**Глава 4**

**4.1 Создание онтологии в OWL**

В этой главе описывается создание онтологии пиццы. Мы используем эту предметную область, потому что в ней можно представить множество полезных примеров.

**Упражнение 2  Создайте новую онтологию OWL**

|  |  |
| --- | --- |
|  | 1.Запуск Protege 4 |
|  | 2.При появлении диалогового окна «Добро пожаловать в Protege» нажмите создать новые OWL онтологии |
|  | 3.Появится мастер «Создание URI онтологии». Каждая онтология создается с использованием уникального имени - идентификатора ресурса (URI). Замените значение по умолчанию на http://www.pizza.com/ ontologies/pizza.owl и нажмите «Далее». |
|  | 4.Вы можете сохранить ваши онтологии в файл на вашем компьютере. Вы можете просмотреть ваш жесткий диск и сохранить ваши онтологии в новый файл, вы можете назвать ваш файл 'pizza.owl'. Как только вы выбрали файл нажмите 'Finish'. |

После этого новый пустой файл Protege будет создан и появится окно «Active Ontology Tab», представленное на рисунке 4.1. Как видно из рисунка 4.1, «Active Ontology Tab» отображает информацию об онтологии. Например, онтология URI может быть изменена, аннотации на онтологии, такие как комментарии, могут быть добавлены и отредактированы, а также имена и импорт можно настроить с помощью этой вкладки.

☺- Protege 4.2 создает новый проект при запуске, опуская описанные выше действия, характерные для более ранних версий Protege 4. Можно сразу приступить к созданию онтологии. Но лучше сначала сделать Save\_as и указать тип, имя и место хранения вашего проекта. В дальнейшем периодически сохранять проект, выбирая save на главной панели меню. Это необходимо делать, так как неудачные изменения могут привести к непредсказуемым последствиям.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Онтология, что мы создадим, основана на онтологии пиццы, которая была использована в качестве основы для курса по редактированию DAML + OIL онтологий в (http://oiled.man.ac.uk), которые изучаются в университете Манчестера**

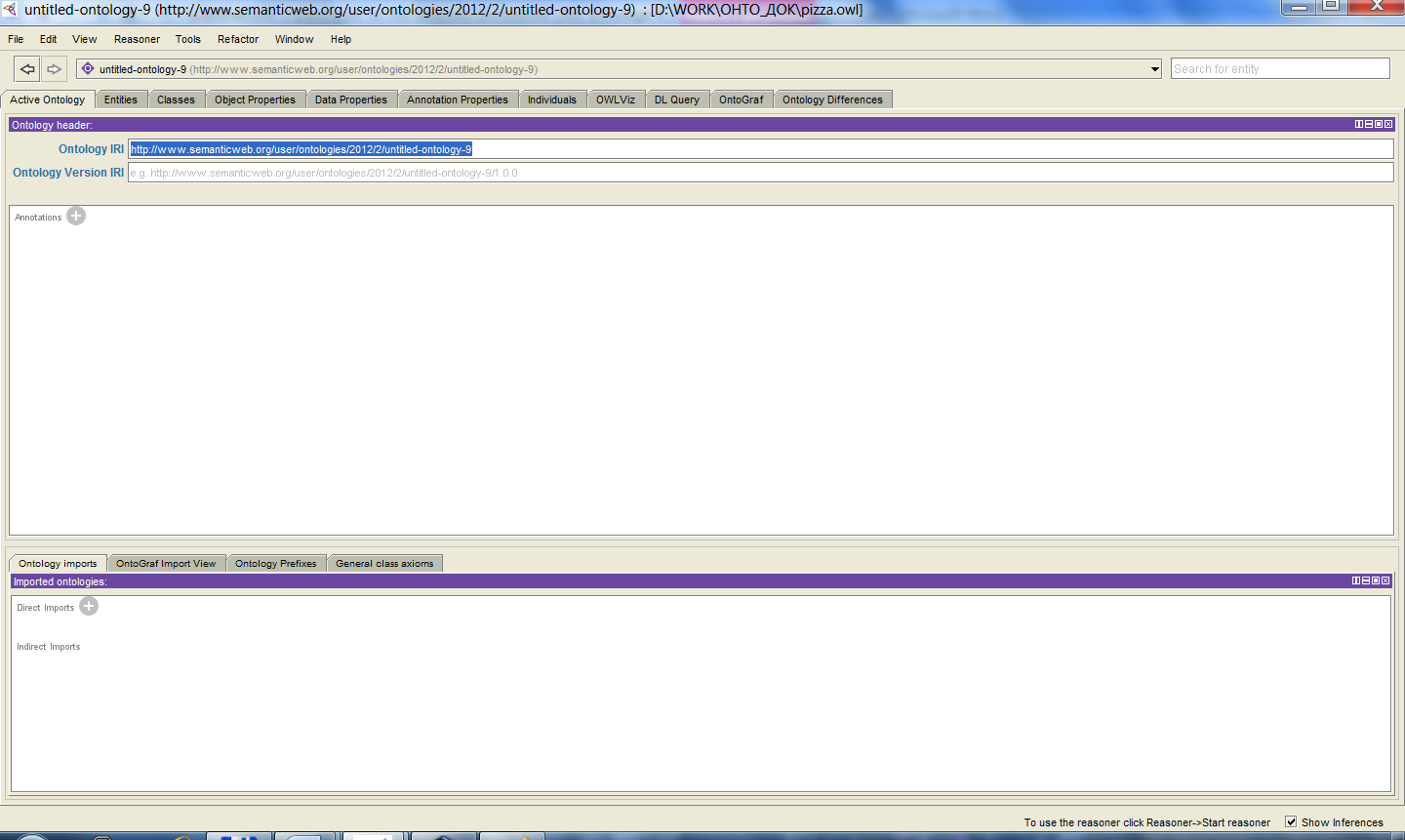
****

Рисунок 4.1. Active Ontology Tab (окно активной онтологии)

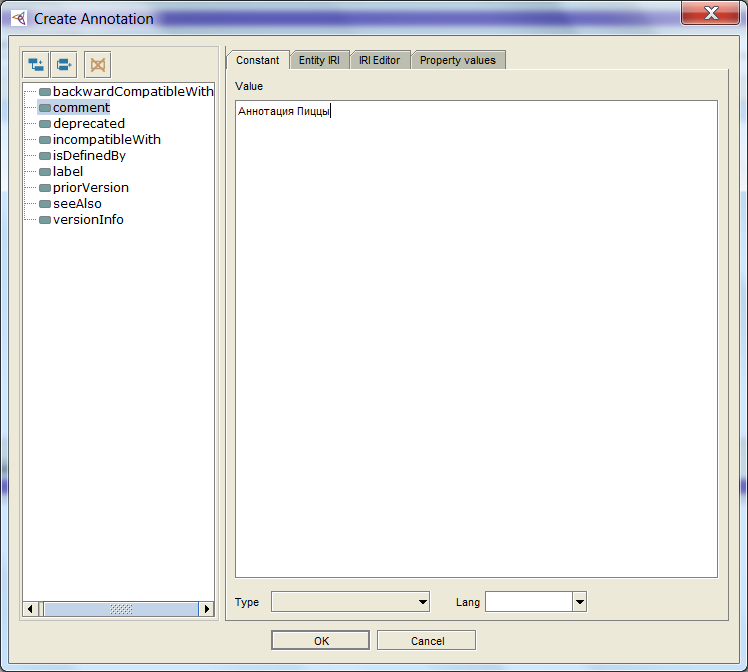


Рисунок 4.1а. Добавление аннотации-комментария

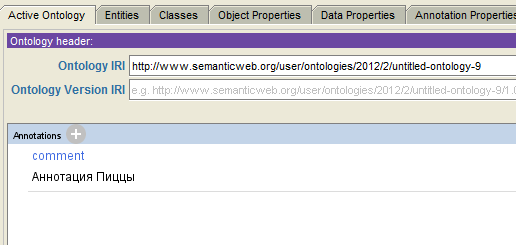


Рисунок 4.2: Вид аннотации онтологии - rdfs:comment

**Упражнение 3: Добавление комментариев к онтологии**

|  |  |
| --- | --- |
|  | 1.  Убедитесь, что выбрана закладка «Активные онтологии» |
|  | 2. В представлении «Онтология аннотации» щелкните значок (+) рядом с аннотациями. В таблице появится окно редактирования. Выберите из списка встроенных аннотации идентификаторы URI «comment-комментарии» и введите свой комментарий в текстовое поле в правой |
|  | 3. Введите комментарий: «онтология пиццы, которая описывает различные пиццы, основанный на их начинке», и нажмите кнопку OK, чтобы сохранить комментарий. Вид аннотации на закладке «Активной онтологии» должен выглядеть, как показано на рисунке 4.2 |

**Именованные классы**

Как упоминалось ранее, основными строительными блоками онтологии OWL являются классы. В Protege 4  создание классов осуществляется в закладке «Classes», показанной на рисунке 4.3. Дерево иерархии первоначально установленного класса представлено на рисунке 4.4. Пустая онтология содержит один класс с именем THING. Как упоминалось ранее, в OWL классы интерпретируются как множества индивидов (или объектов). Класс THING - это класс, представляющий набор, содержащий все объекты предметной области. По этой причине все классы являются подклассами THING2.

