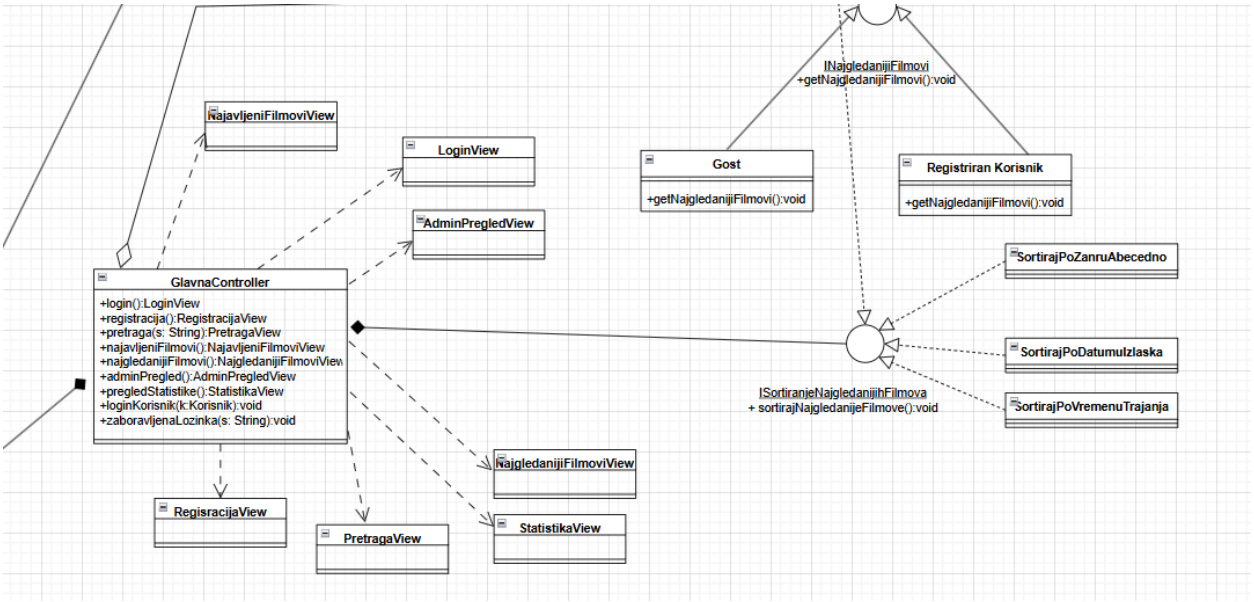
Patterni ponašanja

1. Strategy pattern

Strategy patern izdvaja algoritam iz matične klase i uključuje ga u posebne klase. Pogodan je kada postoje različiti primijenjivi algoritmi(strategije) za neki problem i omogućava klijentu izbog jedne od strategija za korištenje. Korištenje ovog paterna nam olakšava brisanje ili dodavanje novih algoritama koji se po želji mogu koristiti.

Ovaj patern smo implementirali na način da smo uveli tri nova algoritma za sortiranje najgledanijih filmova. Korisnici mogu da biraju da sortiraju najgledanije filmove po zanru abecedno, po vremenu trajanja filma i po datum izlaska.

