**ΑΝΑΠΑΡΑΣΤΑΣΗ ΓΝΩΣΗΣ ΣΤΟΝ ΠΑΓΚΟΣΜΙΟ ΙΣΤΟ**  
2020-2021  
**2η Σειρά Ασκήσεων**

**Ονοματεπώνυμο:** Αδαμόπουλος Κωνσταντίνος

**ΑΜ:** 6270 (1043750)

**Ερώτημα 1**

**Α.** Παρακάτω παρουσιάζεται ο κώδικας του .rdf αρχείου που αναπαριστά την

πρόταση φυσικής γλώσσας :

*Η Ιλιάδα συνθέθηκε απο ποιητή που έζησε τον 8ο αιώνα π.Χ. στην Ιωνία της Μικράς Ασίας.*

<?**xml** version="1.0" encoding = "UTF-8" ?>

<rdf:RDF

xmlns:rdf="http:**//**www.w3.org/1999/02/22-rdf-syntax-ns**#**"

xmlns:uni="http:**//**www.mydomain.org/uni-ns/">

<rdf:Description rdf:about="Ηλιαδα">

<uni:composed\_from rdf:resource="ποιητη"/>

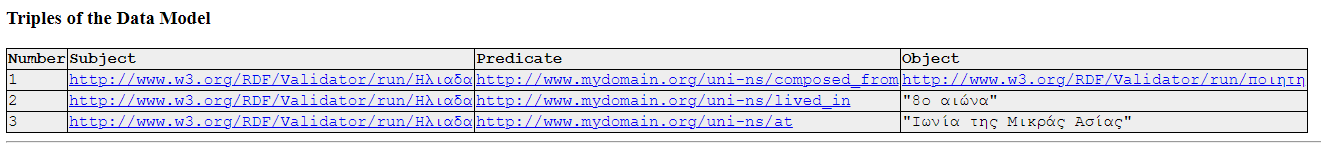
<uni:lived\_in>8ο αιώνα</uni:lived\_in>

<uni:at>Ιωνία της Μικράς Ασίας</uni:at>

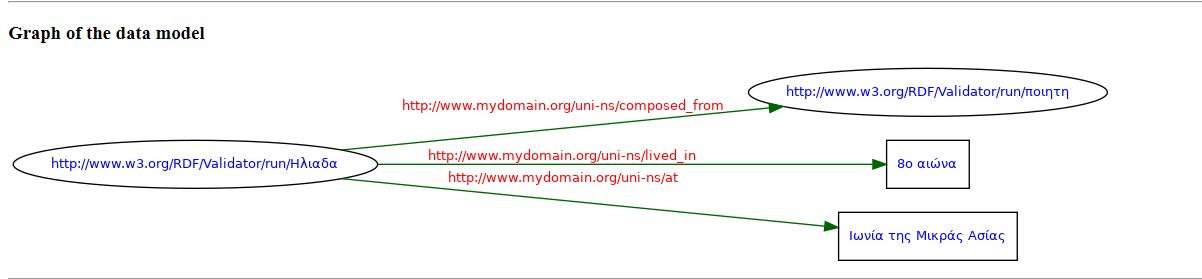
</rdf:Description>

</rdf:RDF>

Οι τριπλέτες είναι :



Και το γράφημα :



**Β.** Παρακάτω παρουσιάζεται ο κώδικας του .rdf αρχείου που αναπαριστά την παραπάνω πρόταση φυσικής γλώσσας χρησιμοποιώντας τουλάχιστον ένα κενό κόμβο:

<?**xml** version="1.0" encoding = "UTF-8" ?>

<rdf:RDF

xmlns:rdf="http:**//**www.w3.org/1999/02/22-rdf-syntax-ns**#**"

xmlns:uni="http:**//**www.mydomain.org/uni-ns/">

<rdf:Description rdf:about="Ηλιαδα">

<uni:poet rdf:resource="urn:uuid:fa123s"/>

</rdf:Description>

<rdf:Description rdf:about="urn:uuid:fa123s">

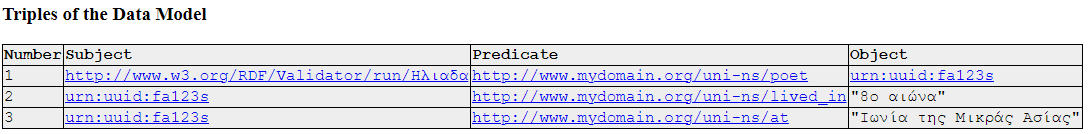
<uni:lived\_in>8ο αιώνα</uni:lived\_in>

<uni:at>Ιωνία της Μικράς Ασίας</uni:at>

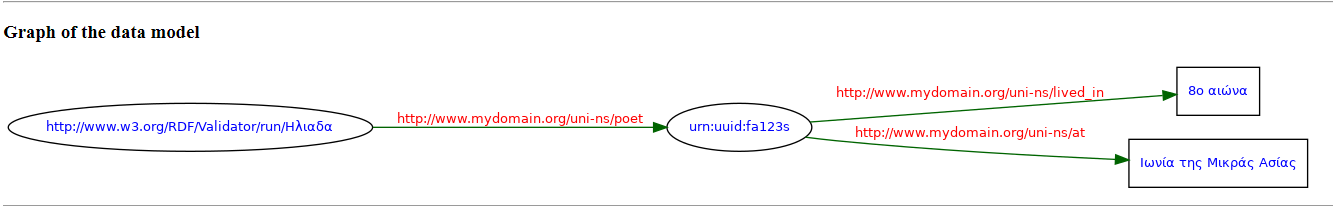
</rdf:Description>

</rdf:RDF>

Οι τριπλέτες είναι :



Και το γράφημα :



**Ερώτημα 2**

Παρακάτω παρουσιάζεται ο κώδικας του .rdf αρχείου που αναπαριστά την πρόταση φυσικής γλώσσας :