Давайте добавим некоторые классы в онтологии для того чтобы определить, какими бывают пиццы.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2Является частью OWL лексики, которая определяется онтологией, расположенной в http://www.w3. org/2002/07/owl/\#**

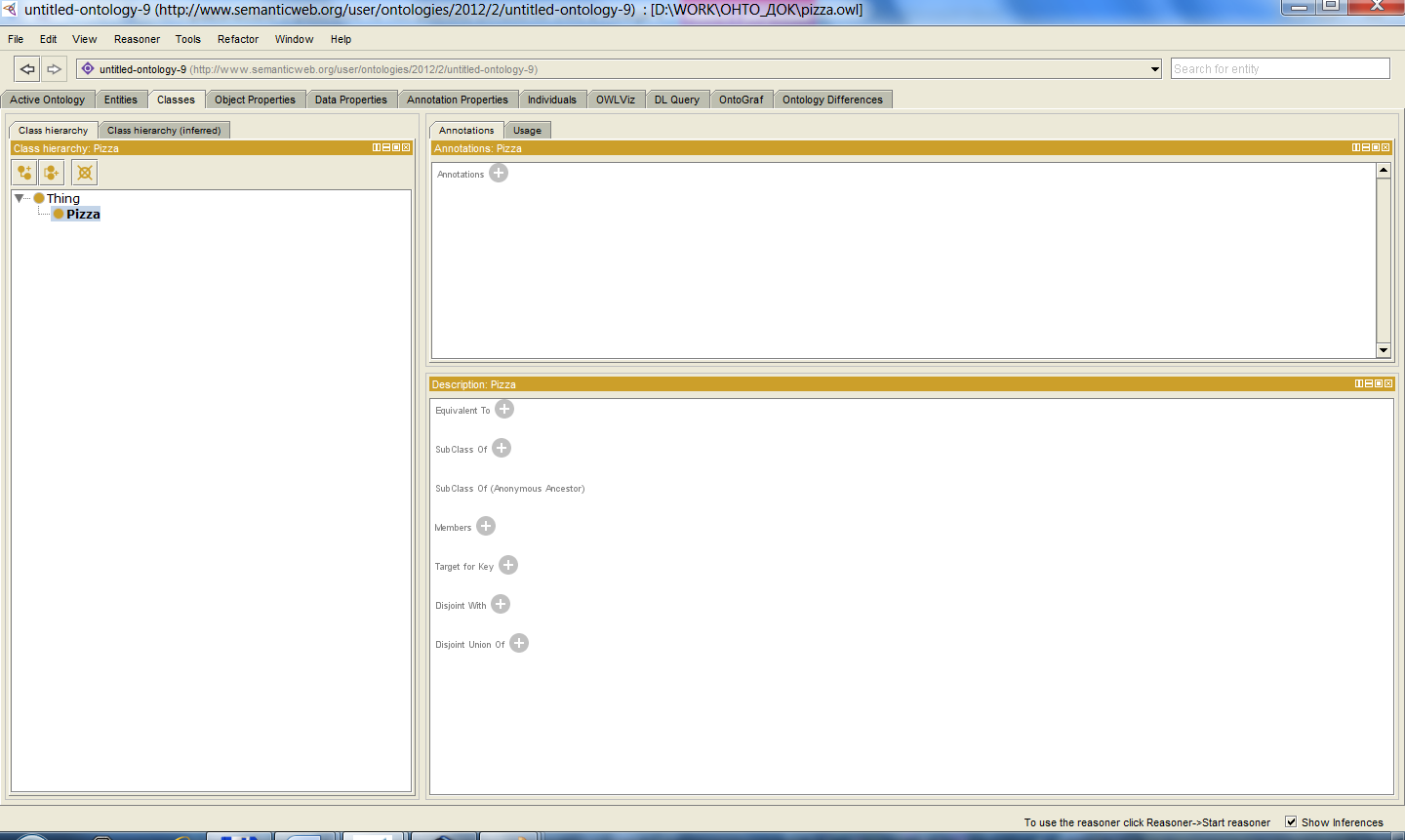
****

Рисунок 4.3: The Classes Tab

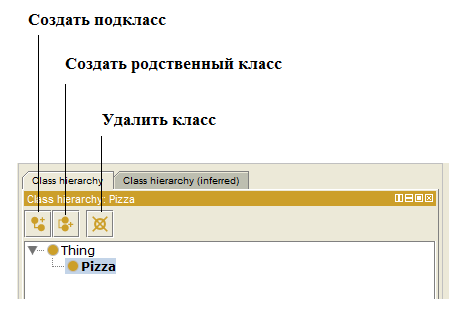


Рисунок 4.4. Иерархия классов

**Упражнение 4: Создание классов Pizza, PizzaTopping и PizzaBase**

|  |  |
| --- | --- |
|  | 1.Убедитесь, что выбрана закладка «Classes». |
|  | 2.Нажмите кнопку «Добавить подкласс», см. рисунок 4.4. Эта кнопка создает новый класс как подкласс выбранного класса (в данном случае мы хотим создать подкласс класса THING). |
|  | 3.Отобразится диалоговое окно для ввода названия вашего класса, введите Pizza (как показано на рисунке 4.5) и нажмите Enter. |
|  | 4.Повторите предыдущие шаги для добавления классов PizzaTopping и PizzaBase, обеспечивая, что флажок установлен на THING до нажатия кнопки «Добавить подкласс» таким образом, чтобы классы, создавались как подклассы THING. |

Иерархия классов теперь должна напоминать иерархии, показанные на рисунке 4.6.

|  |  |
| --- | --- |
| https://lh6.googleusercontent.com/1NakK-G4AAm_2SyxDWET0e52xVXUbzxNFyf77cAhK1YzqnfswwiNl4fHYQQgDqKoFdsjzgZkWxOGwNJ7QVSYAHV2fvtTakAzPn5CWA0QWt700p6OB0ZAdrEdmLg | После создания класса Pizza, вместо того, чтобы заново отмечать THING и с помощью кнопки «Создать подкласс» для создания PizzaTopping и PizzaBase как подклассы THING, используется кнопка «Создать родственный класс» (показанная на рисунке 4.4). В то время как класс Pizza отмечен, используйте кнопку «Создать родственный класс» для создания PizzaTopping и затем используйте  эту кнопку снова (при выделенном классе PizzaTopping) для создания PizzaBase, как родственника класса PizzaTopping - эти классы конечно же по-прежнему создаются как подклассы THING, так как Pizza является подклассом THING. |
| https://lh4.googleusercontent.com/l9K5axOFkj5EIVHsXx5AlVZZ1NwbhpdjXF6icbUN6mgW423q2IeOKhoLCg_s5oOw80261d4pHe6CWfOgRl80FM0c-Os-iPZfXVcVpot-D9ohS082nJ2Zs6B_ccQ | Иерархия классов может также называться таксономией |
| https://lh6.googleusercontent.com/1NakK-G4AAm_2SyxDWET0e52xVXUbzxNFyf77cAhK1YzqnfswwiNl4fHYQQgDqKoFdsjzgZkWxOGwNJ7QVSYAHV2fvtTakAzPn5CWA0QWt700p6OB0ZAdrEdmLg | Хотя нет никаких обязательных требований к именам классов OWL, существуют определенные рекомендации: все имена классов должны начинаться с прописной буквы и не должны содержать пробелов. (Этот вид нотации известен как CamelBack-нотация; именно она используется в данном руководстве). Например, Pizza, PizzaTopping, MargheritaPizza. В качестве альтернативы можно использовать подчеркивание для присоединения слов. Например, Pizza\_Topping. Вне зависимости от выбранной нотации важно быть последовательным. |

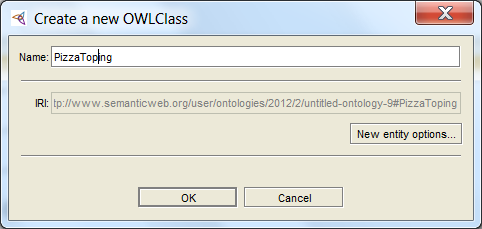


Рисунок 4.5 Задание имени класса

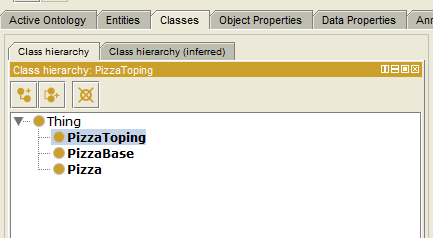


Рисунок 4.6 Созданная иерархия классов

**Непересекающиеся классы (Disjoint Classes)**

Добавив классы Pizza, PizzaTopping и PizzaBase к онтологии, можно указать, что эти классы являются непересекающимися, так что любой объект не может быть экземпляром более чем одного из этих трех классов. Для указания, что выбранный класс не пересекается с другими классами, нажмите на кнопку «Disjoint with», которая расположена в нижней части панели «Описание класса» (см. рис.4.7а).

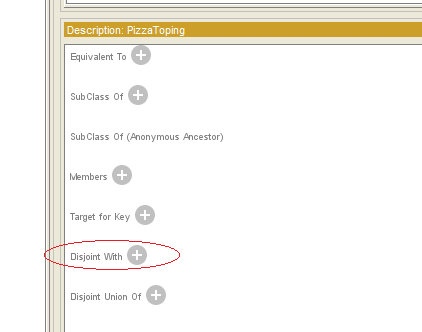


Рисунок 4.7а. Панель описания класса

**Упражнение 5: Сделать Pizza, PizzaTopping и PizzaBase непересекающимися**

|  |  |
| --- | --- |
|  | В иерархии классов выберите класс Pizza |
|  | Нажмите на кнопку «Disjoint With» в режиме «описание класса», появится диалоговое окно, где вы можете выбрать несколько классов (через кнопку Ctrl, рис.4.7б), которые должны быть несвязанными. Это позволит сделать PizzaBase и PizzaTopping (родственные классы Pizza) непересекающимися с классом Pizza |

Обратите внимание, что теперь список непересекающихся классов отображает PizzaTopping и PizzaBase. Выберите класс PizzaBase. Обратите внимание, что в непересекающихся классах отображаются классы, которые в настоящее время непересекаются c PizzaBase, а именно: Pizza и PizzaTopping (см. рис. 4.7в).

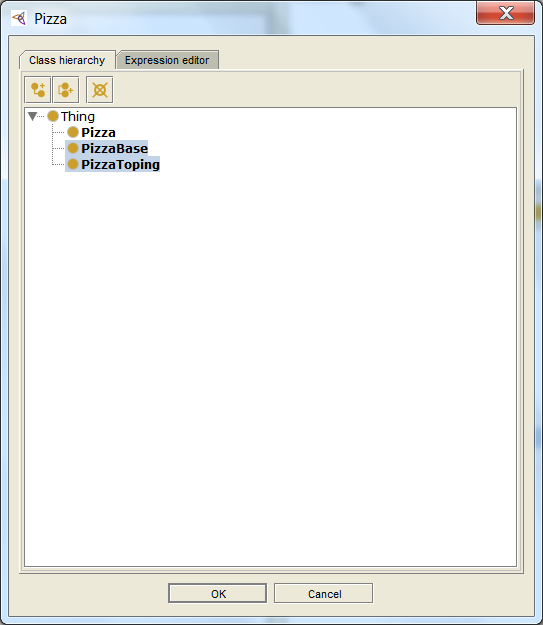


Рисунок 4.7б. Одновременный выбор нескольких классов для указания непересекаемости при помощи кнопки Ctrl

|  |  |
| --- | --- |
| https://lh4.googleusercontent.com/CrMPRV_8Uwmt2EtVzvWag_xPksZvMH-zZAY5UqspbRdDvw4ELVNVpbZ7vd4_xlrcc-VEFokSx9D4ErnllOBo4oEXQv_MP3cQbMVqmSxXBH4NLm2sRH2xYhNyuVM | По умолчанию классы в OWL могут пересекаться. Мы поэтому не можем предположить, что индивид не является членом определенного класса просто потому, что он не утвержден экземпляром этого класса. Для того, чтобы разделить группы классов их необходимо явно сделать непересекающимися. Это гарантирует, что индивид, который является членом одного из классов в группе не может быть членом какого-либо другого класса в этой группе. В нашем примере выше классы Pizza, PizzaTopping и PizzaBase были объявлены непересекающимися. Это означает, что ни один объект не может быть экземпляром более одного из этих классов, например бессмысленно одновременно  быть и Pizza и PizzaBase. |

