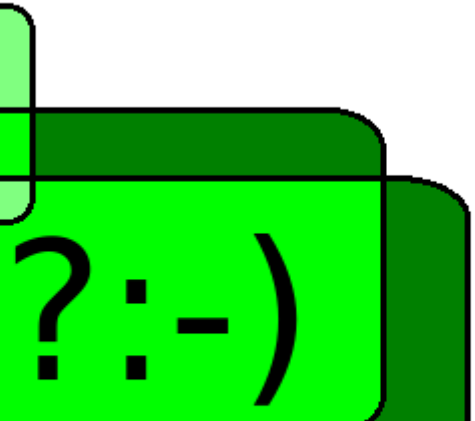
Deklarativno programiranje

Laboratorijske vježbe

Izv. prof. dr. sc. Markus Schatten

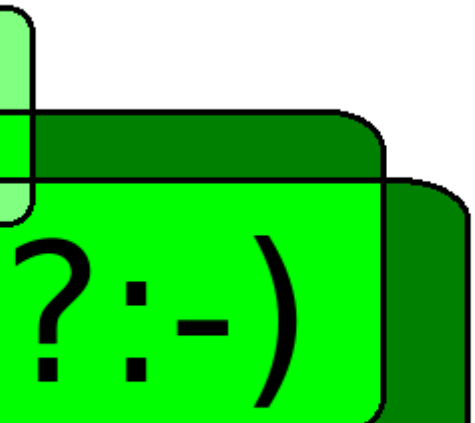
Prezentacija #8



F-logika

engl. **Frame logic** – Logika temeljena na okvirima

Objektno – orijentirani jezik za baze znanja, ontologije, semantički web, inteligentne agente ...



Implementacije

FLORA-2/Ergo Lite – OpenSource (implementirana u XSB Prologu)

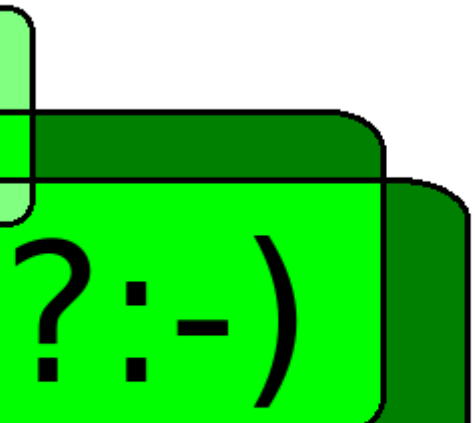
FLORID - samo u istraživačke i edukacijske svrhe

ONTOBROKER, SILK, Ergo.AI – komercijalni alati

F-molekule

Svaki izraz oblika

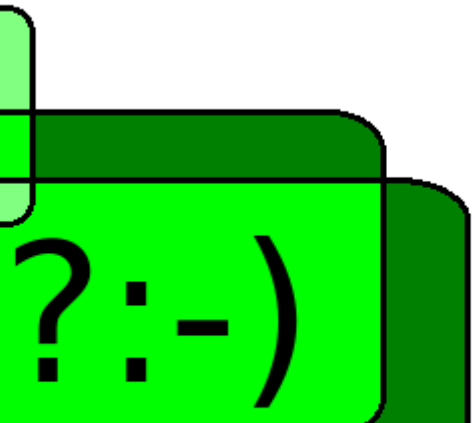
```
naziv_objekta : naziv_klase
```



F-molekule

Svaki izraz oblika

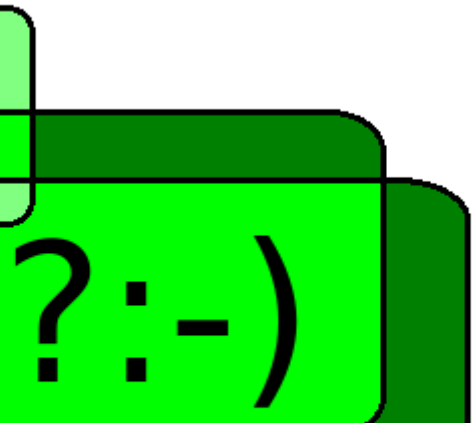
```
naziv_objekta : naziv_klase[

].
```



F-molekule

Svaki izraz oblika

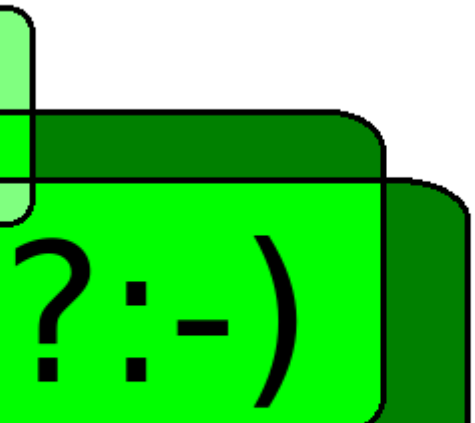
```
naziv_objekta : naziv_klase[
   naziv_atributa_1 -> vrijednost_atributa_1


].
```



F-molekule

Svaki izraz oblika

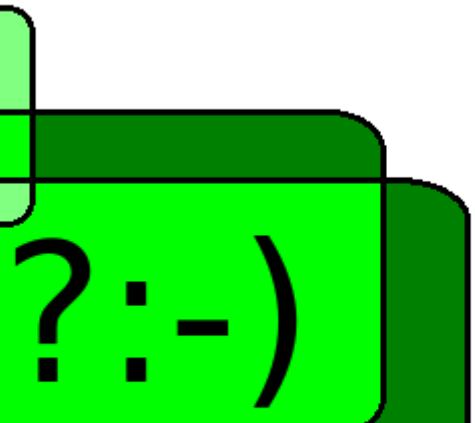
```
naziv_objekta : naziv_klase[
   naziv_atributa_1 -> vrijednost_atributa_1,
   ... ,
   naziv_atributa_n -> vrijednost_atributa_n,



].
```



