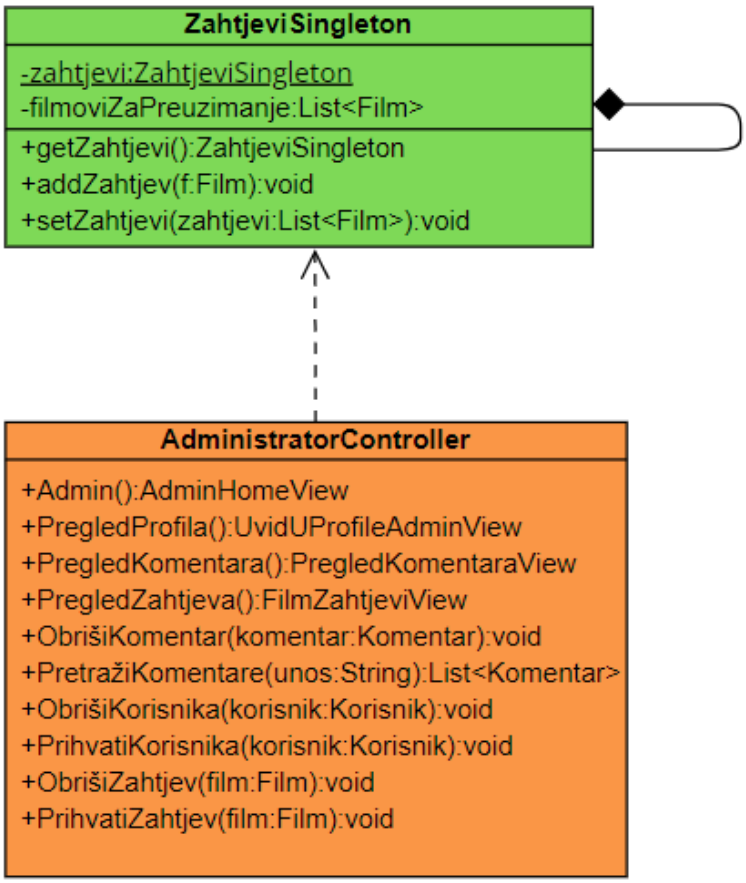
KREACIJSKI PATERNI

1.) Singleton pattern

Uloga Singleton paterna je da osigura da se klasa može instancirati samo jednom i da osigura globalni pristup kreiranoj instanci klase.

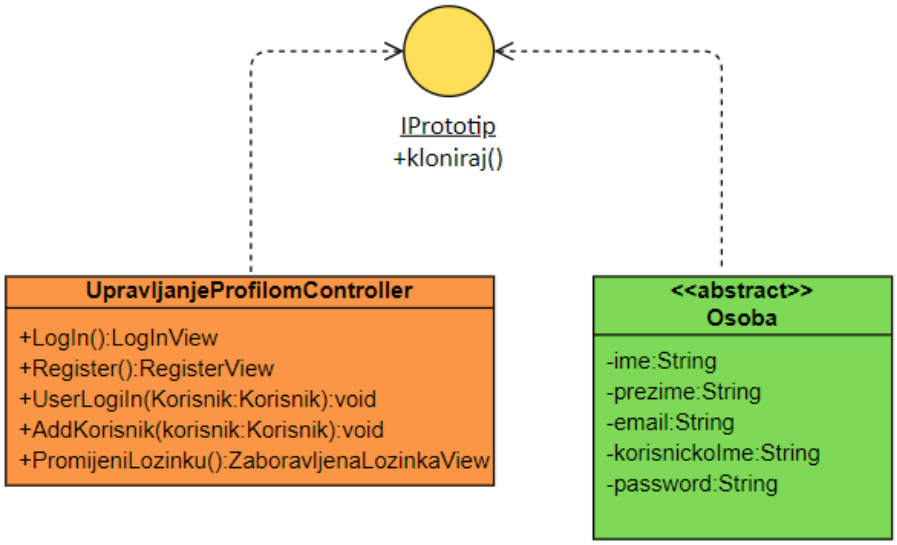
Da bi implementirali ovaj patern u sistemu, morali smo uvesti novu klasu ZahtjeviSingleton, koja sadrži statički atribut, getter i setter. Ovu klasu će koristiti Administrator, koji ima mogućnosti spašavanja filmova sa servera u bazu podataka. Svi filmovi koji se skinu sa servera nalazit će se u instanci singleton objekta u vidu zahtjeva. Te zahtjeve može samo administrator prihvatiti ili odbaciti.



2.) Prototype pattern

Osnovna funkcija ovog paterna je da olakša kreiranje objekata koristeći postojeću instancu, koja se ponaša kao prototip.
Novokreiranom objektu možemo promijeniti određene osobine po potrebi.

U našem sistemu, ovaj patern bi mogli iskoristiti nad klasom Osoba. Ukoliko postoje dvije ili više osoba sa istim imenom i prezimenom, umjesto kreiranja novog objekta, možemo klonirati prvu instancu i izmijeniti odgovarajuće podatke koji se razlikuju (email, username, password). Kreirali bi interfejs IPrototip sa metodom kloniraj koja implementira način kloniranja objekta.

