Assignment 3 - Programming Semantic Systems (17,5 pts)

Instructions

Deadline

Make sure to upload your results **by December 09th, 2022**!

What you should hand in

Please upload your work to TUWEL by December 09th, 2022. Please name all your files with the following format: A[x]_[family name]_[student number], where x represents the number of the assignment.

Your submission should be a zip file (**e.g., A3_Mustermann_1234.zip**) that contains an executable file, complete source codes and a documentation file (**A3_Mustermann_1234.PDF**).

Questions

Please post general questions in the TUWEL discussion forum of the course. You can also discuss problems and issues you are facing there. We appreciate it if you help other students out with general problems or questions regarding the tools used. For obvious reasons, however, please do not post any solutions there.

You can also contact Filip Kovacevic and Laura Waltersdorfer directly (with specific questions) at:

- filip.kovacevic@tuwien.ac.at
- laura.waltersdorfer@tuwien.ac.at

Please use subject line ISS_2022_<your subject> to minimize the probability that your Email gets lost in our inbox.

Assignment 3 - Programming Semantic Systems (17,5 pts)

Introduction

In this assignment, you will exercise important skills and technologies needed for developing a basic semantic application enabled by a knowledge graph. In particular, you will follow these two tasks (each task is described in detail in the next section):

1. Create a simple program that accesses a knowledge graph.
2. Create a documentation file (PDF file) for the application.

Tasks description

Task 1: Development of a simple application that accesses a Knowledge Graph (12,5 points)

In this task, please create an **interactive command-line application** that programmatically accesses and modifies a knowledge graph and presents it to the user. Similar to the Assignment 2, you are free to choose whether to use the example movie ontology [1] (from Assignment 2) or your own ontology (from Assignment 1, to which you might need to add some instances) for this assignment.

For simplicity, we ask you create your application using Apache Jena [2], RDF4J [3], or RDFLib [4] plus OWLRL reasoner engine [5] with *an in-memory storage*. To help you with the development process, we provided example applications for these three options in a GitHub repository [6] as well as the suggested version of the libraries. You may want to consider using additional libraries (e.g., *Apache Common CLI* for Java [7] or *Click* for Python [8]) to build the interactive command-line part of the application – but this is **not** mandatory.

The CLI application will need to provide a menu where users can choose between a set of possible operations described below (**3p**). After finishing an operation, the application will need to present the result of the operation to the user and ask users for the next operation until the users choose the “exit” operation. The application will need to at least include the following operations:

1. Basic operations
    a. (**LoadOntology**) Load an initial ontology with instances, e.g., film ontology (**1p**)
    b. (**ActivateReasoner**) Options for user to select and activate reasoners provided by the chosen APIs (**1p**)
    c. (**ExportGraph**) Export the RDF graph in a user-selected format (**1p**)
2. Ontology operations
    a. (**AddClass**) Add a new ontology class based on user inputs (**1p**)
    b. (**AddProperty**) Add a new property based on user inputs, including domain & range definition (**1p**)
    c. (**AddInstance**) Add ontology instances based on user inputs **OR** by importing a CSV file (**1p**)
3. Basic query operations

   a. (**ShowQueries**) Show a list of SPARQL queries[1] that you have developed in Assignment 2 to users and allow them to choose a query (**1p**).
   b. (**ExecuteSelectQuery**) For SPARQL select queries, execute the user-selected query on the RDF graph and show the query results on the command line (**1p**)
   c. (**ExecuteConstructQuery**) For SPARQL construct queries, save the query result as a turtle file (**1p**).
4. (**Exit**) Exit. (**0,5p**)

In the deliverable, you will need to provide both an executable file and the complete source code. To ensure a smooth grading process, we would **require** you to comply to the following guideline:

- For a Java app, the executable file shall be named **kg_app.jar,** and it should be able to run using the command "**java -jar kg_app.jar**" within Java 11 environments.
- For a Python app, the executable file shall be named **kg_app.py,** and it should be able to run using the command "**python3 kg_app.py**" within Python 3.8 environments.

You can earn up to **2,5 bonus points** if you create a fancy UI (web/native/mobile app) on top of the command line application.

Task 2: Writing a documentation file (5 points)

In this task, you will need to create a short documentation of your application, including (i) **what are the implemented features** in your application and (ii) **where does the source code for each implemented feature are located**, and (iii) **how to execute each feature**. This information is important for us during the grading process and should be provided in a PDF file.

Tools and resources:
[1] film.ttl ontology (also in TUWEL): http://semantics.id/ns/example/film#
[2] Apache Jena: https://jena.apache.org/
[3] Eclipse RDF4J: https://rdf4j.org/
[4] RDFLib: https://github.com/RDFLib/rdflib
[5] OWL-RL for RDFLib: https://github.com/RDFLib/OWL-RL
[6] https://github.com/fekaputra/course-tutorial
[7] https://github.com/apache/commons-cli
[8] https://github.com/pallets/click

[1] *if you didn't do Assignment 2, please create a minimum of 5 queries for this task*