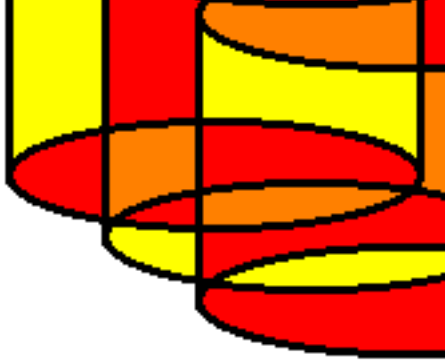
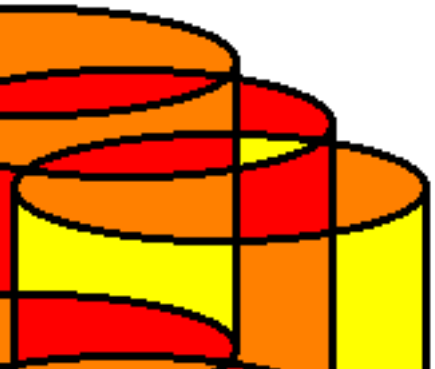
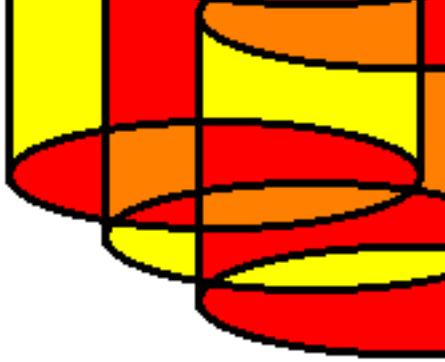
Objektno-orijentirane deduktivne baze podataka



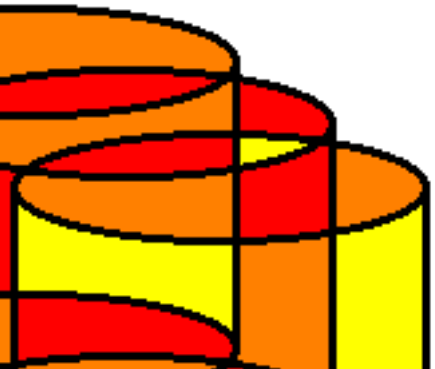
- Spoj objektno-orijentiranog pristupa s deduktivnim bazama podataka
- F-logika – logika temeljena na okvirima
- FLORA-2 – sustav koji implementira F-logiku, transakcijsku logiku i HiLog



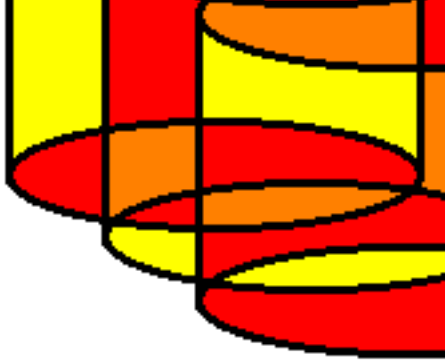
Uvod



- Flora-2, kao i svaki objektno-orijentirani sustav radi s klasama i objektima koji su opisani putem tzv. F-molekula
- Svaka F-molekula se sastoji od identiteta objekta (naziva), opcionalno klase u koju objekt spada, niza atributa, opcionalno niza metoda te na kraju opcionalno modul u kojem objekt vrijedi.

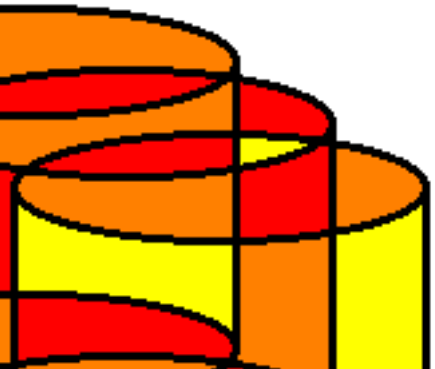


F-molekule

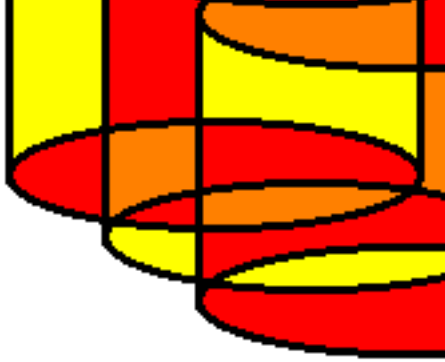


Svaki izraz oblika

```
naziv_objekta : naziv_klase
```



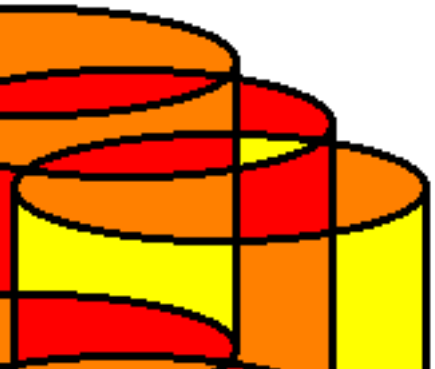
F-molekule



Svaki izraz oblika

```
naziv_objekta : naziv_klase[
```

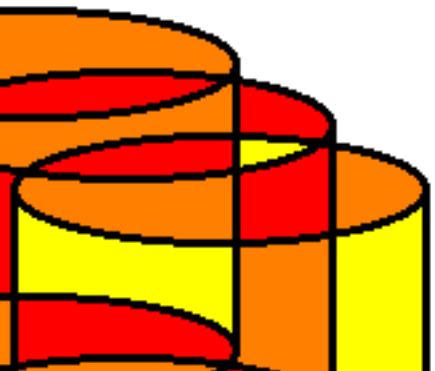
```
]@modul.
```



F-molekule

Svaki izraz oblika

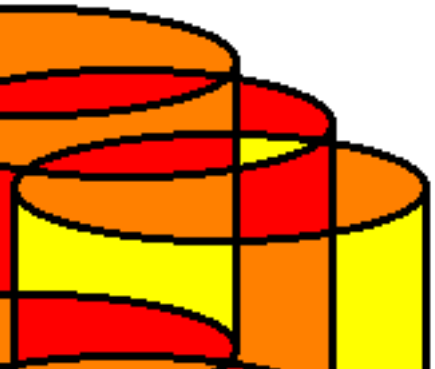
```
naziv_objekta : naziv_klase[
   naziv_atributa_1 -> vrijednost_atributa_1

]@modul.
```