*Η google αναφέρει ότι το Τμήμα Η/Υ & Πληροφορικής βρίσκεται στο Ρίο.*

<?**xml** version="1.0" encoding = "UTF-8" ?>

<rdf:RDF

xmlns:rdf="http:**//**www.w3.org/1999/02/22-rdf-syntax-ns**#**"

xmlns:uni="http:**//**www.mydomain.org/uni-ns/">

<rdf:Description rdf:about="http://mydomain/Statement1">

<rdf:subject rdf:resource="Τμήμα Η/Υ και Πληροφορικής"/>

<rdf:predicate rdf:resource="βρίσκεται"/>

<rdf:object rdf:resource="Ρίο"/>

</rdf:Description>

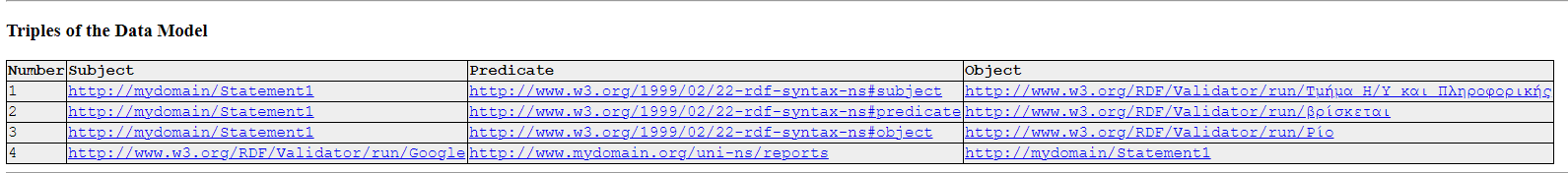
<rdf:Description rdf:about="Google">

<uni:reports rdf:resource="http://mydomain/Statement1"/>

</rdf:Description>

</rdf:RDF>

Οι τριπλέτες είναι :



Και το γράφημα :



**Ερώτημα 3**

Ξεκινάω την δημιουργία της RDFS Οντολογίας εισάγοντας τα απαραίτητα namespaces και έπειτα δημιουργώ τις κλάσεις όπως φαίνονται παρακάτω:

<rdf:RDF xmlns:rdf="http:**//**www.w3.org/1999/02/22-rdf-syntax-ns**#**"

xmlns:rdfs="http:**//**www.w3.org/2000/01/rdf-schema**#**"

xmlns:univ="http:**//**www.mydomain.org/univ/">

<rdfs:Class rdf:about="http://www.mydomain.org/univ/Person">

</rdfs:Class>

<rdfs:Class rdf:about="http://www.mydomain.org/univ/Professor">

<rdfs:subClassOf rdf:resource="http://www.mydomain.org/univ/Person"/>

</rdfs:Class>

<rdfs:Class rdf:about="http://www.mydomain.org/univ/Student">

<rdfs:subClassOf rdf:resource="http://www.mydomain.org/univ/Person"/>

</rdfs:Class>

<rdfs:Class rdf:about="http://www.mydomain.org/univ/Department">

</rdfs:Class>

<rdfs:Class rdf:about="http://www.mydomain.org/univ/Classroom">

</rdfs:Class>

<rdfs:Class rdf:about="http://www.mydomain.org/univ/Lesson">

</rdfs:Class>

Όπως φαίνεται παραπάνω δημιουργώ τις κλάσεις Person, Professor ,Student, Department, Classroom, Lesson χρησιμοποιώντας το ***<rdfs:Class>*** tag. Έπειτα στις κλάσεις Professor και Student έχω εισάγει το tag ***<rdfs:subClassOf>*** και εισάγω σαν πόρο την Person το οποίο σημαίνει ότι η Professor και Student είναι υπό-κλάσεις της κλάσης Person . Αφού δήλωσα τις κλάσεις σειρά έχουν οι ιδιότητες αυτών οι οποίες παραθέτονται παρακάτω .

