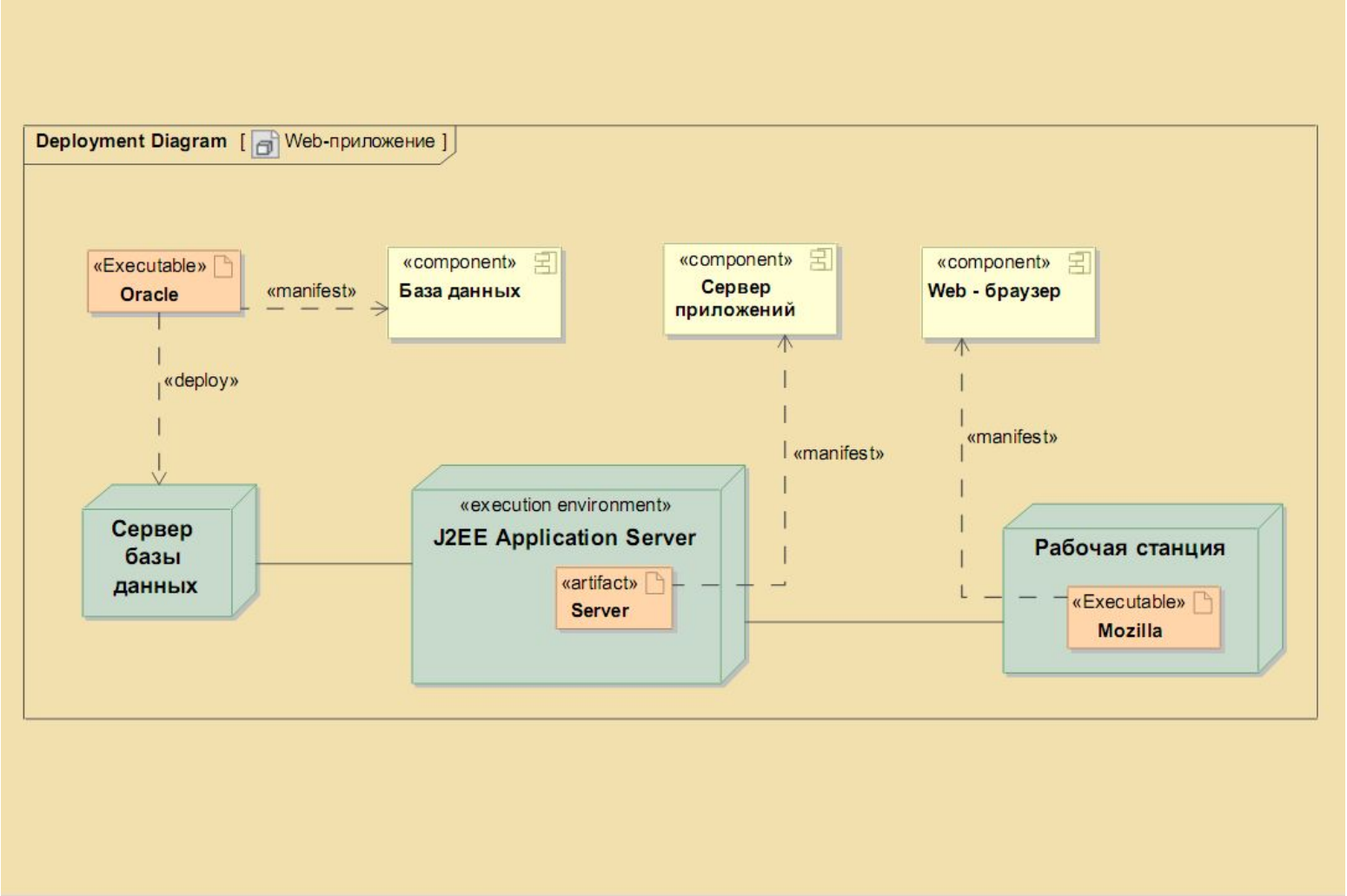
