Kreacijski paterni

Singleton patern:
Možemo primjeniti na obavijest koja se prosljeđuje svim korisnicima, tj. kako svaki korisnik vidi tu obavijesti, nije potrebno praviti zasebnu instancu za svakog korisnika.

Prototype patern:
Pošto u sistemu postoji mogućnost da korisnik obnovi rok narudžbe, ovaj patern bi mogli primjeniti u ovom slučaju, tj. umjesto da korisniku ponovno prikazujemo interfejs dizaniranja, možemo jednostavno kopirati prethodni narudžbu i promjeniti preostalo vrijeme.

Factory method patern:
Ukoliko pored klase MerchStore imamo dodatnu klasu PremiumStore koja takođe nasljeđuje klasu Store, tada bi ovaj patern mogli iskoristiti na način da u zavisnosti od toga koji tip Store-a korisnik želi kreirati, taj tip se i kreira pomoću neke druge klase ili interfejsa.

Abstract factory patern:
Ukoliko pored MerchStore imamo i PremiumStore, te pored NarudzbaDizajnera imamo PremiumNarudzba i ove dvije vrste narudzbi nasljeđuju klasu Narudzba, tada možemo iskoristiti ovaj patern jer bi store klase imaju zajedničku metodu dodajNarudzbu, a narudzbe mogu biti različitog tipa.

Builder patern:
Builder patern se može primjeniti kod klase Store. Kako je za kreiranje store-a potrebno generisanje koda, možemo korisniku dozvoliti da kreira vlastiti kod ili da se kod generiše automatski.