Part 2. Java Program

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import org.apache.jena.ontology.OntModel;
import org.apache.jena.query.Query;
import org.apache.jena.query.QueryExecution;
import org.apache.jena.query.QueryExecutionFactory;
import org.apache.jena.query.QueryFactory;
import org.apache.jena.query.QuerySolution;
import org.apache.jena.query.ResultSet;
import org.apache.jena.rdf.model.ModelFactory;
import org.apache.jena.rdf.model.Resource;

public class RDFprocessing {

	public static void main(String[] args) {

		List<String> departments = new ArrayList<String>(); // departments in 1901.
		List<String> countries = new ArrayList<String>();
		List<String> universities = new ArrayList<String>();
		List<String> germans = new ArrayList<String>();
		List<String> germans_year = new ArrayList<String>();
		List<String> non_germans = new ArrayList<String>();
		List<String> non_germans_year = new ArrayList<String>();
```

```
String baseLaureat = "./libs/data.nt";

OntModel modelLaureat = ModelFactory.createOntologyModel();

modelLaureat.read(baseLaureat, "XML");

System.out.println("Loading is done...");

String query = "SELECT ?name ?other WHERE { ?name
        <http://data.nobelprize.org/terms/share> $other .}";

Query queryLaureat = QueryFactory.create(query);

QueryExecution qeLaureat = QueryExecutionFactory.create(queryLaureat,
                                                        modelLaureat);

ResultSet results = qeLaureat.execSelect();

while(results.hasNext()) {

        QuerySolution qs = results.next();

        Resource list = qs.getResource("name");

        String anotherQuery = "SELECT ?department " + "WHERE {<" +
                        list.toString() + ">
                <http://www.w3.org/2000/01/rdf-schema#label> ?
        department .}";

        Query depQuery = QueryFactory.create(anotherQuery);

        QueryExecution qe = QueryExecutionFactory.create(depQuery,
                                                        modelLaureat);

        ResultSet result = qe.execSelect();

        while(result.hasNext()) {

                QuerySolution temp = result.next();

                String department = temp.getLiteral("department").toString();

                if(department.indexOf("1901") > 0) {

                        // Saves the departments in the year of 1901.

                        departments.add(department.substring(0,
                        department.indexOf("1901") -1 ));

                }

        }
```

```
                qe.close();

        }

        qeLaureat.close();


        query = "SELECT ?name WHERE { ?name
    <http://data.nobelprize.org/terms/category>
<http://data.nobelprize.org/resource/category/Physiology_or_Medicine> . }";

        queryLaureat = QueryFactory.create(query);

        qeLaureat = QueryExecutionFactory.create(queryLaureat, modelLaureat);

        results = qeLaureat.execSelect();

        while(results.hasNext()) {

                QuerySolution qs = results.next();

                String laureatAwardString = qs.getResource("name").toString();

                if(laureatAwardString.contains("laureateaward")) {

                        String anotherQuery = "SELECT ?laureat WHERE { ?laureat
                <http://data.nobelprize.org/terms/laureateAward> <" +
        laureatAwardString + "> .}";

                        String universityQu = "SELECT ?university WHERE {<" +
                laureatAwardString + ">
                <http://data.nobelprize.org/terms/university> ?university .}";

                        Query laureatQuery = QueryFactory.create(anotherQuery);

                        Query universityQuery = QueryFactory.create(universityQu);

                        QueryExecution qe =
                QueryExecutionFactory.create(laureatQuery, modelLaureat);

                        QueryExecution anotherqe =
                QueryExecutionFactory.create(universityQuery, modelLaureat);

                        ResultSet result = qe.execSelect();

                        ResultSet anotherresult = anotherqe.execSelect();


                        while(result.hasNext()) {

                                QuerySolution temp = result.next();
```

```
                    String laureatString =
                temp.getResource("laureat").toString();


                    if(laureatString.contains("laureate")) {

                        String another = "SELECT ?country WHERE {<"
                + laureatString + ">
            <http://dbpedia.org/ontology/birthPlace> ?
        country .}";

                        Query countryQuery =
                    QueryFactory.create(another);

                        QueryExecution q =
                    QueryExecutionFactory.create(countryQuery,
                modelLaureat);

                        ResultSet res = q.execSelect();

                        while(res.hasNext()) {

                            QuerySolution anotherTemp = res.next();

                            String country =
anotherTemp.getResource("country").toString();

                            if(country.contains("country")) {

                                String[] countrys = country.split("/");


        if(!countries.contains(countrys[countrys.length - 1])) {

                                    String temp_country =
countrys[countrys.length - 1];

                                    String[] temp_universities =
anotherresult.next().getResource("university").toString().split("/");

                                    String temp_university =
temp_universities[temp_universities.length - 1];

                                    temp_country =
temp_country.replaceAll("_%28now_", "(");

                                    temp_country =
temp_country.replaceAll("%28", "& ");

