SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Gabriel Ukić

OPIS PROJEKTA MODEL PROIZVODNJE SVINJSKIH PROIZVODA

Varaždin, 2024.

UVOD

Objektno-orijentirano programiranje je metoda programiranja koja glavni naglasak i cilj stavlja na modeliranje i preslikavanje stvarnih entiteta, njihovih akcija i odnosa unutar računalnog programa(„Consultancy Services“). Da bi cilj bio ispunjen, objektno-orijentirano programiranje služi se objektnim pristupom gdje objekti sadržavaju informacije o entitetima, i njihovim radnjama, koje opisuju. Objektni se pristup koristi zbog bržeg razvoja novih programa koji koriste unaprijed izrađene klase(entitete) i mogućnosti sakrivanja podataka pri čemu se štite podaci. Kroz ovakav pristup dobiva se na mogućnosti razvijanja neovisnih dijelova za naknadnu integraciju koje može iznova koristiti pomoću nasljeđivanja.

Tema projekta u kojem je potrebno prikazati izradu modela pomoću tehnike objektno-orijentiranog programiranja je proizvodnja svinjskih proizvoda koje poduzeće prodaje obrtniku. Razlog odabira teme projekta leži u višegodišnjem iskustvu proizvodnje svinjskih suhomesnatih proizvoda te micanja od tipičnih i standardiziranih tema. Kroz iduća poglavlja prezentirati ću klase od kojih se model sastoji te veze između njih. Na kraju će biti prikazana programska implementacija kao i mogućnosti daljnje dorade i poboljšanja.

KLASE

Dakle, iako je tema svinjski proizvodi, središnja klasa preko koje se događa većina aktivnosti ovog modela je klasa Poduzeće koje je najbolje zamisliti kao nekakvu ili lokalnu mesnicu ili seoski OPG koji uzgaja svinje. Poduzeće vrši aktivnosti poput; prijava svinje u sustav, prijava radnika, narudžba hrane za svinje te poslovanje s drugim poslovnim subjektima. Svinje kao klasa sadržavaju, za procese bitne atribute, kao što su: ID koji predstavlja brojčanu oznaku svinje(u stvarnosti se radi o oznaci na naušnici koja je potrebna kasnije za testiranje valjanosti mesa), vrsta svinje, kilaža i ciljana kilaža(težina na koju planiramo dovesti svinju) te namjena svinje gdje određujemo za što svinju uzgajamo.

Uzimajući u obzir da svinje moramo negdje smjestiti, izrađena je klasa Svinjar koja sadržava formalne atribute poput datuma izgradnje i adrese, ali i zakonski potrebnih atributa kao što su: datum čišćenja svinjara, veličina(prostorne veličina) i kapaciteta. Nadalje, nakon što obavimo egzekuciju svinje, bacamo se na posao odvajanja njenih dijelova te na sam proces proizvodnje pa s toga izrađene su klase Proizvod i Skladište. Proizvod predstavlja proizvode koje dobijemo nakon komadanja mesa gdje odvajamo meso na različite dijelove pa se klasa sastoji od atributa vrste proizvoda i datum proizvodnje. Klasa Skladište sastoji se od tri kapaciteta: hladnjača(u kontekstu samih proizvoda ne bitno, hladnjače sadržavaju zamrznuto meso za pečenje ili odojke), frižidera te ostalog kapaciteta koji se odnosi na proizvode kojima nije potrebno hladnije okruženje ili za privremeno skladištenje hrane za

svinje. Unutar veze između ove dvije klase(koje će naknadno biti objašnjeno) nalazi se klasa Sušara s idejom da prima proizvode koji zahtijevaju sušenje, uglavnom se radi o par excellance proizvodima – pršut, panceta, pečenica.

