//Ionica Madalina Cristina 322AA

Cerinta I

1.1, 1.2 si 1.3

Pentru primele 3 cerinte am implementat in mod obisnuit cele 2 ierarhii, ierarhia Produs, User si clasa Shopping Cart, in mod obisnuit, asemenea implementarilor din cadrul laboratoarelor.

Cerinta II

Pentru clasa Server, pe langa metodele de get, am implementat si functia ceruta de s set__UserID__ProductsCart__, unde am parcurs lista de useri din server cu un iterator, apoi am stocat intr-un int empty_user, id-ul userilor, am creat un obiect de tip ShoppingCart si am creat un pair din cele doua mentionate pentru a popula mapul cerut.

Cerinta III

3a.

Pentru cerinta 3a, primul query,pentru a determina toate espressoarele reduse si a le returna intr-o lista am urmat urmatorii pasi. Am creat doua liste, una in care am stocat produsele din server si una in care urma sa pun toate produsele ce indeplinesc conditiile de a fi REDUSE si de a face parte din categora espressor, apoi am returnat lista de produse ce indeplineau cerintele.

3b.

Destul de asemanator cu modul in care am abordat si primul subpunct, am rezolvat si subpunctul b, de data aceasta pentru Useri. Am creat doua liste de useri, una in care am stocat userii din server si una in care urma sa stochez userii ce au un cost de transport mai mic sau egal cu 11.5 si sunt in acelasi timp, useri premium. Astfel, dupa ce am iterat prin lista, am testat daca acestia indeplinesc conditiile si apoi am returnat lista de useri ceruta.

3c.

Pentru a gasit produsele resigilate si cu motivul “lipsa_accesorii” am creat initial doua liste de produse, una in care am stocat produsele din server si una in care am stocat produsele resigilate, dupa ce am testat daca sunt resigilate. Apoi am creat o lista de ResealedProduct unde pe care urma sa o populez cu produse resigilate ce su motivul, “lipsa_accesorii”. Pentru asta m-am folosit de dynamic cast pentru a avea acces la metoda getReason() din ReturnedProduct, de unde este derivata clasa ResealedProduct. Astfel am reusit sa adaug listei, produsele resigilate si care au ca motiv “lipsa_accesorii. Pentru ca functia trebuia sa returneze o lista de produse, am transformat in final lista de ResealedProduct intr-o lista de Product, iterand prin ea si incluzand elementele.

3d.

In mod similar am procedat si la 3d, pentru a gasi toate produsele alimentare si pentru a le sorta dupa nume, tara sip ret. Am creat doua liste, una pe care am populat-o cu produse din server si un pe care urma sa o populez cu produse alimentare. Apoi am creat un dynamic cast pentru a transfroma lista de produse intr-o lista de tip FoodProduct, cu scopul de a putea apela metoda din utility ale carei parametrii sunt de obiecte de tip FoodProduct*, cu scopul de a realiza sortarea dupa nume, tara de origine si pret. Metodele au fost create dupa modelul prezentat la 3f si apelate exact cum este indicat.

3e.

Pentru rezolvarea 3e, am creat o lisat de useri, unde am stocat toti userii din server, apoi am creat un map de frecventa, unde urma sa stochez judetul si id-ul userilor cu scopul de a-i numara, pentru a afla judetul cu cei mai multi useri. Am stocat in map judetul si id-uril apeland metodele (*ip)->getDeliveryData().getCounty() si (*ip)->getUserID(), apoi am iterat prin map si prin lista de useri, iar atunci cand judetul din map nu corespundea cu cel din lista de useri, cream un nou entry, in caz contrar, incrementam cu 1 numarul de utilizatori, cu scopul de a-i numara. Inainte de asta mi-am creat un nou string unde am stocat judetul cu scopul de a identifica judetul cu cei mai multi utilizatori.

Pentru a determina judetul cu cei mai multi utilizatori (judetul cu numar maxim de utilizatori), am declarant un maxim de utilizatori si un string unde aveam sa stochez judetul maxim cu scopul de a face comparatia, apoi am daca numarul meu de utilizatori era mai mare decat maximul declarant, maximul lua valoarea numarului, iar judetul maxim lua valoarea cheii mapului. In ultima parte, am verificat daca userii stau la casa, punand conditia data si daca locuiesc in judetul cu cei mai multi utilizatori, folosindu-ma de stringul declarant anterior, si in final am populat list ape care am sortat-o dupa id, folosind o metoda similara celor anterioare si am returnat-o.

3f.

Pentru ultimul subpunct am creat o lista de produse pe care am populat-o cu produse din server si doua liste de useri, una in care am stocat userii si una in care am stocat userii premium. Am iterat prin lista de useri premium si am facut dynamic cast pentru a accesa metoda de get discounts. Am iterat prin map si am creat un id_product unde am stocat cheia, adica id-ul produsului si l-am transformat in string pentru a face testele urmatoare, deoarece id-ul nu avea un numar de cifre fix. Am creat si o variabila de tip bool pe care am intializat-o cu false (0) pe care o sa o folosesc ulterior. Am iterat prin lista de produse si daca id-ul produsului din lista coincide cu id-ul produsului din map, atunci bool lua valoare true si ma opream din cautata, fiindca gasisem produsul. In continuare am facut un nou if unde am tesat daca produsul are urmatoarele caracteristici: face parte din produsele de pe server, este “telefon” sau “imprimanta” si pentru ca orice produs poate fi redus, a trebuit sa testez daca acesta are prima cifra 2, 3, 4 sau 5 adica putea fi orice fel de produs, in afara de un produs alimentar. Odata indeplinite conditiile populez lista.

Cerinta IV

Pentru LRU Cache, pe langa implementarile de set si get, am rezervat in vector capacitatea data ca parametru, apoi am testat cele doua cazuri prezentate in exemplul dat, atunci cand cahe este plina, adica capacitatea si dimensiunea coincide, sterg ultimul element si adaug la inceput, practice, imping toate

elementele la dreapta si ii fac loc primului, in caz contrar si fericit, daca exista spatiu, adaug simplu la inceput.

Cerinta V

Pentru cerinta bonus, in afara de metodele de get si set clasice, am creat doua metode care cresc si scad cantitatea cu cantitatea ceruta, respectiv verifica daca daca cantitatea este existenta in cos. Cele mai importante metode au fost cele de add si delete, care au fost destul de similare ca si concept, am verificat intai daca cantitatea exista in cos, daca nu returnez 0. Am verificat daca userii din lista exista, folosind userId, in caz contrar return 0, respectiv acelasi lucru si la Product, apoi am tratat cazurile pentru cantitati. Am stocat cantitatea din shopping cart si am comparat cu cantitatea data ca parametru.

Pentru add, daca produsul nu exista in cosul de cumparaturi, il adaugam si modificam cantitatea produsului pe server, decrease, iar daca exista, cresteam cantitatea si actualizam cantitatea de pe server.

Pentru delete, daca produsul nu exista in cosul de cumparaturi, returnam 0, iar daca exista, aveam 2 cazuri, cand vreau sa sterg o cantitate mai mica deact cea existent, scad cantitatea si o cresc in server, daca vreau sa sterg o cantitate mai mare sau egala, adaug in server cantitatea initiala din cos si sterg produsul din cos.