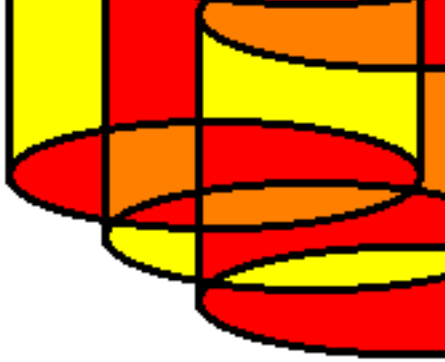
F-molekule

Svaki izraz oblika

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


]@modul.
```

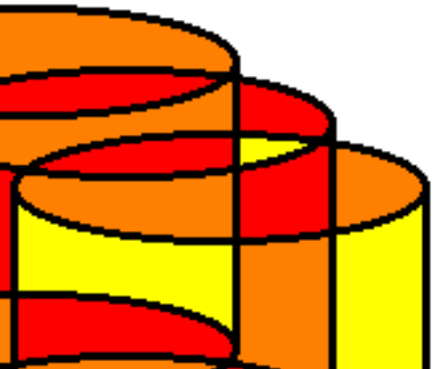


F-molekule



Svaki izraz oblika

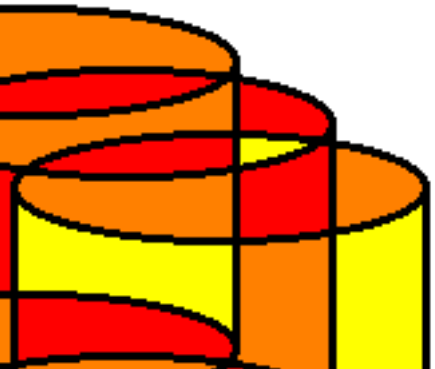
```
naziv_objekta : naziv_klase[
   naziv_atributa_1 -> vrijednost_atributa_1,
   ... ,
   naziv_atributa_n -> vrijednost_atributa_n,
   naziv_metode_1( parametri_1 ) -> rezultat_1


]@modul.
```



F-molekule

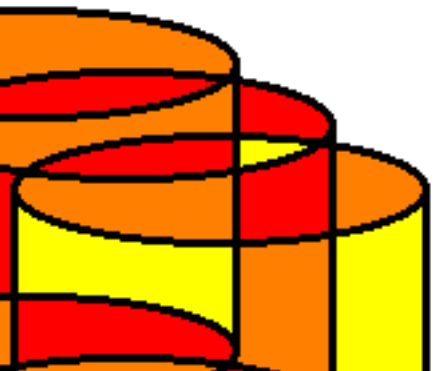
Svaki izraz oblika