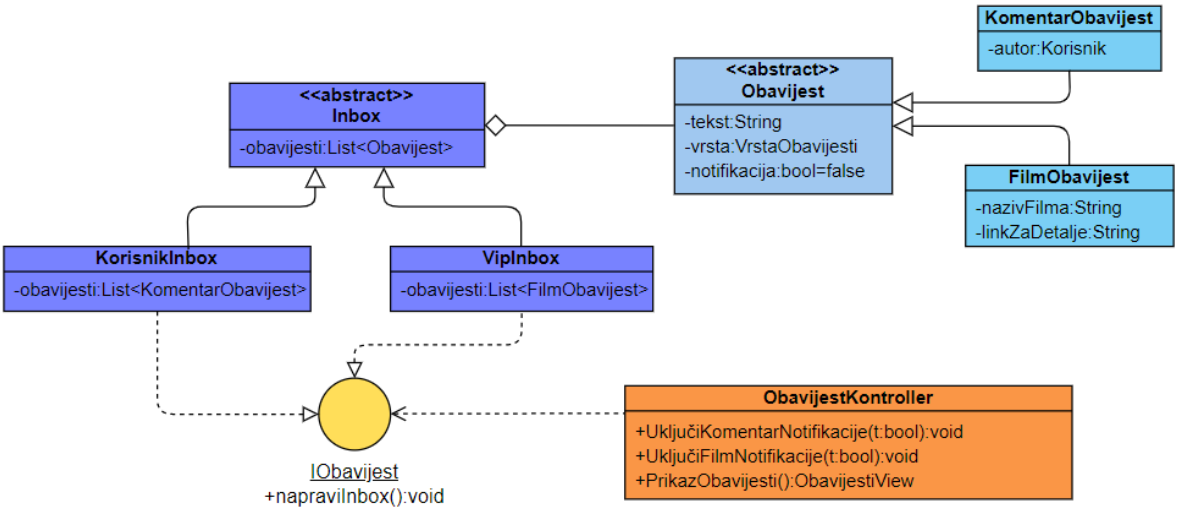


3.) Factory method

Factory method patern omogućava kreiranje objekata na način da podklase odluče koju klasu instancirati.
Ovo se radi preko interfejsa sa jednom metodom koju različite podklase mogu implementirati drugačije.

Ukoliko uvedemo klasu Inbox, koja može biti za VIP korisnika ili za običnog korisnika. U zavisnosti od vrste inboxa, drugačije obavijesti će se kreirati. Ovo je

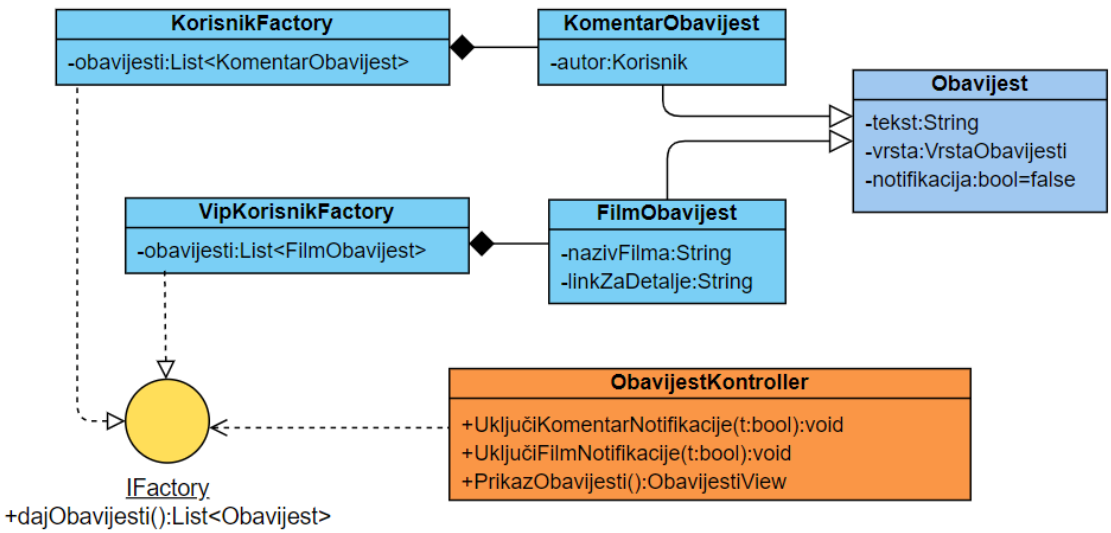
odgovarajući scenario gdje možemo iskoristiti Factory method.



4.) Abstract factory

Ovaj patern nam omogućava kreiranje više familija objekata sa mogućnošću dodavanja novih u budućnosti. Abstract factory odvaja definiciju (klase) produkata od klijenta zbog čega se familije produkata mogu jednostavno izmijenjivati ili ažurirati bez narušavanja strukture klijenta.

Ovaj patern bismo implementirali u našem dijagramu da su nam Korisnik i VIPKorisnik odvojene klase, gdje Korisnik može imati samo KomentarObavijest-i dok VIP samo FilmObavijest-i.



5.) Builder pattern

Builder paterna koristimo kada nam je potrebno primijeniti različite postupke za konstrukciju istog objekta.

Da bismo primijenili ovaj patern u našem sistemu, potrebno je uvesti dvije klase: DefaultBuilder i UserBuilder. Oni sadrže atribut tipa Kolekcija. Uvođenjem interfejsa IBuilder, omogućili smo dva načina konstrukcije Kolekcija. UserBuilder se odnosi na kreiranja kolekcija čije osobine korisnik određuje (naziv, filmove, automatsko dodavanje ocijenjenih filmova u kolekciju), dok DefaultBuilder kreira kolekciju sa podrazumijevanim imenom I filmovima iz baze podataka.