Paterni ponašanja

Strategy patern (implementiran)

Strategy patern je u našem sistemu iskorišten u okviru funkcionalnosti filtriranja knjiga. Naime kako je to prikazano i na korisničkim formama, korisnik kada otvori formu za pretragu dostupnih knjiga može vršiti filtriranje knjiga po tipu i vrsti, gdje su tipovi romani i naučni radovi, a na osnovu tipa vrši se filtriranje dalje po vrstama. U svrhu omogućavanja ova dva tipa filtriranja iskorišten je upravo Strategy patern.

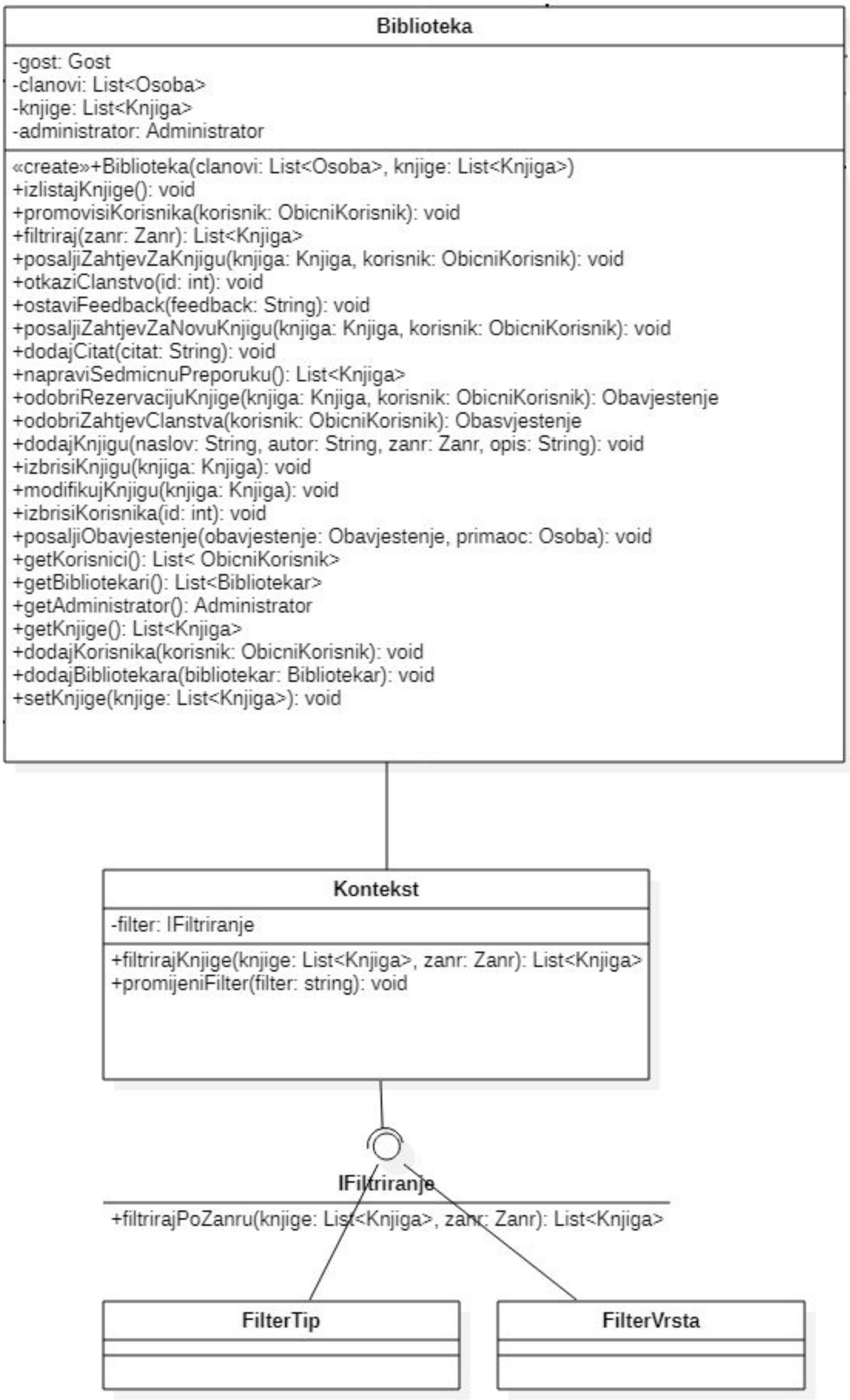
Izmjene na dijagramu klasa

Dodan je interfejs IFiltriranje sa metodom filtrirajPoZanru(knjige: List<Knjiga>, zanr: Zanr): List<Knjiga>. Zatim je dodana klasa Kontekst koja ima atribut tipa IFiltriranje i dvije metode: filtrirajKnjige(in knjige:List<Knjiga>, in zanr:Zanr): List<Knjiga>, promijeniFilter(in filter:string): void. Dvije klase koje realiziraju interfejs su FilterTip i FilterVrsta. Ideja je ta da se u zavisnosti od toga da li treba knjige filtrirati po vrsti ili tipu koristi jedna od ove dvije klasa, naravno unutar klase Kontekst koja pomoću metode promijeniFilter mijenja svoj atribut filter odnosno filter je onog tipa koji je potreban da se izvrši željeno filtriranje.