```
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
]@modul.
```

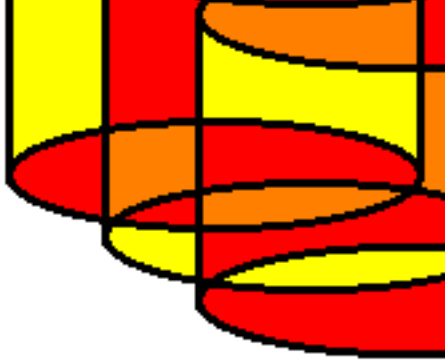


Primjer

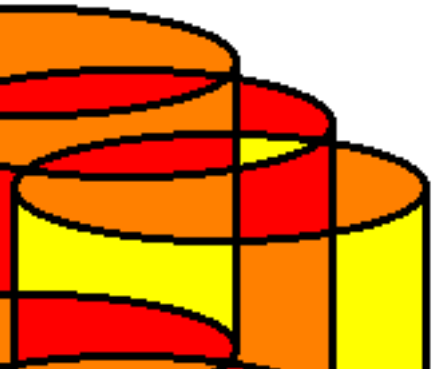
```
Ivek123:osoba[
    ime->Ivan,
    prezime->Presvetli,
    email->'ivek@gmail.com',
    prijatelji->{ana_x, joza98}
]@drustvena_mreza.
```



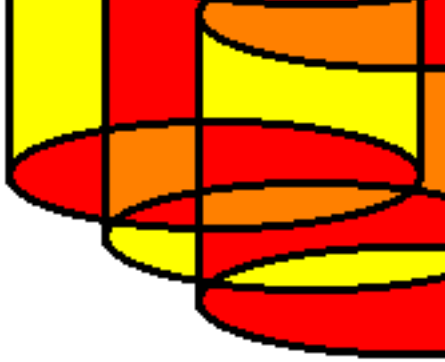
Primjer



- U primjeru imamo objekt čiji je identitet **Ivek123**
- Objekt je instanca klase **osoba**
- Ima četiri atributa (**ime**, **prezime**, **email**, **prijatelji**) čije su vrijednosti respektivno **Ivan**, **Presvetli**, ‘**ivek@gmail.com**’ te skup koji se sastoji od elemenata **ana_x** i **joza98** koji najvjerojatnije predstavljaju druge objekte.
- Objekt je definiran u modulu **drustvena_mreza**



Izrazi hijerarhije klasa

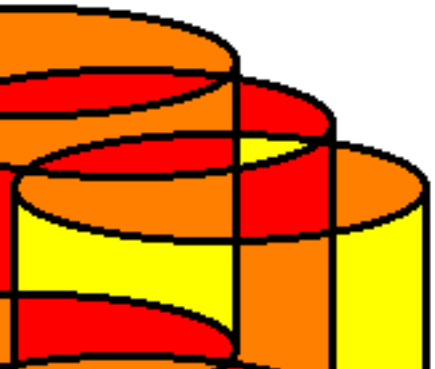


Osim F-molekula imamo i izraze hijerarhije klasa, koji nam omogućuju da modeliramo klase kao podklase drugih, npr.

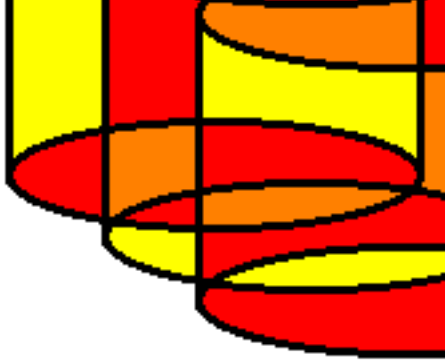
`podklasa::nadklasa@modul.`

Pri čemu je modul opcionalan

`prijatelj::osoba@drustvena_mreza.`



Pravila

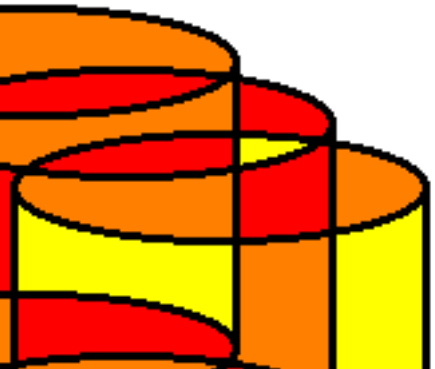


- Dodatno imamo mogućnost implementacije deduktivnih pravila, koja su izrazi oblika:

`glava :- tijelo.`

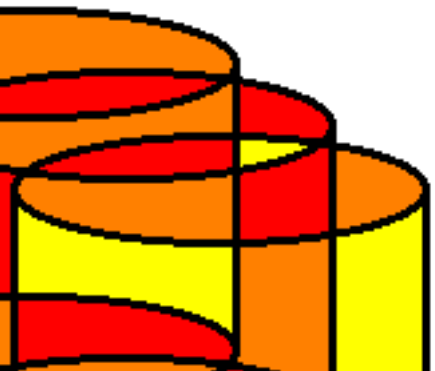
odnosno

`(glava :- tijelo)@modul.`

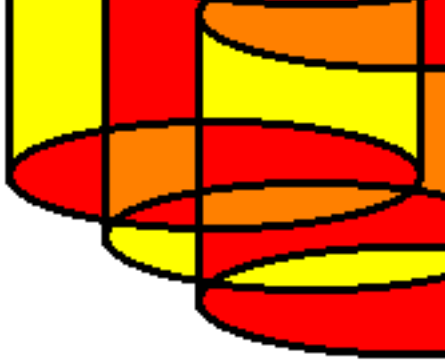


Primjer

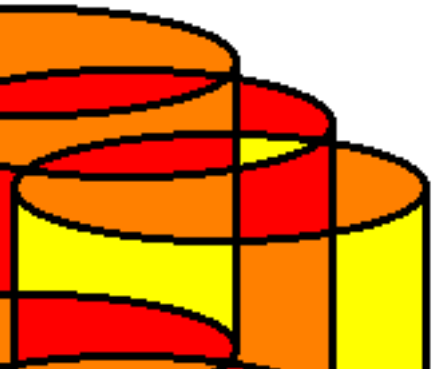
```
(
 ?x:prijatelj :-
      ?x:osoba,
      ?_[ prijatelj->?x ]
)@drustvena_mreza.
```



Primjer



- U ovom primjeru definiramo pravilo koje kaže da će neki objekt (označen varijablom **?x**) biti instanca klase **prijatelj** ako je (**:-** je oznaka implikacije) taj isti objekt instanca klase **osoba** i ako postoji bilo kakav objekt (označen tzv. nebitnom varijablom **?_**) koji za vrijednost svog atributa **prijatelj** ima objekt označen varijablom **?x**
- Ako se navodi modul potrebno je cijelo pravilo staviti u zagradu!

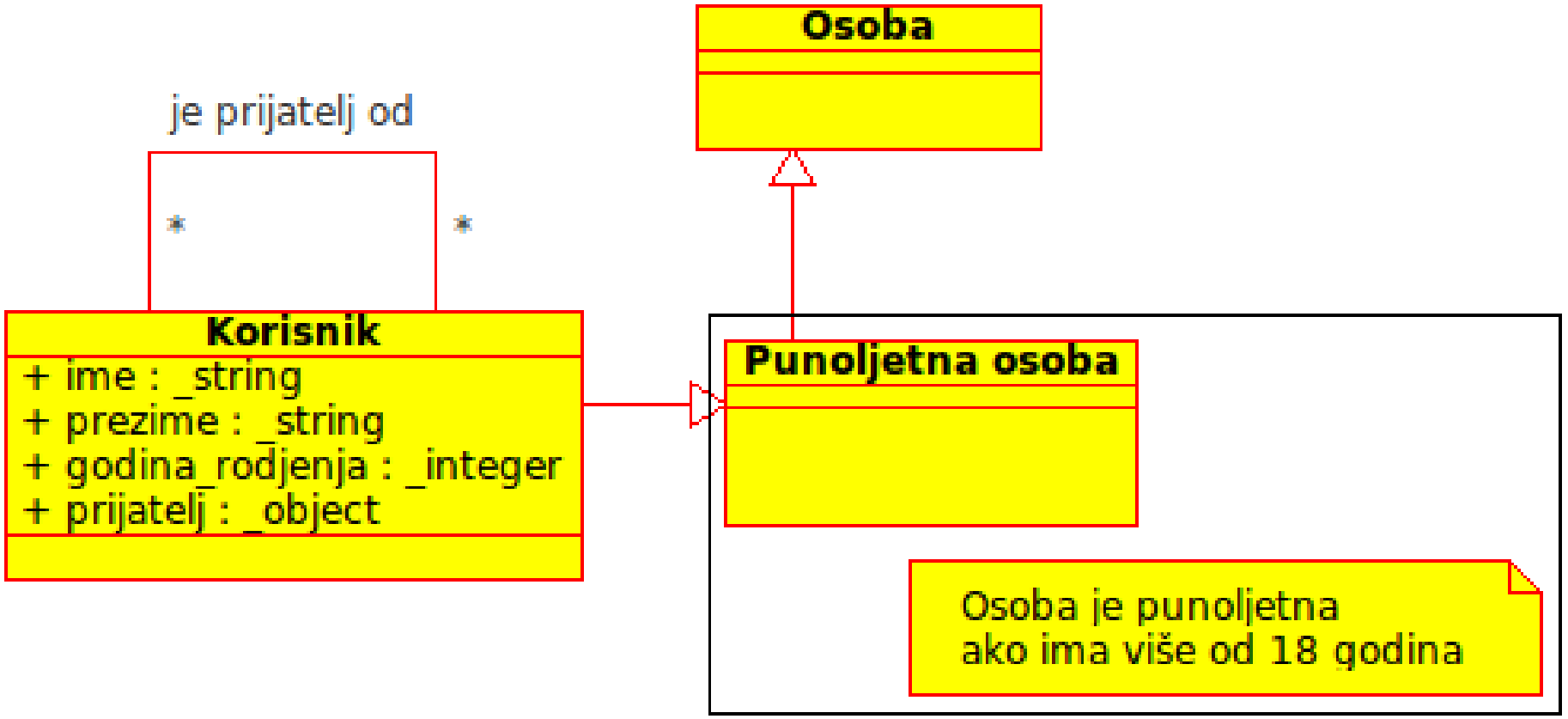


Upute

- Za primjer ćemo kreirati jednu OOD bazu podataka za društvene mreže.
- Iz konzole pokrenite emacs*

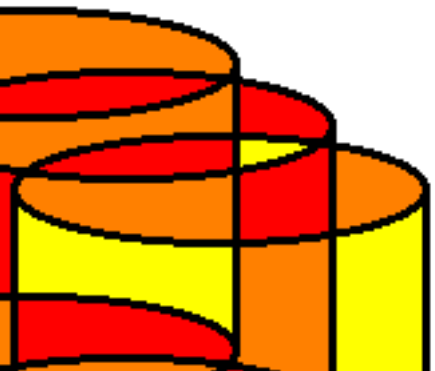
  emacs &
- Pod **Options** kliknite na **Use CUA keys** kako biste aktivirali uobičajene kombinacije tipki za kopiranje, rezanje, ljepljenje (CTRL+c, CTRL+x, CTRL+v)
- Kreirajte novu datoteku **mreza.flr** (nastavak mora biti **.flr** da bi se aktivirao FLORA-2 mode u emacsu)
- U datoteku se upisuje kod koji je naveden u nastavku, a s kombinacijom tipki CTRL+c CTRL+b se kompajlira kod. Emacs tada pita u koji modul želite učitati bazu znanja (default je main, ali se može unesti modul po želji) nakon čega stišćemo ENTER
- Tada se baza znanja učitava i pojavljuje se FLORA-2 konzola u kojoj se mogu postavljati upiti.
- **NAPOMENA**: Kompajliranje ne pohranjuje datoteku, da biste pohranili promjene trebate kliknuti na gumb za pohranu ili stisnuti kombinaciju tipki CTRL-x CTRL+s
- Rezultate isprobavanja (kopiju konzole) pospremite u datoteku **ime_prezime.txt**

Primjer – UML model

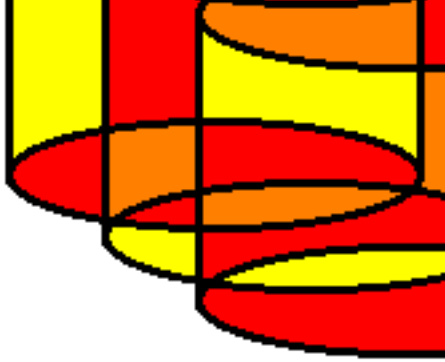


Napomena

- Moguće je definirati shemu baze znanja, ali radi jednostavnosti mi ćemo samo unositi konkretne objekte

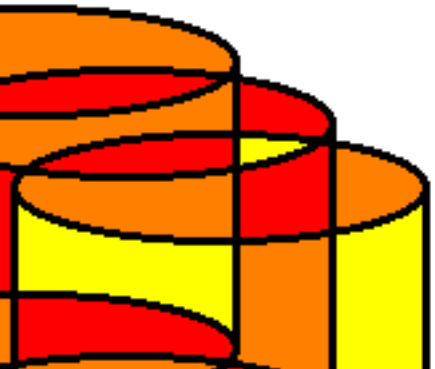


Unos objekata



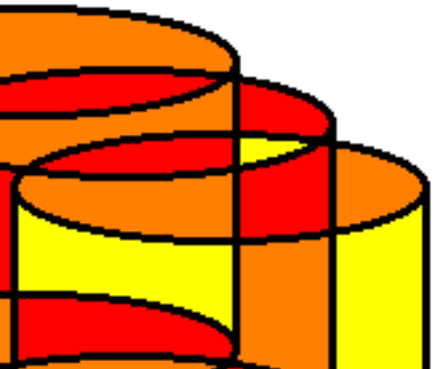
U datoteku mreza.flr unesimo sljedeće objekte:

```
ivek123:Korisnik[
   ime->Ivan,
   prezime->Presvetli,
   godina->1990 ].
```



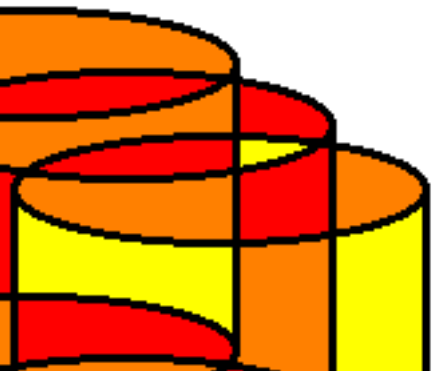
Unos objekata

```
joza_veliki:Korisnik[
   ime->Josip,
   prezime->Prikratki,
   godina_rodjenja->1997,
   prijatelj->bara_xy ].
```



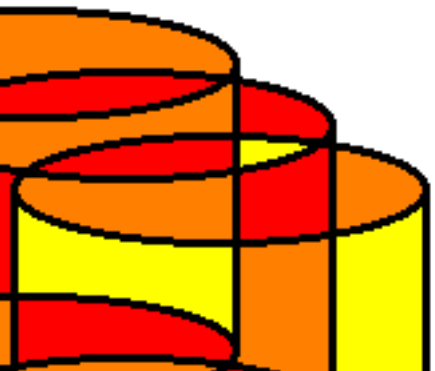
Unos objekata

```
bara_xy:Korisnik[
   ime->Barbara,
   prezime->Jambrescak,
   godina_rodjenja->2010,
   prijatelj->joza_veliki ].
```



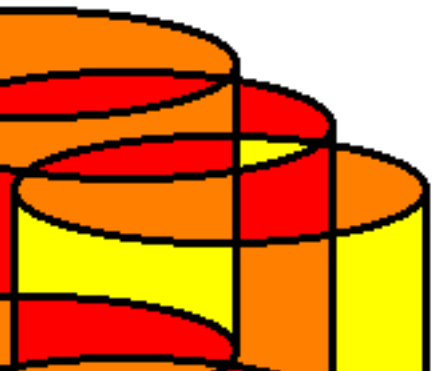
Unos objekata

```
stefa_rulez:Korisnik[
   ime->Stefanija,
   prezime->Prekratki,
   godina_rodjenja->2008,
   prijatelj->{ bara_xy, joza_veliki } ].
```



Upute

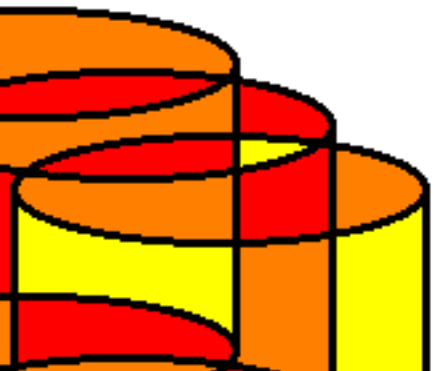
- Nakon unosa objekata kompajlirajmo bazu znanja sa **CTRL+c CTRL+b**
- Učitat ćemo bazu znanja u modul **sm**
- Zatim upite postavljamo na FLORA-2 konzoli



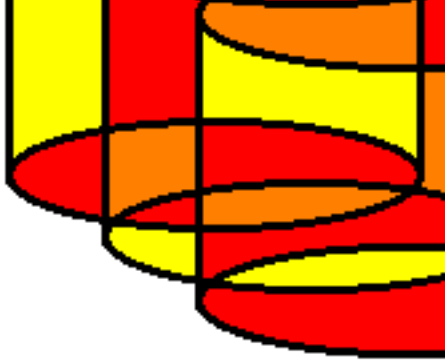
Jednostavni upiti

- Imena i prezimena korisnika

