Documentatie Proiect: Simulator de negociere pentru achizitii

Autor: Diacoenescu Rares Theodor

Grupa: CR2.1B

Facultatea: Facultatea de Automatica, Calculatoare si Electronica, Universitatea din Craiova

Profesor curs: Ganea Eugen

Profesor laborator: Chihaia Valentin

Titlul proiectului: Simulator de negociere pentru achizitii

# Descrierea proiectului

Proiectul 'Simulator de negociere pentru achizitii' reprezinta un sistem software ce simuleaza un proces de negociere in cadrul achizitiilor, bazat pe stilurile de negociere ale participantilor si pe disponibilitatea acestora de a face tranzactii. Acesta include un sistem de matching intre cumparatori si vanzatori, avand ca scop crearea unor scenarii de negociere realiste. Proiectul se bazeaza pe concepte de programare orientata pe obiecte (OOP), fiind structurat pe clase si metode ce permit simularea si gestionarea procesului de negociere.

# Functionalități și Atribute

Proiectul include mai multe clase, fiecare avand functionalitati specifice care contribuie la simularea completa a unui proces de negociere. Mai jos sunt descrise clasele si functionalitatile acestora.

## Clasa Participant

Clasa Participant reprezintă un participant într-o negociere și include toate informațiile necesare pentru a gestiona stilul de negociere și ajustarea prețului pe parcursul procesului. Aceasta conține următoarele atribute și metode:

### Atributele clasei:

1. **name**: Numele participantului (string).
2. **negotiationStyle**: Stilul de negociere al participantului (string).
   * Poate fi una dintre următoarele valori: "Conservative", "Moderate", "Flexible". Acest atribut definește modul în care participantul va ajusta prețul în timpul negocierii.
3. **adjustmentRate**: Rata de ajustare a prețului per rundă (double).
   * Reprezintă procentul cu care participantul va ajusta prețul în fiecare rundă de negociere, în funcție de stilul său de negociere.

### Metodele clasei:

1. **generateRandomPercentage(double min, double max)**:
   * Este o metodă privată care generează un procentaj aleatoriu între valorile min și max folosind un generator de numere aleatoare. Acest procentaj este folosit pentru a adăuga o variație la rata de ajustare a prețului.
2. **Participant(const string& name, const string& style)**:
   * Constructorul clasei, care primește numele participantului și stilul său de negociere. În funcție de stilul de negociere, constructorul setează valoarea ratei de ajustare a prețului:
     + Pentru stilul "Conservative", rata de ajustare este de 5% plus o valoare aleatorie între 0 și 5%.
     + Pentru stilul "Moderate", rata de ajustare este de 10% plus o valoare aleatorie între 0 și 5%.
     + Pentru stilul "Flexible", rata de ajustare este de 15% plus o valoare aleatorie între 0 și 5%.
3. **getName() const**:
   * Returnează numele participantului.
4. **getStyle() const**:
   * Returnează stilul de negociere al participantului.
5. **adjustPrice(double& price)**:
   * Este o metodă virtuală care ajustează prețul pe baza ratei de ajustare a participantului. Se calculează un procentaj din prețul curent și se adaugă la prețul inițial. Această metodă va fi suprascrisă în clasele derivate pentru a implementa comportamente specifice de ajustare a prețului.

## Clasa Negotiation

Clasa Negotiation reprezintă procesul de negociere între un cumpărător (Buyer) și un vânzător (Seller). Aceasta gestionează runde de negociere, ajustează prețurile pe baza stilurilor de negociere ale celor două părți și determină dacă s-a ajuns la un acord. Clasa include atribute pentru prețurile inițiale și curente ale ambelor părți, numărul de runde de negociere, și probabilitatea de eșec timpuriu.

### Atributele clasei:

1. **buyer**: Referință către obiectul de tip Buyer (tip Buyer&).
2. **seller**: Referință către obiectul de tip Seller (tip Seller&).
3. **initialBuyerPrice**: Prețul inițial propus de cumpărător (tip double).
4. **initialSellerPrice**: Prețul inițial propus de vânzător (tip double).
5. **currentBuyerPrice**: Prețul curent al cumpărătorului (tip double).
6. **currentSellerPrice**: Prețul curent al vânzătorului (tip double).
7. **maxRounds**: Numărul maxim de runde de negociere (tip int).
   * Reprezintă numărul maxim de runde în care negocierea va continua. Dacă se ajunge la acest număr, negocierea se încheie automat.
8. **currentRound**: Runda curentă de negociere (tip int).
9. **failureProbability**: Probabilitatea de eșec timpuriu a negocierii (tip double).
   * Reprezintă probabilitatea ca negocierea să eșueze înainte de a ajunge la un acord. Aceasta este o valoare între 0.01 și 0.1.