Unutar poslovnijeg dijela modela, poduzeće da bi postojalo treba nekoga na čelu, tko će ulagati u poslovanje, treba vlasnika pa time imamo i klasu Vlasnik kojoj ovdje ne pridajemo proporcionalni značaj kakav je u pravilu te time ta klasa ima samo jedan unikatni atribut – datum vlasništva. Budući da vlasnik ulaže sredstva, potrebni su nam radnici koji će izvršavati različite poslove unutar procesa proizvodnje. Klasa Radnik sastoji se potrebnih atributa kao što su: datum zaposlenja, radno mjesto (vrsta radnog mjesta), satnice i broja odrađenih sati. Kako veličina Poduzeća može varirati, kreirana je i klasa Radnici koja nije prikazana u modelu, a koja je s klasom Radnik povezana mehanizmom prijateljstva. Nastavno na poslovanje, definirana je klasa Dobavljač koja predstavlja entitet od kojeg naručujemo hranu za naše svinje, a sastoji se od atributa vrsta hrane i količine hrane koju naručujemo. Zadnja klasa iz ovog područja je klasa Obrtnik koja se sastoji od atributa: naziv, vrsta i adrese obrta.

Kako nebi unutar svake klase posebno definirali opće atribute, kreirana je klasa Osoba koja se sastoji od temeljnih podataka: ime, prezime, OIB, datum rođenja, spolna orijentacija i email. Postoje dvije klase koje su definirane kao tip podataka: Adresa i Datum. Zatim, model se sastoji od funkcije Narudžba koja je referenca za najvažniju aktivnost Poduzeća – prodaju proizvoda. Na kraju, imamo enumeracije koje sadrže zapise o vrsti vrijednosti određenih komponenata klasa, pa imamo:

- Namjena svinje: mesnica(svinja se neće rasjeći detaljno, namijenjena za prodaju dijelova za pečenje), odojak, proizvodnja, uzgoj(radi se o svinjama koje se koriste za rasplodnjavanje) i prodaju(svinje koje se prodaju za daljnji uzgoj)
- Radno mjesto: hranitelj, mesar, sušar(radnik u sušari), skladištar i distributer(dostavlja proizvode)
- Spol: muško, žensko, ostalo, neizjašnjeno
- Vrsta hrane: kukuruz, žito, mekinje, pšenica
- Vrsta poduzeća: j.d.o.o, d.o.o, d.d, opg, obrt
- Vrsta obrta: mesnica, trgovina, restoran, food franšiza
- Vrsta proizvoda: panceta, pršut, plećka, čvarci, sušeno meso, kobasice, pečenica, lungić, jetrica, svinjski organi, vratina

VEZE IZMEĐU KLASA, UML DIJAGRAM I KOD

Objektni pristup podrazumijeva da klase budu međusobno povezane pripadajućim vezama: veza nasljeđivanja, veza ovisnosti, asocijacijska veza i njene varijante, agregacijska i kompozicijska veza. Vezom nasljeđivanja, temeljnom vezom modela, povezali smo, kao što je gore navedeno, klasu Osoba s klasama: Radnik, Obrtnik, Vlasnik i Dobavljač gdje klasa Osoba nadklasa navedenima te iz tog razloga unutar njih su definirani samo specifični atributi. Veza ovisnosti je najčešće korištena veza u modelu jer su pomoću nje povezane enumeracije i tipovi podataka s klasama kako bi ih „rasteretili“. Stoga imamo iduće veze:

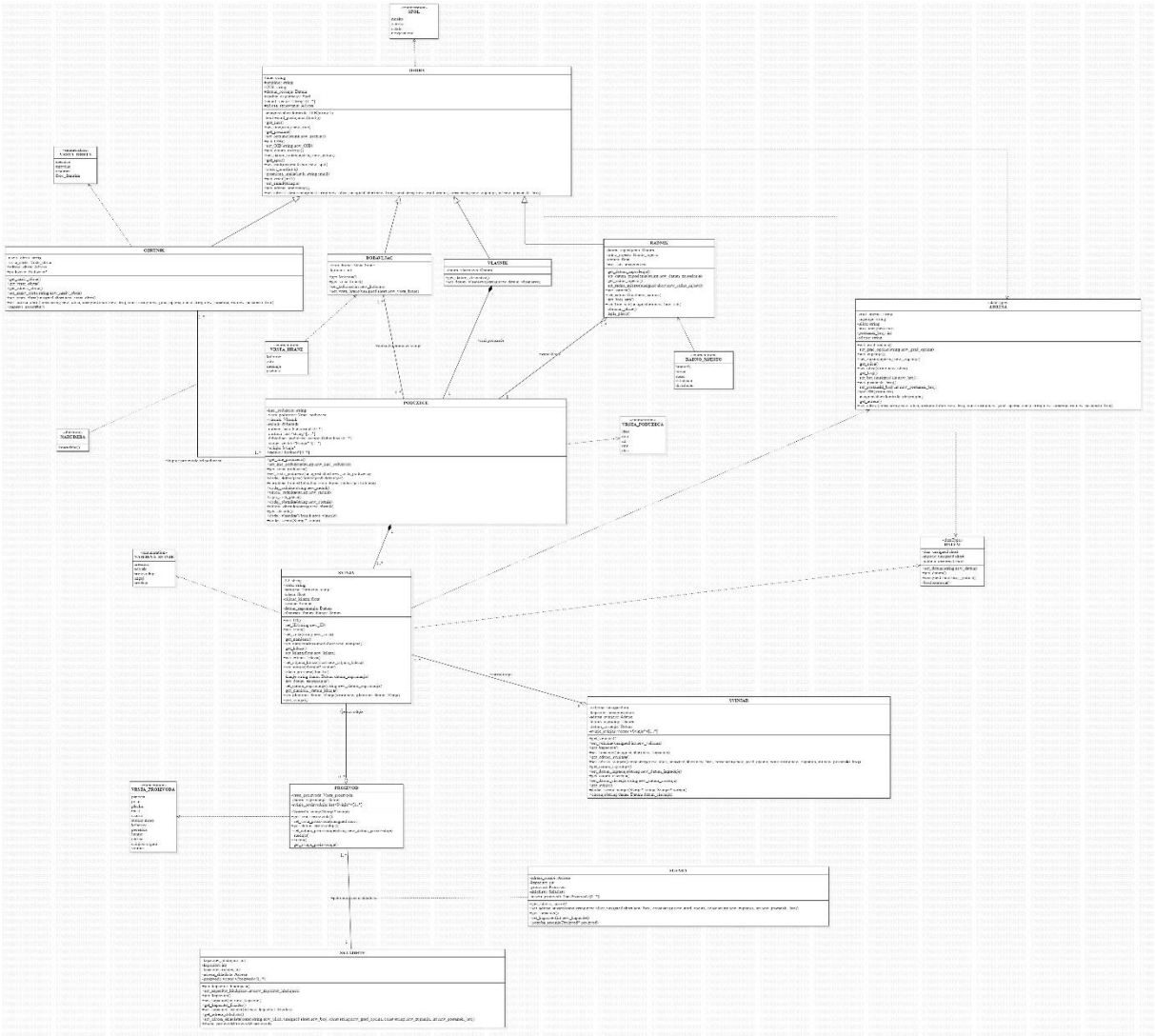
- Osoba s Adresom, Datumom i Spolom
- Svinja s Adresom, Datumom, Namjenom svinje
- Obrtnik s Vrstom obrta
- Radnik s Radnim mjestom
- Proizvod s Vrstom proizvoda
- Dobavljač s Vrstom hrane
- Poduzeće s Vrstom poduzeća

Asocijacijska veza je odnos gdje su stvarni entiteti neovisni, ali su međusobno povezani (Khan,2023). Na konkretnom primjeru ovog modela, radi se o vezi između Poduzeća i Obrtnika gdje Obrtnik kupuje/naručuje proizvode od Poduzeća(definirana je posebna funkcija za ovu radnju), zatim između Poduzeća i Dobavljača gdje se radi o sličnoj aktivnosti kao i u prethodnoj vezi te veza između Proizvoda i Skladišta gdje se proizvodi pohranjuju u skladište, ali i gdje imamo klasu veze Sušara.

