Implementationdescription of the SAIQL Query Engine

Tools and Libraries

The SAIQL Query Engine was developed with the programming language Java and the IDE Eclipse. The following tools and libraries are used:

- WonderWeb OWL API: The included OWL Model is used for intern storage of the OWL ontologies.
- Pellet API: The reasoner Pellet is connected over its own API. It is used for consistency checks.
- JavaCC: JavaCC was used for the development of a Parser, that checks the syntax of the SAIQL-Queries and translates the SAIQL-Queries in a equivalent internal model. The SAIQL-Queries are stored within this model.

Softwarestructure

The software is structured in five packages:

- Sq2.base: This package contains the base classes of the SAIQL-Query Engine. They are responsible for the correct evaluation of the SAIQL-queries.
- Sq2.gui: The GUI of the SAIQL Query Engine is located in this package.
- Sq2.parser: The Parser, that is used for reading the SAIQL-queries.
- Sq2.render.abstractSyntax: The Renderer generates the OWL Abstract Syntax of an OWL ontology.
- Sq2.SAIQL: This package consists of the internal model of the SAIQL-queries and the internal representation of OWL-axioms is there stored.

Example Record

The directory „SAIQLQueryEngine“ contains the data record which is used in the example and in the paper. The record is an OWL ontology with the title „MotorOntology.owl“.

Special Notes

The following information is for a better understanding of the implementation:

- The „Original Ontology“ is the ontology from which the elements are extracted.
- The result of a SAIQL Query is also an OWL Ontology. It is called the „Resulting Ontology“.
- An axiom is a conventional OWL Axiom, e.g. SubClass, Disjoint, etc.
- An axiom-pattern is a special SAIQL construct. An OWL axiom is advanced with SAIQL variables.

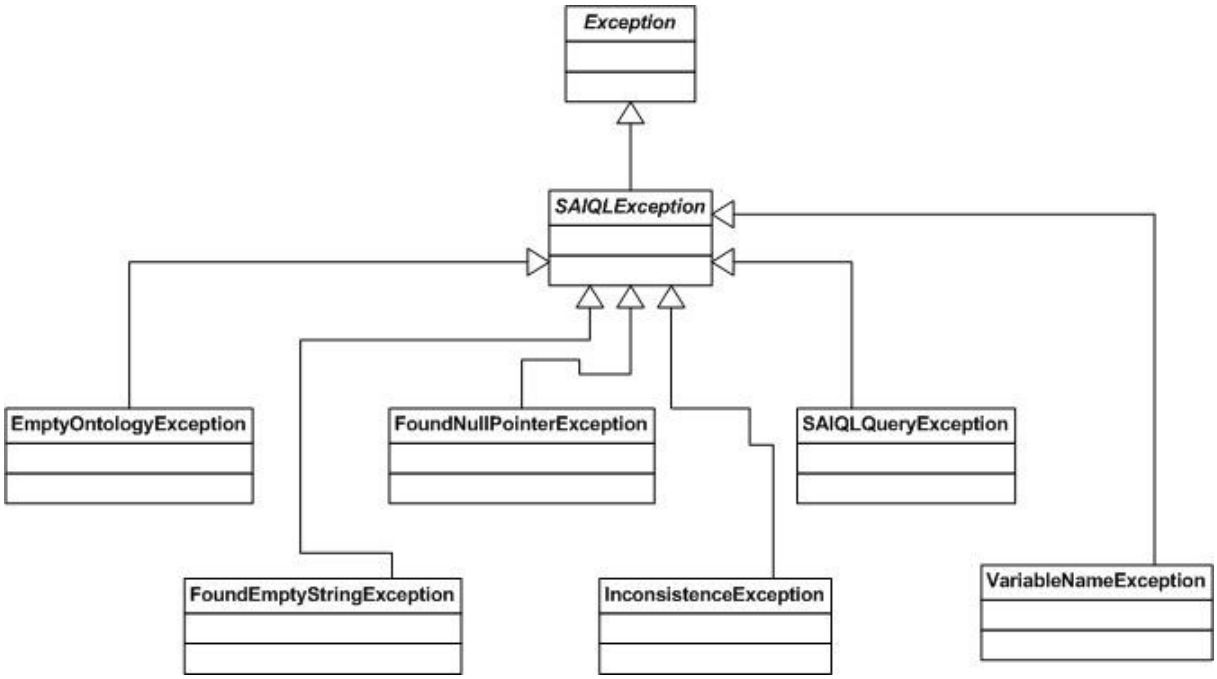
- There are three kinds of SAIQL-variables:

  → ClassName-variables: A variable for a OWL classname. It represents an atomic OWL class.
  → ClassDescription-variables: A variable for a complex OWL class, i.e. a class definition.
  → IndividualName-variable: A variable for an individual, i.e. an instance.