F-molekule

Svaki izraz oblika

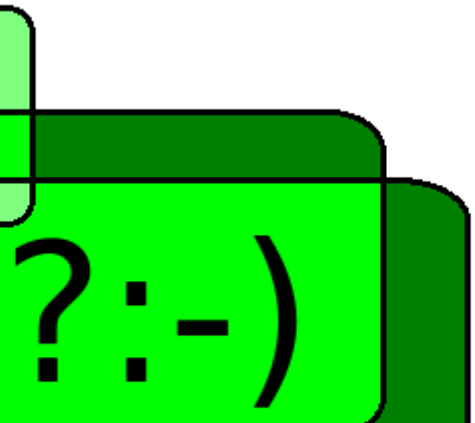
```
naziv_objekta : naziv_klase[
   naziv_atributa_1 -> vrijednost_atributa_1,
   ... ,
   naziv_atributa_n -> vrijednost_atributa_n,
   naziv_metode_1( parametri_1 ) -> rezultat_1


].
```



F-molekule

Svaki izraz oblika

```
naziv_objekta : naziv_klase[
   naziv_atributa_1 -> vrijednost_atributa_1,
   ... ,
   naziv_atributa_n -> vrijednost_atributa_n,
   naziv_metode_1( parametri_1 ) -> rezultat_1,
   ... ,
   naziv_metode_m( parametri_m ) -> rezultat_m
].
```

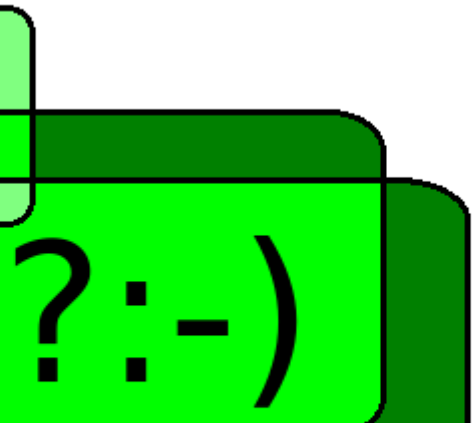


Semantika

Objekt pod nazivom (objekt ID-em) **`naziv_objekta`** koji je instanca klase **`naziv_klase`** ima za atribut pod nazivom **`naziv_atributa_1`** vrijednost **`vrijednost_atributa_1`**, ..., za atribut pod nazivom **`naziv_atributa_n`** vrijednost **`vrijednost_atributa_n`** te pozivom metode **`naziv_metode_1`** s paramterima **`parametri_1`** dobiva se rezultat **`rezultat_1`**, ..., te pozivom metode **`naziv_metode_m`** s paramterima **`parametri_m`** dobiva se rezultat **`rezultat_m`**.

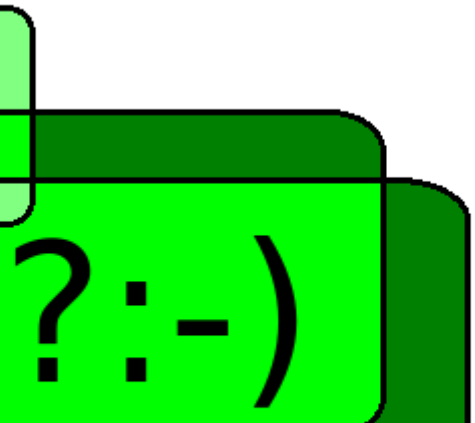
Primjer

```
ivek : doktor [
    ime -> ivan,
    prezime -> presvetli,
    godiste -> 1971,
    radno_vrijeme -> od_do( 7, 15 ),
    pacijenti -> { joza, bara }
].
```



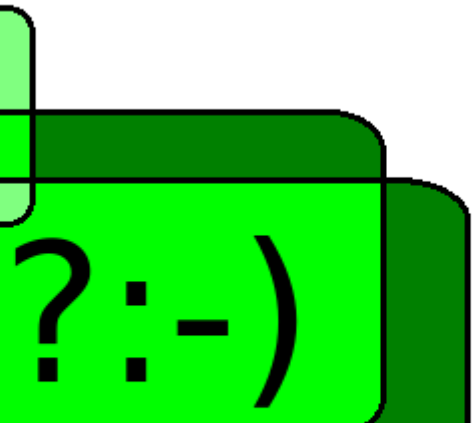
Definicija sheme baze znanja

koristi se operator => umjesto operatora -> te se naznačuju tipovi podataka



Primjer

```
osoba[
    ime => \string,
    prezime => \string,
    adresa => \string,
    godiste => \integer,
    starost( \integer ) => \integer
].
```

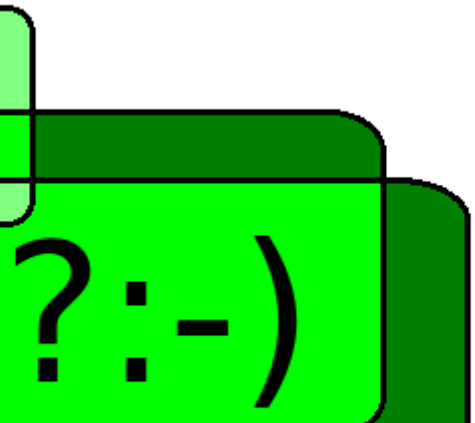


Hijerarhija klasa

Izraz oblika:

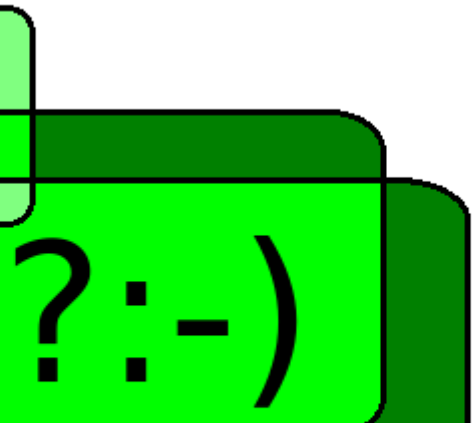
```
klasa_p :: klasa_n.
```

Znači da je klasa pod nazivom **`klasa_p`** podklasa klase pod nazivom **`klasa_n`**, odnosno da je **`klasa_n`** nadklasa klase **`klasa_p`**.