                                    temp_country =
temp_country.replaceAll("%29", ")");
```

```
                                                    temp_country =
temp_country.replaceAll("_", " ");
                                                    temp_country =
temp_country.replaceAll("%26", " ");
                                                    temp_country =
temp_country.replaceAll("%2", " ");
                                                    temp_country =
temp_country.replaceAll("%3", " ");
                                                    temp_university =
temp_university.replaceAll("_", " ");
                                                    temp_university =
temp_university.replaceAll("%28", "& ");
                                                    temp_university =
temp_university.replaceAll("%29", "");
                                                    temp_university =
temp_university.replaceAll("%2C", "");
                                                    temp_university =
temp_university.replaceAll("%27s", "");
                                                    temp_university =
temp_university.replaceAll("%C3%A9", "");
                                                    temp_university =
temp_university.replaceAll("%C3%BCr", "");
                                                    temp_university =
temp_university.replaceAll("%C3%B6", "");
                                                    countries.add(temp_country);
                                                    universities.add(temp_university);
                                                }
                                            }
                                        }
                                        q.close();

                                    }
                                }
```

```
                    qe.close();
                    anotherqe.close();
                }
            }
            qeLaureat.close();

            query = "SELECT ?name ?country WHERE { ?name <http://dbpedia.org/ontology/birthPlace>
?country .}";
            Query queryGerman = QueryFactory.create(query);
            qeLaureat = QueryExecutionFactory.create(queryGerman, modelLaureat);
            results = qeLaureat.execSelect();
            while(results.hasNext()) {
                QuerySolution qs = results.next();
                String countryName = qs.getResource("country").toString();
                String queryName = "SELECT ?name WHERE {<" + qs.getResource("name") + ">
<http://xmlns.com/foaf/0.1/name> ?name .}";
                String queryDate = "SELECT ?date WHERE { ?date
<http://data.nobelprize.org/terms/laureate> <" + qs.getResource("name") + "> .}";
                QueryExecution qe = QueryExecutionFactory.create(QueryFactory.create(queryName),
modelLaureat);
                QueryExecution anotherqe =
QueryExecutionFactory.create(QueryFactory.create(queryDate), modelLaureat);
                ResultSet result1 = qe.execSelect();
                ResultSet result2 = anotherqe.execSelect();

                if(countryName.contains("Germany")) {
                    while(result1.hasNext()) {
                        germans.add(result1.next().getLiteral("name").toString());
                    }
                    while(result2.hasNext()) {
```

```
                        String date_laureat = result2.next().getResource("date").toString();
                        String[] dates_laureat = date_laureat.split("/");
                        if(date_laureat.contains("/nobelprize")) {
                            germans_year.add(dates_laureat[dates_laureat.length - 1]);
                        }
                    }
                } else {
                    while(result1.hasNext()) {
                        non_germans.add(result1.next().getLiteral("name").toString());
                    }
                    while(result2.hasNext()) {
                        String date_laureat = result2.next().getResource("date").toString();
                        String[] dates_laureat = date_laureat.split("/");
                        if(date_laureat.contains("/nobelprize")) {
                            non_germans_year.add(dates_laureat[dates_laureat.length -
1]);
                        }
                    }
                }
            }


            Iterator<String> iterDepartment = departments.iterator();
            Iterator<String> iterCountry = countries.iterator();
            Iterator<String> iterUniversity = universities.iterator();
            Iterator<String> iterGerman = germans.iterator();
            Iterator<String> iterGermanDate = germans_year.iterator();
            Iterator<String> iterNonGerman= non_germans.iterator();
```

```
		Iterator<String> iterNonGermanDate= non_germans_year.iterator();



	System.out.println("**************************************************************************
*********");
		System.out.println("Categories Awarded During The First Edition of The Nobel Prize (1901).");

	System.out.println("**************************************************************************
*********");
		System.out.println("");
		while(iterDepartment.hasNext()) {
			System.out.println(iterDepartment.next());
		}
		System.out.println("");



	System.out.println("**************************************************************************
*********");
		System.out.println("Countries That Have Physiology Or Medicine Prize Winners Along With
University.");

	System.out.println("**************************************************************************
*********");
		System.out.println("");
		while(iterCountry.hasNext()) {
			System.out.println(iterCountry.next() + " : " + iterUniversity.next());
		}
		System.out.println("");

	System.out.println("**************************************************************************
*********");
		System.out.println("Nobel Laureates With Year of Award(Germans & Non-Germans).");
```

```
        System.out.println("**********************************************************************
*********");

                System.out.println("");

                System.out.println("------------------------------   Germans   ----------------------------------");

                while(iterGerman.hasNext()) {

                        System.out.println(iterGerman.next() + " : " + iterGermanDate.next());

                }

                System.out.println("");

                System.out.println("------------------------------   Non-Germans   ----------------------------------");

                System.out.println("");

                while(iterNonGerman.hasNext()) {

                        System.out.println(iterNonGerman.next() + " : " + iterNonGermanDate.next());

                }

        }

}
```