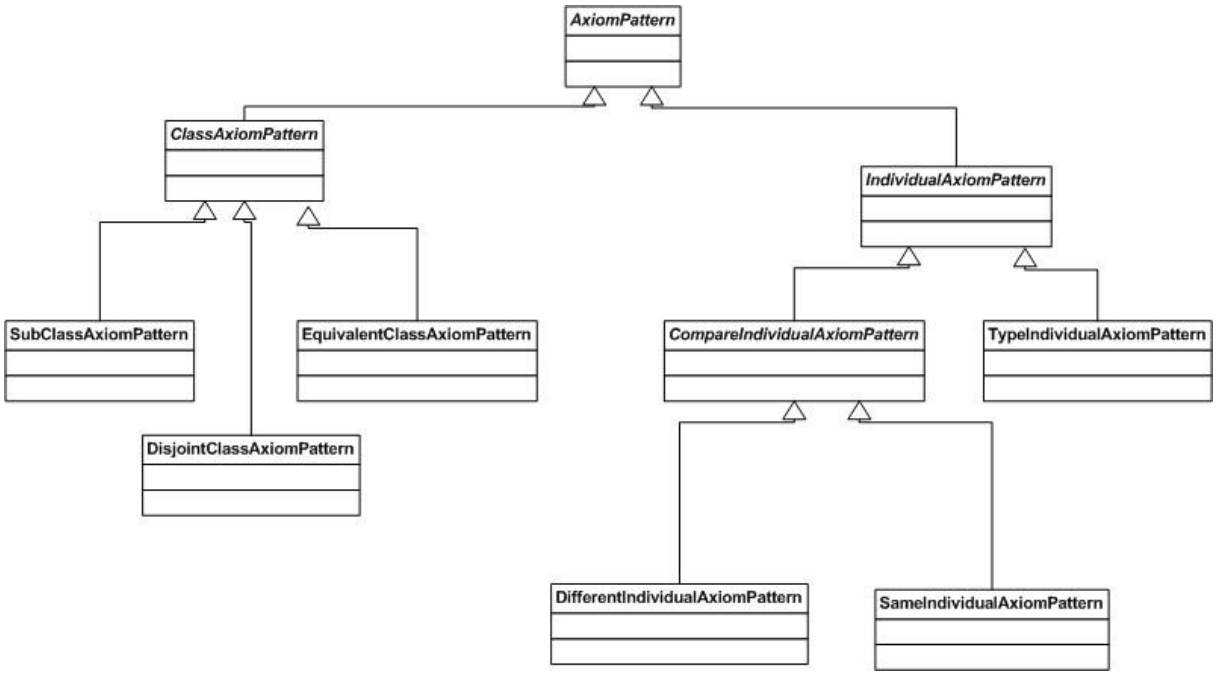
Selected classes and their UML-class-diagrams
In the following sections, important classes and the concerning UML class-diagrams are described. The UML class diagrams demonstrate the relations between the classes.

Exception: The superclass for all self-made exception classes is the class SAIQLException. The following exception classes are derived from this class:

- EmptyOntologyException: The original OWL ontology contains no OWL axioms.
- FoundEmptyStringException: An empty string, e.g. in an URI reference is appeared.
- FoundNullPointerException: There is a nullpointer within a SAIQL Query.
- InconsistenceException: The original OWL ontology is inconsistent. Therefore this ontology may not been used.
- SAIQLQueryException: This exception occurs if the SAIQL Query is syntactic incorret.
- VariableNameException: It appears if the same variablename is used for different kinds of OWL-elements (e.g. instances, classes). Another reason for this exception is the use of a variablename, which is not declared in the LET-clause before.



Internal modeling of the axiom-patterns:
To store the SAIQL-query internal, the axiom-patterns that are used in the SAIQL-query are transformed in an internal model. For this step the following classes are used. The superclass is the abstract class `AxiomPattern`. From this class the classes `ClassAxiomPattern` and `IndividualAxiomPattern` are derived. Further specializations are derived from this two classes (see figure below).