```
?_:Korisnik[ime->?ime, prezime->?prezime]@sm.
```

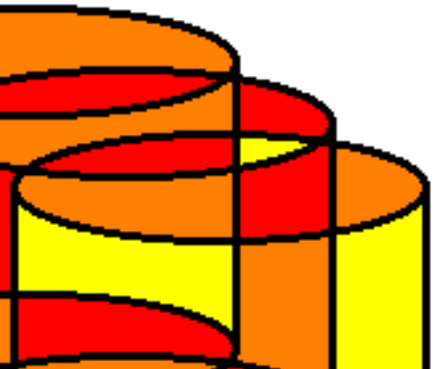


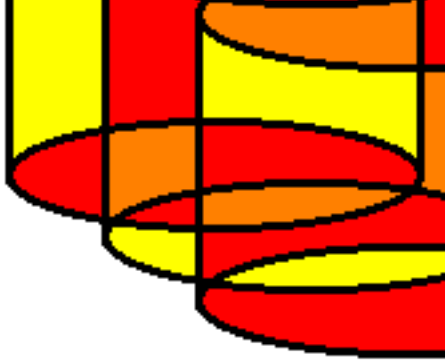
Jednostavni upiti



- Imena i prezimena prijatelja od Štefe

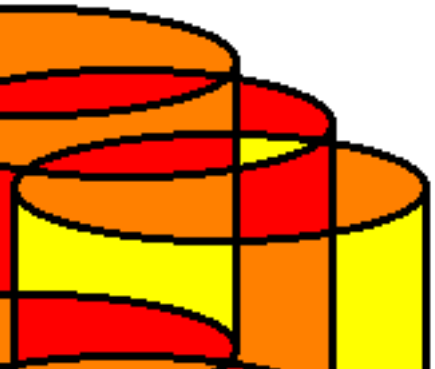
```
stefa_rulez[prijatelj->?_x]@sm,
?_x[ ime->?ime, prezime->?prezime ]@sm.
```

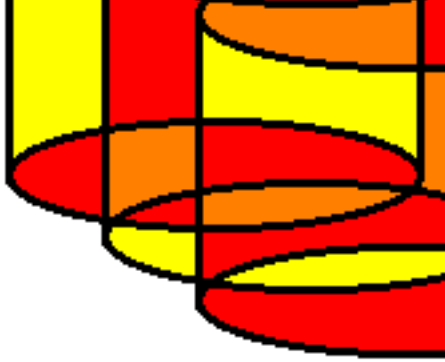




- Ili kraće:

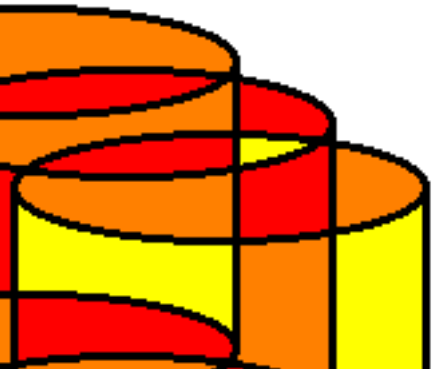
```
(stefa_rulez[prijatelj->?_x],
?_x[ ime->?ime, prezime->?prezime ])@sm.
```



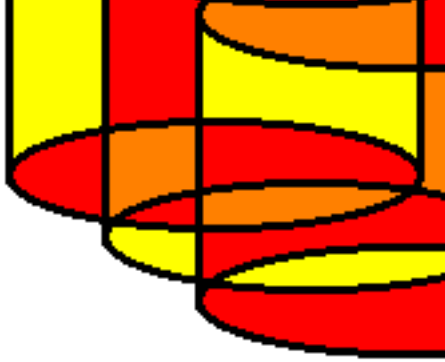


- Ili još kraće:

```
stefa_rulez.prijatelj[
        ime->?ime, prezime->?prezime ]@sm.
```

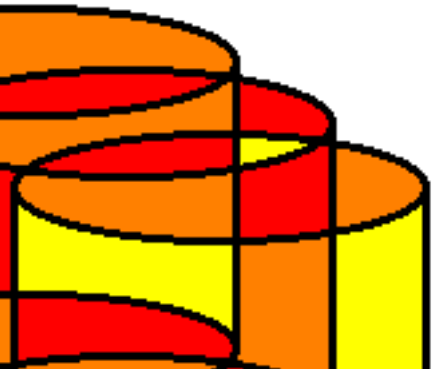


Jednostavni upiti

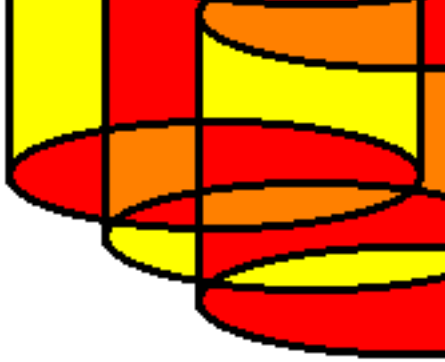


- Imena i prezimena osoba starijih od 18 godina

```
(
?_[ ime->?ime, prezime->?prezime, godina_rodjenja->?_g ],
?god \is 2019 - ?_g,
?god > 18
)@sm.
```



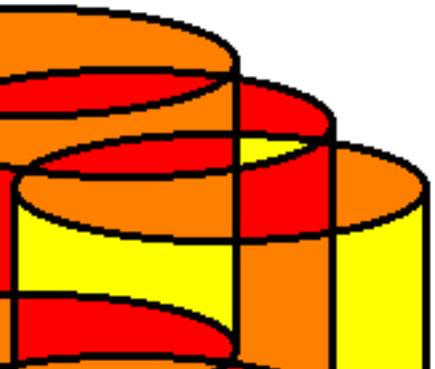
Pravila



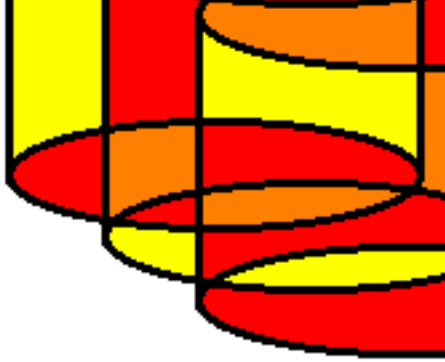
- Svi korisnici su osobe

```
?x:Osoba :- ?x:Korisnik.
```

Napomena: Pravilo dodajemo u datoteku mreza.flr i ponovno kompajliramo!

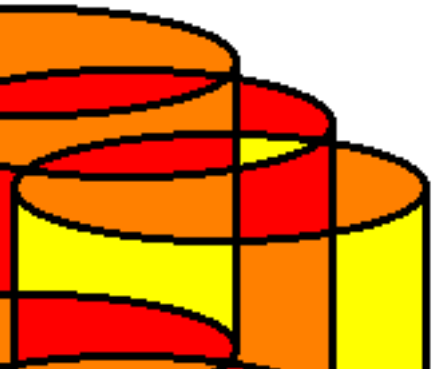


Jednostavni upiti



- Koje su sve osobe u modulu sm baze znanja?

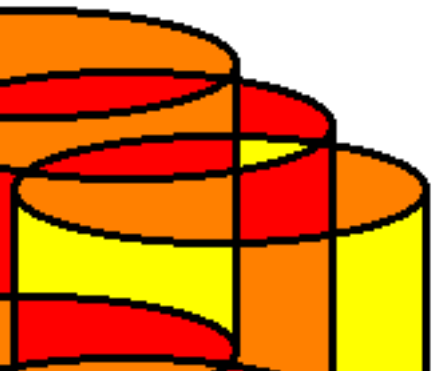
```
?x:Osoba@sm.
```



Pravila

- Koja pravila postoje u bazi znanja?

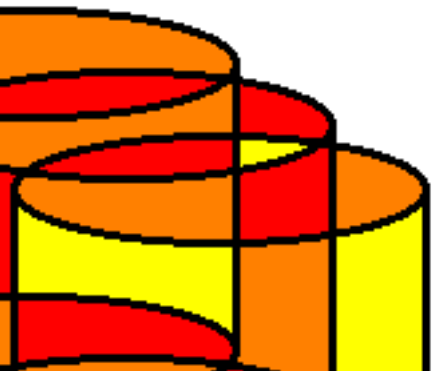
```
clause{ ?glava, ?tijelo }
```



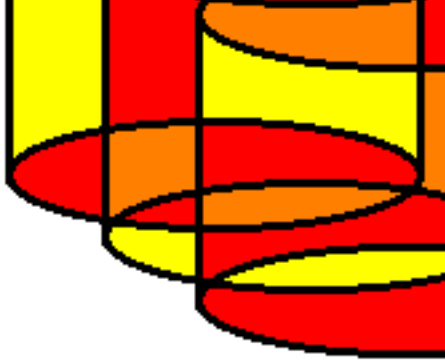
Pravila

- Osoba je punoljetna ako ima 18 ili više godina