### Metodele clasei:

1. **shouldFailEarly()**:
   * Este o metodă privată care determină dacă negocierea va eșua înainte de a ajunge la un acord. Se bazează pe o valoare aleatorie generată între 0 și 1, comparată cu probabilitatea de eșec (failureProbability). Dacă valoarea aleatorie este mai mică decât probabilitatea de eșec, negocierea se oprește.
2. **Negotiation(Buyer& buyer, Seller& seller, double buyerPrice, double sellerPrice, int maxRounds, double failureProbability)**:
   * Constructorul clasei, care primește referințele către obiectele de tip Buyer și Seller, prețurile inițiale ale acestora, numărul maxim de runde și probabilitatea de eșec. Acesta inițializează toate atributele clasei.
3. **conductRound()**:
   * Este metoda care simulează o rundă de negociere. În fiecare rundă, se verifică dacă negocierea va eșua devreme folosind metoda shouldFailEarly(). Dacă negocierea nu eșuează, se ajustează prețurile cumpărătorului și vânzătorului folosind metodele corespunzătoare ale acestora. Runda se încheie, iar negocierea continuă până la atingerea numărului maxim de runde.
4. **isAgreementReached() const**:
   * Este o metodă care verifică dacă s-a ajuns la un acord între cumpărător și vânzător. Acordul este considerat încheiat atunci când prețul cumpărătorului este mai mare sau egal cu prețul vânzătorului.

## Clasa NegotiationManager

Clasa NegotiationManager este responsabilă pentru gestionarea interfeței utilizatorului și a procesului de negociere între cumpărători și vânzători. Ea permite utilizatorului să vizualizeze informații despre cumpărători, vânzători și produse, să exploreze negocierile posibile și să inițieze negocieri între părți. Clasa include funcții pentru a interacționa cu utilizatorul printr-un meniu și pentru a începe negocieri pe baza selecțiilor făcute.

#### Atributele clasei:

1. **buyers**: Un vector de obiecte de tip Buyer (tip vector<Buyer>).
   * Reprezintă lista cumpărătorilor disponibili pentru negociere.
2. **sellers**: Un vector de obiecte de tip Seller (tip vector<Seller>).
   * Reprezintă lista vânzătorilor disponibili pentru negociere.
3. **products**: Un vector de obiecte de tip Product (tip vector<Product>).
   * Reprezintă lista produselor disponibile pentru negociere.

#### Metodele clasei:

1. **displayMenu()**:
   * Este o metodă statică care afișează meniul principal al aplicației. Permite utilizatorului să aleagă opțiuni pentru a vizualiza informații despre vânzători, cumpărători, produse sau negocieri posibile. De asemenea, permite inițierea unei negocieri.
   * Parametri:
     + buyers: lista cumpărătorilor.
     + sellers: lista vânzătorilor.
     + products: lista produselor.
2. **startNegotiation()**:
   * Este o metodă statică care începe o negociere între un cumpărător, un vânzător și un produs selectat. După selectarea unui cumpărător și a unui produs, aceasta va începe procesul de negociere.
   * Parametri:
     + buyer: obiectul de tip Buyer care reprezintă cumpărătorul.
     + seller: obiectul de tip Seller care reprezintă vânzătorul.
     + product: obiectul de tip Product care reprezintă produsul ce va fi negociat.
3. **clearScreen()**:
   * Este o funcție privată care curăță ecranul consolei înainte de a afișa un nou meniu sau informații.

#### Detalii suplimentare:

* Meniul afișat de displayMenu() permite utilizatorului să:
  + Vizualizeze lista de vânzători, cumpărători și produse.
  + Vizualizeze posibilele negocieri pe baza preferințelor cumpărătorilor.
  + Inițieze o negociere selectând un cumpărător și un produs.
* În cadrul negocierii, utilizatorul va selecta un cumpărător din listă și va vizualiza produsele care se potrivesc preferințelor acestuia. După alegerea unui produs, va începe negocierea între cumpărător și vânzător.

## Clasa Buyer

Clasa Buyer reprezintă un participant la negociere care are un stil de negociere și o categorie preferată de produse. Aceasta extinde clasa Participant și implementează logica specifică pentru ajustarea prețurilor în cadrul negocierii.

### Atributele clasei:

* **preferredCategory**: Reprezintă categoria de produse preferată de cumpărător (ex: "Electronics", "Furniture", etc.).

### Constructor:

* **Buyer(const std::string& name, const std::string& style, const std::string& preferredCategory)**:
  + Inițializează un obiect Buyer cu numele, stilul de negociere și categoria preferată de produs.
  + Apelează constructorul clasei de bază Participant pentru a inițializa numele și stilul de negociere.

