KREACIJSKI PATERNI

Singleton [implementacija]

Singleton patern podrazumijeva instanciranje objekta klase samo jednom, te je nakon toga objektu omogućen globalni pristup. Koristi se kada instanciranje objekata klase više od jednom može prouzrokovati probleme kao što su nekorektno ponašanje programa, neadekvatno korištenje resursa ili nekonzistentni rezultati.

U našem sistemu Singleton patern se odnosi na instanciranje klase Repertoar, budući da će u sistemu postojati samo jedan repertoar čiji će se atributi mijenjati jednom mjesečno.

Prototype [implementacija]

Prototype patern omogućava kreiranje objekata bez poznavanja detalja o njegovoj klasi. Omogućava konstrukciju objekta kloniranjem i modifikacijom ranije kreiranog objekta. Koristi se kada je kreiranje novog objekta resursno zahtjevno.

U našem sistemu patern je moguće implementirati na klasi Karta, s obzirom da se u instancama klase Karta za jednu predstavu razlikuju samo njihovi ID-evi. Na ovaj način ćemo izbjeći kreiranje više sličnih objekata sa istim atributima jer će svi atributi biti kopirani, a ID će biti modificiran.

Factory Method

Svrha ovog kreacijskog paterna je da omogući kreiranje objekata tako da podklase odluče koja od njih će biti instancirana. Koristi se kada klijent ne mora ili ne treba znati koji tip objekta će biti instanciran, nego mu je samo bitno da može izvršiti određene funkcionalnosti. Factory podklasa ovim ima punu kontrolu nad kreacijskim procesom, a izbjegnuta je bilo kakva veza između klijenta i konkretnih produkata.

Kako ćemo u sklopu našeg sistema pratiti broj odgledanih predstava za korisnike, te omogućiti određeni popust nakon izvjesnog broja kupljenih karata, ovaj patern bi se mogao u budućoj verziji sistema iskoristiti na način da korisnik dobiva samo obavijest o rezervaciji bez da zna da li je rezervacija sa popustom ili nije.

Abstract Factory

Ovaj patern omogućava kreiranje porodica povezanih objekata bez specificiranja konkretnih klasa. Na osnovu apstraktne porodice produkata kreiraju se konkretne fabrike produkata različitih tipova i kombinacija. Klijent radi samo sa asptraktnim klasama i ne zna za stvarne objekte koji su kreirani fabrikama.

Kako Abstract Factory često predstavlja nadogradnju Factory Method paterna za kompleksnije sisteme, a naš sistem nije dovoljno kompleksan, nije moguća implementacija Abstract Factory-a.

Builder patern

Builder patern omogućava kreiranje različitih tipova objekata koristeći isti konstrukcijski kod. Koristi se kada se objekti mogu podijeliti u skupove jednostavnijih objekata koji se razlikuju samo po permutaciji njihovih dijelova. Njegovom upotrebom se zadovoljava Single Responsibility princip i izbjegava ponavljanje koda.

Kako se Builder patern često koristi u aplikacijama koje kreiraju kompleksne strukture koje nisu prisutne u našem sistemu, nije moguće pronaći njegovu primjenu.