KREACIJSKI PATERNI

SINGLETON PATERN

Singleton patern osigurava da se klasa može instancirati samo jednom i da postoji globalni pristup kreiranoj instanci klase. Ovaj patern se koristi za objekate koje je potrebno samo jednom instancirati i nad kojim je potrebna jedinstvena kontrola pristupa.

Ovaj pattern se može primijeniti pri registraciji korisnika na sistem. Zamišljeno je da se pri tome izvrši jedna konekcija na bazu te da se svi upiti pišu u jednoj klasi te da se kroz nju dobavljaju podaci iz baze. Ovo je najbolje obaviti kroz Logger klasu. Ona će nam osigurati singleton konekciju na bazu i globalni pristup istoj, odnosno bilo koja klasa će moći pristupiti instanci te Logger klase i preko nje dobiti sve željene informacije.

FACTORY METHOD PATERN

Factory Method patern omogućava kreiranje objekata na način da podklase odluče koju klasu instancirati. Različite podklase mogu na različite načine implementirati interfejs. Factory Method instancira odgovarajuću podklasu(izvedenu klasu) preko posebne metode na osnovu informacije od strane klijenta ili na osnovu tekućeg stanja.

Factory Method patern možemo iskoristiti pri kreiranju instanca klase Product odnosno kada moramo odlučiti da li će kreirani Product biti Accessories, Shoes ili Clothing. Imat ćemo klasu ProductCreator koja posjeduje metodu productCreationMethod koja će kreirati odgovarajući tip Product u zavisnosti od parametra koji je proslijeđen koji predstavlja odabrani tip Producta koji korisnik želi kreirati.
U kodu bi navedeni pater izgledao ovako:

```
ProductCreator pc = new ProductCreator();
Product product = pc.productCreationMethod(productType : String);
```

BUILDER PATERN

Builder patern služi za apstrakciju procesa konstrukcije objekta, kako bi se kao rezultat mogle dobiti različite specifikacije objekta koristeći isti proces konstrukcije. Uloga Builder paterna je odvajanje specifikacije kompleksnih objekata od njihove stvarne konstrukcije. Isti konstrukcijski proces može kreirati različite reprezentacije.

Pošto konstruktor klase User sadrži veliki broj parametaran, builder patern možemo iskoristiti za kreiranje objekata tog tipa. Koristeći ovaj patern kreiranje objekta bi obavljali u klasi UserBuilder, pa bi i sve validacije podataka (email, korisničko ime, datum rođenja, informacije o plaćanju), koje bi se inače dešavale u konstuktoru klase User i odgovorajućim seterima za atribute, sada smjestili u tu klasu. Ovo olakšava i dodavanje novih atributa u klasu User.

PROTOTYPE PATERN

Prototype dizajn patern dozvoljava da se kreiraju prilagođeni objekti bez poznavanja njihove klase ili detalja kako je objekat kreiran. Ovaj patern prepušta proces klaniranja samom objektu koji se klonira preko zajedničkog interface-a koji implemtiraju sve klase koje podržavaju kloniranje. Obično taj interface sadrži samo jednu metodu - clone(). Clone metoda vrlo je slična u svim klasama. Metoda stvara objekt trenutne klase i prenosi sve vrijednosti atributa starog objekta u novi.Objekt koji podržava kloniranje naziva se prototip (prototype). Kada zatreba objekat koji je već konfigurisan umjesto da se pravi novi objekat od nule samo se klonira prototip.

Prototype patern je korisno ugraditi u projekte s obzirom da je često postrebno više istih objekata. U našem projektu bi kloniranje podrzavala klasa Products kao i klase koje je nasljeđuju. Klasa User nema smisla da implemtira ovaj interface jer nike doyvoljeno da postoje dva identična korisnika (user-a).

ABSTRACT FACTORY PATERN

Abstract Factory patern nam omogućava da se kreiraju familije povezanih objekata odnosno produkata. Na osnovu apstraktne familije produkata kreiraju se konkretne fabrike produkata različitih tipova i različitih kombinacija a pošto patern odvaja definiciju klase produkata od klijenta, familije produkata je moguće jednostavno prikazivati, mijenjati i ažurirati.

Abstract Factory patern možemo iskoristiti kada korisnik bude vršio filtriranje proizvoda tokom pretrage. Na osnovu korisničkih filtera kreiraće se i fabrika produkata različitih tipova artikala kao i različitih kombinacija. Ovim bi se prikaz artikala učinio dosta efikasnijim s obzriom da Abstract Factory patern upravlja familijama produkata i čuva njihove detalje neovisnim od klijenata.