SYSTEM REKRUTACJI OPARTY O ŚRODOWISKO PROLOG

Wykonał Paweł Pauszek

Ideą programu było stworzenie systemu służącego do rekrutacji wojskowej. Program zadaje użytkownikowi pytania a ten na nie odpowiada. Można powiedzieć, że program pełnił także rolę chatbota. Wszystko opiera się o środowisko języka Prolog, czyli języka logicznego, w którym nie opisuje się co ma program robić (tak jak ma to miejsce w klasycznych językach programowania), tylko co chce się osiągnąć. Logika programu jest wyrażona w kategoriach relacji, reprezentowanych jako fakty i reguły a obliczenia są inicjowane przez wykonanie zapytań dotyczących tych relacji.

Cały program działa na zasadzie systemu ekspertowego, czyli takiego, który wykorzystuje opisaną bazę wiedzy (wcześniej zdefiniowane fakty oraz reguły) do podejmowania decyzji przez człowieka eksperta, w celu rozwiązania problemów. Zakłada on wspomaganie rozwiązywania problemów na zasadach zbliżonych do działania ludzkiego mózgu.

Jako interfejs graficzny w Prologu można wykorzystać XPCE lub moduł JPL, korzystający z elementów Javy, w tym Swinga. Wykorzystałem drugą opcję, czyli moduł JPL. Jest on mocno przestarzały co czyni jego wizualność na dzisiejsze standardy po prostu brzydkim.

Do działania programu niezbędne jest posiadanie SWI-Prolog oraz Javy JRE, które można pobrać za darmo.

Poniżej zamieszczę i opiszę działanie kodu programu:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS P:\Firefox\ChatBot-Prolog-main> swipl -s .\Military-Recruitment-Chatbot.pl
% Extended DLL search path with
%   'd:/Java/jre1.8.0_331/bin/server'
%   'd:/Java/jre1.8.0_331/bin'
Warning: p:/firefox/chatbot-prolog-main/military-recruitment-chatbot.pl:107:
Warning:    Singleton variables: [JOP]
Warning: p:/firefox/chatbot-prolog-main/military-recruitment-chatbot.pl:128:
Warning:    Singleton variables: [Kandydat,JOP]
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.2)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

1 ?- startuj.
```

W celu załadowania programu posłużyłem się poleceniem swipl -s nazwa_pliku.pl . Program startuje dopiero po wprowadzeniu w terminalu polecenia startuj.

Military-Recruitment-Chatbot.pl

```
:- use_module(library(jpl)).
startuj :-
        powitanie.

    cechy(Kandydat, narkotyki) :- zapytaj(Kandydat, "Czy brales kiedykolwiek narkotyki?").
    cechy(Kandydat, uzalezniony) :- zapytaj(Kandydat, "Czy byles lub nadal jestes od czegos uzalezniony?").
    cechy(Kandydat, kategorie) :- zapytaj(Kandydat, "Czy masz kategorie wojskowa A?").
    cechy(Kandydat, wiek) :- zapytaj(Kandydat, "Czy masz wiecej niz 18 lat?").
    cechy(Kandydat, testp) :- zapytaj(Kandydat, "Czy wyrazasz zgode na przeprowadzenie testu psychologicznego?").
    cechy(Kandydat, testf) :- zapytaj(Kandydat, "Czy zgadzasz sie na przeprowadzenie testu sprawnosciowego?").
    cechy(Kandydat, patriota) :- zapytaj(Kandydat, "Czy jestes patriota?").
    cechy(Kandydat, zainteresowanie) :- zapytaj(Kandydat, "Czy interesujesz sie szeroko pojeta militaryzacja?").
    cechy(Kandydat, powinnosc) :- zapytaj(Kandydat, "Czy czujesz wewnetrzny obowiazek obrony swojego panstwa?").
    cechy(Kandydat, smierc) :- zapytaj(Kandydat, "Czy jestes gotowy zginac za swoj kraj?").
    cechy(Kandydat, psychika) :- zapytaj(Kandydat, "Czy kiedykolwiek leczyles sie psychiatrycznie?").
    cechy(Kandydat, psychika2) :- zapytaj(Kandydat, "Czy masz lub miales stwierdzona chorobe psychiczna?").
    cechy(Kandydat, jedzenie) :- zapytaj(Kandydat, "Czy masz dobra tolerancje gastronomiczna?").
    cechy(Kandydat, odpornosc) :- zapytaj(Kandydat, "Czy posiadasz dobra odpornosc (chorowanie maksymalnie 2 razy do roku)?").

```

Najpierw deklaruje chęć używania modułu z biblioteki JPL, która odpowiada w programie za graficzny interfejs.

Po wpisaniu polecenia startuj uruchamiania się „interfejs” powitanie, mający na celu powitanie użytkownika oraz pobranie od niego imienia.

Powyżej są także zadeklarowane reguły dla Kandydata, który posiada określone cechy. Dla każdej z cech, zostaje zadane indywidualne pytanie. Jeżeli użytkownik odpowie twierdząco to zostają do Kandydata przypisane odpowiednie cechy.