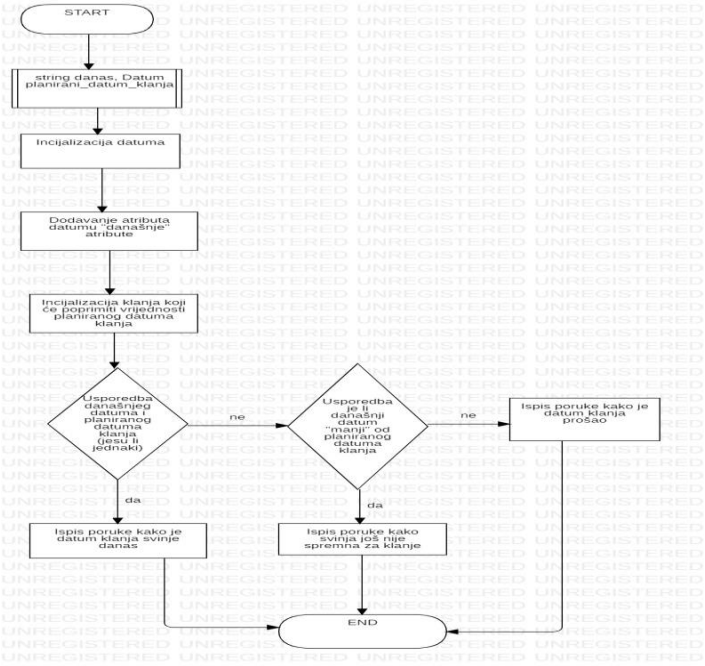
Agregacijska veza je vrsta asocijacijske veze gdje su objekti jedne klase u „vlasništvu“ druge klase, ali mogu postojati izvan nje (Khan, 2023.). Ovdje se to radi o vezama između Svinjara i Svinje gdje svinja egzistira u svinjaru, ali ona izvan samog svinjara može postojati, štoviše ukoliko je njena namjena npr. odojak, može se dogoditi da svinja s tom namjenom nikad ne uđe u svinjar(ukoliko je naručena). Iduća instanca ove veze je Svinja i Proizvod, tri različite namjene svinje označavaju da svinja neće ući u proces proizvodnje proizvoda, ali i proizvod može postojati neovisno o svinji unutar poduzeća ukoliko se dogodi situacija da za izradu kobasica nedostaje mesa ili sala pa se mora naknadno uzimati od nekog drugog. Zadnja takva veza je između Poduzeća i Radnika, sam radnik može postojati izvan poduzeća, odnosno može raditi za drugo poduzeće koje bi samo trebalo promijeniti datum zaposlenja.

Kompozicijska veza je u pravilu jača veza od agregacijske veze jer u slučaju brisanja skupnog objekta brišu se i svi ostali s njim povezani(kompozicijskom vezom)(Khan, 2023.).

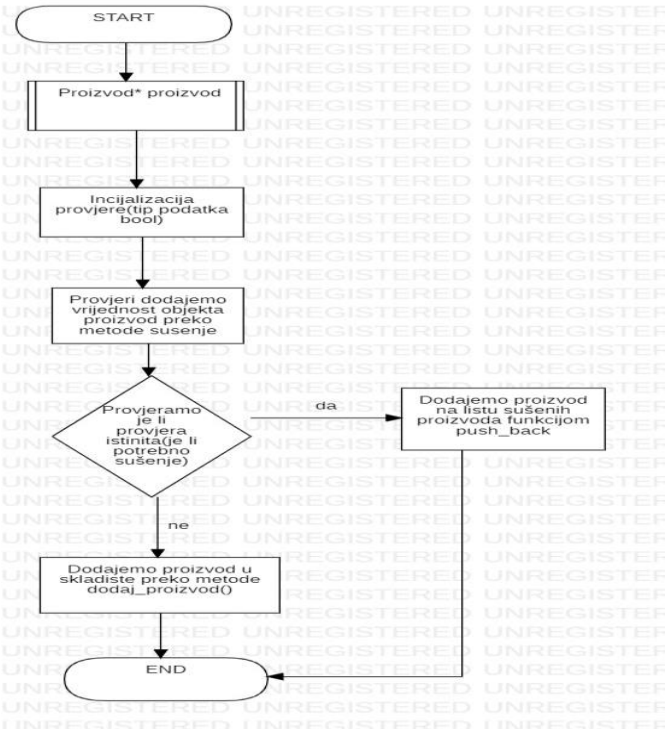
Ovdje pronalazimo dvije kompozicijske veze. Poduzeće i Svinja, svinje u ovom procesu ne mogu postojati izvan poduzeća, samo postojanje svinje van zakonski reguliranog okruženja implicira da se radi eventualno o divljoj svinji te veza između Poduzeća i Vlasnika gdje poduzeće bez vlasnika ne može postojati. Nakon definiranja klasa, njihovih atributa te veze između njih, izrađujemo UML dijagram gdje će se sve gore navedeno i opisano prikazati.



