Restoran

Opis aplikacije

- **Restoran** sadrži određen broj **stolova**, **jelovnik** i **generator** koji generiše dnevno jelo u datom satu.
- Svaki **sto** ima određenog **konobara** koji opslužuje **mušteriju** koja sedne za tim stolom. Takođe svaki sto ima svoj ID, kao i račun koji je mušterija napravila.
- **Konobar** može raditi na više stolova. Funkcije koje konobar obavlja su:
    - Prikazivanje jelovnika i dnevnog menija,
    - Uzimanje porudžbine i
    - Računanje i naplata računa.
- **Mušteriji** je poznato ime i prezime, kao i koliko novca imaju. Kada uđu u restoran odmah dobijaju sto (ako ima trenutno slobodan). Mogu naručivati hranu, platiti i osloboditi sto nakon što završe.
- **Jelovnik** se sastoji iz **jela**, različitih kategorija. Jelo se sastoji iz **sastojaka**. Cena jela se računa kao cena svih sastojaka + prodajne cene + PDV.
- **Dnevni meni** se određuje “random“ za odredjen sat. Bira se po jedno predjelo, glavno jelo i dezert.

Opis klasa

Sastojak

Sastojak ima *naziv, cenu* i *kategoriju*. Pravi se zadavanjem svih tri parametara. Ima gettere za naziv, cenu i kategoriju, kao i seter za cenu.

Jelo

Jelo ima *naziv, vector<Sastojak *> sastojci, cenu, PDV,* i *kategorijuJela, podKategoriju.* Pravi se zadavanjem *naziva, kategorije* i opciono *podKategorije.* Ima gettere za *cenu, naziv i kategorijuJela*. Sadrži funckije: *void dodajSastojak(Sastojak *),* za dodavanje sastojaka, baca ***JeloException*** u slučaju dodavanja sastojka koji već postoji, *void izbaciSastojak(const string &),* za izbacivanje sastojaka, baca ***JeloException*** u slučaju da nije našlo jelo sa datim nazivom, *void postaviCenu(const double &),* koja postavlja cenu jela. Takođe ima i prepisan operator ispisa.

Jelovnik

Jelovnik se sastoji iz niza **Jela**, *vector<Jelo *> jelovnik.* Pravi se bez argumenata, *Jelovnik().* Ima funckije: *void dodajJelo(Jelo *),* koja dodaje jelo u jelovnik, baca ***JeloException*** u slučaju dodavanja jela koje već postoji, *Jelo * getJelo(const string &) const*, koja nalazi jelo preko zadatog imena i vraća to jelo ako postoji u suprotnom baca ***JeloException***, *void izbaciJelo(const string &),* izbacuje jelo iz jelovnika ako postoji u *vector-u* suprotnom baca ***JeloException*** , *vector<Jelo *> getJeloPoKat(KATEGORIJA_JELA) const,* koja vraća niz jela zadate kategorije,

void prikaziJelovnik() const, izlistava ceo jelovnik po kategorijama, *double getCenuJela(const string &) const,* vraća cenu određenog jela.

Osoba

Osoba ima *ime* i *prezime*. Pravi se zadavanjem oba parametra. Konstruktor je definisan kao *protected*, čime se dobija efekat čiste virtualne funckije. Sadrži *protected* funkciju *virtual const string toString() const.*

Mušterija

Mušterija je Osoba. Sadrži *novac, idStola i Konobara*. Pravi se zadavanjem *imena, prezimena, i novca*. Može da rezerviše sto *void rezervisiSto(Restoran *)* i da oslobodi sto ako je račun plaćen, inače baca ***OsobaException*** *void oslobodiSto(),* kao i da se ispišu informacije o mušteriji na standardan izlaz *cout<<Musterija.* Od konobara može da zatraži da vidi ceo jelovnik *void prikaziJelovnik(),* kao i da vidi samo dnevni meni *void zatraziDnevniMeni(const int &).* Takođe može da naruči jelo *void naruciJelo(const string &),* i da plati račun koji je napravio *void PlatiRacun(),* potencijalno baca ***OsobaException****.* Transakciju novca vrši konobar pozivanjem prijateljske funkcije: *friend void Konobar::naplatiRacun(Musterija *, const double &racun)*.

Konobar

Konobar je Osoba, koji ima *zarađenNovac, Restoran *restoran,* i *vector<Sto *> stolovi.* Pravi se zadavanjem *imena, prezimena* i *Restorana* u kome radi. Ima mogucnost dodavanja stolova na kojima radi *void dodajSto(Sto *)* i oslobođavanja stola kada mušterije ustanu *void oslobodiSto(const int &),* kao i da se ispišu informacije o konobaru na standardan izlaz *cout<<Konobar.* Funkcije koje konobar vrši su:

- Uzimanje narudžibe – *void uzmiNarudzbinu(const string &, const int &)*
- Prikazivanje menija mušteriji – *void prikaziMeni()*
- Prikazivanje dnevnog menija – *void prikaziDnevniMeni(const int &)*
- Izdavanje računa – *double izdajRacun(const int &) const*
- Naplata računa – *void naplatiRačun(Musterija *, const double &)*

Sto

Sto ima *bool zauzet, id, racun,* i poznatog konobara koji radu na njemu *Konobar *konobar.* Pravi se zadavanjem konobara. Ima gettere za *racun – double getRacun() const* i za id – *int getID() const* , funkcije *bool isZauzeto() const* i *void setZauzeto(const bool).* Može da uveća račun pri svakoj narudžbini – *void uvecajRacun(const double &)* i da vrati račun na početno stanje nakon što mušterija za tim stolom plati – *void resetujRacun().*

Generator

Generator ima poznat *jelovnik,* dnevni meni – *meni* i niz *cena.* Pravi se zadavanjem jelovnika. Ima getter za *cenu – const double getCena(const int &) const,* kao I funkciju *void dajDnevnoJeloUSatu(const int &sat),* koja ispisuje dnevni meni, generisan pomocu privatne funkcije *void generate(),* koja baca ***RestoranException*** u slučaju da nema bar jedno predjelo, glavno jelo i dezert.

Restoran

Restoran ima *Jelovnik *jelovnik, kasa, vector<Sto *> stolovi, Generator *generator.* Pravi se zadavanjem jelovnika. Može da doda sto u *vector – void dodajSto(Sto *)*, dodeliti prvi slobodan sto mušteriji – *Sto * getSto() const,* baca ***RestoranException*** u slučaju da nema slobodnih stolova*,* prikazati jelovnik – *void prikaziJelovnik() const,* prikazati dnevni meni – *void prikaziDnevniMeni(const int &)* i može da spremi zadato jelo – *double spremiJelo(const string &) const,* koje vraća cenu jela*.* Na kraju se može ispisati stanje kase *void prikažiKasu() const.*

JeloException, OsobaException, RestoranException

Ove 3 klase nasleđuju ugrađenu klasu exception. Sadrže *string msg*. Prave se zadavanjem određene poruke koju ispisuju na standardni izlaz pozivanjem nasleđene funkcije *virtual const char * what() const throw()*.