[325CC]Obedeanu Marian-Alexandru

Readme Tema POO

Implementarea a fost facuta in IntelliJ Idea.

Inainte de a implementa clasele principale(Application, Consumer, Company, Departement, etc), am implementat initial cele 3 clase: Information, Education si Experience.

Pentru clasa Information am ales sa fac in plus o clasa cu numele “Language”, care este caracterizata de un nume si un languageLevel(care este un Enum in care am pus nivelurile “Beginner, Advanced si Experienced”). Am folosit clasa implementata pentru a crea un ArrayList de tip Language.

Am procedat asemanator pentru clasa “Education”, unde am creat un Enum cu numele “EducationLevel” in care am pus posibilele cicluri de educatie(Highschool, Bachelor, Master si Doctor), si am creat un ArrayList de tip EducationLevel.

In clasa “Application” am implementat “Singleton pattern”, creand un obiect de tip Application instantiat cu null, si o metoda de tip “Application” numita “getInstance” in care verific daca obiectul este null, si daca este, instantiez un obiect de tip “Application” cu argumentele “companies” si “users”, apoi returnez obiectul.

In clasa “Consumer”, am avut dificultati in a implementa metoda “getDegreeInFriendship” si nu am implementat-o conform cerintei, nestiind exact cum sa parcurg acel graf. Astfel ca am cautat initial in lista de persoane cunoscute, si daca un “Consumer” este in lista altei persoane, atunci am adaugat 1 la gradul de prietenie. Am cautat si daca doua persoane au fost la aceeasi companie, sau daca au invatat la aceeasi institutie, din nou, pentru fiecare din aceste doua criterii am adaugat 1 la gradul de prietenie. La final am returnat gradul de prietenie.

In clasa “Resume”, clasa interna a clasei “Consumer” am implementat “Builder pattern”, astfel ca am facut o clasa in “Resume” numita “Builder”, teoretic aceasta clasa ar fi trebuit sa fie de statica, insa IDE-ul(IntelliJ IDEA) imi dadea eroare in momentul in care o declaram de tip static, astfel ca am lasat-o de tip “public void”. In clasa nou creata, am creat inca 3 obiecte la fel ca in “Resume”, si am facut setteri pentru fiecare dintre aceste obiecte. In plus am facut o metoda de tip “Resume” cu numele “build” in care am returnat un obiect de tip “Resume” caracterizat de cele 3 obiecte.

In clasa “User”, am facut in plus un obiect de tip “Notification” pentru a implementa “Observer pattern”, care are un setter in metoda “update”. Din aceasta clasa mi s-a parut mai interesanta metoda “getTotalScore”. In aceasta metoda am folosit metodele din clasa “Date” pentru a primi anii de experienta, respectiv lunile in cazul in care diferenta dintre anul de inceput si cel de final este 0(pentru a aproxima prin adaos daca exista o vechime mai mare de 3 luni).

In clasa “Manager” am avut dificultati in implementarea metodei “process”. In aceasta metoda am luat o noua lista de “requests” si am adaugat doar elementele care corespund job-ului pentru care se face request-ul. Dupa care, a trebuit sa sortez elementele in functie de scor crescator, pentru a putea folosi un for de la 0 la numarul de angajati de care este nevoie si a parcurge lista. M-am folosit de clasa “Request”, si am creat un obiect de tip “User” si am dat cast din “Value1”(aceasta valoare intoarce user-ul care trebuie sa fie angajat pentru respectivul job) in user, apoi am verificat daca utilizatorul a fost

angajat sau nu(am verificat daca numele comapniei sale este “”, deoarece asa am initializat numele initial), daca user-ul nu a fost angajat am facut un obiect de tip “Resume”, si un obiect de tip “employee” folosind obiectul de tip “Resume”. Dupa care am cautat in toate departamentele pana am gasit job-ul, iar dupa ce l-am gasit am adaugat angajatul in departamentul corespunzator job-ului. Dupa ce am adaugat angajatul, am eliminat user-ul din lista de observeri. Apoi, am creat o notificare cu mesajul “You have not been accepted” si am notificat toti observerii care nu au fost acceptati la job.

In clasa “Job”, metoda care mi-a pus probleme a fost “apply”. Pentru aceasta metoda am creat in clasa “Company” o metoda (getRecruiter) care creeaza o noua lista de recruiteri sortata crescator si o returneaza. In metoda “apply” din “Job” am cautat recruiter-ul cel mai indepartat si daca l-am gasit am apelat metoda “evaluate”, iar daca nu s-a gasit dupa primul criteriu cerut am cautat recruiter-ul cu cel mai mare rating. In metoda “meetRequirments” am aflat intai anii de experienta in variabila “experience”, si apoi, daca s-au respectat toate constrangerile am returnat “true”, daca nu, “false”.

In clasa “Company”, am adaugat doua metode noi, una care adauga un job nou intr-un departament, si una care elimina un job. In aceste doua metode am creat cate o notificare pentru fiecare(“New job”, respectiv “This job is no longer available”). Aceste notificari sunt folosite in metoda “notifyAllObservers”. De asemenea, aici am implementat si celelalte doua metode pentru a implementa “Observer pattern”, si anume “addObserver” si “removeObserver”.

Am implementat in plus clasa “Factory” pentru implementarea pattern-ului cu acelasi nume. In aceasta clasa folosit un switch cu un string dat ca parametru, care instantiaza, dupa caz, un obiect de tipul celor 4 departamente(IT, Management, Marketing si Finance).

Consider ca dificultatea acestei teme(cel putin implementarea claselor) este medie. Singurul lucru pe care nu l-am inteles, a fost acea parcurgere in latime, drept pentru care nu am reusit sa o implementez cum trebuie. Timpul pe care l-am acordat rezolvarii acestei teme este de doua saptamani.