**Создание онтологии в редакторе Protégé.**

**Рекомендации для практического занятия.**

# Оглавление

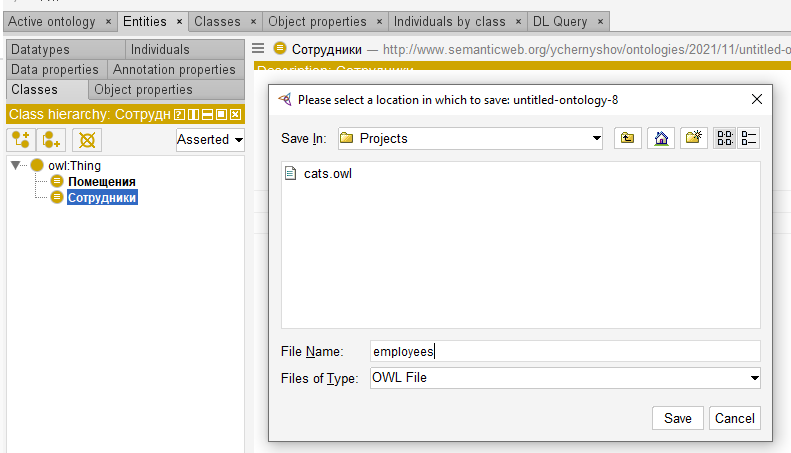
[Общие сведения 1](#_Toc91079021)

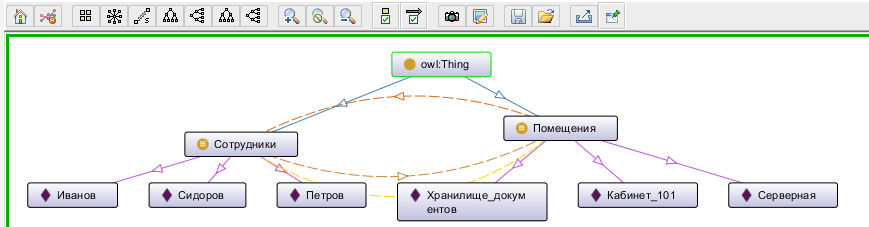
[Установка 2](#_Toc91079022)

[Создание проекта. 2](#_Toc91079023)

[Практические задачи для решения в Protege 3](#_Toc91079024)

[Доступ сотрудников в помещения. 3](#_Toc91079025)

[ 8](#_Toc91079026)

[ 9](#_Toc91079027)

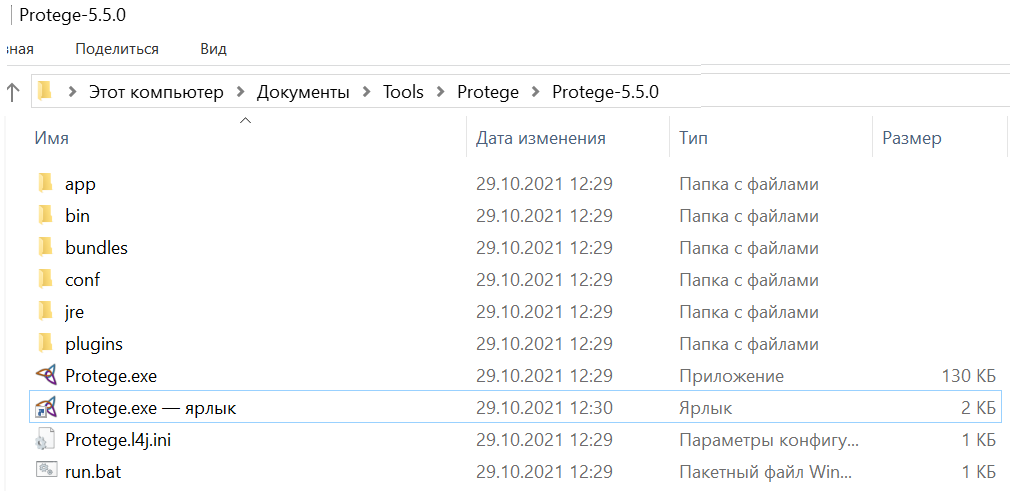
[«Принцесса или тигр» 9](#_Toc91079028)