Svrha korištenja strategy paterna

Svrha korištenja strategy paterna jeste u tome da se različiti algoritmi koji obavljaju sličnu funkcionalnost izdvoje u zasebne klase i da se time korisniku/klijentu omogući da bira koji algoritam želi da se primijeni za obavljanje određenog zadatka. Upravo smo to postigli i u našem sistemu izdvojivši dva načina filtriranja u zasebne klase koje koje su neovisne od klijenta koji ih koristi.

Izmjene na dijagramu klase su prikazane na narednoj slici.


Biblioteka
-gost: Gost
-clanovi: List<Osoba>
-knjige: List<Knjiga>
-administrator: Administrator
«create»+Biblioteka(clanovi: List<Osoba>, knjige: List<Knjiga>)
+izlistajKnjige(): void
+promovisiKorisnika(korisnik: ObicniKorisnik): void
+filtriraj(zanr: Zanr): List<Knjiga>
+posaljiZahtjevZaKnjigu(knjiga: Knjiga, korisnik: ObicniKorisnik): void
+otkaziClanstvo(id: int): void
+ostaviFeedback(feedback: String): void
+posaljiZahtjevZaNovuKnjigu(knjiga: Knjiga, korisnik: ObicniKorisnik): void
+dodajCitat(citat: String): void
+napraviSedmicnuPreporuku(): List<Knjiga>
+odobriRezervacijuKnjige(knjiga: Knjiga, korisnik: ObicniKorisnik): Obavjestenje
+odobriZahtjevClanstva(korisnik: ObicniKorisnik): Obasvjestenje
+dodajKnjigu(naslov: String, autor: String, zanr: Zanr, opis: String): void
+izbrisiKnjigu(knjiga: Knjiga): void
+modifikujKnjigu(knjiga: Knjiga): void
+izbrisiKorisnika(id: int): void
+posaljiObavjestenje(obavjestenje: Obavjestenje, primaoc: Osoba): void
+getKorisnici(): List< ObicniKorisnik>
+getBibliotekari(): List<Bibliotekar>
+getAdministrator(): Administrator
+getKnjige(): List<Knjiga>
+dodajKorisnika(korisnik: ObicniKorisnik): void
+dodajBibliotekara(bibliotekar: Bibliotekar): void
+setKnjige(knjige: List<Knjiga>): void
Kontekst
-filter: IFiltriranje
+filtrirajKnjige(knjige: List<Knjiga>, zanr: Zanr): List<Knjiga>
+promijeniFilter(filter: string): void
IFiltriranje
+filtrirajPoZanru(knjige: List<Knjiga>, zanr: Zanr): List<Knjiga>
FilterTip
FilterVrsta

Observer patern (implementiran)

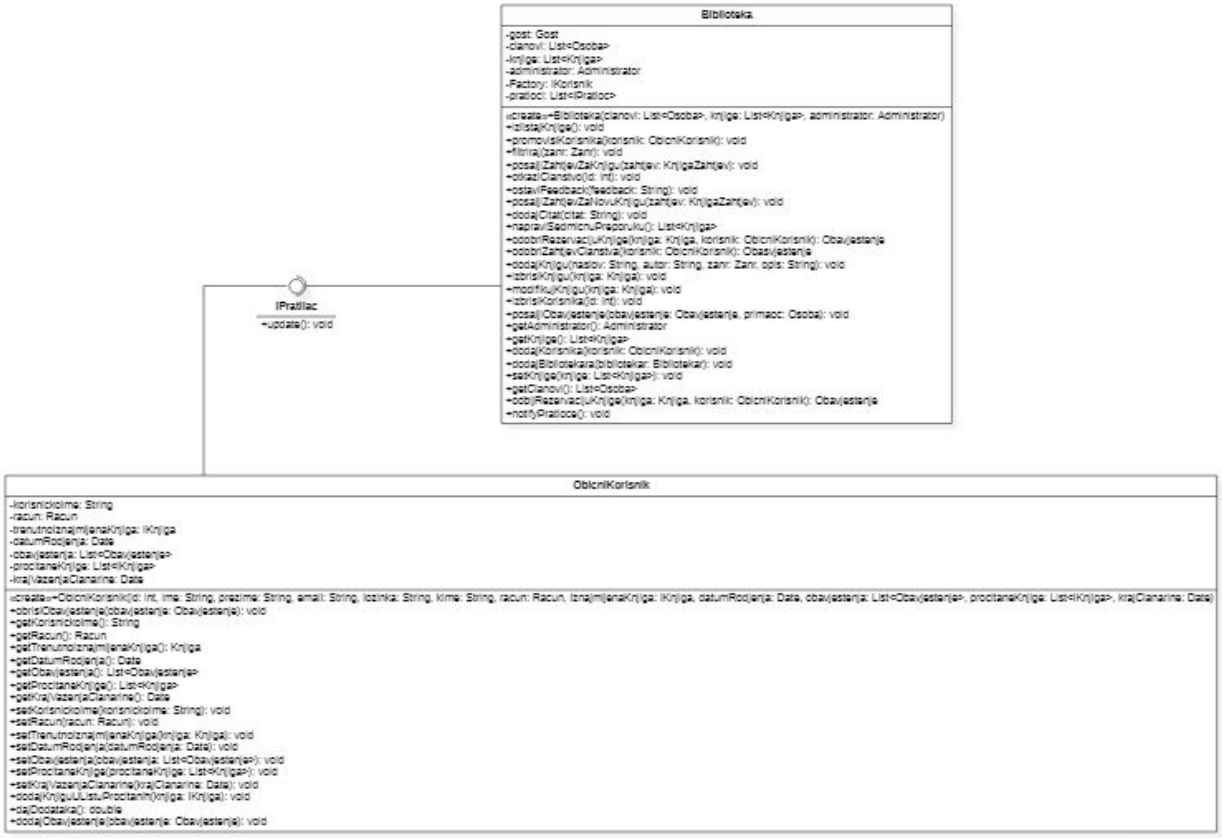
U nasem sistemu observer patern je iskoristen u slucaju kada imamo situaciju da je vise korisnika zainteresovano za istu knjigu, te korisnici koji nisu u mogucnosti rezervisati tu knjigu trenutno mogu se prijaviti da dobiju obavjestenje kada ta knjiga ponovo postane dostupna za rezervaciju.