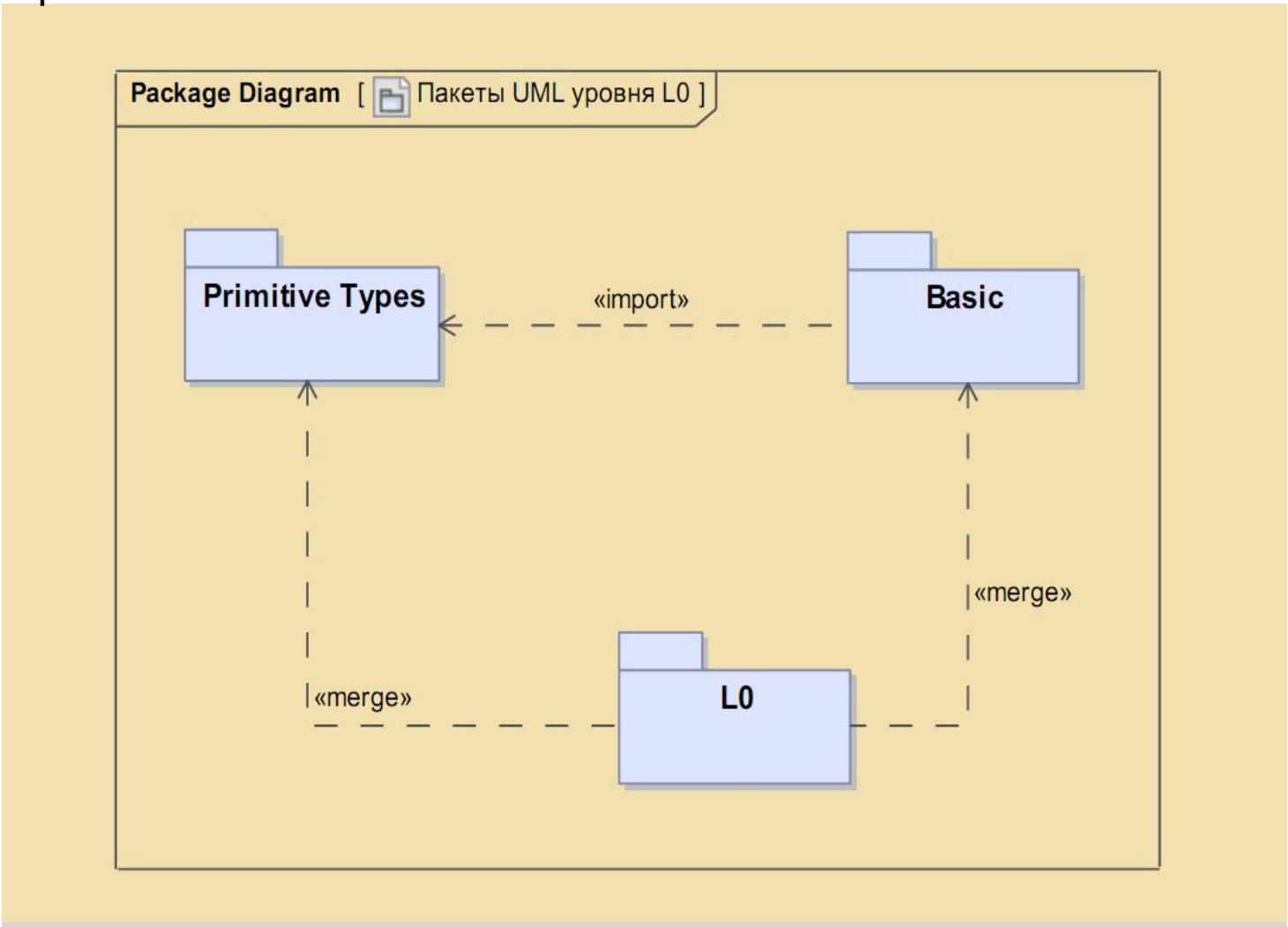