# Общие сведения

Программа Protege для создания, редактирования и использования онтологий разработана в Стэнфордском университете, распространяется бесплатно, ее можно скачать себе на компьютер по ссылке <https://protege.stanford.edu/products.php>, либо воспользоваться web-версией <https://webprotege.stanford.edu/>

# Установка

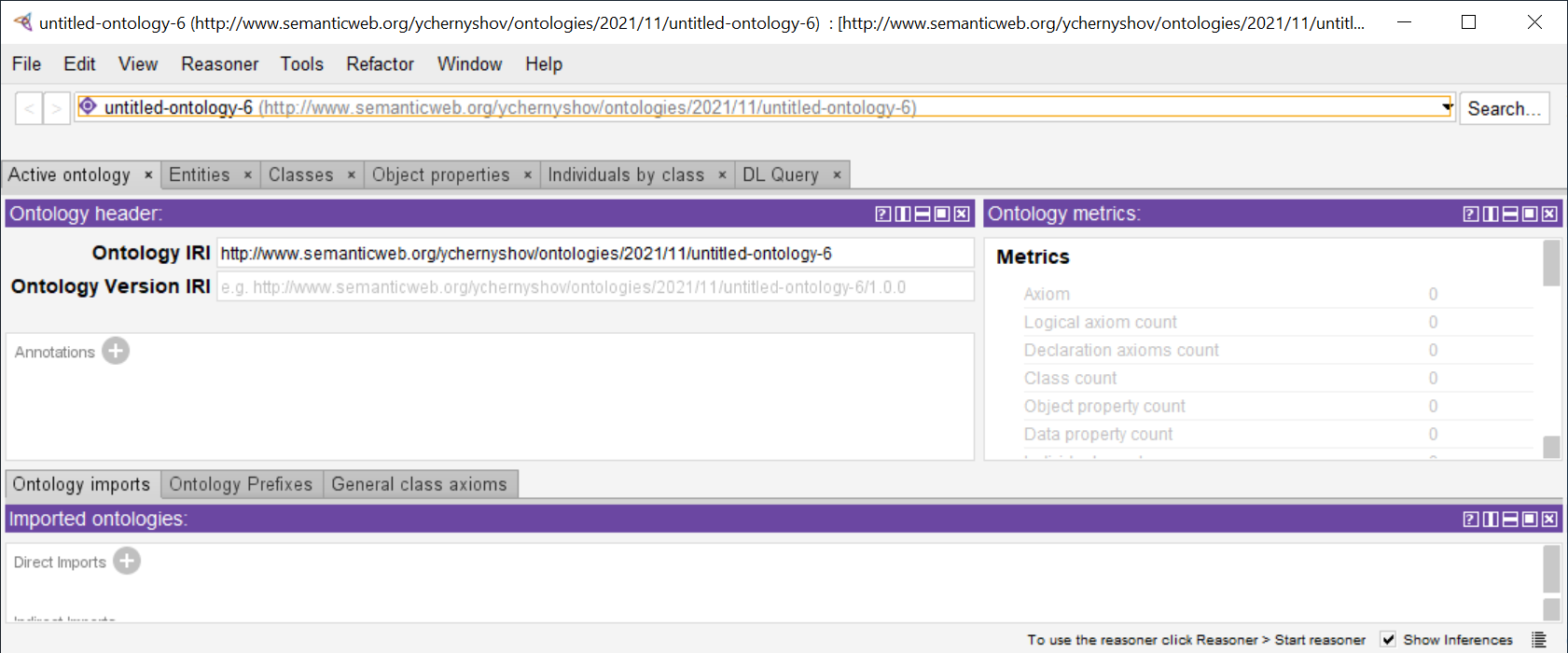
На официальной странице можно скачать архив для использования на локальном компьютере. Важно обратить внимание на операционную систему и разрядность процессора (64 или 32). Актуальная версия Protege-5.5, при необходимости использования 32-разрядной версии надо выбрать более низкую версию Protege, например 4.3. Файл архива выглядит следующим образом.



