

ΑΝΑΠΑΡΑΣΤΑΣΗ ΓΝΩΣΗΣ ΣΤΟΝ ΠΑΓΚΟΣΜΙΟ ΙΣΤΟ

Project 2022-2023

ΔΕΚΕΜΒΡΙΟΣ 2022 - ΙΑΝΟΥΑΡΙΟΣ 2023

Μάριος Στεφανίδης
1067458

Αικατερίνα
Μητροπούλου
1067409



Πίνακας Περιεχομένων

Ερώτημα 1	03
Ερώτημα 3	10
Ερώτημα 4	14
Ερώτημα 5	18
Ερώτημα 6	20
Ερώτημα 7	21



ΕΡΩΤΗΜΑ 1

A. Το γνωστικό επίπεδο της οντολογίας που επιλέχθηκε για τη συγκεκριμένη εργασία, αφορά στην παρουσίαση της δομής ενός νοσοκομείου και των λοιπών στοιχείων που το απαρτίζουν (παραδείγματος χάριν φάρμακα, κλινικές).

B.

- Η οντολογία θα καλύψει τις βασικές κλάσεις ενός νοσοκομείου
- Η οντολογία μπορεί να χρησιμοποιηθεί προκειμένου να παρουσιάσει τις βασικές κλάσεις και τα στοιχεία που χαρακτηρίζουν τους υπαλλήλους ενός νοσοκομείου
- Η οντολογία θα μπορούσε να παρέχει απαντήσεις σε ερωτήσεις επεξήγησης μίας κλάσης, αναζήτηση δυνατών συνδυασμών κλάσεων και παρουσίαση γενικών χαρακτηριστικών που διαθέτει ένας υπάλληλος του νοσοκομείου
- Αν χρησιμοποιηθεί μηχανισμός συμπερασμού, η οντολογία θα μπορούσε να πρόβλεπει νέους συνδυασμούς ιατρονοσηλευτικού προσωπικού σε ομάδες θεραπείας ασθενών ενώ ακόμη θα μπορούσε να προβλέπει φάρμακα και τις αντίστοιχες ασθένειες, στις οποίες μπορούν να δοθούν ανάλογα με τις ανάγκες του ασθενούς

Γ. Περιγραφή Οντολογίας

20 κλάσεις οργανωμένες σε τουλάχιστον τρία επίπεδα ιεραρχίας (υποκλάσεων):

- Hospital: αντιπροσωπεύει τις βασικές κλάσεις του νοσοκομείου
- Public: αντιπροσωπεύει το Δημόσιο Νοσοκομείο
- Private: αντιπροσωπεύει το Ιδιωτικό Νοσοκομείο
- Clinic: αντιπροσωπεύει το είδος της κλινικής (λ.χ. καρδιολογική κλινική)
- Staff: αντιπροσωπεύει το προσωπικό του νοσοκομείου
- HealthCareWorker: αντιπροσωπεύει το ιατρονοσηλευτικό προσωπικό
- MedicalAssistant: αντιπροσωπεύει το προσωπικό (νοσηλευτές) που ασχολείται με τα προγράμματα κ.ο.κ.
- Nurse: αντιπροσωπεύει τις νοσοκόμες/τους νοσοκόμους
- Doctor: αντιπροσωπεύει το ιατρικό προσωπικό
- LogisticsTeam: αντιπροσωπεύει το προσωπικό που βρίσκεται στον τομέα των logistics
- CustomerServiceAgent: αντιπροσωπεύει το προσωπικό που είναι υπεύθυνο για την ποιότητα εξυπηρέτησης των ασθενών
- LogisticsCoordinator: αντιπροσωπεύει το προσωπικό που είναι υπεύθυνο για την εύρυθμη λειτουργία των logistics
- AssociateSupplyOfficer: αντιπροσωπεύει το προσωπικό που είναι υπεύθυνο για τις προμήθειες του νοσοκομείου
- LabTechnician: αντιπροσωπεύει το προσωπικό που απασχολείται στα διάφορα εργαστήρια

- Radiologist: αντιπροσωπεύει τους ραδιολόγους των ακτινογραφικών εργαστηρίων
- Microbiologist: αντιπροσωπεύει τους μικροβιολόγους που απασχολούνται στα μικροβιολογικά εργαστήρια
- MedicalEquipmentRepairTechnician: αντιπροσωπεύει το προσωπικό που είναι υπεύθυνο για την επιδιόρθωση των ιατρικών μηχανημάτων
- Patient: αντιπροσωπεύει τους ασθενείς
- Hospitalized: αντιπροσωπεύει ασθενείς που εισάγονται στο νοσοκομείο
- Visitor: αντιπροσωπεύει τους ασθενείς που έρχονται για απλή επίσκεψη και δεν πρόκειται να εισαχθούν στο νοσοκομείο

6 κλάσεις που αποτελούν υποκλάσεις άλλων (Subsumption):

- Physician: αντιπροσωπεύει γιατρούς που ασχολούνται με μη χειρουργικές θεραπείες
- Surgeon: αντιπροσωπεύει τους χειρουργούς
- CertifiedNursingAssistant: αντιπροσωπεύει τους/τις νοσοκόμους/νοσοκόμες
- ScrubNurse: αντιπροσωπεύει τους/τις νοσηλευτές/νοσηλεύτριες που απαιτούνται κατά τη διάρκεια ενός χειρουργείου
- ClinicalNurseSupervisor: αντιπροσωπεύει τον/την προϊστάμενο/προϊσταμένη των νοσοκόμων/νοσηλευτών
- EmergencyRoomRegisteredNurse: αντιπροσωπεύει τις νοσοκόμες που είναι στη γραμματεία των εφημερεύοντων περιστατικών

6 κλάσεις που είναι ξένες μεταξύ τους (Disjointness):

- Pathology: αντιπροσωπεύει την παθολογική κλινική
- Oncology: αντιπροσωπεύει την ογκολογική κλινική
- Dentistry: αντιπροσωπεύει την οδοντιατρική κλινική
- Pediatrics: αντιπροσωπεύει την παιδιατρική κλινική
- Dermatology: αντιπροσωπεύει τη δερματολογική κλινική
- Cardiology: αντιπροσωπεύει την καρδιολογική κλινική

6 κλάσεις που προκύπτουν από λογική σύνθεση άλλων:

Ένωση (Union):

- Medicine: αποτελεί ένωση των υποκλάσεων Analgesic και Antibiotic
- Disease: αποτελεί ένωση των υποκλάσεων KindeyDisease, AutoimmuneDisease, LungDisease και Cancer

Συμπλήρωμα (Complement):

- Public: αποτελεί συμπλήρωμα της κλάσης Private
- Analgesic: αποτελεί συμπλήρωμα της κλάσης Antibiotic

Τομή (Intersection):

- LungCancer: αποτελεί τομή των κλάσεων LungDisease και Cancer
- KindeyAutoimmune: αποτελεί τομή των κλάσεων AutoimmuneDisease και KindeyDisease

10 κλάσεις που προκύπτουν από περιορισμό (Restriction) σε σχέσεις:

Existential restriction (someValuesFrom):

- Hospital - Clinic: το νοσοκομείο περιέχει τουλάχιστον μία κλινική
- Staff - Degree: κάθε μέλος του προσωπικού διαθέτει τουλάχιστον ένα πτυχίο

Universal restriction (allValuesFrom):

- Room - Hospitalized: τα δωμάτια αποτελούνται από νοσηλευόμενους ασθενείς
- Laboratory - LabTechnician: στα εργαστήρια εργάζονται αποκλειστικά Lab Technicians

hasValue:

- Antibiotic: η κατηγορία αντιβιοτικά περιέχει φάρμακα που περιέχουν την ουσία Antimicrobial_Agents
- Surgeon: ο καρδιοθωρακικός χειρουργός είναι χειρουργός

Minimum/Maximum Cardinality:

- Hospital - Ambulance: κάθε Νοσοκομείο έχει τουλάχιστον 6 ασθενοφόρα
- Storage - Medicine: η αποθήκη μπορεί να περιέχει το πολύ 10.000 φάρμακα

2 κλάσεις που προκύπτουν από συνδυασμό λογικών πράξεων και περιορισμών σε σχέσεις:

- Room - noICU: το δωμάτιο αποτελείται μόνο από Hospitalized και το noICU δεν είναι μονάδα εντατικής θεραπείας, επομένως δε δέχεται βαριές περιπτώσεις
- Patient - Doctor: κάθε ασθενής εξετάζεται από τουλάχιστον ένα άτομο του ιατρονοσηλευτικού προσωπικού, το οποίο δεν μπορεί να είναι νοσοκόμα ή μέρος του προσωπικού που ασχολείται με τα προγράμματα

Δ. Παρακάτω ορίζονται οι ιδιότητες των κλάσεων (object/data properties):

13 ιδιότητες datatype

- email: literal - email μέλος προσωπικού (Staff)
- fullname: literal - ονοματεπώνυμο του ατόμου, στον οποίο ανήκει το πτυχίο (Degree)
- TIN: literal - ΑΦΜ μέλος προσωπικού (Staff)
- SSN: literal - ΑΜΚΑ ασθενούς (Patient)
- numberOfWorkers: positive integer - πλήθος εργαζομένων σε κάθε κλινική (Clinic)

- numberOf: positive integer - πλήθος ασθενοφόρων (Ambulance)
- specialty: literal - ειδικότητα γιατρού (Doctor)
- admissionDate: dateTime - ημερομηνία εισαγωγής Hospitalized Patient
- grade: float - βαθμός πτυχίου (Degree)
- acquireDate: dateTime - ημερομηνία απόκτησης πτυχίου (Degree)
- numberOfBeds: integer - πλήθος κρεβατιών ανά δωμάτιο μη εντατικής θεραπείας (NoICU)
- phone: literal - τηλέφωνο μέλους προσωπικού (Staff)
- staffID: integer - αριθμός ID μέλους προσωπικού (Staff)

