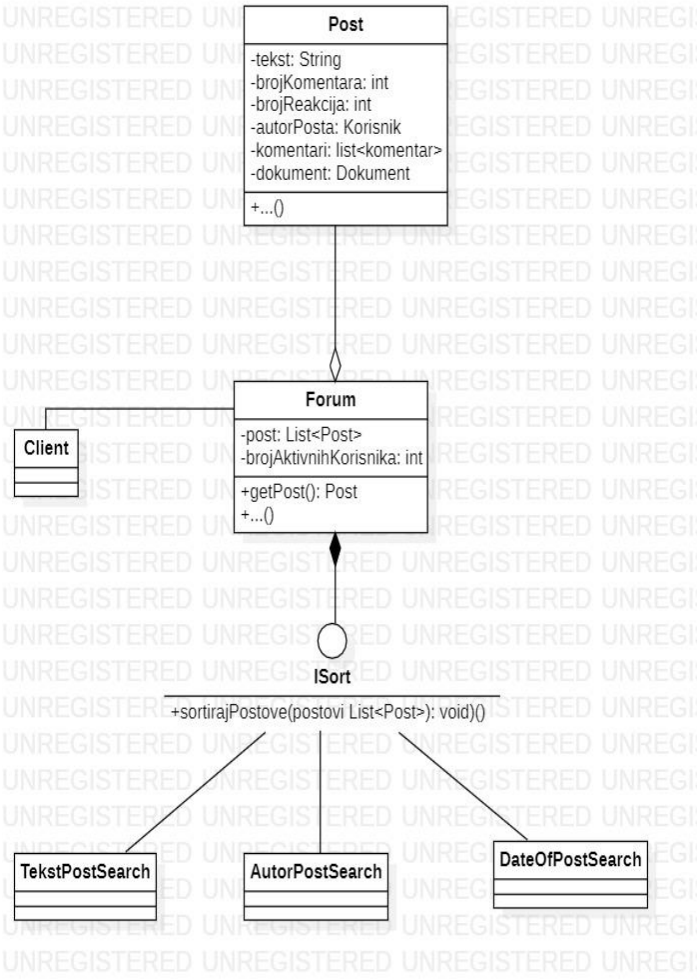
Paterni ponašanja

- 1. Strategy pattern

  Strategy pattern pogodan je kada postoje različiti primjenjivi algoritmi (strategije) za neki problem. U našem slučaju strategy patern možemo iskoristiti prilikom sortiranja podsjetnika na različite načine u ovisnosti da li želimo da sortiramo po datumu kreiranja, abecedi i sl. Uveli bismo Context klasu preko koje Korisnik klasa daje kontekstualne informacije Istrategy interfejsu koji bi u našem slučaju bio Isort interfejs(dodali bismo ga) iz njega se nasljeđuju različite mogućnosti sortiranja naših podsjetnika (po datumu,abecedi,važnosti..itd).

  Također, ukoliko bi korisnik htio pretražiti forum tj. pronaći određeni post, kao predmet pretraživanja bi se mogli naći: tekst posta, autor posta, datum objave posta itd. U tom slučaju bismo imali interfejs IpostSearch kojeg bi implementirale klase TekstPostSearch, AutorPostSearch, DateofPostSearch.



- 2. State pattern

State pattern je dinamička verzija Strategy paterna. Objekat mijenja način ponašanja na osnovu trenutnog stanja. Ovaj patern bismo primjenili da bismo omogućili pogodnosti korisnicima koji udome određeni broj pasa (min. 3 psa) dobijaju pogodnosti npr. besplatan pregled u tačno određenoj veterinarskoj stanici, no to bi iziskivalo veliki broj promjena u našoj aplikaciji pa ovaj patern nećemo implementirati.

- 3. Template method patern

Template method patern koristimo kada kompleksni algoritam izvršava iste korake ali se pojedinačni mogu izvršiti na različit način. Nećemo ga implementirati, ali mogli bi smo

mogli tako što bi omogućiti Korisniku da kreira profil psa koji bi imao iste metode kao i drugi psi ali i dodatne jer bi taj profil označavao psa koji se daje na udomljavanje što do sada nismo uopće uzimali u obzir.

- 4. Observe patern

Observe patern ima ulogu da uspostavi relaciju između objekata tako kada jedan objekat promijeni stanje drugi zainteresirani objekti se obavještavaju. Opet u našem projektu idalna implementacija ovog dijagrama je na primjeru podsjetnika gdje bi omogućili da se korisnik koji je kreirao podsjetnik za određeni datum i kada se taj dan i vrijeme uneseno za podsjetnik desi korisnik bi dobio obavještenje ranije da ima podsjetnik taj dan i u navedeno vrijeme,također možemo iskoristit i za udmoljavanje gdje će se korisniku poslati obavjest nakon ispunjenog zahtjeva kao potvrda udomljenja psa.

Subject bi nam bio administrator tj sistem koji prikazuje pse za udomljavanje

Dodali bi interfejs IObserver koji bi sadržavao metodu slanja potvde za udomljavanje putem e-maila korisniku.