Можно разархивировать в любую папку и запустить файл Protégé.exe

# Создание проекта.

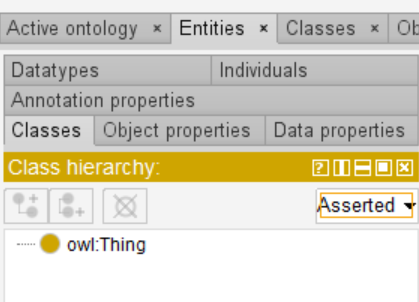
После запуска программы появится стартовое окно.



Для каждого проекта есть уникальный URI, который в Protege почему-то называется IRI. URI это Unique Resource Identifier, уникальный идентификатор ресурса в Интернет.

Protege позволяет записывать триплеты, то есть тройки вида «субъект-предикат-объект».

Раздел «Сущности» (Entities) позволяет описывать субъекты и объекты.



Важные вкладки в этом разделе:

* экземпляры классов (Individuals) это объекты классов, как в объектно-ориентированном программировании. Например, класс «Сервер», объект «prod-serv-002».
* свойства (object properties или data properties), похоже на свойства классов в ООП, однако в онтологии свойства имеют независимую природу, свойство не является неотделимым от класса (как в ООП). Свойство-центричная модель.

Для описанной онтологии можно применять ризонер, который дает предложения по найденным фактам (можно принять или отказаться).

Онтологию можно сохранить во внешний файл, в формате owl, который можно открыть в текстовом редакторе и увидеть, что это xml.

# Практические задачи для решения в Protege: доступ сотрудников в помещения

Рассмотрим ситуацию, в которой нам надо определить до какого помещения имеет доступ каждый из сотрудников. При этом нам известно, что

* Иванов не имеет доступа в серверную и кабинет 101
* Петров не имеет доступа в кабинет 101
* Сидоров не имеет доступа в хранилище документов.

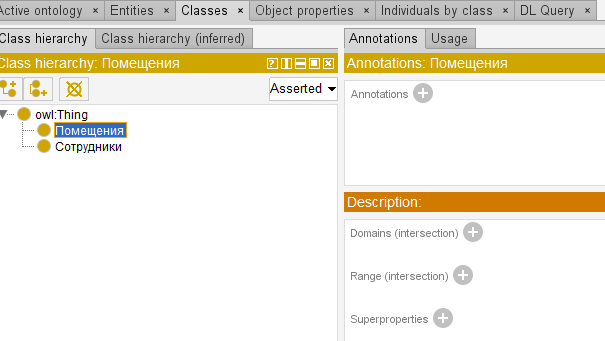
То есть у нас три сотрудника (Иванов, Петров, Сидоров) и три помещения (серверная, кабинет 101 и хранилище документов). Для однозначного решения задачи нам надо приподзакрыть рассматриваемый мир, то есть зафиксировать тот факт, что у нас нет других сотрудников и помещений и что каждый сотрудник имеет доступ ровно к одному помещению, в противном случае не удастся логически однозначно решить задачу. При введении таких строгих органичений решение задачи становится тривиальным, из первого условия сразу следует, что Иванов имеет доступ в хранилище документов (поскольку серверная и кабинет 101 исключаются), Петров (которому остались серверная и кабинет 101) имеет доступ в серверную, а Сидорову остался кабинет 101. Тем проще нам будет проверить результат работы ризонера в Protégé, которому мы и доверим решение этой задачи.

