*Занятие № 15*

*Номер учебной группы:* П-16

*Фамилия, инициалы учащегося:* Язубец Е.В.

*Дата выполнения работы:* 24.11.2022

*Тема работы:* «Разработка диаграмм взаимодействия»

*Результат выполнения работы*

**Задание 1**

Изучил теоретический материал по теме «Построение диаграмм классов в Rational Rose».

**Задание 2**

Изучил теоретический материал по теме «Построение диаграмм объектов в Rational Rose».

**Задание 3**

Построил диаграммы классов и объектов для своего варианта задания в Rational Rose.

**Диаграмма классов(рис-1)**

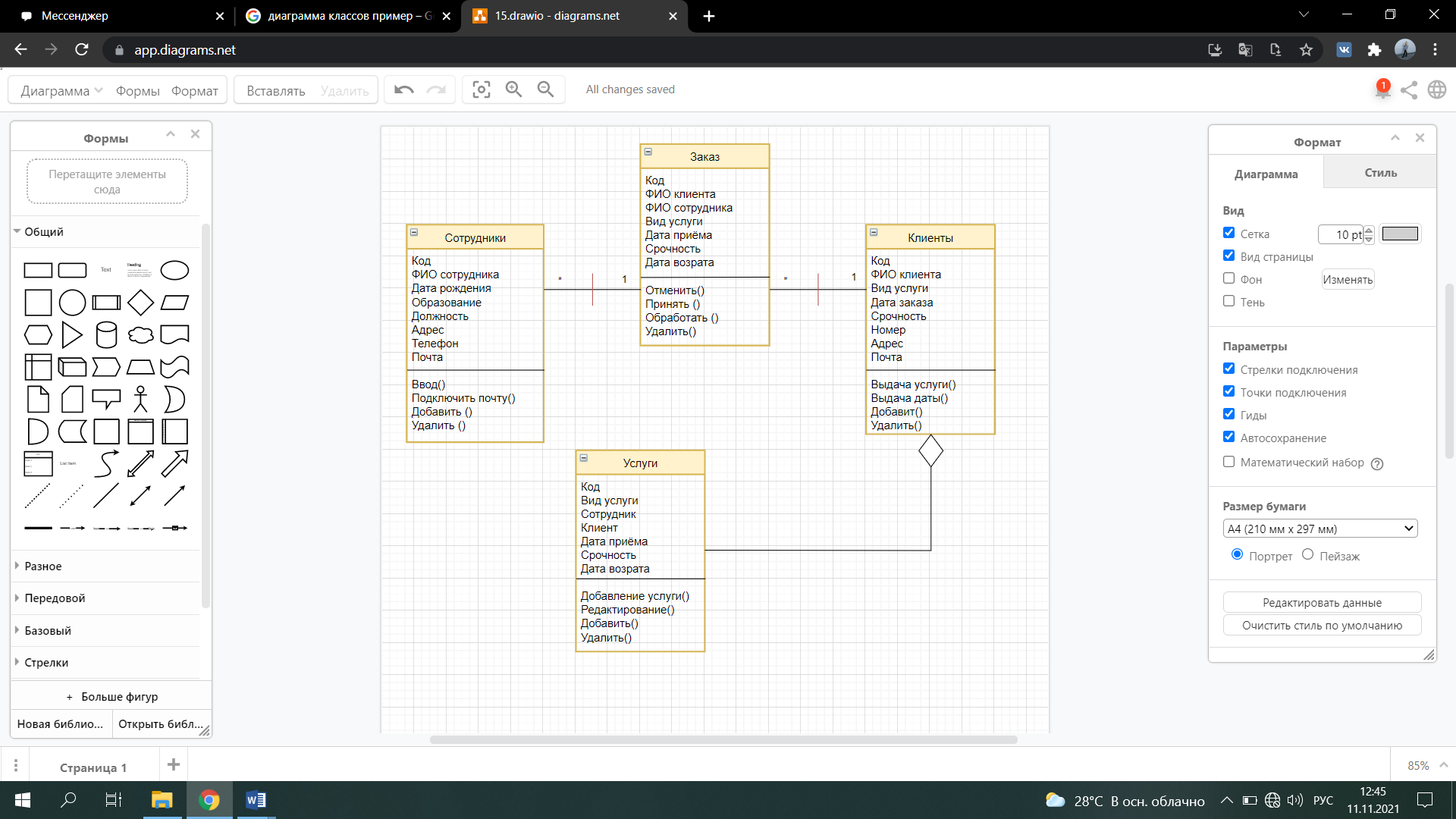


Рисунок-1

На рисунке-1 изображена диаграмма классов, выполненная в Rational Rose. В диаграмме отображена работа программного продукта. Она определяет типы классов системы и различного рода статические связи, которые существуют между ними.

