**1. Titlu**

Recomandare jocuri video

**2. Echipa**

Danci Cosmin - baza cunostite

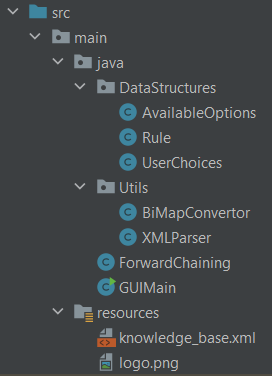
Crista Bogdan - inferenta

Gengiu Sebastian - interfata

**3. Descriere**

Scopul acestei aplicatii este cel de a ajutarea persoanelor de a-si gasi un joc pe gustul lor in functie de anumite optiuni selectate de acestia in interfata grafica, astfel aplicatia ii va recomanda unul sau mai multe titluri de jocuri video.

**4. Cod**

 Acest proiect are structura din poza alaturata:

Pentru inceput avem cele 3 clase folosite ca structuri de date si anume: **AvailableOptions**, **Rule**, **UserChoices**. Aceste clase contin doar metode de “get” si date, fara a avea metode care sa tina de logica programului.

Clasa **AvailableOptions** are ca tipuri de date, 4 String Arrays, corespunzatoare celor 4 categorii de optiuni disponibile, acestea fiind setate in constructor si preluate prin metodele de get.

Clasa **Rule** constituie clasa pentru fiecare obiect de tip regula, avand ca tipuri de date String, cele 4 conditii + 1 concluzie, acestea fiind setate in constructor si preluate prin metodele de get.

Clasa **UserChoices** este clasa corespunzatoare alegerilor facute de utilizator in interfata grafica. Aceasta are ca tipuri de date 4 String-uri, setate in constructor si preluate prin metodele de get, cu mentiunea ca metoda pentru varsta va returna un numar, nu un string.

Pentru clasele de utilitare avem: **BiMapConvertor** si **XMLParser**.

Scopul clasei **XMLParser** este acela de a prelua datele din fisierul xml, a le prelucra si furniza mai departe.

Prima metoda, ***private void doGeneralSetup()***, care are responsabilitatea de a prelua datele din fisier si a le pune sub forma de obiecte, folosind interfata DOM.

A doua metoda, ***public AvailableOptions getAvailableOptions()***, care are ca responsabilitate de a apela metoda de setup general apoi de a construii obiectul de tip AvailableOptions cu datele obtinute si de a-l returna.

A treia metoda, ***public ArrayList<Rule> getRulesArrayList()***, care are ca responsabilitate de a apela metoda de setup general apoi de a construii un ArrayList de reguli cu datele obtinute si de a-l returna.

Scopul clasei **BiMapConvertor** este acela de a crea un dictional bidirectional pentru a face “traducerea” dintre termenii specifici unui sistem expert si termenii folositi de un utilizator simplu. Acest lucru se face cu ajutorul frameworkului Guava. Clasa contine un tip de date *ImmutableBiMap* care este obiectul dictionar folosit.

Prima metoda, ***private void createBiMap()***, are responsabilitatea de a crea dictionarul.

Exemplu de intrare in acest dictionar: .put("Singleplayer", "Prefera sa se joace un joc singur/a").

A doua metoda, ***public UserChoices guiToKnowledgeBaseFormat(String spMp, String platform, String genre, String contentRating)***, are ca responsabilitate de a face conversia datelor din limbaj natural (care ar trebuii sa vin interfata grafica) in limbajul sistem expert, returnand un obiect UserChoices.

A treia metoda, ***private String[] convertStringArrays(String[] strings)***, este o metoda generica care are ca rolul de a ajuta metoda ***knowledgeBaseToGuiFormat(...)***, aceasta facand conversia din limbajul de tip expert din knowledge base in limbajul natural gasit in interfata grafica.

A patra metoda, ***public AvailableOptions knowledgeBaseToGuiFormat(String[] smMp, String[] platform, String[] genre, String[] contentRating)***, se foloseste de functionalitatea metodei ***convertStringArrays(...)*** astfel returnand un obiect de tip AvailableOptions.

Clasa **ForwardChaining** este clasa care contine logica corespunzatoare metodei de forward chaining, aceasta avand ca tipuri de date continute un obiect de tip ArrayList<Rule> si un obiect de tip UserChoices, ambele fiind setate in constructor.

Prima metoda, ***public void doInference()***, este responsabila pentru eliminarea din lista a tuturor regulilor care nu respecta toate conditiile pe care utilizatorul le-a ales, acestea fiind gasite in obiectul UserChoices.

A doua metoda, ***public String getResultedRecommendations()***, are responsabilitatea de a construii stringul final care contine toate concluziile regulilor ramase in lista de reguli, dupa inferenta.

Clasa **GUIMain**, clasa principala responsabila pentru interfata grafica a utilizatorului. Aceast interfata grafica este realizata cu Java Swing. Aceasta clasa contine doua metode principale.

Prima metoda, ***public void initializeUI()***, este metoda principala de initializare a interfetei grafice. Aceasta are responsabilitatea de a crea, organiza si afisa interfata grafica avand ca elemente Frame-ul principal care contine diferite panouri, combo box-uri (optiunile pentru utilizator) si un buton.

A doua metoda, ***public void getConclusion()***, este metoda care se activeaza in momentul in care se apasa butonul. Aceasta este responsabila de a prelua datele selectate de utilizator, de a executa metoda de inferenta si de a extinde interfata prin adaugarea unui panou care are ca componente concluzia primita in urma inferentei.

**5. Țintă**

Aceasta aplicatie are ca public tinta orice pasionat sau nepasionat de jocuri video care este in cautarea unui joc PC sau consola. Interfata aplicatiei este una foarte simpla si intuitiva cu folosind terminologie simpla si cunoscuta de marea majoritate a lumii astfel facand ca aceasta aplicatie sa fie utila si pentru persoanele care nu au cunostinte in domeniul de gaming.

**6. Concluzie**

O curenta limitare a acestui program este faptul ca se va afisa rezultatul final acesta va fi sub forma de text si cam atat. O posibila extindere ar fi prin afisarea in interfata a unei poze si/sau descriere pentru fiecare joc astfel utilizaorul va avea la dispozitie mai multe informatii.