```
    wynik(Kandydat,idealny_zolnierz) :-
        cechy(Kandydat, kategorie),
        cechy(Kandydat, wiek),
        cechy(Kandydat, testp),
        cechy(Kandydat, testf),
        cechy(Kandydat, patriota),
        cechy(Kandydat, zainteresowanie),
        cechy(Kandydat, powinnosc),
        cechy(Kandydat, smierc).

    wynik(Kandydat, dobry_zolnierz) :-
        cechy(Kandydat, kategorie),
        cechy(Kandydat, wiek),
        cechy(Kandydat, testp),
        cechy(Kandydat, testf),
        cechy(Kandydat, patriota).

```

```
    wynik(Kandydat, moze_zolnierz) :-
        cechy(Kandydat, narkotyki),
        cechy(Kandydat, kategorie),
        cechy(Kandydat, wiek),
        cechy(Kandydat, testp),
        cechy(Kandydat, testf).

    wynik(Kandydat, nie_zolnierz) :-
        cechy(Kandydat, narkotyki);
        cechy(Kandydat, uzalezniony);
        cechy(Kandydat, psychika);
        cechy(Kandydat, psychika2).

    wynik(Kandydat, ewentualnie) :-
        cechy(Kandydat, jedzenie),
        cechy(Kandydat, odpornosc).

    wynik(_, inne).
```

Do podjęcia decyzji stworzyłem konkretne schematy (wyniki), które zawierają określone cechy Kandydata. Tak jak opisałem wcześniej, Kandydat określa swoje cechy poprzez interfejs graficzny Tak/Nie, a program na podstawie cech przypisuje mu konkretny schemat. W przypadku kiedy Kandydat nie pasuje pod żaden schemat przypisuje mu się schemat inne.

```
kd(Kandydat):-
    wynik(Kandydat, Decyzja),
    koniec(Kandydat, Decyzja),
    zmien.

pytaj(Kandydat,Pytanie):-
    wyswietl(Kandydat,Pytanie).

:- dynamic tak/1, nie/1.

zapytaj(K, P) :-
    (tak(P) ->
        true
    ;
    (nie(P) ->
        fail
    ;
        pytaj(K, P))
    ).

zmien :- retract(tak(_)),fail.
zmien :- retract(nie(_)),fail.
zmien.
```

„kd” uruchamia się po wpisaniu imienia przez użytkownika. Następuje tutaj dopasowanie cech pod dany schemat i wyświetlenie ostatecznej decyzji.

„pytaj” służy za wyświetlenie pytań z interfejsem graficznym.

Dynamic, assert oraz retract służą do deklaracji, że zmienne tak/nie będą się zmieniać/będą dynamiczne.

Zapytaj służy do wywoływania „pytaj” oraz pomaga przy określaniu cech (przy odpowiedzi ustala czy cecha jest true czy false).

