**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра Систем информатики

Направление подготовки 09.06.01 – Информатика и вычислительная техника

**ОТЧЕТ**

**Обучающегося** Козубенко Андрея Алексеевича **группы № 22215 курса 4**

(Ф.И.О. полностью)

**Тема задания**: Разработка репозитория для редактирования онтологии предметной области

Новосибирск 2025

Оглавление

[Введение 3](#_8xl0k4o7g53y)

[Реализация репозитория онтологий 4](#_18lqawz87498)

[Листинги](#_18lqawz87498) 5

[Заключение](#_ftfmarjidp06) 10

[Список литературы (если будет)](#_8i44esq2u94t) 11

# Введение

Целью данной работы является разработка Python-репозитория для работы с онтологиями предметной области, хранящимися в графовой базе данных Neo4j.

Для достижения цели были поставлены следующие задачи:

* реализовать методы для управления классами онтологии (создание, обновление, удаление, получение родителей и потомков);
* реализовать методы для работы с объектами классов (создание, обновление, удаление);
* реализовать поддержку атрибутов классов: DatatypeProperty и ObjectProperty;
* реализовать сбор сигнатуры класса (collect\_signature), включающей все его свойства;
* реализовать метод получения всей онтологии и корневых классов.

# Реализация репозитория онтологий

# Для реализации лабораторной был создан класс **OntologyRepository**, наследующийся от Neo4jRepository. В нем были реализованы следующие группы методов:

# **Работа с онтологией**:

# get\_ontology() – получение всей онтологии;

# get\_ontology\_parent\_classes() – поиск корневых классов;

# get\_class(), get\_class\_parents(), get\_class\_children() – работа с отдельным классом.

# **Управление классами**:

# create\_class(), update\_class(), delete\_class() – создание, редактирование и удаление классов вместе с потомками и атрибутами.

# **Атрибуты классов**:

# add\_class\_attribute() / delete\_class\_attribute() – работа с DatatypeProperty;

# add\_class\_object\_attribute() / delete\_class\_object\_attribute() – работа с ObjectProperty.

# **Объекты классов**:

# create\_object(), update\_object(), get\_object(), delete\_object().

# **Сигнатуры**:

# collect\_signature() – сбор информации обо всех параметрах (DatatypeProperty) и объектных связях (ObjectProperty).

**Тестирование и пример использования**

Пример сценария (см. example\_usage\_ontology.py)

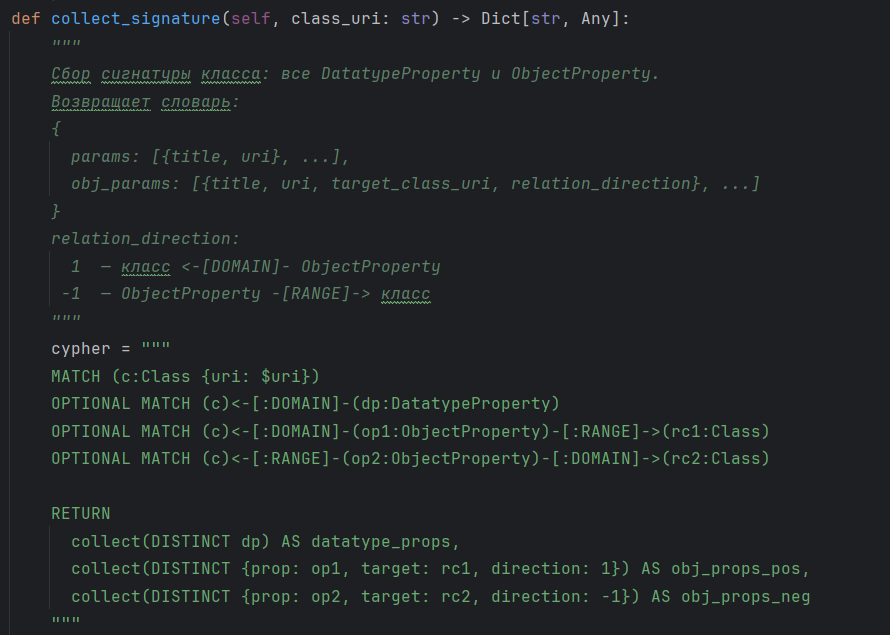
**Реализация**

[**https://github.com/andrew-kozubenko/Linguistics**](https://github.com/andrew-kozubenko/Linguistics)

