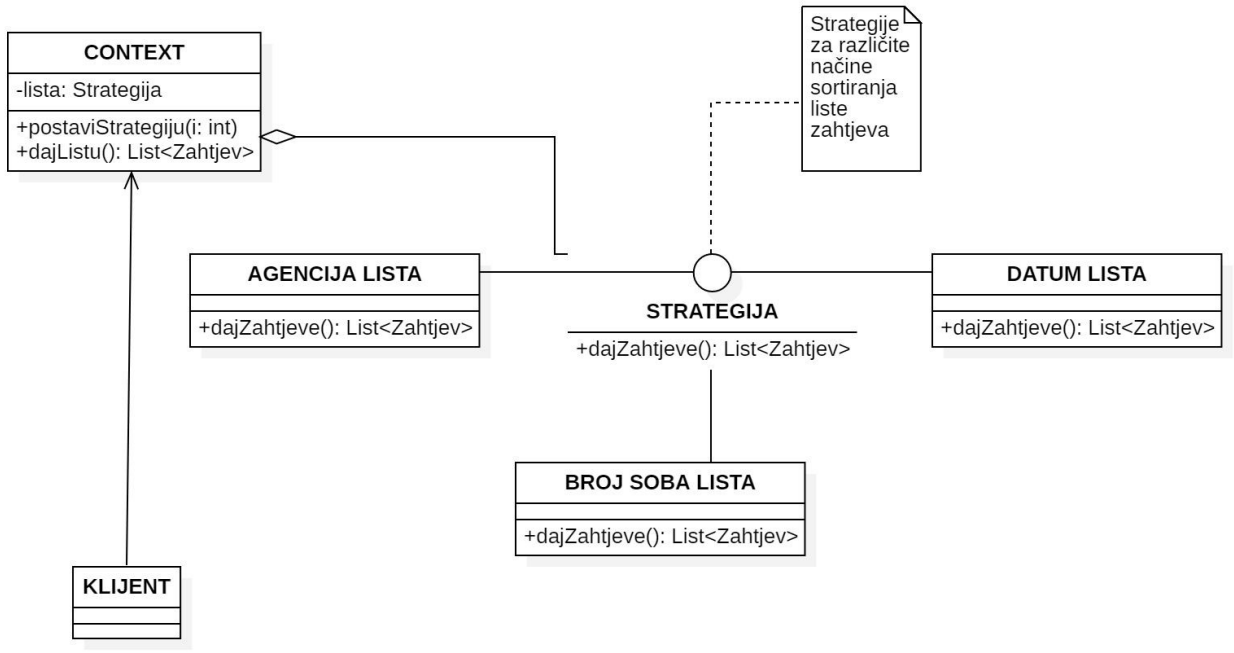
Paterni ponašanja

1. Strategy patern

Uloga Strategy patern je da izdvoji algoritam iz matične klase i uključi ga u posebne klase.

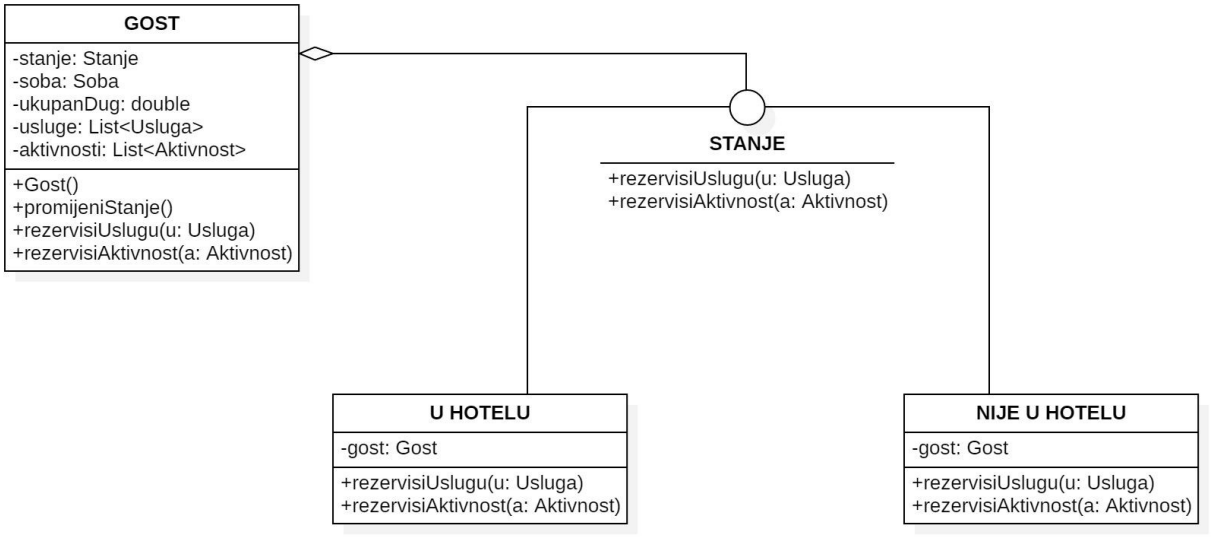
Ovaj patern smo iskoristili u našem projektu. Menadžer ima mogućnost da pregleda zahtjeve od strane agencija, sortirane po različitim kriterijima. Klasa *Context* ima atribut lista tipa interfejs Strategija kojeg implementiraju klase *AgencijaLista*, *BrojSobaLista*, *DatumLista*. Svaka od tih klasa iz metode *dajZahtjeve* vraća zahtjeve sortirane po odgovarajućem kriteriju.