Izmjene na dijagramu klasa

Definisemo interfejs IPratilac koji ce imati metodu update() kojeg ce implementirati klasa ObicniKorisnik. U klasu Biblioteka dodajemo List<IPratilac> u kojoj ce se nalaziti svi korisnici koji zele da dobiju obavjestenje kad knjiga ponovo postane slobodna i dodajemo metodu notifyPratioce() koja im salje tu informaciju.

Svrha korištenja observer paterna

Svrha koristenja observer paterna je da obavjestenje o promjeni stanja posmatranog objekta posaljemo svim objektima koji ga prate(observaju). Upravo to smo postigli u nasem sistemu tako sto sve korisnike u isto vrijeme obavijestimo o dostupnosti knjige.



Iterator patern

Iterator patern bismo mogli iskoristiti u nasem sistemu pri kretanju kroz liste u klasi Biblioteka, ali posto vec postoji foreach petlja primjenjivanje ovog paterna bi bila suvisna.

Ovaj patern bismo mogli iskoristiti ako umjesto listi kao atribut imamo mapu korisnika i njihovih iznajmljenih knjiga, te ako bismo na primjer željeli naci korisnike koje nisu iznajmili knjigu u tom trenutku.
Definisemo interfejs IEnumerable sa metodom getEnumerator() kojeg implementira klasa Kolekcija koja pored metode getEnumerator() moze imati i druge metode koje pružaju vrijednosti kolekcije u nekom drugom redoslijedu ili obliku. U klasu Biblioteka dodajemo atribut tipa Map<ObicniKorisnik, Knjiga> i povezujemo klasu sa interfejsom IEnumerable.

State patern

State patern bismo mogli iskoristiti u nasem sistemu ako uvedemo da je knjiga u stanju “slobodna”, a kad se odobri njena rezervacija predje u stanje “rezervisana”.
Definisemo klasu Kontekst koja održava instancu stanja, koja definira tekući kontekst i interfejs koji je od interesa za klijenta. Odnosno, svaka knjiga ce imati svoj kontekst u kojem ce biti definisano njeno trenutno stanje(rezervisana ili slobodna). Klijent klasa komunicira sa ovom klasom. Definisemo i interfejs IState koji definise ponasanja povezana sa mogucim stanjama rezervacije pomocu metode handle(), ovaj interfejs implementuju klase SlobodnoStanje i RezervisanoStanje.

Template Method patern

Template Method patern bismo mogli iskoristiti u nasem sistemu ako promijenimo implementaciju razlicitog placanja clanarine obicnog i vip korisnika.
Definisemo klasu PlacanjeClanarine u kojem se nalazi metoda TemplateMethod(). Ovu klasu povezujemo je sa interfejsom IClanarina koji ima metode platiOsnovicu() i platiDodatak(). Ovaj interfejs implementuju klase ObicniKorisnik i VipKorisnik. TemplateMethod() u svom dijelu poziva ove funkcije i u zavisnosti od tipa korisnika platice se drugačija cijena. TemplateMethod() bi se pozivala u klasi Biblioteka u kojoj bismo dodali metodu platiClanarinu().