Observer bi bio korisnik koji udomljava pse i tako smanjuje broj pasa koje je moguće udomiti

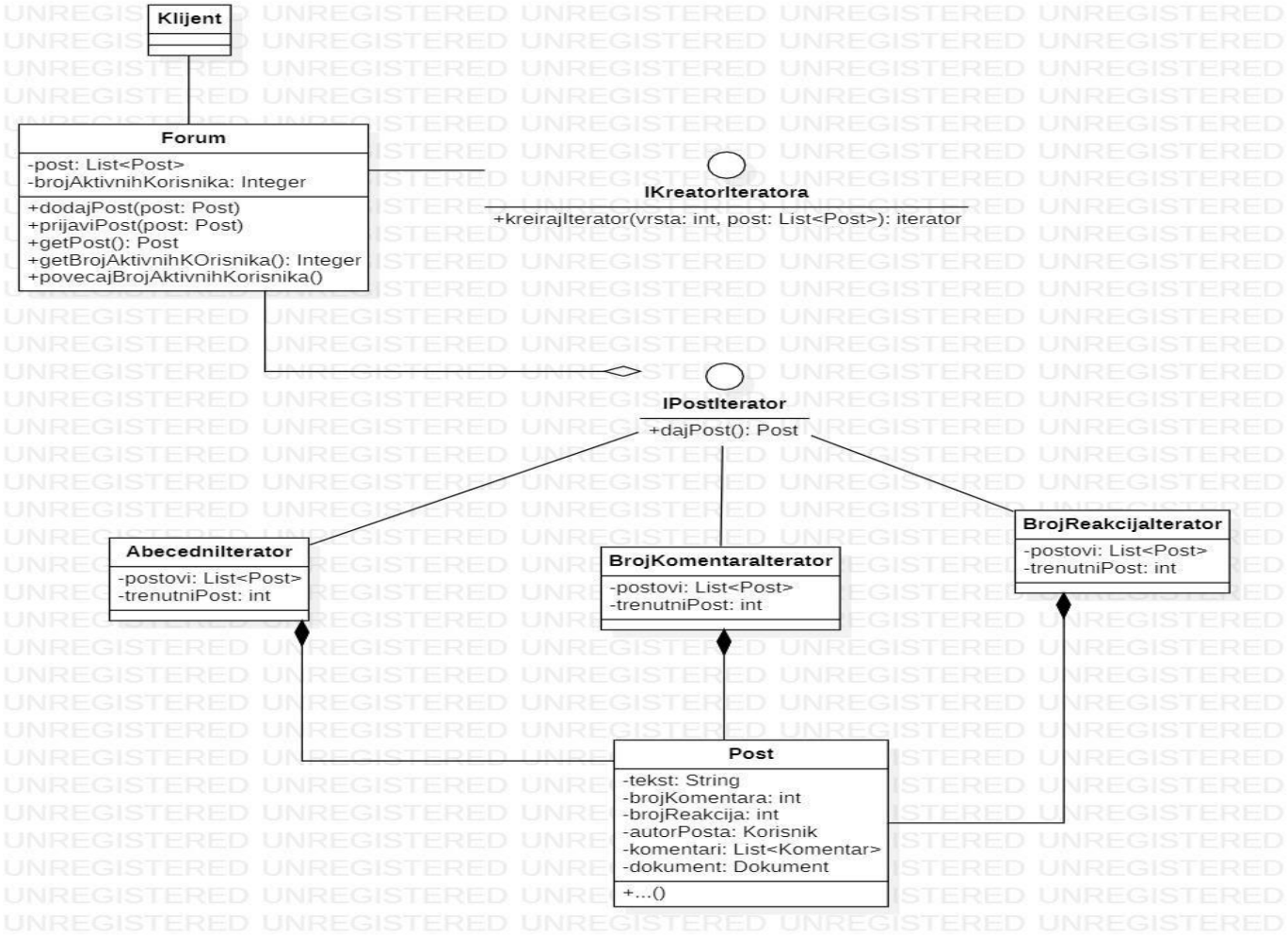
Notify-e-mail od sistema za sve udomitelje s korisnim informacijama State-zahtjev za udomljavanje

- 5. Iterator patern

Iterator pattern nam omogućava sekvencijalni pristup elementima kolekcije bez poznavanja kako je kolekcija strukturirana.

Nas sistem sadrzi klase Forum koja sadrzi atribut List<Post> te klasu Post koja sadrzi atribut List<Komentar>, ujedno to su mjesta gdje bismo mogli primijeniti Iterator pattern.

To radimo na način da kreiramo interfejs IPostIterator, AbecedniIterator, BrojKomentaraIterator, BrojReakcijaIterator..., što bi nam omogucilo da iteriramo kroz listu postova sortiranih po: abecedi, broju komentara, broju reakcija...



- 6. Chain of responsibility

Osnovna uloga Chain-of-responsibility patterna je da jedan kompleksni proces obrade razdvoji na način da više objekata procesiraju primljenje podatke na razlicite načine.

Obzirom na primitivnost aplikacije koju kreiramo, ovaj pattern nema neku konkretnu primjenu. Međutim, ukoliko bismo unaprijedili aplikaciju na način da bismo omogućili

korisnicima da kreiraju grupne postove, sam postupak bi se izvršavao korištenjem interfejsa IZajednickaObjava. Svaki učesnik u postu bi trebao lancano potvrditi samu objavu tj sadrzaj posta putem metode iz IZajednickaObjava.

- 7. Mediator patern

Mediator pattern enkapsulira protokol za komunikaciju među objektima dozvoljavajući da objekti komuniciraju bez međusobnog poznavanja interne strukture objekta.

Pristup sekcije „Forum“ aplikacije KomPas imaju svi korisnici, međutim aktivno učestvovanje u istom imaju samo Registrovani korisnici. Iako se provjera da li je korisnik registrovan ili ne vrši na mnogo jednostavniji način, ukoliko bismo htjeli u ovoj situaciji primijeniti Mediator pattern to bismo uradili na način da bismo kreirali interfejs IprovjeraKorisnika (ima ulogu posrednika) koji bi sadržao metodu koja bi vršila provjeru korisnika, da li je registrovan ili ne.