PROXY

Primjena:

Ovaj patern cemo iskoristiti za kontrolu pristupa komentiranja i ocjenjivanja filma. Ako je korisnik registrovan kao kritičar on će moći izvršiti operaciju komentiranja i ocjenjivanja, a ako nije onda će mu se ponuditi opcija koja mu omogućuje da se registruje kao kritičar.

FLYWEIGHT

Primjena:

Ovaj patern bi mogli primjeniti na objektima klase MovieInfo, u slučaju kad nije moguće dobiti informacije o filmu potrebno je imati default MovieInfo koje će služiti kao placeholder. Da ne bi imali više objekata klase MovieInfo u memoriji dovoljno je imati jedan i djeliti ga među objektima klase Movie koja umjesto da sadrži čitav objekat MovieInfo može samo sadržavati referencu, kao što je napr. id ili indeks nekog niza koji sadrži objekte MovieInfo. Time bi zauzeće memorije bilo manje, a samim tim i performanse sistema bolje.

COMPOSITE

Primjena:

Patern Struktura (ili Composite) je pogodan u slučaju kada je potrebno obrisati korisnike sistema, tačnije Kritičare. Kako je Kritičar u mogućnosti ostavljanja recenzija na filmove, kada njegov profil bude obrisan, trebale bi biti i sve recenzije koje je on napisao biti obrisane. Tako da je definisan interfejs koji implementiraju klase Critic i CommentRating. Ulogu Component klase igra CommentRating, dok je Composite klasa u našem slučaju Critic. Sami klijent koji je direktno povezan sa ovim paternom jeste Administrator klasa, obzirom da je administratoru data mogućnost brisanja korisnika.

ADAPTER

Primjena:

Patern Adapter se može najlakše primjeniti u slučaju uske povezanosti klasa Administrator i Editor. Obzirom da interfejs koji posjeduje Editor klasa, može biti u većini slučajeva biti iskorištena i od strane Administrator klase, kako je to na neki način klasa na ‘višem nivou moći’. Klasa Editor će nam biti Adaptee jer je tu interfejs koji trebamo prilagoditi za upotrebu i od strane ITarget klase koja će kod nas biti klasa Administrator. Adapter nam služi kako bismo implementirali novi zahtijevani interfejs preko već postojećeg.

DECORATOR

Ovaj patern ćemo iskoristiti prilikom izrade CriticController klase jer kritičar ima sve funkcionalnosti kao i registrovani korisnik i još dodatne koje će biti u decorator klasi.

FACADE

Ovaj patern ćemo iskoristiti prilikom pravljenja Home stranice, gdje ćemo objediniti funkcionalnosti CommentView-a, StreamView-a, DownloadView-a unutar HomeView kao i funkcionalnosti kontrolera CommentController i MovieController unutar MainController.

BRIDGE

Primjena:

Ono gdje bi bilo najpogodnije iskoristiti Bridge pattern, odnosno takozvani most je upravu u klasi Movie. U toj klasi, možemo imati više različitih implementacija, koje potiču iz određene apstrakcije. U konkretnom smislu, Abstraction klasa bi bila sama klasa Movie, odnosno interfejs apstrakcije koji bi sadržavao ono što bi bilo uvijek isto, kao što su možda ocjene, recenzije i slično. Pored toga klasa Bridge definira apstrakciju i može imati više implementacija. Ovdje zasad imamo dvije, prva implementacija, bi bila nova klasa MovieStream, koja bi bila zadužena konkretno za streamovanje filma, a druga implementacija bi bila isto nova klasa MovieDownload, čija bi uloga bila samo posjedovanje informacije vezanih za skidanje filmova. Ovako smo razdvojili dvije različite implementacije vezane za klasu Movie. Na ovaj način, ako kasnije se odluči raditi još nešto sa filmom (dodavanje titlova i tome slično) može biti lagano usklađeno u ovakav patern kako bi se izbjeglo zapetljavanje.