ΑΝΑΠΑΡΑΣΤΑΣΗ ΓΝΩΣΗΣ ΣΤΟΝ ΠΑΓΚΟΣΜΙΟ ΙΣΤΟ

2020-2021

PROJECT

Ονοματεπώνυμο: Αδαμόπουλος Κωνσταντίνος

ΑΜ: 236270 (1043750)

Ερώτημα 1

A. Η οντολογία που δημιούργησα αφορά ένα κατάστημα οχημάτων όπου ο πελάτης μπορεί να πουλήσει ή να αγοράσει ένα όχημα της αρεσκείας του, και το κατάστημα κρατάει πληροφορίες σχετικά με τον αγοραστή/πωλητή αλλά και για τα χαρακτηριστικά των οχημάτων .

B. Η οντολογία αφορά διάφορες κατηγορίες οχημάτων που πουλά ένα αντίστοιχο κατάστημα. Κρατάει πληροφορίες για τα χαρακτηριστικά αυτών καθώς για τους πελάτες που έχουν αγοράσει από το κατάστημα ή ακόμα και από αυτούς που έχουν πουλήσει κάποιο όχημα στο κατάστημα.

Η οντολογία μπορεί να χρησιμοποιηθεί από τον υπεύθυνο του καταστήματος ώστε να έχει την πλήρη επίγνωση για το τι οχήματα είναι διαθέσιμα, ποια πουλάνε περισσότερο πόσα μεταχειρισμένα οχήματα υπάρχουν, να μπορεί οποιαδήποτε στιγμή να δει τα χαρακτηριστικά του τάδε οχήματος κ. α

Η οντολογία θα μπορεί να απαντά στις παρακάτω ερωτήσεις:

- **Ποιο όχημα είναι το γρηγορότερο**
- **Ποιος πελάτης έχει αγοράσει το τάδε όχημα .**
- **Ποιο όχημα έχει λιγότερους ρίπους .**
- **Ποιο όχημα είναι μεταχειρισμένο.**
- **Ποιος πελάτης έχει πουλήσει το όχημά του στο κατάστημα.**
- **Από ποια χώρα έχει κατασκευαστεί το τάδε όχημα .**
- **Ποια οχήματα είναι παρόμοια σε χαρακτηριστικά .**
- **Ποια οχήματα έχουν την ιδία ιπποδύναμη .**
- **Τον αριθμό των θέσεων που έχει το κάθε όχημα .**
- **Ποια χρονιά δημιουργήθηκε το μοντέλο .**
- **Τι καύσιμα καταναλώνει το όχημα.**
- **Ποια οχήματα καταναλώνουν ένα συγκεκριμένο καύσιμο .**
- **Τι προϊόν μπορεί να μεταφέρει και αν .**

- **Ποια είναι τα μεγάλα φορτηγά και ποια τα μικρά .**
- **Ποια είναι η τιμή του συγκεκριμένου οχήματος .**
- **Ποιο είναι το όνομα του οχήματος**
- **Ποιο είναι το όνομα του πελάτη το email του η ηλικία του και το φύλο του.**
- **Ποια είναι τα οχήματα που έχουν κατασκευαστεί ανά χώρα .**
- **Ποιος πελάτης έχει κάνει την μεγαλύτερη αγορά στο κατάστημα .**

Αν εφαρμοστεί κάποιος μηχανισμός συμπερασμού τότε θα έχουμε επιπλέον πληροφορία όπως να μπορούμε να δούμε ότι έχει αγοράσει ο πελάτης από το κατάστημα, ποιο όχημα είναι το γρηγορότερο από όλα , σε ποιον πελάτη ανήκουν τα οχήματα(μεταχειρισμένα) που έχουμε αγοράσει καθώς και ποιο είναι το φιλικότερο προς το περιβάλλον όχημα , την κατηγορία του οχήματος, το μέγεθος του οχήματος(μόνο για φορτηγά),τα οχήματα που έχουν κατασκευαστεί από την τάδε χώρα .

Η παραπάνω οντολογία θα μπορούσε να χρησιμοποιηθεί σε μια εφαρμογή για ένα κατάστημα που πουλάει/αγοράζει μοτοσυκλέτες, αμάξια και φορτηγά ή υποκατηγορίες αυτών. Ο χρήστης θα μπορεί μέσω της εφαρμογής να βλέπει τι υπάρχει διαθέσιμο από τα παραπάνω οχήματα, να έχει ιστορικό πελατών και να μπορεί να τι έχει αγοράσει ο κάθε πελάτης, να μπορεί να δει ποια από τα οχήματα που διαθέτει το κατάστημα είναι μεταχειρισμένα, καθώς και να έχει άμεση πρόσβαση στα χαρακτηριστικά του κάθε οχήματος. Θα μπορεί να προσθέσει νέα οχήματα ή ακόμα και να επεξεργαστεί κάποιο από τα ήδη υπάρχοντα .

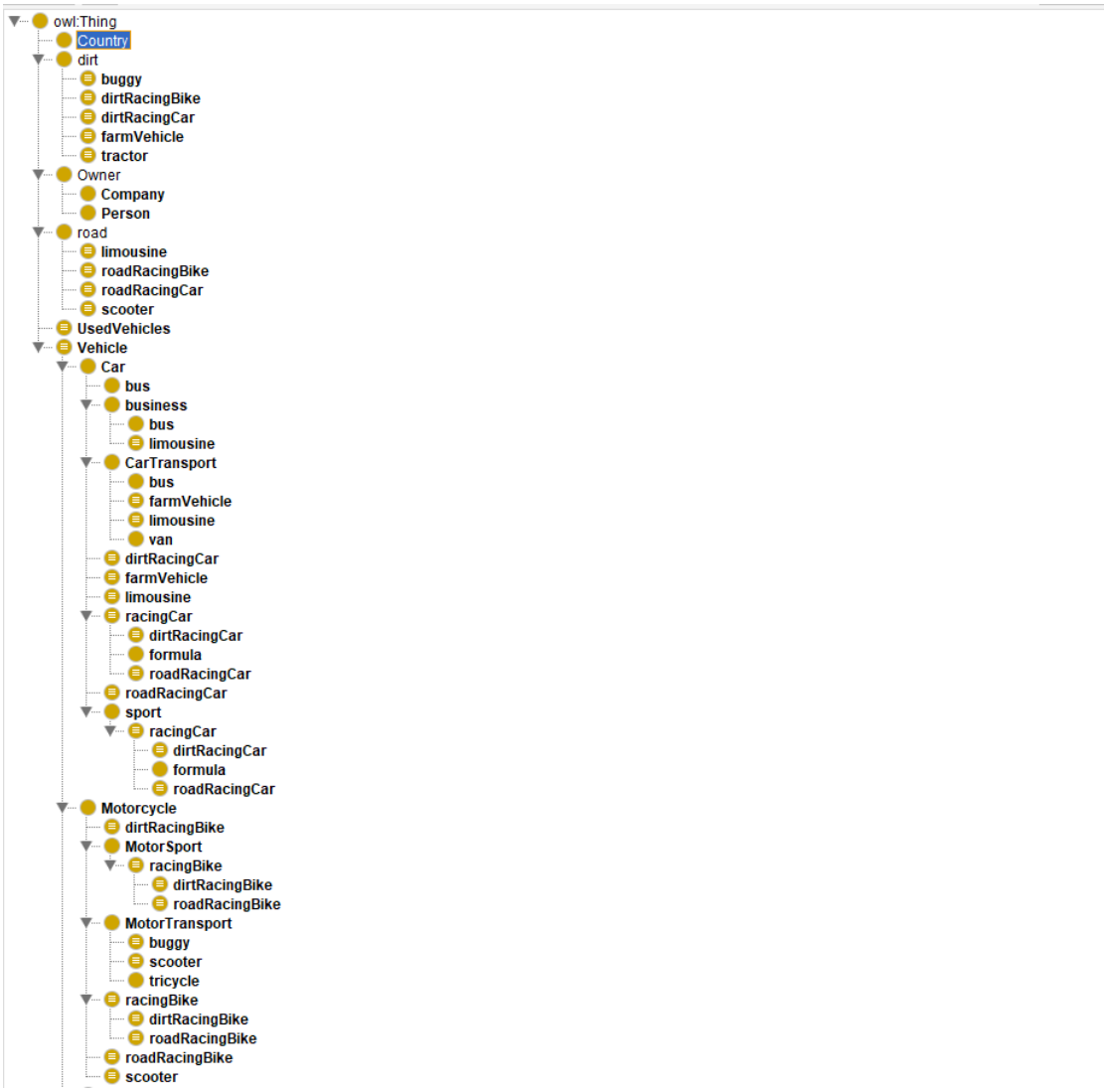
Γ. Οι κλάσεις που περιέχει η οντολογία είναι:

- **Country :** Κλάση η οποία περιέχει τις χώρες παρασκευής των οχημάτων .
- **dirt :** Κλάση η οποία περιλαμβάνει τους τύπους οχημάτων που είναι κατάλληλα για οδήγηση σε χωμάτινους δρόμους .
- **Owner:** Κλάση η οποία περιέχει τον ιδιοκτήτη του οχήματος/πελάτη .
- **Company :** Υπο-Κλάση της Owner η οποία περιέχει τις εταιρίες ιδιοκτήτες .
- **Person :** Υπο-Κλάση της Owner η οποία περιέχει τα άτομα ιδιοκτήτες-πωλητές .
- **road :** Κλάση η οποία περιλαμβάνει τους τύπους οχημάτων που είναι κατάλληλα για οδήγηση σε ασφαλτομένους δρόμους .
- **UsedVehicle :** Κλάση η οποία περιέχει τα μεταχειρισμένα οχήματα η οποία είναι ισοδύναμη με την κλάση Vehicle και εμφανίζεται μόνο αν το χρήστης «πουλήσει» το όχημα του στο κατάστημα (μεσώ της ιδιότητας soldBy)
- **Vehicle :** Κλάση η οποία περιέχει τους τύπους οχημάτων .
- **Car :** Υπο-κλάση της Vehicle η οποία περιέχει τα οχήματα που είναι αμάξια .
- **CarTransport :** Υπο-κλάση της Car η οποία περιέχει τα επιβατικά «αμάξια» .
- **business :** Υπο-κλάση της Car η οποία περιέχει τα «επαγγελματικά αμάξια» .
- **sport :** Υπο-κλάση της Car η οποία περιέχει τα «sport αμάξια» .
- **bus :** Κλάση περιορισμού η οποία περιέχει τα λεωφορεία .
- **limousine :** Κλάση τομής των CarTransport, road, business η οποία περιέχει τις λιμουζίνες .
- **farmVehicle :** Κλάση τομής των CarTransport και dirt η οποία περιέχει τα αγροτικά οχήματα .
- **van :** Κλάση περιορισμού η οποία περιέχει τα οχήματα(αμάξια) τύπου Van .

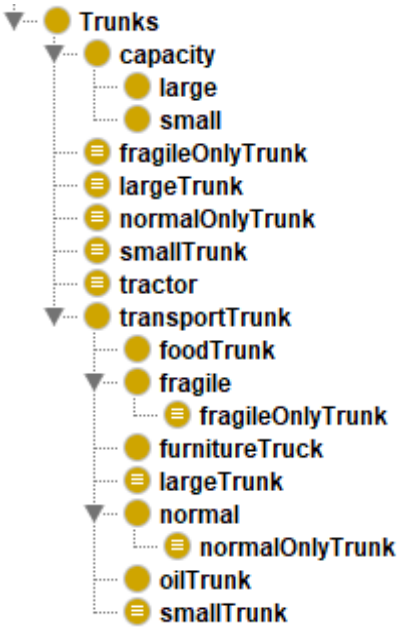
- **racingCar :** Κλάση τομής των Car και sport η οποία περιέχει τα αγωνιστικά αμάξια .
- **dirtRacingCar :** Κλάση τομής των dirt και racingCar η οποία περιέχει τα αγωνιστικά αμάξια που μπορούν να αγωνιστούν μόνο σε χωμάτινους δρόμους .
- **formula :** Κλάση περιορισμού η οποία περιέχει τα πολύ γρήγορα μονοθέσια .
- **roadRacingCar :** Κλάση τομής των racingCar και road η οποία περιέχει τα αγωνιστικά αμάξια που μπορούν να αγωνιστούν μόνο σε ασφαλτομένους δρόμους .
- **Motorcycle :** Υπο-κλάση της Vehicle η οποία περιέχει τα οχήματα που είναι μηχανές .
- **MotorSport :** Υπο-κλάση της Motorcycle η οποία περιέχει τις «sport μηχανές» .
- **MotorTransport :** Υπο-κλάση της Motorcycle η οποία περιέχει τις επιβατικές μηχανές .
- **racingBike :** Κλάση τομής των Motorcycle και MotorSport η οποία περιέχει τις αγωνιστικές μηχανές .
- **dirtRacingBike :** Κλάση τομής των dirt και racingBike η οποία περιέχει τις αγωνιστικές μηχανές που μπορούν να αγωνιστούν μόνο σε χωμάτινους δρόμους .
- **roadRacingBike :** Κλάση τομής των road και racingBike η οποία περιέχει τις αγωνιστικές μηχανές που μπορούν να αγωνιστούν μόνο σε ασφαλτομένους δρόμους .
- **buggy :** Κλάση τομής των MotorSport και dirt η οποία περιέχει τις «μηχανές» τύπου buggy .
- **tricycle :** Κλάση περιορισμού η οποία περιέχει τις τρικυκλες μηχανές .
- **Trunks :** Υπο-κλάση της Vehicle η οποία περιέχει τα οχήματα που είναι φορτηγά .
- **capacity :** Υπο-κλάση της Trunks η οποία καθορίζει την χωρητικότητα του φορτηγού .
- **large :** Υπο-κλάση της Capacity η οποία δηλώνει την μεγάλη χωρητικότητα .
- **small :** Υπο-κλάση της Capacity η οποία δηλώνει την μικρή χωρητικότητα .
- **transportTrunk :** Υπο-κλάση της Trunks η οποία περιέχει τα μεταφορικά φορτηγά .
- **smallTrunk :** Κλάση τομής των transportTruck και του συμπληρώματος της large και περιέχει τα μικρά σε χωρητικότητα φορτηγά .
- **largeTrunk :** Κλάση τομής των transportTruck και του συμπληρώματος της small και περιέχει τα μεγάλα σε χωρητικότητα φορτηγά .
- **foodTrunk :** Κλάση περιορισμού η οποία περιέχει τα φορτηγά που μεταφέρουν τρόφιμα .
- **furnitureTrunk :** Κλάση περιορισμού η οποία περιέχει τα φορτηγά που μεταφέρουν έπιπλα .
- **oilTrunk :** Κλάση περιορισμού η οποία περιέχει τα φορτηγά που μεταφέρουν πετρέλαιο .
- **fragile :** Κλάση η οποία περιέχει τα φορτηγά που ειδικεύονται στην μεταφορά εύθραυστων αντικειμένων .
- **fragileOnlyTrunk :** Κλάση η οποία περιέχει τα φορτηγά που μπορούν να μεταφέρουν εύθραυστα αντικείμενα και έχουν μικρή ή μεγάλη χωρητικότητα .
- **normal :** Κλάση η οποία περιέχει τα φορτηγά που μεταφέρουν κανονικά αντικείμενα .
- **normalOnlyTrunk :** Κλάση η οποία περιέχει τα φορτηγά που μπορούν να μεταφέρουν συνηθισμένα αντικείμενα και έχουν μικρή ή μεγάλη χωρητικότητα .

- **tractor :** Κλάση η οποία είναι συνδυασμός τομής και περιορισμού και περιέχει τα οχήματα τύπου τρακτέρ .
- **scooter :** Κλάση η οποία είναι συνδυασμός τομής και περιορισμού και περιέχει τα οχήματα τύπου σκούτερ τα οποία λειτουργούν με ηλεκτρισμό και είναι χαμηλού κυβισμού .

Η ιεραρχία των κλάσεων πλήρως ανοιγμένη φαίνεται στην παρακάτω εικόνα :



Συνέχεια της από πάνω εικόνας



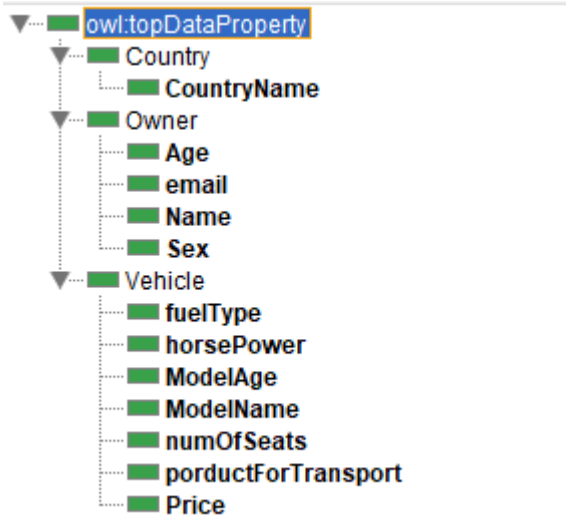
Δ. Παρακάτω παρουσιάζονται οι ιδιότητες τις παραπάνω οντολογίας. Ξεκινώντας με τις Data Type Properties.