```
powitanie :-
    jpl_new('javax.swing.JFrame', ['Chatbot'], Frame),
    jpl_new('javax.swing.JLabel',['Rozmowa na stanowisko wojskowe'],Label),
    jpl_new('javax.swing.JPanel',[],Pan),
    jpl_call(Pan,add,[Label],_),
    jpl_call(Frame,add,[Pan],_),
    jpl_call(Frame, setLocation, [200,50], _),
    jpl_call(Frame, setSize, [800,700], _),
    jpl_call(Frame, setVisible, [@(true)], _),
    jpl_call(Frame, toFront, [], _),
    jpl_new('javax.swing.JOptionPane', [], JOP),
    jpl_call('javax.swing.JOptionPane', showInputDialog, [Frame,'Prosze podaj swoje imie'], Anserw),
    atomic_list_concat(['Witaj ', Anserw ,' w elektronicznym systemie rekrutacji wojskowej. Zadamy Ci pare pytan, prosze odpowiadaj szczerze. '], Witaj),
    jpl_call(JOP, showMessageDialog, [Frame, Witaj], _),
    jpl_call(Frame, dispose, [], _),

    (Anserw == @(null) ->
        write('Przerwano'),
        fail
    ;
        kd(Anserw)
    ).
```

Powitanie jest „interfejsem” powitalnym. Odpowiada on za interfejs graficzny dzięki korzystaniu z komponentów Java Swing. Odwoływanie do Swinga tworzę ca pomocą jpl_call lub jpl_new. Powitanie pyta także użytkownika o imię oraz wywołuje kd. W celu połączenia stringów stosuje atomic_list_concat.

```
104
105 wyswietl(Kandydat, Pytanie) :-
106     atomic_list_concat([Kandydat,' ',Pytanie], Atom),
107     jpl_new('javax.swing.JFrame', ['Chatbot'], Frame),
108     jpl_new('javax.swing.JLabel',['Rozmowa na stanowisko wojskowe'],Label),
109     jpl_new('javax.swing.JPanel',[],Pan),
110     jpl_call(Pan,add,[Label],_),
111     jpl_call(Frame,add,[Pan],_),
112     jpl_call(Frame, setLocation, [200,50], _),
113     jpl_call(Frame, setSize, [800,700], _),
114     jpl_call(Frame, setVisible, [@(true)], _),
115     jpl_call(Frame, toFront, [], _),
116     jpl_new('javax.swing.JOptionPane', [], JOP),
117     jpl_call('javax.swing.JOptionPane', showConfirmDialog, [Frame,Atom], Anserw),
118     jpl_call(Frame, dispose, [], _),
119
120     ((Anserw == 0) ->
121         assert(tak(Pytanie))
122     ;
123         assert(nie(Pytanie)), fail).
124
```

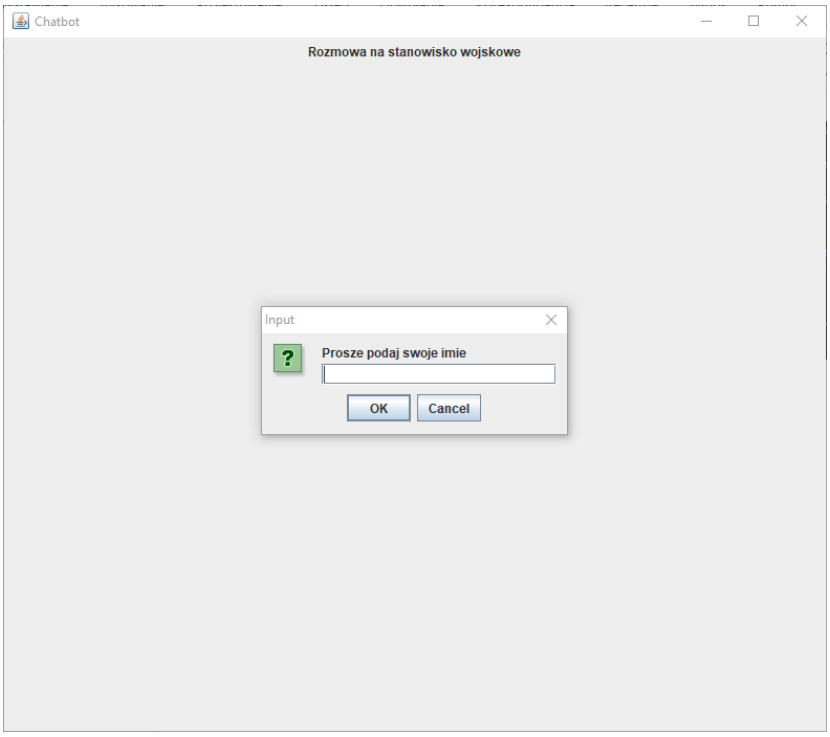
Wyswietl odpowiada za stworzenie dla każdego pytania interfejsu graficznego i zadanie samego pytania. Kliknięcie odpowiedzi jako Yes zwraca 0, a No lub Cancel zwraca 1 i 2. Przypisuje tutaj także odpowiedzi 0 czyli Yes jako „tak” a pozostałym „nie”.