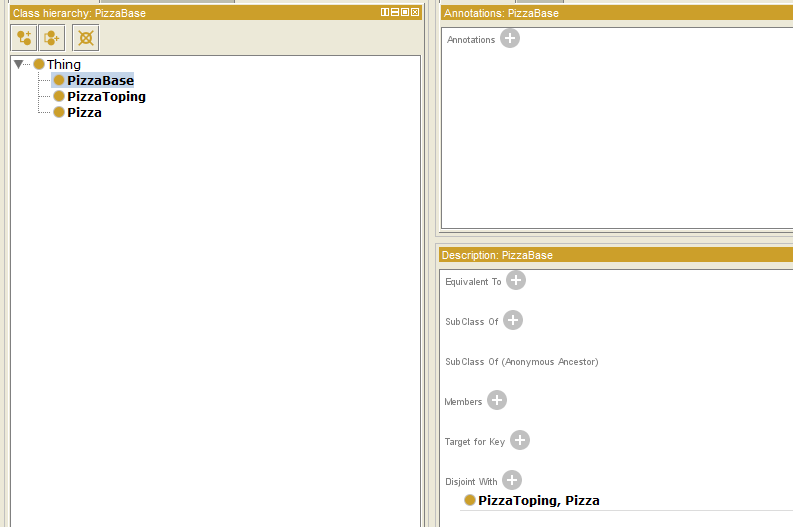


Рисунок 4.7в. Отображение несвязанных классов

**Использование иерархии классов при создании классов**

В этом разделе мы будем использовать инструмент «Создать иерархию классов» для добавления некоторых подклассов класса PizzaBase.

**Упражнение 6: Используйте инструмент «Создать иерархию классов» для создания ThinAndCrispy и DeepPan как подклассы PizzaBase**

|  |  |
| --- | --- |
|  | 1. В иерархии классов выберите класс PizzaBase. |
|  | 2. В главном меню из инструметов “Tools -Сервис”   Protege выберите «Create Class hierarchy - Создать иерархию классов» (рис.4.8а). |
|  | 3. Появится панель инструментов, показанная на рисунке 4.8б. Поскольку мы выбрали класс PizzaBase, первый переключатель в верхней части инструмента установлен на выбранный нами PizzaBase. Если бы мы не выбрали PizzaBase перед запуском инструмента, то иерархия может использоваться для выбора нужного класса. |
|  | 4. Нажмите кнопку «Далее» (Continue) на панели инструмента — будет отображаться страница, показанная на рисунке 4.8. Теперь нам нужно указать в окне подклассы PizzaBase, которые мы хотим создать. В большой текстовой области введите в поле имя класса ThinAndCrispyBase (для тонкой и хрустящей пиццы) и нажмите enter. На новой строке введите имя класса DeepPanBase, чтобы набранное выглядело так, как показано на рисунке 4.8. |
|  | 5. Нажмите кнопку «Далее» на панели инструмента. Проверяются указанные имена на соответствие стилю, который ранее был упомянут (без пробелов и т.д.). Инструмент также проверяет уникальность - имена двух классов не могут быть одинаковыми. Если есть какие-либо ошибки в именах классов, они будут представлены на этой странице наряду с предложениями о внесении исправлений. |

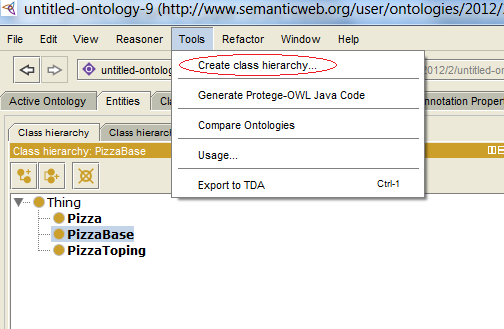


Рисунок 4.8а Выбор меню создание иерархии класса

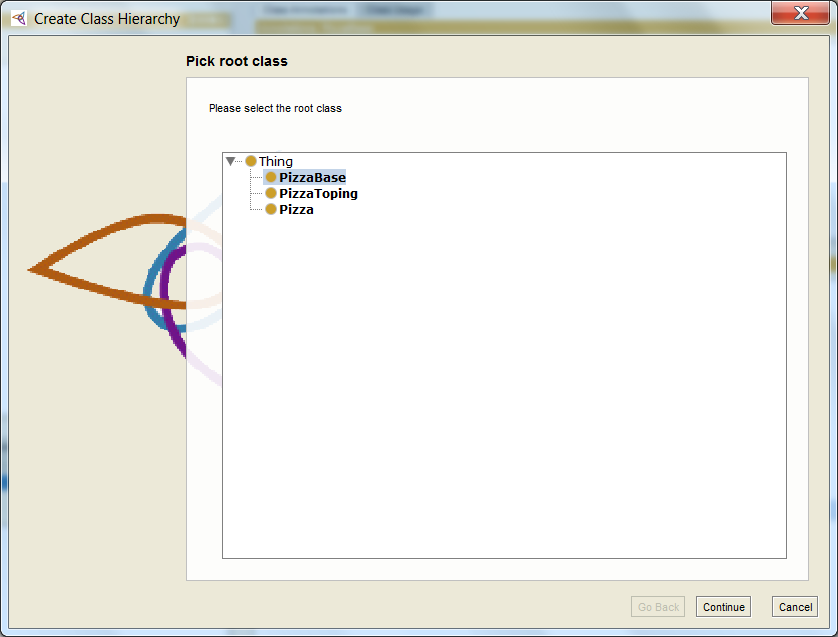


Рисунок 4.8б Панель инструмента для иерархии классов.

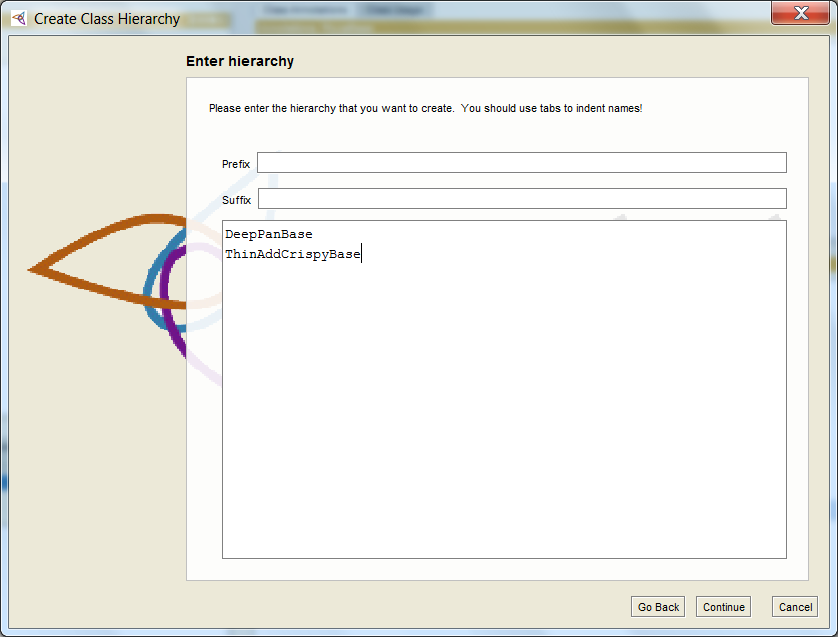


Рисунок 4.8. Создайте иерархию классов: Ввод классов

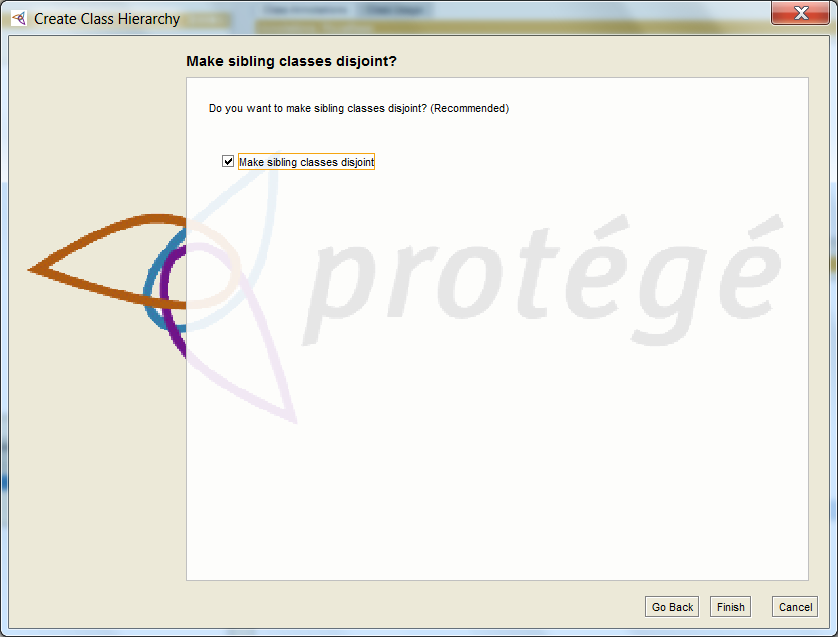


Рисунок 4.8в

После нажатия кнопки «Далее» инструмент создает классы, делает их несвязанными. Нажмите «Готово» (Finish) для закрытия инструмента. Онтология теперь должна иметь ThinAndCrispyBase и DeepPanBase, как подклассы PizzaBase (рис. 4.8г). Эти новые классы должны быть несвязанными между собой. Таким образом PizzaBase не может быть тонкой и хрустящий и одновременно пышной. Как Вы уже догадались, что инструмент значительно ускорит процесс добавления классов, если для добавления к онтологии у нас много классов.

