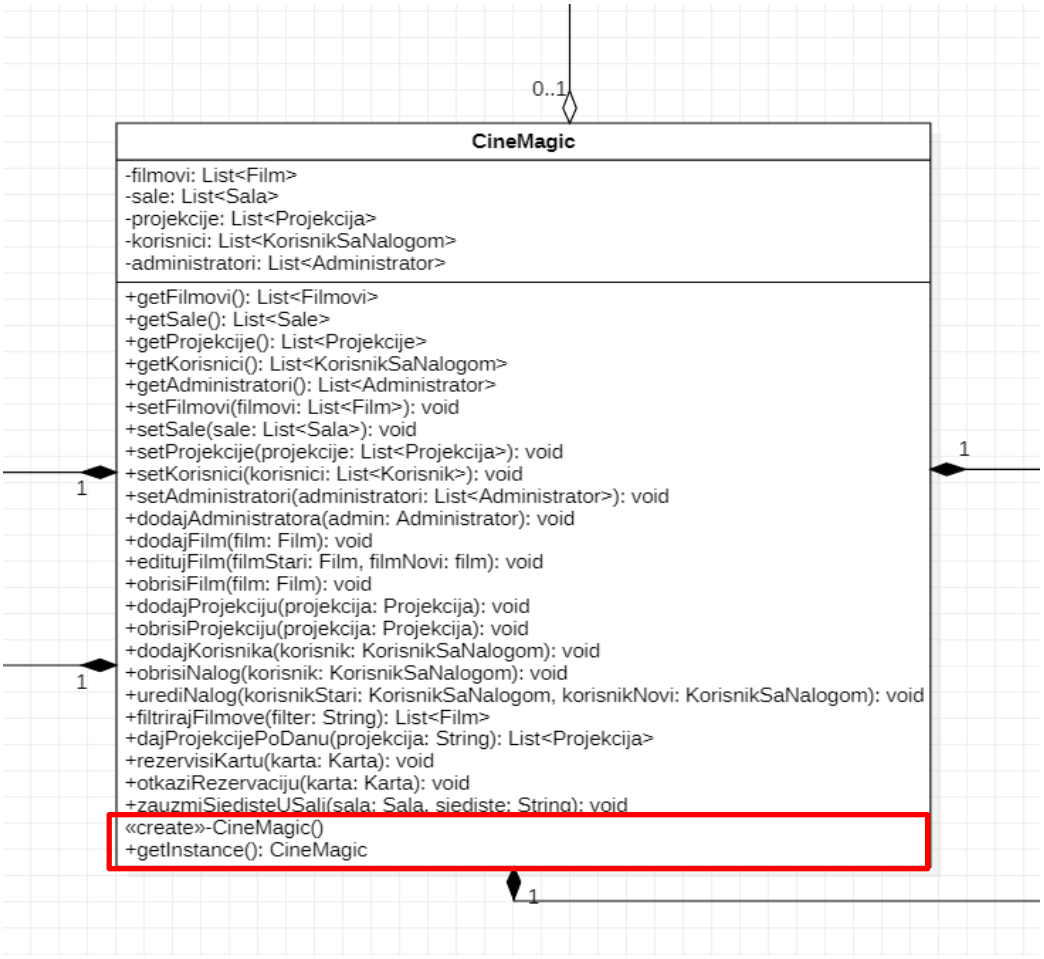
KREACIJSKI PATERNI

1. Singleton pattern

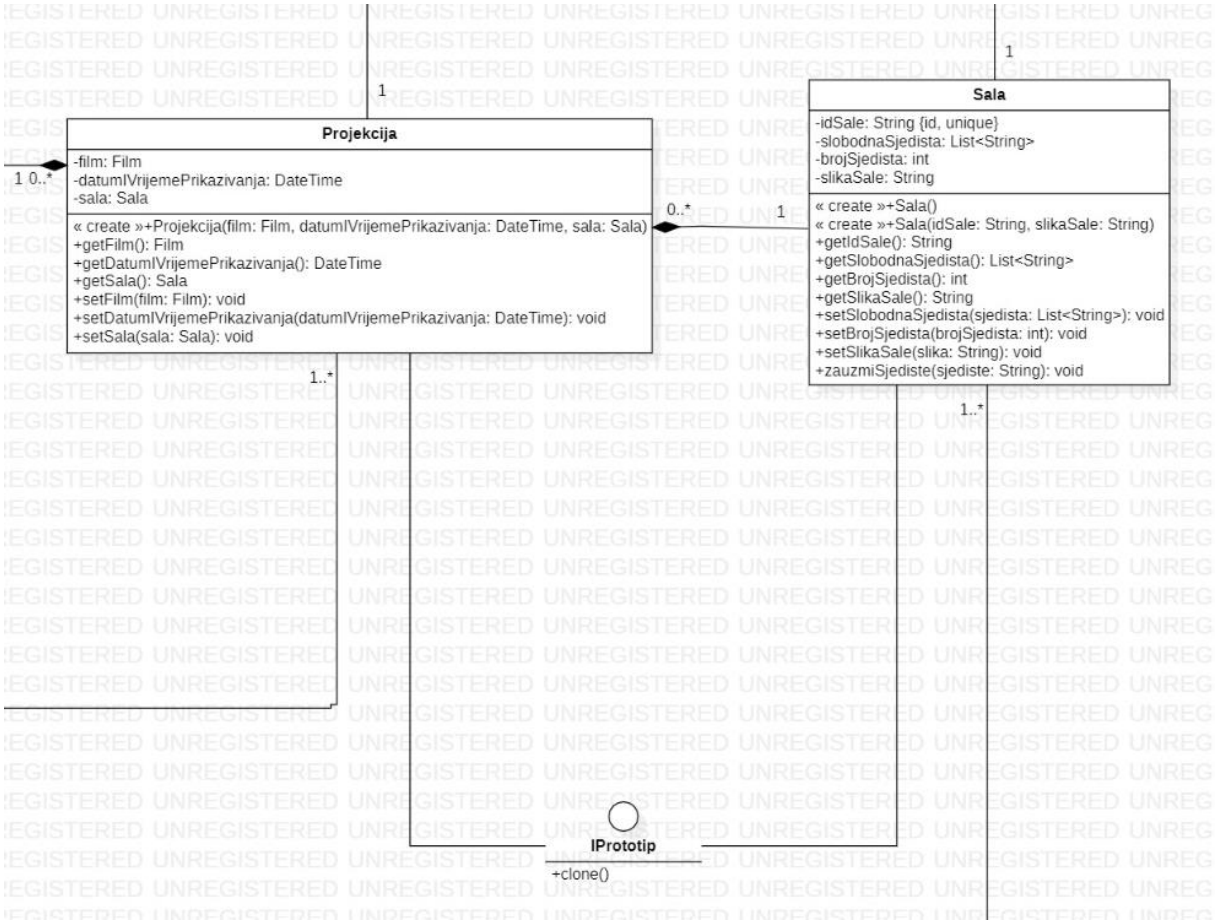
Singleton patern služi kako bi se neka klasa mogla instancirati samo jednom. Na ovaj način može se omogućiti i tzv. lazy initialization, odnosno instantacija klase tek onda kada se to prvi put traži. Osim toga, osigurava se i globalni pristup jedinstvenoj instanci - svaki put kada joj se pokuša pristupiti, dobiti će se ista instanca klase. Ovo olakšava i kontrolu pristupa u slučaju kada je neophodno da postoji samo jedan objekat određenog tipa. Možemo promjeniti ovaj patern na klasi CineMagic, jer će biti potrebno da imamo samo jednu instancu ove klase. Kada bi imali više instanci došlo bi do komplikacija.