На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами.

Вид и интерпретация диаграммы классов существенно зависит от точки зрения (уровня абстракции): классы могут представлять сущности предметной области (в процессе анализа) или элементы программной системы (в процессах проектирования и реализации).

Данная диаграмма содержит 2 ассоциации и 1 агрегацию.

**Диаграмма объектов(рис-2)**

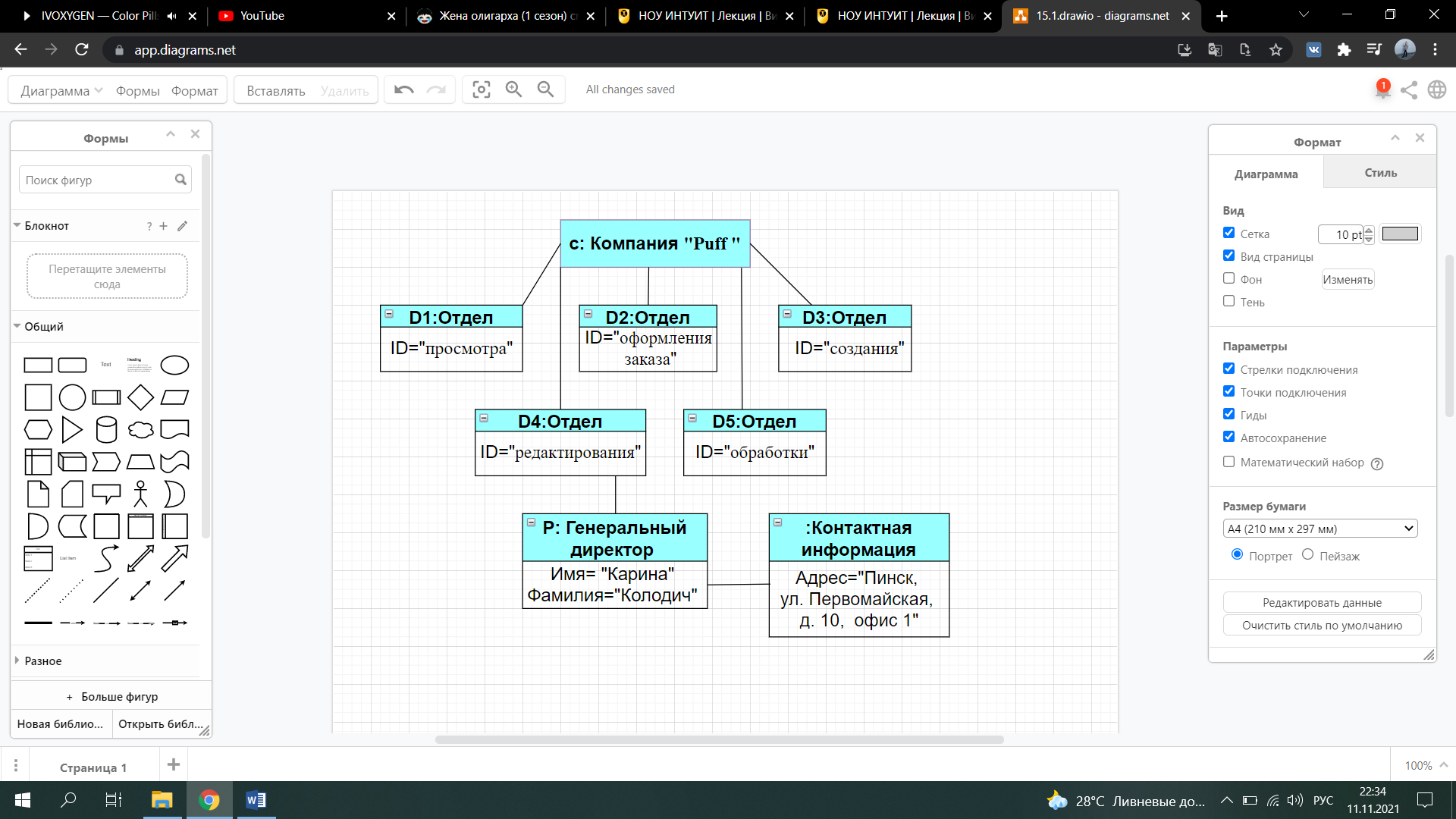


Рисунок-2

На рисунке-2 изображена диаграмма объектов, выполненная в Rational Rose. В диаграмме отображена работа программного продукта. Диаграммы объектов являются производными от диаграмм классов, поэтому диаграммы объектов зависят от диаграмм классов.

Диаграммы объектов представляют собой экземпляр диаграммы классов. Основные понятия одинаковы для диаграмм классов и диаграмм объектов. Диаграммы объектов также представляют статическое представление системы, но это статическое представление представляет собой снимок системы в определенный момент.