Έχουμε:

- Την ιδιότητα Country η οποία περιέχει την υπο-ιδιότητα **CountryName(Functional)** η οποία έχει το όνομα της κάθε χώρας και έχει ως πεδίο ορισμού την κλάση Country και μπορεί να πάρει μόνο αλφαριθμητικές τιμές .
- Την ιδιότητα Owner η οποία περιέχει την υπο-ιδιότητα **Age(Functional)** που είναι η ηλικία του πελάτη και έχει ως πεδίο ορισμού την κλάση Person και μπορεί να λαβει μόνο θετικούς ακέραιους, την υπο-ιδιότητα **email** που είναι η ηλεκτρονική διεύθυνση του ιδιοκτήτη και έχει ως πεδίο ορισμού την κλάση Owner και μπορεί να πάρει μόνο αλφαριθμητικές τιμές. Την υπο-ιδιότητα **Name(Functional)** η οποία περιέχει το όνομα του ιδιοκτήτη και μπορεί να πάρει μόνο αλφαριθμητικές τιμές και την υπο-ιδιότητα **Sex(Functional)** η οποία περιέχει το φύλο του ιδιοκτήτη και έχει ως πεδίο ορισμού την κλάση Person και μπορεί να λαβει μόνο τις τιμές “Male” και “Female” .
- Την ιδιότητα Vehicle η οποία περιέχει την υπο-ιδιότητα **fuelType** η οποία αφορά το είδος καυσίμου που χρησιμοποιεί το συγκεκριμένο όχημα, το πεδίο ορισμού της είναι η κλάση Vehicle και μπορεί να πάρει μια από τις τιμές “Electrism”, ”Oil” και “Petrol” , την υπο-ιδιότητα **ModelAge(Functional)** η οποία αφορά την χρονολογία δημιουργίας του συγκεκριμένου οχήματος και έχει ως πεδίο ορισμού της την κλάση Vehicle και μπορεί να πάρει μόνο θετικούς ακέραιους, την υπο-ιδιότητα **ModelName(Functional)** η οποία αφορά την ονομασία του συγκεκριμένου οχήματος και έχει ως πεδίο ορισμού της την κλάση Vehicle και μπορεί να πάρει μόνο αλφαριθμητικές τιμές, την υπο-ιδιότητα **numOfSeats** η οποία έχει να κάνει με τον αριθμό των θέσεων στο όχημα έχει ως πεδίο ορισμού της την κλάση Vehicle και μπορεί να πάρει μόνο θετικούς ακέραιους, την υπο-ιδιότητα **productForTransport(Functional)** η οποία δείχνει το τύπο φορτίου που έχει το φορτηγό, έχει ως πεδίο ορισμού της την κλάση Trunks και μπορεί να πάρει μόνο τις τιμές “Cars”, ”Food”, “Furniture”, “Oil” , την υπο-ιδιότητα **Price(Functional)** η οποία

περιέχει την τιμή του συγκεκριμένου οχήματος και έχει πεδίο ορισμού την κλάση Vehicle και δέχεται μόνο τιμές τύπου double και τέλος την ιδιότητα **horsePower(Functional)** η οποία περιέχει την ιπποδύναμη του συγκεκριμένου οχήματος και έχει πεδίο ορισμού την κλάση Vehicle και δέχεται μόνο τιμές τύπου double .

Το σχηματικό διάγραμμα των ιδιοτήτων είναι :

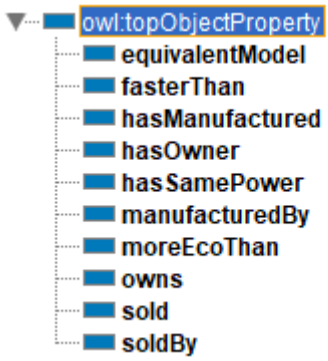


Έπειτα συνεχίζουμε με τις Object Type Properties οι οποίες είναι :

- Η **equivalentModel** η οποία είναι συμμετρική και χρησιμοποιείται όταν θέλουμε να δηλώσουμε ότι δύο οχήματα έχουν παρόμοια χαρακτηριστικά(π.χ τιμή) . Το πεδίου ορισμού της είναι η κλάση Vehicle καθώς και το πεδίο τιμών της .
- Η **fasterThan** η οποία είναι μεταβατική και χρησιμοποιείται για να δηλώσουμε ότι ένα όχημα είναι γρηγορότερο από ένα άλλο. Το πεδίου ορισμού της είναι η κλάση Vehicle καθώς και το πεδίο τιμών της .
- Η **hasOwner** η οποία είναι συναρτησιακή και αντίστροφη της owns και από αυτήν μπορούμε να δούμε τον ιδιοκτήτη του οχήματος . Το πεδίου ορισμού της είναι η κλάση Vehicle και το πεδίο τιμών της η κλάση Owner .
- Η **hasSamePower** η οποία είναι συμμετρική και χρησιμοποιείται όταν θέλουμε να δηλώσουμε ότι δύο οχήματα έχουν ίδια ιπποδύναμη . Το πεδίου ορισμού της είναι η κλάση Vehicle καθώς και το πεδίο τιμών της .
- Η **manufacturedBy** η οποία είναι συναρτησιακή και δηλώνει την χώρα κατασκευής του οχήματος. Το πεδίου ορισμού της είναι η κλάση Vehicle και το πεδίο τιμών της η κλάση Country .
- Η **moreEcoThan** η οποία είναι μεταβατική και χρησιμοποιείται για να δηλώσουμε ότι ένα όχημα είναι φιλικότερο προς το περιβάλλον από ένα άλλο. Το πεδίου ορισμού της είναι η κλάση Vehicle καθώς και το πεδίο τιμών της .
- Η **owns** η οποία είναι αντί-συναρτησιακή και αντίστροφη της hasOwner και από αυτήν μπορούμε να δούμε τα οχήματα που έχει στην κατοχή του ο ιδιοκτήτης . Το πεδίου ορισμού της είναι η κλάση Owner και το πεδίο τιμών της η κλάση Vehicle .

- Η **sold** η οποία είναι αντίστροφη της soldBy , αντί-συναρτησιακή και αφορά την πώληση ενός οχήματος από τον πελάτη σε εμάς(π.χ κατάστημα). Το πεδίου ορισμού της είναι η κλάση Person και το πεδίο τιμών της η κλάση Vehicle .
- Η **soldBy** η οποία είναι αντίστροφη της sold και συναρτησιακή και μας δείχνει από ποιον πελάτη αγοράσαμε το τάδε όχημα. Το πεδίου ορισμού της είναι η κλάση Vehicle και το πεδίο τιμών της η κλάση Person .
- Η **hasManufactured** η οποία είναι αντίστροφη της manufacturedBy και μας δείχνει τα οχήματα που έχουν κατασκευαστεί από μια συγκεκριμένη χώρα. Το πεδίου ορισμού της είναι η κλάση Country και το πεδίο τιμών της η κλάση Vehicle .

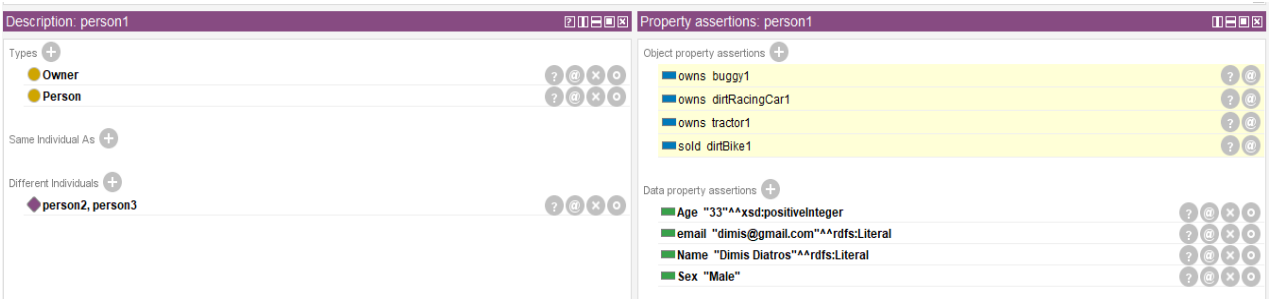
Το σχηματικό διάγραμμα των ιδιοτήτων είναι :



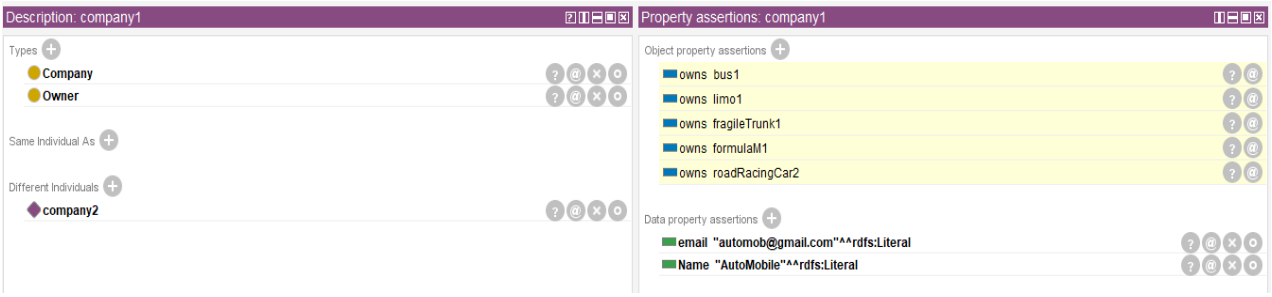
Ε.

Παρακάτω παρουσιάζονται κάποια στιγμιότυπα της παραπάνω οντολογίας :

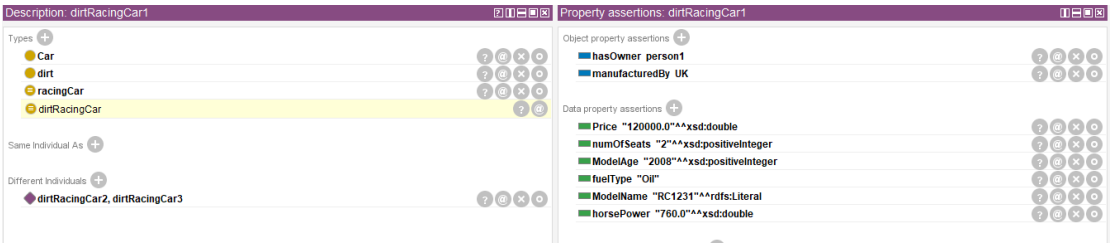
Ένα στιγμιότυπο τις κλάσης **Person**:



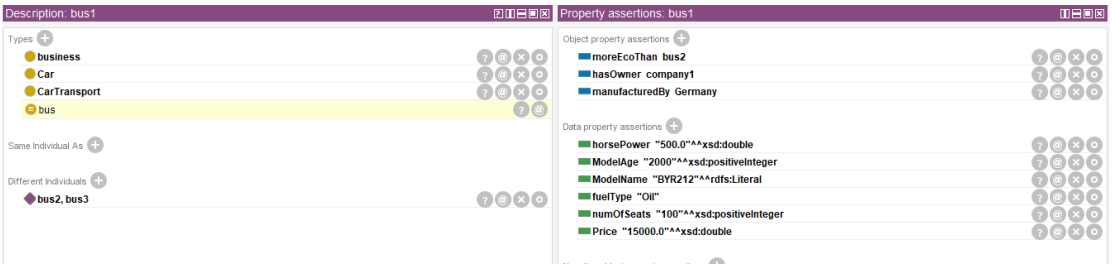
Ένα στιγμιότυπο της κλάσης **Company**:



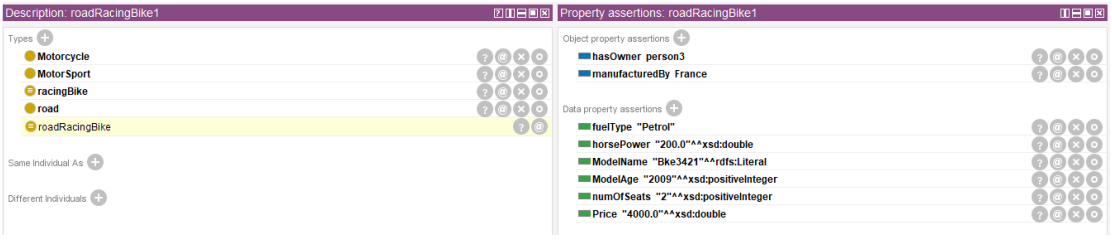
Ένα στιγμιότυπο της κλάσης **dirtRacingCar** :



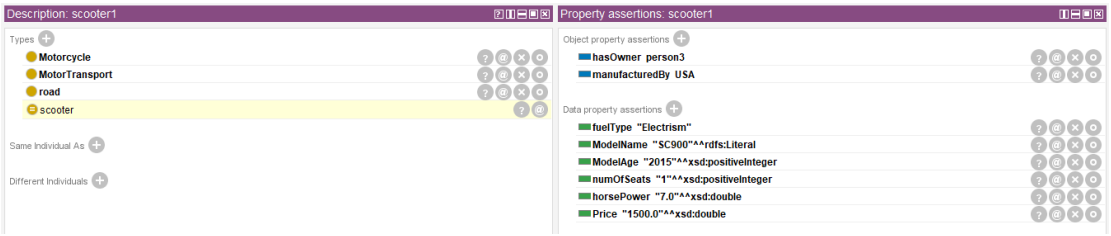
Ένα στιγμιότυπο της κλάσης **bus** :



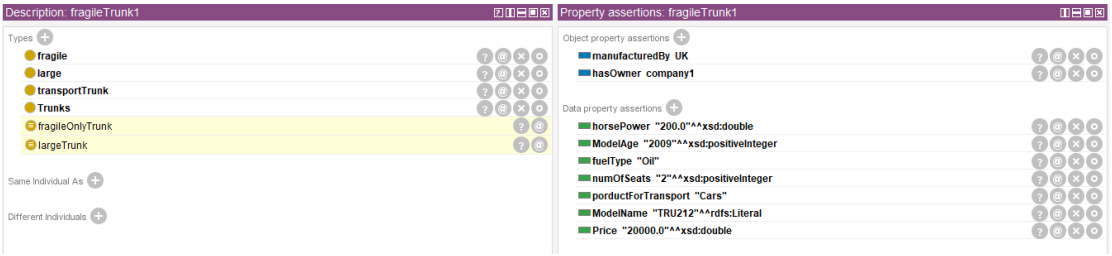
Ένα στιγμιότυπο της κλάσης **roadRacingBike** :



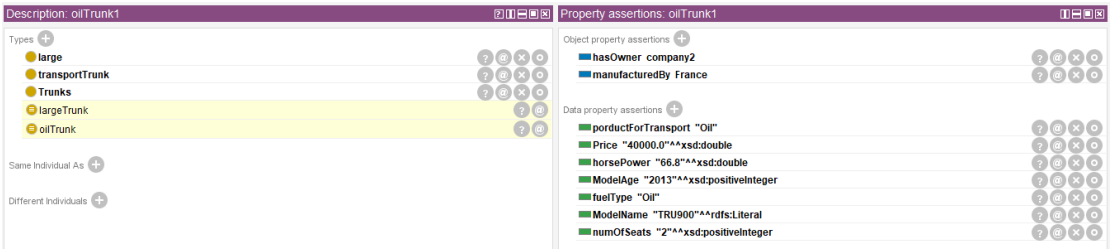
Ένα στιγμιότυπο της κλάσης **scooter** :



Ένα στιγμιότυπο της κλάσης **fragileOnlyTrunk** και της **largeTrunk** :



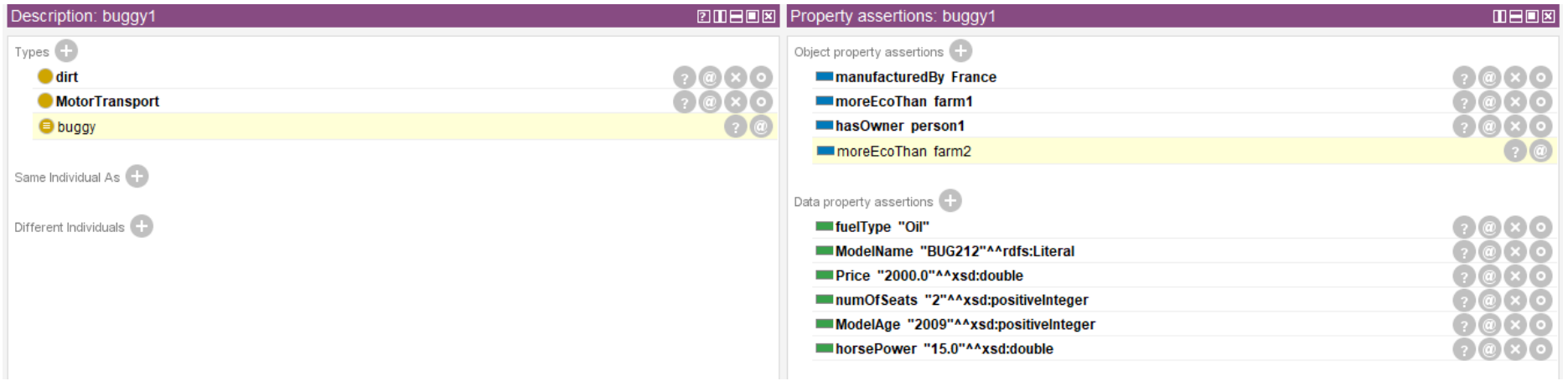
Ένα στιγμιότυπο της κλάσης **oilTrunk** :



Ερώτημα 3

Παρακάτω παρουσιάζονται περιπτώσεις όπου μέσω του μηχανισμού συμπερασμού παράγεται επιπλέον γνώση για την οντολογία .