<rdfs:Property rdf:about="http://www.mydomain.org/univ/has\_name">

<rdfs:domain rdf:resource="http://www.mydomain.org/univ/Person"/>

<rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal"/>

</rdfs:Property>

<rdfs:Property rdf:about="http://www.mydomain.org/univ/has\_phone">

<rdfs:domain rdf:resource="http://www.mydomain.org/univ/Person"/>

<rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal"/>

</rdfs:Property>

<rdfs:Property rdf:about="http://www.mydomain.org/univ/has\_email">

<rdfs:domain rdf:resource="http://www.mydomain.org/univ/Person"/>

<rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal"/>

</rdfs:Property>

<rdfs:Property rdf:about="http://www.mydomain.org/univ/has\_age">

<rdfs:domain rdf:resource="http://www.mydomain.org/univ/Person"/>

<rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Integer"/>

</rdfs:Property>

<rdfs:Property rdf:about="http://www.mydomain.org/univ/member\_of">

<rdfs:domain rdf:resource="http://www.mydomain.org/univ/Person"/>

<rdfs:range rdf:resource="http://www.mydomain.org/univ/Department"/>

</rdfs:Property>

<rdfs:Property rdf:about="http://www.mydomain.org/univ/teaches">

<rdfs:domain rdf:resource="http://www.mydomain.org/univ/Professor"/>

<rdfs:range rdf:resource="http://www.mydomain.org/univ/Lesson"/>

</rdfs:Property>

<rdfs:Property rdf:about="http://www.mydomain.org/univ/les\_name">

<rdfs:domain rdf:resource="http://www.mydomain.org/univ/Lesson"/>

<rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal"/>

</rdfs:Property>

<rdfs:Property rdf:about="http://www.mydomain.org/univ/taught\_by">

<rdfs:domain rdf:resource="http://www.mydomain.org/univ/Lesson"/>

<rdfs:range rdf:resource="http://www.mydomain.org/univ/Professor"/>

</rdfs:Property>

<rdfs:Property rdf:about="http://www.mydomain.org/univ/dep\_name">

<rdfs:domain rdf:resource="http://www.mydomain.org/univ/Department"/>

<rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal"/>

</rdfs:Property>

<rdfs:Property rdf:about="http://www.mydomain.org/univ/dep\_city">

<rdfs:domain rdf:resource="http://www.mydomain.org/univ/Department"/>

<rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal"/>

</rdfs:Property>

<rdfs:Property rdf:about="http://www.mydomain.org/univ/room\_name">

<rdfs:domain rdf:resource="http://www.mydomain.org/univ/Classroom"/>

<rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal"/>

</rdfs:Property>

<rdfs:Property rdf:about="http://www.mydomain.org/univ/room\_capacity">

<rdfs:domain rdf:resource="http://www.mydomain.org/univ/Classroom"/>

<rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Integer"/>

</rdfs:Property>

<rdfs:Property rdf:about="http://www.mydomain.org/univ/room\_department">

<rdfs:domain rdf:resource="http://www.mydomain.org/univ/Classroom"/>

<rdfs:range rdf:resource="http://www.mydomain.org/univ/Department"/>

</rdfs:Property>

Για να φτιάξουμε τις ιδιότητες που θα έχει μια κλάση χρησιμοποιούμε το tag ***<rdfs:Property rdf:about=…>*** όπου στο rdf:about θα είναι το όνομα της ιδιότητας μαζί με το URI . Έπειτα ορίζουμε τα tags ***<rdfs:domain>*** και ***<rdfs:range>*** όπου στο ***<rdfs:domain>*** ορίζουμε την κλάση στην οποία ανήκει η ιδιότητα και στο ***<rdfs:range>*** ορίζουμε το πεδίο τιμών της ιδιότητας . Συγκεκριμένα για την ιδιότητα has\_name βλέπουμε ότι το domain της έχει την κλάση Person και το range της έχει την τιμή Literal που σημαίνει ότι η ιδιότητα has\_name ανήκει στην κλάση Person και παίρνει αλφαριθμητικές τιμές. Το ίδιο ισχύει και για τις ιδιότητες has\_phone,has\_email και η has\_age ανήκει και αυτή στην κλάση Person μόνο που δέχεται ακέραιους αριθμούς αντί για αλφαριθμητικές τιμές. Στην ιδιότητα member\_of βλέπουμε ότι το domain της έχει την τιμή Person αλλά το range έχει την τιμή Department αυτό σημαίνει ότι η ιδιότητα αυτή ανήκει στην κλάση Person και ότι οι τιμές που μπορεί να πάρει αυτή η ιδιότητα είναι τύπου Department η οποία όπως είδαμε πιο πάνω είναι κλάση .Η ιδιότητα teaches ανήκει στην κλάση Professor και οι τιμές που μπορεί να πάρει είναι τύπου Lesson (κλάση) . Η ιδιότητα les\_name ανήκει στην κλάση Lesson και οι τιμές που μπορεί να πάρει είναι αλφαριθμητικές . Η ιδιότητα taught\_by ανήκει στην κλάση Lesson και οι τιμές που μπορεί να πάρει είναι τύπου Professor (κλάση). Οι ιδιότητες dep\_name, dep\_city ανήκουν στην κλάση Department και οι τιμές που μπορούν να πάρουν είναι αλφαριθμητικές. Η ιδιότητα room\_name ανήκει στην κλάση Classroom και οι τιμές που μπορεί να πάρει είναι αλφαριθμητικές. Η ιδιότητα room\_capacity ανήκει στην κλάση Classroom και οι τιμές που μπορεί να πάρει είναι ακέραιες και η ιδιότητα room\_department ανήκει στην κλάση Classroom και οι τιμές που μπορεί να πάρει είναι τύπου Department (κλάση) .Έπειτα δημιουργούμε στιγμιότυπα για τις παραπάνω κλάσεις τα οποία υπάρχουν στο έγγραφο του ερωτήματος 3.

**Ερώτημα 4**

1. Το SPARQL query το οποίο επιστρέφει τα τηλέφωνα των καθηγητών είναι:

PREFIX univ: <http://www.mydomain.org/univ/>

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX rdfs: <http://www.w3.org/2000/01/22-rdf-schema#>

**SELECT** ?name ?phone

**WHERE**{

?x rdf:**type** univ:Professor.

?x univ:has\_phone ?phone .

?x univ:has\_name ?name.

}

Η μεταβλητή ?x περιέχει μόνο τα στιγμιότυπα τα οποία είναι τύπου

Professor και έπειτα για το αντικείμενο που έχει η μεταβλητή ?x παίρνουμε την τιμή της ιδιότητας **univ:has\_phone** και την τιμή της ιδιότητας **univ:has\_name** και τις αποθηκεύουμε στις μεταβλητές **?name** και **?phone** αντίστοιχα .

Το οποίο επιστρέφει σαν έξοδο:

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Όπου η στήλη name δείχνει το όνομα των καθηγητών και η στήλη phone δείχνει τα τηλεφωνά τους .

1. Το SPARQL query το οποίο επιστρέφει τα τηλέφωνα των μαθητών που έχουν ηλικία πάνω από 23 έτη είναι:

PREFIX univ: <http://www.mydomain.org/univ/>

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX rdfs: <http://www.w3.org/2000/01/22-rdf-schema#>

**SELECT** ?name ?phone

**WHERE**{

?x rdf:**type** univ:Student .

?x univ:has\_phone ?phone .

?x univ:has\_name ?name.

?x univ:has\_age ?age .

**FILTER**(xsd:**int**(?age) > 23)

}

Η μεταβλητή ?x περιέχει μόνο τα στιγμιότυπα τα οποία είναι τύπου

Student και έπειτα για το αντικείμενο που έχει η μεταβλητή ?x παίρνω την τιμή της ιδιότητας **univ:has\_phone** ,την τιμή της ιδιότητας **univ:has\_name** και την τιμή της ιδιότητας **univ:has\_age** και τις αποθηκεύουμε στις μεταβλητές **?name** **?phone** και **?age** αντίστοιχα και έπειτα λόγω του **FILTER** φιλτράρουμε το αποτέλεσμα ανάλογα με το τι τιμή έχει η μεταβλητή ?age. Αν είναι μεγαλύτερη από 23 κρατάω το αποτέλεσμα αλλιώς συνεχίζω με το επόμενο entry.

Το οποίο επιστρέφει σαν έξοδο:

Εικόνα που περιέχει κείμενο, ηλεκτρονικές συσκευές

Περιγραφή που δημιουργήθηκε αυτόματα

Όπου η στήλη name δείχνει το όνομα των μαθητών και η στήλη phone δείχνει τα τηλεφωνά τους .

1. Το SPARQL query το οποίο επιστρέφει το όνομα όλων των ατόμων που είναι μέλη σε τμήμα που είναι στην Πάτρα είναι:

PREFIX univ: <http://www.mydomain.org/univ/>

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX rdfs: <http://www.w3.org/2000/01/22-rdf-schema#>

**SELECT** ?name ?members\_of

**WHERE** {

?x univ:has\_name ?name;

univ:member\_of ?members\_of.

?members\_of univ:dep\_city ?city

**FILTER**(?city="Patras")

}

Η μεταβλητή ?x «δέχεται» τα στιγμιότυπα τα οποία έχουν την ιδιότητα univ:has\_name που ουσιαστικά είναι οι καθηγητές(Professor class) και οι μαθητές (Student class) και αποθηκεύει την τιμή στην μεταβλητή ?name. Έπειτα παίρνω την τιμή της ιδιότητας univ:member\_of την αποθηκεύω στην μεταβλητή ?members\_of και στην συνέχεια χρησιμοποιώ την μεταβλητή ?members\_of η οποία περιέχει το Τμήμα στο οποίο ανήκει ο μαθητής/καθηγητής και παίρνω την τιμή της ιδιότητας univ:dep\_city και την αποθηκεύω στην μεταβλητή ?city.Έπειτα μέσω του FILTER κοιτάω για το αν η μεταβλητή ?city έχει την τιμή Patras αν ναι τότε το αποτέλεσμα καταγράφεται αλλιώς συνεχίζω στα επόμενα entries .

Το οποίο επιστρέφει σαν έξοδο:

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Όπου η στήλη name δείχνει το όνομα των ατόμων και η στήλη members\_of δείχνει τα τμήματα στα οποία ανήκουν τα άτομα αυτά .