Kao što se vidi i na slici, za sve klase su definirani i neki dodatni atributi poput vektora i lista te određeni objekti klasa s pokazivačima. Ono što gore nije navedeno, definirane su i metode od kojih se klase sastoje. Velika većina tih metoda su standardne get i set metode za atribute kao npr. get_ime() i set_ime(string new_ime) unutar osobe, ali postoje i metode koje su izrađene posebno za ovaj model. Uzmimo za primjer metodu klanje koja će nam provjeriti u kakvom su vremenskom odnosu trenutni(tekući) datum i planirani datum klanja svinje:



Dakle, metoda uspoređuje datume te na temelju njih ispisuje poruku je li datum svinje prošao. Jedna od složenijih metoda je metoda kontrole potrebe sušenja unutar klase Sušara. Ideja metode je da nakon što primi proizvod, inicijalizira bool tip podatka gdje pridružujemo vrijednost objekta te provjeravamo trebamo li sušiti proizvod. Ako proizvod treba sušiti dodajemo ga na listu sušenih proizvoda(bilo push_back() ili push_front() metodama), a ako nije, dodajemo ga u skladište metodom dodaj proizvod koja ga stavlja na listu proizvoda u skladištu:



Ono što se također da primijetiti je i da su unutar dijagrama veza definirani kardinaliteti i obvezatnosti klasa unutar veza između istih.

Programski dio izrađen je u Microsoft Visual Studiu 2019, gdje su korištene biblioteke: <iostream>, <string>, <cstring>, <ctime>, <cstdlib> te biblioteke za rad s listama i vektorima (<list> i <vector>). Svaka klasa je isprogramirana tako da sadrži header datoteku i cpp datoteku. Header datoteka sadrži sve atribute i metode određene klase gdje je naznačena njihova vidljivost, dok u cpp datoteci su raspisane sve metode klasa. Main dio programa sadrži izbornik gdje korisnik odabire funkcionalnosti programa, a u nastavku su inicijalizirani poneki objekti kako bi program imao njih što više na raspolaganju.

```
int izbornik() {
    cout << "----------------------IZBORNIK----------------------" << endl;
    cout << "1- narudzba iz perspektive obrtnika" << endl;
    cout << "2- unos novog radnika i ispis svih placa" << endl;
    cout << "3- dodavanje svinje u svinjar" << endl;
    cout << "4- dodavanje svinje u poduzece" << endl;
    cout << "5- unos svinje i provjera namjene svinje za procesurianje" << endl;
    cout << "6- unos proizvoda i potreba susenja" << endl;
    cout << "7- unos dobavljaca i naručivanje hrane" << endl;
    cout << "9- izlaz iz programa" << endl;
    cout << "Vas unos: " << endl;
    int izbor;
    cin >> izbor;
    return izbor;
}
```

Ideja je da se odabirom narudžbe da mogućnost korisniku da unese narudžbu(dalje razrađeno), zatim da mu se pruži mogućnost unosa radnika i uvida u ispisivanje plaća svih zaposlenika. Iduće tri mogućnosti se tiču svinje, a to su: dodavanje svinja u svinjar i njegov unos, prijava svinje u poduzeće i provjera datuma klanja te unos i provjera namjene svinje za daljnje procesuiranje. Na kraju, dva odabira se odnose na naručivanje hrane od dobavljača i provjeru potrebe sušenja. Unutar koda nalazi se klasa Radnici koja je prijateljska klasa klasi Radnik koja se sastoji od idućih atributa i metoda:

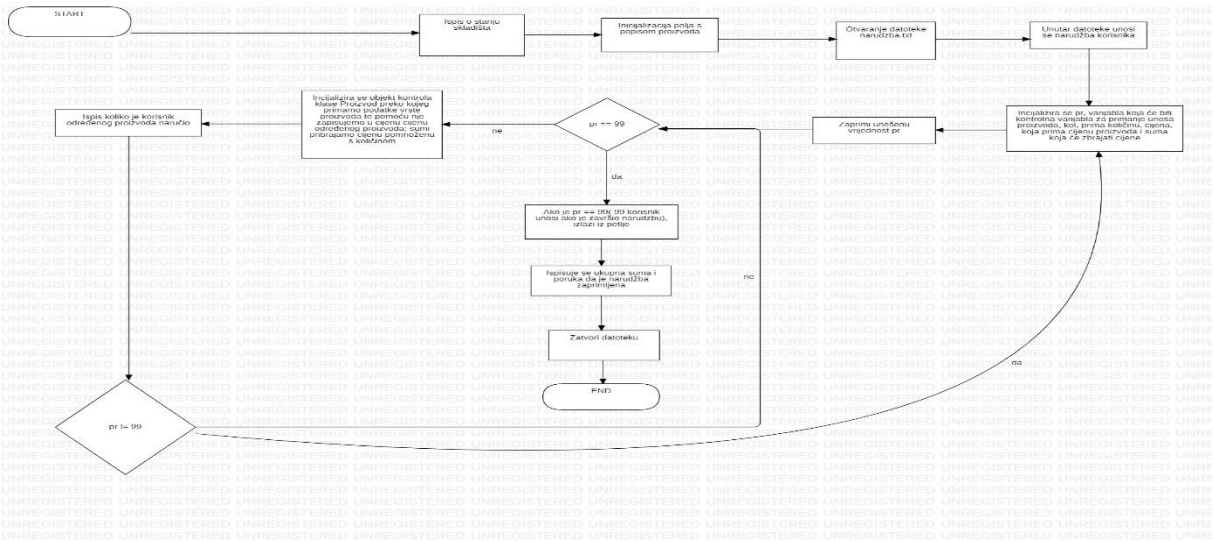
```
#include <list>
#include <fstream>
#include "Radnik.h"

using namespace std;

class Radnik;
class Radnici {
private:
    list <Radnik> svi_radnici;
public:
    Radnik* get_radnik(unsigned int i);
    void set_radnik(Radnik new_radnik);
    void import_radnici(const char* popis_radnika);
    void export_radnici(const char* popis_radnika);
    list <Radnik>& lista_radnika();
};
```

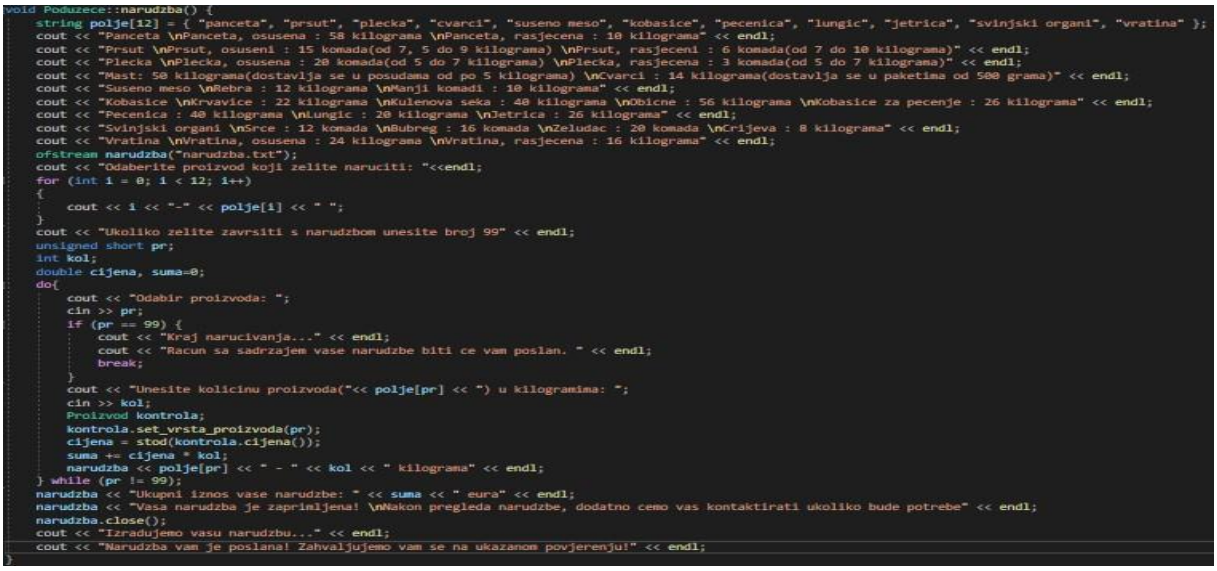
Program je dinamičan i opširan, s obzirom na veze između klasa bilo je potrebno implementirati više vrsta lista, vektora te samim tim i pokazivača te preko njih izrađivati metode. Najviše metoda sadržava najdetaljnije izrađena klasa Poduzeće.

