Assignment#5

***Note:**
Assignment#5 contributes to 10% of the total mark of the course.
Deadline: June 30, 2020 before 14:00 (provisional)
Submission: Send your work as an A#5-XXX.zip file, for XXX your student ID, via the PedagogySquare platform. The zip file should contain a .java file.

Q1. CW5 (CourseWork 5) is all about querying implementing and OWL based querying system. Your task will be to query the reasoner for sub-classes, equivalent classes and instances, and store the results in a QueryResult object.

You are asked to implement the following methods in the Java class CW5 (project layout the same as CW3!). Do not alter any class other than CW5, which is the only one you are (allowed) to submit!

The classes App.java and PizzaOrderingSystemApp.java provide a, hopefully fun, way to test what you are doing is correct. You can run them in the same way as you ran the unit tests in CW3, simply by hitting the small white and green arrow icon in Eclipse when the class is open. If you are not interested in having fun, then you can, in the same way as you did last time, use the implemented unit tests to figure out whether you are on the right track. Please note that this time passing the unit tests does not automatically mean full marks. We will use some of our own tests to further validate your solution. Once you are done, please zip and submit CW5.java to the PedagogySquare platform. Good Luck!

Tasks:

1. `public Set<QueryResult> performQuery (OWLClassExpression exp, QueryType type) {…}`

This method takes in an arbitrary expression in OWL, like “Person and hasBirthYear value 1964” or “hasTopping some MeatTopping” and queries the ontology for the following three types of knowledge:
* EquivalentClasses: you are asked to return all those (named) classes that are equivalent to the expression (exp), using the (already fully initialized) reasoner.
* Sub-classes: return all those classes that are (named) sub-classes of the expression, using the reasoner.
* Instances: return all those individuals that are instances of the expression, using the reasoner.

Query results are stored in query result objects. These are created for example as follows:

```
QueryResult qr = new QueryResult(ind, false, type);
```

Note that you need to include the information whether the inference is direct or indirect. Example: Given three classes with A subclass B subclass C, then A is a

direct subclass of B and B is a direct subclass of C, but A is an indirect subclass of C (through B!). In order to solve this problem, you will query the reasoner for direct and indirect sub-class separately. After creating the QueryResult, add it to the provided set.

2. `public Boolean isValidPizza (OWLClassExpression exp) {…}`
In this method, you check whether the supplied class expression exp is a valid pizza expression, i.e., whether it is inferred to be a Pizza.

3. `public Set<QueryResult> filterNamedPizzas (Set<QueryResult> results) {…}`
This question is similar to the one before, only that you are asked to filter from a set of results those that correspond to NamedPizza's, such as Margherita or AmericanHot.

4. `public Set<OWLClassExpression> getAllSuperClassExpressions (OWLClass ce) {…}`
This question requires a bit of thinking. You are asked to query the ontology for all information you can find about the class ce. In order to get an idea of what "ALL" means, you can open Protégé and look at the "Classes" tab. If you click on a class, you will find that it often has not only super-classes and equivalent-classes, but also inherited super-classes (Anonymous Ancestors), and of course indirect super-classes. Unfortunately the reasoner will not easily give you access to those. In order to get full marks for this task, it is sufficient to query for all super-classes (direct and indirect, using the reasoner), and somehow find a way to obtain the anonymous super-classes using the ontology directly (that will need a bit of thought, do not despair too quickly). A perfect solution will test, for all sub-expressions in the ontology, whether ce is a sub-concept of it. Attempt this only if you are really confident with the OWL API by now.