Запускаем редактор Protege

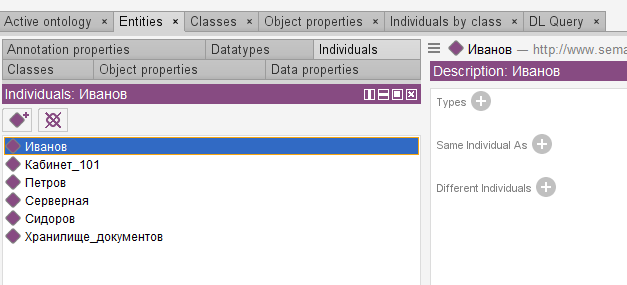


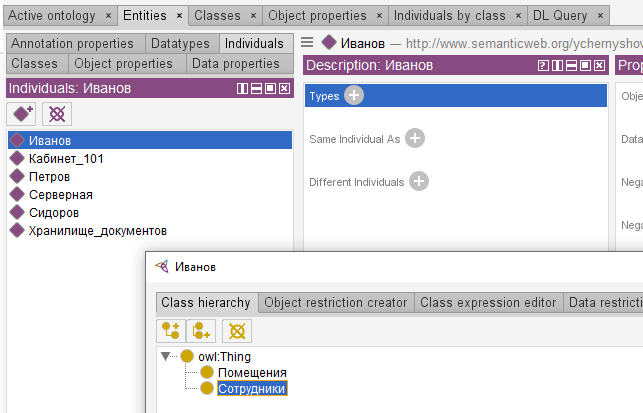
И в появившемся рабочем окне начнем создавать онтологию для нашей задачи

Сначала создадим два класса: «Сотрудники» и «Помещения».

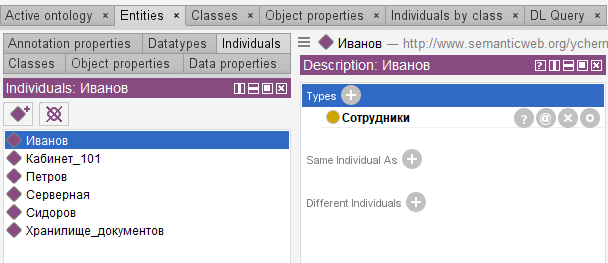


Затем создадим экземпляры этих классов и установим их свойства.

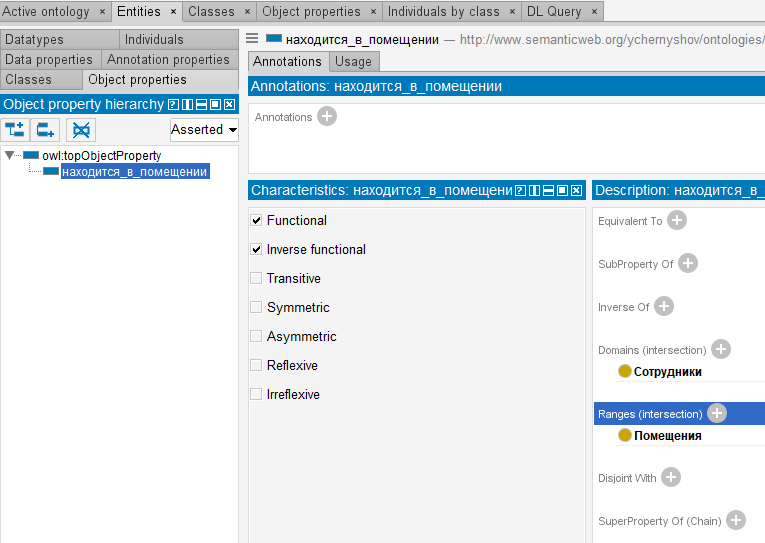


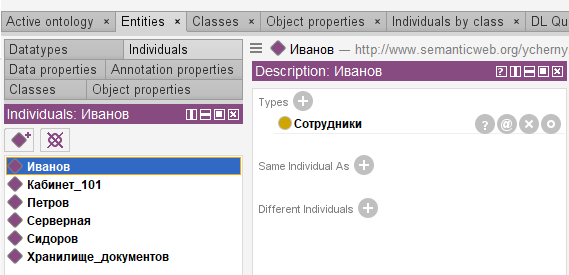


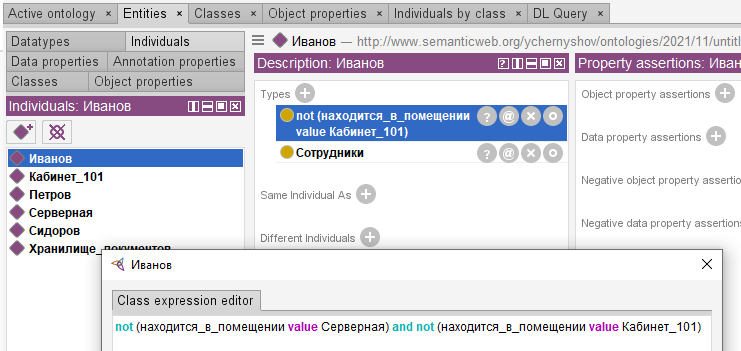
Аналогичную процедуру проделаем для остальных сотрудников (Петров, Сидоров) и помещений.