Находится в **lab2/ontology\_repository.py**

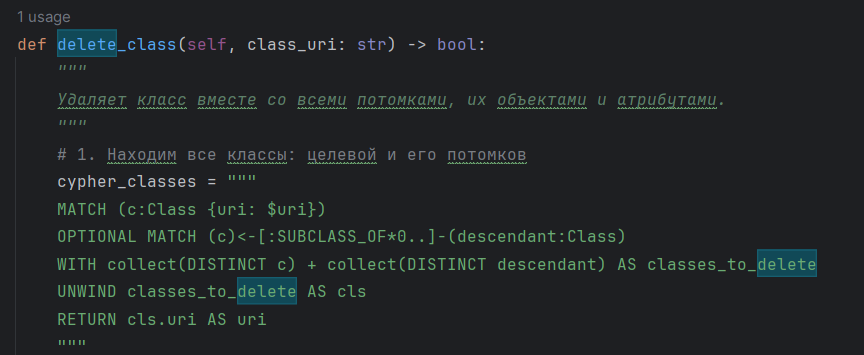
# Листинги

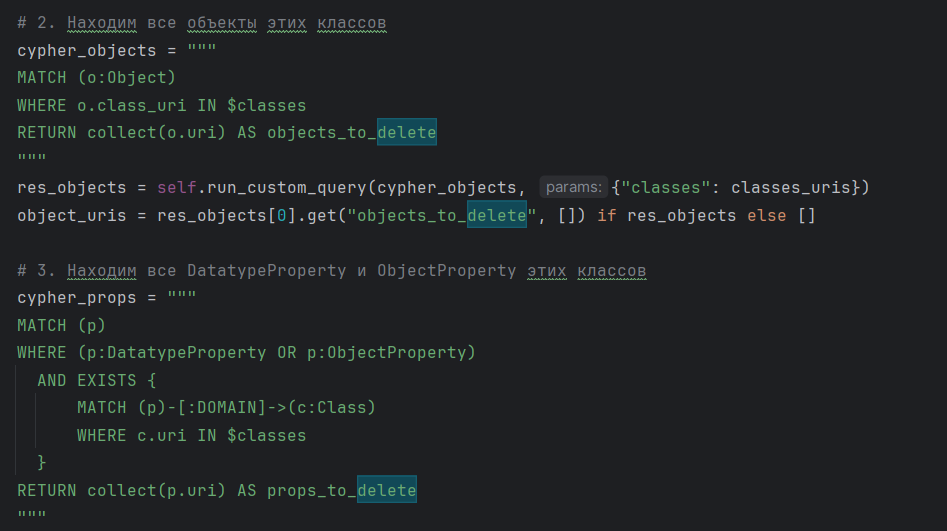
Листинг 1 – Метод для сбора сигнатуры класса:



Данный метод позволяет собрать все параметры и связи класса, что делает возможным динамическое построение объектов.

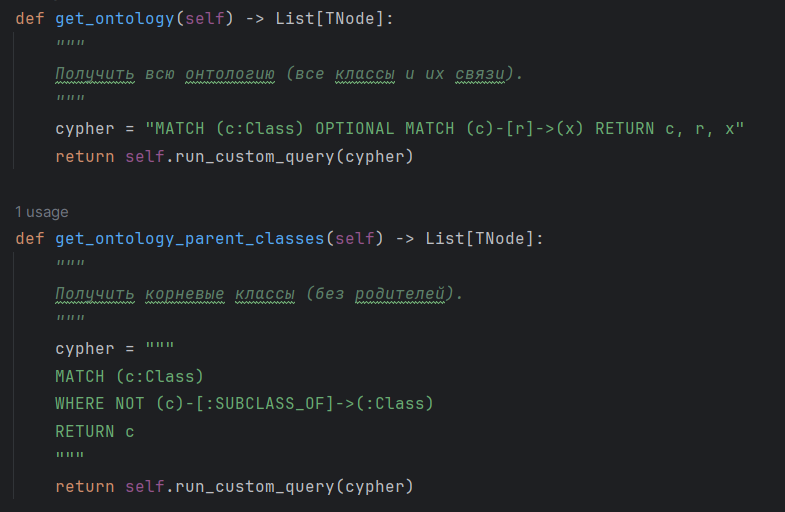
Листинг 2 – Метод для удаления класса вместе с его потомками, объектами и атрибутами:

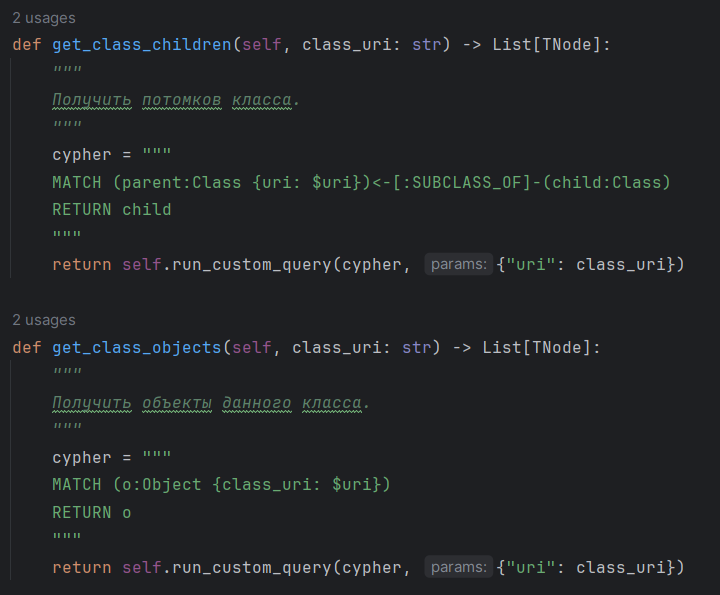


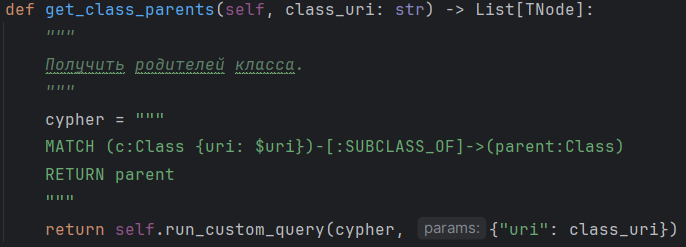


Данный метод позволяет удалить класс вместе с его потомками, объектами и атрибутами.

Листинг 3 - Геттеры

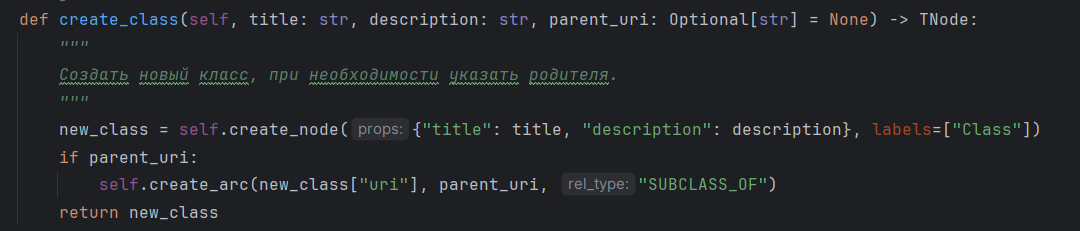


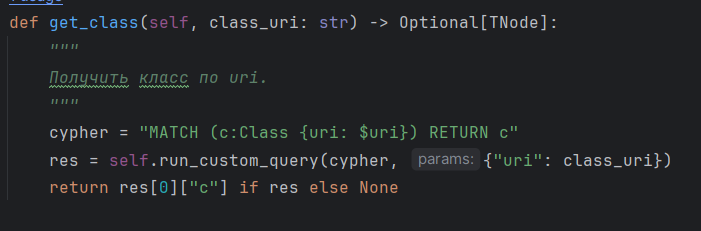


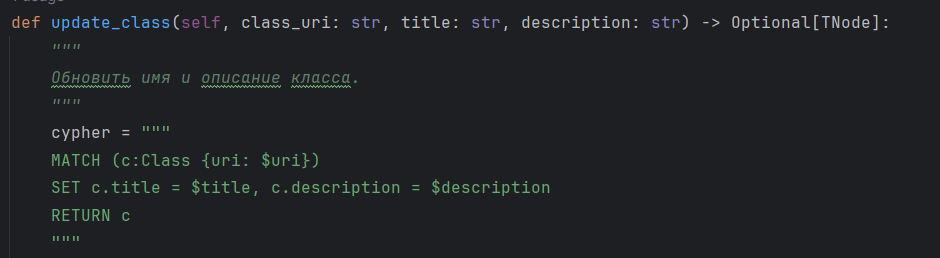


Позволяют получать данные

Листинг 4 – CRUD для классов (delete\_class описан в листинге 2 из-за своей специвики)