1.



Από το στιγμιότυπο της κλάσης buggy παίρνουμε τις εξής χρήσιμες πληροφορίες :

1. Ότι το στιγμιότυπο/όχημα είναι τύπου buggy και
2. ότι είναι πιο οικολογικό από το όχημα farm2 .

Η πρώτη πληροφορία που λαμβάνουμε εκφράζεται σε φυσική γλώσσα ως: Το buggy1 είναι στιγμιότυπο τύπου buggy .Και ως τριπλέτα :

Subject	Predicate	Object
http://www.mydomain.com/vehicles/buggy	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
http://www.mydomain.com/vehicles/buggy1	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/buggy

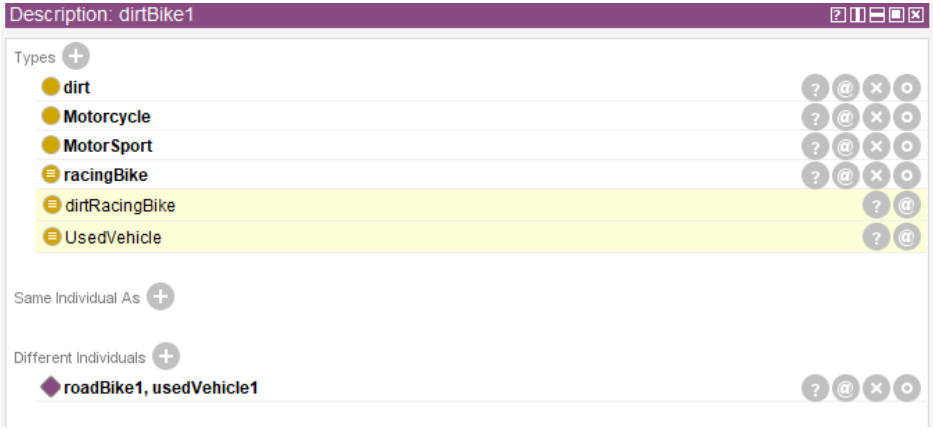
Η παραπάνω γνώση προέκυψε διότι ο reasoner “βλέπει” ότι το στιγμιότυπο είναι τύπου dirt και MotorTransport και ότι η τομή αυτών των δύο είναι ίση με το τύπο buggy .

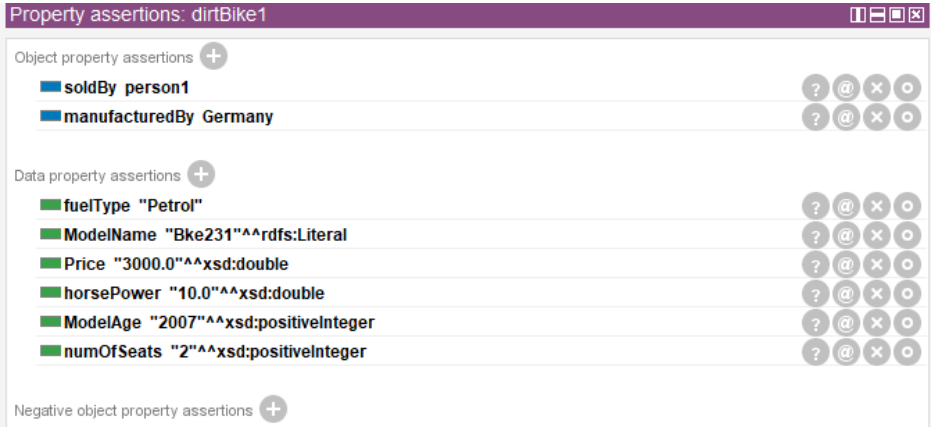
Και η δεύτερη πληροφορία εκφράζεται σε φυσική γλώσσα ως : Το buggy1 είναι πιο οικολογικό από το farm2 . Και ως τριπλέτα:

Subject	Predicate	Object
http://www.mydomain.com/vehicles/moreEcoThan	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Object Property
http://www.mydomain.com/vehicles/moreEcoThan	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Transitive Property
http://www.mydomain.com/vehicles/buggy1	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/buggy
http://www.mydomain.com/vehicles/farm1	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/farmVehicle
http://www.mydomain.com/vehicles/buggy1	http://www.mydomain.com/vehicles/moreEcoThan	http://www.mydomain.com/vehicles/farm2

Η παραπάνω γνώση προέκυψε διότι ο reasoner “βλέπει” ότι έχει δηλωθεί το ότι το buggy1 είναι πιο οικολογικό(moreEcoThan) από το farm1 και ότι το farm1 είναι πιο οικολογικό(moreEcoThan) από το farm2 και γνωρίζοντας ότι η ιδιότητα αυτή είναι μεταβατική κάνει τον παραπάνω συμπερασμό .

2.





Από το στιγμιότυπο της κλάσης dirtRacingBike παίρνουμε τις εξής πληροφορίες :

1. Ότι το στιγμιότυπο/όχημα είναι τύπου dirtRacingBike και
2. Ότι είναι και μεταχειρισμένο όχημα το οποίο προκύπτει από την ιδιότητα soldBy .

Η πρώτη πληροφορία που λαμβάνουμε εκφράζεται σε φυσική γλώσσα ως: Το dirtBike1 είναι στιγμιότυπο τύπου dirtRacingBike .Και ως τριπλέτα :

Subject	Predicate	Object
http://www.mydomain.com/vehicles/ dirtRacingBike	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
http://www.mydomain.com/vehicles/dirtBike1	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/dirtRacingBike

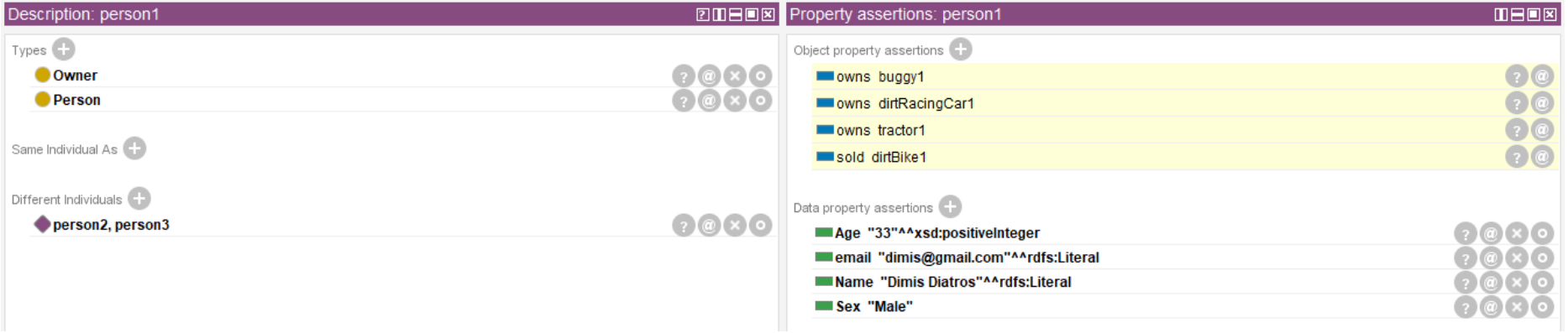
Η παραπάνω γνώση προέκυψε διότι ο reasoner “βλέπει” ότι το στιγμιότυπο είναι τύπου dirt και racingBike και ότι η τομή αυτών των δύο είναι ίση με το τύπο dirtRacingBike .

Και η δεύτερη πληροφορία εκφράζεται σε φυσική γλώσσα ως: Το dirtBike1 είναι μεταχειρισμένο .Και ως τριπλέτα :

Subject	Predicate	Object
http://www.mydomain.com/vehicles/usedVehicle	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
http://www.mydomain.com/vehicles/dirtBike1	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/usedVehicle

Η παραπάνω γνώση προέκυψε διότι ο reasoner “βλέπει” ότι το στιγμιότυπο dirtBike1 πουλήθηκε από τον person1 μέσω της ιδιότητας soldBy μετά βλέπει ότι η ιδιότητα soldBy έχει πεδίο ορισμού την κλάση Vehicle και πεδίο τιμών την κλάση Person καθώς και ότι η κλάση UsedVehicle είναι ίση με την τομή της κλάσης Vehicle και τον περιορισμό soldBy some Person και εφόσον βλέπει ότι η ιδιότητα soldBy έχει τιμή κλάσης Person και το στιγμιότυπο είναι τύπος οχήματος τότε συμπεραίνει ότι ανήκει στην κλάση UsedVehicle .

3.



Από το στιγμιότυπο της κλάσης Person μέσω του μηχανισμού συμπερασμού το τι οχήματα έχει στην κατοχή του ο «Person1»(ιδιότητα owns) αλλά και τι μας έχει πουλήσει(ιδιότητα sold) .
Η ιδιότητα owns εκφράζεται σε φυσική γλώσσα ως: Ο person1 έχει στην κατοχή του το buggy1 . Και ως τριπλέτα:

Subject	Predicate	Object
http://www.mydomain.com/vehicles/owns	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#ObjectProperty
http://www.mydomain.com/vehicles/person1	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/Person
http://www.mydomain.com/vehicles/buggy1	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/buggy
http://www.mydomain.com/vehicles/person1	http://www.mydomain.com/vehicles/owns	http://www.mydomain.com/vehicles/buggy1

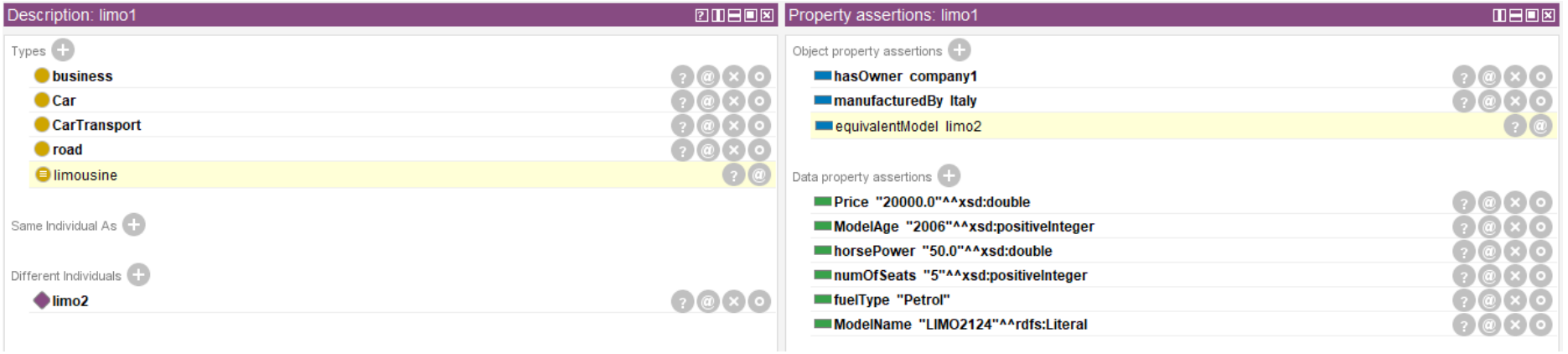
Ο reasoner συμπεραίνει την παραπάνω πληροφορία διότι βλέπει ότι η ιδιότητα hasOwner είναι αντίστροφη της owns και ότι το buggy1 έχει σαν ιδιοκτήτη (ιδιότητα hasOwner) το στιγμιότυπο person1 .

Και η ιδιότητα sold εκφράζεται σε φυσική γλώσσα ως: Ο person1 πούλησε την dirtBike1 .Και ως τριπλέτα:

Subject	Predicate	Object
http://www.mydomain.com/vehicles/sold	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#ObjectProperty
http://www.mydomain.com/vehicles/person1	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/Person
http://www.mydomain.com/vehicles/dirtBike1	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/dirtRacingBike
http://www.mydomain.com/vehicles/person1	http://www.mydomain.com/vehicles/sold	http://www.mydomain.com/vehicles/dirtBike1

Ο reasoner συμπεραίνει την παραπάνω πληροφορία διότι βλέπει ότι η ιδιότητα sold είναι αντίστροφη της soldBy και ότι το dirtBike1 πουλήθηκε (soldBy) από το στιγμιότυπο person1 .

4.



Από το στιγμιότυπο της κλάσης limousine παίρνουμε τις εξής πληροφορίες :

1. Ότι το στιγμιότυπο/όχημα είναι τύπου limousine και
2. Ότι έχει παρόμοια χαρακτηριστικά με το στιγμιότυπο «limo2».

Η πρώτη πληροφορία που λαμβάνουμε εκφράζεται σε φυσική γλώσσα ως: Η limo1 είναι στιγμιότυπο τύπου limousine .Και ως τριπλέτα :

Subject	Predicate	Object
http://www.mydomain.com/vehicles/limousine	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
http://www.mydomain.com/vehicles/limo1	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/limousine

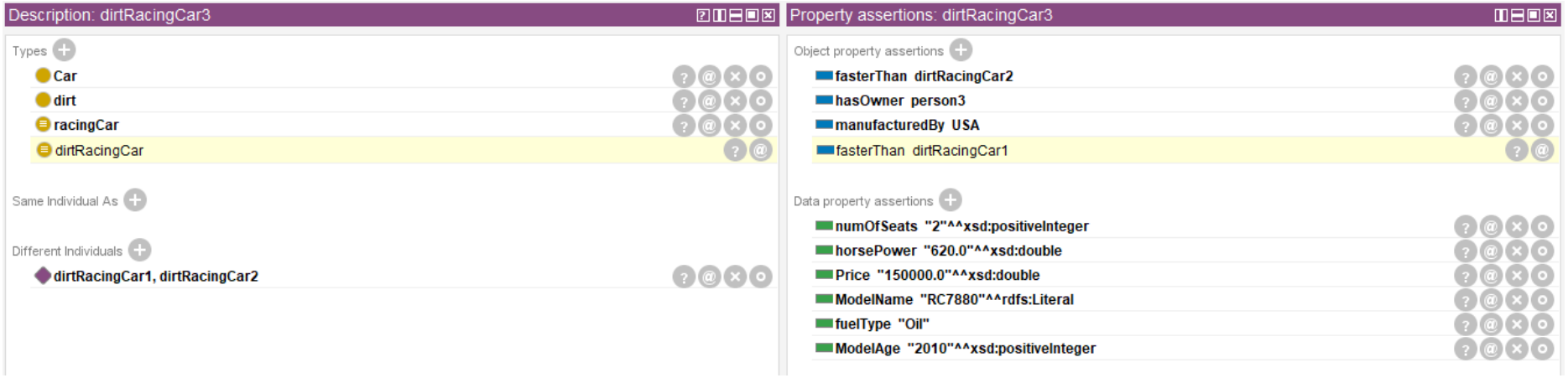
Η παραπάνω γνώση προέκυψε διότι ο reasoner “βλέπει” ότι το στιγμιότυπο είναι τύπου business και carTransport και road και ότι η τομή αυτών των τριών είναι ίση με το τύπο limousine .

Και η δεύτερη πληροφορία εκφράζεται σε φυσική γλώσσα ως: Η limo1 έχει παρόμοια χαρακτηριστικά με την limo2 .Και ως τριπλέτα:

Subject	Predicate	Object
http://www.mydomain.com/vehicles/equivalentModel	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#ObjectProperty
http://www.mydomain.com/vehicles/equivalentModel	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#SymmetricProperty
http://www.mydomain.com/vehicles/limo1	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/limousine
http://www.mydomain.com/vehicles/limo2	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/limousine
http://www.mydomain.com/vehicles/limo1	http://www.mydomain.com/vehicles/equivalentModel	http://www.mydomain.com/vehicles/limo2

Ο reasoner συμπεραίνει την παραπάνω πληροφορία διότι βλέπει ότι η ιδιότητα equivalentModel είναι συμμετρική και ότι το στιγμιότυπο limo2 έχει παρόμοια χαρακτηριστικά(equivalentModel) με το στιγμιότυπο limo1 .

5.



Από το στιγμιότυπο της κλάσης dirtRacingCar παίρνουμε τις εξής πληροφορίες :

1. Ότι το στιγμιότυπο/όχημα είναι τύπου dirtRacingCar και
2. Ότι είναι και ότι είναι πιο γρήγορο από το «dirtRacingCar1» το οποίο συμπεραίνει ο μηχανισμός συμπερασμού αφού έχουμε δηλώσει ότι το στιγμιότυπο («dirtRacingCar3») είναι πιο γρήγορο από το «dirtRacingCar2» και ότι το «dirtRacingCar2» είναι πιο γρήγορο από το «dirtRacingCar1» .