4 ιδιότητες να οριστούν με τις αντίστοιχες αντίστροφες (inverse):

- belongsTo - hasDoctor: ο γιατρός ανήκει σε μία κλινική και η κλινική έχει κάποιο γιατρό
- treatedBy - Treats: ένας ασθενής εξετάζεται από κάποιο γιατρό και ένας γιατρός εξετάζει κάποιον ασθενή
- staysIn - accommodates: ένας Hospitalized Patient μένει σε ένα δωμάτιο και ένα δωμάτιο φιλοξενεί έναν ασθενή
- hasDegree - isOwnedBy: ένα μέλος του προσωπικού έχει κάποιο πτυχίο και ένα πτυχίο ανήκει σε ένα μέλος του προσωπικού

2 ιδιότητες να είναι συμμετρικές (symmetric):

- worksWith: αν κάποιος εργάζεται με κάποιον άλλο τότε ο δεύτερος εργάζεται με τον πρώτο
- staysWith: αν ένας Hospitalized Patient μένει με έναν άλλο τότε ο δεύτερος μένει με τον πρώτο

2 ιδιότητες να είναι μεταβατικές (transitive):

- sharesRoomWith: ένας Hospitalized Patient A μοιράζεται το δωμάτιο με τον B, ο B μοιράζεται με τον Γ, άρα ο Γ μοιράζεται το δωμάτιο με τον A
- isColleaguesWith: ο MedicalAssistant A είναι συνάδελφος με τον B, ο B είναι συνάδελφος με τον Γ, άρα ο Γ είναι συνάδελφος με τον A

2 ιδιότητες να είναι συναρτησιακές (functional):

- staysIn: ο ασθενής μένει σε ένα δωμάτιο
- worksAtLaboratory: ο Radiologist δουλεύει σε ένα εργαστήριο

2 ιδιότητες να είναι inverse functional:

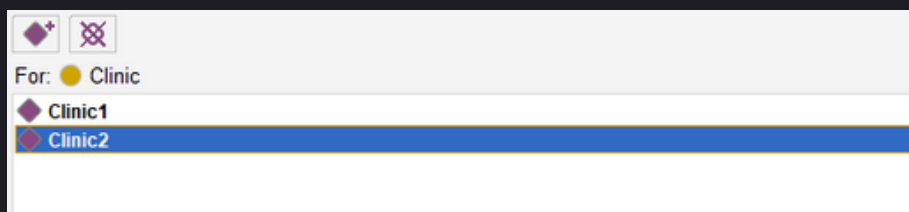
- hasEmployee: ένα μέλος του logisticsTeam δουλεύει σε ένα νοσοκομείο
- accommodatesPatient: η ICU δέχεται έναν Hospitalized Patient

4 ιδιότητες να αποτελούν subproperties άλλων ιδιοτήτων:

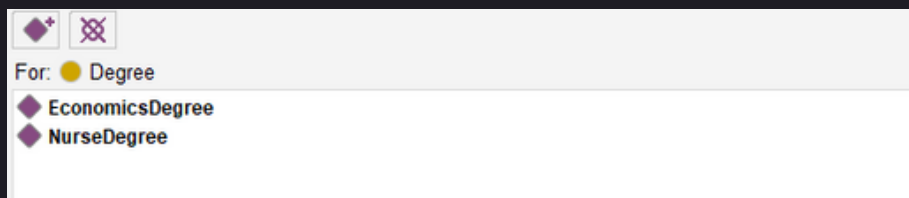
- staysIn - staysInICU/staysInNoICU: ο Hospitalized αφού εισαχθεί, επιλέγεται σε τι τύπου δωμάτιο θα μείνει (εντατικής θεραπείας ή μη εντατικής θεραπείας)
- hasEmployee - promotes: το νοσοκομείο μπορεί να προαγάγει ένα μέλος του logisticsTeam εφόσον προηγούμενως το έχει προσλάβει
- hasEmployee - givesLeave: το νοσοκομείο μπορεί να δώσει άδεια σε ένα μέλος του logisticsTeam αφότου το προσλάβει
- hasEmployee - givesBonus: το νοσοκομείο μπορεί να δώσει δώρο σε ένα μέλος του logisticsTeam αφότου το προσλάβει

Ε. Μερικά ενδεικτικά στιγμιότυπα για τις κλάσεις της οντολογίας:

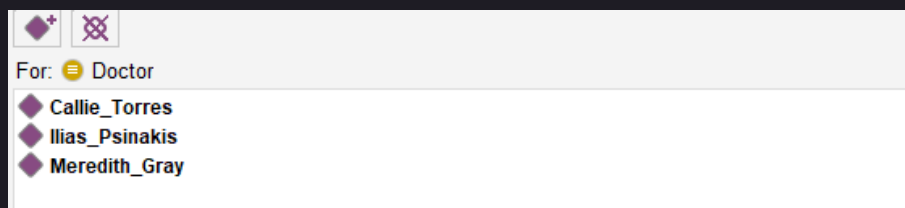
Clinic:



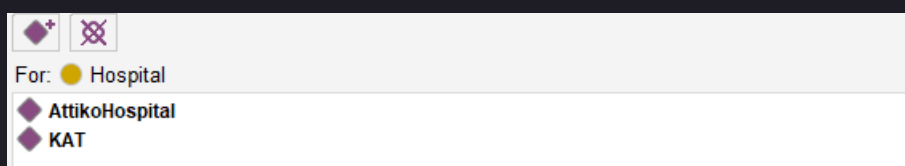
Degree:



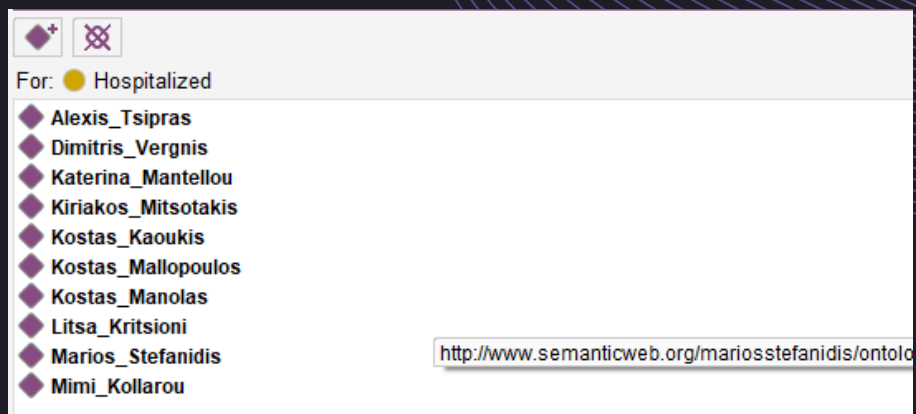
Doctor:



Hospital:

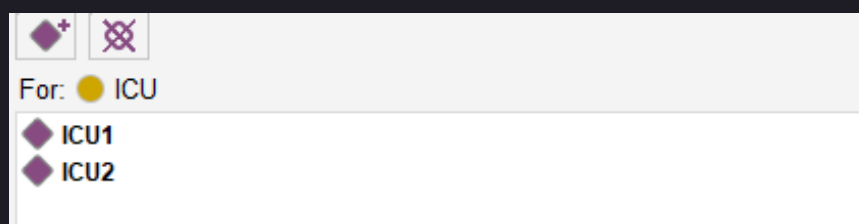


Hospitalized:



A screenshot of a web application interface. At the top, there are two icons: a purple diamond with a plus sign and a purple diamond with a cross. Below them is a header bar with the text "For: Hospitalized". The main content area is a list of names, each preceded by a purple diamond icon. The names are: Alexis_Tsipras, Dimitris_Vergnis, Katerina_Mantellou, Kiriakos_Mitsotakis, Kostas_Kaoukis, Kostas_Mallopoulos, Kostas_Manolas, Litsa_Kritsioni, Marios_Stefanidis, and Mimi_Kollarou. A URL is visible in the bottom right corner: <http://www.semanticweb.org/mariosstefanidis/ontolo>.

ICU:



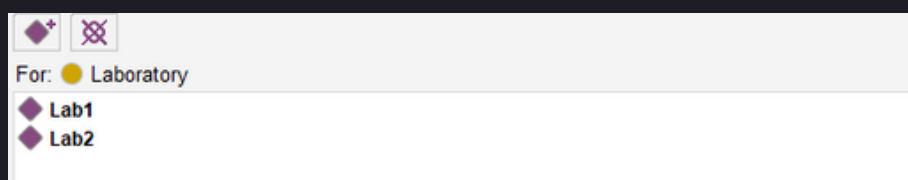
A screenshot of a web application interface. At the top, there are two icons: a purple diamond with a plus sign and a purple diamond with a cross. Below them is a header bar with the text "For: ICU". The main content area is a list of names, each preceded by a purple diamond icon. The names are: ICU1 and ICU2.

NoICU:



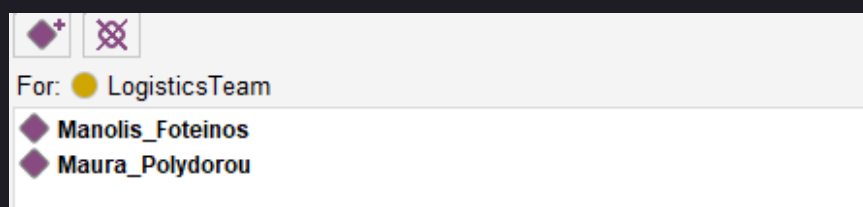
A screenshot of a web application interface. At the top, there are two icons: a purple diamond with a plus sign and a purple diamond with a cross. Below them is a header bar with the text "For: NoICU". The main content area is a list of names, each preceded by a purple diamond icon. The name is: NoICU1.