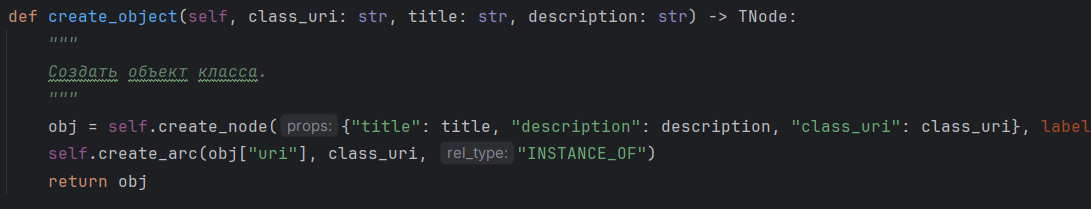
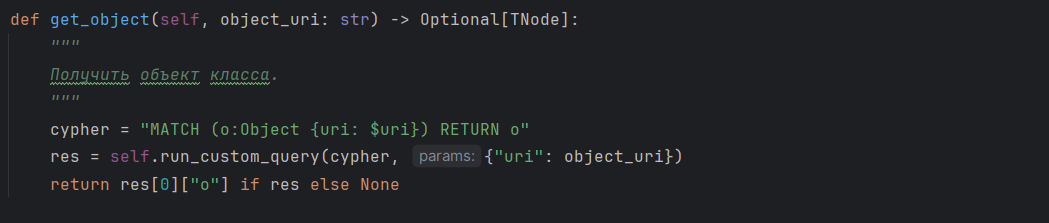


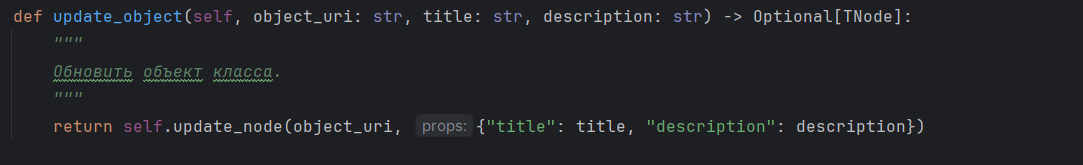


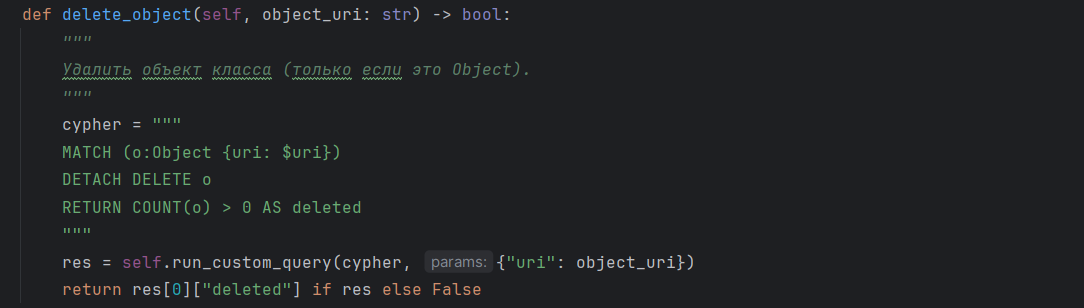
Листинг 5 – Создание и удаление атрибутов класса



Листинг 6 – CRUD для объектов





# Заключение

В результате проведенной работы были реализованы:

1. Методы для работы с классами и объектами онтологии.
2. Методы для добавления и удаления свойств классов (DatatypeProperty, ObjectProperty).
3. Метод сбора сигнатуры класса для использования при создании и обновлении объектов.
4. Полный набор операций для редактирования онтологии в Neo4j.

Таким образом, все задачи, поставленные во введении, были выполнены.

# Список литературы

1. Neo4j documentation: <https://neo4j.com/docs>