### Metodele clasei:

1. **getPreferredCategory()**:
   * Returnează categoria preferată a cumpărătorului.
2. **adjustPrice(double& offer)**:
   * Aceasta este o metodă suprascrisă din clasa Participant care ajustează prețul unei oferte în funcție de stilul de negociere al cumpărătorului.
   * **Conservative**: Reducerea ofertei cu un procent între 1% și 5%.
   * **Moderate**: Reducerea ofertei cu un procent între 5% și 10%.
   * **Flexible**: Reducerea ofertei cu un procent între 10% și 15%.
   * Metoda returnează noul preț ajustat.

### Implementarea metodei adjustPrice:

* **generateRandomPercentage(min, max)**: Această funcție, care se află în fișierul Utils.h, generează un procent aleatoriu între valorile minime și maxime date. Aceasta este folosită pentru a calcula ajustarea prețului în funcție de stilul de negociere al cumpărătorului.

## Clasa Seller

Clasa Seller reprezintă un participant la negociere care are un stil de negociere și poate ajusta prețul unui produs în cadrul negocierii. Aceasta extinde clasa Participant și implementează logica specifică pentru ajustarea prețurilor în funcție de stilul său de negociere.

#### Atributele clasei:

* **adjustmentRate** (tip: double):
  + Reprezintă rata de ajustare a prețului aplicată de vânzător, exprimată în procente. Această valoare determină cât de mult va reduce vânzătorul prețul unui produs în timpul negocierii, în funcție de stilul său de negociere.

#### Constructor:

* **Seller(const std::string& name, const std::string& style)**:
  + Inițializează un obiect Seller cu numele și stilul de negociere al vânzătorului.
  + Apelează constructorul clasei de bază Participant pentru a inițializa numele și stilul de negociere.
  + **Parametri**:
    - name: Numele vânzătorului.
    - style: Stilul de negociere al vânzătorului (de exemplu, "Conservative", "Moderate", "Flexible").

#### Metodele clasei:

1. **adjustPrice(double& price)**:
   * Aceasta este o metodă suprascrisă din clasa Participant care ajustează prețul unui produs în funcție de stilul de negociere al vânzătorului.
   * **Comportament**:
     + Reducerea prețului cu un procent determinat de stilul de negociere al vânzătorului.
     + **Stiluri de negociere**:
       - **Conservative**: Reducerea prețului cu un procent între 1% și 5%.
       - **Moderate**: Reducerea prețului cu un procent între 5% și 10%.
       - **Flexible**: Reducerea prețului cu un procent între 10% și 15%.
   * **Parametri**:
     + price: Prețul produsului care urmează să fie ajustat.
   * **Returnează**:
     + double: Returnează 0 în prezent, dar ar putea fi extins pentru a returna o valoare care indică succesul ajustării prețului.

#### Implementarea metodei adjustPrice:

Metoda adjustPrice calculează un procent aleatoriu în funcție de stilul de negociere al vânzătorului și ajustează prețul produsului. De exemplu, dacă stilul de negociere este "Moderate", prețul va fi redus cu un procent aleatoriu între 5% și 10%.

## Clasa Matching

Clasa Matching este responsabilă pentru realizarea unui proces de potrivire între un cumpărător, un vânzător și un produs. Aceasta conține o metodă statică care verifică dacă un anumit cumpărător și un vânzător sunt potriviți pentru a negocia un produs pe baza categoriei preferate a cumpărătorului și a categoriei produsului.

#### Metodele clasei:

1. **findMatch(const Buyer& buyer, const Seller& seller, const Product& product)**:
   * Verifică dacă un cumpărător și un produs sunt potrivite pe baza categoriei preferate a cumpărătorului și a categoriei produsului.
   * **Parametri**:
     + buyer: Obiect de tip Buyer, care reprezintă cumpărătorul.
     + seller: Obiect de tip Seller, care reprezintă vânzătorul.
     + product: Obiect de tip Product, care reprezintă produsul oferit.
   * **Returnează**:
     + bool: true dacă categoria preferată a cumpărătorului corespunde cu categoria produsului, altfel false.

#### Implementare:

Metoda findMatch compară categoria preferată a cumpărătorului cu categoria produsului. Dacă acestea sunt identice, se consideră că există o potrivire și metoda returnează true. Altfel, returnează false.

#### Metodele clasei:

1. **findMatch(const Buyer& buyer, const Seller& seller, const Product& product)**:
   * Verifică dacă un cumpărător și un produs sunt potrivite pe baza categoriei preferate a cumpărătorului și a categoriei produsului.
   * **Parametri**:
     + buyer: Obiect de tip Buyer, care reprezintă cumpărătorul.
     + seller: Obiect de tip Seller, care reprezintă vânzătorul.
     + product: Obiect de tip Product, care reprezintă produsul oferit.
   * **Returnează**:
     + bool: true dacă categoria preferată a cumpărătorului corespunde cu categoria produsului, altfel false.