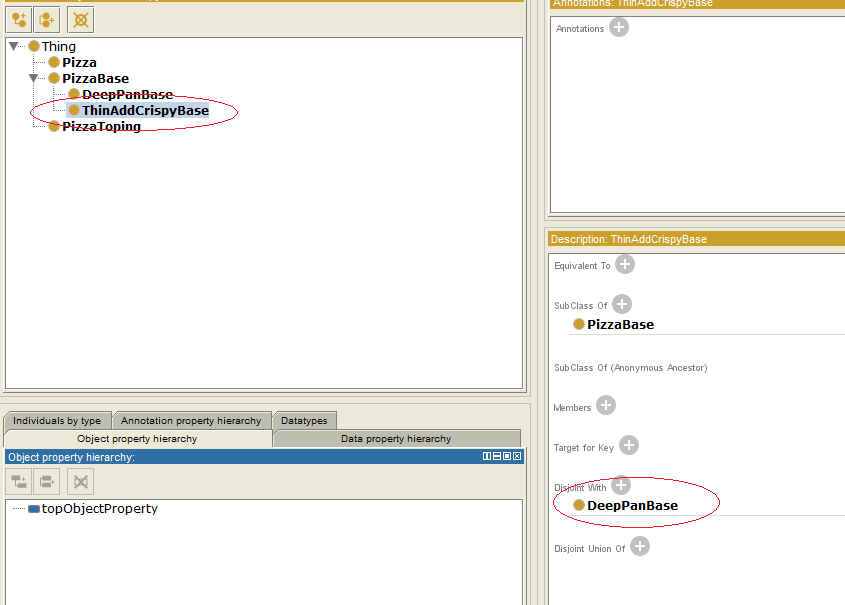


Рисунок 4.8г. Результат добавления несвязанных классов

|  |  |
| --- | --- |
| https://lh5.googleusercontent.com/ZVOBil6hkV7qZi7SiZeRmiyngHWU3mQcl5OVpqFMl2syuthC38r4k5KtZgqowJdLv2OerEnTAQZqAxZ5jHivaApWQv7QVHls9hdkfFQu6poD27XiBX4dGbuyvto | На одной из закладок мастера (рис.4.8) «Создания иерархии класса» вводятся классы, которые вновь создаются. Если нам нужно было создать много классов, которые имеют префикс или суффикс, мы могли бы использовать параметры для автоматической приставки или окончания имени и этот текст автоматически добавлялся бы к именам классов, которые мы создавали. Например, если бы мы задали префикс “Pizzabase\_”  то имена классов получились бы Pizzabase\_ DeepPanBase и Pizzabase\_ThinAndCrispyBase |

**Создание подклассов начинок пиццы PizzaTopping**

Теперь, когда у нас есть некоторые базовые классы, создадим подклассы начинок пиццы **PizzaTopping**. Для пользы дела, начинки объединяются в различные категории — мясные начинки, растительные начинки, сырные начинки и начинки из морепродуктов.

**Упражнение 7: Создание подклассов класса PizzaTopping**

|  |  |
| --- | --- |
|  | 1. В иерархии классов выберите класс PizzaTopping. |
|  | 2. Вызовите инструмент «Создать иерархию классов...» так же, как инструмент был вызван в предыдущем упражнении. |
|  | 3. Убедитесь, что выбран параметр PizzaTopping и нажмите кнопку «Далее». |
|  | 4. Мы хотим, чтобы все из PizzaTopping класса заканчивались одинаково, для чего в поле «Суффикс», введите «Topping». Инструмент будет помогать нам при вводе, автоматически добавляя суффикс «Topping» для всех наших имен классов. |
|  | 5. Инструмент позволяет ввести иерархию классов с отступом дерева. С помощью клавиш Enter и Tab в области текста введите имена классов, как показано на рис 4.9. Обратите внимание, что в именах классов должны быть отступы, например, SpicyBeef, который мы хотим, чтобы он был подклассом Meat, необходимо было написать под Meat и с отступом Tab. Аналогичным образом, Pepperoni является подклассом Meat, также набран под Meat ниже SpicyBeef и также с отступом. |
|  | 6. После задания структурированного списка классов, нажмите кнопку «Далее» и затем убедитесь что https://lh3.googleusercontent.com/2WQoTFuBh0rdqNwuuhPL9KZch_3o41PGQNK3ko8lGJWF5EJXD_iXlSe955Epjm82dssfM9t6Z7VBA5yBxcuKDOvGnGM_JZHbdcZCfTopVVjHW2e4TDJKGwfIriUотмечена таким образом, чтобы новые родственные классы делались несвязанными друг с другом. |
|  | 7. Нажмите кнопку «Готово» для создания классов. Нажмите «Готово» снова закрыть инструмент создания иерархии. |

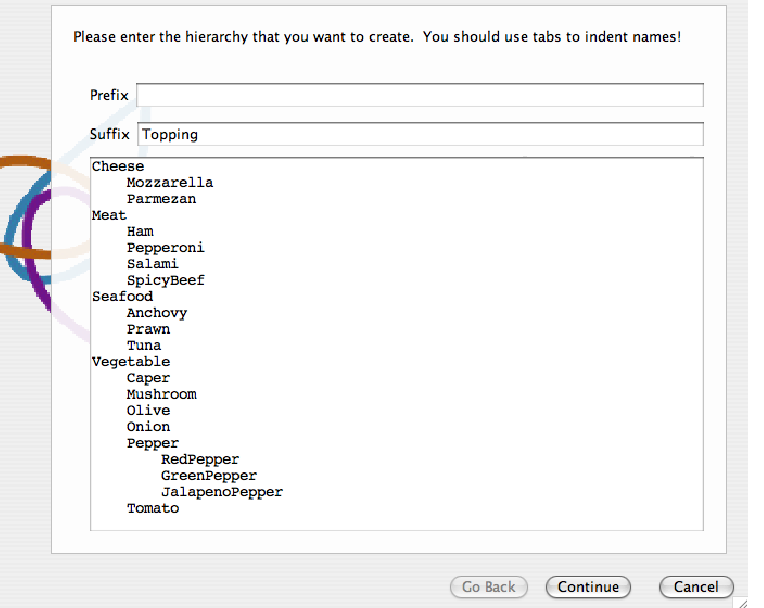


Рисунок 4.9: Topping иерархия

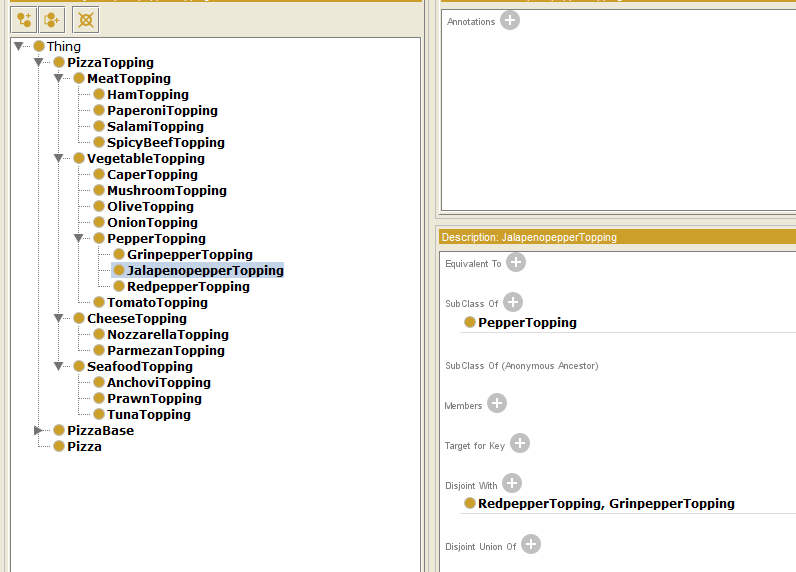


Рисунок 4.10: Иерархия классов

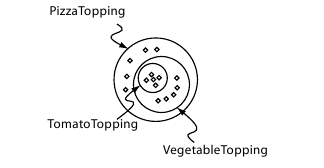


Рисунок 4.11: Смысл подкласса — все индивиды, которые являются членами класса TomatoTopping являются членами класса VegetableTopping и членами класса PizzaTopping, так как мы объявили, что TomatoTopping является подклассом класса VegetableTopping, который является подклассом PizzaTopping

|  |  |
| --- | --- |
| https://lh4.googleusercontent.com/K9lgvUvIS8L3cP1wXUPTsclvsHyoTAFG1KqwyHsZ7ScvACjY7pAh6eSXM63GozWaTxLdyfXd5qLl0knZAhFXUL_d3kmpqIV-d4sHfyh3QwujVImQXurMtbAW7Ws | До этого момента мы создали несколько классов с простым  именем, некоторые из которых являются подклассами других классов. Строительство  иерархии классов может показаться довольно интуитивным до настоящего времени. Однако, что на самом деле означает быть подклассом чего либо в OWL? К примеру, что означает для VegetableTopping быть подклассом PizzaTopping, или TomatoTopping быть подклассом VegetableTopping? В OWL подклассы наследуются. Другими словами, если VegetableTopping является подклассом PizzaTopping тогда все экземпляры VegetableTopping являются экземплярами PizzaTopping, без исключения — если что-то VegetableTopping, то это означает, что это также PizzaTopping, как показано на рисунке 4.11.  Именно по этой причине мы педантично назвали все наши классы начинок с суффиксом 'topping', например, HamTopping. Несмотря на то, что сами названия классов не несут формальной семантики в OWL (и в других языках онтологии), если мы назвали Ветчину HamTopping, то тогда каждый человек поймет, что это что-либо, что является ветчиной и является также представителем MeatTopping и также PizzaTopping. |

**4.3 Свойства в OWL**

Свойства в OWL представляют отношения. Существует два основных типа свойств: свойства объектов и свойства типа данных (DataType). Свойствами объекта являются отношения между двумя индивидами. В этой главе мы сосредоточимся на свойствах объекта; свойства типов данных описаны в главе 5.

Свойства объекта связывают индивидов. OWL также имеет третий тип свойства - свойства аннотации. Свойства аннотации могут использоваться для добавления информации (метаданные — данные о данных) для классов, отдельных индивидов и свойств объектов/типов данных. Рисунок 4.12 изображает собой пример каждого типа свойства. Свойства могут быть созданы с помощью вкладки «Свойства объекта», показаной на рисунке 4.13. На рисунке 4.14 показаны кнопки, расположенные в левом верхнем углу вкладки «Свойства объекта», которые используются для создания свойств OWL. Как показано на рисунке 4.14, имеются кнопки для создания свойств DataType,