1. Το SPARQL query το οποίο επιστρέφει το όνομα όλων των τάξεων των οποίων το τμήμα είναι στην Πάτρα και έχουν χωρητικότητα μεγαλύτερη από 150 είναι:

PREFIX univ: <http://www.mydomain.org/univ/>

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX rdfs: <http://www.w3.org/2000/01/22-rdf-schema#>

**SELECT** ?classroom\_name

**WHERE** {

?x rdf:**type** univ:Classroom.

?x univ:room\_name ?classroom\_name;

univ:room\_capacity ?cap

**filter**(xsd:**int**(?cap)>150)

?x univ:room\_department ?dep.

?dep univ:dep\_city ?city

**FILTER**(?city="Patras") }

Το οποίο επιστρέφει σαν έξοδο:

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Όπου η στήλη classroom\_name δείχνει το όνομα των τάξεων που είναι σε τμήμα το οποίο είναι στην Πάτρα και έχουν χωρητικότητα μεγαλύτερη του 150 .

**Ερώτημα 5**

**Α.** Για το **Α.** ερώτημα δημιουργώ για αρχή έναν file chooser frame για να μπορώ να διαλέξω το αρχείο. Έπειτα διαβάζω το αρχείο και δημιουργείται το μοντέλο που αναπαριστά το .rdf αρχείο που διαβάστηκε, στην συνέχεια δημιουργώ ένα SPARQL query σε μια String μεταβλητή και έπειτα μέσω της ***Query query = QueryFactory.create(queryString);*** Δημιουργώ ένα **Query object** το οποίο εκτελώ με βάση το model που έχει δημιουργηθεί διαβάζοντας το .rdf αρχείο προηγουμένως . Μετά για όλα τα entries που επιστρέφει το query(for-loop) παίρνω τον rdf node που αντιστοιχεί στην τιμή που έχει η μεταβλητή ?x του SPARQL ερωτήματος και γράφω αυτή την τιμή στον πίνακα dataTable του GUI και ενεργοποιώ την λειτουργία δημιουργίας νεών εγγραφών καθώς και την λειτουργία εμφάνισης των Statements για ένα συγκεκριμένο URI . Η συνάρτηση που επιτελεί τα παραπάνω είναι η:

**private** void open\_btnActionPerformed(java.awt.event.ActionEvent evt) {

**FileFilter** filter = **new** FileNameExtensionFilter("RDF file", "rdf");

jFileChooser.addChoosableFileFilter(filter);

jFileChooser.setFileFilter(filter);

DefaultTableModel tableModel =(DefaultTableModel) dataTable.getModel();

int returnVal = jFileChooser.showOpenDialog(**this**);

**if** (returnVal == JFileChooser.APPROVE\_OPTION) {

file = jFileChooser.getSelectedFile();

**String** path = file.getAbsolutePath();

path\_label.setText(file.getName());

**InputStream** in = FileManager.get().open(path);

**InputStreamReader** rin = **new** **InputStreamReader**(in,**Charset**.forName("UTF-8").newDecoder());

model = ModelFactory.createDefaultModel();

model.read(rin,"");

// model.write(System.out);

**String** queryString = "PREFIX univ: <http://www.mydomain.org/univ/> PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX rdfs: <http://www.w3.org/2000/01/22-rdf-schema#> SELECT ?x {?x rdf:type univ:Department . }";

Query query = QueryFactory.create(queryString);

query.serialize(**new** IndentedWriter(**System**.out,**true**));

QueryExecution qexec = QueryExecutionFactory.create(query,model);

**ResultSet** rs = qexec.execSelect();

**for** (; rs.hasNext();){

QuerySolution rb = rs.nextSolution();

RDFNode x = rb.get("x");

**System**.out.println(x.toString());

tableModel.addRow(**new** **Object**[]{x});

}

**URI** = "http://www.mydomain.org/univ/";

}

dataCategory.setEnabled(**true**);

stmt\_btn.setEnabled(**true**);

}

Τώρα στην συνέχεια του **Α.** ερωτήματος, όταν ο χρήστης επιλέξει ένα από τα τμήματα που αναγράφονται τότε θα εκτελεστεί η μέθοδος ***dataTableMouseClicked*** η οποία θα τρέξει μόνο όταν γίνει ένα click event στον πίνακα dataTable .Όταν ξεκινήσει η εκτέλεση αυτής της μεθόδου ξανά διαβάζει το αρχείο ώστε να πάρει τις τελευταίες αλλαγές, αν έχουν γίνει στο έγγραφο και αμέσως μετά παίρνει τον αριθμό της γραμμής του πίνακα που επέλεξε ο χρήστης και με βάση τον αριθμό της γραμμής παίρνει την τιμή που έχει ο πίνακας σε εκείνη την γραμμή και την αποθηκεύει στην μεταβλητήdepartment. ‘Επειτα δημιουργώ και ένα instance της κλάσης AllData που είναι η κλάση του παραθύρου που θα εμφανιστούν τα τελικά δεδομένα , στην συνέχεια εκτελούνται τα απαραίτητα SPARQL ερωτήματααπό τα οποία παίρνουμε το όνομα του τμήματος που επέλεξε ο χρήστης καθώς και όλους τους Καθηγητές, Μαθητές, Μαθήματα και Τάξεις που έχει το συγκεκριμένο τμήμα και τα εισάγει στους πίνακες professorsTable , studentsTable, lessonsTable και classesTable αντίστοιχα μέσω της μεθόδου writeToTable η οποία ανήκει στην κλάση AllData που είναι η κλάση του παραθύρου που περιέχει τους παραπάνω πίνακες .Η μέθοδος writeToTable δέχεται 4 παραμέτρους τύπου String από τις οποίες η πρώτη αφορά το σε ποιον πίνακα θα γράψει τις υπόλοιπες τρείς παραμέτρους που είναι τα δεδομένα που ανακτώνται από τα SPARQL ερωτήματα .