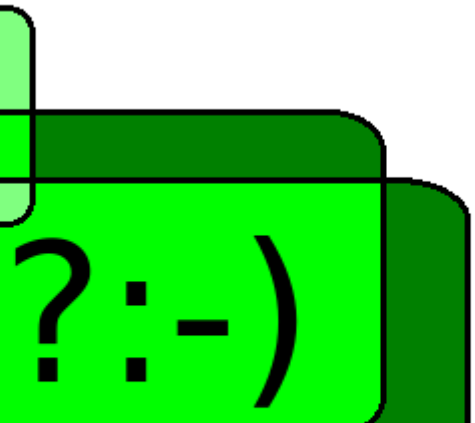
Primjer

```
doktor :: osoba[
    radno_vrijeme => struct,
    pacijenti => pacijent
].
```



Metode

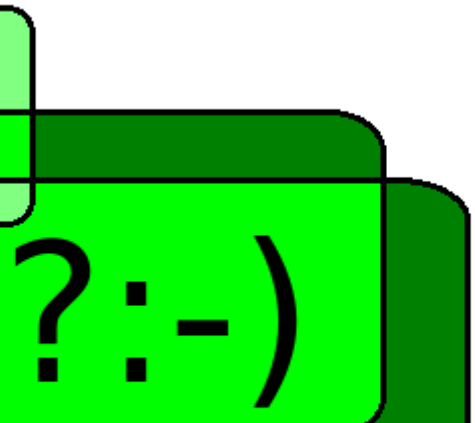
Metode definiramo u obliku pravila



Primjer

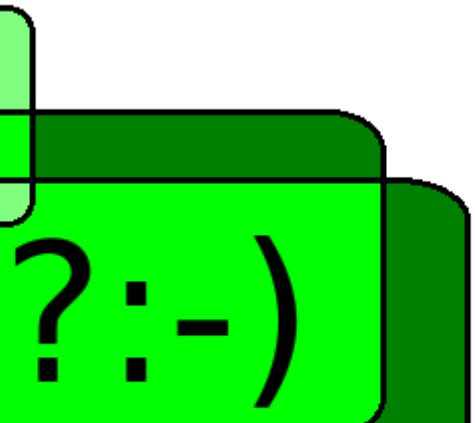
```
?osoba[ starost( ?sada ) -> ?starost ] :-
    ?osoba[ godiste -> ?godiste ],
    ?starost \is ?sada - ?godiste.
```

Uočite da se u za varijable koristi znak upitnik!

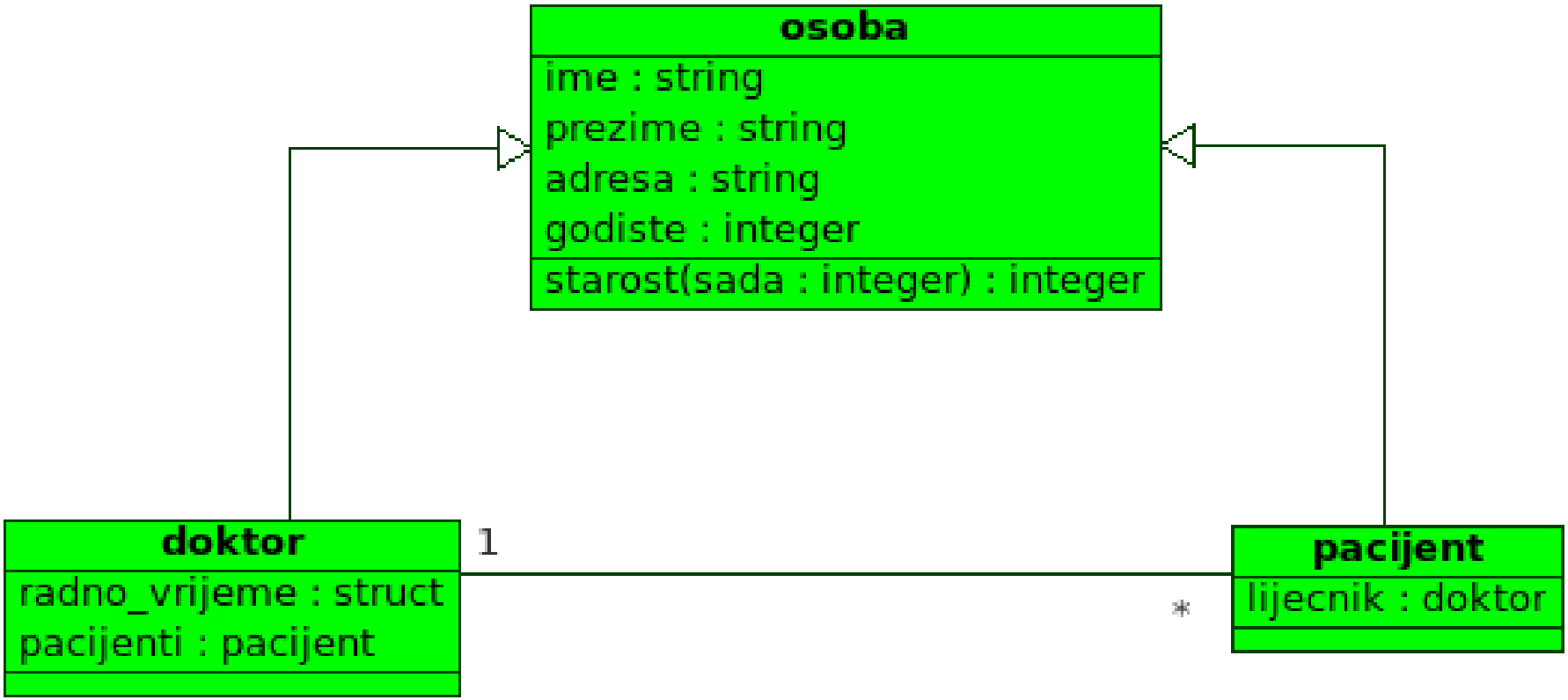


Upute

- U nastavku ćemo implementirati bazu znanja vezanu uz doktore i pacijente.
- Kreirajte datoteku doktori.flr (mora biti nastavak .flr kako bi Emacs pokrenuo Flora-2 mode) te u nju upisujte definicije klasa, metoda i objekata.