Диаграмма пакетов

Диаграммы пакетов отображают зависимости между пакетами: импортирование пакета и слияние пакета



```
public class HelloWorld
{
  public static void main(String[] args)
  {
    System.out.println("Hello, world!");
  }
}
```

HelloWorld.java

Приложение Hello, World

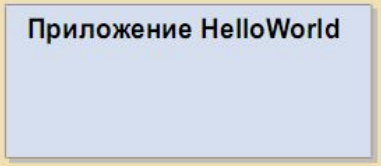
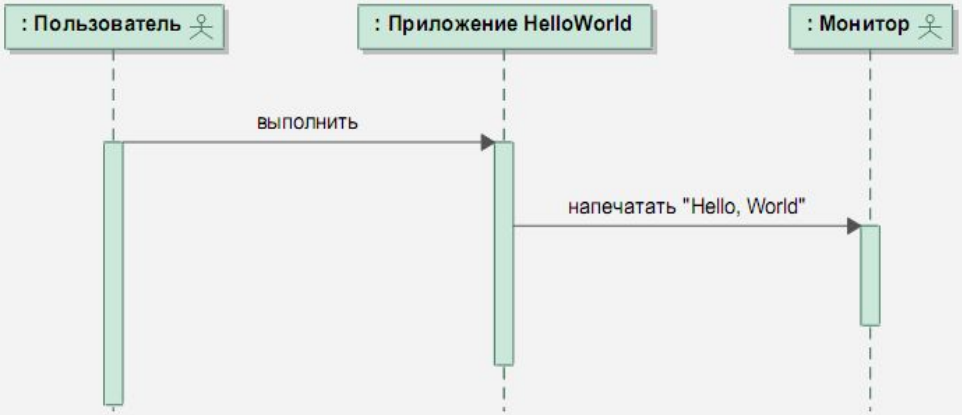
```
> javac ./HelloWorld.java
> java HelloWorld
```

HelloWorld.java

⬇

HelloWorld.class

Уровень системы

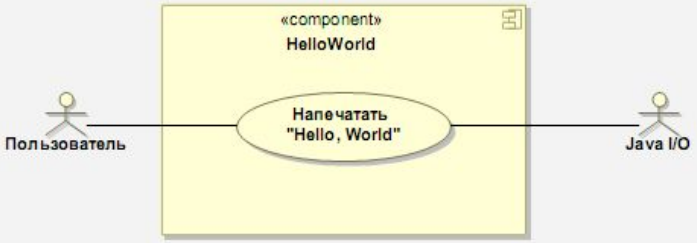
Использование	
Структура	
Поведение	

Уровень модуля

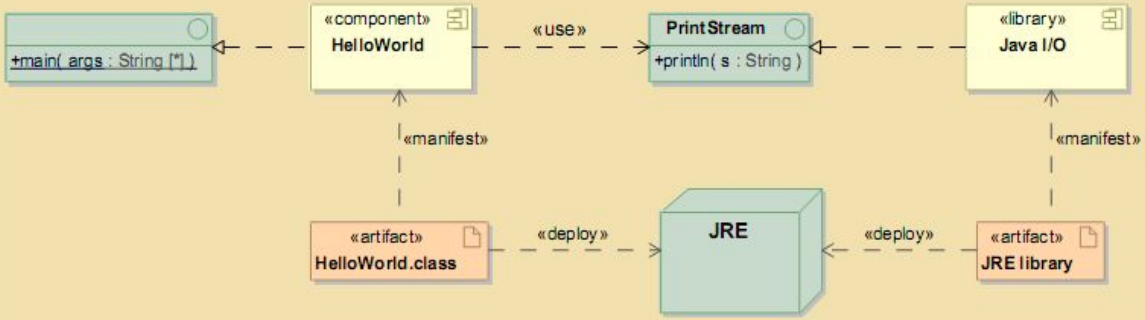
Использование	«component» Приложение HelloWorld; Пользователь; Напечатать "Hello, World"; Библиотека ввода-вывода
Структура	«component» Приложение HelloWorld; «component» Библиотека ввода-вывода
Поведение	: Пользователь; : Приложение HelloWorld; : Библиотека ввода-вывода; выполнить(); напечатать("Hello, World")