METODA NARUDZBA

Metoda narudžba je iz perspektive poslovanja, najbitnija metoda čitavog modela. Dok je unutar UML dijagrama ona definirana kao zasebna funkcija unutar veze Poduzeća i Obrtnika, u programskom dijelu ona je definirana unutar Poduzeća, dok se u klasi Obrtnik nalazi metoda napravi_narudzbu() gdje preko pokazivača na Poduzeće pristupamo metodi.

Kako onda funkcionira metoda? Najprije, deklariramo polje koje sadržava 11 elemenata odnosno 11 vrsta proizvoda te se nakon toga ispisuje poruka u kojoj se nalazi trenutno stanje skladišta. Neki proizvodi su jednostavno navedeni kao npr. lungić, dok neki su razdijeljeni kao pršut gdje odvajamo osušene pršute i one svježe rasječene. Nakon toga otvara se tekstualna datoteke „narudzba“, kroz for petlju ispisujemo brojčani meni proizvoda pa korisnik unosi numerički što želi naručiti i u kojoj količini. Cijelo vrijeme naručivanja korisnika provodi se u petlji koja traje sve dok varijabla za odabir proizvoda nije jednaka broju 99 koja označava kraj procesa naručivanja, a čitavo vrijeme dohvaćamo cijenu proizvoda te je konstantno pribrajamo zbroju svih cijena. Kada korisnik završi s narudžbom, zatvara se upisivanje te se ispisuje poruka kako se izrađuje narudžba. Korisnik po završetku metode, dobiva u vidu tekstualne datoteke svoju narudžbu s ukupnim iznosom te obavijesti kako će biti naknadno kontaktiran vezano za detalje (recimo ako je naručeno sušeno meso pa korisnik kasnije povratno obavještava poduzeće želi li rebra ili manje komade). Dijagram toka aktivnosti i programski kod metode:





```
void PoDuzece::narudzba() {
    string polje[12] = { "panceta", "prsut", "plecka", "cvarci", "suseno meso", "kobasice", "pecenica", "lungic", "jetrica", "svinjski organi", "vratina" };
    cout << "Panceta \nPanceta, osusena : 58 kilograma \nPanceta, rasjecena : 10 kilograma" << endl;
    cout << "Prsut \nPrsut, osusen1 : 15 komada(od 7, 5 do 9 kilograma) \nPrsut, rasjecen1 : 6 komada(od 7 do 10 kilograma)" << endl;
    cout << "Plecka \nPlecka, osusena : 20 komada(od 5 do 7 kilograma) \nPlecka, rasjecena : 3 komada(od 5 do 7 kilograma)" << endl;
    cout << "Mast: 50 kilograma(dostavlja se u posudama od po 5 kilograma) \nCvarci : 14 kilograma(dostavlja se u paketima od 500 grama)" << endl;
    cout << "Suseno meso \nRebra : 12 kilograma \nManji komadi : 10 kilograma" << endl;
    cout << "Kobasice \nKrvavice : 22 kilograma \nKulenova seka : 40 kilograma \nObicne : 56 kilograma \nKobasice za pecenje : 26 kilograma" << endl;
    cout << "Pecenica : 40 kilograma \nLungic : 20 kilograma \nJetrica : 26 kilograma" << endl;
    cout << "Svinjski organi \nSrce : 12 komada \nBubreg : 16 komada \nZeludac : 20 komada \nCrijeva : 8 kilograma" << endl;
    cout << "Vratina \nVratina, osusena : 24 kilograma \nVratina, rasjecena : 16 kilograma" << endl;
    ofstream narudzba("narudzba.txt");
    cout << "Odaberite proizvod koji zelite narucit1: "<<endl;
    for (int i = 0; i < 12; i++)
    {
        cout << i << "-" << polje[i] << " ";
    }
    cout << "Ukoliko zelite zavrsiti s narudzbom unesite broj 99" << endl;
    unsigned short pr;
    int kol;
    double cijena, suma=0;
    do{
        cout << "Odabir proizvoda: ";
        cin >> pr;
        if (pr == 99) {
            cout << "Kraj narucivanja..." << endl;
            cout << "Racun sa sadrzajem vase narudzbe biti ce vam poslan. " << endl;
            break;
        }
        cout << "Unesite kolicinu proizvoda("<< polje[pr] << ") u kilogramima: ";
        cin >> kol;
        Proizvod kontrola;
        kontrola.set_vrsta_proizvoda(pr);
        cijena = stod(kontrola.cijena());
        suma += cijena * kol;
        narudzba << polje[pr] << " - " << kol << " kilograma" << endl;
    } while (pr != 99);
    narudzba << "Ukupni iznos vase narudzbe: " << suma << " eura" << endl;
    narudzba << "Vasa narudzba je zaprimljena! \nNakon pregleda narudzbe, dodatno cemo vas kontaktirati ukoliko bude potrebe" << endl;
    narudzba.close();
    cout << "Izradujemo vasu narudzbu..." << endl;
    cout << "Narudzba vam je poslana! Zahvaljujemo vam se na ukazanom povjerenju!" << endl;
}
```