Свойства объекта и свойства Datatype могут быть отмечены как свойства Аннотации

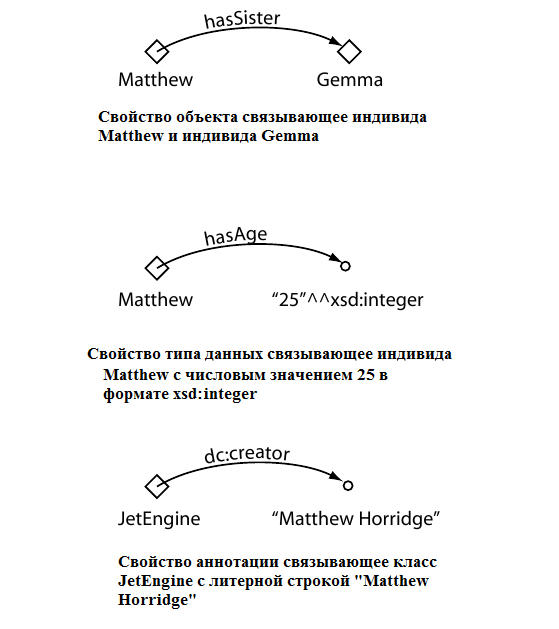
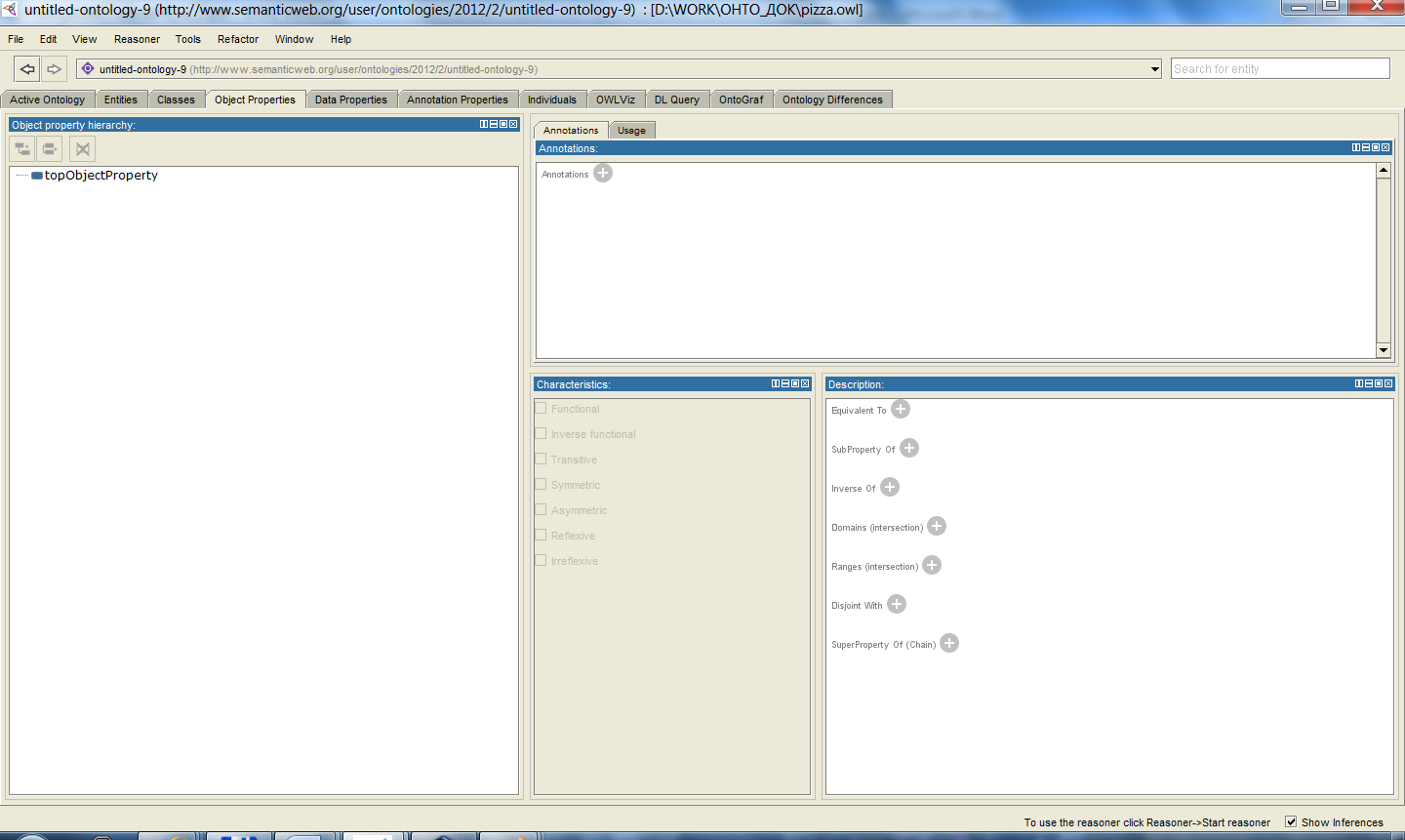


Рисунок 4.12: Различные типы свойств OWL



4.13 Рисунок: Object PropertiesTab

**Свойства объекта и свойства аннотации**

Большинство свойств, создаваемых в этом учебнике, будут свойствами объекта.

☺ В Protégé 4.2 по умолчанию автоматически создаются свойства объектов верхнего уровня (TopObjectProperty), по аналогии с  классом THING. А почему бы и нет? Если быть последовательными, то класс THING такой же класс, как и все создаваемые пользовательские классы, следовательно, к нему применимы все инструменты описания. Наверно это сделано для того, чтобы легче было объединять проекты, разработанные автономно. Значит все свойства объектов пользователя в этом смысле будут подсвойствами супер-свойства TopObjectProperty.

**Упражнение 8: Создание свойства объекта под названием hasIngredient**

|  |  |
| --- | --- |
|  | 1. Перейдите на вкладку «Свойства объекта». Использование «Добавить подсвойство объекта» кнопку (см рисунок 4.14) для создания нового свойства объекта, являющегося субсвойством TopObjectProperty |
|  | 2. Введите Имя свойства hasIngredient, в  диалоговом окне «Имя свойства»,' которое всплывает, как показано на рисунке 4.15 |

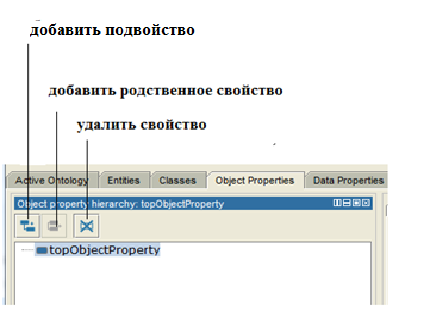


Рисунок 4.14. Функции окна иерархии свойств объектов

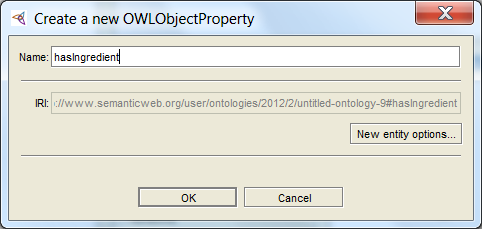


Рисунок 4.15 Задание имени свойства объекта

|  |  |
| --- | --- |
| https://lh4.googleusercontent.com/s9Dc_GK9EWX2GvJkhceMZyy7QcG9nIDCLcSCuZ0TLk3yURG_vr_BWyAjTSA4KTGx46GO4HFJ2ur1NGA3Gmz8KuddchMVSXgVZb_UMBooJSFyT0_fobiUpSE65R4 | Хотя нет никаких ограничений на имена для свойств, мы рекомендуем имена свойств начинать с буквы нижнего регистра, не иметь пробелов и использовать накапливающиеся (ранее задействованные) имена. Мы также рекомендуем создавать свойства с префиксом слова has- «имеет» или словa is- "является', например hasPart, isPartOf, hasManufacturer, isProducerOf. Настоящая Конвенция не только имеет цель сделать яснее свойства для людей, но она дает возможность также воспользовался генератором «Английская проза подсказки», которая использует такие наименования, там, где это возможно, чтобы обеспечивать больше возможности для чтения выражений из описания класса.  Генератор "Английская проза подсказки” отображает описание классов и т.д. в более естественной форме английского языка, с целью более легкого понимания описания класса. Подсказки появляются, когда указатель мыши наведен на описание класса в пользовательском интерфейсе. |

Добавив свойство hasIngredient («имеет Инградиент»), мы теперь добавим два дополнительных свойства — hasTopping и hasBase. В OWL свойства могут иметь подсвойства, таким образом, чтобы это отражало форму иерархии свойств. Подсвойства специализируются их супер-свойством (так же, как подклассы специализируются их суперклассами). Например свойство hasMother («имеет мать») специализируются более общим свойством hasParent («имеет родителей»). В нашей онтологии  Pizza свойства hasTopping и hasBase должны создаваться как подсвойства hasIngredient. Если свойство hasTopping (или свойство hasBase) связывает двух индивидов это означает, что два индивида связаны в свойстве hasIngredient.

**Упражнение 9: Создание hasTopping и hasBase - подчиненные свойства hasIngredient**

|  |  |
| --- | --- |
|  | 1. Чтобы создать свойство hasTopping как подсвойство свойства hasIngredient, выберите свойство hasIngredient в иерархии свойств на вкладке «Свойства объекта». |
|  | 2. Нажмите на кнопку «Добавить подсвойство». Будет создано новое свойство объекта как подсвойство свойства hasIngredient |
|  | 3. Задайте имя нового свойства «hasTopping». |
|  | 4. Повторите описанные выше шаги, создайте свойство «hasBase». |

Обратите внимание, что аналогично возможно создание подсвойства свойства DataType. Однако это не позволяет смешивать и сопоставлять свойства объектов и свойства типа данных с их подсвойствами. Например невозможно создать свойство объекта, которое является свойством подсвойства DataType и наоборот.

**4.5 Обратные свойства (inverse)**

Каждое свойство объекта может иметь соответствующее обратное свойство. Если некоторые свойство связывает индивида ***а*** с некоторым индивидом **b**, то его обратное свойство связывает индивид **b** с индивидом ***а***. Например, рисунок 4.16 показывает свойства hasParent («имеет родителей») и его обратное свойство hasChild («имеет детей») — если Мэтью hasParent Жан, то из обратного свойства мы можем заключить, что Жан hasChild Мэтью.

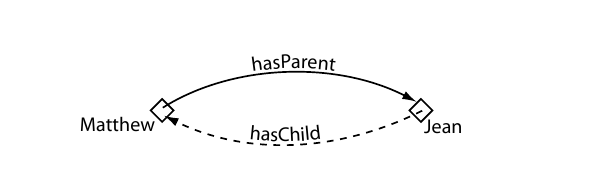


Рисунок 4.16: Пример обратного свойства: hasParent имеет обратное свойство, которым является hasChild

Обратные свойства могут быть созданы/указаны на панели свойств, показанной на рисунке 4.17. Для полноты картины мы определим обратные свойства для наших существующих свойств в онтологии пиццы.