Laboratory:





A screenshot of a web application interface. At the top, there are two icons: a purple diamond with a plus sign and a purple diamond with a cross. Below them is a header bar with the text "For: Laboratory". The main content area is a list of names, each preceded by a purple diamond icon. The names are: Lab1 and Lab2.


LogisticsTeam:





A screenshot of a web application interface. At the top, there are two icons: a purple diamond with a plus sign and a purple diamond with a cross. Below them is a header bar with the text "For: LogisticsTeam". The main content area is a list of names, each preceded by a purple diamond icon. The names are: Manolis_Foteinos and Maura_Polydorou.


MedicalAssistant:





For:  MedicalAssistant


 Fotini_Aggelaki


 Katerina_Papadopoulou

 Maria_Tsavea



Nurse:





For:  Nurse

 Katerina_Mitropoulou



Radiologist:





For:  Radiologist


 Elena_Paparizou


Room:





For:  Room

 Room1

 Room12



 Room125


 Room2


 Room36


<http://www.semanticweb.org/mariosstefanidis/ontologies/202>


Staff:




For:  Staff

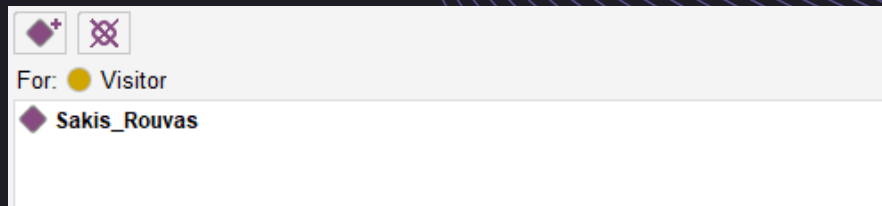
 Aggeliki_Kourou

 Konstantinos_Argiros

 Miltos_Pasxalidis

 Spiros_Vlachos

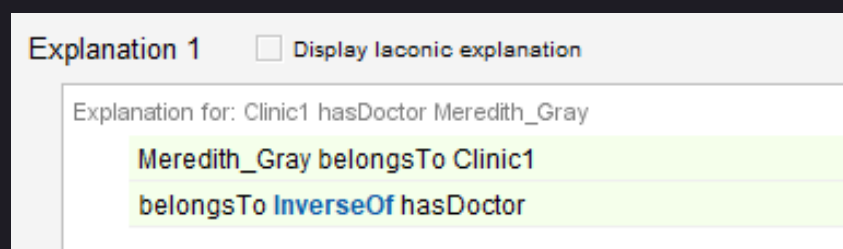
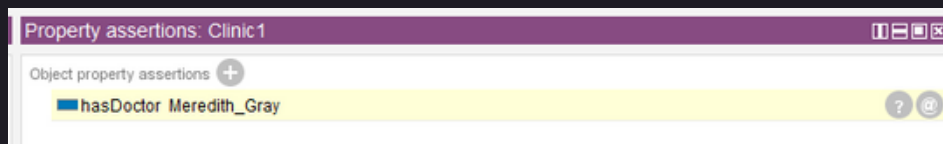
Visitor:



ΕΡΩΤΗΜΑ 3

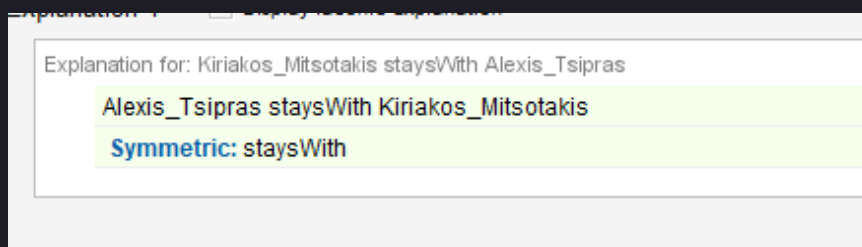
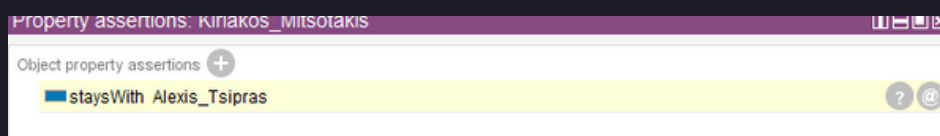
Clinic - Doctor (Inverse):

Η Meredith Gray ανήκει στην κλινική 1
Η κλινική 1 έχει γιατρό τη Meredith Gray



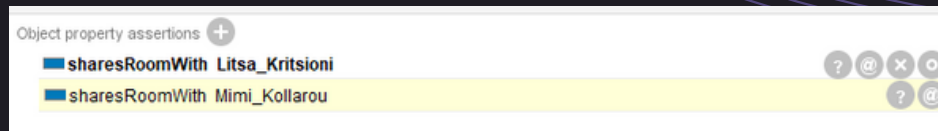
Hospitalized (Symmetric):

Ο Κυριάκος Μητσοτάκης μένει στο ίδιο δωμάτιο με τον Αλέξη Τσίπρα
Άρα ο Αλέξης Τσίπρας μένει στο ίδιο δωμάτιο με τον Κυριάκο Μητσοτάκη



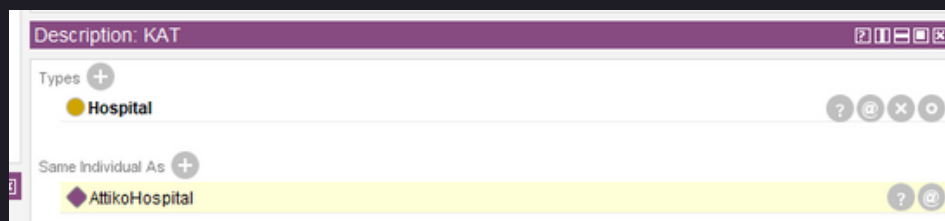
Hospitalized (Transitive):

Ο Κώστας Καούκης μοιράζεται το δωμάτιο με τη Λίτσα Κριτσιώνη
Η Λίτσα Κριτσιώνη μοιράζεται το δωμάτιο με τη Μιμή Κολλάρου
Άρα ο Κώστας Καούκης μοιράζεται το δωμάτιο με τη Μιμή Κολλάρου



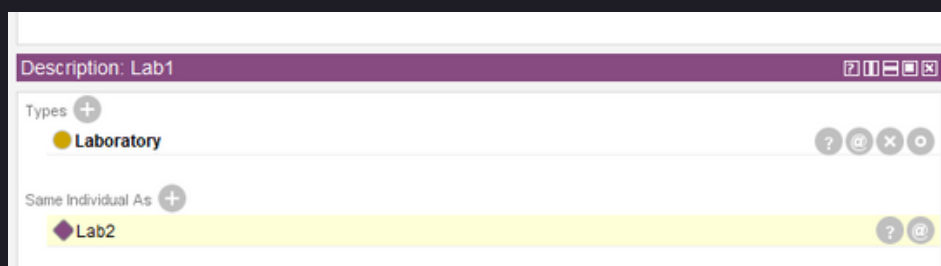
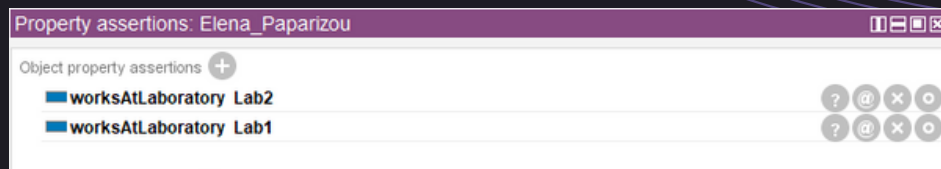
Hospital - Staff (Inverse Functional):

Το ΚΑΤ έχει υπάλληλο τη Μαύρα Πολυδώρου
Το Αττικόν έχει υπάλληλο τη Μαύρα Πολυδώρου
Άρα, το Αττικόν και το ΚΑΤ είναι το ίδιο instance Hospital



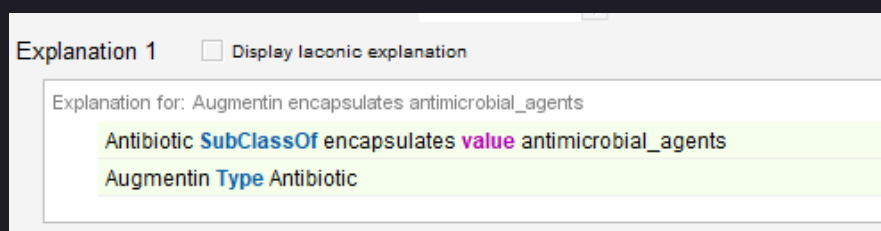
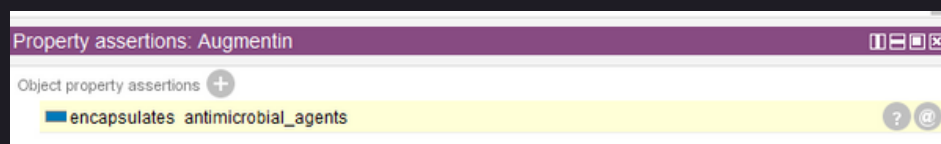
Laboratory - Radiologist (Functional):

Ο radiologist δουλεύει στο Lab1
Ο radiologist δουλεύει στο Lab2
Άρα, το Lab1 ταυτίζεται με το Lab2



