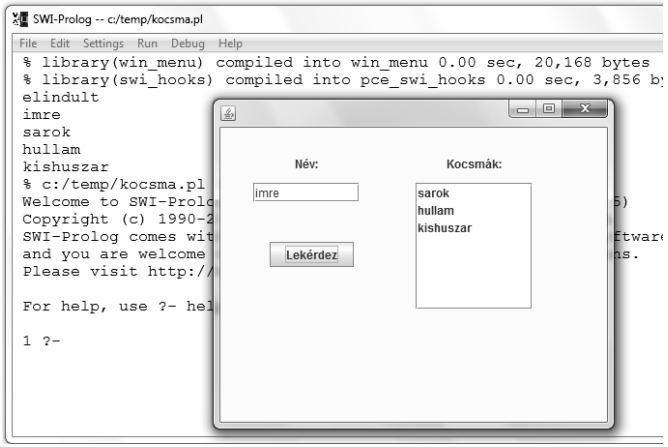
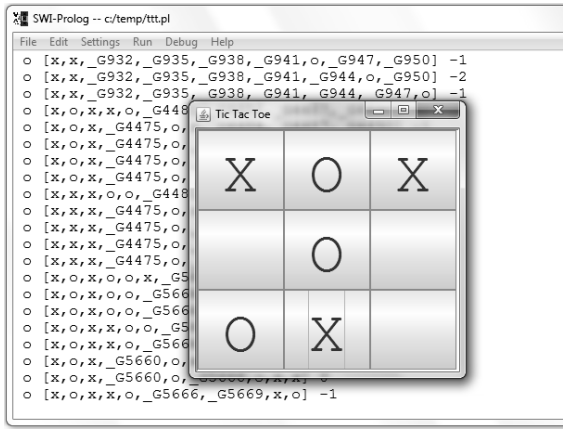
# *Logikai programozás*

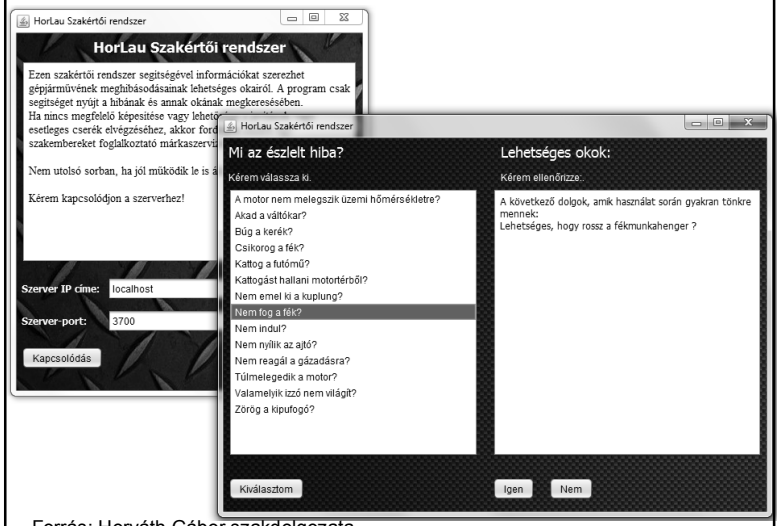
10.