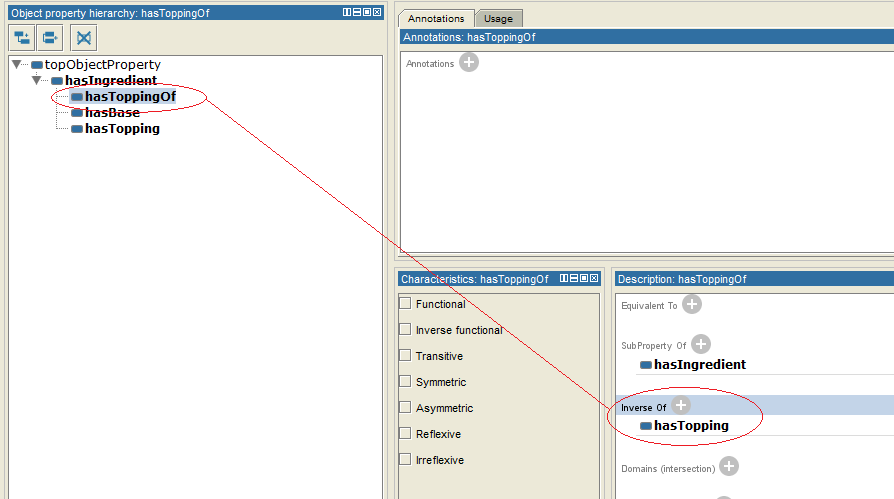


Рисунок 4.17: Просмотр обратных свойств

**Упражнение 10: Создайте некоторые обратные свойства**

|  |  |
| --- | --- |
|  | 1. Кнопка «Добавить родственное свойство объекта» на вкладке «Свойства объекта» для создания нового свойства объекта с именем isIngredientOf – «инградиент в (начинке)» (это будет задано обратное свойство для свойства hasIngredient «имеет инградиент»). |
|  | 2. Нажмите значок (+) рядом с кнопкой «inverse Of» на панели «Описание свойства», как показано на рисунке 4.17. Будет отображаться диалоговое окно (рис.4.17а), из которого можно выбирать свойства. Выберите свойство hasIngredient и нажмите «OK». Свойства hasIngredient теперь должны быть отображены в представлении «Inverse Of». |
|  | 3. Выберите свойство hasBase. |
|  | 4. Нажмите добавить значок (+) рядом с «Inverse Of» на представлении «Описание свойства». Создайте новое свойство в этом диалоговом окне, под названием isBaseOf. Выберите это свойство и нажмите кнопку «OK». Заметьте, что hasBase теперь имеет обратное свойство назначен назвал это BaseOf. Вы можете при желании разместить новое свойство isBaseOf как подчиненных свойств isIngredientOf (N.B, как это будет получено будет объяснено позднее, когда будет использоваться реазонер). |
|  | 5. Выберите свойство hasTopping |
|  | 6. Нажмите добавить значок (+) рядом с «Inverse Of» на представлении «Описание свойства». Используйте диалоговое окно Свойства, которое всплывает для создания свойства isToppingOf и нажмите «OK». |

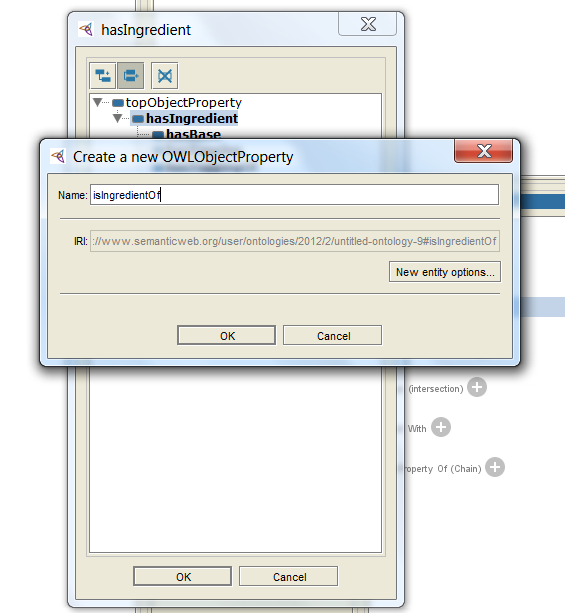


Рисунок 4.17а. Окно для создания нового обратного свойства

**4.6 Характеристики свойств объектов OWL**

OWL позволяет значение свойства обогатить за счет использования множества характеристик. В следующих разделах рассматриваются различные характеристики, которые могут иметь свойства:

**4.6.1 Функциональные свойства (functional)**

Если свойство является функциональным, то для данного индивида  может существовать не более одного индивида, который имеет отношение к первому индивиду через это свойство. Рисунок 4.18 показывает пример функционального свойства hasBirthMother — «кто-то может иметь только одну биологическую мать». Если мы говорим, что функционально Жан hasBirthMother Пегги и мы также говорим, что функционально Жан hasBirthMother Маргарет, то только потому, что hasBirthMother является функциональным свойством, мы можно заключить, что Пегги и Маргарет должны быть одним и тем же человеком. Однако, следует отметить, что если Пегги и Маргарет были четко названы двумя разными людьми, то вышеуказанные рассуждения могут привести к противоречию.

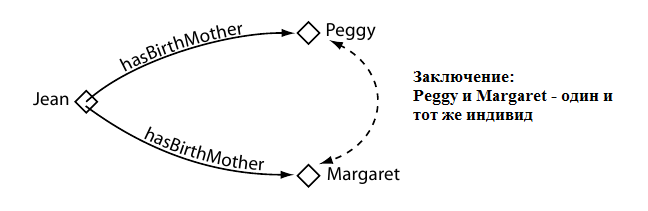
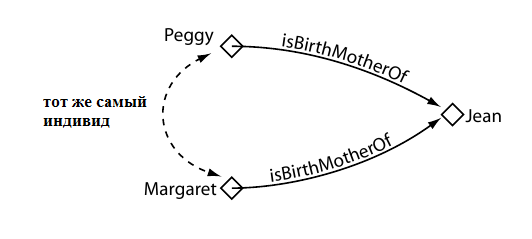


Рисунок 4.18: Пример функционального свойства: hasBirthMother

|  |  |
| --- | --- |
| https://lh6.googleusercontent.com/6D0k0uWyyw6uIauu_DstS06D9SkifdsgDMkgakzAzk0WtOPuEPOoZUUmd5RvIEFHT4iMiDLBBOa90rkskOhNdb1dioQ311_nD7ZF424dRwXSomkGzSxiPJv8ycA | Функциональные свойства, также известны как однозначные свойства, а также как индивидуальные особенности индивида. |

**4.6.2 Обратные функциональные свойства**

Если свойство является обратным функциональному свойству, то это значит, что свойство является обратным функциональным. Для конкретного индивида может быть множество индивидов, относящиеся к первому индивиду через это свойство. На рисунке 4.19 показан пример обратного функционального свойства isBirthMotherOf. Это обратное свойство для свойства hasBirthMother — так как hasBirthMother является функциональным, то isBirthMotherOf является обратным функциональным. Если мы заявляем, что Пегги – биологическая мать Жан, и мы также что Маргарет биологическая мать Жан, то мы также можем сделать вывод, что Пегги и Маргарет – это один и тот же человек.



4.19 Рисунок: Пример обратного функционального свойства: isBirthMotherOf

**4.6.3 Транзитивные свойства (transitive)**

Если свойство транзитивное и свойство связывает индивида ***a*** и индивида ***b***, а также индивида ***b*** связывает с индивидом ***c***, то мы можем вывести, что этот индивид ***а*** связан с индивидом **c** через это свойство. К примеру на рисунке 4.20 показан пример транзитивного свойства hasAncestor. Если индивид Мэтью стал родоначальником Питер, и Питер имеет предка Уильяма, то мы можно сделать вывод что Мэтью стал родоначальником  Уильяма - это обозначается пунктирной линией на рис. 4.20.

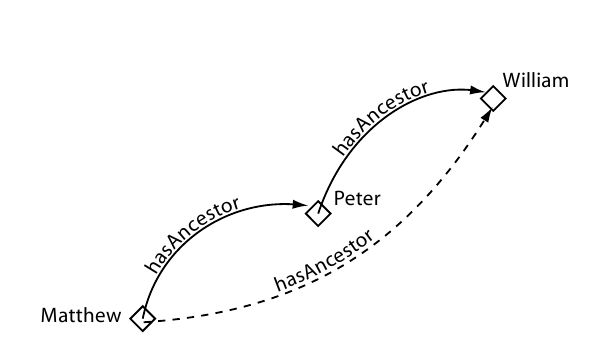


Рисунок 4.20: Пример транзитивного свойства: hasAncestor

**4.6.4 Симметричные свойства (Simmetric)**

Если свойство ***p*** симметричное, и свойство связывает индивида ***а*** с индивидом ***b***, то индивид ***b*** связан также с индивидом ***а*** через свойство ***р***. Рисунок 4.21 показывает пример симметричного свойства. Если индивид Мэтью связан с индивидом Джемма через свойство hasSibling, то можно считать, что Джемма должна быть также связана с Мэтью через свойство hasSibling. Другими словами, если Мэтью имеет родственника Джемма, то Джемма должна иметь родственника Мэтью. Таким образом, симметричное свойство является самим себе обратным.

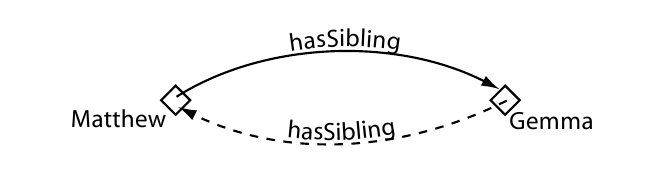


Рисунок 4.21: Пример симметричного свойства: hasSibling

Мы хотим сделать свойство hasIngredient транзитивным, так чтобы например если PizzaTopping имеет ингредиент, то сама Pizza также имела этот ингредиент. Чтобы задать характеристики свойств используют панель характеристик, представленную на рисунке 4.22.