Primjer baze znanja

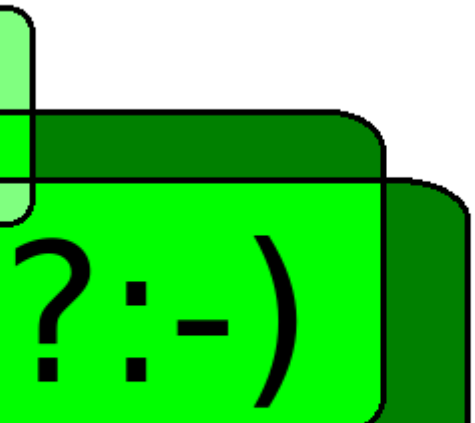


doktori.flr

```
osoba[
    ime => \string,
    prezime => \string,
    adresa => \string,
    godiste => \integer,
    starost( \integer ) => \integer
].
```

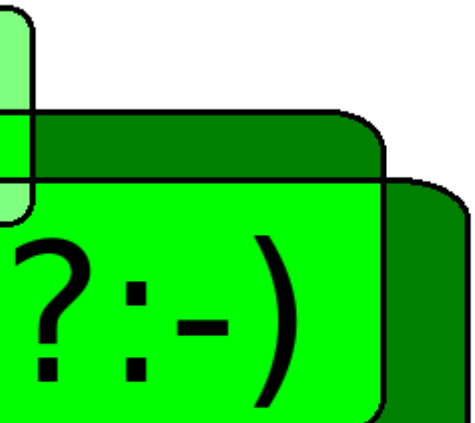
doktori.flr

```
?osoba[ starost( ?sada ) -> ?starost ] :-
    ?osoba[ godiste -> ?godiste ],
    ?starost \is ?sada - ?godiste.
```



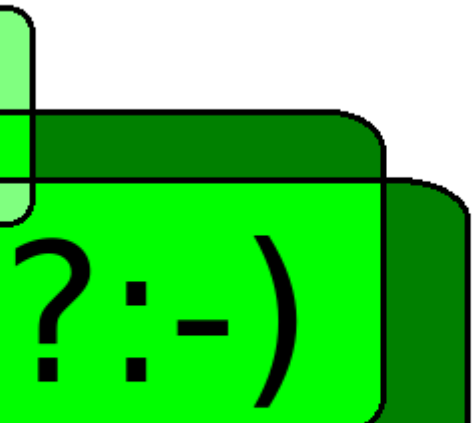
doktori.flr

```
doktor :: osoba[
    radno_vrijeme => struct,
    pacijenti => pacijent
].
pacijent :: osoba[
    lijecnik => doktor
].
```



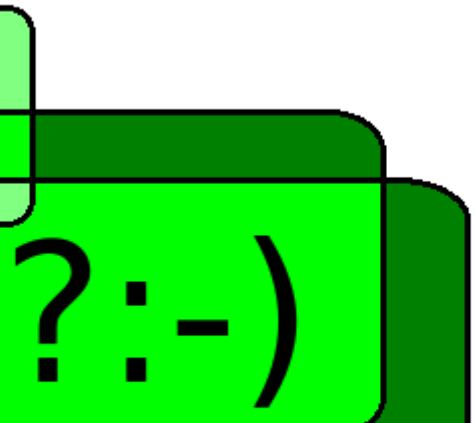
doktori.flr

```
ivek : doktor [
    ime -> Ivan,
    prezime -> Presvetli,
    godiste -> 1971,
    radno_vrijeme -> od_do( 7, 15 ),
    pacijenti -> { joza, bara }
].
```



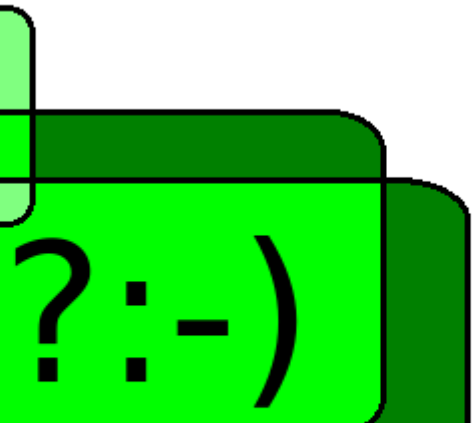
doktori.flr

```
bara : pacijent[
    ime -> Barica,
    prezime -> Jambrek,
    godiste -> 1975,
    lijecnik -> ivek
].
```



doktori.flr

```
joza : pacijent[
    ime -> Joza,
    prezime -> Stefanec,
    godiste -> 1965,
    lijecnik -> ivek
].
```



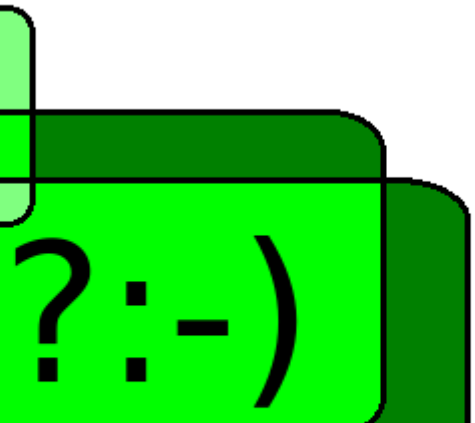
Upute

Za učitavanje modula u Emacsu koristimo kombinaciju tipki C-c C-b (CTRL+c CTRL+b).

Sustav tada pita u koji modul želimo učitati trenutnu datoteku te nudi modul main. Za naše potrebe možemo prihvatiti modul main pritiskom na tipku ENTER.

Nakon toga učitava se Flora-2 konzola.

Upite u nastavku upisujte u konzolu, a kopiju konzole pohranite u datoteku ime_prezime.txt



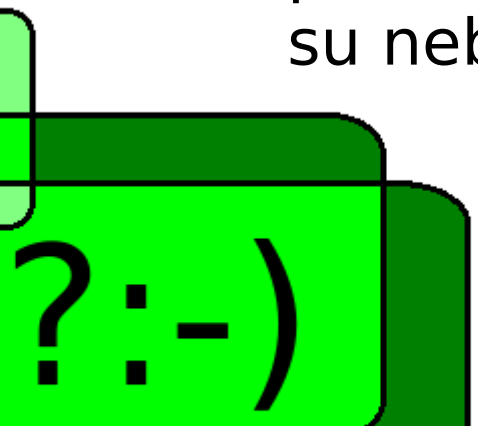
Primjeri

Upit koji vraća imena i prezimena svih osoba.

```
?_ : osoba[
    ime -> ?ime,
    prezime -> ?prezime ].
```

Uočite da u bazi znanja niti jedna osoba nije izravno definirana! Sustav je zaključio putem hijerarhije klasa da su svi pacijenti i svi doktori osobe.