## KIINDULÓ PÉLDÁK



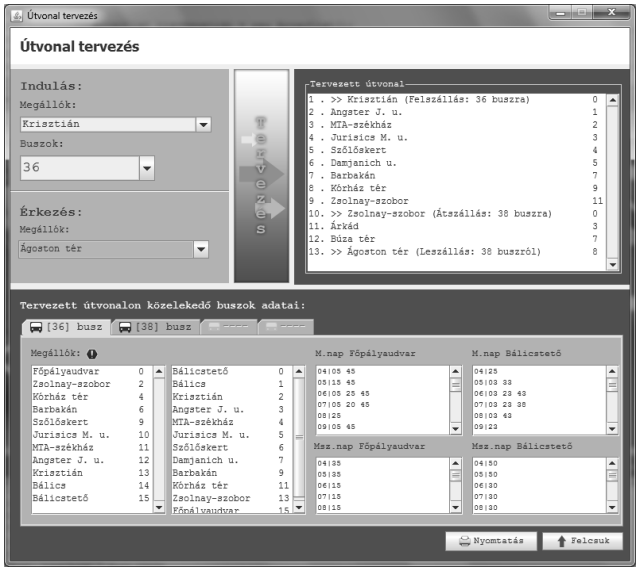



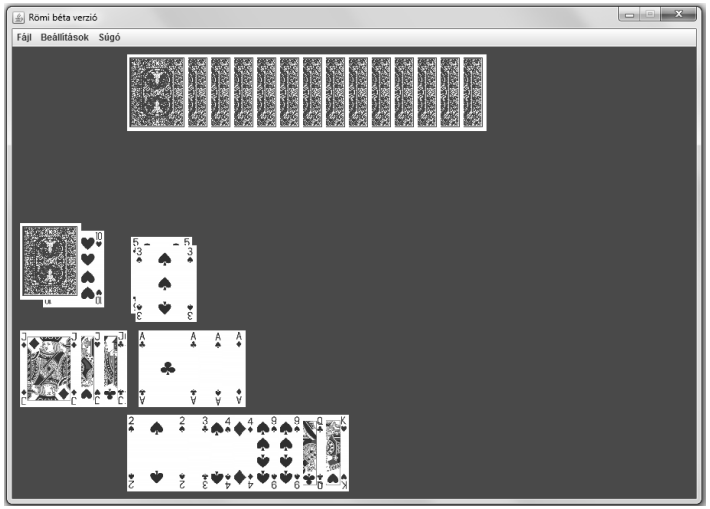

http://www.csupomona.edu/~jrfisher/www/prolog_tutorial/8_4.html



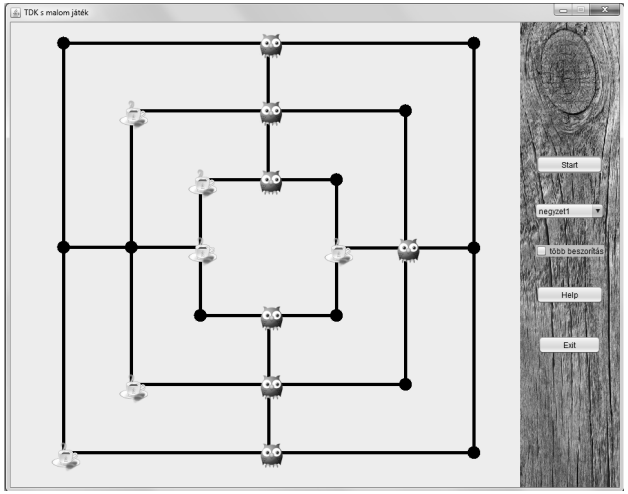

Forrás: Horváth Gábor szakdolgozata


Forrás: Schunk Máté szakdolgozata




Forrás: Balázs Lajos szakdolgozata




Forrás: Fenyvesi Tamás szakdolgozata

## JAVA – PROLOG KAPCSOLAT

1. Kliens-szerver kapcsolat
2. JPL
3. egyéb

## JAVA – PROLOG KAPCSOLAT

Szerver:

Ugyanolyan Java szerver, mint amilyet Java-ban tanultunk, egyetlen dologra kell odafigyelni:

Ha Prolog kliensnek küldünk üzenetet, akkor az üzenet végére pontot kell írnunk.

## JAVA – PROLOG KAPCSOLAT

Ehhez a szerverhez bármilyen kliens kapcsolódhat, Java is és Prolog is.

Mi kell ahhoz, hogy létrejöjjön a kapcsolat?

Meg kell adni a szerver IP címét és a megfelelő portot, és definiálni kell a szükséges socketet.