Η συνάρτηση ***dataTableMouseClicked*** βρίσκεται στην κλάση ***Ask2Frame*** και ο κώδικας της φαίνεται παρακάτω :

**private** void dataTableMouseClicked(java.awt.event.MouseEvent evt) {

**InputStream** in = FileManager.get().open(file.getAbsolutePath());

**InputStreamReader** rin = **new** **InputStreamReader**(in,**Charset**.forName("UTF-8").newDecoder());

model.read(rin,"");

int row = dataTable.getSelectedRow();

int col = 0;

**Object** department = dataTable.getValueAt(row,col);

**String** str\_dep = department.toString();

**System**.out.printf("Dep! "+str\_dep);

**System**.out.printf("Clicked! %d \n",dataTable.getSelectedRow());

// dataWindow.setVisible(true);

Literal title=**null**,

name = **null**,

age=**null**,

phone=**null**,

cap=**null**,

taught\_by=**null**;

AllData dataWindow = **new** AllData();

**String** queryString = "PREFIX univ: <http://www.mydomain.org/univ/>"+

"PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>"+

"PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>" +

"PREFIX rdfs: <http://www.w3.org/2000/01/22-rdf-schema#>"+

"SELECT ?name WHERE {?x rdf:type univ:Department ; univ:dep\_name ?name . "+

"FILTER(?x=<"+str\_dep+">)"+

"}";

Query query = QueryFactory.create(queryString);

query.serialize(**new** IndentedWriter(**System**.out, **true**));

QueryExecution qexec = QueryExecutionFactory.create(query, model);

**ResultSet** rs = qexec.execSelect();

**if**(rs.hasNext()){

**for** (; rs.hasNext();){

QuerySolution rb = rs.nextSolution();

title = rb.getLiteral("name");

**System**.out.printf("Dep! " + title.toString());

}

qexec.close();

dataWindow.setTitle("Τμήμα "+title.toString());

}**else**{

JOptionPane.showMessageDialog(**this**, "Το Τμήμα που επέλεξες δέν έχει δεδομένα.");

**return**;

}

**String** ProfQuerystr = "PREFIX univ: <http://www.mydomain.org/univ/>"

+ "PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>"

+ "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>"

+ "PREFIX rdfs: <http://www.w3.org/2000/01/22-rdf-schema#>"

+ "SELECT ?name ?age ?phone WHERE { ?x rdf:type univ:Professor . ?x univ:has\_name ?name . ?x univ:has\_age ?age . ?x univ:has\_phone ?phone ."

+ "?x univ:member\_of ?dep ."

+ "FILTER(?dep=<"+str\_dep+">)}";

Query prof\_query = QueryFactory.create(ProfQuerystr);

prof\_query.serialize(**new** IndentedWriter(**System**.out, **true**));

QueryExecution p\_qexec = QueryExecutionFactory.create(prof\_query, model);

**ResultSet** p\_rs = p\_qexec.execSelect();

**for** (; p\_rs.hasNext();) {

QuerySolution p\_rb = p\_rs.nextSolution();

name = p\_rb.getLiteral("name");

age = p\_rb.getLiteral("age");

phone = p\_rb.getLiteral("phone");

**System**.out.printf("Dep! " + name.toString());

dataWindow.writeToTable("professor",name.toString(),age.toString(),phone.toString());

}

p\_qexec.close();

**String** StudQuerystr = "PREFIX univ: <http://www.mydomain.org/univ/>"

+ "PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>"

+ "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>"

+ "PREFIX rdfs: <http://www.w3.org/2000/01/22-rdf-schema#>"

+ "SELECT ?name ?age ?phone WHERE { ?x rdf:type univ:Student . ?x univ:has\_name ?name . ?x univ:has\_age ?age . ?x univ:has\_phone ?phone ."

+ "?x univ:member\_of ?dep ."

+ "FILTER(?dep=<" + str\_dep + ">)}";

Query stud\_query = QueryFactory.create(StudQuerystr);

stud\_query.serialize(**new** IndentedWriter(**System**.out, **true**));

QueryExecution s\_qexec = QueryExecutionFactory.create(stud\_query, model);

**ResultSet** s\_rs = s\_qexec.execSelect();

**for** (; s\_rs.hasNext();) {

QuerySolution s\_rb = s\_rs.nextSolution();

name = s\_rb.getLiteral("name");

age = s\_rb.getLiteral("age");

phone = s\_rb.getLiteral("phone");

**System**.out.printf("Dep! " + name.toString());

dataWindow.writeToTable("student", name.toString(), age.toString(), phone.toString());

}

s\_qexec.close();

**String** ClassQuerystr = "PREFIX univ: <http://www.mydomain.org/univ/>"

+ "PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>"

+ "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>"

+ "PREFIX rdfs: <http://www.w3.org/2000/01/22-rdf-schema#>"

+ "SELECT ?name ?capacity WHERE { ?x rdf:type univ:Classroom . ?x univ:room\_name ?name . ?x univ:room\_capacity ?capacity ."

+ "?x univ:room\_department ?dep ."

+ "FILTER(?dep=<" + str\_dep + ">)}";

Query class\_query = QueryFactory.create(ClassQuerystr);

class\_query.serialize(**new** IndentedWriter(**System**.out, **true**));

QueryExecution c\_qexec = QueryExecutionFactory.create(class\_query, model);

**ResultSet** c\_rs = c\_qexec.execSelect();

**for** (; c\_rs.hasNext();) {

QuerySolution c\_rb = c\_rs.nextSolution();

name = c\_rb.getLiteral("name");

cap = c\_rb.getLiteral("capacity");

**System**.out.printf("Dep! " + name.toString());

dataWindow.writeToTable("classroom", name.toString(), cap.toString(),"");

}

c\_qexec.close();

**String** LessonQuerystr = "PREFIX univ: <http://www.mydomain.org/univ/>"

+ "PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>"

+ "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>"

+ "PREFIX rdfs: <http://www.w3.org/2000/01/22-rdf-schema#>"

+ "SELECT ?name ?teacher WHERE { ?prof rdf:type univ:Professor . ?prof univ:teaches ?y . ?y univ:les\_name ?name . ?prof univ:has\_name ?teacher ."

+ "?prof univ:member\_of ?dep ."

+ "FILTER(?dep=<" + str\_dep + ">)}";

Query lesson\_query = QueryFactory.create(LessonQuerystr);

lesson\_query.serialize(**new** IndentedWriter(**System**.out, **true**));

QueryExecution l\_qexec = QueryExecutionFactory.create(lesson\_query, model);

**ResultSet** l\_rs = l\_qexec.execSelect();

**for** (; l\_rs.hasNext();) {

QuerySolution l\_rb = l\_rs.nextSolution();

name = l\_rb.getLiteral("name");

taught\_by = l\_rb.getLiteral("teacher");

**System**.out.printf("Dep! " + name.toString());

dataWindow.writeToTable("lessons", name.toString(), taught\_by.toString(),"");

}

l\_qexec.close();

dataWindow.setVisible(**true**);

}

Η συνάρτηση ***writeToTable*** βρίσκεται στην κλάση ***AllData*** και ο κώδικας της φαίνεται παρακάτω :

**public** void writeToTable(String table,String v1,String v2,String v3){

**if**(table=="professor"){

DefaultTableModel tableModel =(DefaultTableModel) professorsTable.getModel();

tableModel.addRow(**new** **Object**[]{v1,v2,v3});

}**else** **if**(table=="student"){

DefaultTableModel tableModel =(DefaultTableModel) studentsTable.getModel();

tableModel.addRow(**new** **Object**[]{v1,v2,v3});

}**else** **if**(table=="classroom"){

DefaultTableModel tableModel =(DefaultTableModel) classesTable.getModel();

tableModel.addRow(**new** **Object**[]{v1,v2});

}**else** **if**(table=="lessons"){

DefaultTableModel tableModel =(DefaultTableModel) lessonsTable.getModel();

tableModel.addRow(**new** **Object**[]{v1,v2});

}

}

Παρακάτω παρουσιάζονται screenshots από την εκτέλεση του Α. ερωτήματος της εφαρμογής .