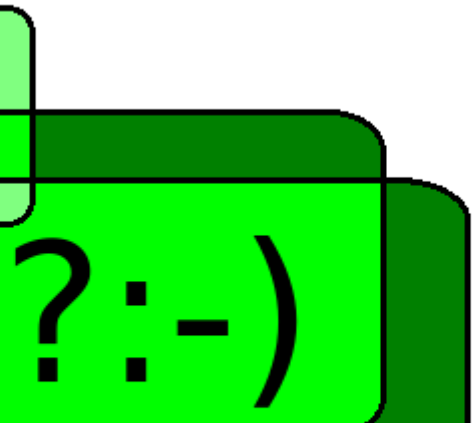
Također, za razliku od Prolog-a, varijable se pišu s prefiksom ?. U skladu s time varijable s prefiksom ?_ su nebitne (engl. don’t care) varijable.



Primjeri

Upit koji vraća prezime i starost svih pacijenata

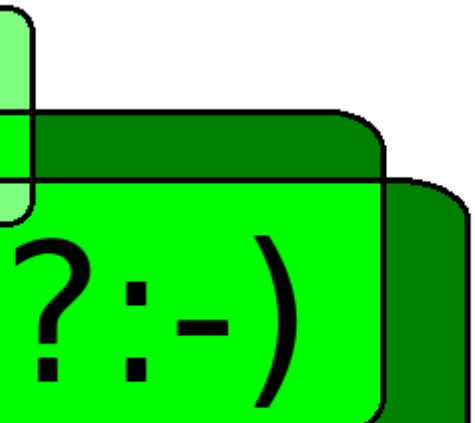
```
?_ : pacijent[
  prezime -> ?prezime,
  starost( 2019 ) -> ?starost ].
```



Primjeri

Upit koji vraća imena svih pacijenata starijih od 50 godina

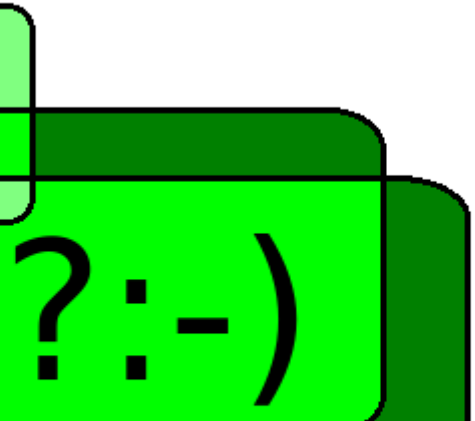
```
?_ : pacijent[
        ime -> ?ime,
        starost( 2019 ) -> ?_s ],
         ?_s > 50.
```



Primjeri

Upit koji vraća imena i prezimena svih pacijenata doktora Presvetli.

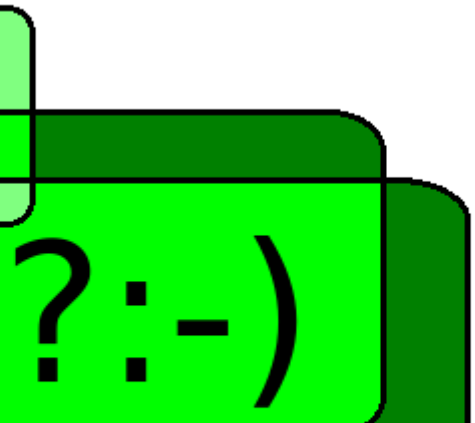
```
?_ : doktor[
     prezime -> Presvetli,
     pacijenti -> ?_pac ],
?_pac[
     ime -> ?ime,
     prezime -> ?prezime ].
```



Primjeri

ili

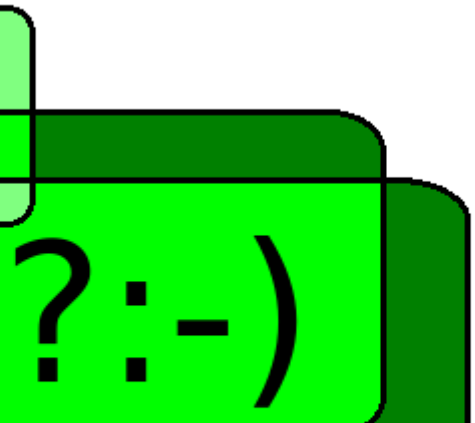
```
(?_dok : doktor[
  prezime -> Presvetli ]).pacijenti[
            ime -> ?ime,
            prezime -> ?prezime ].
```



Primjeri

Upit koji vraća radno vrijeme doktora iveka

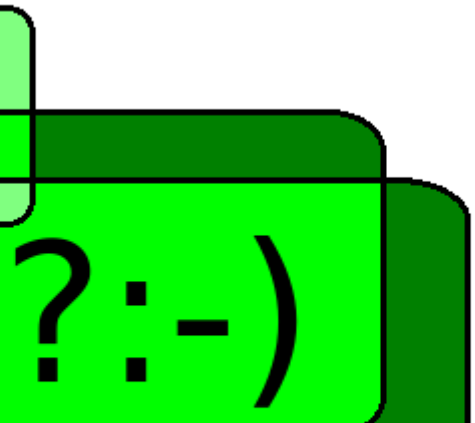
```
ivek[
   radno_vrijeme -> od_do( ?od, ?do )
   ].
```



Primjeri

ili

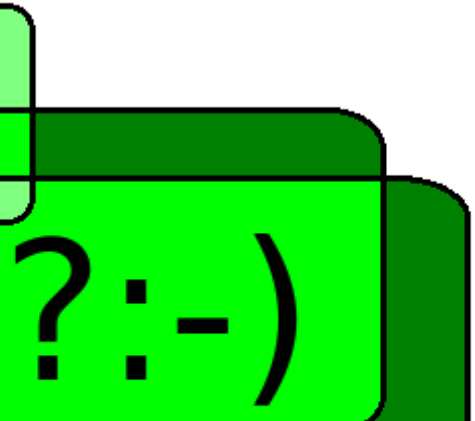
```
ivek.radno_vrijeme = od_do( ?od, ?do ).
```