2. Prototype pattern

Prototype patern omogućava smanjenje kompleksnosti kreiranja novog objekta tako što se uvodi operacija kloniranja. Na taj način prave se prototipi objekata koje je moguće replicirati više puta a zatim naknadno promijeniti jednu ili više karakteristika, bez potrebe za kreiranjem novog objekta nanovo od početka. Ovime se osigurava pojednostavljenje procesa kreiranja novih instanci, posebno kada objekti sadrže veliki broj atributa koji su za većinu instanci isti.

U našem projektu smo implementirali ovaj pattern, tako što smo dodali interfejs *IPrototip*, koji ima metodu *clone()*. Ovim interfejsom smo smanjili kompleksnost kreiranja novog objekta, a klase koje ga implementiraju su *Projekcija* i *Sala*.



3. Factory Method pattern

Factory method patern služi za omogućavanje instanciranje različitih vrsta podklasa koristeći factory metodu koja odlučuje koja će se podklasa instancirati i koja programska logika izvršiti. Na ovaj način osigurava se ispunjavanje O SOLID principa, jer se kod za kreiranje objekata različitih naslijeđenih klasa ne smješta samo u jednu zajedničku metodu, već svaka podklasa ima svoju logiku za instanciranje željenih klasa, a samo instanciranje kontroliše factory metoda koju različite klase implementiraju na različit način. Ovaj pattern nismo implementirali. Njegova implementacija bi mogla biti moguća kod kreiranja korisnika. Dodali bi interfejs IKorisnik, koji bi imao metodu *dajUsera(),* dodali bi i *Kreator* klasu koja bi imala metodu *FactoryMethod().* Ova metoda odlučuje koju klasu instancirati.

4. Abstract Factory pattern

Abstract factory patern služi kako bi se izbjeglo korištenje velikog broja if-else uslova pri kreiranju različitih hijerarhija objekata. Ukoliko postoji više tipova istih objekata te različite klase koriste različite podtipove, te klase postaju fabrike za kreiranje objekata zadanog podtipa bez potrebe za specificiranjem pojedinačnih objekata. Na ovaj način

se, korištenjem nasljeđivanja, ukida potreba za postojanjem if-else uslova jer određeni tip fabrike sadrži određene tipove objekata i zna se tačno koju podklasu će instancirati. Nismo implementirali ovaj pattern, ali kada bi se odlučili da ga implementiramo, ne bi ga bilo teško uvesti. Iskoristili bi ga za smanjenje if-else uslova pri filtriranju po žanrovima, kreiranjem *IFactory* interfejsom kojim bismo olakšali postupak dobijanja instance potrebnog žanra. Međutim, mi smo ovo riješili na drugi način, pomoću enumeracije.

5. Builder pattern

Builder patern služi za apstrakciju procesa konstrukcije objekta, kako bi se kao rezultat mogle dobiti različite specifikacije objekta koristeći isti proces konstrukcije. Ovaj patern koristi se kako bi se izbjeglo kreiranje kompleksne hijerarhije klasa te kako bi se izbjegao kompleksni programski kod konstruktora jedne klase koja može imati različite konfiguracije atributa. Različiti dijelovi konstrukcije objekta izdvajaju se u posebne metode koje se zatim pozivaju različitim redoslijedom ili se poziv nekih dijelova izostavlja, kako bi se dobili željeni različiti podtipovi objekta bez potrebe za kreiranjem velikog broja podklasa. Nismo implementirali dati pattern, ali kada bi naša aplikacija bila dostupna u više gradova, mogli bismo implementirati ovaj pattern na sljedeći način. Pri ulasku na našu stranicu tražila bi se informacija o gradu u kojem se korisnik nalazi, i pomoću te informacije bi *KojiGradBuilder* klasa, koja implementira *IBuilder* interfejs, generisala početnu stranicu sa projekcijama i filmovima koji se projektuju u odabranom gradu.