Meg kell adni a szükséges input/output csatornákat.

```
connect(Port) :-
        tcp_socket(Socket),
        tcp_connect(Socket,localhost:Port),
                % most a localhost-ra kapcsolódik
        tcp_open_socket(Socket,INs,OUTs),
                % megadjuk az Input/Output csatornákat
        assert(connectedReadStream(INs)),
        assert(connectedWriteStream(OUTs)),
                % tároljuk őket egy adatbázisban

        write(OUTs,prolog),
                % azonositja magat a szervernek küldött szóval
         nl(OUTs),
                % a sorvégjelet is kiküldi
        flush_output(OUTs),
                % fizikailag is kiküldi
        sleep(1).
                % kis várakoztatás

:- connect(3700).
```

## JAVA – PROLOG KAPCSOLAT

Ezek után oda kell figyelni a megoldandó protokollokra.

Beolvasás:

```
connectedReadStream(IStream),
        % kiveszi az adatbázisból az input csatornát
read(IStream,In),
        % beolvassa az adatot
```

## JAVA – PROLOG KAPCSOLAT

Ezek után oda kell figyelni a megoldandó protokollokra.

Kiíratás:

```
connectedWriteStream(OStream),
        % kiveszi az adatbázisból az output csatornát
write(OStream,Out),
nl(OStream),
flush_output(OStream),
        % kiírja rá az adatot (Out) és kiküldi fizikailag is
```

## JAVA – PROLOG KAPCSOLAT

Példa:

Java szerver, Java ill. Prolog kliens.

A szerver beolvassa a klienstől kapott számot, és visszaadja annak négyzetét.

A Java kliens egyetlen beolvasott szám négyzetét kéri le, a Prolog kliens pedig elküldi a szervernek az első tíz számot, lekéri a négyzetüket és kiíratja.

```
start(Szam) :- Szam>10,writeln('vége').

start(Out) :- connectedWriteStream(OStream),
                % kiveszi az adatbázisból az output csatornát
        write(OStream,Out), nl(OStream), flush_output(OStream),
                % kiírja rá az adatot (Out) és kiküldi fizikailag is

        connectedReadStream(IStream), read(IStream,In),
        % kiveszi az adatbázisból az input csatornát és beolvassa az
        % adatot
        writef('%w négyzete: %w \n',[Out,In]),
                % kiírja a képernyőre

        UjOut is Out+1, sleep(1),
                % várakozik, majd a következő számmal újra meghívja
                % az eljárást
        start(UjOut).

:- start(1).
        % ez is automatikusan indul
```

Szerver:

```
BufferedReader in …
PrintWriter out …

ID=in.readLine();
        // beolvassa a kliens azonosítóját

keres=in.readLine();
valasz= Integer.parseInt(keres);
valasz= valasz*valasz;

if(ID.equals("prolog")){
        out.println(String.valueOf(valasz)+'.');
        out.flush();
}else if(ID.equals("java")){
        out.println(String.valueOf(valasz));
        out.flush();
}
```

## JAVA – PROLOG KAPCSOLAT

Másik példa:

Már megint kocsma.

Adjunk meg Java-ban egy nevet, és kérjük le a Prolog-tól azt, hogy az illető milyen kocsmákba jár.

## JAVA – PROLOG KAPCSOLAT

Szerver: két klienst enged, egy Java és egy Prolog alkalmazást:

```
private ServerSocket ss;
private Socket[] s = new Socket[2];

while (db < 2) {
    s[db] = ss.accept();
    System.out.println("Létrejött a " + (db + 1) + ". kapcsolat");
    db++;
}
kiszolgal();
```

```
in1 = new BufferedReader(…(s[0].getInputStream()));
out1 = new PrintWriter(s[0].getOutputStream());
in2 = new BufferedReader(…(s[1].getInputStream()));
out2 = new PrintWriter(s[1].getOutputStream());

ID1 = in1.readLine();
        // azonosítja a klienst
ID2 = in2.readLine();

        if (ID1.equals("java")) {
            inp = in2;
            outp = out2;
            inj = in1;
            outj = out1;
        } else {
            inp = in1;
            outp = out1;
            inj = in2;
            outj = out2;
        }
```

```
start :- writeln('elindult'),
        connectedReadStream(IStream),
        read(IStream,Nev),
        writeln(Nev),
        forall(jar(Nev,Kocsma), ki(Kocsma)),
        connectedWriteStream(OStream),
        write(OStream,vege),
        nl(OStream),
        flush_output(OStream).

ki(Kocsma) :-
        connectedWriteStream(OStream),
        write(OStream,Kocsma),
        nl(OStream),
        flush_output(OStream).

:- start.
```

## JAVA – PROLOG KAPCSOLAT

A szerver-kliens kapcsolat során el kell indítani a

- szervert
- Java klienst
- Prolog klienst

Nehézkes, elavult.

