**package** synalp.commons.input.knowledgeBase;

**import** java.io.IOException;

**import** java.util.Set;

**import** com.hp.hpl.jena.ontology.DatatypeProperty;

**import** com.hp.hpl.jena.ontology.Individual;

**import** com.hp.hpl.jena.ontology.ObjectProperty;

**import** com.hp.hpl.jena.ontology.OntClass;

**import** com.hp.hpl.jena.rdf.model.Resource;

**import** com.hp.hpl.jena.util.iterator.ExtendedIterator;

**public** **interface** IOntologyAnalysis {

// Method that provides the list of the ontology's classes

/\*\*

\* **@return** a Set of OntClass(Interface that represents an ontology node characterising a class

\* description)

\*/

**public** **abstract** Set<OntClass> getClasses();

// Method that creates a text from the label skos definition

/\*\*

\* **@param** fileTextFromKB that is text from Knowledge Base

\*/

**public** **abstract** **void** CreateTextFromDefinition(String fileTextFromKB) **throws** IOException;

// Method that provides the list of the ontology's datatypesPoperties

/\*\*

\* **@return** an ExtendedIterator of DatatypeProperty(Interface that encapsulates the class of

\* properties whose range values are datatype values)

\*/

**public** **abstract** ExtendedIterator<DatatypeProperty> getDatatypeProperties();

// Method that provides the list of the ontology's objectPoperties

/\*\*

\* **@return** an ExtendedIterator of ObjectProperty(Interface encapsulating properties whose range

\* values are restricted to individuals)

\*/

**public** **abstract** ExtendedIterator<ObjectProperty> getObjectProperties();

// Method that provides the list of the ontology's individuals

/\*\*

\* **@return** a Set of Individual(Interface that encapsulates an individual in an ontology, sometimes

\* referred to as a fact or assertion, or a member of the a-box. In order to be recognised

\* as an individual, rather than a generic resource, at least one rdf:type statement,

\* referring to a known class, must be present in the model)

\*/

**public** **abstract** Set<Individual> getIndividuals();

// Method that provides the list of all ontology's concepts

/\*\*

\* **@return** a Set of Resource(An RDF Resource)

\*/

**public** **abstract** Set<Resource> getOntoConcepts();

// Method that provides the resources corresponding to a word

/\*\*

\* **@param** word which will be linked.

\* **@return** an OntClass(Interface that represents an ontology node characterising a class

\* description)

\*/

**public** **abstract** OntClass getResource(String word);

// Method that checks if two classes are disjoint or not

/\*\*

\* **@param** c1 that is OntClass (Interface that represents an ontology node characterising a class

\* description)

\* **@param** c2 that is OntClass (Interface that represents an ontology node characterising a class

\* description)

\* **@return** true or false

\*/

**public** **abstract** **boolean** isDisjoint(OntClass c1, OntClass c2);

}