Tema

Programare Orientata pe Obiecte

Student: Hodoboc-Velescu Tudor

Grupa: 325CC

Grad de dificultate: mediu-dificil

Timp alocat rezolvarii temei: 35-40 ore

Detalii despre modeul de implementare

Cerinta 1 – Arhitectura aplicatiei

1.1 Application

Au fost implementate toate metodele de add/remove/get date in cerinta.
Metoda getJobs traverseaza lista de companii si verifica daca acestea se regasesc intre cele date ca argument. In caz afirmativ, se vor prelua toate job-urile disponibile si vor fi adaugate intr-o lista separata. Aceasta din urma va fi intoarsa de catre metoda.

1.2 Consumer

Au fost adaugate toate metodele specificate in cerinta. Pe langa acestea au fost adaugate metode de get pentru diversele elemente ale clase(elementele din resume, lista de prieteni). Am adaugat si o metoda suplimentara, getTotalExperience, ce intoarce numarul de ani de experienta din intreaga istorie a consumatorului.
Metoad meanGPA parcurge intreaga lista de Education si ia nota corespunzatoare fiecareia. La final, se calculeaza media acestora si se intoarce.
Metoda getGraduationYear se uita prin toate elementele de tip Education, cautand pe cea care are gradul de “college” si intoarce anul absolvirii acestei etape. Daca nu este gasita etapa sau nu este terminate, se intoarce null.
Metoda getDegreeInFriendship face o parcurgere in latime prin intreaga retea de socializare a consumatorului, aplicand algoritmul bfs. Toti prietenii consumatorului de la pasul curent sunt verificati daca au fost vizitati. In caz

contrar, se verifica daca acestia sunt consumatorul cautat. Daca da, se intoarce gradul actual, daca nu sunt adaugati in coada pentru a li se verifica lista de prieteni. De asemenea, creste un contor. Dup ace s-a verificat toata lista(si nu a fost gasit consumatorul cautat), acest contor se va adauga la un numar ce va reprezenta toti consumatorii de pe nivelul urmator ce vor trebui sa fie verificati. Se decrementeaza un contor ce va reprezenta numarul de consumatori ramasi de verificat la nivelul actual. Cand se ajunge la 0, se schimba nivelul: consumatorii de pe nivelul urmator vor deveni nivelul actual, iar gradul va creste cu o unitate; pe nivelul urmator vom avea 0 consumatori notati acum. Daca sunt verificati toti consumatorii disponibili, se intoarce gradul maxim(pentru consumatorul nostrum) + 1.

1.3 Resume

Aceasta metoda a fost creata in interiorul clasei consumer. Are implementa un constructor, metode de set/add/get pentru fiecare element al acesteia

1.4 Information

Clasa contine datele personale ale consumatorului. Are implementate metodele de get/set pentru fiecare element al clasei si o metoda toString.

1.5 Education

Clasa are implementata un constructor, metode de get/set pentru fiecare element. De asemenea, are implemenatata si metoda compareTo, ce compara doua elemente de tip Education in modul descris in cerinta.

Tin sa precizez ca am implementat si o clasa MyDate ce modeleaza o data calendaristica. Aceasta implementeaza interfata comparable si are definite metodele de set/get pentru fiecare variabila. Aceasta clasa a fost implementata pentru a simplifica unele operatii viitoare.

Am descris si exceptia InvalidDatesException ce este aruncata cand datele de inceput si final din cadrul unui obiect de tip Education nu sunt conforme.

1.6 Experience

Clasa descrie experienta unui consumator. Are constructor, metode get/set pentru fiecare element. Asemenea clasei anterioare, ma folosesc de MyDate pentru a descrie datele de inceput si sfarsit si InvalidDatesException este

aruncata in momentaul in care acestea nu sunt conforme. Clasa implementeaza interfata Comparable si metoda compareTo, facandu-se ordonarea cum este descrisa in cerinta.

1.7 User

Pe langa un constructor, metode de get/set/remove pentru lista de companii, aceasta clasa are implementata metoda getTotalScore ce calculeaza scorul dupa formula data in cerinta, folosindu-se de metodele getTotalExperience si meanGPA implementate anterior.

De asemenea, am implementat metoda convert ce intoarce o instanta de tip employee. Se cauta job-ul in lista de job-uri a companiilor din aplicatie noastra. Se creaza obiectul employee cu informatia user-ului, un obiect de tip education al acestuia, noua companie si salariul asociat job-ului. Apoi sunt adaugate restul obiecte de tip education si experience si se intoarce obiectul obtinut.

1.8 Employee

Au fost implementate un constructor, metode de set/get pentru salariu si companie.