Primjeri

Upit koji vraća radno vrijeme svih doktora

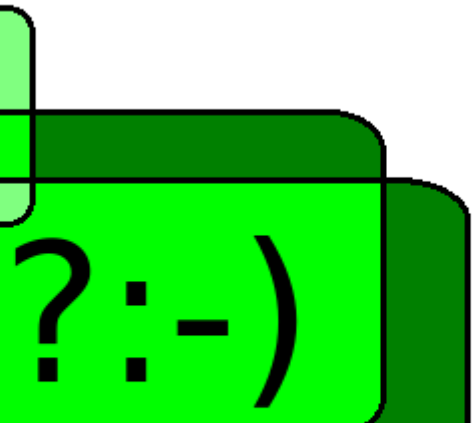
```
?_ : doktor[
     radno_vrijeme -> od_do( ?od, ?do ) ].
```



Primjeri

ili

```
( ?_ : doktor ).radno_vrijeme =
                od_do( ?od, ?do ).
```



Agregirajuće (skupovne) operacije

- imaju oblik:

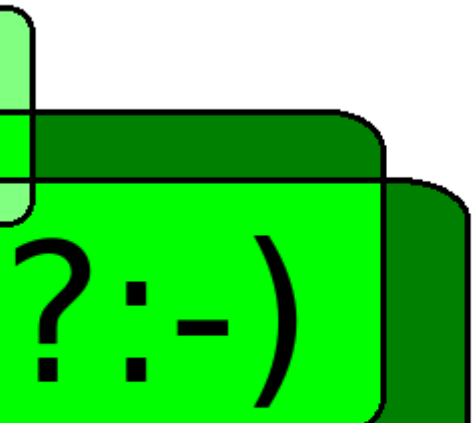
```
agg{?X[?Gs] | query}
```

- Pri čemu je:
  - **`agg`** – skupovni operator
  - **`?X`** – varijabla agregacije
  - **`[?Gs]`** – lista grupirajućih varijabli (opcionalno)
  - **`query`** – upit po kojem se obavlja grupiranje

Pravila

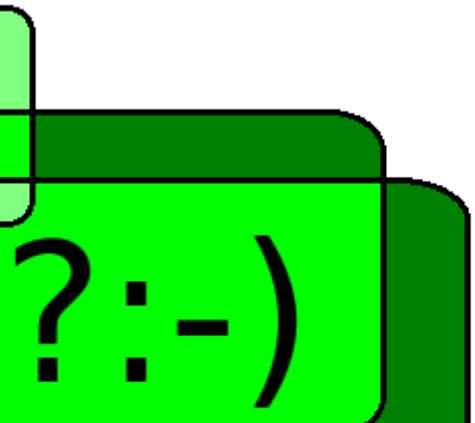
- Upit mora sadržavati **sve** varijable iz **`[?Gs]`** uključujući i varijablu **`?X`**

- **`[?Gs]`** ne smije sadržavati **`?X`**



Skupovni operatori

- Flora2 podržava sljedeće agregirajuće operatore:
  - **min** – nalazi minimalnu vrijednost
  - **max** – nalazi maksimalnu vrijednost
  - **count** – nalazi ukupan broj rješenja
  - **sum** – vraća sumu rješenja (samo brojevi)
  - **avg** – vraća prosjek rješenja (samo brojevi)
  - **collectbag** – vraća listu svih rješenja
  - **collectset** – vraća listu rješenja bez ponavljanja



Upute

- U nastavku ćemo kreirati jednu vrlo jednostavnu bazu znanja kako bismo isprobali agregirajuće operacije i module.
- Kreirajte novu datoteku agg.flr i u nju upisujte opisnike objekata.
- Nakon što ste upisali objekte, osim pokretanja pohranite datoteku u direktoriju u kojem se izvodi Emacs (C-x C-s, CTRL+x CTRL+s) kako bismo je izravno mogli učitati u konzolu.
- Ako datoteka nije pohranjena u trenutnom direktoriju, morat ćete do nje pristupiti upisivanjem putanje do datoteke.
- Upite i isprobavanja na konzoli pridodajte u datoteku ime_prezime.txt

Primjer

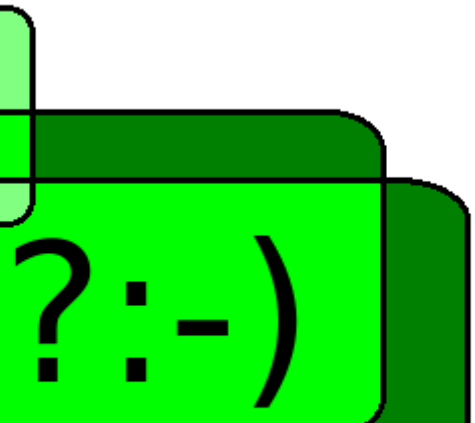
- Neka je zadana sljedeća baza znanja (**agg.flr**):

```
ivek:osoba[ godine->32, spol->musko ].
joza:osoba[ godine->31, spol->musko ].
bara:osoba[ godine->25, spol->zensko ].
```

Primjer

- Upit koji vraća prosječnu starost osoba

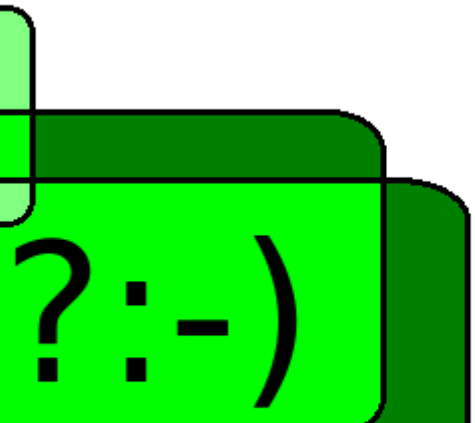
```
?x = avg{ ?_g | ?_:osoba[ godine->?_g ] }.
```