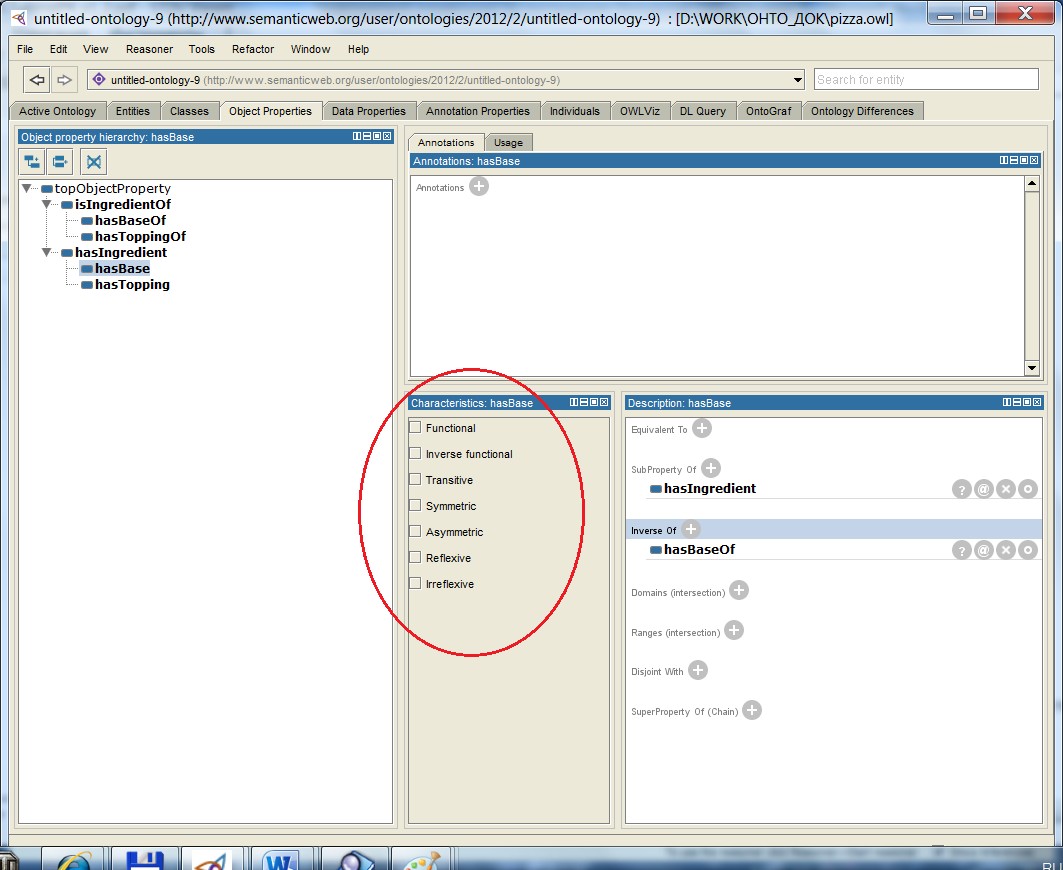


Рисунок 4.22: Панель характеристик свойств

**Упражнение 11: Сделать свойство hasIngredient транзитивным**

|  |  |
| --- | --- |
|  | 1. Выберите свойство hasIngredient в иерархии свойств на вкладке «Свойства объекта». |
|  | 2. Отметьте поле «Transitive» галочкой на представлении «характеристики свойства». |
|  | 3. Выберите свойство isIngredientOf, которое является обратной hasIngredient. Установить транзитивную отметку. |

|  |  |
| --- | --- |
| https://lh6.googleusercontent.com/Mxv3v_Q5tImWRwxVb_Jn7-WYkWk8V0JW6-eQiJfYkEdkBtwcEeFfYXt4IlpYc7eltSwDuh5Q0Z0VoUi8J4eXsLDFajOVdlN4Htdd7rbDYrjQM7qQ78gPd31ez04\* | Если свойство является транзитивным, то его обратное свойство также должно быть транзитивным.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  На момент написания статьи это необходимо было делать вручную в 4.2 Protege. Тем не менее, реазонер будет считать, что если свойство транзитивно, обратное свойство также является транзитивным. |
| https://lh4.googleusercontent.com/hbjVRpXJhA7e-sKi17PchdulPHxe9PV0vCDk-3-vXUWjYu2xwHpghlyHv8Fg3SzVU5zBSfdbQVMt7cLAs-jlhIZfb2GYVPeWqlw0WKJoR9LaJIi5-Isf-muKQrY\* | Обратите внимание, что если свойство transitive, то оно не может быть функциональным  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Причиной этого является то, что транзитивные свойства, по своей природе, могут образовывать " цепочки " индивидов. Создание транзитивного свойства функциональным, следовательно, не имеет смысла. |

Теперь мы хотим сказать, что наша Pizza может иметь только одну базу. Существует множество способов, как это может быть достигнуто. Однако для этого мы сделаем свойство hasBase функциональным, так чтобы оно задавало только одно значение для выбранного индивида.

**Упражнение 12: Сделать свойство hasBase функциональным**

|  |  |
| --- | --- |
|  | 1. Выберите свойство hasBase |
|  | 2. Отметьте галочкой поле «Functional» на панели Характеристики свойства. |

Если мы работаем с свойствами datatype, то список характеристик свойств уменьшен до единственного свойства функциональности. Это потому что OWL DL не поддерживает свойств типов данных, таких как транзитивность, симметричность и не позволяет иметь обратные свойства.

**4.6.5 Асимметричные свойства (Asimmetric)**

Если свойство ***p*** асимметричное, и свойство связывает индивида ***а*** с индивидом ***b***, то индивид ***b*** не может быть связан с индивидом ***a*** через свойство ***p***. Рисунок 4.23 показывает пример асимметричного свойства. Если индивид Роберт связаны с индивидом Дэвид через свойство isChildOf, то можно сделать вывод что Давид не связан с Робертом через свойство isChildOf. Однако, разумно заявить, что Давид может быть связан с другим индивидом через свойство isChildOf. Другими словами Если Роберт является потомком Дэвида, то Дэвид не может быть ребенком Роберта, но Давид может быть ребенком Билла.

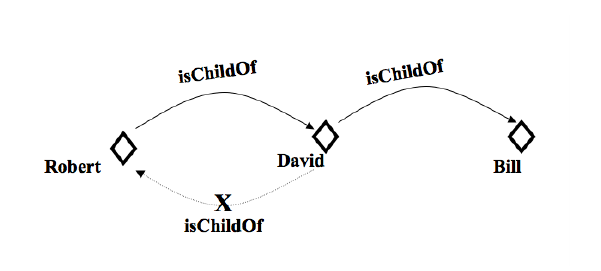


Рисунок 4.23: Пример асимметричного свойства hasChildOf

**4.6.6 Рефлексивные свойства (Reflexive)**

Свойство ***Р*** называется рефлексивным, когда индивид ***а*** должен быть связан с собой. На рисунке 4.24 мы можем видеть пример с помощью свойства «knows». Индивид Джордж должен иметь отношение к себе с помощью свойства «knows». Другими словами Джордж должен знать себя. Однако кроме того, возможно для Джорджа знать других людей; Поэтому индивид Джордж может иметь связь с индивидом Симон через свойство «knows».

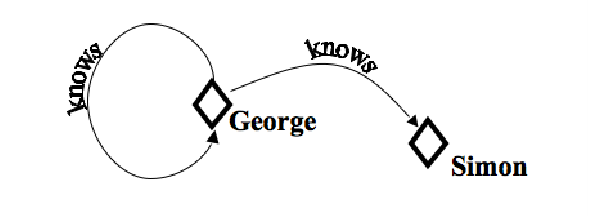


Рисунок 4.24: Пример рефлексивного свойства: knows

**4.6.7 Иррефлексивные свойства (irreflexive)**

Если свойство ***p*** иррефлексивное, то оно может быть охарактеризовано как свойство, которое связывает индивида ***а*** с индивидом ***b***, где индивид ***а*** и индивид ***b***  обязательно разные. В качестве примера будет свойство motherOf («является матерью»): индивид Алиса может быть связан с индивидом Боб с помощью свойства motherOf, но Алиса не может быть motherOf самой себя (Рисунок 4,25).

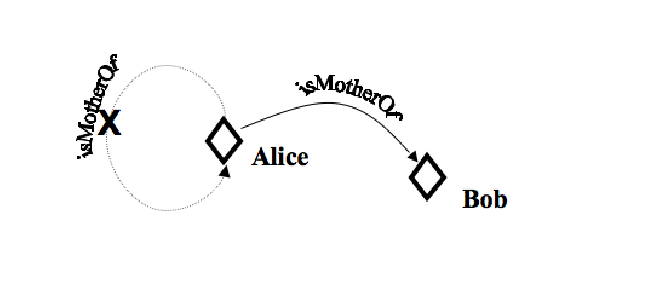


Рисунок 4,25: Пример иррефлексивного свойства: isMotherOf

**4.7 Домены и диапазоны свойств**

Свойства могут иметь домен и указанный диапазон. Свойства связывают индивидов из доменов с индивидами в диапазонах. К примеру в нашем онтологии Pizza, свойства hasTopping вероятно свяжет индивидов, принадлежащих к классу Pizza c индивидами, принадлежащими к классу PizzaTopping. В этом случае Pizza является доменом свойства hasTopping, а диапазоном — PizzaTopping — это показано на рисунке 4.26.

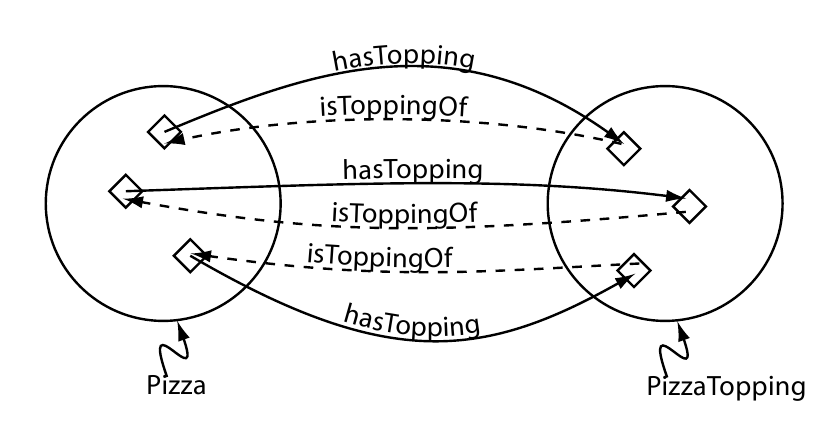


Рисунок 4.26: Домен и диапазон для свойства hasTopping и его обратного свойства isToppingOf. Доменом свойства hasTopping является класс Pizza, а диапазоном свойства hasTopping является класс PizzaTopping. Домен и диапазон для isToppingOf – это смена ролей домена и диапазона для hasTopping

|  |  |
| --- | --- |
| https://lh3.googleusercontent.com/UZGeGxorbO90h1mvSwPcd9CZ3VZukPtjNZ6CTNZ-YvDtDLK3mfw_7zf_Z0bY4VVq7Mus5HmQX3FCLuriNi1DgpQE8G2-cn-RFcN3AsZM79aHr1CcRRdgNMg0XdY | **Свойства доменов и диапазонов в OWL**-Важно понимать, что в OWL домены и диапазоны не следует рассматривать как препятствия для проверки. Они используются как «аксиомы» в аргументации. Например, если свойство hasTopping имеет домен класс Pizza и мы затем применяем свойство hasTopping для мороженого (индивиды которого являются членами класса мороженое) это правило не приведет к ошибке. Оно будет использоваться для вывода, что класс мороженого должен быть подклассом  класса Pizza! |

Теперь мы хотим указать, что свойство hasTopping имеет диапазон PizzaTopping. Для этого используется панель диапазона (Description), показанная на рисунке 4.27.