Η πρώτη πληροφορία που λαμβάνουμε εκφράζεται σε φυσική γλώσσα ως: Το dirtRacingCar3 είναι στιγμιότυπο τύπου dirtRacingCar .Και ως τριπλέτα :

Subject	Predicate	Object
http://www.mydomain.com/vehicles/dirtRacingCar	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class

http://www.mydomain.com/vehicles/dirtRacingCar3	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/dirtRacingCar

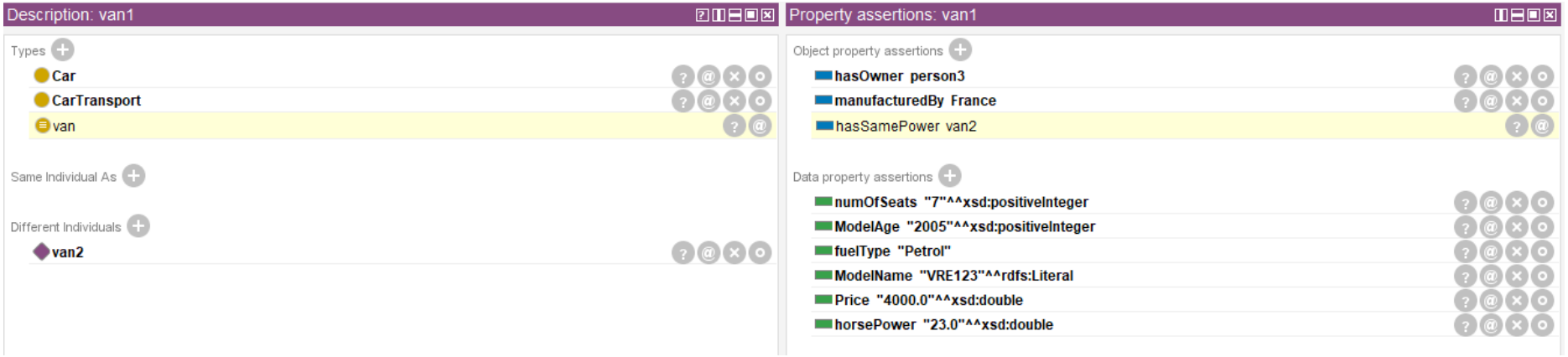
Η παραπάνω γνώση προέκυψε διότι ο reasoner “βλέπει” ότι το στιγμιότυπο dirtRacingCar3 είναι τύπου dirt και racingCar και ότι η τομή αυτών των τριών είναι ίση με το τύπο dirtRacingCar .

Και η δεύτερη πληροφορία εκφράζεται σε φυσική γλώσσα ως: Το dirtRacingCar3 είναι πιο γρήγορο από το dirtRacingCar1. Και ως τριπλέτα:

Subject	Predicate	Object
http://www.mydomain.com/vehicles/fasterThan	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#ObjectProperty
http://www.mydomain.com/vehicles/fasterThan	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Transitive Property
http://www.mydomain.com/vehicles/dirtRacingCar3	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/ dirtRacingCar
http://www.mydomain.com/vehicles/ dirtRacingCar1	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/ dirtRacingCar
http://www.mydomain.com/vehicles/ dirtRacingCar3	http://www.mydomain.com/vehicles/ fasterThan	http://www.mydomain.com/vehicles/ dirtRacingCar1

Η παραπάνω γνώση προέκυψε διότι ο reasoner “βλέπει” ότι έχει δηλωθεί το ότι το dirtRacingCar3 είναι πιο γρήγορο(fasterThan) από το dirtRacingCar2 και ότι το dirtRacingCar2 είναι πιο γρήγορο(fasterThan) από το dirtRacingCar1 και γνωρίζοντας ότι η ιδιότητα αυτή είναι μεταβατική κάνει τον παραπάνω συμπερασμό .

6.



1. Από το στιγμιότυπο της κλάσης van εξάγεται η επιπλέον πληροφορία ότι το στιγμιότυπο van1 έχει την ίδια «δύναμη» με το στιγμιότυπο van2 .Η παραπάνω πληροφορία σε φυσική γλώσσα: το στιγμιότυπο van1 έχει την ίδια «δύναμη» με το στιγμιότυπο van2 . Και σε τριπλέτα :

Subject	Predicate	Object
http://www.mydomain.com/vehicles/hasSamePower	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#ObjectProperty
http://www.mydomain.com/vehicles/hasSamePower	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#SymmetricProperty
http://www.mydomain.com/vehicles/van1	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/van
http://www.mydomain.com/vehicles/van2	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/van

http://www.mydomain.com/vehicles/van2	http://www.mydomain.com/vehicles/hasSamePower	http://www.mydomain.com/vehicles/van1

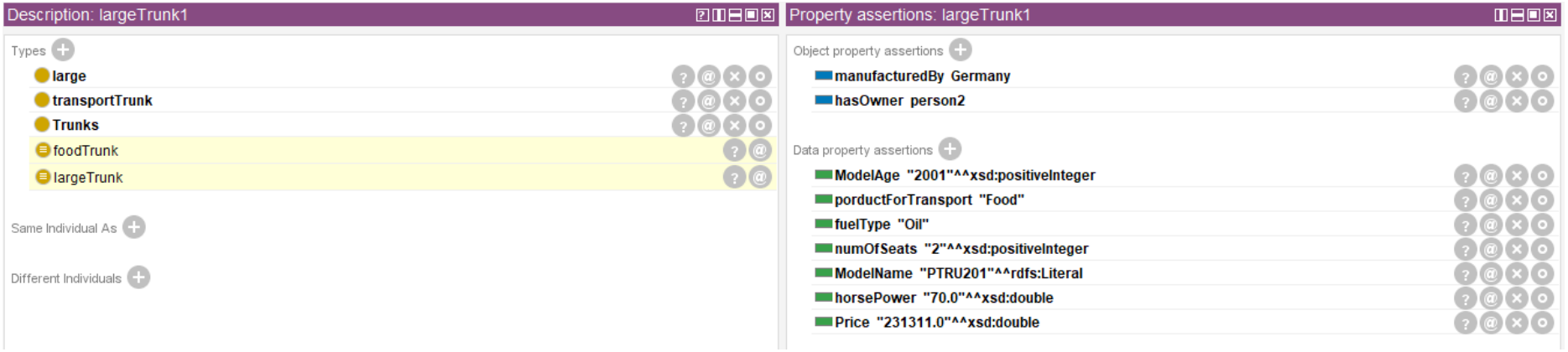
Ο reasoner συμπεραίνει την παραπάνω πληροφορία διότι βλέπει ότι η ιδιότητα hasSamePower είναι συμμετρική και ότι το στιγμιότυπο van2 έχει ίδια «δύναμη»(hasSamePower) με το στιγμιότυπο van1 .

2. Και η δεύτερη πληροφορία που λαμβάνουμε εκφράζεται σε φυσική γλώσσα ως: Το van1 είναι στιγμιότυπο τύπου van .Και ως τριπλέτα :

Subject	Predicate	Object
http://www.mydomain.com/vehicles/van	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
http://www.mydomain.com/vehicles/van1	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/van

Η παραπάνω γνώση προέκυψε διότι ο reasoner “βλέπει” ότι το στιγμιότυπο van1 είναι τύπου CarTransport και ότι η DataType ιδιότητα numOfSeats(τιμή 7) είναι μεταξύ του 6 και του 10 .

7.



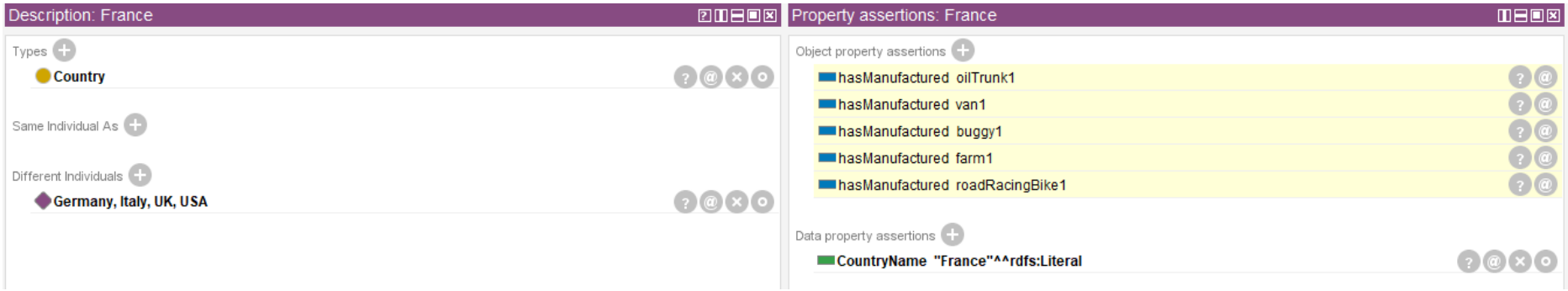
Από το στιγμιότυπο «largeTrunk1» βλέπουμε πως εξάγεται επιπλέον γνώση η οποία είναι ότι το φορτηγό είναι μεγάλης χωρητικότητας και ότι χρησιμοποιείται για μεταφορά τροφίμων το οποίο συμπεραίνει ο μηχανισμός διότι η ιδιότητα “productForTransport” έχει την τιμή “Food” .

Η παραπάνω πληροφορία σε φυσική γλώσσα: Το φορτηγό είναι για μεταφορά τροφίμων και έχει μεγάλη χωρητικότητα . Και σε τριπλέτα:

Subject	Predicate	Object
http://www.mydomain.com/vehicles/largeTrunk	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
http://www.mydomain.com/vehicles/foodTrunk	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
http://www.mydomain.com/vehicles/largeTrunk1	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/largeTrunk
http://www.mydomain.com/vehicles/largeTrunk1	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/foodTrunk

Ο reasoner συμπεραίνει ότι το στιγμιότυπο oilTrunk1 είναι μεγάλης χωρητικότητας (τύπου largeTrunk) διότι είναι τύπου large και transportTrunk και “βλέπει” ότι η τομή των δύο παραπάνω κλάσεων είναι ίση με την κλάση largeTrunk και για το ότι ανήκει και στην κλάση foodTrunk το συμπεραίνει διότι “βλέπει” ότι η ιδιότητα productForTransport έχει την τιμή Food .

8.

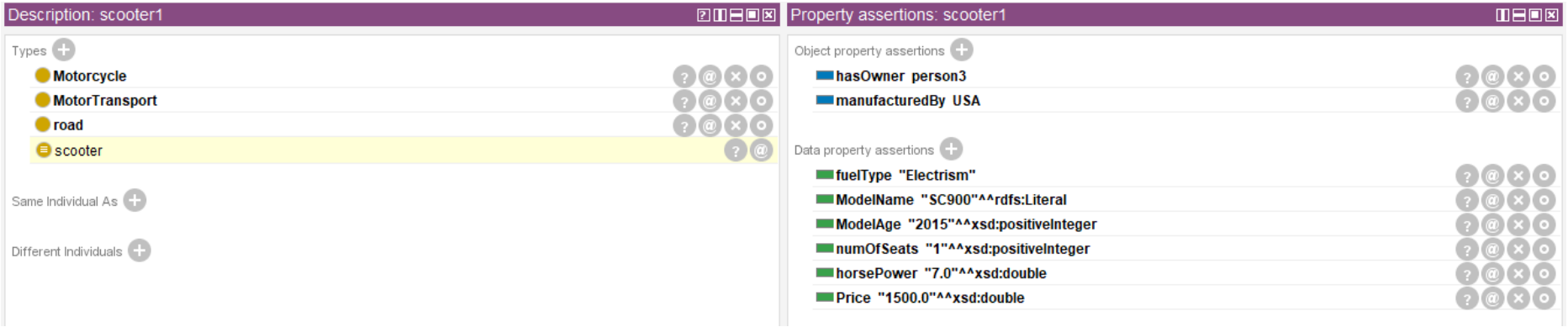


Από το στιγμιότυπο «France» και με την χρήση του μηχανισμού συμπερασμού βλέπουμε ποια οχήματα έχουν κατασκευαστεί από την συγκεκριμένη χώρα .Η παραπάνω πληροφορία σε φυσική γλώσσα: Ποια οχήματα έχουν κατασκευαστεί στην Γαλλία .Και σε τριπλέτα :

Subject	Predicate	Object
http://www.mydomain.com/vehicles/hasManufactured	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#ObjectProperty
http://www.mydomain.com/vehicles/oilTrunk1	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/oilTrunk
http://www.mydomain.com/vehicles/France	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/Country
http://www.mydomain.com/vehicles/France	http://www.mydomain.com/vehicles/hasManufactured	http://www.mydomain.com/vehicles/oilTrunk1

Ο reasoner συμπεραίνει την παραπάνω πληροφορία διότι βλέπει ότι η ιδιότητα hasManufactured είναι αντίστροφη της manufacturedBy και ότι το oilTrunk1 έχει κατασκευαστεί (manufacturedBy) στην Γαλλία(στιγμιότυπο της κλάσης Country) .

9.



Από το στιγμιότυπο «largeTrunk1» βλέπουμε πως εξάγεται επιπλέον γνώση η οποία μας δείχνει ότι το στιγμιότυπο scooter1 είναι τύπου scooter .Και ως τριπλέτα:

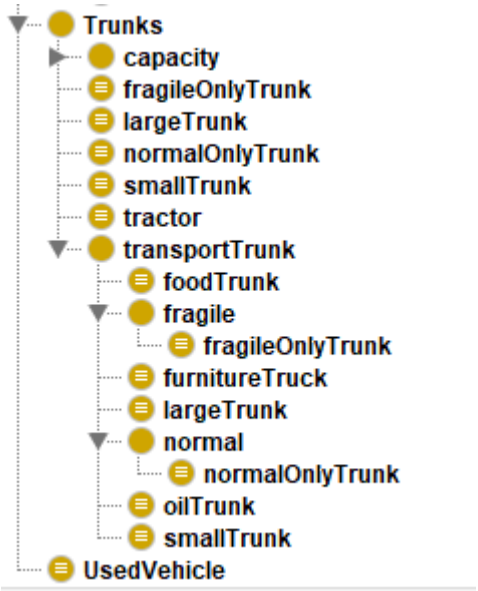
Subject	Predicate	Object
http://www.mydomain.com/vehicles/scooter	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
http://www.mydomain.com/vehicles/scooter1	https://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.mydomain.com/vehicles/scooter

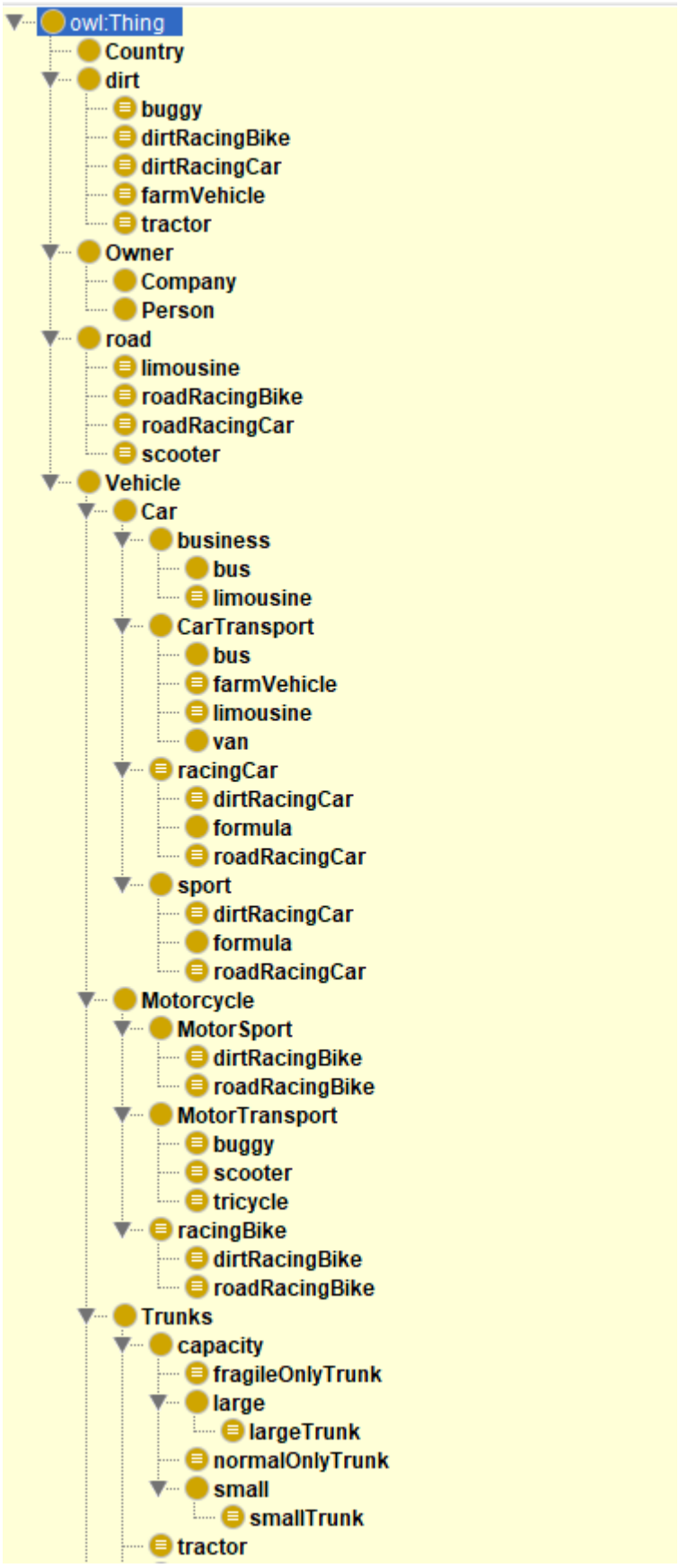
Η παραπάνω γνώση προέκυψε διότι ο reasoner “βλέπει” ότι το στιγμιότυπο scooter1 είναι τύπου MotorTransport, road και η ιδιότητα fuelType έχει την τιμή Electrism και ότι η τομή αυτών των τριών είναι ίση με το τύπο scooter .