```
?x:PunoljetnaOsoba :-
    ?x[ godina_rodjenja->?g ],
    ?dob \is 2019 - ?g,
    ?dob >= 18
```

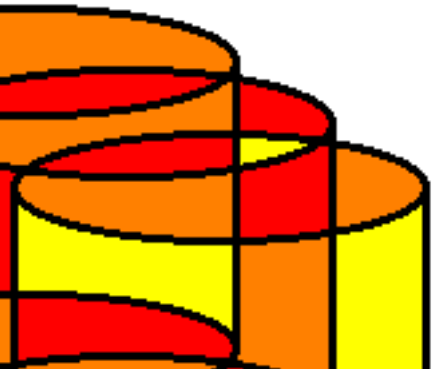


Primjer



- Imena i prezimena punoljetnih osoba:

```
(
 ?_:PunoljetnaOsoba[
    ime->?ime,
    prezime->?prezime ]
)@sm.
```



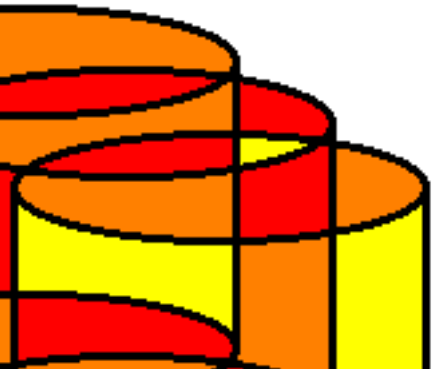
Primjer

- Imena i prezimena punoljetnih osoba koji su prijatelji s maloljetnim osobama:

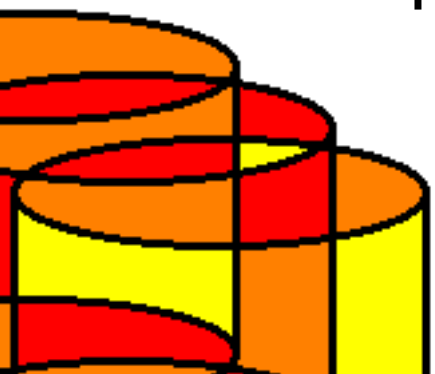
```
(
 ?_:PunoljetnaOsoba[
    ime->?ime,
    prezime->?prezime,
    prijatelj->?m ],
 \naf ?m:PunoljetnaOsoba
)@sm.
```

Pojašnjenje

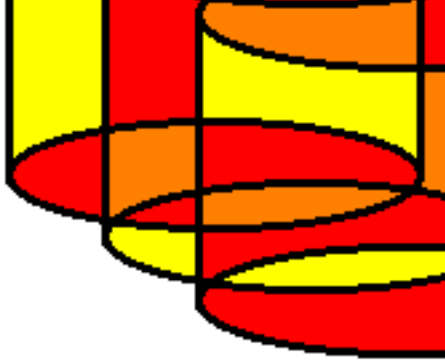
- Operator **naf** je negacija kao neuspjeh (engl. *negation as failure*)
- Izrazom smo rekli da **?m** nije instanca klase **PunoljetnaOsoba**



Ažuriranje OO baze znanja

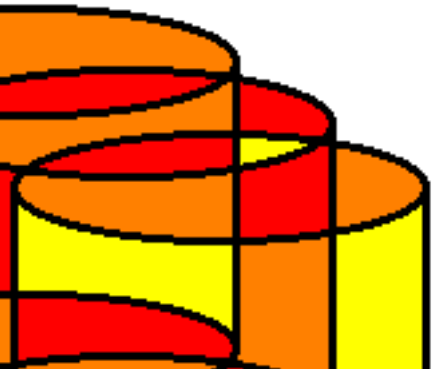


- Moguće je ažurirati
  - Rečenice logički (insert{...}, insertall{...}, delete{...}, deleteall{...}, erase{...}, eraseall{...})
  - Rečenice transakcijski (t_insert{...}, t_insertall{...}, t_delete{...}, t_deleteall{...}, t_erase{...}, t_eraseall{...})
  - Module (newmodule{...}, erasemodule{...})
  - Pravila (insertrule{...}, insertrule_a{...}, deleterule{...}, deleterule_a{...})



Upute

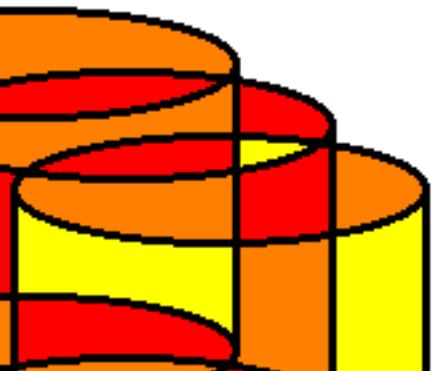
- Rezultate isprobavanja (kopiju konzole) pospremite u datoteku ime_prezime.txt



Primjer

- Na FLORA-2 konzoli isprobajmo sljedeće:

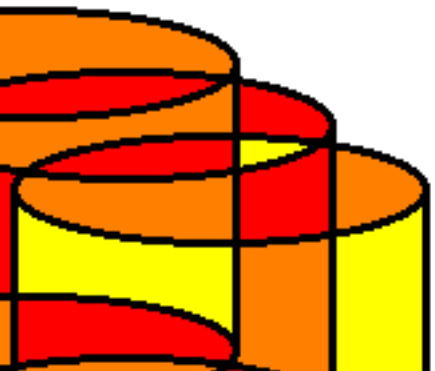
```
flora2 ?- newmodule{ dm }.
…
Yes
```



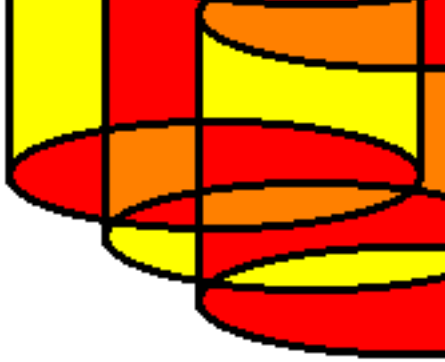
Primjer

- Na FLORA-2 konzoli isprobajmo sljedeće:

```
flora2 ?- newmodule{ dm }.
…
Yes
flora ?- insert{ ivek:osoba@dm }.
```

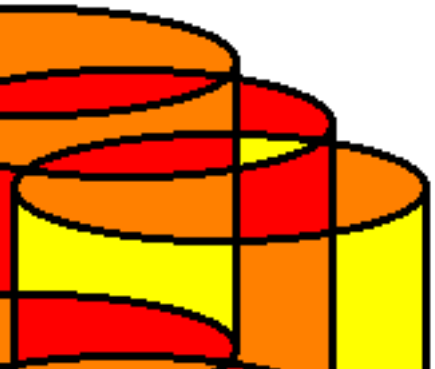


Primjer

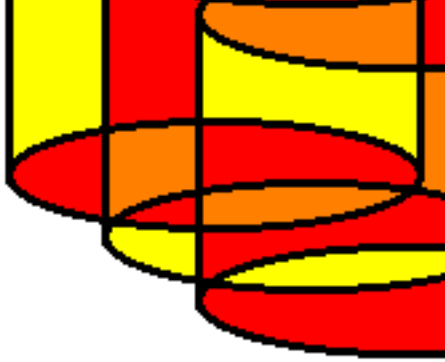


- Na FLORA-2 konzoli isprobajmo sljedeće:

```
flora2 ?- newmodule{ dm }.
…
Yes
flora ?- insert{ ivek:osoba@dm }.
flora ?- insert{ bara:osoba@dm }.
```

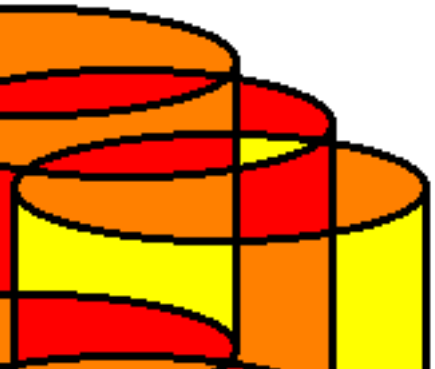


Primjer



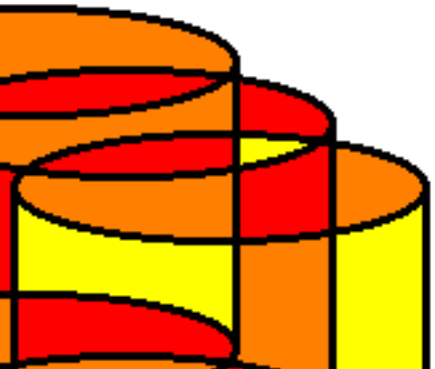
- Na FLORA-2 konzoli isprobajmo sljedeće:

```
flora2 ?- newmodule{ dm }.
…
Yes
flora ?- insert{ ivek:osoba@dm }.
flora ?- insert{ bara:osoba@dm }.
flora ?- ?o:osoba@?modul.
```



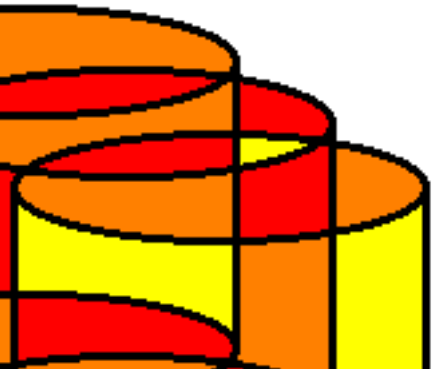
Primjer

```
insert{ ?x:covjek@dm | ?x:osoba@dm }.
```



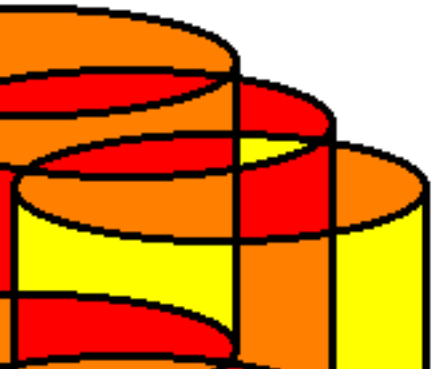
Primjer

```
insert{ ?x:covjek@dm | ?x:osoba@dm }.
?x:covjek@dm.
```



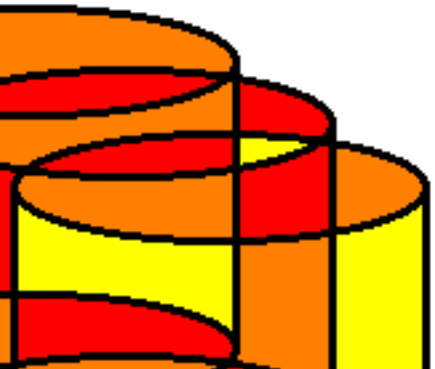
Primjer

```
insert{ ?x:covjek@dm | ?x:osoba@dm }.
?x:covjek@dm.
delete{ ?x:covjek@dm }.
```



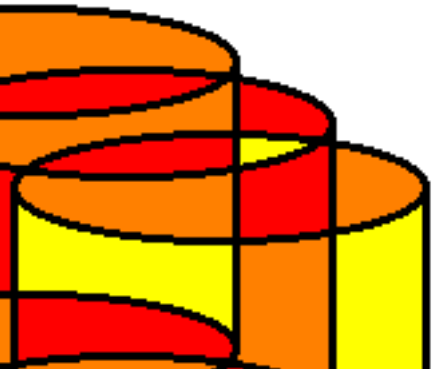
Primjer

```
insert{ ?x:covjek@dm | ?x:osoba@dm }.
?x:covjek@dm.
delete{ ?x:covjek@dm }.
?x:covjek@dm.
```



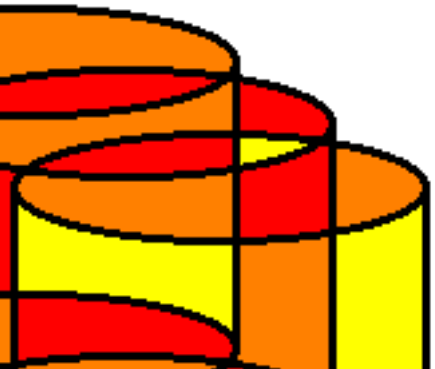
Primjer

```
insert{ ?x:covjek@dm | ?x:osoba@dm }.
?x:covjek@dm.
delete{ ?x:covjek@dm }.
?x:covjek@dm.
insertrule{ (?x:covjek :- ?x:osoba)@dm }.
```



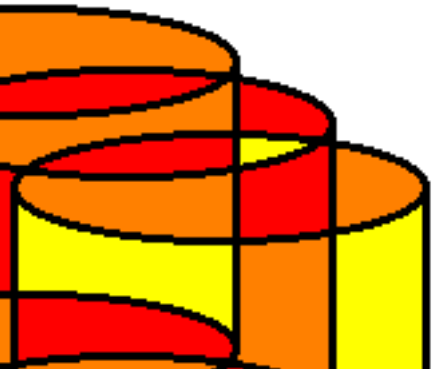
Primjer

```
insert{ ?x:covjek@dm | ?x:osoba@dm }.
?x:covjek@dm.
delete{ ?x:covjek@dm }.
?x:covjek@dm.
insertrule{ (?x:covjek :- ?x:osoba)@dm }.
?x:covjek@dm.
```



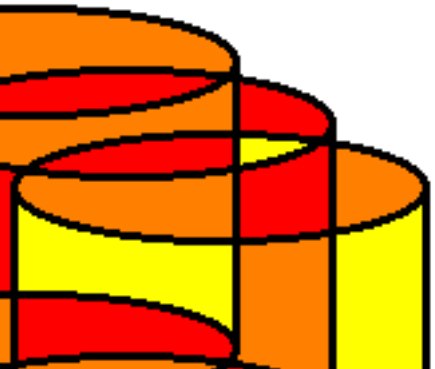
Primjer