2. State patern

Uloga State paterna je da mijenja način ponašanja nekog objekta na osnovu trenutnog stanja.

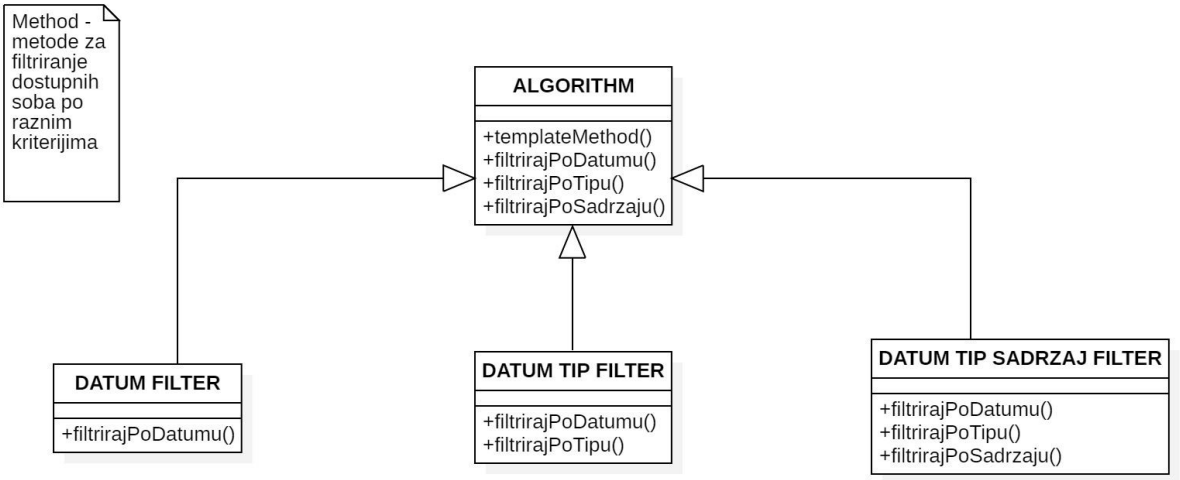
Ovaj patern smo iskoristili u našem projektu. Gost se može nalaziti u dva stanja – trenutno je u hotelu ili trenutno nije u hotelu. Za vrijeme boravka u hotelu, gost može rezervisati uslugu ili aktivnost, a to ne može uraditi ako u tom trenutku ne boravi u hotelu. Ovu funkcionalnost implementirali smo na sljedeći način. Klasa *Gost* ima atribut *stanje*, te metodu *promijeniStanje.* Interfejs *Stanje* sa metodama iz klase *Gost* vezanim za rezervaciju usluge i aktivnosti implementiraju pojedinačna stanja *UHotelu* i *NijeUHotelu*. Ako se npr. metoda *rezervisiUslugu* pozove kada je gost u stanju *NijeUHotelu*, ta metoda neće uraditi ništa, a ako se pozove kada gost jeste u hotelu, zaista će se rezervisati usluga.



3. Template Method patern

Template Method patern definira osnovnu strukturu algoritma, ali omogućava da se određeni koraci algoritma mogu implementirati različito u različitim podklasama.

Ovaj patern smo iskoristili u našem projektu. Klasa *Algorithm* je apstraktna klasa sa metodama *templateMethod*, *filtrirajPoDatumu*, *filtrirajPoTipu* i *filtrirajPoSadrzaju*. Ovim smo postigli da algoritam iz liste soba izdvaja sobe koje su slobodne u nekom određenom periodu, zavisno od toga da li je gost specificirao koji tip sobe želi, te koje sadržaje želi da ima u sobi. Dakle, ako gost specificira samo datum boravka, primijenit će se klasa *DatumFilter*, a ako pored datuma specificira i tip sobe, primijenit će se klasa *DatumTipFilter*.



4. Observer patern

Observer patern omogućava da o promjeni stanja jednog objekta budu obaviješteni svi “zainteresovani” objekti.

Ovaj patern nismo iskoristili u našem projektu. Mogli smo ga iskoristiti kada bismo željeli da prilikom dodavanja nove aktivnosti u hotel, o tome budu obaviješteni gosti koji u tom trenutku borave u hotelu. Tada bismo imali klasu *Publisher* koja bi imala kao atribut listu svih gostiju koji trebaju primati obavijest, a klasa *Hotel* bi kao atribut imala objekat tipa *Publisher*. Imali bismo i interfejs *IListener* sa metodom *update* kojeg bi implementirala klasa *Gost*. Prilikom dodavanja nove aktivnosti, pozivali bismo metodu *notify* klase *Publisher* koja bi pozivima metode *update* obavijestila sve prijavljene goste o postojanju nove aktivnosti.

5. Iterator patern

Iterator patern omogućava sekvencijalni pristup elementima kolekcije bez poznavanja strukture kolekcije.

Ovaj patern nismo iskoristili u našem projektu. Kada bismo u sistemu upravljali i narudžbama jela u hotelskom restoranu, patern bismo mogli iskoristiti na sljedeći način. Pretpostavimo da imamo klasu *Meni* u kojoj se nalazi lista objekata tipa *Jelo*. Tada bi klasa *Meni* implementirala interfejs *Collection* sa metodom *createIterator*, a dodali bismo i klasu *JeloIterator* koja bi implementirala interfejs *Iterator* sa metodama *hasNext* i *next*. Bilo bi moguće, pomoću iteratora, kretati se kroz elemente kolekcije *Meni*.