Primjer

- Upit koji vraća prosječnu starost osoba grupirano prema spolu

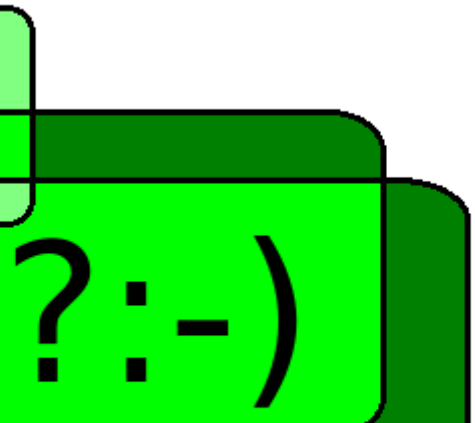
```
?x = avg{
      ?_g [ ?s ] |
      ?_:osoba[
        godine->?_g,
        spol->?s ]
}.
```



Primjer

- Upit koji vraća skupove (liste) osoba grupirane prema spolu

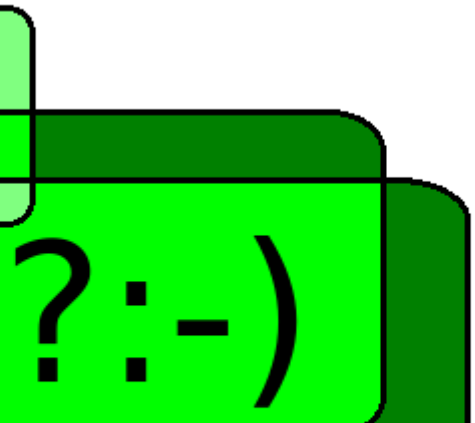
```
?x = collectset{ ?_o [ ?s ] |
    ?_o:osoba[ spol->?s ] }.
```



Moduli i rad s modulima

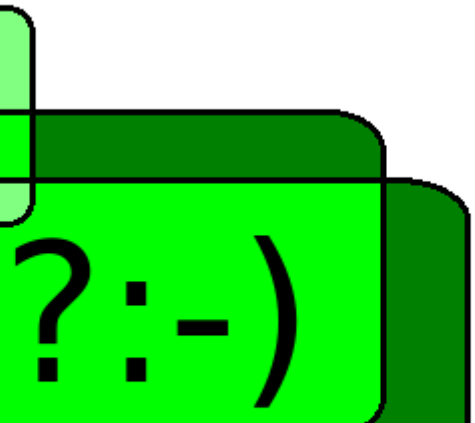
- Moduli su programska apstrakcija koja se sastoji od naziva i sadržaja.

- Omogućavaju nam fleksibilno djeljenje programskog koda u više datoteka.



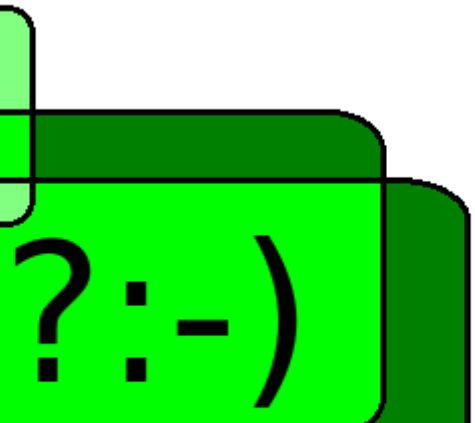
- Za učitavanje modula koristimo sintaksu:

```
[datoteka>>modul].
```



Primjer

```
[agg>>go1].
```

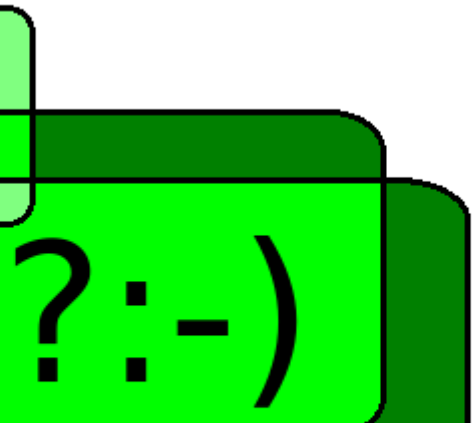


Primjer

```
[agg>>go1].
```

Na taj je način isti modul moguće više puta učitati pri čemu je svaki modul instanca za sebe

```
[agg>>go2].
```

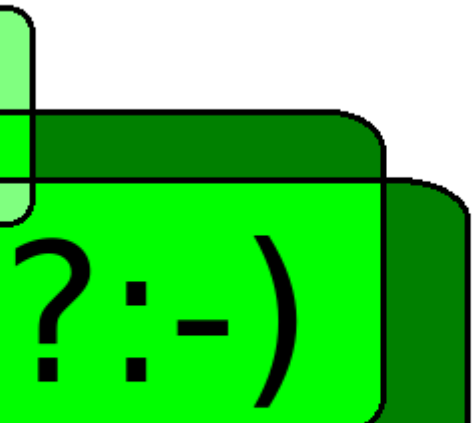


- Za postavljanje upita nad modulima koristimo sintaksu:

```
upit @ modul.
```

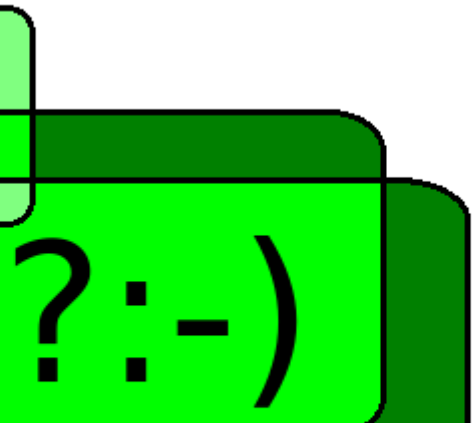
- Primjer.

```
?x:osoba @ go1.
```



- Moguće je i postaviti upit koji nam vraća naziv modula u kojem vrijedi upit, npr.

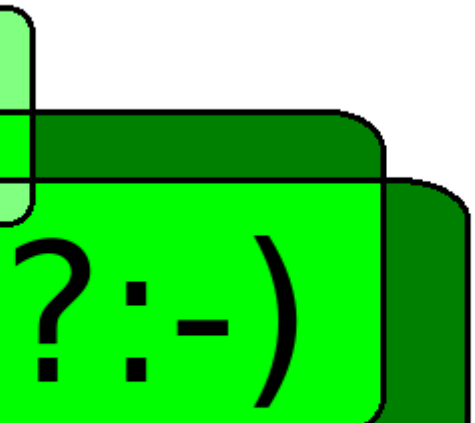
```
?x:osoba @ ?modul.
```



Modul \prolog

- Specifičan je modul \prolog koji nam omogućava interakciju s predikatima iz XSB Prologa.

