Kreacijski paterni

1. Singleton pattern

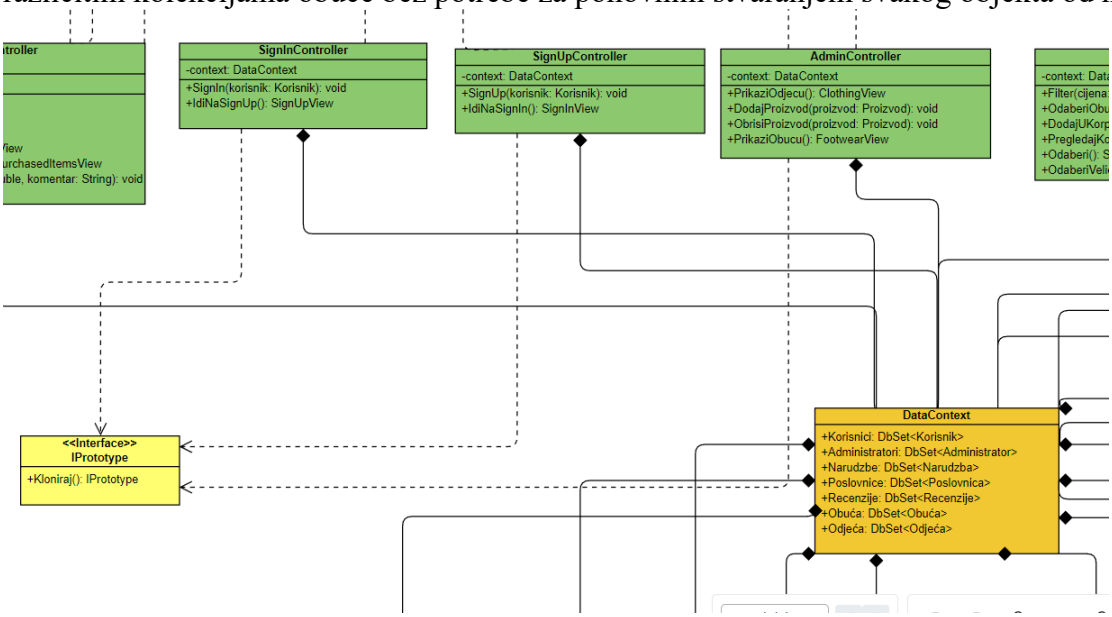
Singleton pattern moguće je iskoristiti na sljedeći način:

Korpa za kupovinu:
Naša web aplikacija podržava dodavanje proizvoda u korpu za kupovinu. Možemo koristiti Singleton pattern za implementaciju klase koja predstavlja korpu. Na taj način ćemo osigurati da se samo jedna instanca korpe koristi tokom cijele sesije kupovine.

2. Prototype pattern

Prototype pattern moguće je iskoristiti na sljedeće načine:

Kreiranje korisničkih profila:
Naša web aplikacija ima korisničke profile, Prototype pattern se može primijeniti za stvaranje novih profila na temelju postojećih prototipa. Možemo imati prototip za svaku vrstu profila, kao što su korisnik ili administrator. Kada korisnik stvori novi profil, koristimo prototip za stvaranje kopije i prilagodimo je s jedinstvenim podacima korisnika. Na taj način, možemo brzo stvarati nove profile s minimalnim ponavljanjem koda.
Kreiranje kolekcija obuće:
Naša web aplikacija nudi razne stilove obuće, poput cipela ili čizama, Prototype pattern se može primijeniti za stvaranje kolekcija obuće. Možemo imati prototip za svaki stil obuće, koji uključuje osnovne karakteristike i dizajn. Kada korisnik odabere stil obuće, koristimo prototip za stvaranje kopije i prilagodimo je veličinom i drugim pojedinostima. Na taj način možemo efikasno upravljati različitim kolekcijama obuće bez potrebe za ponovnim stvaranjem svakog objekta od nule.