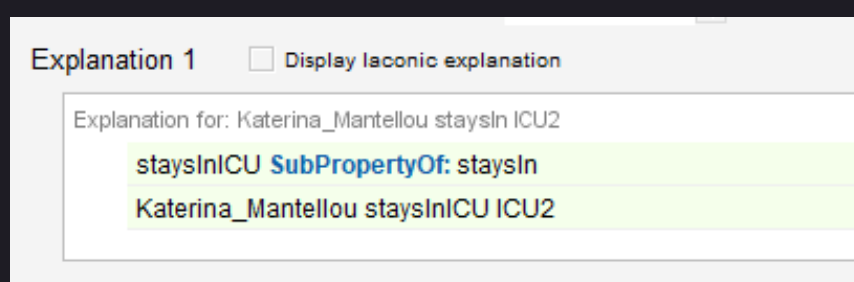
Medicine - Antibiotic (hasValue):

Ένα στοιχείο της κλάσης antibiotic περιέχει το συστατικό antimicrobial_agents
Επομένως η Augmentin που είναι antibiotic, πρέπει να περιέχει το παραπάνω συστατικό



Subproperty:

Η ασθενής Κατερίνα Μαντέλλου είναι Hospitalized και μένει σε δωμάτιο
Αφού μένει σε δωμάτιο, τότε μένει είτε σε μονάδα εντατικής θεραπείας είτε σε απλό δωμάτιο



Η Μαύρα Πολυδώρου είναι υπάλληλος ενός νοσοκομείου
Αφού η Μαύρα είναι υπάλληλος, τότε το νοσοκομείο μπορεί να της δώσει bonus, άδεια ή προαγωγή

Explanation 2 ☐ Display laconic explanation

Explanation for: AttikoHospital givesBonus Manolis_Foteinos

- 1) promotes **SubPropertyOf**: hasEmployee
- 2) **InverseFunctional**: hasEmployee
- 3) KAT hasEmployee Maura_Polydorou
- 4) AttikoHospital promotes Maura_Polydorou
- 5) KAT givesBonus Manolis_Foteinos

Universal Restriction:

Ο ασθενής Μάρκος Τσιγκούνης θα πρέπει να είναι Hospitalized για να μπορεί να μείνει σε δωμάτιο

Description: Markos_Tsigkounis

Types +

- Patient
- Hospitalized

Explanation 1 ☐ Display laconic explanation

Explanation for: Markos_Tsigkounis Type Hospitalized

- 1) Room **SubClassOf** consistsOf **only** Hospitalized
- 2) Room125 **Type** Room
- 3) Room125 consistsOf Markos_Tsigkounis

AND:

Δημιουργείται μία υποκλάση KidneyAutoimmuneDisease, όπου λόγω του AND θα είναι υποκλάση και της AutoimmuneDisease και της KidneyDisease