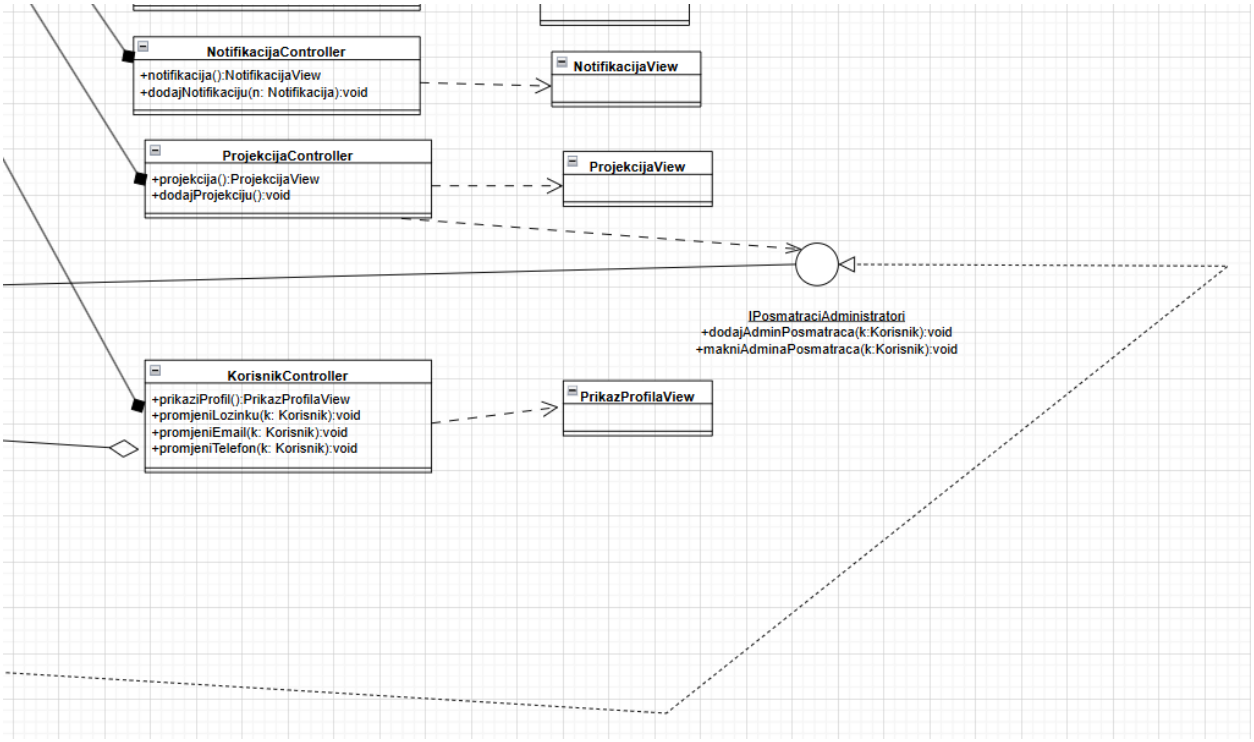


2. Observer pattern

Uloga Observer paterna je da uspostavi relaciju između objekata tako da kada jedan objekat promijeni stanje, drugi zainteresirani objekti se obavještavaju.

Jednu od mogućnosti što smo dodali jeste da Zaduzeni ljudi mogu obavijestiti posmatrace o desavanjima tokom projekcije i promjenama u tijeku projekcije te i mogucnost da obavijeste sve korisnike koji gledaju taj film u datoj projekciji.

Povezali smo s klasom Korisnici i Projekcije.



3. State pattern

State patern omogućava objektu da mijenja svoja stanja, od kojih zavisi njegovo ponašanje. Sa promjenom stanja objekat se počinje ponašati kao da je promijenio klasu. Stanja se ne mijenjaju po želji klijenta, već automatski, kada se za to steknu uslovi. State Pattern je dinamička verzija Strategy paterna.

Ovaj pattern bismo mogli iskoristiti npr da u sistemu imamo klase VIP i NonVip zajedno s interfejsom IStanje. Ideja je sljedeca da Vip korisnici imaju mogućnost pregleda više trailer jednog filma dok NonVip korisnici imaju mogućnost jednog osnovnog trailera. Nalazila bi se metoda pregledTrailera koja bi na osnovu tipa korisnika bila drugačije realizovana.

4. Template method patern

Template method patern služi za omogućavanje izmjene ponašanja u jednom ili više dijelova. Najčešće se primjenjuje kada se za neki kompleksni algoritam uvijek trebaju izvršiti isti koraci, ali pojedinačne korake moguće je izvršiti na različite načine.

U našoj aplikaciji to bi mogli uraditi na način da prilikom logina na aplikaciju korisnik koji je administrator ako bi želio da koristi svoje administratorske ovlasti morao bi unijeti pin koji

bi mu dodjeli mogućnost korištenja u suprotnom dok ne unese on će se smatrati običnim korisnikom. Realizovali bismo neku metodu adminPin() koja bi vršila provjeru pina i dodjelila role ako je taj pin tačan.

5. Iterator

Ovaj patern omogućava sekvencijalni pristup elementima kolekcije bez poznavanja kako je kolekcija struktuirana.

Ovo bi smo mogli npr implementirati tako što bi osmisliti način na koji bi se iterativno prolazilo kroz kolekciju filmova tj listu filmova koji su dostupni za gledanje u kinu. To bi smo mogli uraditi tako što bi dodali tipa klase VremenskiIterator i tipa OcjenaIterator gdje bi birali između toga na koji način bi se prolazilo kroz filmove da li putem vremenski predloženi termina (poredani od manjeg ka većem) ili putem Ocjena za dati film.

6. Chain of responsibility patern

Chain of responsibility patern namijenjen je tome kako bi se jedan kompleksni proces obrade razdvojio na način da više objekata na različite načine procesiraju primljene podatke.

Ovo bismo mogli realizovati tako da smo dali tipa mogućnost administratoru da kreira obavijesti. Imali bismo dvije vrste to su random (za korisnike i goste) i komentarfilm (samo za korisnike). Administrator kreira nesto od ovoga, za random unosi neki tekst dok za komentarfilm unosi neki komentar za određeni film. Stvari koje nemaju se automatski nalaze na internetu.

7.) Medijator patern

Mediator patern namijenjen je za smanjenje broja veza između objekata. Umjesto direktnog međusobnog povezivanja velikog broja objekata, objekti se povezuju sa međuobjektom medijatorom, koji je zadužen za njihovu komunikaciju. Kada neki objekt želi poslati poruku drugom objektu, on šalje poruku medijatoru, a medijator prosljeđuje tu poruku namijenjenom objektu ukoliko je isto dozvoljeno.

Ovo bismo mogli implementirati tako da kao kreiramo nekog tipa “bota” koji bi da imamo log svih komentara koji su postavljeni u sekcijama za filmove provjeravao te logove i pronalazio riječi koje nisu adekvatne za našu kino stranicu te bi poslao obavijest administratorima da uklone datu riječ i sankcionišu korisnika.

8. Command

Command je patern ponašanja koji pretvara zahtjev u samostalni objekt koji sadrži sve informacije o zahtjevu. Ova transformacija vam omogućava da prosljeđujete zahtjeve kao argumente metode, odlažete ili stavljate u red izvršenja zahtjeva i podržavate operacije koje se ne mogu izvršiti. Ovaj patern možemo iskoristiti za neke komande, koje se nalaze na različitim mjestima a proizvode isti kod.