Όταν έχω διαλέξει το αρχείο το οποίο είναι το ερώτημα3.rdf σύμφωνα με την εκφώνηση.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Όταν διαλέξω το τμήμα D1 :

Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

**Β.** Για το **Β.** ερώτημα δημιουργώ ένα combo Box στο παράθυρο της κλάσης Ask2Frame το οποίο περιέχει τις κατηγορίες Professor, Student , Department, Lesson, Classroom. Όταν ο χρήστης επιλέξει ένα από αυτά τότε ανοίγει το αντίστοιχο παράθυρο, ο χρήστης γράφει τα δεδομένα στα πεδία των ιδιοτήτων της κατηγορίας που διάλεξε και έπειτα πατώντας το κουμπί Add προσθέτει το instance στο .rdf αρχείο .Η λειτουργία για το πιο παράθυρο θα ανοίξει με βάση το τι επέλεξε ο χρήστης υλοποιεί η μέθοδος ***dataCategoryActionPerformed*** η οποία όταν ο χρήστης πατήσει click σε κάποια επιλογή η μέθοδος παίρνει την επιλογή του χρήστη και δημιουργεί ένα instance του παραθύρου που αντιστοιχεί σε μία από τις παραπάνω κατηγορίες .Έπειτα αφού δημιουργηθεί το αντικείμενο καλείται η μέθοδος ***getModel ή getModelTable*** στην περίπτωση που ο χρήστης θέλει να εισάγει νέο τμήμα στο .rdf αρχείοόπου αυτές η μέθοδοι παίρνουν το μονοπάτι του αρχείου, το URI και το model που δημιουργείται από το διάβασμα του .rdf αρχείου (και το model του πινακα dataTable στην περίπτωση του Department ) και τα αποθηκεύουν στις μεταβλητές των αντίστοιχων κλάσεων . Οι μέθοδοι που κάνουν τα παραπάνω είναι οι:

***dataCategoryActionPerformed*** η οποία είναι στην κλάση ***Ask2Frame*** και οι μεθόδοι ***getModel ή getModelTable*** βρίσκονται στις κλάσεις ***ProfessorData,StudentData,LessonData,ClassroomData και DepartmentData*** αντίστοιχα.

Ο κώδικας τους παρουσιάζεται παρακάτω :

**private** void dataCategoryActionPerformed(java.awt.event.ActionEvent evt) {

**String** ch = dataCategory.getSelectedItem().toString();

**System**.out.println("DataChoses:"+ch);

**if**(ch.equals("Professor")){

ProfessorData pd = **new** ProfessorData();

pd.getModel(model,**URI**,file.getAbsolutePath());

pd.setVisible(**true**);

}**else** **if**(ch.equals("Lesson")){

LessonData ld = **new** LessonData();

ld.getModel(model,**URI**,file.getAbsolutePath());

ld.setVisible(**true**);

}**else** **if**(ch.equals("Student")){

StudentData sd = **new** StudentData();

sd.getModel(model,**URI**,file.getAbsolutePath());

sd.setVisible(**true**);

}**else** **if**(ch.equals("Department")){

DepartmentData dd = **new** DepartmentData();

DefaultTableModel tableModel =(DefaultTableModel) dataTable.getModel();

dd.getModelTable(model,**URI**,file.getAbsolutePath(),tableModel);

dd.setVisible(**true**);

}**else** **if**(ch.equals("Classroom")){

ClassroomData cd = **new** ClassroomData();

cd.getModel(model,**URI**,file.getAbsolutePath());

cd.setVisible(**true**);

}

}

**public** void getModel(Model m,String uri\_var,String f){

model = m;

uri = uri\_var;

file = f;

}

**public** void getModelTable(Model m,String uri\_var,String f,DefaultTableModel tmodel){

model = m;

uri = uri\_var;

file = f;

tableModel = tmodel;

}

Όταν ο χρήστης πατήσει το κουμπί Add σε ένα από τα παράθυρα που αναφέρθηκαν πιο πάνω τότε εκτελείται η μέθοδος ***addToRdfActionPerformed*** η οποία παίρνει τα δεδομένα που εισήγαγε ο χρήστης και έπειτα ελέγχει για το αν κάποιο πεδίο είναι κενό. Αν ναι τότε εμφανίζεται κατάλληλο μήνυμα, αλλιώς δημιουργώ ένα Resource object το οποίο περιέχει την τιμή URI και το όνομα του αντικειμένου της κατηγορίας που θέλει να εισάγει στο αρχείο. Έπειτα δημιουργώ τόσα Property objects όσα και οι ιδιότητες που έχει η κάθε κατηγορία και τα «συνδέω» με το Resource object που δημιούργησα πιο πάνω και σε αυτές τις ιδιότητες βάζω τις τιμές που έχει εισάγει ο χρήστης και έπειτα αφού το Resource έχει προστεθεί στο model κατά την δημιουργία του , γράφω το model στο αρχείο που επέλεξε αρχικά ο χρήστης . Στην περίπτωση που ο χρήστης εισάγει ένα νέο τμήμα η μέθοδος αφού γράψει τα νέα δεδομένα στο αρχείο, θα το ξανά διαβάσει και θα εκτελέσει ένα SPARQL ερώτημα για να πάρει μόνο το τμήμα που προστέθηκε και να το εισάγει στον πίνακα dataTable με αυτό τον τρόπο ενημερώνεται ο πίνακας των τμημάτων μετά από κάθε προσθήκη τμήματος .

Η μέθοδος ***addToRdfActionPerformed*** βρίσκεταιστις κλάσεις ***ProfessorData,StudentData,LessonData,ClassroomData και DepartmentData*** αντίστοιχα .