POBOLJŠANJA I ZAKLJUČAK

Programski dio je mogao biti napravljen bez uporabe naredbe „*using namespace std*“ kako bi kod bio malo čitljiviji. Moguće poboljšanje unutar programa su dodavanje prijateljskih klasa Svinje, Obrtnici i Dobavljači. Također, izrada datoteke koja bi ispisivala stanje skladišta umjesto ispisivanja stanja preko konzole.

Sve bitne funkcionalnosti programa obuhvaćene su u izborniku te rade optimalno. Idealno bi bilo kada bi klasa Obrtnik bila razrađena kao klasa Kupac, koja bi bila obuhvatnija i proširenija kako bi u model uzimala u obzir ako individualac želi kupiti proizvod od poduzeća. Smatram da bi ovako izrađen model bio dobar temelj za izradu aplikacije koja bi olakšala poduzeću (ili barem vlasniku) kontrolu proizvodnje, planiranje, distribuiranje proizvoda kao i vođenje evidencije oko svinja i proizvoda, a i lakši kontakt i dobivanje povratnih informacija od strane obrtnika. Izradom ovog modela, kako programskog koda, tako i UML dijela projekta, pokušao sam prikazati znanje koje sam stekao kroz pohađanje ovog kolegija.

LITERATURA

1.Consultancy Services *Što je objektno-orijentirano programiranje (OOP) ?* Preuzeto 10.06.2024 s https://og-cs.hr/sto-je-objektno-orijentirano-programiranje-oop/

2.Lovrenčić, A. i Konecki, M. (2017). Programiranje u 14 lekcija, Tekstualne datoteke (str.333-357). Varaždin: Fakultet organizacije i informatike

3.Muhammad Humza Khan (2023.) *Understanding Object-Oriented Relationships: Inheritance, Association, Composition, and Aggregation* , Preuzeto 10.06.2024. s https://medium.com/@humzakhalid94/understanding-object-oriented-relationships-inheritance-association-composition-and-aggregation-4d298494ac1c

Za izradu projekta korišteni su materijali s predavanja i laboratorijskih vježbi kolegija „Objektno-orijentirano programiranje“