Teoria cu masini si motociclete

main.pro

```
implement main
    open core

    domains
      tipMasina = logan(); ford(); volkswagen().
        tipMotocicleta = a(); b(); c().
        infoMasina = infoMasina(tipMasina Tip, integer An, integer Pret ).
        infoMotocicleta = infoMotocicleta(tipMotocicleta Tip, integer An, integer
Pret).
       tipMijlocDeTransport = infoMasina(tipMasina Tip, integer An, integer Pret);
                    infoMotocicleta(tipMotocicleta Tip, integer An, integer Pret).

    class facts - fapteMasini
        are : (string Nume, tipMijlocDeTransport InfoMasina).

    clauses
        run():-
            console::init(),
            file::consult("..\\masini.txt", fapteMasini),
            fail.

      run():-
            stdIO::write(" ---------- Posesorii de Ford -------------- \n"),
            are(X, infoMasina(ford(),_,_)),
                stdIO::writef("% are Ford \n", X),
            fail.

        run():-
            stdIO::write(" ---------- Posesorii de motociclete ------------- \n"),
            are(X, infoMotocicleta(_,_,_)),
                stdIO::writef("% are motocicleta \n", X),
            fail.

       run():-
            stdIO::write(" ---------- Ce are Dorel -------------- \n"),
            are("Dorel", X),
                stdIO::writef("Dorel are autovehiculul % \n", X),
            fail.

       run():-
            stdIO::write(" ---------- Ce are Dorel -------------- \n"),
            are("Dorel", infoMasina(Tip,_,_)),
                stdIO::writef("Dorel are masina % \n", Tip),
            fail.
```

```
        run():-
             stdIO::write(" ---------- Autovehicule scumpe ------------- \n"),
              are(X, infoMasina(_,_,Pret)),
                 Pret > 2000,
                 stdIO::writef("% are masina cu pretul % \n", X, Pret),
                 fail.
         run() :-
              are(X, infoMotocicleta(_,_,Pret)),
                 Pret > 2000,
                 stdIO::writef("% are motocicleta cu pretul % \n", X, Pret),
             fail.

         run():-
             stdIO::write("\n---------------------------------\n").

end implement main

goal
    mainExe::run(main::run).
```

masini.txt

```
clauses
   are("Adrian",InfoMasina(logan(),2007,1600)).
   are("Cristian",InfoMasina(ford(),2000,1200)).
   are("Mihai",InfoMasina(ford(),2013,6300)).
   are("Dorel",InfoMasina(logan(),2011,1900)).
   are("Dorel",InfoMasina(volkswagen(),2014,8900)).
   are("Dorel",InfoMotocicleta(b(),2011,900)).
   are("Nelu",InfoMotocicleta(a(),2016,7400)).
   are("Sandu",InfoMotocicleta(b(),2010,850)).
```

Rulare

```
 ---------- Posesorii de Ford -------------
Adrian are Ford
Dorel are Ford
 ---------- Posesorii de motociclete -------------
Dorel are masina
Nelu are masina
Sandu are masina
 ---------- Ce are Dorel -------------
Dorel are autovehiculul infoMasina(logan,2011,1900)
Dorel are autovehiculul infoMasina(volswagen,2014,8900)
Dorel are autovehiculul infoMotocicleta(b,2011,900)
 ---------- Ce are Dorel -------------
Dorel are masina logan
Dorel are masina volswagen
 ---------- Autovehicule scumpe -------------
Mihai are masina cu pretul 6300
Dorel are masina cu pretul 8900
Nelu are motocicleta cu pretul 7400
```

