Copadore

OOAD 2019-2020

<GRUPA 7>

Članovi tima:

Mujkanović Mirza

Mujkić Fejsal

# Strukturalni patterni:

|  |  |
| --- | --- |
| Naziv | Singleton |
| Uloga | Uloga Singleton paterna je da osigura da se klasa može instancirati samo jednom i da osigura globalni pristup kreiranoj instanci klase. |
| Da li ga koristimo u projektu? | DA |
| Primjena | Na klasi „Videoteka“ ćemo primijeniti ovaj pattern , omogućit ćemo da se ova klasa može samo jednom istancirati jer postoji samo jedna videoteka i tako osiguravamo globalni pristup instanci te klase. |

|  |  |
| --- | --- |
| Naziv | Prototype |
| Uloga | Uloga Prototype paterna je da kreira nove objekte klonirajući jednu od postojećih prototip instanci (postojeći objekat). |
| Da li ga koristimo u projektu? | NE |
| Primjena | Efikasnost ovog paterna bi se mogla iskoristiti kod pravljenja različitih istanci iz klase „Film“ u projektu, gdje za rezultat dobijemo da nije potrebno kreiranje novog filma nego samo izmjena nekog od postojećih filmova |

|  |  |
| --- | --- |
| Naziv | Factory Method |
| Uloga | Uloga Factory Method paterna je da omogući kreiranje objekata na način da podklase odluče koju klasu instancirati |
| Da li ga koristimo u projektu? | NE |
| Primjena | Efikasnost ovog paterna bi se mogla iskoristiti kod pravljenja različitih istanci iz klase „Film“ u projektu, gdje za rezultat dobijemo da nije potrebno kreiranje novog filma nego samo izmjena nekog od postojećih filmova |

# Kreacijski patterni:

|  |  |
| --- | --- |
| Naziv | Proxy |
| Uloga | Primjenom ovog paterna omogućava se kontrola pristupa objektima, te se onemogućava manipulacija objektima ukoliko neki uslov nije ispunjen, odnosno ukoliko korisnik nema prava pristupa traženom objektu. |
| Da li ga koristimo u projektu? | DA |
| Primjena | Primijenit ćemo ovaj pattern jer imamo upsivanje u bazu , odnosno dodavanje korisnika, administratora ili filmova a nismo izvršili zabranu pristupa običnom korisniku za pisanje u bazu, dakle tu mogućnost ima samo administrator. |

|  |  |
| --- | --- |
| Naziv | Decorator pattern |
| Uloga | Osnovna namjena mu je da omogući dinamičko dodavanje novih elemenata i ponašanja postojećim elementima |
| Da li ga koristimo u projektu? | NE |
| Primjena | Ovaj pattern bi se u našem projektu mogao primijeniti kod dodavanja nove opcije za prikaz filmova(rezolucija, 3D prikaz,....) |

|  |  |
| --- | --- |
| Naziv | Factory method pattern |
| Uloga | Uloga ovog patterna je da omogući kreiranje objekata na način da podklase odluče koju klasu istancirati |
| Da li ga koristimo u projektu? | NE |
| Primjena | Ne postoji dobro mjesto za njegovu upotrebu o ovom projektu. |

**Paterni ponašanja**

|  |  |
| --- | --- |
| Naziv | Strategy pattern |
| Uloga | Izdvaja algoritam iz matične klase i uključuje ga u posebne klase. Omogućava klijentu izbor jednog od algoritma iz familije algoritama za korištenje. |
| Da li ga koristimo u projektu? | DA |
| Primjena | Ovaj pattern ćemo primijeniti u našem projektu prilikom izbora klijenta koji od načina plaćanja želi da koristi prilikom kupovine filma, gdje bira izmedju kartičnog plaćanja i plaćanja putem paypala. |

|  |  |
| --- | --- |
| Naziv | TemplateMethod |
| Uloga | Omogućava izdvajanje određenih koraka algoritma u odvojene podklase. |
| Da li ga koristimo u projektu? | DA |
| Primjena | Iskoristit ćemo ga kod sortiranja filma po cijeni i ocjeni, tako što ćemo algoritam sortiranja izdvojiti u posebnu klasu. Npr sortiranje po ocjeni:  Algorithm – Sortiranje, TemplateMethod – Bubble sort, Iprimitives – ocjena filma, AnyClass – Film, Operation – Poređenje dva filma |

|  |  |
| --- | --- |
| Naziv | Observer pattern |
| Uloga | Uloga ovog patterna je da uspostavi relaciju izmedju objekata tako da kada jedan objekat promijeni stanje, drugi zainteresirani objekti se obavještavaju. |
| Da li ga koristimo u projektu? | NE |
| Primjena | Dobar primjer primjene ovog patterna bi bio ako dođe do promjene cijene filma koji je na korisnikovoj listi želja, da taj korisnik dobije obavještenje. |