Raduna Daniel Florin, 325CC

Cerinta 1-Arhitectura aplicatiei

Pentru majoritatea claselor o sa explic cel putin o metoda implementata.

1. Application:

- public ArrayList<Job> getJobs(List<String> companies);

Pentru aceasta metoda am folosit un ArrayList de job-uri. Am parcurs lista de companii din aplicatie si lista cu nume de companii. Cand gasesc o companie cu un nume din acea lista, adaug in ArrayList-ul de joburi toate joburile de la compania respectiva.

2. Consumer:

- public int getDegreeInFriendship(Consumer consumer);

Pentru aceasta metoda am folosit un LinkedList de Consumer care sa imite o coada. Un HashMap cu perechi <Consumer, Integer>(visit) unde adaugam un Consumer si nivelul pe care se afla. Se adauga Consumerul curent in coada si in visit. Dupa aceea aplicam urmatorul algoritm: Scoatem un element din coada, parcurgem toti prietenii sai. Daca gasim Consumerul cautat returnam nivelul elementului + 1. Altfel verificam daca visit contine vecinul. Daca nu, il adaugam pe el si nivelul elementului scos + 1. Pentru asta folositm metoda get(Object key).

- public Integer getGraduationYear();

Aici am parcurs lista cu educatie din resume si am verificat daca nivelul de educatie este “licenta”. Daca da si data de sfarsit nu este null, punem intr-o variabila care la inceput este null, anul din data de sfarsit prin metoda getYear(). Daca nu se gaseste o educatie “licenta” sau daca data de sfarsit este null, metoda va returna null.

3. Resume: O clasa interna clasei Resume. Nu am implementat nicio metoda aici.
4. Information: O clasa cu date private in care folosim principiul incapsularii. Aici am scris doar gettere si settere.

5. Education: Aici in constructor verificam daca data de sfarsit este diferita de null si daca data de inceput este dupa data de sfarsit. Daca da, aruncam o exceptie InvalidDatesException. Am mai comparat doua obiecte Education prin interfata Comparable.

6. Experience: Este asemanatoare cu Education.

7. User:

- public Double getTotalScore();

Pentru aceasta metoda a trebuit sa aflam numarul de ani de experienta.. Asa ca am parcurs lista de experiente a user-ului. Si am adunat diferenta dintre data de inceput si data de sfarsit. Pentru asta am folosit metoda ChronoUnit.Days.between(). Care calculeaz numaul de zile diferenta intre doua date. Pentru a aproxima ca in enunt, am imparitit acest numar de zile la 365 si am folosit Math.ceil pentru adaos. Dupa aplicam formula.

8. Employee: In aceasta clasa nu e nicio metoda speciala

9. Recruiter:

- public int evaluate(Job job, User user);

In aceasta metoda luam o variabila in care punem rezultatul formulei din enunt.Dupa care parcurgem lista de companii din

Application si cand gasim o compania cu numele egal cu numele companiei jobului., adaugam in lista de Request a Managerului companiei, un Request cu parametrii din enunt. Apoi crestem ratingul recruiterului si returnam variabila cu rezultatul.

10. Manager:

- public void process(Job job);
  Parcurgem colectia de Request-uri, pentru fiecare Request verificam daca userul din acel Request este in lista de users din aplicatie. Daca da il stergem de acolo, il convertim la employee si il adaugam in lista de angajati a departamentului care contine jobul. Daca nu mai raman locuri disponibile pentru job, il "inchidem".

11. Job:

- public boolean meetsRequirments(User user);
  O metoda in care verificam daca un user respecta cele 3 Constraint-uri.

12. Constraint: O clasa care reprezinta limite pentru, anul absolvirii, media academica si anii de experienta.

13. Company:

- public Recruiter getRecruiter(User user);

O metoda in care calculam cel mai indepartat social recruiter fata de user. Cand gasim un recruiter la distanta maxima il adaugam intr-o lista. Daca size() este mai mare decat 1 il returnam pec el cu cel mai mare rating. Daca nu il returnam pe singurul recruiter din lista.

14. Department: Aici nu exista nicio metoda mai speciala.
15. IT, Management, Marketing, Finance: 4 clase care mostenesc Department si in care bugetul de salarii se schimba dupa anumite criterii.

Cerinta 3: Sabloane de proiectare

1. Singleton:
   Am folosit in clasa Application, si am creat o unica instanta pentru a parcurge listele de companii si de utilizatori.
2. Factory:

   Am folosit ca sa creez obiecte Department. Am creeat o clasa DepartmentFactory, in care cu o metoda factory() creez obiecte in functie de tipul acestora(IT, Management, Marketing, Finance).

3. Builder:
   Am folosit in clasa Resume.