## JAVA – PROLOG KAPCSOLAT (2)

Korszerűbb megoldás: JPL

Kétirányú Java-Prolog kapcsolat. Eszköze:

swipl\lib\jpl.jar

Használatához be kell állítani a path-ban:
C:\Program Files\swipl\bin;
C:\Program Files\swipl\lib\jpl.jar;

C:\Program Files\Java\jdk1.7.0_51\bin;
(vagy újabb – de ez úgyis be van ☺)

+ NetBeans Libraries\Add Jar (hozzáadni a jpl.jar-t)
vagy Maven beállítás

## JAVA – PROLOG KAPCSOLAT (2)

Maven beállítás:

Maven projekt létrehozása, a pom.xml-be beleírni ezt:

```
<dependencies>
    <dependency>
        <groupId>jpl</groupId>
        <artifactId>jpl</artifactId>
        <version>3.1.4-alpha</version>
    </dependency>
</dependencies>
```

## JAVA – PROLOG KAPCSOLAT (2)

Ez a kapcsolat csak a 6-os SWI verzióig működik.

A 7… verziókban (2015 nyarától) már a JPL7 szerepel, ahhoz nem jó ez a függőség.
(Újabbat nem találtam ☹)

Használatához mintapéldák pl. itt:

https://github.com/SWI-Prolog/packages-jpl/tree/8fc80a5d946ece7c16efb7d895420b7604485bd8

https://github.com/SWI-Prolog/packages-jpl/tree/master/examples/java

doksi: http://jpl7.org/

Megjegyzés: vagy mindkettő 32 bites vagy mindkettő 64 bites legyen.

## JAVA – PROLOG KAPCSOLAT (2)

JPL példa:

```
import org.jpl7.Query;
import org.jpl7.Term;

Query query = new Query("consult('./prolog/kocsma.pl')");
System.out.println("A kapcsolódás " +
        (query.hasSolution() ? "sikerült" : "nem sikerült"));
```

prolog mappa: src mappa mellett

## JAVA – PROLOG KAPCSOLAT (2)

JPL példa:

```
query = new Query("jar(Ki,Hova)");
System.out.println("A kocsmába járók:");
Map<String, Term> megoldas;
while (query.hasMoreSolutions()) {
    megoldas = query.nextSolution();
    System.out.println("Ki = " + megoldas.get("Ki")
            + ", Hova = " + megoldas.get("Hova"));
}
```

## JAVA – PROLOG KAPCSOLAT (2)

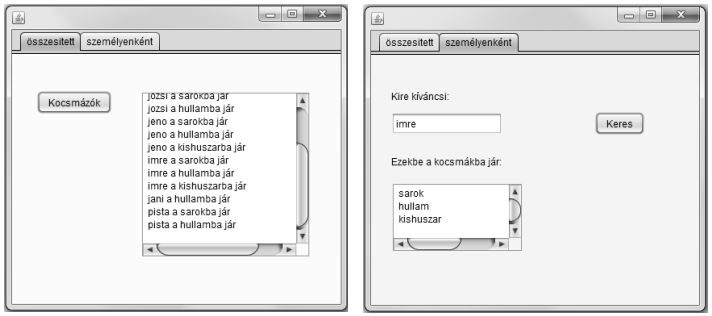
JPL példa:

```
// Ugyanez másképp:
String cel = "jar(Ki,Hova)";
Map<String, Term>[] megoldasok = Query.allSolutions(cel);
System.out.println("\nA kocsmába járók (2):");
for (Map<String, Term> map : megoldasok) {
    System.out.println("Ki = " + map.get("Ki")
            + ", Hova = " + map.get("Hova"));
}
```

