KREACIJSKI PATERNI

Potreba za dodavanjem kreacijskih paterna u sistem se prvobitno ogleda u cilju povećanja nivoa principa enkapsuiliranja, kako bi se dovelo do što većeg odvajanja korištenja sistema i u ovom slučaju kreiranja klasa koje sistemu trebaju. Cilj ovog postupka je sličan kao i kod mnogih drugih paterna, a to jeste smanjenje mogućnosti greške, te ako se greška i desi, pokušati ćemo minimizirati njen utjecaj. Želimo da greška uvijek utiče na aspekte sistema, a ne na čitav sistem.

U našem slučaju kreacijski paterni koje ćemo primjeniti su:

- Singleton patern
- Builder patern

1.**Singleton pattern**

-ovaj patern osigurava da postoji samo jedna instanca određene klase koja je dostupna globalno. U softver inženjeringu, singleton pattern je dizajn pattern koji ograničava instanciranje klase na jednu jedinu instancu. Ovo je korisno kada je potreban tačno jedan objekt za koordinaciju akcija u cijelom sistemu, i koristi se tamo gdje je potrebna samo jedna instanca klase za kontrolu radnje tokom cijelog izvršenja. Singleton klasa ne bi trebala imati više instanci ni u kom slučaju i po svaku cijenu. Singleton patern možemo primjeniti na obavijesti koja se prosljedjuje svim korisnicima koji uspješno izvrše narudžbu, tj. kako korisnik vidi obavijest, nije potrebno praviti zasebnu instancu za svakog korisnika.

2. **Builder pattern**

-koristi se za konstruisanje složenih objekata korak po korak. **Builder pattern** se koristi u slucaju kreiranja kompleksnih objekata gdje nerijetko dolazi do pojave velikog broja parametara (konstruktor pretezno) od kojih vecina moze biti neiskoristena.Takodjer, moze biti iskoristen u situacijama kada se razliciti objekti kreiraju na slicne nacine. Koristan je kada objekat ima mnogo opcionalnih parametara ili složene konfiguracije.Primjer builder paterna u našoj aplikaciji mogao bi se ogledati u kreiranju kompleksnog objekta narudžbe koja može imati različite dodatke, napomene i posebne zahtjeve itd.