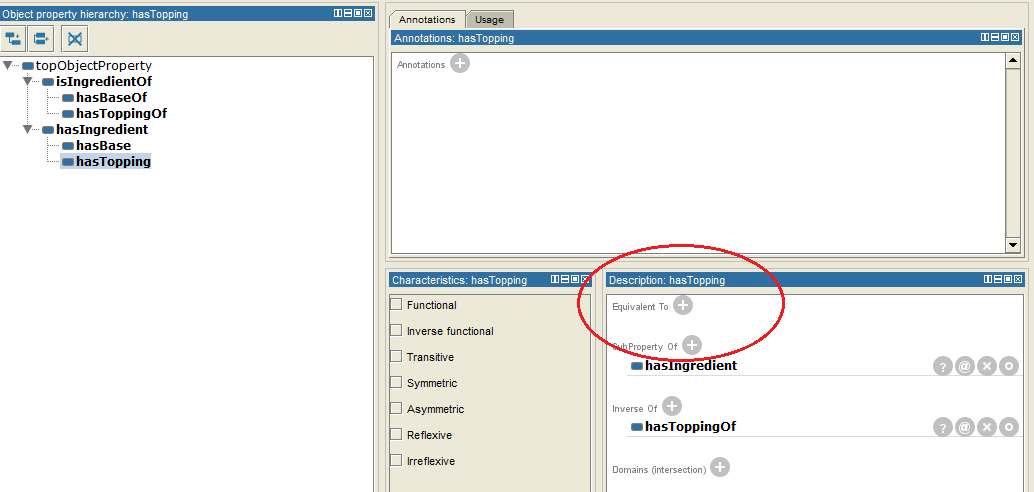


Рисунок 4.27: Панель описания свойств (для свойств объекта)

**Упражнение 13: Укажите диапазон hasTopping**

|  |  |
| --- | --- |
|  | 1. Убедитесь, что выбрано свойство hasTopping в иерархии свойств на вкладке «Свойства объекта» |
|  | 2. Нажмите значок «Добавить» (+) рядом с «Ranges (Intersection)» на панели «Описание свойства» (Рисунок 4.27). Появится диалоговое окно, которое позволяет выбирать классы из иерархии классов онтологии (рисунок 4.27а). |
|  | 3. Выберите PizzaTopping и нажмите кнопку «OK».Класс PizzaTopping теперь должен быть отображены в списке диапазона. |

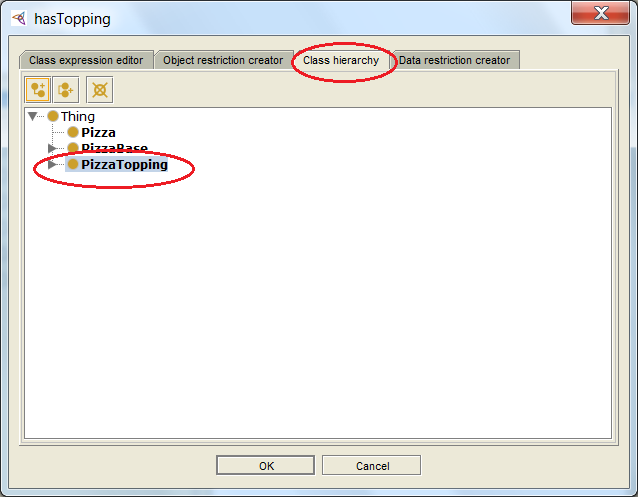


Рисунок 4.27а. Панель выбора класса в диапазон свойства

|  |  |
| --- | --- |
| https://lh3.googleusercontent.com/1KYzZim0VWO0hRgPvGvdzqvIv9vDrtWChGWhWxfFtTOSxCQLcT7ju9XM-E_BNH1yhIGzQOg5loqKRn7pwlW8c2GYPSJmtfPRW0KqRNrcM2OSIDv7bcDETNYa16E | Можно указать несколько классов, в качестве диапазона для свойства. Если в Protege 4 указано несколько классов в качестве диапазона свойства, то интерпретируется пересечение  классов. Например, если для свойства **has** (**имеет**) в представлении диапазона перечисленны класс **мужчина** и класс **женщина**, то диапазон свойства будет интерпретироваться как мужчины в браке с женщинами. (*т.е. все мужчины, которые* ***имеют*** *женщину, и все женщины, которые* ***имеют*** *мужчину –* ☺). |

Чтобы задать домен свойства используется панель домена, показанная на рисунке 4.28.

**Упражнение 14: Укажите класс Pizza как домен свойства hasTopping**

|  |  |
| --- | --- |
|  | 1. Убедитесь, что выбрано свойство hasTopping в иерархии свойств на вкладке «Свойства объекта». |
|  | 2. Нажмите значок «Добавить» (+) рядом с «Domains(Intersections)» на панели «Описание свойств». Появится диалоговое окно, которое позволит выбирать класс из иерархии классов онтологии. |
|  | 3. Выберите Pizza и нажмите на кнопку ОК. Класс Pizza теперь должен отображаться в списке доменов. |

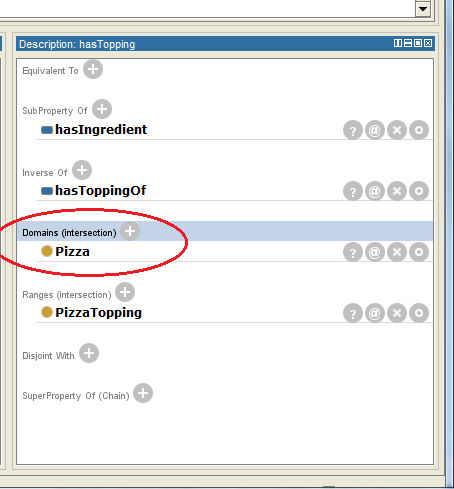


Рисунок 4.28: Вид домена свойства hasTopping

|  |  |
| --- | --- |
| https://lh5.googleusercontent.com/wF0JAJbxOMDmqJSuYepUVnKbcJEh85ZOx4OdjCF0QB8VKCnumZ1MXFpUcA7KAcdY94N0RnaqiJ3oa8qwKTBqeQW9O1I5EkjtJBVuzEFWZf_ypnN7ZpEiAlkX7oc | То, что мы сделали будет означать, что индивиды, которые используются на стороне домена (слева) свойства hasTopping будут интерпретированы в качестве членов класса Pizza. Любые индивиды, которые используются на стороне ранга (справа) свойства hasTopping будут интерпретироваться в качестве членов класса PizzaTopping. Например, если у нас есть индивиды ***а*** и ***b*** и утверждение в форме: ***a*** hasTopping ***b***, то это будет означать, что ***а*** является членом класса Pizza, и что ***b*** является членом класса PizzaTopping .  Это будет случае, даже если *а* не объявлен членом класса Pizza и/или *b* не объявлен членом класса PizzaTopping. |

|  |  |
| --- | --- |
| https://lh3.googleusercontent.com/nFSh1q2WDcctxFkD1Z2nUGhqdYkTuP1J7ejYwefaKo6y-VcgzDHZnE0BHOEWSbodSGSibkY4dVZYoHZwfc865ncu65-CFGaj0pfuyBRG2C1-dx5hAZBCft57Ik8 | Посмотрите на свойство isToppingOf, которое является обратным свойству hasTopping. Обратите внимание, что Protege были автоматически заполнены домен и диапазон свойства isToppingOf потому, что были указаны домен и диапазон обратного свойства. Диапазон isToppingOf - это домен обратного свойства hasTopping, и домен isToppingOf — это диапазон обратного свойства hasTopping. Это показано на рисунке 4,26.  ☺: Кстати Protégé 4.2 автоматически этого не сделал при выключеном реазонере. Нужно в настройках реазонера включить контроль домена и диапазона и стартануть реазонер, тогда автоматически будут определяться домен и диапазон для инверсных свойств |

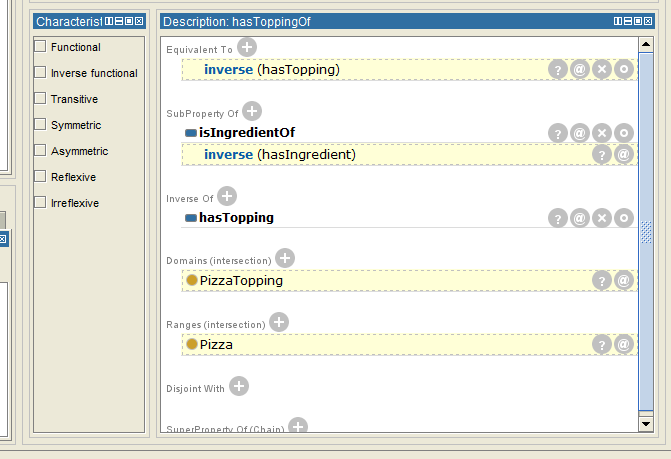


Рисунок 4.26(☺). При включенном реазонере автоматически определены домен и диапазон обратного свойства – изменения недоступны

**Упражнение 15: Укажите домен и диапазон для свойства hasBase и его обратное свойство — BaseOf**

|  |  |
| --- | --- |
|  | 1. Выберите свойство hasBase. |
|  | 2. Укажите Pizza как домен свойства hasBase. |
|  | 3. Укажите PizzaBase как диапазон свойства hasBase |
|  | 4. Выберите свойство isBaseOf. Обратите внимание на то, что домен isBaseOf - это диапазон обратного свойства hasBase, и что диапазон isBaseOf - это домен обратного свойства hasBase. |
|  | 5. Сделать свойству isBaseOf домен PizzaBase. |
|  | 6. Сделайте свойству isBaseOf диапазон  Pizza.  ☺ - здесь как раз тот случай, когда реазонер выключен и это надо делать вручную |

|  |  |
| --- | --- |
| https://lh6.googleusercontent.com/-N5e3EMVe0LeLCiEtW52KUDL1610PhpX-w-81v-zV-cmN_Ztcuaw6PdMON6huYnDaImUCbjKFMn4kwzHe8DuBoSxGamqOBTEPsXv1nyufae_yLipy1zSAa0WymI | На предыдущих этапах мы показали, что домены и диапазоны для свойств также настраиваются для обратных свойств правильным образом. В общем, домен для прямого свойства — это диапазон для ее обратного свойства и диапазон значений для прямого свойства — это домен для его обратного свойства — 4.26 рисунок иллюстрирует это для hasTopping и isToppingOf. |
| https://lh5.googleusercontent.com/GFHkUN9bz2emPN6oUXGOpPELv-9ax2EOy9LQ_zD9GBAOAK_0vAau9UN9XtIaGjSplswpncskH6suTX26OOBFln1ifNShPpawC8zxAvjoRZ4SAulvesldk-v7RjM | Хотя мы установили домены и диапазоны различных свойств для целей данного учебного курса, мы обычно не советуем делать это. Тот факт, что домен и диапазон условия не являются ограничениями и тот факт, что они могут вызвать «неожиданные» классификации результатов, привносят проблемы и неожиданные побочные эффекты. Эти проблемы и побочные эффекты особенно трудно отслеживаются в большой онтологии. |