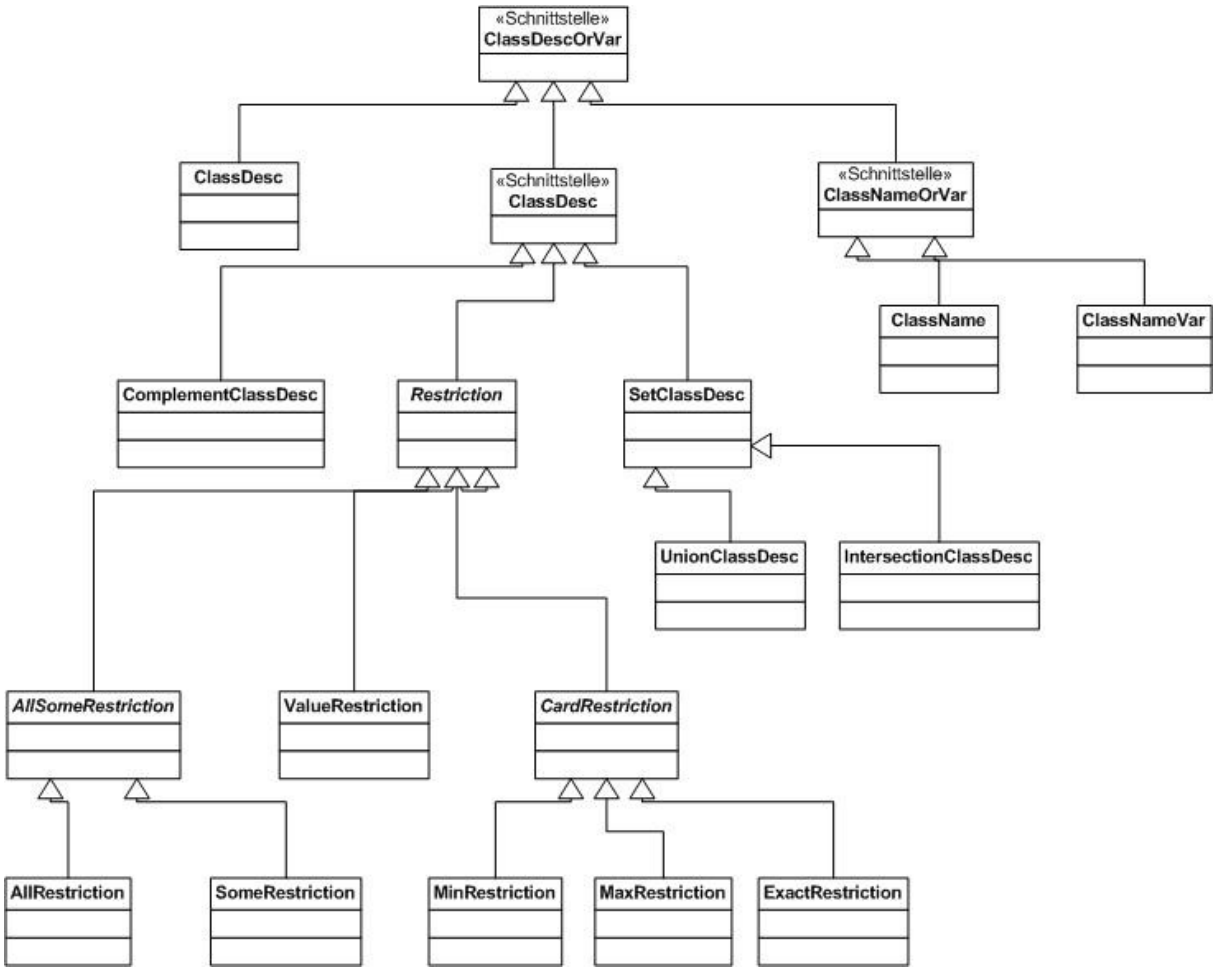
Internal modeling of the OWL-axioms:
Also the OWL-axioms are internal modelled. Thus for each Axiom-pattern there exists an corresponding OWL-axiom, where all variables are substituted by concrete values, i.e. class names, instances, etc.

Internal modeling of OWL-instances (individuals) and their variables:
SAIQL-expressions can contain concrete values for OWL-instances and also variables, representing OWL-instances. Therefore the abstract class `IndividualNameOrVar` exists. From this class the classes `IndividualName` and `IndividualNameVar` are derived. The class IndividualName is used for the names of OWL-instances. The class IndividualNameVar is designed for the SAIQL-variables. The SAIQL-variables represent names of OWL-instances.

Internal modeling of atomic OWL classes and their variables:
SAIQL-expressions may contain concrete values for OWL-classnames and also variables, representing OWL-classnames. Therefore the abstract class `ClassNameOrVar` exists. From this class the classes `ClassName` and `ClassNameVar` are derived. The class `ClassName` is used for the names of OWL-classnames and accordingly the class `ClassNameVar` is used for SAIQL-variables. OWL- classnames are for atomic OWL-classes.

Internal modeling of atomic and complex OWL-classes and their variables:
For administrating complex OWL-classes and their variables in SAIQL-expressions, the interface `ClassdescOrVar` is used. From this interface, the following classes and interfaces are derived:

- ClassDescVar: Is designed for variables, representing complex OWL-classes.
- ClassDesc: is used for complex OWL-classes.
- ClassNameOrVar: is used for atomic OWL-classes and their variables.



The Process of Query-Evaluation
The following sections consider the evaluation of SAIQL-queries. We assume that the original ontology is already loaded.

The description below adverts especially to the method startMain in the class SAIQLQueryEngine:

1. The SAIQL-Query, which is original a string is internal treated as an object of the class SAIQLQuery.
2. From the original ontology all classnames, komplex classes and the names of instances are extracted. The following objects are generated:
   ```
   ConceptNameSet cnSet
   InstanceNameSet inSet
   ConceptDescriptionSet cdSet
   ```
3. The set of all (syntactic) possible solutions is built, by generating the cartesian product of the set the classnames, instances and complex classes. (For details see the SAIQL-paper).
4. In an object of the class `WhereClauseEvaluation` all elements of the set of possible solutions are checked, if they are valid concerning

the conditions from the WHERE-clause. The set of valid elements build the solution.

5. In an object of the class `ConstructClauseEvaluation` for each element of the set of valid solutions, new OWL-axioms are created, which are displayed as resulting ontology.