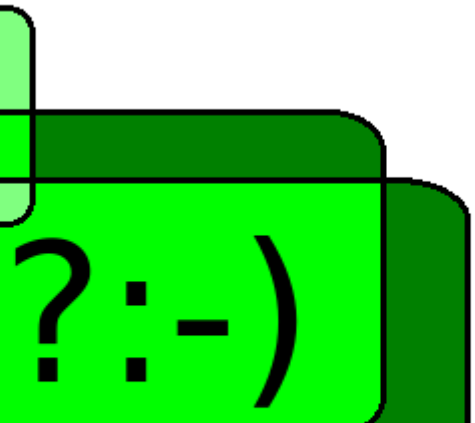
```
writeln( 'Pozdrav iz Prologa' )@\prolog.
```



Modul \prolog

- Za pozivanje predikata iz XSB Prolog modula koristi se sintaksa

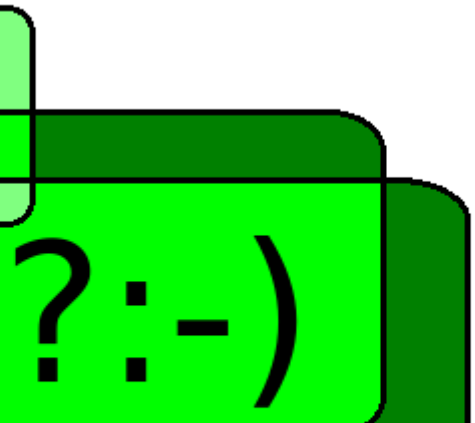
```
predikat@\prolog(naziv_modula).
  predikat@\plg(naziv_modula).
```



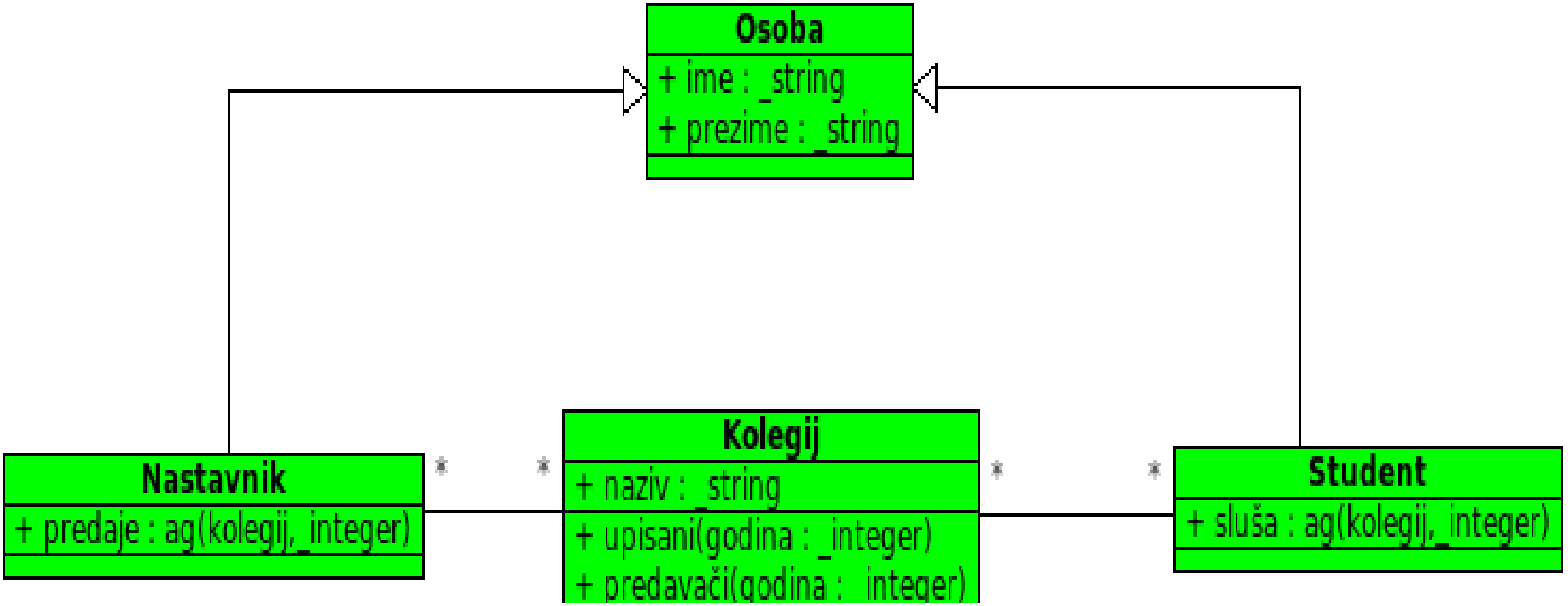
Primjer

```
?x = [1,2,3], member(?y,?x)@\plg(basics).
```

Uočite da na taj način možete koristiti sve ugrađene predikate i sve module koje smo do sada koristili!



Zadatak


Osoba
+ ime : _string
+ prezime : _string
Nastavnik
+ predaje : ag(kolegij, integer)
Kolegij
+ naziv : _string
+ upisani(godina : _integer)
+ predavači(godina : integer)
Student
+ sluša : ag(kolegij, integer)

Zadatak

- Kreirajte bazu znanja o nastavnicima i studentima koja je opisana u prethodnom UML dijagramu klasa te ju pohranite u datoteku kolegiji.flr
- Potrebno je implementirati:
  - Strukturu baze znanja (definiciju klasa i pravila) – atributi predaje: ag(kolegij, godina) odnosno sluša: ag(kolegij, godina) imaju sljedeću semantiku: nastavnik predaje određeni kolegij određene akademske godine odnosno student sluša određeni kolegij određene akademske godine
  - Dvije metode (pravila) upisani/1 i predavači/1 klase kolegij koji primaju akademsku godinu (cjelobrojna vrijednost) te vraćaju listu upisanih studenata odnosno listu predavača na tom kolegiju zadane akademske godine

Comic relief

https://tinyurl.com/nauci-prog