#### Implementare:

Metoda findMatch compară categoria preferată a cumpărătorului cu categoria produsului. Dacă acestea sunt identice, se consideră că există o potrivire și metoda returnează true. Altfel, returnează false.  
  
Clasa Matching furnizează o metodă statică pentru a verifica dacă un cumpărător și un produs sunt potrivite pentru negociere pe baza categoriei preferate a cumpărătorului. Aceasta ajută la realizarea unui proces de potrivire eficient între participanți în cadrul unui sistem de negociere.

## Clasa DataLoader

Clasa DataLoader este responsabilă pentru încărcarea datelor dintr-un fișier de text și popularea colecțiilor de obiecte de tipul Buyer, Seller și Product. Aceasta permite citirea datelor dintr-un fișier structurat, cu secțiuni pentru vânzători, cumpărători și produse, și adăugarea acestora în vectorii corespunzători.

#### Metodele clasei:

1. **loadFromFile(const std::string& filename, std::vector<Buyer>& buyers, std::vector<Seller>& sellers, std::vector<Product>& products)**:
   * Citește un fișier de tip text care conține informații despre vânzători, cumpărători și produse, și adaugă obiectele corespunzătoare în vectorii buyers, sellers și products.
   * **Parametri**:
     + filename: Calea fișierului care conține datele de încărcat.
     + buyers: Vectorul în care vor fi adăugați cumpărătorii încărcați din fișier.
     + sellers: Vectorul în care vor fi adăugați vânzătorii încărcați din fișier.
     + products: Vectorul în care vor fi adăugate produsele încărcate din fișier.
   * **Comportament**:
     + Deschide fișierul specificat de filename și citește fiecare linie.
     + Dacă linia conține un comentariu (secțiune marcată cu #), schimbă secțiunea curentă (pentru vânzători, cumpărători sau produse).
     + Citește datele corespunzătoare fiecărei secțiuni și creează obiectele corespunzătoare de tip Seller, Buyer sau Product.
     + Dacă un produs nu are un vânzător valid, afișează un mesaj de avertizare.

### Concluzie:

Clasa DataLoader facilitează încărcarea datelor dintr-un fișier structurat și popularea obiectelor de tip Buyer, Seller și Product. Aceasta asigură o separare clară între diferitele tipuri de date și permite o încărcare eficientă a informațiilor într-un sistem de negociere.

## Clasa Product

Clasa Product reprezintă un produs disponibil pentru negociere. Aceasta conține informații despre numele, prețul, categoria și proprietarul produsului, care este un obiect de tip Seller. Clasa oferă metode pentru accesarea acestor informații.

#### Atributele clasei:

* **name** (tip: std::string):
  + Reprezintă numele produsului (de exemplu, "Laptop", "Sofa", etc.).
* **price** (tip: double):
  + Reprezintă prețul produsului.
* **owner** (tip: Seller\*):
  + Este un pointer către obiectul Seller, care reprezintă proprietarul produsului. Acesta este responsabil pentru stabilirea prețului și negocierea acestuia.
* **category** (tip: std::string):
  + Reprezintă categoria produsului (de exemplu, "Electronics", "Furniture", etc.).

#### Constructor:

* **Product(const std::string& name, double price, Seller\* owner, const std::string& category)**:
  + Inițializează un obiect Product cu numele, prețul, proprietarul și categoria produsului.
  + **Parametri**:
    - name: Numele produsului.
    - price: Prețul produsului.
    - owner: Pointer către obiectul Seller care deține produsul.
    - category: Categoria produsului.

#### Metodele clasei:

1. **getName()**:
   * Returnează numele produsului.
   * **Returnează**:
     + const std::string&: Numele produsului.
2. **getPrice()**:
   * Returnează prețul produsului.
   * **Returnează**:
     + double: Prețul produsului.
3. **getOwner()**:
   * Returnează pointerul către obiectul Seller, care este proprietarul produsului.
   * **Returnează**:
     + Seller\*: Pointer către obiectul Seller.
4. **getCategory()**:
   * Returnează categoria produsului.
   * **Returnează**:
     + const std::string&: Categoria produsului.

Clasa Product reprezintă un produs disponibil pentru negociere, având atributele esențiale pentru descrierea acestuia, precum numele, prețul, categoria și proprietarul. Aceasta furnizează metode pentru accesarea acestor informații.

# Concluzii

Proiectul 'Simulator de negociere pentru achizitii' reprezinta o implementare a unui sistem de simulare a procesului de negociere, folosind concepte de programare orientata pe obiecte si avand ca scop crearea unor scenarii realiste de negociere. Acesta poate fi folosit atat pentru studii de caz, cat si pentru antrenamente in domeniul negocierii.