```
126 koniec(Kandydat, Decyzja) :-
127     jpl_new('javax.swing.JFrame', ['Chatbot'], Frame),
128     jpl_new('javax.swing.JLabel',['Rozmowa na stanowisko wojskowe'],Label),
129     jpl_new('javax.swing.JPanel',[],Pan),
130     jpl_call(Pan,add,[Label],_),
131     jpl_call(Frame,add,[Pan],_),
132     jpl_call(Frame, setLocation, [200,50], _),
133     jpl_call(Frame, setSize, [800,700], _),
134     jpl_call(Frame, setVisible, [@(true)], _),
135     jpl_call(Frame, toFront, [], _),
136     jpl_new('javax.swing.JOptionPane', [], JOP),
137
138     ((Decyzja == nie_zolnierz) ->
139         Decyzja2 = 'Przykro nam ale nie spelniasz podstawowych kryteriow. Nie mozesz przejsc dalej. Zostajesz odrzucony.'
140     ;(Decyzja == moze_zolnierz) ->
141         Decyzja2 = 'Niestety ale nie zakwalifikowales sie w pierwszej turze ale wciaz nasz szanse w drugiej.'
142     ;(Decyzja == dobry_zolnierz; Decyzja == idealny_zolnierz) ->
143         Decyzja2 = 'Gratulacje! Idealnie nadajesz sie na zolnierza. Na pewno sie do Ciebie odezwiemy.'