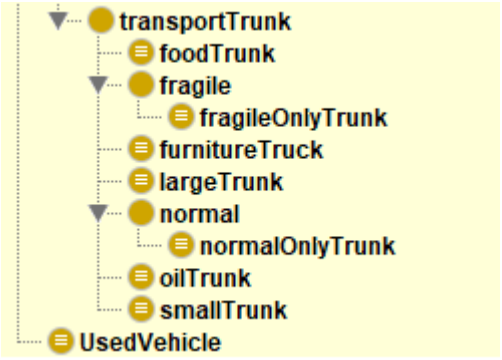
Ερώτημα 4

a. Παρακάτω παρουσιάζονται δύο screenshots για την ιεραρχία των κλάσεων όπως αυτές εισήχθησαν και αφότου εφαρμόστηκε σε αυτήν κάποιος μηχανισμός συμπερασμού .

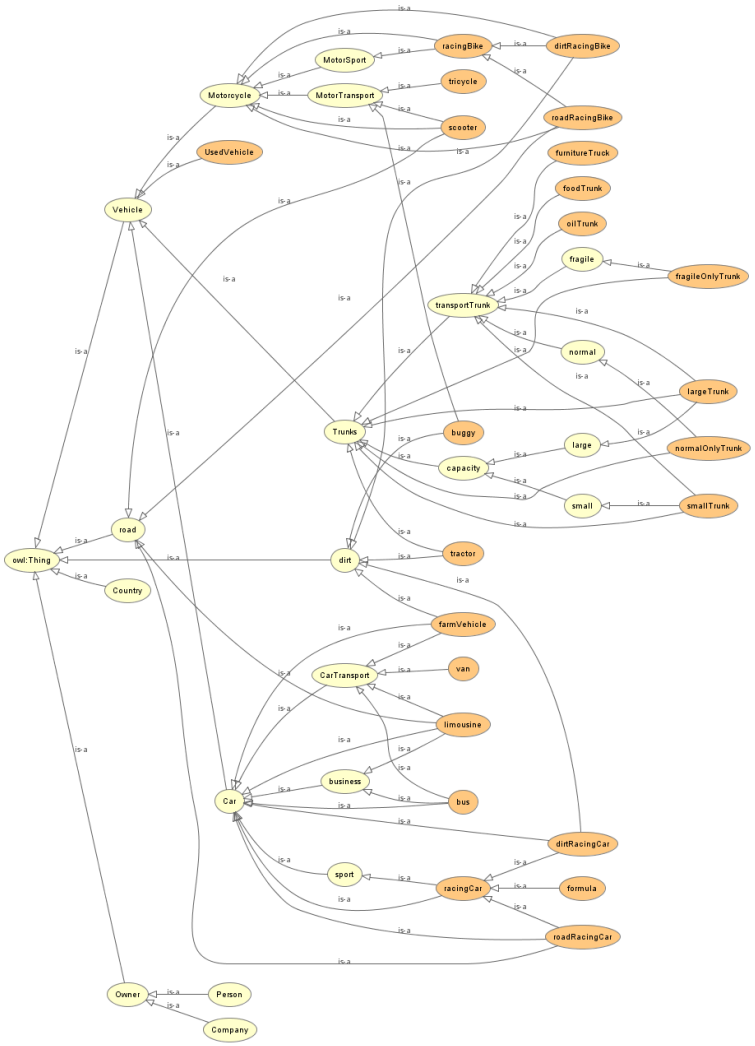

owl:Thing
Country
dirt
buggy
dirtRacingBike
dirtRacingCar
farmVehicle
tractor
Owner
Company
Person
road
limousine
roadRacingBike
roadRacingCar
scooter
Vehicle
Car
bus
business
bus
limousine
CarTransport
bus
farmVehicle
limousine
van
dirtRacingCar
farmVehicle
limousine
racingCar
dirtRacingCar
formula
roadRacingCar
roadRacingCar
sport
racingCar
dirtRacingCar
formula
roadRacingCar
Motorcycle
dirtRacingBike
MotorSport
racingBike
dirtRacingBike
roadRacingBike
MotorTransport
buggy
scooter
tricycle
racingBike
dirtRacingBike
roadRacingBike
roadRacingBike
scooter

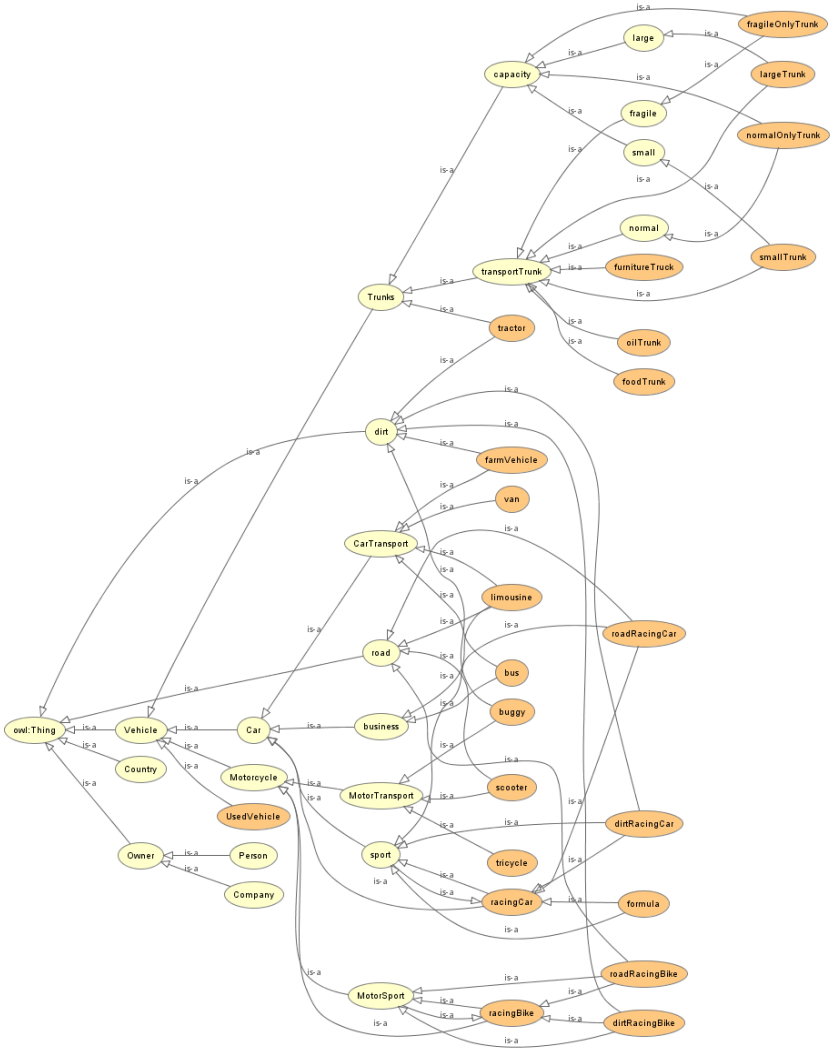

Trunks
capacity
fragileOnlyTrunk
largeTrunk
normalOnlyTrunk
smallTrunk
tractor
transportTrunk
foodTrunk
fragile
fragileOnlyTrunk
furnitureTruck
largeTrunk
normal
normalOnlyTrunk
oilTrunk
smallTrunk
UsedVehicle


owl:Thing
Country
dirt
buggy
dirtRacingBike
dirtRacingCar
farmVehicle
tractor
Owner
Company
Person
road
limousine
roadRacingBike
roadRacingCar
scooter
Vehicle
Car
business
bus
limousine
CarTransport
bus
farmVehicle
limousine
van
racingCar
dirtRacingCar
formula
roadRacingCar
sport
dirtRacingCar
formula
roadRacingCar
Motorcycle
MotorSport
dirtRacingBike
roadRacingBike
MotorTransport
buggy
scooter
tricycle
racingBike
dirtRacingBike
roadRacingBike
Trunks
capacity
fragileOnlyTrunk
large
largeTrunk
normalOnlyTrunk
small
smallTrunk
tractor


transportTrunk
foodTrunk
fragile
fragileOnlyTrunk
furnitureTruck
largeTrunk
normal
normalOnlyTrunk
oilTrunk
smallTrunk
UsedVehicle

b. Παρακάτω ακολουθούν τα γραφήματα των παραπάνω ιεραρχιών (Αριστερό γράφημα : Asserted Model , Δεξιό γράφημα: Inferred Model)





c. Οι διαφορές μεταξύ είναι οι εξής:

- Η κλάση bus στο Asserted Model είναι υποκλάση της κλάσης Car , business και CarTransport ενώ στο Inferred Model είναι υποκλάση μόνο της κλάσης CarTransport και business . Αυτό συμβαίνει διότι ο reasoner αντιλαμβάνεται ότι αφού είναι υποκλάση των CarTransport και business τα οποία είναι υποκλάσεις της Car τότε προφανώς ανήκει στην κλάση Car .
- Η κλάση farmVehicle στο Asserted Model είναι υποκλάση της κλάσης Car , dirt και CarTransport και έχει δηλωθεί ότι η κλάση farmVehicle είναι η τομή των CarTransport και dirt ενώ στο Inferred Model είναι υποκλάση μόνο της κλάσης CarTransport και dirt. Αυτό συμβαίνει διότι ο reasoner βλέπει ότι η κλάση είναι η τομη των CarTransport και dirt με αποτέλεσμα να εισάγει την farmVehicle ως υποκλάση αυτών δηλώνοντας έμμεσα ότι είναι και τύπου Car (αφού είναι σε υποκλάση της κλάσης Car).
- Η κλάση dirtRacingCar στο Asserted Model είναι υποκλάση της κλάσης Car , και ίση με την τομή των dirt και racingCar ενώ στο Inferred Model είναι υποκλάση μόνο της κλάσης racingCar ,sport και dirt. Αυτό συμβαίνει διότι ο reasoner βλέπει ότι η κλάση είναι η τομη των racingCar και dirt και ότι η κλάση racingCar είναι ίση με την κλάση sport με αποτέλεσμα να εισάγει την dirtRacingCar ως υποκλάση των racingCar ,sport και dirt .
- Η κλάση limousine στο Asserted Model είναι υποκλάση της κλάσης Car , και ίση με την τομή των road, business και CarTransport ενώ στο Inferred Model είναι υποκλάση μόνο της κλάσης CarTransport , road και business. Αυτό συμβαίνει διότι ο reasoner βλέπει ότι η κλάση είναι η τομη των road, business και CarTransport με αποτέλεσμα να εισάγει την limousine ως υποκλάση των road, business και CarTransport.
- Η κλάση roadRacingCar στο Asserted Model είναι υποκλάση της κλάσης Car , και ίση με την τομή των road και racingCar ενώ στο Inferred Model είναι υποκλάση μόνο της κλάσης racingCar ,sport και road. Αυτό συμβαίνει διότι ο reasoner βλέπει ότι η κλάση είναι η τομη των racingCar και road και ότι η κλάση racingCar είναι ίση με την κλάση sport με αποτέλεσμα να εισάγει την roadRacingCar ως υποκλάση των racingCar ,sport και road .
- Η κλάση racingCar στο Asserted Model είναι υποκλάση της κλάσης Car και της sport , και ίση με την τομή των Car, sport ενώ στο Inferred Model είναι υποκλάση της κλάσης Car και ίση με την κλάση sport . Αυτό συμβαίνει διότι ο reasoner βλέπει ότι η κλάση είναι η τομη των Car και sport καθώς και ότι η racingCar είναι υποκλάση της sport οπότε τις θέτει ίσες .
- Η κλάση fragileOnlyTrunk στο Asserted Model είναι υποκλάση της κλάσης Trunks και ίση με την τομή των fragile, (largeTrunk or smallTrunk) που είναι υποκλάσεις των κλάσεων large,small αντίστοιχα. Ενώ στο Inferred Model είναι υποκλάση της κλάσης capacity και fragile . Αυτό συμβαίνει διότι ο reasoner βλέπει ότι η κλάση είναι η τομη των fragile, (largeTrunk or smallTrunk) και ξέρει ότι τα largeTrunk, smallTrunk είναι υποκλάσεις των large,small(και της transportTrunk) τα οποία είναι υποκλάσεις της capacity.

Ερώτημα 5

a.

1. Ερώτημα σε φυσική γλώσσα: **Ποιο/α όχημα/τα είναι μεταχειρισμένο/α .**
   Το παραπάνω ερώτημα σε **SPARQL** είναι:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX uni: <http://www.mydomain.com/vehicles/>
SELECT ?instance ?name WHERE{
       ?x rdf:type owl:Class
       FILTER(?x=uni:UsedVehicle)
       ?instance rdf:type ?x .
       OPTIONAL{?instance uni:ModelName ?name .}
}
```

Και τα αποτελέσματα είναι :

?instance	?name
uni:dirtBike1	Bke231^^rdfs:Literal
uni:usedVehicle1	XMY123^^rdfs:Literal
uni:roadBike1	PBE900^^rdfs:Literal

2. Ερώτημα σε φυσική γλώσσα: **Ποιο/α όχημα/τα έχει/έχουν λιγότερους ρίπους .**
   Το παραπάνω ερώτημα σε **SPARQL** είναι:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX uni: <http://www.mydomain.com/vehicles/>
SELECT DISTINCT ?ecoVehicles ?instance WHERE
{      ?x rdf:type owl:Class
       FILTER(?x=uni:Vehicle)
       ?instance rdf:type ?x ;
       uni:moreEcoThan ?y .
       OPTIONAL {?instance uni:ModelName ?ecoVehicles .}
}
```

Και τα αποτελέσματα είναι :

?ecoVehicles	?instance
ERC2015^^rdfs:Literal	uni:roadRacingCar2
BYR212^^rdfs:Literal	uni:bus1
BUG212^^rdfs:Literal	uni:buggy1
DRT90^^rdfs:Literal	uni:farm1
TRI800^^rdfs:Literal	uni:tricycle1

3. Ερώτημα σε φυσική γλώσσα: **Ποια οχήματα είναι παρόμοια σε χαρακτηριστικά .**
   Το παραπάνω ερώτημα σε **SPARQL** είναι:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX uni: <http://www.mydomain.com/vehicles/>
SELECT DISTINCT ?model1 ?model2 WHERE{
        ?x rdf:type owl:Class
        FILTER(?x=uni:Vehicle)
        ?z rdf:type ?x ;
        uni:equivalentModel ?y.
        ?z uni:ModelName ?model1.
        ?y uni:ModelName ?model2
}
```

   Και τα αποτελέσματα είναι :

?model1	?model2
LIMO2124^^rdfs:Literal	LIMO3441^^rdfs:Literal
LIMO3441^^rdfs:Literal	LIMO2124^^rdfs:Literal
F12011^^rdfs:Literal	F12010^^rdfs:Literal
F12010^^rdfs:Literal	F12011^^rdfs:Literal
DRT90^^rdfs:Literal	DRT221^^rdfs:Literal
DRT221^^rdfs:Literal	DRT90^^rdfs:Literal

   Ανά δεύτερη γραμμή φαίνεται γνώση που προκύπτει από τον μηχανισμό συμπερασμού .

4. Ερώτημα σε φυσική γλώσσα: **Ποια είναι τα οχήματα που έχουν κατασκευαστεί ανά χώρα κατά αλφαβητική σειρά.**
   Το παραπάνω ερώτημα σε **SPARQL** είναι:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX uni: <http://www.mydomain.com/vehicles/>
SELECT ?instance ?name ?country WHERE{
{
        ?x rdf:type owl:Class
        FILTER(?x=uni:Country)
        ?country rdf:type ?x .
} UNION {
        ?country uni:hasManufactured ?instance.
        OPTIONAL {?instance uni:ModelName ?name .}
}
}ORDER BY ASC(?country)
```

Και τα αποτελέσματα είναι :

?instance	?name	?country
uni:France		
uni:oilTrunk1	TRU900^^rdfs:Literal	uni:France
uni:van1	VRE123^^rdfs:Literal	uni:France
uni:buggy1	BUG212^^rdfs:Literal	uni:France
uni:farm1	DRT90^^rdfs:Literal	uni:France
uni:roadRacingBike1	Bke3421^^rdfs:Literal	uni:France
		uni:Germany
uni:bus1	BYR212^^rdfs:Literal	uni:Germany
uni:dirtBike1	Bke231^^rdfs:Literal	uni:Germany
uni:dirtRacingCar2	RC5000^^rdfs:Literal	uni:Germany
uni:limo2	LIMO3441^^rdfs:Literal	uni:Germany
uni:formulaM2	F12011^^rdfs:Literal	uni:Germany
uni:largeTrunk1	PTRU201^^rdfs:Literal	uni:Germany
uni:bus3	BUS4501^^rdfs:Literal	uni:Germany
		uni:Italy
uni:limo1	LIMO2124^^rdfs:Literal	uni:Italy
uni:tractor1	TRC2355^^rdfs:Literal	uni:Italy
uni:formulaM1	F12010^^rdfs:Literal	uni:Italy
uni:normalTrunk1	TRU2010^^rdfs:Literal	uni:Italy
uni:bus2	BYH231^^rdfs:Literal	uni:Italy
		uni:UK
uni:fragileTrunk1	TRU212^^rdfs:Literal	uni:UK
uni:van2	ERRW3221^^rdfs:Literal	uni:UK
uni:dirtRacingCar1	RC1231^^rdfs:Literal	uni:UK
uni:farm2	DRT221^^rdfs:Literal	uni:UK
uni:roadRacingCar1	RRC435^^rdfs:Literal	uni:UK
		uni:USA
uni:scooter1	SC900^^rdfs:Literal	uni:USA
uni:dirtRacingCar3	RC7880^^rdfs:Literal	uni:USA
uni:roadRacingCar2	ERC2015^^rdfs:Literal	uni:USA

5. Ερώτημα σε φυσική γλώσσα: **Ποιο είναι το ακριβότερο όχημα που έχει αγοράσει ο κάθε πελάτης .**
   Το παραπάνω ερώτημα σε **SPARQL** είναι:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX uni: <http://www.mydomain.com/vehicles/>
SELECT ?Owner (MAX(?price) as ?maxPrice) (SAMPLE(?name) as
?model) WHERE{
        ?x rdf:type owl:Class
        FILTER(?x=uni:Owner)
        ?Owner rdf:type ?x;
        uni:owns ?y.
```

```
        ?y uni:Price ?price .
        ?y uni:ModelName ?name .
}
GROUP BY ?Owner
ORDER BY DESC(?maxPrice)
```

Και τα αποτελέσματα είναι :

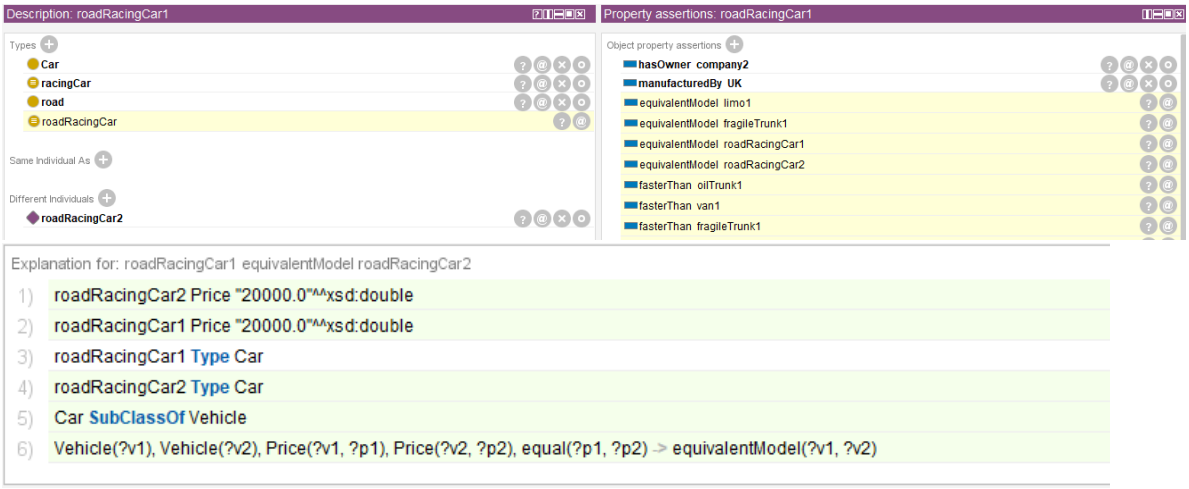
?Owner	?maxPrice	?model
uni:company1	1.05E+07	ERC2015^^rdfs:Literal
uni:company2	1056001	BUS4501^^rdfs:Literal
uni:person2	231311	DRT90^^rdfs:Literal
uni:person3	150000	RC7880^^rdfs:Literal
uni:person1	120000	RC1231^^rdfs:Literal

b.

Παρακάτω ακολουθούν οι κανόνες που εφαρμόζονται στην οντολογία .

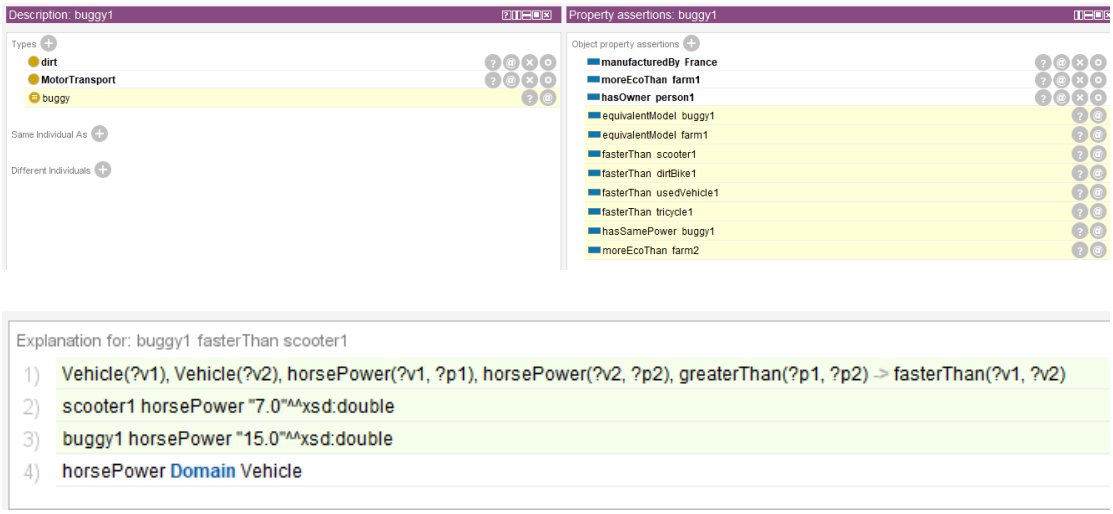
1. *Vehicle(?v1) ^ Vehicle(?v2) ^ Price(?v1, ?p1) ^ Price(?v2, ?p2) ^ swrlb:equal(?p1, ?p2) -> equivalentModel(?v1, ?v2)*

   Ο παραπάνω κανόνας εκφράζεται σε φυσική γλώσσα ως εξής: Αν τα οχήματα v1 και v2 έχουν ίδιες τιμές τότε τα οχήματα αυτά έχουν παρόμοια χαρακτηριστικά .



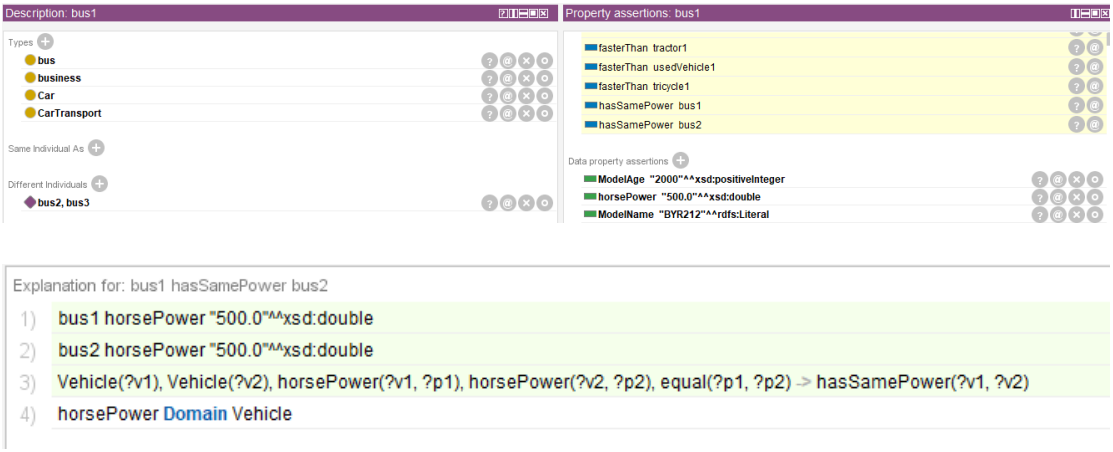
2. *Vehicle(?v1) ^ Vehicle(?v2) ^ horsePower(?v1, ?p1) ^ horsePower(?v2, ?p2) ^ swrlb:greaterThan(?p1, ?p2) -> fasterThan(?v1, ?v2)*

Ο παραπάνω κανόνας εκφράζεται σε φυσική γλώσσα ως εξής: Αν το όχημα v1 έχει μεγαλύτερη ιπποδύναμη από το όχημα v2 τότε το v1 είναι γρηγορότερο από το όχημα v2 .



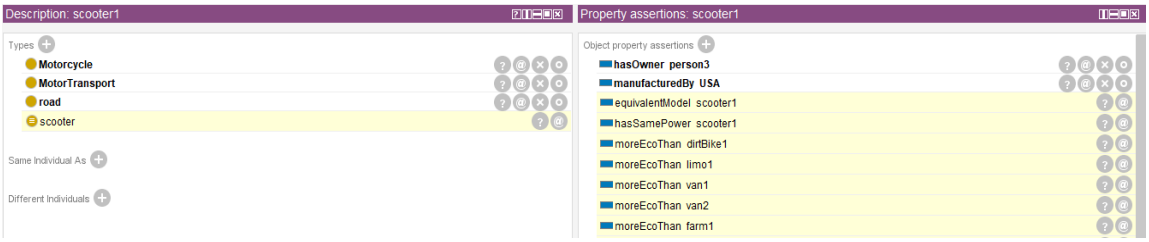
3. *Vehicle(?v1) ^ Vehicle(?v2) ^ horsePower(?v1, ?p1) ^ horsePower(?v2, ?p2) ^ swrlb:equal(?p1, ?p2) -> hasSamePower(?v1, ?v2)*

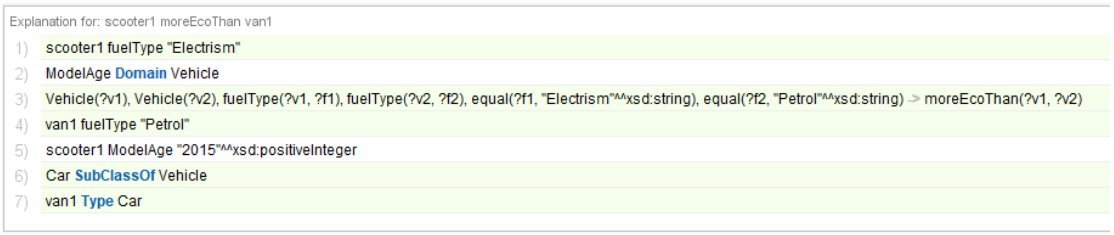
Ο παραπάνω κανόνας εκφράζεται σε φυσική γλώσσα ως εξής: Αν το όχημα v1 έχει ίση ιπποδύναμη με το όχημα v2 τότε το v1 είναι «ισοδύναμο» με το όχημα v2 .



4. *Vehicle(?v1) ^ Vehicle(?v2) ^ fuelType(?v1, ?f1) ^ fuelType(?v2, ?f2) ^ swrlb:equal(?f1, "Electrism") ^ swrlb:equal(?f2, "Petrol") -> moreEcoThan(?v1, ?v2)*

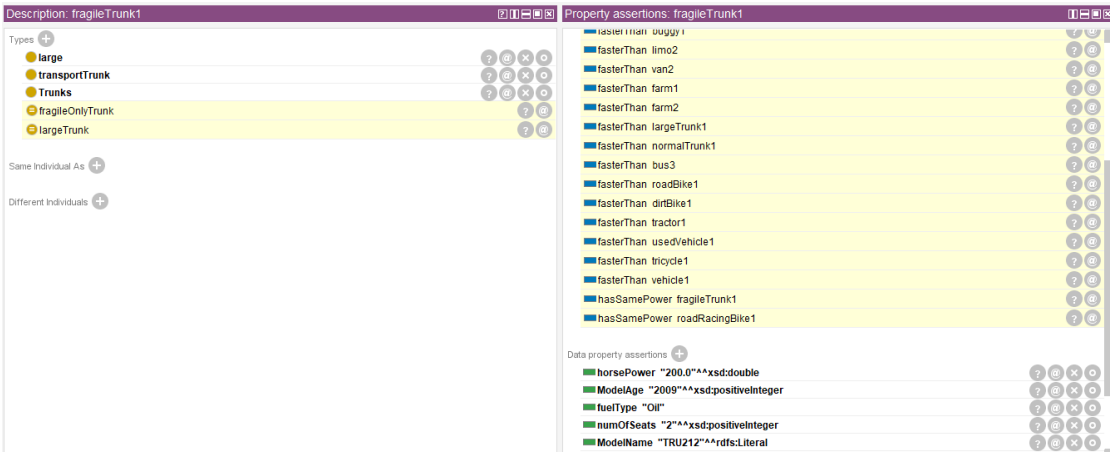
Ο παραπάνω κανόνας εκφράζεται σε φυσική γλώσσα ως εξής: Αν το όχημα v1 «καίει» Ηλεκτρισμό και το όχημα v2 «καίει» Βενζίνη τότε το όχημα v1 είναι ποιο οικολογικό από το v2 .

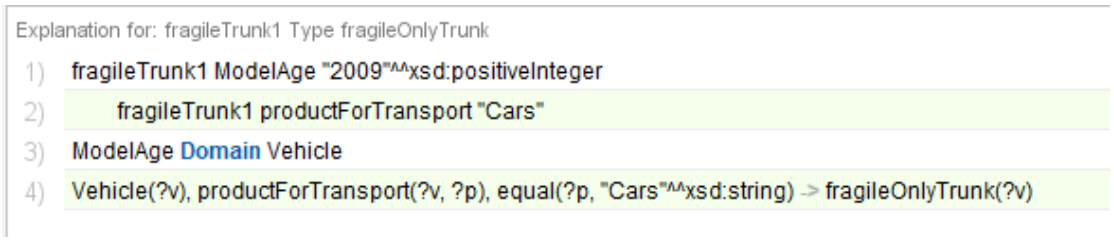




5. *Vehicle(?v) ^ productForTransport(?v, ?p) ^ swrlb:equal(?p, "Cars") -> fragileOnlyTrunk(?v)*

   Ο παραπάνω κανόνας εκφράζεται σε φυσική γλώσσα ως εξής: Αν το όχημα ?v μεταφέρει αμάξια τότε είναι φορτηγό που είναι για μεταφορές «ευαίσθητων» προϊόντων .

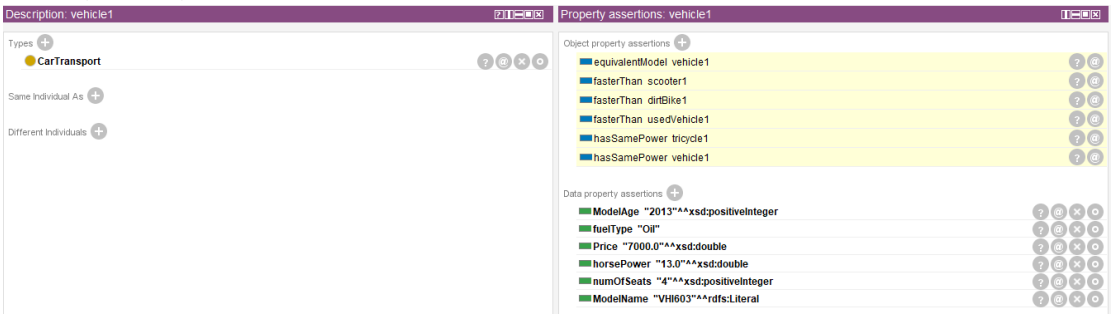




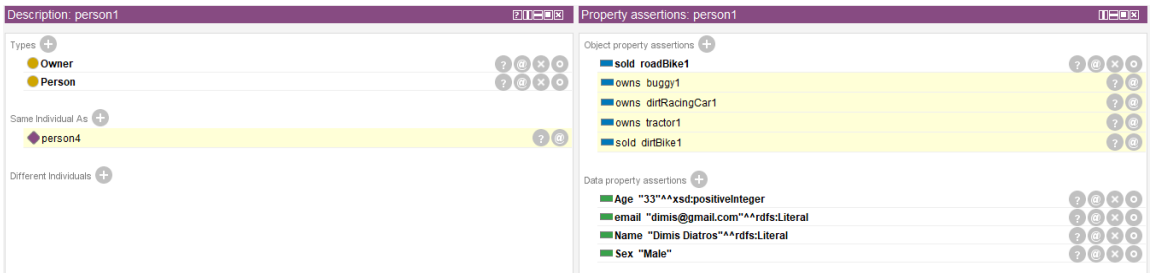
Ερώτημα 6

- ***Open-world-assumption*** :
  Στο open-world-assumption(OWA) ο παρατηρητής θα «παραδεχτεί» ότι η γνώση του είναι ελλιπής πάνω σε μία ερώτηση σε αντίθεση για παράδειγμα με το closed-world-assumption(CWA) όπου ο παρατηρητής θα απαντήσει(με π.χ True, False)

ακόμα και αν έχει ελλιπείς γνώσεις .Ένα παράδειγμα όπου εμφανίζεται το OWA είναι στο στιγμιότυπο vehicle1 όπου παρόλο ότι έχω δηλώσει ότι είναι τύπου CarTransport που είναι υπό-κλάση της Car δεν μπορεί να συμπεράνει ότι ανήκει και στην κλάση Car .



- ***Non-unique-name-assumption*** :
  Το Non-unique-name-assumption(NUNA) είναι η υπόθεση που κάνει η οντολογία η οποία λέει ότι ακόμα και αν 2 στιγμιότυπα έχουν διαφορετικά ονόματα ή/και ID's αυτό δεν σημαίνει ότι είναι διαφορετικά μεταξύ τους. Μια περίπτωση όπου η υπόθεση αυτή λαμβάνει μέρος στην οντολογία μου είναι μεταξύ των στιγμιότυπων person1 και person4 όπου επειδή ο person1 έχει πουλήσει το roadBike1 και ο person4 έχει κάνει το ίδιο ο reasoner συμπεραίνει ότι είναι το ίδιο άτομο .



Βέβαια αν εισαχθεί κάποιο functional property στο person4 τότε θα υπάρξει ασυνέπεια .

Ερώτημα 7

Για την εφαρμογή χρησιμοποίησα το JDK 11 .

Τα εργαλεία/βιβλιοθήκες που χρησιμοποίησα για την κατασκευή της εφαρμογής είναι το JENA API για την διαχείριση της οντολογίας και για reasoner χρησιμοποίησα τον pellet (https://github.com/stardog-union/pellet) ο οποίος είναι συμβατός με το JENA API .

Επίσης επειδή η JENA δεν υποστηρίζει οντολογίες οι οποίες είναι γραμμένες σε OWL/XML syntax δημιούργησα ένα άλλο αρχείο στο οποίο αποθήκευσα την οντολογία με RDF/XML syntax(μέσω του protégé) που υποστηρίζεται από την JENA .

Για να εκτελέσετε την εφαρμογή πηγαίνετε στο *Ερώτημα 7/OWLApp/target/* ανοίξτε *cmd* και εισάγετε την παρακάτω εντολή : ***java -jar OWLApp-1.0-SNAPSHOT.jar*** .

Παρακάτω παρουσιάζεται η μέθοδος η οποία φορτώνει την οντολογία που επέλεξε ο χρήστης και εφαρμόζει το μοντέλο συμπερασμού που έχει οριστεί .

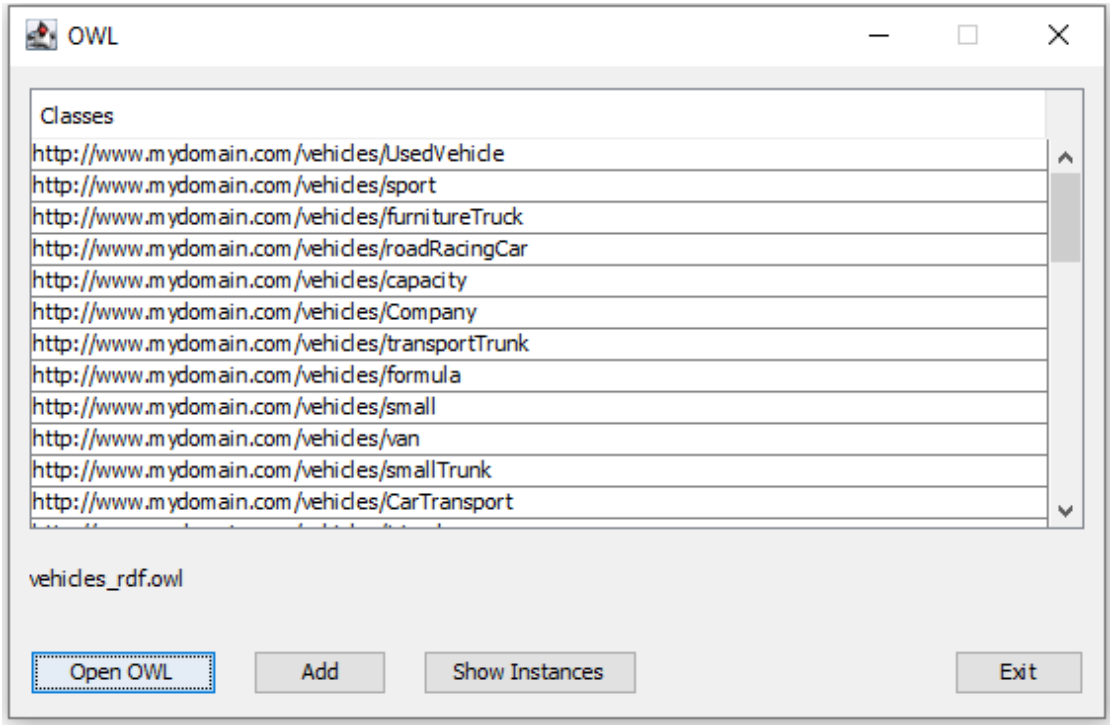
```
private void loadOWLMouseClicked(java.awt.event.MouseEvent evt) {