Теперь создадим свойство объектов «находится\_в\_помещении», которое будет связывать Сотрудников и Помещения. При создании свойства объекта (object property) необходимо указать сущности, которые связывает этот предикат. В нашем случае связываются Cотрудники (это пункт Domain) и Помещения (Ranges). Сотрудники могут находится в помещениях, а помещения могут содержать сотрудников, свойство является функциональным и действует из домена Сотрудники в область Помещения. С учетом физического смысла один сотрудник может находиться ровно в одном помещении, этот факт тоже учитывается в определении свойства «находится\_в\_помещении»: этому свойству устанавливаются параметры Functional и Inverse Functional, получается бинарное отношение «один-к-одному».

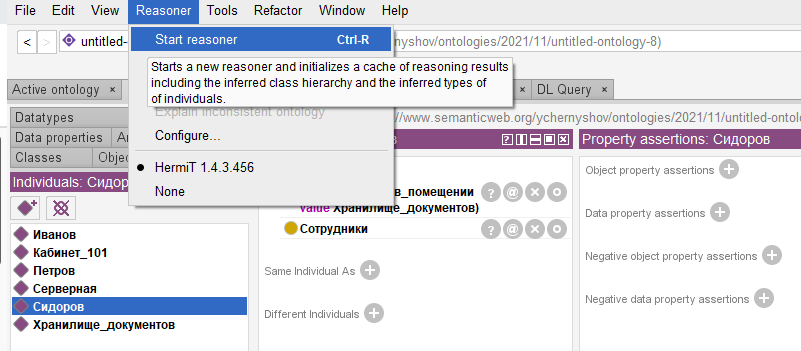


Теперь добавим в нашу онтологию информацию о том, что нам известно о местонахождении сотрудников. 

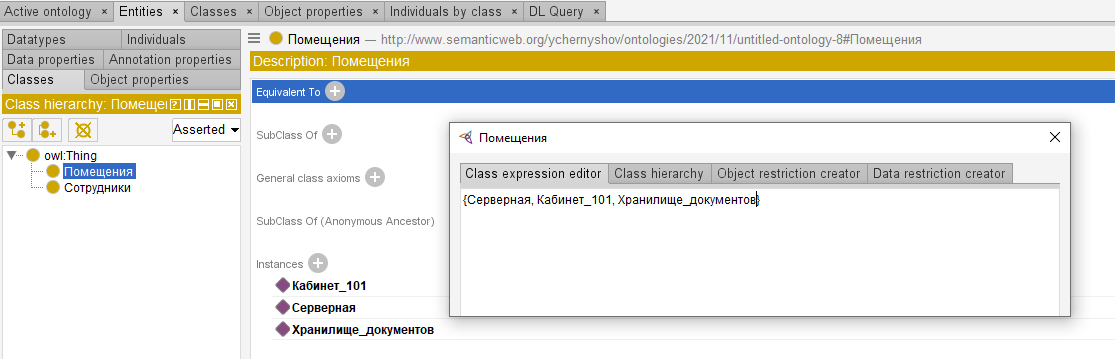


Аналогичную информацию добавляем о сотрудниках Петров и Сидиров, используя логические предикаты not и and.