Уровень модулей Java

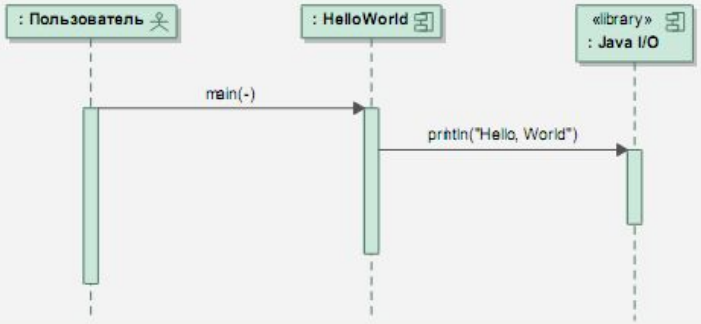
Использование



Структура

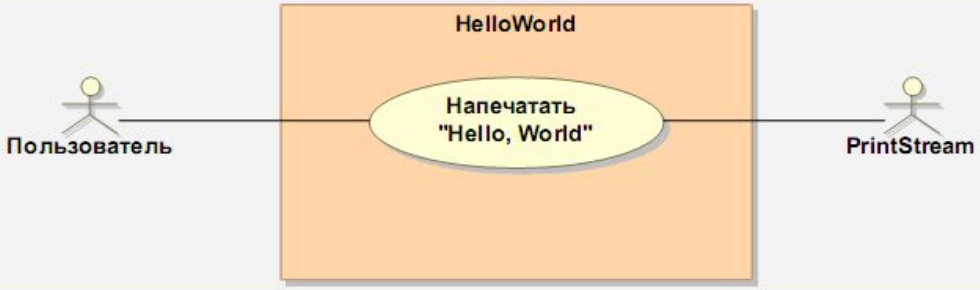


Поведение

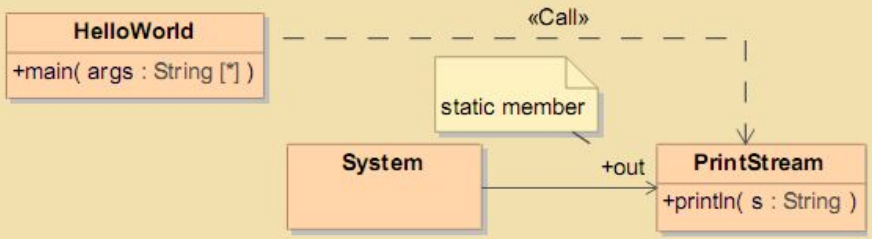
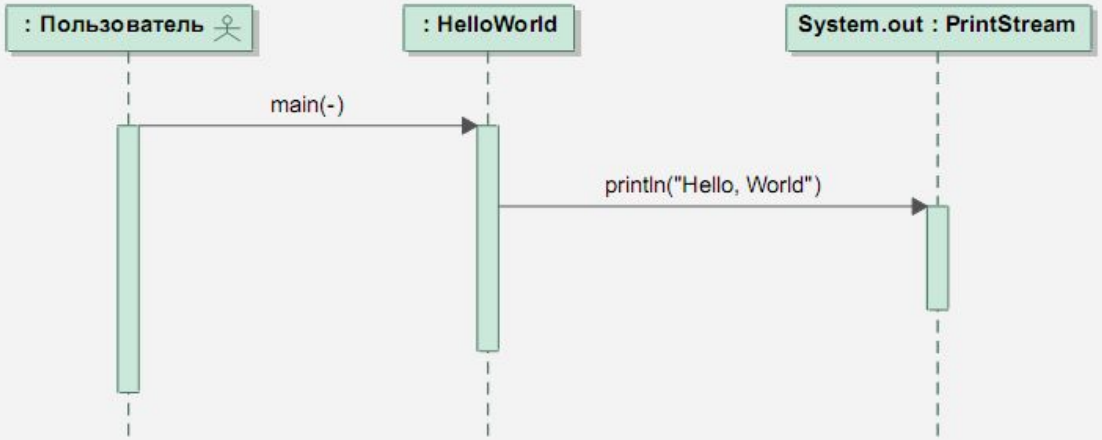


Уровень объектов Java

Использование



Структура



Поведение