        FileFilter filter = new FileNameExtensionFilter("OWL file",
"owl");
        jFileChooser.addChoosableFileFilter(filter);
        jFileChooser.setFileFilter(filter);
        DefaultTableModel tableModel =(DefaultTableModel)
dataTable.getModel();

dataTable.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        int returnVal = jFileChooser.showOpenDialog(this);
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            owlFile = jFileChooser.getSelectedFile();
            path = owlFile.getAbsolutePath();
            InputStream in = FileManager.get().open(path);
            InputStreamReader rin = new InputStreamReader(in);
            filePathLabel.setText(owlFile.getName());
            final OntModel model =
ModelFactory.createOntologyModel(PelletReasonerFactory.THE_SPEC);
            try{
                        model.read(rin,null);
                        model.prepare();
                        KnowledgeBase kb = ((PelletInfGraph)
model.getGraph()).getKB();
                        boolean consistent = kb.isConsistent();
                        if(consistent){
                            publicOwl = model;
                        }else{
                         throw new InconsistentOntologyException();
                        }
            }catch(InconsistentOntologyException e){
                JOptionPane.showMessageDialog(this, e);
            }
            //System.out.println("Ontology loaded!!");
            ExtendedIterator<OntClass> i = model.listNamedClasses();
            while(i.hasNext()) {
                OntClass ontClass = (OntClass) i.next();
                if("Thing".equals(ontClass.getLocalName()) ||
"Nothing".equals(ontClass.getLocalName())){
                    continue;
                }
                //System.out.println(ontClass.toString());
                classesList.add(ontClass);
                tableModel.addRow(new Object[]{ontClass});
            }
            //System.out.println("END");
            addInstance.setEnabled(true);
            showInstances.setEnabled(true);
        }
}
```

Όταν ο χρήστης πατήσει το κουμπί «*Open OWL*» τότε εμφανίζεται ένας FileChooser ο οποίος δέχεται μόνο αρχεία με κατάληξη .owl . Έπειτα αν ο χρήστης επιλέξει κάποιο αρχείο οντολογίας τότε διαβάζω το περιεχόμενο του και δημιουργώ το μοντέλο της οντολογίας `OntModel model` στο οποίο «λέμε» ότι θα χρησιμοποιήσει σαν μηχανισμό συμπερασμού τον Pellet Reasoner(`PelletReasonerFactory.THE_SPEC`) . Το μοντέλο διαβάζει τα δεδομένα και έπειτα ξεκινάει η διαδικασία του συμπερασμού .Αφού τελειώσει η παραπάνω διαδικασία κοιτάω για το αν υπάρχουν ασάφειες/ασυνέπειες, αν υπάρχουν τότε «πετάω» ένα exception `InconsistentOntologyException e` και εμφανίζεται κατάλληλο μήνυμα αλλιώς εκχωρώ στην μεταβλητή *publicOwl* τύπου *OntModel* το αρχικό μοντέλο ώστε να το χρησιμοποιήσουμε και πιο μετά . Έπειτα παίρνουμε όλες τις κλάσεις της οντολογίας(`model.listNamedClasses()`) και της εισάγω στον πίνακα *dataTable* και στο τέλος της μεθόδου ενεργοποιώ τα κουμπιά *addInstance* και *showInstances* .

Παρακάτω φαίνεται το αποτέλεσμα της εκτέλεσης της παραπάνω μεθόδου .



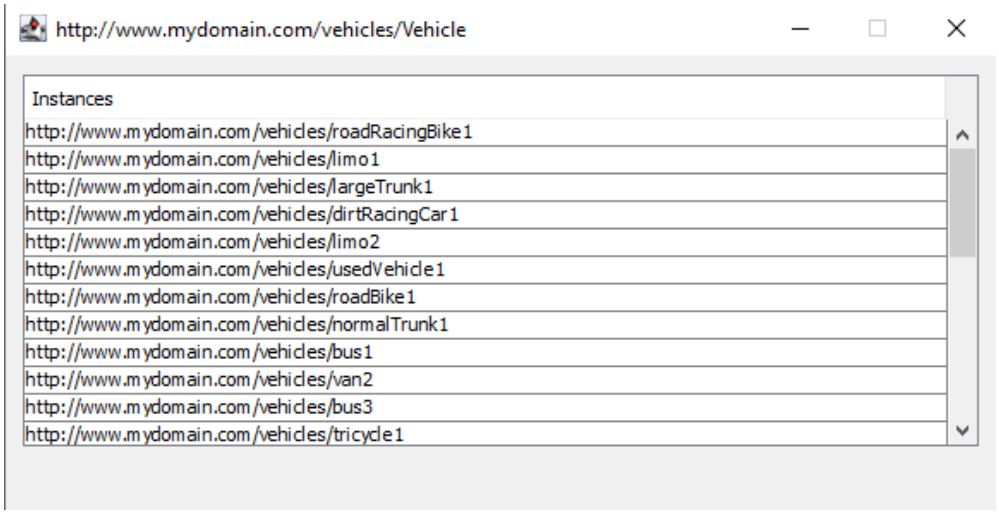
Έπειτα παρουσιάζεται η μέθοδος η οποία εκτελείται όταν ο χρήστης θελήσει να δει τα στιγμιότυπα κάποιας κλάσης η οποία εκτελείται εφόσον ο χρήσης επιλέξει κάποια κλάση από τον πίνακα *dataTable* και πατήσει το κουμπί *«Show Instances»*.

```
private void showInstancesMouseClicked(java.awt.event.MouseEvent evt)
{
        int row = dataTable.getSelectedRow();
        List<String> instances = new  ArrayList<String>();
        String owl_class = classesList.get(row).toString();
        String instancesQ = "PREFIX owl:
<http://www.w3.org/2002/07/owl#> " +
                            "PREFIX rdf:
<http://www.w3.org/1999/02/22-rdf-syntax-ns#> " +
                            "PREFIX rdfs:
<http://www.w3.org/2000/01/rdf-schema#> " +
```

```
                                    "PREFIX uni:
<http://www.mydomain.com/vehicles/> "+
                                    "SELECT ?instances WHERE{"+
                                    "?x rdf:type owl:Class ."+
                                    "FILTER(?x=<"+owl_class+">)"+
                                    "?instances rdf:type ?x .}";
        Query query = QueryFactory.create(instancesQ);
        QueryExecution qexec = SparqlDLExecutionFactory.create(query,
publicOwl);
        ResultSet rs = qexec.execSelect();
        try{
            if (rs.hasNext()) {
                for (; rs.hasNext();) {
                    QuerySolution rb = rs.nextSolution();
                    Resource res = rb.getResource("instances");
                    //System.out.println(res.toString());
                    instances.add(res.toString());
                }
                qexec.close();
            }
        }catch(InternalReasonerException err){
            instances.add("Τα στιγμιότυπα δεν μπορεσαν να φορτώσουν
(InternalReasonerException) ");
        }catch(InconsistentOntologyException e){
            JOptionPane.showMessageDialog(this, "Δεν μπορει να δειξει
τα στιγμιοτυπα λογω ασυνεπειας .");
        }
        InstanceTable it = new InstanceTable();
        it.addToTable(owl_class,instances);
        it.setVisible(true);
}
```

Η μέθοδος ξεκινάει παίρνοντας τον αριθμό της γραμμής του πίνακα *dataTable* που έχει επιλέξει ο χρήστης , έπειτα δημιουργώ μια λίστα(`List<String> instances`) για να αποθηκεύω τα στιγμιότυπα, παίρνω την κλάση που επέλεξε ο χρήστης και αμέσως μετά εκτελώ ένα ερώτημα SPARQL(με χρήση της SPARQLDL ώστε να ληφθεί υπόψη και ο συμπερασμός `SparqlDLExecutionFactory`) το οποίο μου επιστρέφει τα στιγμιότυπα για την συγκεκριμένη κλάση τα οποία και αποθηκεύω στην λίστα που αναφέρθηκε παραπάνω. Αμέσως μετά δημιουργώ ένα στιγμιότυπο της κλάσης *InstanceTable* (`InstanceTable it = new InstanceTable()`) η οποία είναι η κλάση του παραθύρου οπού εμφανίζονται τα παραπάνω στιγμιότυπα τα οποία προστίθενται στον πίνακα της κλάσης *InstanceTable* με την μέθοδο *addToTable* η οποία έχει σαν ορίσματα την κλάση που επέλεξε ο χρήστης και την λίστα με τα στιγμιότυπά της .

Παρακάτω φαίνεται το αποτέλεσμα της εκτέλεσης της παραπάνω μεθόδου .



Τέλος παρουσιάζονται οι μέθοδοι που υλοποιούν την προσθήκη νέων στιγμιοτύπων στην οντολογία .

Όταν ο χρήστης πατήσει το κουμπί «*Add*» τότε εκτελείται η μέθοδος :

```
private void addInstanceMouseClicked(java.awt.event.MouseEvent evt) {
        AddInstance addInstanceWindow = new
AddInstance(classesList,publicOwl,path);
        addInstanceWindow.setVisible(true);

    }
```

Η οποία δημιουργεί ένα στιγμιότυπο της κλάσης *AddInstance* η οποία είναι το παράθυρο από το οποίο ο χρήστης μπορεί να εισάγει νέα στιγμιότυπα .Αφότου ανοίξει το παράθυρο ο χρήστης πληκτρολογεί στο πεδίο Type την κλάση στην οποία θέλει να προσθέσει νέο στιγμιότυπο και έπειτα επιλέγει το κουμπί «*Check*» . Η μέθοδος που εκτελείται με το πάτημα του «*Check*» είναι η :

```
private void checkTypesActionPerformed(java.awt.event.ActionEvent
evt) {
        String txt = classesTxt.getText();
        namespace = classesList.get(0).getNameSpace();
        Boolean flag = false;
        if(txt.length()==0){
            JOptionPane.showMessageDialog(this, "Το πεδίο κειμένου
είναι άδειο");
        }else{
            for(OntClass c: classesList){
                String name = c.getLocalName();
                if(this.checkString(name,txt)){
                    cls = c;
                    flag = true;
                    break;
                }
            }
            if(flag==false){
```

```
                JOptionPane.showMessageDialog(this, "Εισαγες λάθος
κλάση!");
            }
        }
        if(flag){
            individualNameTxt.setEnabled(true);
            OntClass superclass =
publicOwl.getOntClass(namespace+"Vehicle");
            OntClass Pesronsuperclass =
publicOwl.getOntClass(namespace+"Owner");
            if(cls.hasSuperClass(superclass) ||
cls.getLocalName().equals("Vehicle") ||
cls.getLocalName().equals("dirt") ||
cls.getLocalName().equals("road")){
                name_txt.setEnabled(true);
                age_txt.setEnabled(true);
                manuafacturedBy.setEnabled(true);
                fuelType.setEnabled(true);
                hasOwner.setEnabled(true);
                horsePower.setEnabled(true);
                vehiclePrice.setEnabled(true);
                soldBy.setEnabled(true);
                numOfSeats.setEnabled(true);
                productForTransport.setEnabled(true);
                category = "Vehicle";
            }else if(cls.getLocalName().equals("Country")){
                name_txt.setEnabled(true);
                category = "Country";
            }else if(cls.hasSuperClass(Pesronsuperclass) ||
cls.getLocalName().equals("Owner")){
                name_txt.setEnabled(true);
                owner_email_txt.setEnabled(true);
                owner_sex.setEnabled(true);
                age_txt.setEnabled(true);
                soldVehicle.setEnabled(true);
                ownsVehicle.setEnabled(true);
                category = "Owner";
            }
            applyBtn.setEnabled(true);
            checkTypes.setEnabled(false);
        }
    }
```

Σημείωση: Η μέθοδος *checkString* υπάρχει ώστε η είσοδος του χρήση στο πεδίο Type να μην είναι case sensitive . Ο κώδικας της μεθόδου είναι :