1.9 Recruiter

Clasa contine un constructor, metoda getRating si metoda evaluate. Se verifica daca user-ul indeplineste conditiile de angajare a job-ului folosindu-se metoda acestui, meetsRequirments. In caz afirmativ, se cauta compania in aplicatie, si se trimite o cerere managerului, iar user-ul este adaugat in lista de condidati a acestuia. In final, se intoarce score-ul obtinut de user.

1.10 Manager

Clasa contine un contructor, metode de add/remove a request-urilor. Am implementat o metoda hasRequests ce verifica daca mai exista request-uri neverificate pentru un anumit job.

Metoda precess verifica toate request-urile ce tin de job-ul dat ca parametru. Cererile sunt sortate descrescator dupa scorul obtinut.

Se verifica toate pana cand sunt umplute toate pozitiile sau nu mai sunt cereri de verificat. Daca user-ul nu a fost deja angajat, adica se mai afla inscris

in aplicatie, acesta va fi eliminat din lista de useri si va fi convertit la employee. Este adaugat un nou obiect de tip Education corespunzator noului job si este adaugat departamentului corespunzator. La final se verifica daca mai exista cereri neverificate, caz in care acestea sunt respinse si job-ul se inchide.

1.11 Job

Clasa contine un constructor, metode de set/get, metoda isOn verifica daca job-ul este disponibil, metoda close ce inchide job-ul. De asemenea, am implemenatete metodele meetsRequirments ce verifica daca un user indeplineste toate constrangerile necesare si metoda apply. Aceasta din urma verifica daca job-ul mai este disponibil. In caz afirmativ, se alege un recruiter din companie si user-ul este evaluat.

1.12 Constraint

Clasa are o limita inferioara si una superior. Are doua metode (fiecare pentru o variabila de tip double sau int) care intoarce o variabila booleana, in functie daca argumentul este sau nu in limitele acceptate.

1.13 Company

Clasa are metodele de add/move/remove/contains dorite, plus metode de get(spre exemplu pentru manager) pentru a usura unele operatii. Metoda getJobs verifica toate job-urile companiei si le adauga intr-o colectie pe cele care sunt disponibile, iar apoi intoarce colectia. Metoda getRecruiter verifica care recruiter are cel mai mare degree cu user-ul dat ca argument, folosindu-se metoda getDegreeInFriendship.

1.14 Department

Clasa are constructor, metode de add/remove/get.

1.15 IT

Se foloseste de constructorul clasei parinte. Metoda getTotalSalaryBudget face suma tuturor salariilor si o intoarce.

1.16 Management

Se foloseste de constructorul clasei parinte. Metoda getTotalSalaryBudget face suma tuturor salariilor, tinand cont de taxa de 16%, si o intoarce.

1.17 Marketing

Se foloseste de constructorul clasei parinte. Metoda getTotalSalaryBudget face suma tuturor salariilor, tinand cont de taxa ce variaza in functie cu salariul, si o intoarce.

1.18 Finance

Se foloseste de constructorul clasei parinte. Metoda getTotalSalaryBudget face suma tuturor salariilor, tinand cont de taxa ce variaza cu experienta, si o intoarce.

Cerinta 3 – Sabloane de proiectare

3.1 Singleton pattern

Clasa Application a fost create tinandu-se cont de acest design pattern. Astfel, avem o instanta a clasei, data privata, un constructor private si o metoda getInstance ce intoarce instanta unica a clasei. De asemenea, se foloseste lazy instantiation.

3.2 Observer pattern

Au fost create interfetele Observer si Subject cu metodele specifice. Clasa company a implementat Subject, iar User Observer. Astfel, pentru a aplica correct pattern-ul, s-a creat o lista de observers in clasa Company si metode de add/remove si notify pentru observers, acestea fiind apelate asa cum a fost descris in cerinta. Clasa User are implementata metoda update, ce afiseaza mesajul notificarii. De asemenea, am implementata clasa Notification pentru aceste metode.

3.3 Builder pattern

Clasa Resume a fost implementata pentru a putea fi instantiate folosindu-se de pattern-ul Builder. Astfel, construtorul este privat si nu poate fi apelat decat cu clasa interna ResumeBuilder. Clasa permite instantierea oricarui tip de resume, indiferent de numarul de obiecte de tip Experience sau Education. Asa cum este specificat in cerinta, constructorul arunca o exceptie de tip ResumeIncompleteException daca datele pentru instantiere nu sunt corecte.

3.4 Factory Pattern

S-a implementat o clasa DepartmentFactory ce creaza un obiect de tip Department in functie de necesitatea utilizatorului.