# JAVA – PROLOG KAPCSOLAT (2)

```
Scanner scanner = new Scanner(System.in);
System.out.print("\nKérem adja meg, kire kíváncsi! ");
try {
    String nev = scanner.nextLine();
    String kerdes = "jar('" + nev + "',Hova).";
    Query q = new Query(kerdes);
    if (!q.hasSolution()) {
        System.out.println("Nincs ilyen ember");
    } else {
        System.out.println(nev + " kedvenc kocsmái:");
        while (q.hasMoreSolutions()) {
            System.out.println(q.nextSolution().get("Hova"));
        }
    }
} catch (Exception e) {
    e.printStackTrace();
}
```

# JAVA – PROLOG KAPCSOLAT (2)




# JAVA – PROLOG KAPCSOLAT (2)

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

    String cel = "jar(Ki,Hova)";
    Map<String, Term>[] megoldasok = Query.allSolutions(cel);

    for (Map<String, Term> megoldas : megoldasok) {
        txtEredmenyek.append( megoldas.get("Ki") +
                              " a " + megoldas.get("Hova")+ "ba jár\n");
    }
}
```

# JAVA – PROLOG KAPCSOLAT (2)

```
private void btnKeresActionPerformed(java.awt.event.ActionEvent evt) {
    txtKocsmak.setText("");
    String nev = txtKi.getText();
    String kerdes = "jar('" + nev + "',Hova).";
    Query q = new Query(kerdes);
    if (!q.hasSolution()) {
        JOptionPane.showMessageDialog(this, "Nincs ilyen ember");
    } else {
        while (q.hasMoreSolutions()) {
            txtKocsmak.append(q.nextSolution().get("Hova") + "\n");
        }
    }
}
```

## JAVA – PROLOG KAPCSOLAT (2)

Mivel bármelyik tabulált felülettel kezdhetjük, ezért a konzultációt célszerű a frame betöltésekor hívni.

```
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new FoFrame().setVisible(true);
        konzultacio();
    }

    private void konzultacio() {
        Query query = new Query("consult('./prolog/kocsma.pl')");
        if (!query.hasSolution()) {
            JOptionPane.showMessageDialog(null, "Nem sikerült a konzultáci
        }
    }
});
```

## JAVA – PROLOG KAPCSOLAT

A kiadott példa +

http://jpl7.org/

https://github.com/SWI-Prolog/packages-jpl/tree/8fc80a5d946ece7c16efb7d895420b7604485bd8

https://github.com/SWI-Prolog/packages-jpl/tree/master/examples/java

http://www.swi-prolog.org/pldoc/man?section=jpl

www.google.com

Egyéb Java-Prolog kapcsolódási lehetőségek:

http://kaminari.istc.kobe-u.ac.jp/PrologCafe/

http://platform.netbeans.org/tutorials/60/nbm-prolog.html

Prolog + web (tutorial):

http://www.pathwayslms.com/swipltuts/html/

## EGY KIS KITEKINTÉS

Néhány alkalmazási terület:

- Szakértői rendszerek
- Nyelvészeti alkalmazások
- Ontológiák, szemantikus web alapja

Ontológia (nagyon pongyolán): fogalmak, viszonyok leírására szolgáló rendszer (a programspecifikáció általánosítása).

Szemantikus web (nagyon pongyolán): tartalomra alapozott internetes keresési technológiák.

## EGY KIS KITEKINTÉS

http://jena.apache.org/

http://home.mit.bme.hu/~fandrew/diplomaterv_hu.html

http://www.cs.man.ac.uk/~horrocks/FaCT/

http://www.tankonyvtar.hu/hu/tartalom/tamop425/0005_34_internetes_keresorendszerek_scorm_10/1032_a_szemantikus_web.html

http://www.tankonyvtar.hu/hu/tartalom/tamop425/0005_34_internetes_keresorendszerek_scorm_10/1034_ontolgia.html

stb.