```
private Boolean checkString(String str1,String str2){
        String upperArr[] = {str1.toUpperCase(),str2.toUpperCase()};
        String lowArr[] = {str1.toLowerCase(),str2.toLowerCase()};
        if(upperArr[0].equals(upperArr[1]) ||
lowArr[0].equals(lowArr[1]))
            return true;
        return false;
}
```

Όπου παίρνει το κείμενο που έχει γραφτεί στο πεδίο, αν είναι κενό τότε εμφανίζεται σχετικό μήνυμα αλλιώς ελέγχω για το αν η κλάση που έγραψε ο χρήστης υπάρχει . Αν όχι τότε εμφανίζεται κατάλληλο μήνυμα αλλιώς κοιτάω σε ποια κλάση ανήκει η κλάση που είσαγε ο χρήστης (Superclass) .Αν η κλάση έχει ως υπερ-κλάση την *«Vehicle»* ή είναι η *«Vehicle», «dirt», «road»* τότε ενεργοποιούνται τα πεδία που αφορούν τα οχήματα και θέτει την μεταβλητή *category* ίση με την τιμή Vehicle (*String type*), αλλιώς αν η κλάση έχει ως υπερ-κλάση την «*Country*» ενεργοποιούνται τα πεδία που αφορούν την χώρα και θέτει την μεταβλητή *category* ίση με την τιμή Country (*String type*), αλλιώς αν η κλάση έχει ως υπερ-κλάση την «*Owner*» ή είναι η *«Owner»* ενεργοποιούνται τα πεδία που αφορούν τους πελάτες και θέτει την μεταβλητή *category* ίση με την τιμή Owner (*String type*) και στο τέλος ενεργοποιείται το κουμπί *«Add»* ενώ απενεργοποιείται το κουμπί *«Check»* . Ο χρήστης γράφει ή και όχι στα πεδία και έπειτα επιλέγει το κουμπί *«Add»* το οποίο εκτελεί την μέθοδο :

```
private void applyBtnActionPerformed(java.awt.event.ActionEvent evt)
{

        String indName = individualNameTxt.getText();
        List<JTextField> widgets = new ArrayList<JTextField>();
        if(indName.length()==0){
            JOptionPane.showMessageDialog(this, "Το πεδίο κειμένου
είναι άδειο");
        }else{
            if(category.equals("Vehicle")){
                Individual individual =
publicOwl.createIndividual(namespace+indName,cls);
                ObjectProperty objHasOwner =
publicOwl.getObjectProperty(namespace+"hasOwner");
                ObjectProperty objmanuafacturedBy =
publicOwl.getObjectProperty(namespace+"manufacturedBy");
                ObjectProperty objSoldBy =
publicOwl.getObjectProperty(namespace+"soldBy");
                DatatypeProperty dataName =
publicOwl.getDatatypeProperty(namespace+"ModelName");
                DatatypeProperty dataAge =
publicOwl.getDatatypeProperty(namespace+"ModelAge");
                DatatypeProperty dataPower =
publicOwl.getDatatypeProperty(namespace+"horsePower");
                DatatypeProperty dataFuel =
publicOwl.getDatatypeProperty(namespace+"fuelType");
                DatatypeProperty dataPrice =
publicOwl.getDatatypeProperty(namespace+"Price");
                DatatypeProperty dataSeats =
publicOwl.getDatatypeProperty(namespace+"numOfSeats");
                DatatypeProperty dataProduct =
publicOwl.getDatatypeProperty(namespace+"productForTransport");
                widgets.add(individualNameTxt);
                widgets.add(name_txt);
                widgets.add(age_txt);
                widgets.add(manuafacturedBy);
                widgets.add(hasOwner);
                widgets.add(vehiclePrice);
                widgets.add(numOfSeats);
                widgets.add(horsePower);
                widgets.add(soldBy);
```

```
                if(name_txt.getText().length()>0){
                    Literal l =
publicOwl.createTypedLiteral(name_txt.getText(),
"http://www.w3.org/2000/01/rdf-schema#Literal");
                    individual.addProperty(dataName,l);
                }if(age_txt.getText().length()>0){

individual.addProperty(dataAge,publicOwl.createTypedLiteral(age_txt.g
etText(), XSDDatatype.XSDpositiveInteger));
                }if(manuafacturedBy.getText().length()>0){
                    Individual country =
publicOwl.getIndividual(namespace + manuafacturedBy.getText());
                    if(country==null){
                        JOptionPane.showMessageDialog(this, "Η χώρα
που δίαλεξες δεν είναι καταχυρωμένη !");
                    }
                    if(country.isIndividual()){

individual.addProperty(objmanuafacturedBy,publicOwl.createResource(na
mespace+country.getLocalName()));
                    }
                }if(hasOwner.getText().length()>0){
                    Individual owner =
publicOwl.getIndividual(namespace + hasOwner.getText());
                    if(owner==null) {
                        JOptionPane.showMessageDialog(this, "Ο
πελάτης που δήλωσες δεν έχει καταχωρηθεί");
                    }
                    if (owner.isIndividual()) {
                        individual.addProperty(objHasOwner,
publicOwl.createResource(namespace+owner.getLocalName()));
                    }
                }if(vehiclePrice.getText().length()>0){

individual.addProperty(dataPrice,publicOwl.createTypedLiteral(vehicle
Price.getText(), XSDDatatype.XSDdouble));
                }if(numOfSeats.getText().length()>0){

individual.addProperty(dataSeats,publicOwl.createTypedLiteral(numOfSe
ats.getText(), XSDDatatype.XSDpositiveInteger));
                }if(horsePower.getText().length()>0){

individual.addProperty(dataPower,publicOwl.createTypedLiteral(horsePo
wer.getText(), XSDDatatype.XSDdouble));

}if(productForTransport.getSelectedItem().equals("None")==false){
                    individual.addProperty(dataProduct, (String)
productForTransport.getSelectedItem());
                }if(soldBy.getText().length()>0){
                    Individual owner1 =
publicOwl.getIndividual(namespace + soldBy.getText());
                    if (owner1.isIndividual()) {
                        individual.addProperty(objSoldBy,
publicOwl.createResource(namespace+owner1.getLocalName()));
                    }else if(owner1==null) {
                        JOptionPane.showMessageDialog(this, "Ο
πελάτης που δήλωσες δεν έχει καταχωρηθεί");
```

```
                    }
                }
                individual.addProperty(dataFuel, (String)
fuelType.getSelectedItem());

                try{
                    publicOwl.prepare();
                    KnowledgeBase kb = ((PelletInfGraph)
publicOwl.getGraph()).getKB();
                    boolean consistent = kb.isConsistent();
                    if(consistent==false){
                        throw new InconsistentOntologyException();
                    }else{
                        FileWriter out = null;
                        try {
                            out = new FileWriter(path);
                            publicOwl.write(out, "RDF/XML");
                        } catch (IOException ex) {

Logger.getLogger(AddInstance.class.getName()).log(Level.SEVERE, null,
ex);
                        }
                        for(JTextField t: widgets){
                            t.setText("");
                            t.setEnabled(false);
                        }
                        fuelType.setEnabled(false);
                        productForTransport.setEnabled(false);

                    }
                }catch(InconsistentOntologyException e){
                    JOptionPane.showMessageDialog(this, e);
                    this.dispose();
                }
            }else if(category.equals("Country")){
                Individual individual =
publicOwl.createIndividual(namespace+indName,cls);
                DatatypeProperty dataName =
publicOwl.getDatatypeProperty(namespace+"CountryName");
                if (name_txt.getText().length() > 0) {
                    Literal l =
publicOwl.createTypedLiteral(name_txt.getText(),
"http://www.w3.org/2000/01/rdf-schema#Literal");
                    individual.addProperty(dataName, l);
                }
                try {
                    publicOwl.prepare();
                    KnowledgeBase kb = ((PelletInfGraph)
publicOwl.getGraph()).getKB();
                    boolean consistent = kb.isConsistent();
                    if (consistent == false) {
                        throw new InconsistentOntologyException();
                    } else {
                        FileWriter out = null;
                        try {
                            out = new FileWriter(path);
                            publicOwl.write(out, "RDF/XML");
```

```
                        } catch (IOException ex) {
Logger.getLogger(AddInstance.class.getName()).log(Level.SEVERE, null,
ex);
                        }
                        individualNameTxt.setText("");
                        individualNameTxt.setEnabled(false);
                        name_txt.setText("");
                        name_txt.setEnabled(false);
                    }
                } catch (InconsistentOntologyException e) {
                    JOptionPane.showMessageDialog(this, e);
                    this.dispose();
                }
            }else if(category.equals("Owner")){
                Individual individual =
publicOwl.createIndividual(namespace+indName,cls);
                DatatypeProperty dataName =
publicOwl.getDatatypeProperty(namespace+"Name");
                DatatypeProperty dataAge =
publicOwl.getDatatypeProperty(namespace+"Age");
                DatatypeProperty dataSex =
publicOwl.getDatatypeProperty(namespace+"Sex");
                DatatypeProperty dataEmail =
publicOwl.getDatatypeProperty(namespace+"email");
                ObjectProperty objSold =
publicOwl.getObjectProperty(namespace+"sold");
                ObjectProperty objowns =
publicOwl.getObjectProperty(namespace+"owns");
                widgets.add(individualNameTxt);
                widgets.add(name_txt);
                widgets.add(age_txt);
                widgets.add(owner_email_txt);
                widgets.add(soldVehicle);
                widgets.add(ownsVehicle);
                if(name_txt.getText().length() > 0){
                    Literal l =
publicOwl.createTypedLiteral(name_txt.getText(),
"http://www.w3.org/2000/01/rdf-schema#Literal");
                    individual.addProperty(dataName, l);
                }if(age_txt.getText().length() > 0){

individual.addProperty(dataAge,publicOwl.createTypedLiteral(age_txt.g
etText(), XSDDatatype.XSDpositiveInteger));
                }if(owner_email_txt.getText().length() > 0){
                    Literal l =
publicOwl.createTypedLiteral(owner_email_txt.getText(),
"http://www.w3.org/2000/01/rdf-schema#Literal");
                    individual.addProperty(dataEmail, l);
                }if(soldVehicle.getText().length() > 0){
                    if(soldVehicle.getText().contains(",")){
                        String[] data =
soldVehicle.getText().split(",");
                        List<Individual> list = new
ArrayList<Individual>();
                        for(String d: data){
```

```
                            Individual ind =
publicOwl.getIndividual(namespace + d);
                            if(ind.isIndividual()){
                                list.add(ind);
                            }
                            if(ind==null){
                                JOptionPane.showMessageDialog(this,
"Το "+d+" δεν είναι στιγμιότυπο .");
                                return ;
                            }
                        }
                        for(Individual ind: list){
                            individual.addProperty(objSold,
publicOwl.createResource(namespace+ind.getLocalName()));
                        }
                    }else{
                        Individual ind =
publicOwl.getIndividual(namespace + soldVehicle.getText());
                        if(ind==null) {
                            JOptionPane.showMessageDialog(this, "Το
όχημα που δήλωσες δεν έχει καταχωρηθεί");
                        }
                        if(ind.isIndividual()){
                            individual.addProperty(objSold,
publicOwl.createResource(namespace+ind.getLocalName()));
                        }
                    }
                }if(ownsVehicle.getText().length() > 0){
                    if(ownsVehicle.getText().contains(",")){
                        String[] data =
ownsVehicle.getText().split(",");
                        List<Individual> list = new
ArrayList<Individual>();
                        for(String d: data){
                            Individual ind =
publicOwl.getIndividual(namespace + d);
                            if(ind.isIndividual()){
                                list.add(ind);
                            }
                            if(ind==null){
                                JOptionPane.showMessageDialog(this,
"Το "+d+" δεν είναι στιγμιότυπο .");
                                return ;
                            }
                        }
                        for(Individual ind: list){
                            individual.addProperty(objowns,
publicOwl.createResource(namespace+ind.getLocalName()));
                        }
                    }else{
                        Individual ind =
publicOwl.getIndividual(namespace + ownsVehicle.getText());
                        if(ind==null){
                            JOptionPane.showMessageDialog(this, "Το
όχημα που δήλωσες δεν έχει καταχωρηθεί");
                        }
                        if(ind.isIndividual()){
```

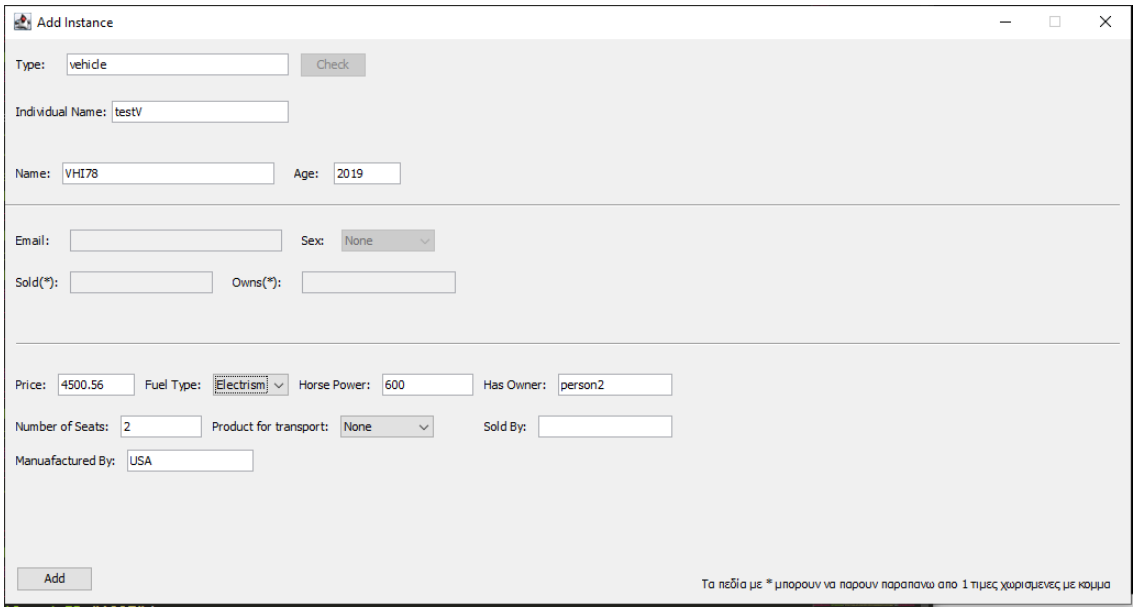
```
                            individual.addProperty(objowns,
publicOwl.createResource(namespace+ind.getLocalName()));
                        }
                    }
                }
                if(owner_sex.getSelectedItem().toString()!="None"){
                    individual.addProperty(dataSex, (String)
owner_sex.getSelectedItem());
                }
                try {
                    publicOwl.prepare();
                    KnowledgeBase kb = ((PelletInfGraph)
publicOwl.getGraph()).getKB();
                    boolean consistent = kb.isConsistent();
                    if (consistent == false) {
                        throw new InconsistentOntologyException();
                    } else {
                        FileWriter out = null;
                        try {
                            out = new FileWriter(path);
                            publicOwl.write(out, "RDF/XML");
                        } catch (IOException ex) {

Logger.getLogger(AddInstance.class.getName()).log(Level.SEVERE, null,
ex);
                        }
                        for (JTextField t : widgets) {
                            t.setText("");
                            t.setEnabled(false);
                        }
                        owner_sex.setEnabled(false);
                    }
                } catch (InconsistentOntologyException e) {
                    JOptionPane.showMessageDialog(this, e);
                    this.dispose();
                }
            }
            checkTypes.setEnabled(true);
            category = null;
            applyBtn.setEnabled(false);
        }

}
```

Όπου παίρνει το κείμενο που έχει γραφτεί στο πεδίο Individual Name, αν είναι κενό τότε εμφανίζεται σχετικό μήνυμα αλλιώς ελέγχω την τιμή της μεταβλητής category . Αν είναι ίση με *«Vehicle»,«Owner»* ή *«Country»* τότε δημιουργώ το στιγμιότυπο με βάση το κείμενο από το παραπάνω πεδίο και παίρνω τις ιδιότητες που το αφορούν με βάση την *category* .Έπειτα ελέγχω για το αν τα πεδία έχουν τιμές. Αν έχουν τότε εισάγω την ιδιότητα στο στιγμιότυπο με την τιμή της από το πεδίο και στο συγκεκριμένο τύπο όπως έχει οριστεί στην οντολογία. Έπειτα εκτελούμε τον reasoner και ελέγχω για ασυνέπειες. Αν υπάρχουν τότε το παράθυρο κλείνει, αλλιώς γράφω το ανανεωμένο μοντέλο στο αρχείο και ενεργοποιείται το κουμπί *«Check»* ,απενεργοποιείται το κουμπί *«Add»* καθώς και τα αντίστοιχα πεδία ώστε ο χρήστης να μπορεί να εισάγει στιγμιότυπο για άλλη κλάση .

Παρακάτω παρουσιάζεται η προσθήκη νέου στιγμιότυπου :



Και στο αρχείο :

```
<rdf:Description rdf:about="http://www.mydomain.com/vehicles/testV">
  <fuelType>Electrism</fuelType>
  <horsePower rdf:datatype="http://www.w3.org/2001/XMLSchema#double">600</horsePower>
  <numOfSeats rdf:datatype="http://www.w3.org/2001/XMLSchema#positiveInteger">2</numOfSeats>
  <Price rdf:datatype="http://www.w3.org/2001/XMLSchema#double">4500.56</Price>
  <hasOwner rdf:resource="http://www.mydomain.com/vehicles/person2"/>
  <manufacturedBy rdf:resource="http://www.mydomain.com/vehicles/USA"/>
  <ModelAge rdf:datatype="http://www.w3.org/2001/XMLSchema#positiveInteger">2019</ModelAge>
  <ModelName rdf:datatype="http://www.w3.org/2000/01/rdf-schema#Literal">VHI78</ModelName>
  <rdf:type rdf:resource="http://www.mydomain.com/vehicles/Vehicle"/>
</rdf:Description>
```