Teoria cu masini utilizand tipuri compuse

main.pro

```
implement main
    open core

    domains
        tipMasina = logan(); ford(); volswagen().
        infoMasina = infoMasina(tipMasina Tip, integer An, integer Pret ).

    class facts - fapteMasini
        are : (string Persoana, infoMasina InfoMasina).

    clauses
        run():-
            console::init(),
            file::consult("..\\masini.txt", fapteMasini),
            fail.

 run():-
            stdIO::write(" ---------- Posesorii de logan ------------- \n"),
            are(X, infoMasina(logan(),_,_)),
                stdIO::writef("% are Logan \n", X),
            fail.

         run():-
            stdIO::write(" ---------- Posesorii de masini ------------- \n"),
            are(X, infoMasina(_,_,_)),
                stdIO::writef("% are masina \n", X),
            fail.

       run():-
            stdIO::write(" ---------- Ce are Dorel ------------- \n"),
            are("Dorel", X),
                stdIO::writef("Dorel are masina % \n", X),
            fail.

       run():-
            stdIO::write(" ---------- Ce are Dorel ------------- \n"),
            are("Dorel", infoMasina(Tip,_,_)),
                stdIO::writef("Dorel are masina % \n", Tip),
            fail.

       run():-
            stdIO::write(" ---------- Masini scumpe ------------- \n"),
            are(X, infoMasina(_,_,Pret)),
                Pret > 2000,
```

```
            stdIO::writef("% are masina cu pretul % \n", X, Pret),

            fail.

        run():-
            stdIO::write("\n--------------------------------\n").
end implement main

goal
    mainExe::run(main::run).
```

masini.txt

```
clauses
   are("Adrian",InfoMasina(logan(),2007,1600)).
   are("Cristian",InfoMasina(ford(),2000,1200)).
   are("Mihai",InfoMasina(ford(),2013,6300)).
   are("Dorel",InfoMasina(logan(),2011,1900)).
   are("Dorel",InfoMasina(volswagen(),2014,8900)).
```

Rulare

```
 ---------- Posesorii de logan --------------
Adrian are Logan
Dorel are Logan
 ---------- Posesorii de masini --------------
Adrian are masina
Cristian are masina
Mihai are masina
Dorel are masina
Dorel are masina
 ---------- Ce are Dorel --------------
Dorel are masina infoMasina(logan,2011,1900)
Dorel are masina infoMasina(volswagen,2014,8900)
 ---------- Ce are Dorel --------------
Dorel are masina logan
Dorel are masina volswagen
 ---------- Masini scumpe --------------
Mihai are masina cu pretul 6300
Dorel are masina cu pretul 8900
```

2.3 Teoria cu masini si motociclete

main.pro

```
implement main
    open core

    domains
      tipMasina = logan(); ford(); volswagen().
        tipMotocicleta = a(); b(); c().
        infoMasina = infoMasina(tipMasina Tip, integer An, integer Pret ).
        infoMotocicleta = infoMotocicleta(tipMotocicleta Tip, integer An, integer
Pret).
        tipMijlocDeTransport =  infoMasina(tipMasina Tip, integer An, integer Pret
```

```
 ) ;
                                                infoMotocicleta(tipMotocicleta Tip, int
eger An, integer Pret).
    class facts - fapteMasini
        are : (string Persoana, tipMijlocDeTransport InfoMasina).

    clauses
        run():-
            console::init(),
            file::consult("..\\masini.txt", fapteMasini),
            fail.

      run():-
            stdIO::write(" ---------- Posesorii de Ford -------------- \n"),
            are(X, infoMasina(logan(),_,_)),
                stdIO::writef("% are Ford \n", X),
            fail.

        run():-
            stdIO::write(" ---------- Posesorii de motociclete --------------
 \n"),
            are(X, infoMotocicleta(_,_,_)),
                stdIO::writef("% are motocicleta \n", X),
            fail.

       run():-
            stdIO::write(" ---------- Ce are Dorel -------------- \n"),
            are("Dorel", X),
                stdIO::writef("Dorel are autovehiculul % \n", X),
            fail.

       run():-
            stdIO::write(" ---------- Ce are Dorel -------------- \n"),
            are("Dorel", infoMasina(Tip,_,_)),
                stdIO::writef("Dorel are masina % \n", Tip),
            fail.

       run():-
            stdIO::write(" ---------- Autovehicule scumpe -------------- \n"),
             are(X, infoMasina(_,_,Pret)),
                Pret > 2000,
                stdIO::writef("% are masina cu pretul % \n", X, Pret),
                fail.
        run() :-
             are(X, infoMotocicleta(_,_,Pret)),
                Pret > 2000,
                stdIO::writef("% are motocicleta cu pretul % \n", X, Pret),
            fail.
```

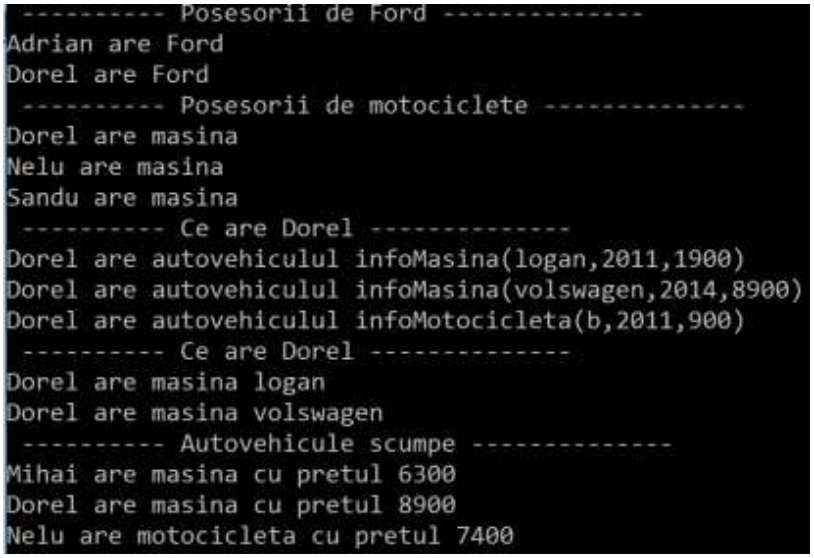
```
        run():-
            stdIO::write("\n---------------------------------\n").
end implement main

goal
    mainExe::run(main::run).
```

masini.txt

```
clauses
   are("Adrian",InfoMasina(logan(),2007,1600)).
   are("Cristian",InfoMasina(ford(),2000,1200)).
   are("Mihai",InfoMasina(ford(),2013,6300)).
   are("Dorel",InfoMasina(logan(),2011,1900)).
   are("Dorel",InfoMasina(volswagen(),2014,8900)).
   are("Dorel",InfoMotocicleta(b(),2011,900)).
   are("Nelu",InfoMotocicleta(a(),2016,7400)).
   are("Sandu",InfoMotocicleta(b(),2010,850)).
```

Rulare



Tema:

Temele de laborator **sunt obligatorii și vor forma nota de la laborator**.Din notele proiectelor voi forma o medie, medie care va reprezenta 80% din nota de laborator.

NU UITAȚI SĂ ARHIVAȚI PROIECTELE după ce, înainte le-ati pus pe toate în același folder ce poartă numele dvs, ARHIVA CONȚINE TOTUL NU DOAR MAIN.PRO ȘI TXT-UL ! (sau mai rău, să copiați totul în word, sau să îmi trimiteți alte formate...)
ARHIVA POARTĂ NUMELE DVS!
Tema trebuie trimisă în timp util!

Proiectele care nu respectă aceste cerințe **nu vor fi notate**!

Cerinta pentru a doua nota de laborator este urmatoarea:

Creati un program in Visual Prolog 7,5 care sa aiba mai multe persoane cu mai multe bunuri posedate de fiecare.
Faceti diverse interogari care se preteaza pe proiectul fiecaruia.
Cu cat aveti mai multe clause si mai multe interogari aveti nota mai mare.
Să se introducă mimim 5 persoane și 5 obiecte fiecare, și să se facă minim 5 interogări pentru nota 5 (cinci) și respectiv 10 persoane și 15 obiecte fiecare, și să se facă minim 25 interogări pentru nota 10(zece).

Va urez succes!