Οι κώδικες παρουσιάζονται παρακάτω :

Στην κλάση ***ProfessorData***:

**private** void addToRdfActionPerformed(java.awt.event.ActionEvent evt) {

**String** name = **this**.name\_field.getText();

**String** age = **this**.age\_field.getText();

**String** email = **this**.email\_field.getText();

**String** phone = **this**.phone\_field.getText();

**String** lesson\_str = **this**.less\_field.getText();

**String** department\_str = **this**.dep\_field.getText();

**if**(!"".equals(name) && !"".equals(age) && !"".equals(email) && !"".equals(phone) && !"".equals(lesson\_str) && !"".equals(department\_str)){

Resource professor = model.createResource(uri+name);

Property p1 = model.createProperty(uri,"has\_name");

Property p2 = model.createProperty(uri,"has\_phone");

Property p3 = model.createProperty(uri,"has\_email");

Property p4 = model.createProperty(uri,"has\_age");

Property p5 = model.createProperty(uri,"teaches");

Property p6 = model.createProperty(uri,"member\_of");

professor.addProperty(RDF.type,model.createResource(uri+"Professor"));

professor.addProperty(p1,name);

professor.addProperty(p2,phone);

professor.addProperty(p3,email);

professor.addProperty(p4,age);

professor.addProperty(p5,model.createResource(uri+lesson\_str));

professor.addProperty(p6,model.createResource(uri+department\_str));

**FileOutputStream** out = **null**;

**OutputStreamWriter** sw = **null**;

**try** {

out = **new** **FileOutputStream**(file,**false**);

sw = **new** **OutputStreamWriter**(out,**Charset**.forName("UTF-8").newEncoder());

} **catch** (**FileNotFoundException** ex) {

**Logger**.getLogger(ProfessorData.class.getName()).log(**Level**.SEVERE, **null**, ex);

}

RDFDataMgr.write(sw, model, Lang.RDFXML);

}**else**{

JOptionPane.showMessageDialog(**this**,"Κάποιο απο τα πεδία δέν έχει τιμή.");

}

}

Στην κλάση ***StudentData***:

**private** void addToRdfActionPerformed(java.awt.event.ActionEvent evt) {

**String** name = **this**.stud\_name.getText();

**String** age = **this**.stud\_age.getText();

**String** email = **this**.stud\_email.getText();

**String** phone = **this**.stud\_phone.getText();

**String** department\_str = **this**.stud\_dep.getText();

**if** (!"".equals(name) && !"".equals(age) && !"".equals(email) && !"".equals(phone) && !"".equals(department\_str)) {

Resource student = model.createResource(uri + name);

Property p1 = model.createProperty(uri, "has\_name");

Property p2 = model.createProperty(uri, "has\_phone");

Property p3 = model.createProperty(uri, "has\_email");

Property p4 = model.createProperty(uri, "has\_age");

Property p5 = model.createProperty(uri, "member\_of");

student.addProperty(RDF.type, model.createResource(uri + "Student"));

student.addProperty(p1, name);

student.addProperty(p2, phone);

student.addProperty(p3, email);

student.addProperty(p4, age);

student.addProperty(p5, model.createResource(uri + department\_str));

**FileOutputStream** out = **null**;

**OutputStreamWriter** sw = **null**;

**try** {

out = **new** **FileOutputStream**(file,**false**);

sw = **new** **OutputStreamWriter**(out,**Charset**.forName("UTF-8").newEncoder());

} **catch** (**FileNotFoundException** ex) {

**Logger**.getLogger(ProfessorData.class.getName()).log(**Level**.SEVERE, **null**, ex);

}

RDFDataMgr.write(sw, model, Lang.RDFXML);

} **else** {

JOptionPane.showMessageDialog(**this**, "Κάποιο απο τα πεδία δέν έχει τιμή.");

}

}

Στην κλάση ***LessonData***:

**private** void addToRdfActionPerformed(java.awt.event.ActionEvent evt) {

**String** name = **this**.less\_name.getText();

**String** teacher = **this**.taught\_by.getText();

**if**(!"".equals(name) && !"".equals(teacher)){

Resource lesson = model.createResource(uri+name);

Property p1 = model.createProperty(uri,"les\_name");

Property p2 = model.createProperty(uri,"taught\_by");

lesson.addProperty(RDF.type,model.createResource(uri+"Lesson"));

lesson.addProperty(p1,name);

lesson.addProperty(p2,teacher);

**FileOutputStream** out = **null**;

**OutputStreamWriter** sw = **null**;

**try** {

out = **new** **FileOutputStream**(file,**false**);

sw = **new** **OutputStreamWriter**(out,**Charset**.forName("UTF-8").newEncoder());

} **catch** (**FileNotFoundException** ex) {

**Logger**.getLogger(ProfessorData.class.getName()).log(**Level**.SEVERE, **null**, ex);

}

RDFDataMgr.write(sw, model, Lang.RDFXML);

}**else**{

JOptionPane.showMessageDialog(**this**,"Κάποιο απο τα πεδία δέν έχει τιμή.");

}

}

Στην κλάση ***ClassroomData***:

**private** void addToRdfActionPerformed(java.awt.event.ActionEvent evt) {

**String** name = **this**.class\_name.getText();

**String** cap = **this**.class\_cap.getText();

**String** department\_str = **this**.class\_dep.getText();

**if** (!"".equals(name) && !"".equals(cap) && !"".equals(department\_str)) {

Resource classroom = model.createResource(uri + name);

Property p1 = model.createProperty(uri, "room\_name");

Property p2 = model.createProperty(uri, "room\_capacity");

Property p3 = model.createProperty(uri, "room\_department");

classroom .addProperty(RDF.type, model.createResource(uri + "Classroom"));

classroom .addProperty(p1, name);

classroom .addProperty(p2, cap);

classroom .addProperty(p3, model.createResource(uri + department\_str));

**FileOutputStream** out = **null**;

**OutputStreamWriter** sw = **null**;

**try** {

out = **new** **FileOutputStream**(file,**false**);

sw = **new** **OutputStreamWriter**(out,**Charset**.forName("UTF-8").newEncoder());

} **catch** (**FileNotFoundException** ex) {

**Logger**.getLogger(ProfessorData.class.getName()).log(**Level**.SEVERE, **null**, ex);

}

RDFDataMgr.write(sw, model, Lang.RDFXML);

} **else** {

JOptionPane.showMessageDialog(**this**, "Κάποιο απο τα πεδία δέν έχει τιμή.");

}

}

Στην κλάση ***DepartmentData***:

**private** void addToRdfActionPerformed(java.awt.event.ActionEvent evt) {

**String** name = **this**.dep\_name.getText();

**String** city = **this**.dep\_city.getText();

**if** (!"".equals(name) && !"".equals(city)) {

Resource department = model.createResource(uri + name);

Property p1 = model.createProperty(uri, "dep\_name");

Property p2 = model.createProperty(uri, "dep\_city");

department.addProperty(RDF.type, model.createResource(uri + "Department"));

department.addProperty(p1, name);

department.addProperty(p2, city);

**FileOutputStream** out = **null**;

**OutputStreamWriter** sw = **null**;

**try** {

out = **new** **FileOutputStream**(file,**false**);

sw = **new** **OutputStreamWriter**(out,**Charset**.forName("UTF-8").newEncoder());