```
insert{ ?x:covjek@dm | ?x:osoba@dm }.
?x:covjek@dm.
delete{ ?x:covjek@dm }.
?x:covjek@dm.
insertrule{ (?x:covjek :- ?x:osoba)@dm }.
?x:covjek@dm.
erasemodule{ dm }.
```

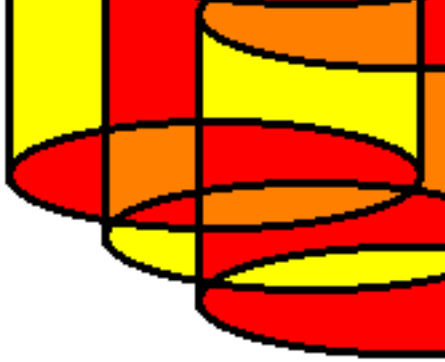


Primjer

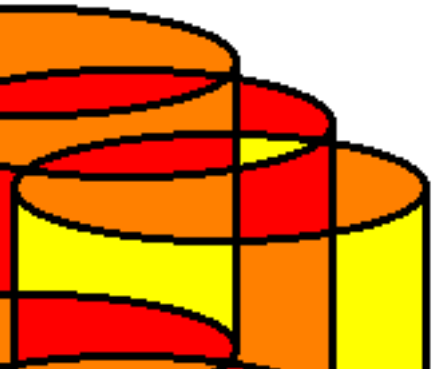
```
insert{ ?x:covjek@dm | ?x:osoba@dm }.
?x:covjek@dm.
delete{ ?x:covjek@dm }.
?x:covjek@dm.
insertrule{ (?x:covjek :- ?x:osoba)@dm }.
?x:covjek@dm.
erasemodule{ dm }.
?x:osoba@dm.
```



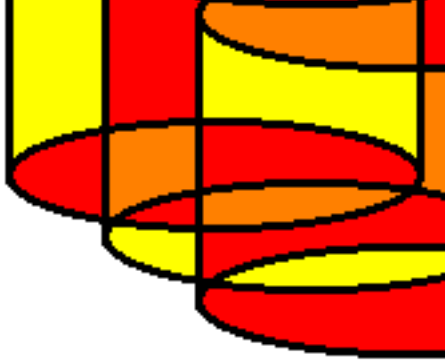
Perzistentni moduli



- Kako bi sačuvali podatke u Flora-2 koristi se integrirani modul **persistentmodules**
- Omogućava nam spajanje željenog modula na (relacijsku) bazu podataka u kojoj se pohranjuju podaci modula



Upute



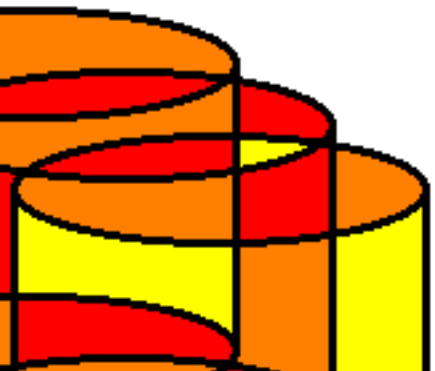
- Spojit ćemo FLORA-2 stroj na SQLite bazu podataka putem ODBC sučelja
- Sljedeće naredbe upisujemo na FLORA-2 konzoli
- Provjerite pristupne podatke da budu u skladu s podacima u .odbc.ini
- Pripazite u .odbc.ini je SQLite veza definirana prema temporary datoteci! Ako hoćete da rezultati ostanu i nakon ponovnog pokretanja računala izmjenite putanju u .odbc.ini do neke datoteke u Vašem home direktoriju!

Primjer

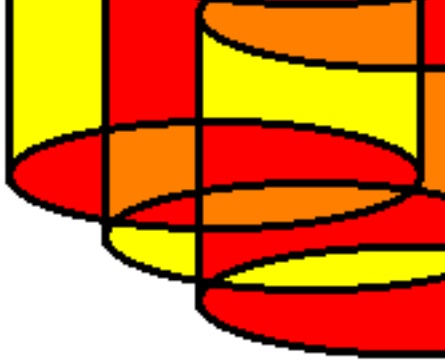
```
[persistentmodules>>pm].
newmodule{ mod1 }.
mod1[attach(sqllitedb,?_,vjezbe,vjezbe)]@pm.
insert{ a:c[ x->y ]@mod1 }.
\halt.
```

Ovom se naredbom konzola ugasila. Pokrećemo je ponovno primjerice kompajlom (CTRL+c CTRL+b) ili u meniju Flora-2 > Start Flora2 process.

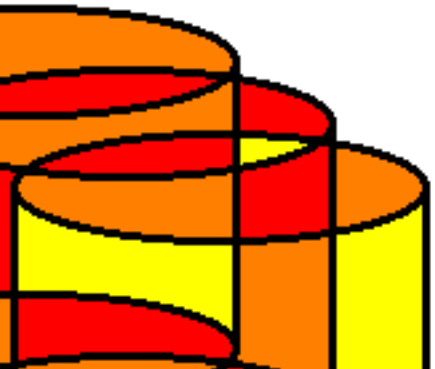
```
[persistentmodules>>pm].
newmodule{ mod1 }.
mod1[attach(sqllitedb,?_,vjezbe,vjezbe)]@pm.
?a[ ?x->?y ]@mod1.
```



Pojašnjenje



- Prvo smo učitali modul za perzistentnost
- Zatim smo kreirali novi modul (mod1)
- Tada smo modul povezali s bazom podataka u pozadini
- Onda smo ažurirali bazu znanja novom činjenicom
- I na kraju ugasili proces konzole.
- Kad smo ponovno pokrenuli konzolu i kreirali novi modul te ga spojili na istu bazu podataka, vidimo nakon upita da su podaci ostali pohranjeni.
- **NAPOMENA:** U takvom perzistentnom modulu pohranjuju se samo dinamički dodani podaci (pomoću insert, delete), podaci u datoteci se ne pohranjuju u bazu podataka!



Zadatak

- Kreirajte novu ODBC vezu (dodavanjem novog zapisa u .odbc.ini ili izmjenom postojećeg) za SQLite bazu podataka koja će se pohraniti u datoteci ime_prezime.db
- Implementirajte OODBP koja je u skladu s UML dijagramom klasa na sljedećem slajdu (potrebno je unesti barem 3 objekta iz svake klase, izraze hijerarhije klasa te pravilo vezano uz prihvat zadatka opisano u komentaru u dijagramu).
- Primjer korištenja pravila (pod pretpostavkom da je učitan modul igra):

```
flora2 ?- ?i[ prihvaca->?zad ]@igra.

?i = igrac1
?zad = zad1
```

- Kopiju naredbi za unos objekata i pravila pohranite u datoteku naredbe.txt

UML dijagram klasa