Urse Adrian Dinu
321CC
Grad de dificultate: 8/10
Timpul alocat: 3 saptamani
IDE:IntelliJ Idea

Clasa Application:
Am adaugat un field de tip Application instance, pentru a implementa Singleton pattern. Initial instanta este nula, deoarece am folosit instantiere intarziata.
Am facut metoda getInstance care creeaza o instanta a aplicatiei daca ea este nula, in caz contrar o intoarce.
Pe langa metodele din cerinta, am implementat alte metode auxiliare:
-Metode de cautare a unui user/employee/recruiter dupa id (folosite pentru a crea reteaua sociala)
-Metoda de cautare a unui user/employee/recruiter/manager dupa numele complet/nume (folosita pentru Profile Page)
-Metode de cautare a unui user/employee/recruiter/manager dupa email (folosite pentru sistemul de autentificare)

Clasa Consumer:

Clasa Resume:
Constructor: Am verificat daca Information sau Colectia Experienta sunt nule,in acest caz am aruncat exceptia ResumeIncompleteException.In caz contrar am create obiectul.
Am adaugat o clasa interna ResumeBuilder pentru a implementa Builder Pattern.

Metoda getDegreeInFriendship: Am creat o lista pentru a marca consumerii vizitati, o coada si un dictionar ce contine gradele de prietenie cu toti prietenii consumerului current(this).
Am pus consumer-ul current(this) in coada, l-am marcat ca vizitat si am pus gradul lui 0.
Am facut un while = cat timp coada mai are elemente, extrag primul consumer din coada si parcurg toti prietenii lui.Marchez ca vizitati prietenii lui, pun in coada prietenul current si ii setez gradul=gradul elementului extras din coada + 1.
In final, daca dictionarul contine consumer-ul dat ca parametru, atunci intorc gradul lui, in caz contrar intorc 0.
Metoda meanGPA – Parcurg colectia educatie, contorizez cate obiecte sunt si fac suma mediilor de finalizare. In final fac media.

Clasa Information
Am setat toate campurile ca private.
Am creat o clasa interna InformationBuilder, pentru a folosi Builder pattern.

Clasa Education
Constructor:Am verificat daca data de sfarsit este dupa data de inceput, in acest caz am create obiectul. In caz contrar am aruncat exceptia InvalidDatesException.
Metoda compareTo: Am facut comparare descrescatoare dupa data de sfarsit, daca sunt egale am comparat descrecator dupa media de finalizare.Daca nu se pot compara dupa data de sfarsit, am comparat crescator dupa data de inceput.

Clasa Experience:
Constructor:Am verificat daca data de sfarsit este dupa data de inceput, in acest caz am create obiectul. In caz contrar am aruncat exceptia InvalidDatesException.
Metoda compareTo: Am facut comparare descrescatoare dupa data de sfarsit.Daca nu se pot compara, atunci am comparat crescator dupa numele companiei.

Clasa User:
Implementeaza interfata Observer
Constructor:Am create obiectul si am adaugat utlizatorul in listele de observers ale companiilor pe care le urmareste.
Metoda getTotalScore:Am parcurs colectia Experienta, am calculat numarul anilor de experienta.In final am returnat numarul anilor de experienta *1.5 +meanGPA
Metoda update: (Observer Pattern)
Afiseaza mesajul notificarii date ca parametru

Clasa Recruiter:
Am calculat scorul = rating*user.getTotalScore.Am marit rating-ul cu 0.1.Am adaugat in lista de cereri ai managerului o noua cerere ce contine job-ul, user-ul, recruiter-ul si scorul. In final am returnat scorul.

Clasa Manager:
Metoda process:Am create o lista de cereri care contine doar cererile pentru jobul dat ca parametru.Am ordonat-o descrescator dupa scor.
Am create o lista de useri respinsi.
Am parcurs lista de cereri, am verificat daca mai exista locuri disponibile pentru job, am verificat daca user-ul mai este in lista de utilizatori ai aplicatiei si daca meetRequirements.In acest caz, l-am

convertit in employee, l-am notificat (ca a fost acceptat), l-am adaugat in departamentul corespunzator, am sters cererea lui din lista de cereri ai managerului, din lista de candidati ai jobului si din listele de observeri ale celorlalte companii.In caz contrar, am adaugat user-ul in lista de respinsi.
In final am notificat utilizatorii respinsi ca nu au fost acceptati si ca job-ul a fost inchis.