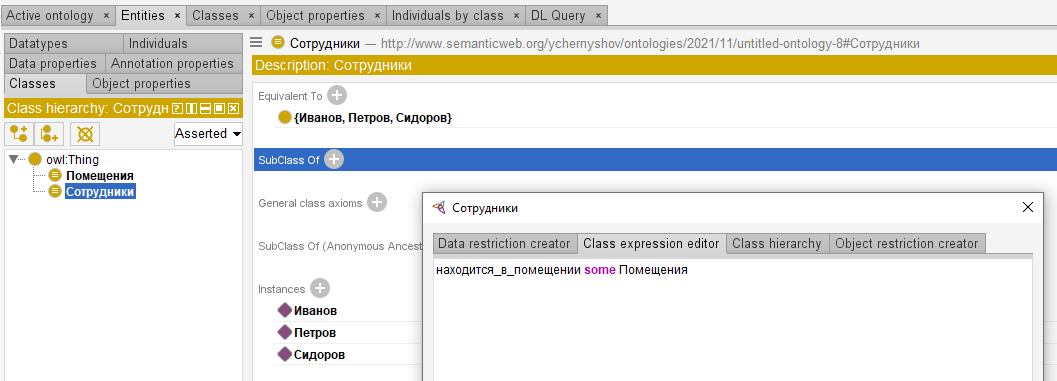
Мы можем попробовать запустить ризонер,



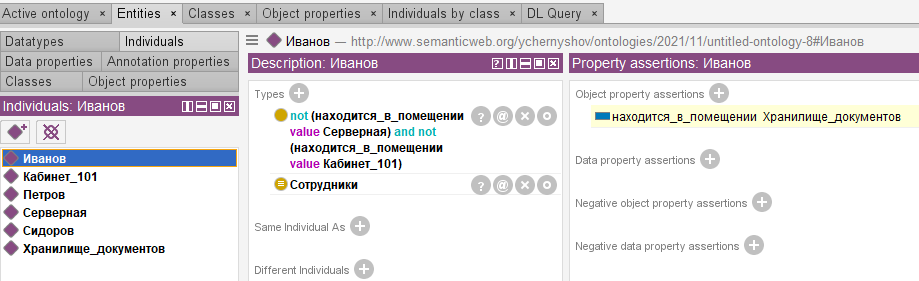
Однако каких-то существенных фактов извлечь не удастся, так как в текущей постановке у нас все еще сохраняется возможность добавления новых объектов (как сотрудников, так и помещений) в соответствии с гипотезой открытого мира. Чтобы ризонер смог однозначно разрешить задачу определения местонахождения сотрудников в помещениях нам надо «подзакрыть» классы Сотрудники и Помещения.



Для окончательного «закрытия мира» нашей онтологии необходимо в явном виде описать тот факт, что каждый сотрудник обязательно должен быть хотя бы в одной комнате.



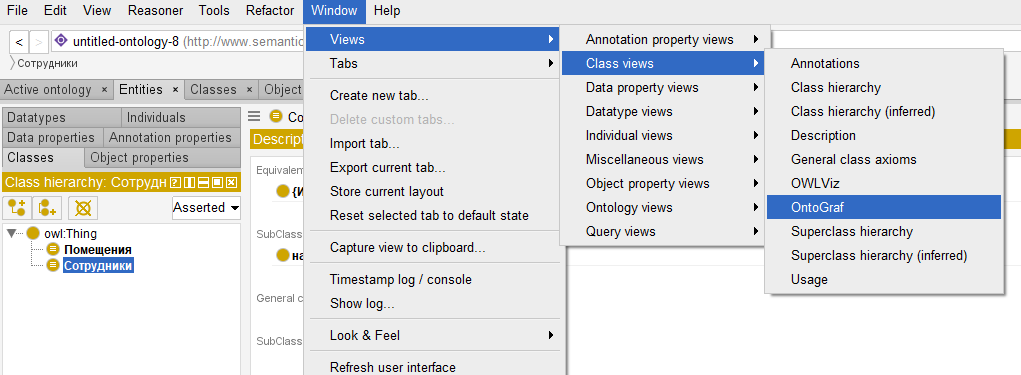
Запустив (или синхронизировав) ризонер, мы получим информацию о точном местонахождении сотрудников в конкретном помещении. Например, мы однозначно установили, что Иванов находится в хранилище документов.



Мы можем сохранить выбранную онтологию, например, в формате owl.

## 

В результате получим owl файл, этот файл можно открыто обычным текстовым редактором, изучить его содержание.



В результате мы получим хорошую визуализацию.

## 