3. Factory method

Factory method moguće je iskoristiti na sljedeći način:

Factory za boje odjeće:
Factory method možemo koristiti za stvaranje objekata koji predstavljaju različite boje odjeće, poput Crvena, Plava, Zelena, itd. Factory boja može imati metodu stvoriBoju(boja) koja na temelju pružene boje instancira i vraća objekt koji predstavlja odgovarajuću boju. Ovo olakšava dodavanje novih boja odjeće u budućnosti.

4. Abstract factory pattern

Abstract factory pattern moguće je iskoristiti na sljedeći način:

Abstract factory za proizvode:
Abstract Factory pattern možemo iskoristiti za stvaranje familije proizvoda koji su međusobno povezani. Na primjer, možemo imati abstract factory OdjevniProizvodiFactory koja definira apstraktne metode poput stvoriGornjiDio(), stvoriDonjiDio() i stvoriObucu(). Zatim možemo implementirati konkretne factories poput SportskiProizvodiFactory, EleganciaProizvodiFactory i CasualProizvodiFactory, koje će stvarati specifične proizvode (npr. sportske odjevne predmete, elegantnu odjeću ili ležernu odjeću). Na taj način možemo osigurati da se odjevni predmeti u projektu međusobno slažu i imaju konzistentan stil.
Abstract factory za lokalizaciju:
Ukoliko bi naša web aplikacija podržava više jezika, Abstract Factory pattern mogao bi se koristiti za stvaranje lokaliziranih objekata. Na primjer, možemo imati abstract factory LokalizacijaFactory koja definira apstraktne metode poput stvoriNaslov(), stvoriOpis() i stvoriDugme(). Zatim možemo implementirati konkretne factories kao što su na primjer EngleskiLokalizacijaFactory, FrancuskiLokalizacijaFactory i NjemačkiLokalizacijaFactory, koje će stvarati objekte lokalizirane na odgovarajućem jeziku. Na taj način možemo lahko prilagoditi svoju aplikaciju za različite jezike.

5. Builder pattern

Builder pattern moguće je iskoristiti na sljedeći način:

Builder za korpu za kupovinu:
Naša web aplikacija ima funkcionalnost korpe za kupovinu, te Builder pattern možemo koristiti za konstrukciju korpe sa stavkama. Možemo imati KorpaBuilder koji ima metode poput dodajStavku(), ukloniStavku(), ocistiKorpu(), itd. Korisnik može koristiti ove metode za dodavanje, uklanjanje i upravljanje stavkama u korpi. Kada je korpa konstruirana, možemo pozvati metodu dajKorpu() koja će vratiti objekt koji predstavlja korpu sa svim stavkama. Na taj način korisnici mogu graditi i prilagođavati svoju korpu prije nego što obave plaćanje.
Builder za filtiranje proizvoda:
Naša web aplikacija omogućava korisnicima da filtriraju proizvode u aplikaciji na temelju određenih kriterija. Builder pattern možemo koristiti za izgradnju filtera. Možemo imati FilterProizvodaBuilder koji ima metode poput dodajVelicinu(), dodajBoju(), postaviCijenu() i tako dalje. Korisnici mogu koristiti ove metode kako bi postavili kriterije filtriranja proizvoda. Kada je filter konstruiran, možemo pozvati metodu dajFiltriraneProizvode() koja će vratiti objekt koji sadrži proizvode koji zadovoljavaju postavljene kriterije.
Builder za korisničke profile:
Naša web aplikacija ima korisničke profile s različitim atributima, te Builder pattern možemo koristiti za izgradnju korisničkih profila. Možemo imati KorisnickiProfilBuilder koji ima metode poput postaviIme(), postaviPrezime(), postaviEmail() i tako dalje. Korisnici mogu koristiti ove metode kako bi postavili željene atribute svog profila. Kada je profil konstruiran, možemo pozvati metodu dajKorisnickiProfil() koja će vratiti objekt koji predstavlja profil korisnika.