Clasa Job:
Metoda apply: Selectez recruiter-ul cel mai indepartat, folosind metoda getRecruiter din clasa company, apelez evaluate, adaug user-ul in lista de candidati ai jobului si in lista de observers ai companiei.
Metoda meetsRequirments:Am verificat daca user-ul current indeplineste cele 3 Constraint-uri.In acest caz, am returnat true.In caz contrar am returnat false.

Clasa Company:
Implementeaza interfata Subject
Am adaugat la campurile clasei o lista de observers si o notificare.
Metoda getRecruiter:Am parcurs lista de recruiteri din companie. Pentru fiecare recruiter calculez gradul de prietenie cu user-ul dat ca parametru, il retin pe cel care are gradul cel mai mare.Daca doi recruiteri au acelasi grad, atunci ei se compara dupa rating.
Am creat metodele addObserver, remove si notifyAllObservers pentru a implementa Observer Pattern.
Metoda getDepartment:
Intoarce un department dupa tip

Clasa Department:
Metoda Add(Job):Am adaugat jobul dat ca parametru in lista de joburi, am creat notificare noua, am setat notificare companiei, dupa am apelat notifyAllObservers pentru a notifica toti observatorii.
Am adaugat metoda abstracta getType, care este implementata in departamentele care mostenesc aceasta clasa. (folosita pentru a identifica tipul unui department)

Clasele IT,Management,Marketing,Finance:
Mostenesc clasa Department
Fiecare dintre ele implementeaza metodele getTotalSalaryBudget si getType.

Testarea aplicatiei:
Am folosit org.json.simple.
Am creat doua fisiere de intrare.Unul pentru companii si joburi si unul pentru reteaua sociala.
Am construit companiile pe baza datelor citite.
Am adaugat userii, angajatii, recruiterii si managerii.
Dupa am adaugat job-urile.
Utilizatorii aplica pentru joburile din companiile preferate.
Managerii proceseaza joburile disponibile.

Interfata grafica:
Am folosit IntelliJ GUI Designer.
Am implementat un sistem de autentificare.Pagina de login are 2 field-uri (email si password) si un buton de login.Dupa in functie de utilizator, Home Page-ul va avea butoane corespunzatoare tipului de utilizator.

Admin-ul are access doar la Admin Page, user-ul/recruiter-ul/angajatul au access doar la Profile Page, iar Manager-ul are access la Manager Page si Profile Page.
Admin Page contine 3 butoane Users, Companies si un buton de BACK care duce la Home Page. Butonul Users afiseaza o lista cu utilizatorii din aplicatie.Daca se selecteaza un utilizator din aplicatie, se afiseaza datele lui.Butonul Companies afiseaza o lista cu companiile din aplicatie.Daca se selecteaza o companie, se afiseaza departamentele acesteia.Daca se selecteaza un department se afiseaza o lista cu angajati si job-urile disponibile din acel department avand posibilitatea de a calcula bugetul total de salarii.

Manager Page contine o lista cu managerii din aplicatie si un buton de BACK, care duce la Home Page. Daca se selecteaza un manager, se afiseaza o lista cu cererile de angajare.Daca se selecteaza o cerere, se afiseaza detalii despre candidat, avand posibilitatea de a-l accepta sau de a-l refuza.

Profile Page contine un Text Field pentru a cauta user-ul si un buton de BACK, care duce la Home Page.Daca se gaseste user-ul cautat, atunci se afiseaza datele lui personale, educatia si experienta.

Home Page contine butoane corespunzatoare tipului de utilizator si un buton de Logout care duce la pagina de autentificare.

Pentru logare ca admin:
Username=admin,password=admin (admin,admin)
Pentru logare ca user:
Username= email-ul user-ului,password=numele user-ului scris cu minuscule (Ex. dedmund@gmail.com, edmund)

Pentru logare ca employee:
Username= email-ul angajatului, password=numele angajatului scris cu minuscule (Ex. harmonyl@gmail.com, lorinda)
Pentru logare ca recruiter:
Username= email-ul recruiter-ului, password=numele recruiter-ului scris cu minuscule (Ex. jwoody@gmail.com, woody)
Pentru logare ca manager:
Username= email-ul manager-ului, password=numele manager-ului scris cu minuscule (Ex. spichai@gmail.com, pichai)