Диаграммы объектов используются для визуализации набора объектов и их отношений в качестве экземпляра.

На данной диаграмме представлены отделы и генеральный директор, контактная информация для компании.

**Задание 4**

Ответил на контрольные вопросы.

1. Каково назначение диаграмм классов?

**Ответ:** Целью создания диаграммы классов является графическое представление статической структуры декларативных элементов системы (классов, типов и т. п.) Она содержит в себе также некоторые элементы поведения (например — операции), однако их динамика должна быть отражена на диаграммах других видов (диаграммах коммуникации, диаграммах состояний).

2. Для чего используется диаграмма классов на стадии анализа?

**Ответ:** На стадии анализа мы используем диаграммы классов, чтобы выделить общие роли и обязанности сущностей, обеспечивающих требуемое поведение системы. На стадии проектирования мы пользуемся диаграммой классов, чтобы передать структуру классов, формирующих архитектуру системы.

3. Для чего используется диаграмма классов на стадии проектирования?

**Ответ:** На стадии проектирования диаграммы классов используются, чтобы передать структуру классов, формирующих архитектуру системы.

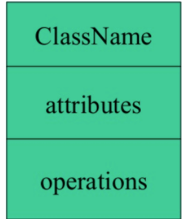
Каждый класс должен иметь имя; если имя слишком длинно, его можно сократить или увеличить сам значок на диаграмме. Имя каждого класса должно быть уникально в содержащем его проекте.

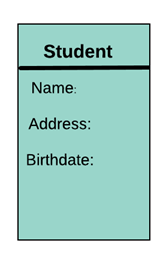
4. Назовите основные компоненты диаграмм классов.

**Ответ:** Основные элементы диаграммы классов UML:

1. Имя класса
2. Атрибуты
3. операции

**Имя класса**

[](https://coderlessons.com/wp-content/uploads/images/gur/fa491c3ffcb27ffd10c46f67917032dc.png) Имя класса требуется только в графическом представлении класса. Появляется в самом верхнем отсеке. Класс — это план объекта, который может иметь одинаковые отношения, атрибуты, операции и семантику. Класс отображается в виде прямоугольника, включая его имя, атрибуты и операции в отдельных отсеках.

[](https://coderlessons.com/wp-content/uploads/images/gur/db1113b03432a78cc1aafe959ca75eef.png)**Атрибуты:**

Атрибут именуется свойством класса, который описывает моделируемый объект. На диаграмме классов этот компонент расположен чуть ниже отсека имени.

Атрибуты характеристики

* Атрибуты обычно записываются вместе с фактором видимости.
* Публичная, частная, защищенная и пакетная — это четыре видимости, которые обозначаются знаками +, -, # или ~ соответственно.
* Видимость описывает доступность атрибута класса.
* Атрибуты должны иметь осмысленное имя, которое описывает его использование в классе.

5. Назовите основные типы статических связей между классами.

**Ответ:** Существует четыре типа связей в UML:

* Зависимость
* Ассоциация
* Обобщение
* Реализация

Эти связи представляют собой базовые строительные блоки для описания отношений в UML, используемые для разработки хорошо согласованных моделей.

Первая из них – *зависимость* – семантически представляет собой связь между двумя элементами модели, в которой изменение одного элемента (независимого) может привести к изменению семантики другого элемента (зависимого).  
*Ассоциация* – это структурная связь между элементами модели, которая описывает набор связей, существующих между объектами.

Третья связь – *обобщение* – выражает специализацию или *наследование*, в котором специализированный элемент (потомок) строится по спецификациям обобщенного элемента (родителя).

Четвертая – *реализация* – это семантическая связь между классами, когда один из них (поставщик) определяет соглашение, которого второй (клиент) обязан придерживаться.

6. Что такое признак видимости?

**Ответ:** Синтаксис UML для свойств классов (в отдельных программных средствах, например, в IBM UML Modeler, порядок записи параметров может быть иным):

<признак видимости> <имя атрибута>: <тип данных>

            = <значение по умолчанию>

<признак видимости> <имя операции> <(список аргументов)>

Видимость свойства указывает на возможность его использования другими классами. Один класс может "видеть" другой, если тот находится в области действия первого и между ними существует явное или неявное отношение. В языке UML определены три уровня видимости:

* public (общий) — любой внешний класс, который "видит" данный, может пользоваться его общими свойствами. Обозначаются знаком " + " перед именем атрибута или операции;
* protected (защищенный) — только любой потомок данного класса может пользоваться его защищенными свойствами. Обозначаются знаком " # ";
* private (закрытый) — только данный класс может пользоваться этими свойствами. Обозначаются символом " - " .