144     ;(Decyzja == inne) ->
145         Decyzja2 = 'Niestety zostales odrzucony. Nie pasujesz do schematu zolnierza. Pamietaj takze, ze wyrazenie zgody na testy jest obowiazkowe by przejsc
        dalej.'
146     ;(Decyzja == ewentualnie) ->
147         Decyzja2 = 'Nie moglismy dopasowac twoich odpowiedzi do naszych wymagan dlatego zostaly Ci zadane dodatkowe pytania. Z twoich odpowiedzi wynika, ze
        zostaniesz powolany tylko w przypadku braku personelu.'
148     ),
149
150     jpl_call('javax.swing.JOptionPane', showMessageDialog, [Frame,Decyzja2], _),
151     jpl_call(Frame, dispose, [], _).
```

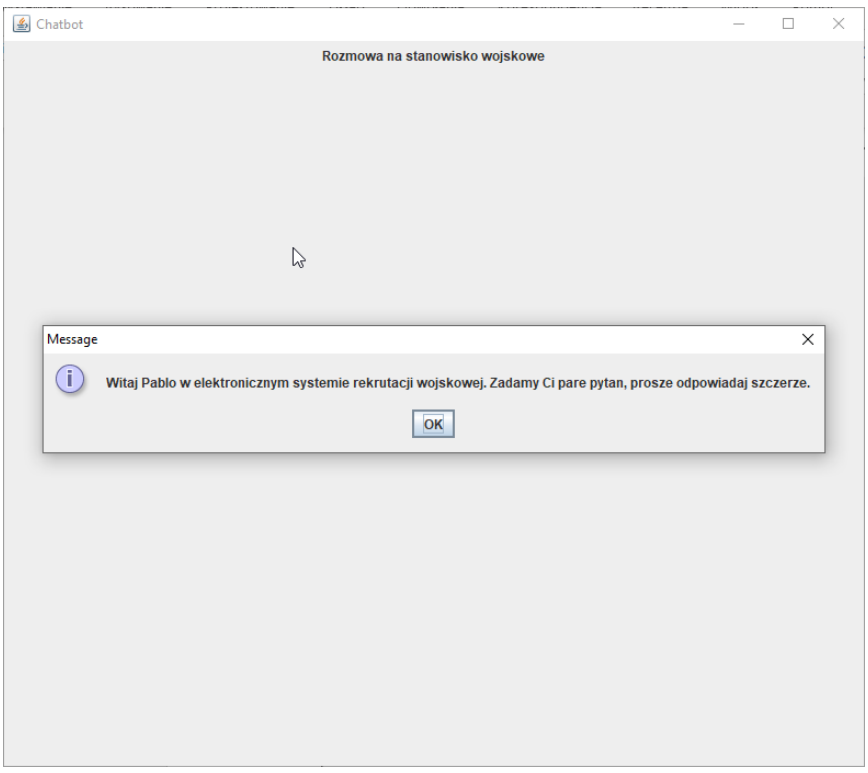
Koniec jest ostatecznym interfejsem programu. Poza ponownym stworzeniem interfejsu graficznego, odpowiada on za wypisanie decyzji w sposób przyjemny i czytelny dla oka (sprawdza decyzje i przypisuje drugiej decyzji opisowy tekst).

Demonstracja działania programu:

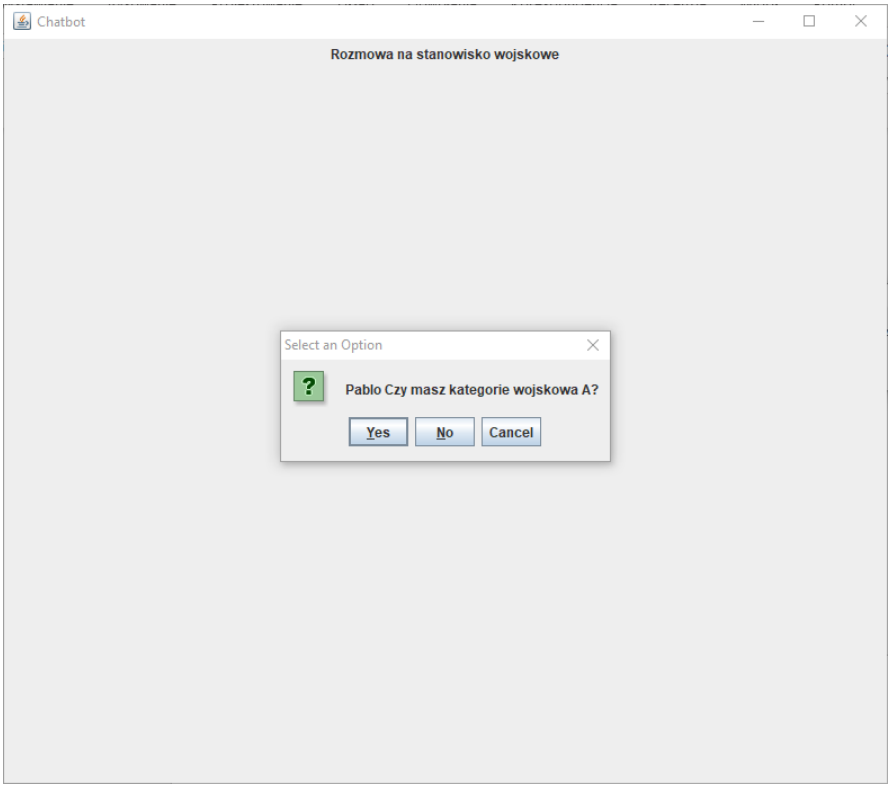
Pierwsze uruchomienie:



Wpisanie imienia:



Zadawanie pytań:



Określenie decyzji: