Kako realizovati sve funkcionalnosti?

1. Registracija korisnika

Korisnik može da otvori svoj račun na način da popuni obrazac sa podacima: (naziv, e-mail, broj telefona, datum rođenja, šifra) i bira pri tome da li će njegov račun biti korišten u svrhu prodaje (prodavač) ili kupovine (kupac). Nakon unosa podataka, klijent bira opciju NapraviProfil pri čemu se vrši validacija (račun sa istim podacima ne smije već postojati, šifra mora biti validna (određeni broj karaktera..) i sl), te ako su podaci validni poziva se metoda DodajKorisnika, koja dodaje novog korisnika u bazu podataka.

Dakle, postojat će apstraktna bazna klasa Korisnik čiji će *atributi* biti podaci o tom korisniku:

String naziv,

String OpisProfila,

String e-mail,

Datum datumRodjenja,

String broj telefona,

Račun račun,

decimal Recenzija,

String adresa,

String šifra

METODE: get, set, IzmijeniProfil (kasnije navedene detaljne metode), PromijeniRecenziju...

Iz ove klase Korisnik će biti naslijeđene klase Kupac i Prodavač (veza između klasa: Generalizacija označava da neka klasa nasljeđuje drugu klasu).

Ovdje uviđamo da će postojati potreba za kontejnerskom klasom Prodavači, čiji će atribut biti lista Prodavača, tako da će objekat ove klase sadržavati sve prodavače koji već postoje, a metode koje će se odnositi na dodavanje ili brisanje prodavača bit će upravo metode vezane za ovu klasu.

(Veza između klasa Prodavači i Prodavač je veza Asocijacija -> Agregacija, koja označava da ukoliko se klasa koja posjeduje atribut druge klase obriše, druga klasa može nezavisno postojati -> ima smisla da postoji neki Prodavač i ako se izbriše skupina prodavača kojoj on pripada, jer može nastaviti postojati u nekoj drugoj skupini ili raditi „samostalno“).

Klasa: Prodavači

Atributi: list<Prodavač> prodavači;

Metode: get, set, DodajProdavaca, IzbrisiProdavaca...

Također, na sličan način bit će realizovana kontejnerska klasa Kupci, čiji će atribut biti lista Kupaca, tako da će objekat ove klase sadržavati sve kupce koji već postoje, a metode koje će se odnositi na dodavanje ili brisanje kupaca bit će upravo metode vezane za ovu klasu.

(Veza između klasa Kupci i i Kupac je veza Asocijacija -> Agregacija, koja označava da ukoliko se klasa koja posjeduje atribut druge klase obriše, druga klasa može nezavisno postojati -> ima smisla da postoji neki Kupac i ako se izbriše skupina kupaca kojoj on pripada, jer može nastaviti postojati u nekoj drugoj skupini).

Klasa:Kupci

Atributi: list<Kupac> kupci;

Metode: get, set, DodajKupca, IzbrisiKupca...

Na ovaj način, pomoću kontejnerskih klasa bit će omogućeno lakše upravljanje svim Korisnicima (Prodavačima i Kupcima), njihovo brisanje, dodavanje i sl...

Što se tiče klase Korisnik, ovdje je ispoštovan SRP (Single Responsibility Principe: Svaka klasa treba imati samo jednu ulogu; on, dakle, zahtijeva da svaka klasa ima samo jednu odgovornost, odnosno da klasa vrši samo jedan tip akcija kako ne bi ovisila o prevelikom broju konkretnih implementacija), što se odnosi na to da je ova klasa zadužena samo za funkcionalnosti vezane za profil korisnika, dakle ispunjava zahtjeve SRP da klasa treba imati samo jednu ulogu.

Također, klasa je otvorena za nadogradnje, ali zatvorena za modifikacije, čime je ispunjen OCP (Open/Closed Principle: Klasa treba biti otvorena za nadogradnje, ali zatvorena za modifikacije) : Moguće je da potrebe korisnika budu zahtijevale nove nadogradnje (da se dodaju još neke metode ili atributi prema potrebama Korisnika), ali ove već postojeće metode i atribute klase, nakon što se ispravno implementiraju, neće trebati mijenjati, jer predstavljaju osnovne funkcionalnosti.

Ovdje se može primjetiti i da je ispoštovan LSP (Liskov Substitution Principle: Svaka osnovna klasa treba biti zamjenjiva svim svojim podtipovima bez da to utječe na ispravnost rada programa), zbog toga što je bazna klasa definisana kao apstraktna, tako da će se, u suštini, uvijek „zamjenjivati“ klasama Kupac i Prodavač.

Također, može se primjetiti da je ispoštovan i DIP (Dependency Inversion Principle: Sistem klasa i njegovo funkcionisanje treba ovisiti o apstrakcijama, a ne o konkretnim implementacijama), tj. Ovaj princip zahtijeva da pri nasljeđivanju od strane više klasa bazna klasa uvijek bude apstraktna. Razlog za ovo je što je teško koordinisati veliki broj naslijeđenih klasa i konkretnu baznu klasu ukoliko ista nije apstraktna, a da pritom kod bude čitak i jednostavan za razumijevanje. Mi smo to upravo postigli definisanjem bazne klase Korisnik apstraktnom, nakon što smo odlučili da će iz nje biti naslijeđeno više klasa (Kupac i Prodavač).

Klasa Kupac (i kasnije klasa Gost) će implementirati interfejs PregledProizvoda(*) sa metodama: PogledajOpisProizvoda, PogledajCijenu, PogledajDetalje ... (veza Realizacija: se odnosi na korištenje operacija interfejsa od strane neke klase).

Ovdje je ispoštovan ISP (Interface Segregation Principle: Bolje je imati više specifičnih interfejsa, nego jedan generalizovani). Ovaj princip zahtijeva da i svi interfejsi zadovoljavaju princip S, odnosno da svaki interfejs obavlja samo jednu vrstu akcija, a ovaj interfejs zadužen je samo za akcije vezane za pregled proizvoda, što eventualno uključuje i akciju kupovine proizvoda, ali su sve akcije vezane za jednu klasu (klasu Proizvod).

Dakle, možemo zaključiti da je ovdje ispunjeno svih 5 SOLID principa.

Klasa Račun bit će realizovana radi plaćanja prilikom narudžbe (u slučaju Kupca, tj. Kada odabere način plaćanja karticom) i radi transakcije na račun Prodavača (kada neko od Kupaca naruči proizvod sa njegovog profila).

Klasa: Račun

Atributi:

String nazivBanke,

String brojRačuna,

Datum datumValidnostiKartice.

Metode: get, set, ProslijediPodatkeBanki.... (pošto će vanjski sistem vršiti transakcije vezane za kartice).

Veza između klase Račun i klase Korisnik -> Asocijacija: kompozicija.

Prijava korisnika

Prijava korisnika će se realizovati unosom e-maila i šifre, pri čemu je potrebno da se izvrši validacija unesenih podataka. To zahtijeva metodu ProvjeriPodatke za vec postojećeg korisnika.

Ukoliko su podaci validni, korisnik pristupa svom korisničkom računu.
Ukoliko podaci nisu validni, korisniku se šalje poruka o neuspješnoj prijavi na račun i zahtjev da ponovo unese podatke (Vjerovatno će biti realizovano bacanjem izuzetka prilikom neuspješne prijave, tako da korisnik može ponovo unositi podatke sve dok Prijava ne bude uspješna).

2. Dodavanje novih artikala u ponudu prodavača

Ovo nas već dovodi do procesa prodaje proizvoda, to znači da će postojati klasa Proizvod, sa sljedećim atributima:

Klasa: Proizvod

Atributi:

String naziv,

Statistika StatistikaProdaje (Veza između klase Proizvod i klase Statistika: Asocijacija -> Agregacija, koja označava da ukoliko se klasa koja posjeduje atribut druge klase obriše, druga klasa i dalje može nezavisno postojati -> ima smisla da neki proizvod idalje nezavisno postoji, iako se njegova statistika obriše),

String opisProizvoda,

Int IDProizvoda,

String kategorija,

Decimal cijena,

Statistika statistikaProdaje: (veza Statistika i Proizvod: Asocijacija-> Kompozicija: statistika prodaje nekog proizvoda ne može postojati samostalno niti se dodijeliti nekom drugom proizvodu ako se obriše Proizvod kojem ona pripada),

Int trenutnaZaliha,

Int minimalnaZaliha,

List<String> kljucneRijeci

Metode: get, set, IzmijeniProizvod -> DodajKljucnuRijec, PromijeniCijenu,

PromijeniOpisProizvoda, DefinirajMinimalnuKolicinuZaliha, postaviNovuTrenutnuZalihu, DodajAkcijuNaCijenu..

Ovdje uviđamo da će postojati potreba za kontejnerskom klasom Trgovina, koja će se „brinuti“ o svim proizvodima koji će postojati u ponudi svih prodavača (poput prave trgovine u kojoj se nalaze proizvodi polagani na policama). Dakle, metode koje se odnose na dodavanje proizvoda, brisanje proizvoda i sl. bit će vezane za ovu kontejnersku klasu Trgovina.

(Veza između klasa Proizvod i Trgovina je veza Asocijacija -> Kompozicija, koja označava da ukoliko se klasa koja posjeduje atribut druge klase obriše, druga klasa ne može nezavisno postojati -> nema smisla da postoje Trgovina ako se izbrišu svi Proizvodi koji čine ovu kontejersku klasu).

klasa: Trgovina

Atributi: list<Proizvod> proizvodi;

Metode: get, set, DodajProizvod, IzbrisiProizvod...

Očito je da će metodom IzmijeniProizvod, tj. metodama DodajKljucnuRijec, PromijeniCijenu,

PromijeniOpisProizvoda, DefinirajMinimalnuKolicinuZaliha, DodajAkcijuNaCijenu.. biti realizovana i funkcionalnost Modifikacija ponuđenih artikala, koju će moći obaviti isključivo prodavač u čijoj se ponudi nalazi taj proizvod.

Ovdje je ispunjen SRP SOLID princip, jer klasa Proizvod ima samo jednu odgovornost, vezanu za eventualne promjene proizvoda.

Također je ispunjen OCP Princip, jer su moguće nadogradnje klase, ali su trenutno navedene metode i atributi dovoljni da bi se pokrile osnovne funkcionalnosti, koje neće trebati modificirati.

(*) interfejs PregledProizvoda (jer će ga različite nepovezane klase implementirati: klasa Kupac i klasa Gost) : PogledajOpisProizvoda, PogledajCijenu, PogledajDetalje...

Ovdje je ispunjen ISP Princip.

Ovaj interfejs je osmišljen zato što se nudi mogućnost da se artikli pregledaju i ako je neko pristupio kao gost, a ne samo kao već prijavljeni kupac. Tada takav klijent ima mogućnost da pretražuje artikle i vidi njihove cijene, opise i detalje, ali nema opciju naručivanja (kupovine).

Zbog toga će vjerovatno postojati neka klasa Gost koja će također implementirati interfejs PregledProizvoda (PogledajOpisProizvoda, PogledajCijenu, PogledajDetalje).

(veza Realizacija između Gost i interfejsa PregledProizvoda: se odnosi na korištenje operacija interfejsa od strane neke klase).

3. Pretraživanje artikala

Bit će omogućeno pretraživanje proizvoda prema kategorijama i prema ključnoj riječi.

Već je objašnjeno na koji način će biti realizovana klasa Proizvod.

Metode za pretraživanje proizvoda odnose se na klasu Trgovina.

Ukoliko Korisnik odabere pretraživanje proizvoda po kategorijama, nakon što odabere kategoriju, svi proizvodi čiji atribut *kategorija* ima odabranu vrijednost, bit će prikazani korisniku. To će biti realizovano pomoću metode PretražiPoKategoriji.
Ukoliko korisnik bude pretraživao proizvode pomoću ključne riječi, onda će nakon unosa te ključne riječi, biti izlistani svi proizvodi, čija je jedna od ključnih riječi u kljucneRijeci unesena riječ (pretražiti list<string> kljucneRijeci da li sadrži unesenu ključnu riječ); metoda: bool PretražiKljučnuRiječ (string ključna) – te u zavisnosti od toga da li je unesena ključna riječ pronađena u ključneRijeci metoda će vratiti true/false. Korisniku se prikazuje svaki proizvod za koji ova metoda vrati true.

4. Kupovina artikala

Postojat će klasa Narudžba, te ukoliko nakon obavljene pretrage artikala, kupac odluči da naruči neki od njih, instancira se novi objekat tipa Narudžba :

Atributi:

Int IDNarudžbe,

Kupac Naručilac, (Veza između Kupac i Narudžba: Asocijacija -> Agregacija, koja označava da ukoliko se klasa koja posjeduje atribut druge klase obriše, druga klasa može nezavisno postojati -> ima smisla da postoji Kupac ako se izbriše Narudžba: taj Kupac može da bude naručilac neke druge narudžbe ili da postoji samostalno).

List<Proizvod>naruceniProizvodi, (veza Asocijacija između klasa Narudžba i Proizvod -> podvrsta: Agregacija, koja označava da ukoliko se klasa koja posjeduje atribut druge klase obriše, druga klasa može nezavisno postojati -> ako se izbriše Narudžba, Proizvod može samostalno postojati ili biti dodijeljen nekoj drugoj narudžbi).

Decimal cijenaNarudzbe,

VrstaPlaćanje vrstaPlacanja, (ako uvedemo novi tip objekta enum VrstaPlaćanja{Kartica, PrilikomDostave};

Datum datumNaručivanja,

Datum rokDostave,

String statusNarudžbe,

String Napomene,

Metode: get, set, PopuniNarudžbu, PotvrdiNarudžbu, IspostaviRačun, OtkažiNarudžbu, OdobriNarudžbu, OdbijNarudžbu, PošaljiNaDostavu, PošaljiUArhivu.

U zavisnosti od vrijednosti atributa vrstaPlacanja klase Narudžba poziva se jedna od metoda: NaplatiPrekoKartice ili NaplatiPrilikomDostave.

// uvedemo enum tip objekta enum VrstaPlaćanja{Kartica, PrilikomDostave};

Ovdje su ispunjeni sljedeći SOLID principi:

SRP (ova klasa ima samo jednu odgovornost vezanu za realizaciju narudžbe proizvoda),
OCP (otvorena je za eventualne nadogradnje, a zatvorena za modifikacije);

5. Plaćanje narudžbe

Ako je izabrani način plaćanje karticom, bit će pozvana metoda NaplatiPrekoKartice, koja će račun za narudžbu proslijediti vanjskom sistemu za autorizaciju kartica. S druge strane, ako bude odabrano plaćanje prilikom dostave, bit će pozvana metodaNaplatiPrilikomDostave, koja će izmijeniti atribut Napomena tako što će staviti napomenu da je potrebno izvršiti naplatu prilikom dostave naružbe.

6. Praćenje naručenog artikla

Zamišljeno je da kupac ima mogućnost da prati putanju dostave svoje naružbe. To jeste, kupac će moći pratiti kada je njegova narudžba stigla u sistem za dostavu, pri čemu će se status narudžbe mijenjati (Naručeno->UprocesuDostave->Dostavljeno).
// enum StatusNarudžbe {Naručeno, UprocesuDostave, Dostavljeno};
Ukoliko status narudžbe ne poprimi vrijednost Dostavljeno do trenutka rokDostave, kupac ima pravo na reklamaciju. U svakom slučaju, kupac nakon isteka rokDostave zaprima poruku, da uradi recenziju profila prodavača od kojeg je naručio proizvod. Isto tako, prodavač, nakon isteka tog istog roka zaprima poruku da obavi recenziju profila Kupca (poziva se metoda PromijeniRecenziju klase Korisnik).

Kada kupac zaprimi narudžbu, to jeste narudžba poprimi status Dosavljeno, ona odlazi u historiju kupovine profila kupca.

7. Pružanje informacija i pomoći klijentima

Ova funkcionalno treba samo da ostvari mogućnost klijentima da jasno imaju uvid u svoja prava (kao kupci ili prodavači).

8. Upravljanje korisničkim računom (Modifikacija računa)

Ova funkcionalnost će biti realizovana preko metoda klasa Kupac i Prodavač: „IzmijeniProfil“ (metode): PromijeniSifru, PromijeniNaziv, PromijeniEmail, PromijeniOpisProfila

9. Upravljanje prodajom

Prodavaču se nudi mogućnost da dodaje nove proizvode na svoj prodajni profil (DodajProizvod), da uklanja proizvod (met. IzbrisiProizvod), da mijenja cijene ponuđenih artikala(PromijeniCijenu), mijenja opise artikala(PromijeniOpisProizvoda), definira minimalnu dozvoljenu količinu zaliha artikla(DefinirajMinimalnuKolicinu), te da nudi posebne ponude proizvoda (akcije) (DodajAkcijuNaCijenu). Također, upravljanje prodajom podrazumijeva i statističko praćenje prodaje i potražnje za proizvodima (preko metoda klase Statistika : DajStatistiku, DajStatistikuMjeseca, DajstatistikuGodine i sl.. koje će na osnovu podataka o prodaji nekog prodajnog profila (na osnovu statistike za svaki proizvod na profilu) računati ovakve statističke podatke).

Klasa: Statistika

Atributi:

Int UkupnoProdanihProizvoda,

Int ProdanoOvajMjesec,

Int ProdanoOveGodine,

Metode: DajStatistikuMjeseca, DajStatistikuGodine, DajUkupnuStatistiku i sl..

Ovdje su ispunjeni sljedeći SOLID principi:
SRP (Klasa Statistika ima samo jednu odgovornost – zadužena je isključivo za statističke proračune prodaje nego prodajnog profila),

OCP (otvorena je za eventualne nadogradnje, ukoliko se ukaže potreba za nekim novim funkcionalnostima vezanim za statistiku, npr. Da se doda metoda koja preračunava dnevnu statistiku prodaje, koja istražuje kada je najveća potražnja za određenim proizvodima, i sl...), ali su osnovne metode i atributi dovoljni da bi pokrili osnovne funkcionalnosti, koje neće trebati modifikovati.