Equivalent To +

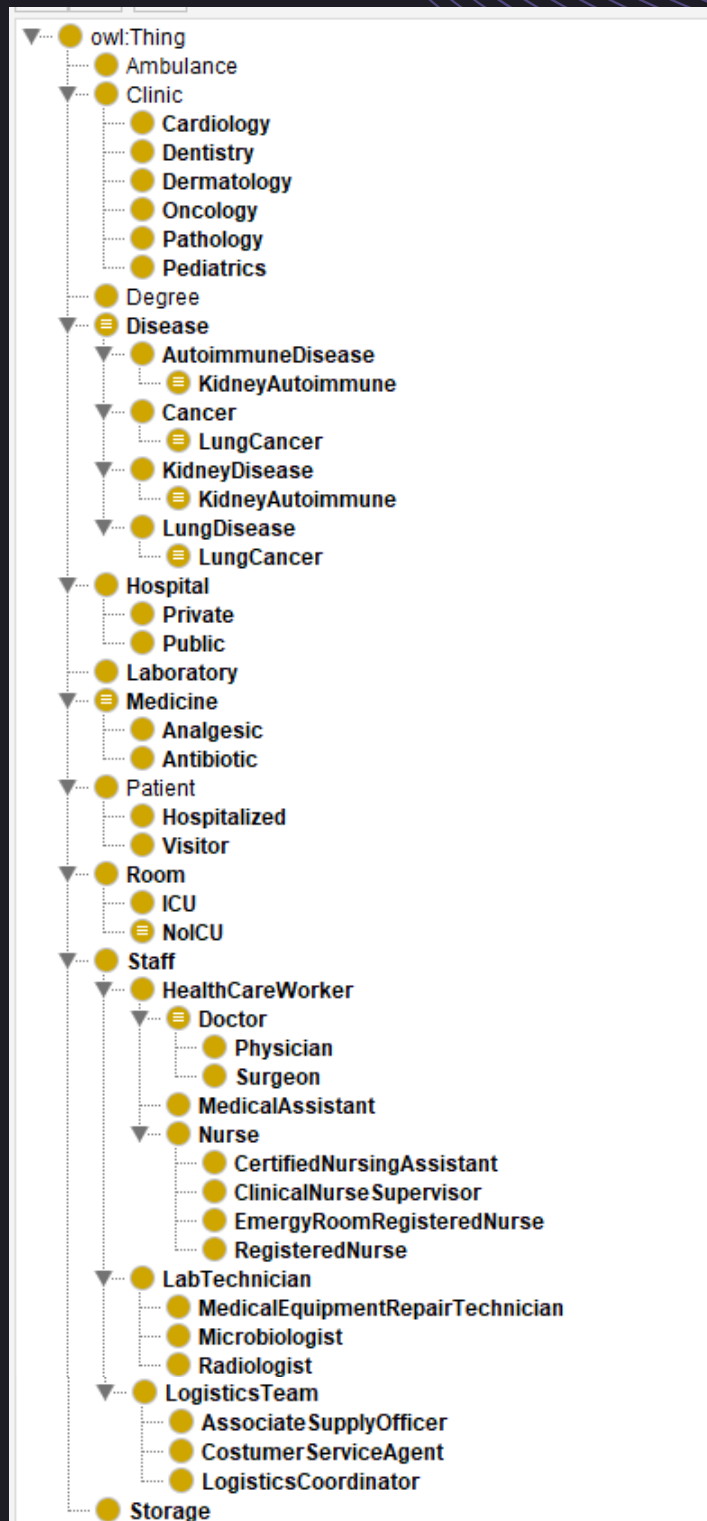
- AutoimmuneDisease
- and KidneyDisease

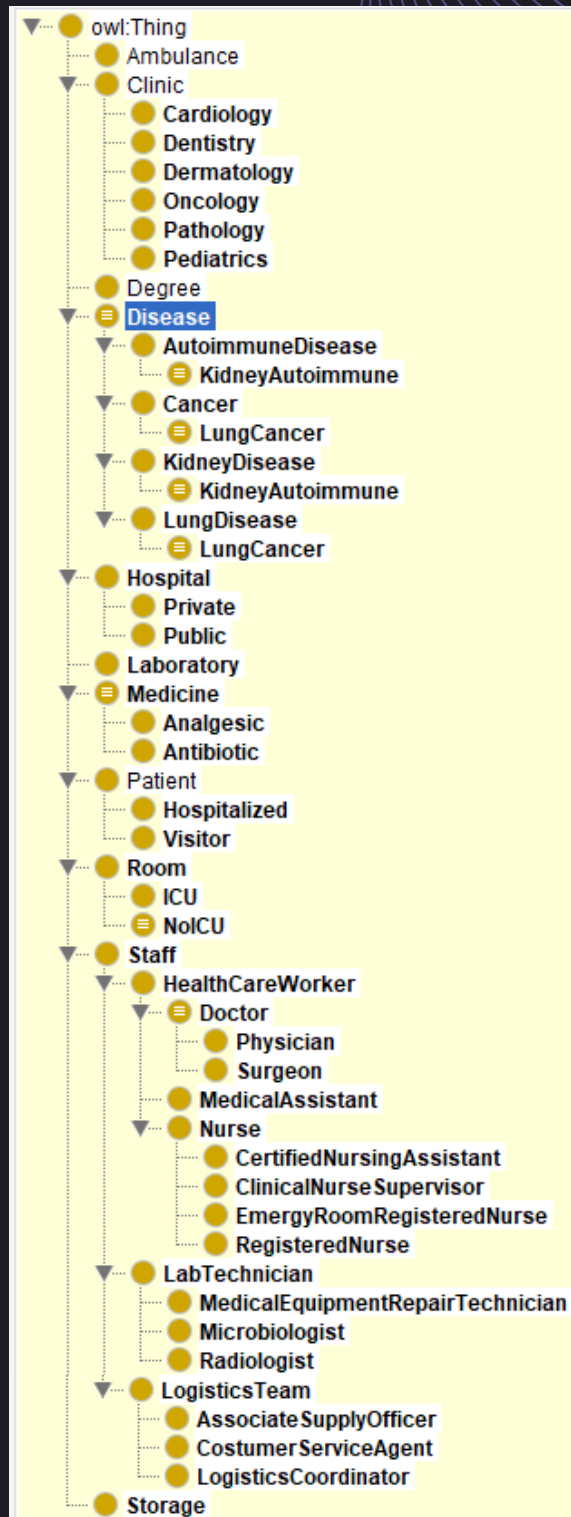
SubClass Of +

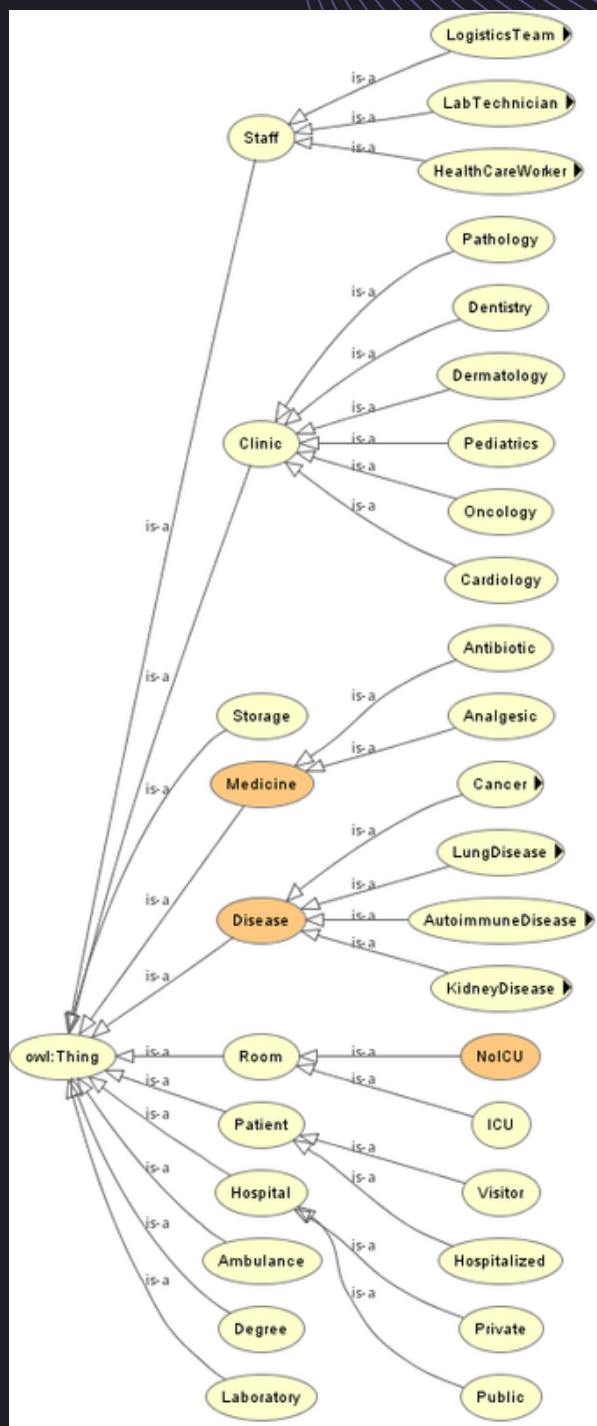
- AutoimmuneDisease
- KidneyDisease

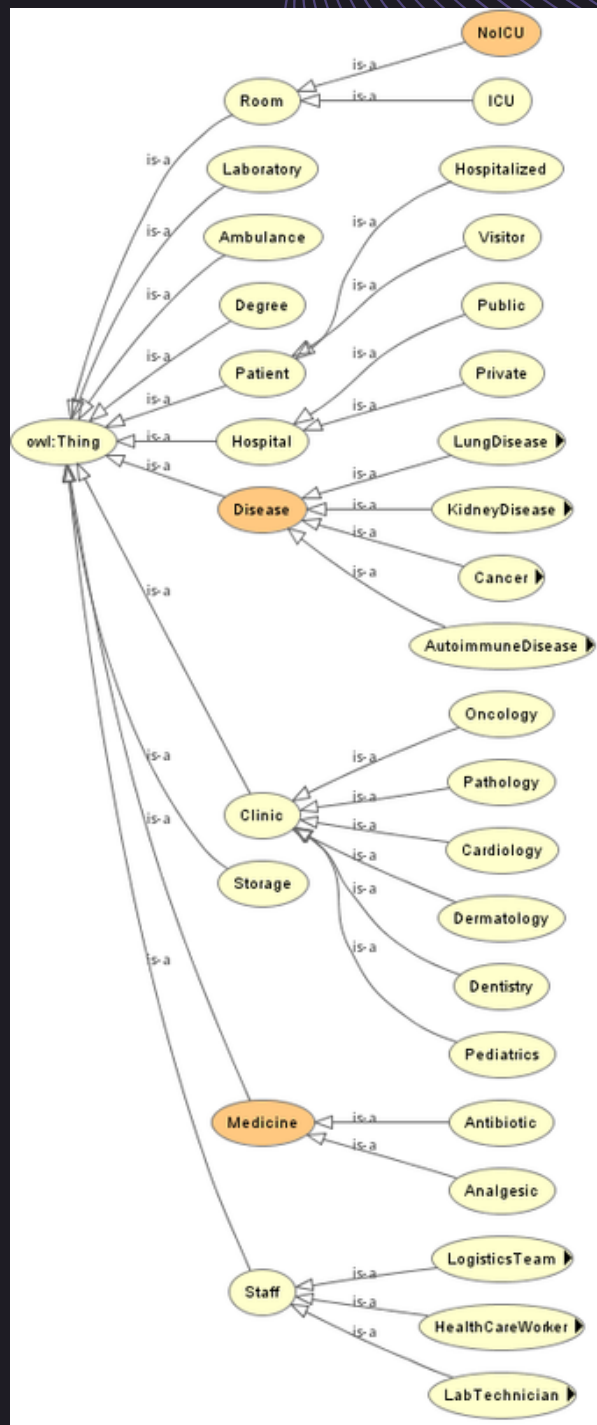
ΕΡΩΤΗΜΑ 4

α.









γ. Η Asserted Class Hierarchy είναι η ιεραρχία που δείχνει την ταξινόμηση της οντολογίας που προκύπτει χρησιμοποιώντας αξιώματα `subClassOf`. Η Inferred Hierarchy δείχνει την ταξινόμηση όπως προκύπτει από τον reasoner. Στη συγκεκριμένη περίπτωση, δεν υπάρχουν διαφορές μεταξύ των ιεραρχιών των κλάσεων και των αντίστοιχων γράφων που φαίνονται παραπάνω.

ΕΡΩΤΗΜΑ 5

α.

1. **Περιγραφή:** Εμφάνισε όλους τους Doctor και το αντίστοιχο specialty τους, είτε έχουν, είτε δεν έχουν staffID, ταξινομημένους κατά όνομα.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?Doctor ?specialty ?ID
WHERE {
    ?Doctor rdf:type <http://www.semanticweb.org/mariosstefanidis/ontologies/2022/11/hospital#Doctor>;
    <http://www.semanticweb.org/mariosstefanidis/ontologies/2022/11/hospital#specialty> ?specialty;
    OPTIONAL
    {
        ?Doctor <http://www.semanticweb.org/mariosstefanidis/ontologies/2022/11/hospital#staffID> ?ID
    }
}
ORDER BY ?Doctor
```

Doctor	specialty	ID
Callie_Torres	"Orthopedic""<http://www.w3.org/2000/01/rdf-schema#Li	
Ilias_Psinakis	"Andrologist""<http://www.w3.org/2000/01/rdf-schema#Li	
Meredith_Gray	"General Surgeon""<http://www.w3.org/2000/01/rdf-sche: "153""<http://www.w3.org/2001/XMLSchema#integer>	

2. **Περιγραφή:** Εμφάνισε όλα τα Clinic που είναι τύπου Oncology ή Pathology, ταξινομημένα σύμφωνα με το όνομά τους.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?Clinic
{
    {
        ?Clinic rdf:type <http://www.semanticweb.org/mariosstefanidis/ontologies/2022/11/hospital#Oncology>.
    }
    UNION
    {
        ?Clinic rdf:type <http://www.semanticweb.org/mariosstefanidis/ontologies/2022/11/hospital#Pathology>
    }
}
ORDER BY ?Clinic
```

Clinic
Cytopathology
Forensic_Pathology
MedicalOncology
Radiation_Oncology

3. **Περιγραφή:** Εμφάνισε τους γιατρούς που έχουν staffID μεγαλύτερο από 155.

```
SELECT ?Doctor ?ID
WHERE
{
    ?Doctor rdf:type <http://www.semanticweb.org/mariosstefanidis/ontologies/2022/11/hospital#Doctor>;
    <http://www.semanticweb.org/mariosstefanidis/ontologies/2022/11/hospital#staffID> ?ID

    FILTER (?ID>155)
}
```

	Doctor	ID
Ilias_Psinakis		"200" ^{^^} <http://www.w3.org/2001/XMLSchema#integer>

4. **Περιγραφή:** Εμφάνισε τα 2 top degree που έχουν το μεγαλύτερο βαθμό από τα υπόλοιπα.

```
SELECT ?Degree (MAX(?grade) AS ?Max_Grade)
WHERE
{
    ?Degree rdf:type <http://www.semanticweb.org/mariosstefanidis/ontologies/2022/11/hospital#Degree>.
    ?Degree <http://www.semanticweb.org/mariosstefanidis/ontologies/2022/11/hospital#grade> ?grade.
}
GROUP BY ?Degree
ORDER BY desc(?Max_Grade)
LIMIT 2
```

	Degree	Max_Grade
EconomicsDegree		"8.9" ^{^^} <http://www.w3.org/2001/XMLSchema#float>
BusinessAdministrationDegree		"8.6" ^{^^} <http://www.w3.org/2001/XMLSchema#float>

5. **Περιγραφή:** Εμφάνισε αφενός τα μοναδικά νοσοκομεία που διαθέτουν staff που έχει λάβει πλήθος αδειών μεγαλύτερο από 5 μέρες και αφετέρου το νούμερο του κινητού τους, αν διαθέτουν/αν έχει καταχωρηθεί.

```
SELECT DISTINCT ?Hospital ?phone
WHERE
{
    ?Hospital rdf:type <http://www.semanticweb.org/mariosstefanidis/ontologies/2022/11/hospital#Hospital>.
    ?Hospital <http://www.semanticweb.org/mariosstefanidis/ontologies/2022/11/hospital#hasEmployee> ?Staff.
    ?Staff <http://www.semanticweb.org/mariosstefanidis/ontologies/2022/11/hospital#numberOfLeave> ?leave
    OPTIONAL
    {
        ?Staff <http://www.semanticweb.org/mariosstefanidis/ontologies/2022/11/hospital#phone> ?phone
    }
    FILTER(?leave > 5)
}
```

	Hospital	phone
KAT		"6945777958" ^{^^} <http://www.w3.org/2000/01/rdf-schema#Literal>
KAT		"6945777958" ^{^^} <http://www.w3.org/2000/01/rdf-schema#Literal>
AttikoHospital		"6945777958" ^{^^} <http://www.w3.org/2000/01/rdf-schema#Literal>

Σημείωση: Όπως έχει αναλυθεί στο ερώτημα 3, ο reasoner θεωρεί πως τα νοσοκομεία KAT και AttikoHospital είναι το ίδιο νοσοκομείο.

