Tema POO - We are Hiring - aplicatie de angajare - 2020-2021
Digori Parascovia - 323CC
Timp de lucru: implementare activa 2 saptamani
Dificultate: mediu spre avansat(sunt multe detalii)

Pentru verificare interfetei grafice:
Se va rula clasa LoginFrame din package-ul grafic, care permite autentificarea ca 3 instante:
->Autentificare ca user:
Username pentru toti userii - user
Parolele pentru useri: Edmund2021; Matvei2021; Haci2021; Spartak2021
->Autentificare ca admin:
Username admin: admin
Parola admin: 12345
->Autentificare ca manager:
Username pentru toti managerii: manager
Parolele managerilor: Google2021; Amazon2021

1. Daca esti logat ca admin poti vizualiza lista de companii si lista de useri ai aplicatiei.
Cu dublu click pe numele companiei din lista poti sa deschizi o fereastra cu datele companiei care contine 3 tabele: lista de angajati, bugetul pentru fiecare departament din companie si lista de job-uri.
In pagina de useri poti sa cauti user-ul inserand orice caractere din numele lui in search bar, cu dublu click pe numele user-ului se va deschide pagina de profil a sa cu toate informatiile din resume.

2. Daca esti logat ca user, odata ce te loghezi se va deschide pagina de profil a user-ului logat;

3. Daca esti logat ca manager, vei putea vizualiza lista de requests a manager-ului si lista de angajati in companie. Pentru a accepta un request se va apasa pe butonul accept si request-ul de va sterge, iar user-ul se va converti in employee si adauga in lista de employees a companiei.
Pentru a sterge un request si a nega user-ul se va apasa butonul denial, care doar va sterge request-ul din lista.

*Pentru testarea functionalitatii de baza a temei cu functia process, cerinta 1 si 2, trebuie decomentate din clasa Test liniile 550-564, se vor afisa angajatii din companie iar userii observeri vor primi notificare cand job-ul este inchis.

Concluzii generale din tema:
-> Dupa exemeplul dat in fisierul consumers.json se va angaja la Google-Tamara Haci si Julia Matvei, iar la Amazon se va angaja Linette Spartak; Daniel Edmund - va ramane user;
-> Am implementat pattern-urile: Singleton, Observer, Factory, Builder, Strategy(ca parte din bonus)
-> Logarea utilizatorilor sub diferite instante ca parte din bonus.

Functionalitatea de baza a aplicatiei este in package temaPOO in care am implementat:
1. Application - reprezinta aplicatia noastra, contine lista de companii si useri. Implementeaza pattern-ul Singleton pentru a pastra o singura instanta a clasei Applicaiton peste tot.
2. Consumer - clasa abstracta care reprezinta utilizatorul uman al aplicatiei noastre.
Contine clasa interna Resume care implementeaza pattern-ul Builder si lista de cunoscuti a consumatorului. Am mai adaugat campurile visited si grad in consumator pentru a ma ajuta sa determin daca cosnumatorul a fost sau nu visitat cand implementez bfs-ul din getDegreeinFriendship() si sa mentin grad-ul acestuia.
Pe langa metoda getGraduationYear am implementat dupa acelasi principiu si getExperienceYears() care o sa ma ajute ulterior in alte metode.
Metoda getDegreeInFriendship() primeste ca parametru un consumer si determina gradul de prietenie dintre 2 consumatori, efectuand un bfs.
3. Resume - clasa interna lui consumer, implementeaza pattern-ul builder si retine obiectul de tip Information, colectia de Education si colectia de Experience.
4. Information - contine toate datele personale ale consumatorului, citite din fisier;
5. Education/Experience - este clasa care va retine informatiile referitoare la un ciclu de educatie/experienta al unui utilizator. Implementeaza interfata comparable si sorteaza lista de educatie/experienta dupa cum cere entuntul temei.
7. User - mosteneste clasa Consumer si reprezintă utilizatorul care își caută de muncă.
Aici mai retin un camp appliedJobs cu joburile la care a aplicat user-ul pentru a ma ajuta ulterior sa il sterg din lista de observeri a celorlate companii, in cazul in care este angajat undeva.

Pentru logare, user-ului i se va genera un username si o parola.
Deasemenea pentru user o sa citesc fisier-ul min_Salary.txt care arata preferintele unui user la angajare, pentru a putea implementa patter-ul Strategy, astfel user-ul va aplica doar la companiile care satisfac strategia user-ului: vrea un anumit salariu minim sau sa intre pe o anumita positie; Toti aici implementez si metoda update pentru patter-ul Observer;

8. Employee - mosteneste clasa Consumer si reprezintă un utilizator care este deja angajat într-o companie.
9. Recruiter - va mosteni clasa Employee si va fi adaugat atat in lista de employees in departamente dar si separat in lista de recruiteri a companiei;
10. Manager - extinde clasa Employee, contine o colectie de cereri de angajare si prin metoda process angajeaza automat userii cu cel mai mare scor in companie, in acord cu numarul de positii deschise la un job.
11. Job - contine informatiile aferente unui job si contrangerile sale, atunci cand user-ul aplica la un job mai intaii verific daca user-ul intalneste basic requirements pentru job, si daca da il adaug in lista candidatilor la job si il trimit recruiter-ului catre evaluare.
12. Constrant - contine contrangerile job-ului legate de anii de experienta, graduation year si GPA, am ales sa le pastrez ca string pentru ca sunt mai usor de manipulat ulterior.
13. Company - modelează o companie în cadrul aplicatiei si reprezinta subiectul pentru pattern-ul observer. Pe langa metodele de baza cerute am implementat si metodele necesare pentru a adauga, sterge trimite notificari observerilor din companie.
14. Department si DepartmentFactory - clasa abstractă care modeleaza departamentele de IT, Marketing, Management si Finance din comapanie. Implementeaza pattern-ul factory.
15. Test - detalii in comentarii, doar citesc fiecare categorie de consumatori si le adaug datele necesare. Pentru lista de conuscuti creez un Hashmap cu AllConsumers ca sa pastrez relatile asa cum sunt reprezentate in fisierul listAcquintances.txt
16. Notification - pentru pattern-ul observer, notification va fi de 3 tip-uri CLOSED_JOB, DENIAL si NEW_JOB;

Am folosit ca IDE pentru proiect IntelliJ.
Concluzie: tema este interesanta dar si chinuitoare, mi s-a parut mai ok implementare claselor decat a interfatei, se pare ca libraria swing nu e atat de flexibila cu pozitionarea elementelor.