} **catch** (**FileNotFoundException** ex) {

**Logger**.getLogger(ProfessorData.class.getName()).log(**Level**.SEVERE, **null**, ex);

}

RDFDataMgr.write(sw, model, Lang.RDFXML);

} **else** {

JOptionPane.showMessageDialog(**this**, "Κάποιο απο τα πεδία δέν έχει τιμή.");

}

**InputStream** in = FileManager.get().open(file);

**InputStreamReader** rin = **new** **InputStreamReader**(in,**Charset**.forName("UTF-8").newDecoder());

model.read(rin,"");

**String** queryString = "PREFIX univ: <http://www.mydomain.org/univ/> PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX rdfs: <http://www.w3.org/2000/01/22-rdf-schema#> SELECT ?x {?x rdf:type univ:Department .FILTER(?x=<" + uri + name + ">)}";

Query query = QueryFactory.create(queryString);

query.serialize(**new** IndentedWriter(**System**.out,**true**));

QueryExecution qexec = QueryExecutionFactory.create(query,model);

**ResultSet** rs = qexec.execSelect();

rs.hasNext();

QuerySolution rb = rs.nextSolution();

RDFNode x = rb.get("x");

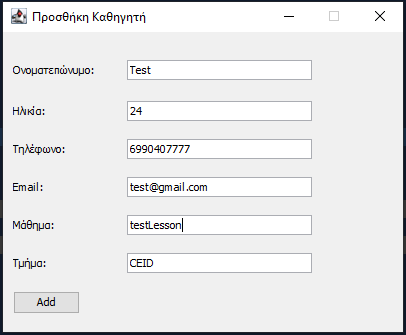
**System**.out.println(x.toString());

tableModel.addRow(**new** **Object**[]{x});

}

Παρακάτω παρουσιάζονται screenshots από την εκτέλεση του Β. ερωτήματος της εφαρμογής .

Προσθήκη Καθηγητή:



Το entry στο .rdf αρχείο :

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Προσθήκη Μαθητή:

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Το entry στο .rdf αρχείο :

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Προσθήκη Τμήματος:

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Το entry στο .rdf αρχείο :

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Προσθήκη Μαθήματος:

Εικόνα που περιέχει κείμενο

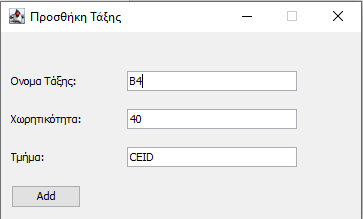
Περιγραφή που δημιουργήθηκε αυτόματα

Το entry στο .rdf αρχείο :

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Προσθήκη Τάξης:



Το entry στο .rdf αρχείο :

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Αν επιλέξουμε έπειτα το τμήμα CEID από τον πίνακα των τμημάτων θα δούμε ότι :

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Από το οποίο βλέπουμε ότι όλα τα δεδομένα που εισήγαγε ο χρήστης πιο πάνω εγγράφηκαν σωστά και όλα όσα υπάρχουν στους πίνακες αφορούν το τμήμα CEID .

**Γ.** Για το **Γ.** ερώτημα δημιούργησα ένα textField στο οποίο ο χρήστης θα μπορεί να εισάγει το URI του resource που επιθυμεί και πατώντας το κουμπί Show Statements θα μπορεί να δει όλες τις τριπλέτες που υπάρχουν για αυτό στον γράφο . Πατώντας το παραπάνω κουμπί θα εκτελεστεί η μέθοδος ***stmt\_btnActionPerformed*** η οποία παίρνει το κείμενο που έχει εισάγει ο χρήστης στο παραπάνω textField και έπειτα κοιτάει για το αν υπάρχει το URI μέσα στο κείμενο που έχει εισάγει ο χρήστης. Αν το κείμενο είναι κενό ή δεν περιέχει το URI τότε εμφανίζεται κατάλληλο προειδοποιητικό μήνυμα αλλιώς δημιουργώ ένα μοντέλο συμπερασμού με βάση το μοντέλο που δημιουργείται από το διάβασμα του .rdf αρχείου και μετά παίρνω όλους τους πόρους που περιέχει ο πόρος τον οποίον δήλωσα στο textField και για κάθε statement που περιέχει ο πόρος παίρνω τα ***Subject, Predicate και Object*** και τα γράφω στον πίνακα που υπάρχει στο παράθυρο τύπου StatementData μέσω της μεθόδου ***addToTable*** το οποίο παράθυρο το δημιουργώ αφού έχω εισάγει σωστά δεδομένα στο textField .

Ο κώδικας της μεθόδου ***stmt\_btnActionPerformed*** βρίσκεται στην κλάση ***Ask2Frame*** καιπαρουσιάζεται παρακάτω :

**private** void stmt\_btnActionPerformed(java.awt.event.ActionEvent evt) {

**String** text = uri\_value.getText();

**Boolean** contains = text.contains(**URI**);

**if**(text.equals("") || !contains ){

JOptionPane.showMessageDialog(**this**, "Λάθος URI.");

}**else**{

StatementData stmts = **new** StatementData();

InfModel inf\_model = ModelFactory.createRDFSModel(model);

Resource infr = inf\_model.getResource(text);

StmtIterator iter = infr.listProperties();

**while**(iter.hasNext()){

**Statement** stmt = iter.nextStatement();

**System**.out.print(" " + stmt.getSubject().toString());

**System**.out.print(" " + stmt.getPredicate().toString());

**System**.out.print(" " + stmt.getObject().toString());

**System**.out.print("\n");

stmts.addToTable(stmt.getSubject().toString(), stmt.getPredicate().toString(), stmt.getObject().toString());

}

stmts.setVisible(**true**);

}

}

Ο κώδικας της μεθόδου ***addToTable*** βρίσκεται στην κλάση ***StatementData*** καιπαρουσιάζεται παρακάτω :

**public** void addToTable(String subject, String predicate,String object){

DefaultTableModel tableModel =(DefaultTableModel) stmtTable.getModel();

tableModel.addRow(**new** **Object**[]{subject,predicate,object});

}

Παρακάτω παρουσιάζονται screenshots από την εκτέλεση του Γ. ερωτήματος της εφαρμογής .

Για URI: <http://www.mydomain.org/univ/P1>

Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

Οπού βλέπουμε ότι ο καθηγητής P1 εκτός από Professor είναι και Person .

Και για ένα μάθημα για παράδειγμα το <http://www.mydomain.org/univ/L1> έχουμε :

Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

Όπου και εδώ βλέπουμε ότι βλέπει ότι το <http://www.mydomain.org/univ/L1> είναι τύπου Lesson παρόλο που κάτι τέτοιο δεν έχει δηλωθεί στο .rdf αρχείο .