β.

	name
S1	autogen0:supervises(?x, ?y) ^ autogen0:associatesWith(?y, ?z) -> autogen0:colleaguesWith(?x, ?z)
S2	autogen0:treatedBy(?x, ?y) ^ autogen0:belongsTo(?y, ?z) -> autogen0:isTreatedIn(?x, ?z)
S3	autogen0:hasDisease(?x, ?y) ^ autogen0:diseasesIsTreatedBy(?y, ?z) -> autogen0:patientPrescribed(?x, ?z)
S4	autogen0:surgeonAt(?x, ?y) ^ autogen0:hasScrubNurse(?y, ?z) -> autogen0:helpedBy(?x, ?z)
S5	autogen0:isTreatedIn(?x, ?y) ^ autogen0:hasDoctor(?y, ?z) -> autogen0:canBeTreatedBy(?x, ?z)

S1: $\text{supervises}(?x, ?y) \wedge \text{associatesWith}(?y, ?z) \rightarrow \text{isColleaguesWith}(?x, ?z)$

Αν ένας γιατρός είναι επιβλέπων ενός άλλου και ο δεύτερος συνεργάζεται με έναν τρίτο, τότε ο πρώτος και ο τρίτος είναι συνάδελφοι.

S2: $\text{treatedBy}(?x, ?y) \wedge \text{belongsTo}(?y, ?z) \rightarrow \text{isTreatedIn}(?x, ?z)$

Αν ένας ασθενής εξετάζεται από έναν γιατρό, ο οποίος ανήκει σε κάποια συγκεκριμένη κλινική, τότε ο ασθενής θεραπεύεται στη συγκεκριμένη κλινική.

S3: $\text{hasDisease}(?x, ?y) \wedge \text{diseasesIsTreatedBy}(?y, ?z) \rightarrow \text{patientPrescribed}(?x, ?z)$

Αν ένας ασθενής έχει μία ασθένεια και η συγκεκριμένη ασθένεια θεραπεύεται με ένα φάρμακο, τότε στον ασθενή θα συνταγογραφηθεί αυτό το φάρμακο.

S4: $\text{surgeonAt}(?x, ?y) \wedge \text{hasScrubNurse}(?y, ?z) \rightarrow \text{helpedBy}(?x, ?z)$

Αν ένας χειρουργός εργάζεται σε κάποια συγκεκριμένη κλινική και η κλινική έχει κάποια νοσοκόμα χειρουργείου, τότε η νοσοκόμα θα βοηθήσει στο χειρουργείο το χειρουργό.

S5: $\text{isTreatedIn}(?x, ?y) \wedge \text{hasDoctor}(?y, ?z) \rightarrow \text{canBeTreatedBy}(?x, ?z)$

Αν ένας ασθενής θεραπεύεται σε μία κλινική και η κλινική έχει έναν γιατρό τότε αυτός ο γιατρός μπορεί να θεραπεύσει τον ασθενή.

ΕΡΩΤΗΜΑ 6

Ως **open-world assumption** θεωρείται η υπόθεση σύμφωνα με την οποία, για τη διεξαγωγή του συμπεράσματος, θεωρείται ότι η αλήθεια μίας δήλωσης μπορεί να ισχύει, ανεξάρτητα αν η τελευταία είναι ευρέως γνωστή.

Τέτοιο παράδειγμα στην παρούσα οντολογία αποτελεί το στιγμιότυπο *Lupus*, το οποίο ανήκει στην κλάση *AutoimmuneDisease* αλλά και στην κλάση *KidneyDisease*, αφού ανήκει στην κλάση *KidneyAutoimmune*.

Ως **non-unique-name assumption** θεωρείται η υπόθεση, κατά την οποία στιγμιότυπα με διαφορετικά ονόματα/ετικέτες ανήκουν στην ίδια κλάση. Αυτού του είδους την υπόθεση εφαρμόζει το *Protege*, όταν υπάρχει κάποια δήλωση στο πεδίο *EquivalentTo*.

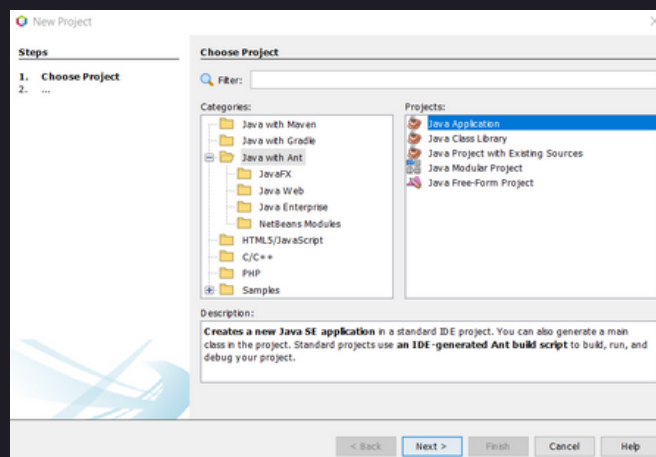
Τέτοιο παράδειγμα στην παρούσα οντολογία αποτελεί η κλάση *Disease*, η οποία είναι αντίστοιχη με οποιαδήποτε από τις κλάσεις *AutoimmuneDisease*, *Cancer*, *KidneyDisease* ή *LungDisease*.

ΕΡΩΤΗΜΑ 7

Η εφαρμογή συντάχθηκε σε Java (version 11) χρησιμοποιώντας το εργαλείο Netbeans για την ανάπτυξη γραφικού περιβάλλοντος. Χρησιμοποιήθηκαν οι εξής βιβλιοθήκες, με σκοπό την διαχείριση του αρχείου οντολογίας .owl:

```
import org.apache.jena.rdf.model.*;
import org.apache.jena.ontology.*;
import org.apache.jena.util.iterator.ExtendedIterator;
import openllet.jena.PelletReasonerFactory;
import org.apache.jena.query.*;
import org.apache.jena.query.QueryExecution;
import org.apache.jena.query.QueryExecutionFactory;
import org.apache.jena.query.QueryFactory;
```

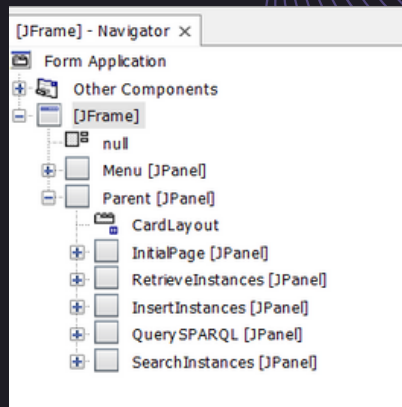
Η εφαρμογή δημιουργήθηκε ως εξής File -> New Project -> Java with Ant -> Java Application



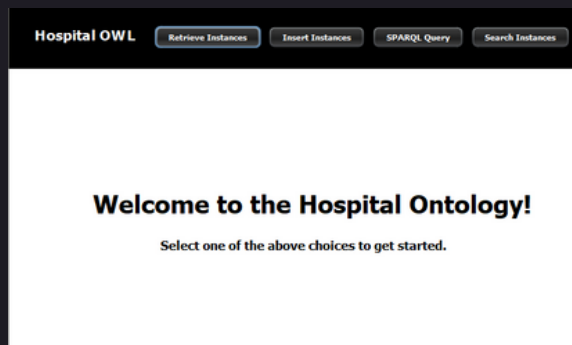
Ακολουθώντας την παρακάτω διαδρομή εντοπίζονται τα .jar αρχεία που απαιτούνται για την χρήση των παραπάνω βιβλιοθηκών (έχουν ήδη ενσωματωθεί στην εφαρμογή αλλά σε περίπτωση προβλήματος, μπορούν να βρεθούν και στο φάκελο Jena-Libraries που υπάρχει σε αυτό το .zip αρχείο):

δεξί κλικ Exercise7-GUI -> Properties -> Libraries -> Classpath

Η συγκεκριμένη εφαρμογή δημιουργήθηκε με τη λειτουργία CardLayout που προσφέρει το εργαλείο Netbeans. Πιο συγκεκριμένα, υπάρχει το Menu (JPanel), σύμφωνα με το οποίο μπορεί ο χρήστης να περιηγηθεί στις διάφορες επιλογές που προσφέρει η εφαρμογή. Ανάλογα με την επιλογή που θα πραγματοποιήσει ο χρήστης, υπάρχει ένα δεύτερο JPanel (Parent), το οποίο μεταβάλλεται αντίστοιχα.



Παρακάτω φαίνεται η αρχική σελίδα της εφαρμογής. Όπως αναφέρθηκε, υπάρχει ένας navigator, που εμφανίζει τις λειτουργίες που καθορίζονται από την εκφώνηση της άσκησης.



Η main που εκτελείται κατά την εκκίνηση της εφαρμογής ουσιαστικά φορτώνει την οντολογία hospital.owl που έχει δημιουργηθεί και εφαρμόζει το μοντέλο συμπερασμού (inference)

```
model = ModelFactory.createOntologyModel(OntModelSpec.OWL_DL_MEM);
model.read("hospital.owl", null);
//create model with pellet reasoner
m = ModelFactory.createOntologyModel(PelletReasonerFactory.THE_SPEC);
//load file
m.read("hospital.owl", null);
//set base
base = "http://www.semanticweb.org/mariosstefanidis/ontologies/2022/11/hospital#";
m.setStrictMode(false);

InfModel inf = ModelFactory.createRDFSModel(model);
```

Σημαντικό να αναφερθούν είναι κάποιες βασικές συναρτήσεις που έχουν δημιουργηθεί και καλούνται συχνά κατά τη διάρκεια λειτουργίας της εφαρμογής.

- **retrieveClasses** -> Εκτελείται κάθε φορά που πραγματοποιείται μετάβαση από ένα JPanel σε ένα άλλο (μέσω των επιλογών που προσφέρει το μενού της εφαρμογής), προκειμένου το αντίστοιχο JComboBox (όταν υπάρχει) να περιέχει τις κλάσεις που υπάρχουν στην παρούσα οντολογία που έχει δημιουργηθεί
- **getProperties** -> Αντίστοιχα συμβαίνει και για τα properties (object/data properties) που έχουν δημιουργηθεί στην παρούσα οντολογία


```

protected void retrieveClasses(JComboBox cb) {
    //string array in which the results will be stored
    ArrayList<String> classesArray = new ArrayList<>();
    //iterator that contains all the classes of the OWL ontology
    ExtendedIterator<OntClass> iter = model.listNamedClasses();

    //iterate
    while (iter.hasNext()) {
        //get the ontology class
        OntClass myClass = (OntClass) iter.next();
        //get its name
        String lnClass = myClass.getLocalName();

        if (!myClass.toString().contains("http")) {
            continue;
        }
        //if it returns a name, then add it to the array
        if (lnClass != null) {
            classesArray.add(lnClass);
        }
    }

    //pass every item in the array to the JComboBox GUI item
    classesArray.forEach(i -> {
        cb.addItem(i);
    });
}

```

```

protected void getProperties(String myClass, JComboBox jcb) {
    //get the selected class from the corresponding JComboBox
    //and create the correct URI
    OntClass selectedClass = model.getOntClass(base + myClass);

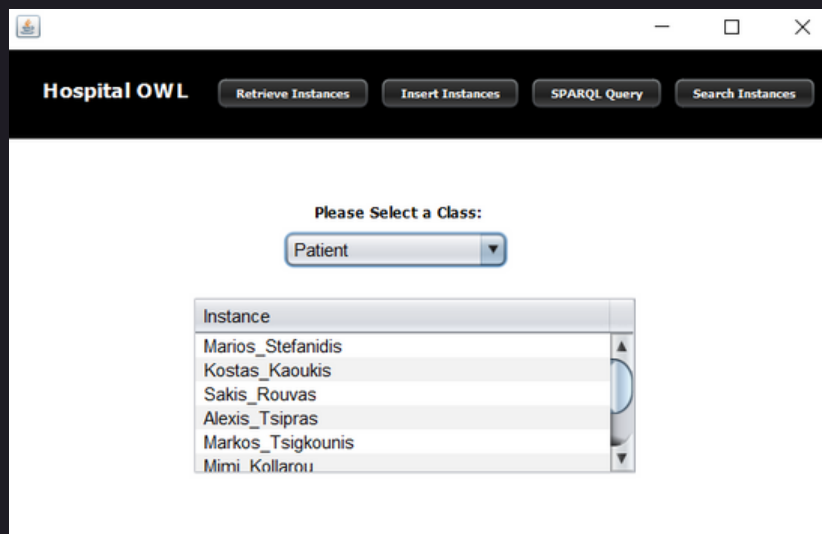
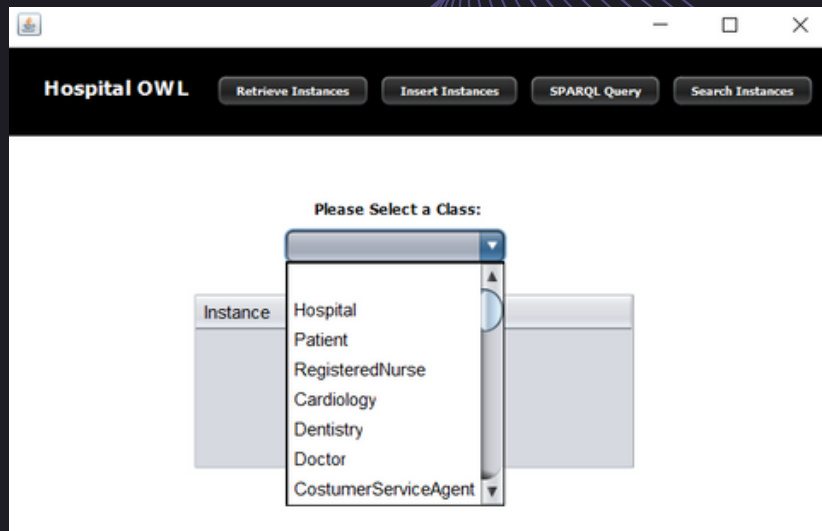
    //iterator that contains all the properties of the specific class
    ExtendedIterator<OntProperty> prop = selectedClass.listDeclaredProperties(true);
    //string array in which the results will be stored
    ArrayList<String> propArray = new ArrayList<>();

    //iterate
    while (prop.hasNext()) {
        //get the property
        OntProperty op = prop.next();
        //add it the array
        propArray.add(op.getLocalName());
    }

    //pass every item in the array to the JComboBox GUI item
    propArray.forEach(i -> {
        jcb.addItem(i);
    });
}

```

Επιλέγοντας από το μενού την λειτουργία **Retrieve Instances**, εμφανίζεται το παρακάτω παράθυρο. Όπως φαίνεται, το αντίστοιχο JComboBox περιέχει όλες τις κλάσεις που εντοπίζονται στην οντολογία. Όταν ο χρήστης επιλέξει μία από αυτές, τότε στον πίνακα εμφανίζονται όλα τα instances τις συγκεκριμένης κλάσης.

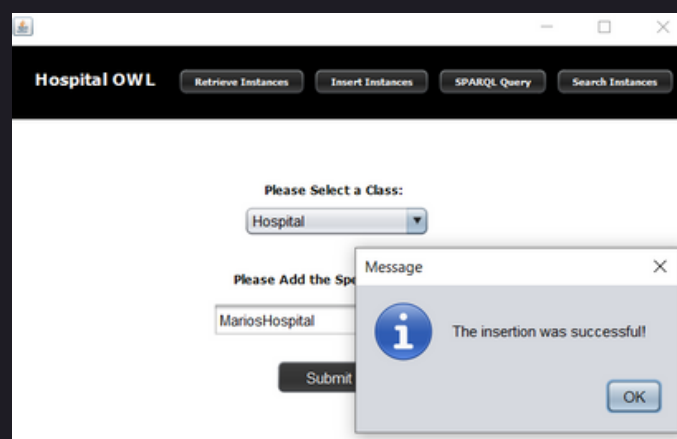
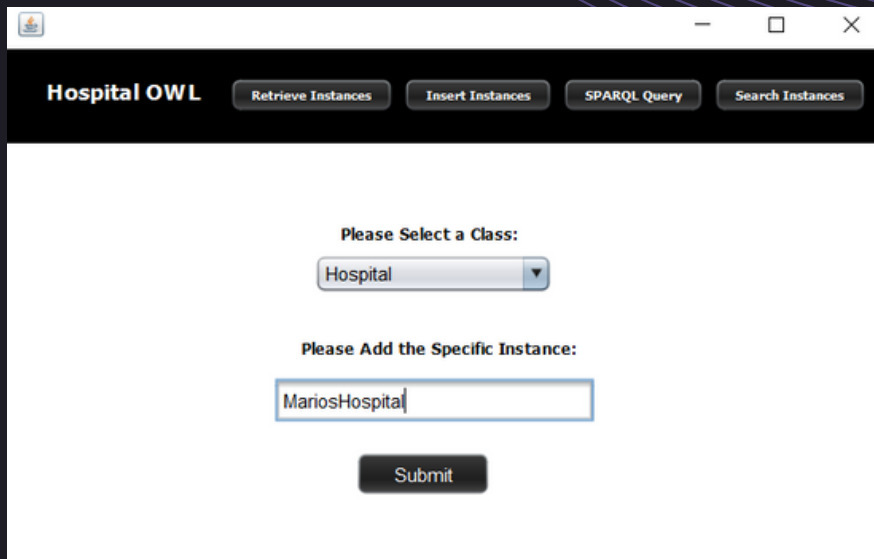


Παρακάτω φαίνεται ο κώδικας που έχει υλοποιηθεί για την παραπάνω λειτουργία, ο οποίος εκτελείται κάθε φορά που ο χρήστης επιλέγει μια από τις κλάσεις που υπάρχει στο `jComboBox`.

```
private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {
    //get the item that the user selected from the jComboBox
    String selected_class = jComboBox1.getSelectedItem().toString();
    //get the ontology class
    OntClass myClass = m.getOntClass(base + selected_class);
    //initialize the table
    DefaultTableModel tableModel = (DefaultTableModel) jTable3.getModel();
    tableModel.setRowCount(0);
    //iterator that contains all the instances of the selected class
    ExtendedIterator instances = myClass.listInstances();

    //iterate
    while (instances.hasNext()) {
        //get the instance
        Individual myInstance = (Individual) instances.next();
        //get the name of the instance
        String individual = myInstance.getLocalName();
        //add it to the table
        String tbData[] = {individual};
        tableModel.addRow(tbData);
    }
}
```

Επιλέγοντας από το μενού την λειτουργία **Insert Instances**, εμφανίζεται το παρακάτω παράθυρο. Το αντίστοιχο JComboBox περιέχει όλες τις κλάσεις που εντοπίζονται στην οντολογία. Ο χρήστης επιλέγοντας μία από αυτές και έπειτα τη λειτουργία submit, μπορεί να προσθέσει ένα καινούργιο instance της συγκεκριμένης κλάσης.



Παρακάτω φαίνεται ο κώδικας που έχει υλοποιηθεί για την παραπάνω λειτουργία, ο οποίος δημιουργεί ένα νέο .owl αρχείο που περιέχει το instance που δημιούργησε ο χρήστης.

```
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {  
    //get the selected class from the corresponding JComboBox  
    String selectedClass = jComboBox2.getSelectedItem().toString();  
    //get the name of the instance that the user typed  
    String newInstance = jTextField1.getText();  
    //get the ontology class  
    OntClass sel_Class = model.getOntClass(base + selectedClass);  
    //create the instance of the specific ontology class  
    Individual instance = model.createIndividual(base + newInstance, sel_Class);  
  
    OutputStream output;  
    //create a new .owl file which contains the instance  
    //that was created from the user  
    try {  
        output = new FileOutputStream("hospitalFinal.owl");  
        model.write(output);  
        JOptionPane.showMessageDialog(null, "The insertion was successful!");  
    }  
    catch (FileNotFoundException ex) {  
        System.out.println("Something went wrong.");  
    }  
}
```

Παρακάτω φαίνεται η καταχώρηση του καινούργιου instance.

```
<hospital:isOwnedBy>
  <hospital:Staff rdf:about="http://www.semanticweb.org/mariosstefanidis/ontologies/2022/11/hospital#Spiros_Vlachos">
    <hospital:staffID rdf:datatype="http://www.w3.org/2001/XMLSchema#integer"
    >86</hospital:staffID>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual"/>
  </hospital:Staff>
</hospital:isOwnedBy>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual"/>
</hospital:Degree>
<hospital:Hospital rdf:about="http://www.semanticweb.org/mariosstefanidis/ontologies/2022/11/hospital#MariosHospital"/>
<swrl:Imp>
```

Επιλέγοντας από το μενού την λειτουργία **SPARQL Query**, εμφανίζεται το παρακάτω παράθυρο. Το JComboBox που βρίσκεται αριστερά, περιέχει όλες τις κλάσεις που εντοπίζονται στην οντολογία ενώ το JComboBox που βρίσκεται δεξιά, περιέχει τις ιδιότητες που χαρακτηρίζουν την προηγουμένως επιλεγθείσα κλάση. Τα queries μπορούν να εκτελεστούν, είτε με την συμπλήρωση του πεδίου Value, είτε όχι. Αναλόγως, πραγματοποιείται και το αντίστοιχο query.

The image shows two screenshots of the 'Hospital OWL' application interface, specifically the 'SPARQL Query' section. Both screenshots have a header bar with buttons: 'Retrieve Instances', 'Insert Instances', 'SPARQL Query' (highlighted), and 'Search Instances'.

Top Screenshot:

- 'Please Select a Class:' dropdown: Doctor
- 'Please Select a Property:' dropdown: specialty
- 'Please Provide a Value:' text box: (empty)
- 'Query Result(s)' list: Ilias_Psinakis, Callie_Torres, Meredith_Gray
- 'Submit' button: present

Bottom Screenshot:

- 'Please Select a Class:' dropdown: Staff
- 'Please Select a Property:' dropdown: staffID
- 'Please Provide a Value:' text box: 102
- 'Query Result(s)' list: Miltos_Pasxalidis
- 'Submit' button: present

Παρακάτω φαίνεται ο κώδικας που έχει υλοποιηθεί για την παραπάνω λειτουργία. Αναλόγως, αν το πεδίο Value είναι συμπληρωμένο ή όχι, εκτελείται το αντίστοιχο SPARQL Query. Όπως φαίνεται και στον κώδικα, τα queries που τρέχει η εφαρμογή, έχουν συγκεκριμένη δομή.

```

private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
    //get the context of the JComboBoxes that the user selected
    String selectedProperty = jComboBox4.getSelectedItem().toString();
    String selectedClass = jComboBox3.getSelectedItem().toString();
    String queryString;
    //initialize the table
    DefaultTableModel tblModel = (DefaultTableModel) jTable2.getModel();
    tblModel.setRowCount(0);

    //run the corresponding SPARQL query depending on the context of
    // the jTextField (if it's empty or not)
    if (jTextField2.getText().isEmpty()) {
        queryString = "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>"
            + "PREFIX owl: <http://www.w3.org/2002/07/owl#>"
            + "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>"
            + "PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>"
            + "PREFIX : <http://www.semanticweb.org/mariosstefanidis/ontologies/2022/11/hospital#>"
            + "SELECT ?x\n"
            + "WHERE\n"
            + "{\n"
            + "?x rdf:type : " + selectedClass + ";\n"
            + " " + selectedProperty + " ?y\n"
            + "}";
    }
    else {
        String propValue = jTextField2.getText();

        queryString = "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>"
            + "PREFIX owl: <http://www.w3.org/2002/07/owl#>"
            + "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>"
            + "PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>"
            + "PREFIX : <http://www.semanticweb.org/mariosstefanidis/ontologies/2022/11/hospital#>"
            + "SELECT ?x\n"
            + "WHERE\n"
            + "{\n"
            + "?x rdf:type : " + selectedClass + ";\n"
            + " " + selectedProperty + " ?y\n"
            + "FILTER (?y = " + propValue + ")\n"
            + "}";
    }

    //create query
    Query query = QueryFactory.create(queryString);
    //execute query
    try (QueryExecution qexec = QueryExecutionFactory.create(query, m)) {
        //get the result(s) of the query
        ResultSet result = qexec.execSelect();
        //iterate
        for (; result.hasNext(); ) {
            QuerySolution qsol = result.nextSolution();
            String tblData[] = {qsol.get("x").asNode().getLocalName()};
            tblModel.addRow(tblData);
        }
    }
}

```

Επιλέγοντας από το μενού την λειτουργία **Search Instances**, εμφανίζεται το παρακάτω παράθυρο. Το JComboBox που βρίσκεται αριστερά, περιέχει όλες τις κλάσεις που εντοπίζονται στην οντολογία ενώ το JComboBox που βρίσκεται δεξιά, περιέχει τις ιδιότητες που χαρακτηρίζουν την προηγουμένως επιλεγθείσα κλάση. Δίνοντας μία συγκεκριμένη τιμή στο πεδίο Value, στον πίνακα επιστρέφονται όλα τα instances που ανήκουν στην κλάση που έχει επιλεγθεί και σύμφωνα με τις παραμέτρους που έχουν δοθεί από τον χρήστη.

The screenshot shows a web application titled "Hospital OWL". At the top, there are four buttons: "Retrieve Instances", "Insert Instances", "SPARQL Query", and "Search Instances". The "Search Instances" button is highlighted. Below the buttons, there are two dropdown menus. The first is labeled "Please Select a Class:" and has "Hospitalized" selected. The second is labeled "Please Select a Property:" and has "sharesRoomWith" selected. Below these, there is a text input field labeled "Please Provide a Value:" with the text "Mimi_Kollarou" entered. At the bottom, there is a table with the header "Instance(s)" and two rows of data: "Kostas_Kaoukis" and "Litsa_Kritsioni". To the right of the table is a "Submit" button.

Παρακάτω φαίνεται ο κώδικας που έχει υλοποιηθεί για την παραπάνω λειτουργία.

```
private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {  
    //get ontology class  
    final OntClass myClass = m.getOntClass(base + jComboBox5.getSelectedItem().toString());  
    //iterator that contains all the instances of the ontology class  
    ExtendedIterator instances = myClass.listInstances();  
    //create the property based on the input of the user  
    OntProperty property = m.createOntProperty(base + jComboBox6.getSelectedItem().toString());  
    //initialize table  
    DefaultTableModel tblModel4 = (DefaultTableModel) jTable4.getModel();  
    tblModel4.setRowCount(0);  
  
    String propertyValue = base + jTextField3.getText();  
    //iterate  
    while (instances.hasNext()) {  
        //get the instance  
        OntResource ori = (OntResource) instances.next();  
        try {  
            //check if it is not empty  
            if (ori.getPropertyValue(property) != null) {  
                //if the instance's value and the value that the user has given is equal  
                if (propertyValue.equals(ori.getPropertyValue(property).toString())) {  
                    //add new row to the table  
                    String tbData[] = {ori.getLocalName()};  
                    tblModel4.addRow(tbData);  
                }  
            }  
        }  
        catch (Exception ex) {  
            System.out.println("Something went wrong.");  
        }  
    }  
}
```

Γενικές Παρατηρήσεις:

1. Το βασικό αρχείο hospital.owl που βρίσκεται στο .zip αρχείο είναι αποθηκευμένο (Protege) σε format OWL/XML Syntax. Ωστόσο, το αρχείο hospital.owl που υπάρχει στον φάκελο Exercise7-GUI (εφαρμογή Netbeans) είναι αποθηκευμένο σε μορφή RDF/XML syntax, προκειμένου να χρησιμοποιηθούν οι απαραίτητες βιβλιοθήκες.
2. Κατά τη διάρκεια ανάπτυξης της οντολογίας δημιουργήθηκαν παραπάνω κλάσεις, ιδιότητες κ.λ.π. σε σύγκριση με εκείνες που αναφέρονται στο ερώτημα 1, προκειμένου να υλοποιηθούν και τα υπόλοιπα ζητούμενα.