Paterni ponašanja

Strategy

Strategy patern izdvaja algoritam iz matične klase i uključuje ga u posebne klase. Definira porodicu algoritama, enkapsulira svaki od njih i omogućava klijentu izbor jednog od algoritama.

U okviru aplikacije DressCode, Strategy pattern bi se mogao koristit kod sortiranja artikala gdje korisniku omogućeno da izabere sortiranje po vrsti artikla, veličina, boja, spolu itd.

Ovime se izbjegava pretjerano grananje u kodu. Postižemo da možemo zamijeniti algoritme u toku izvršavanja programa, odvojiti implementaciju od koga koji ga koristi i ne narušavamo Open/Closed princip jer možemo uvesti nove strategije bez promjene postojećih klasa.

State

Mijenja način ponašanja objekata na osnovu trenutnog stanja. Za razliku od Strategy paterna gdje klijent odlučuje koju strategiju će odabrati kod State paterna objekat sam mijenja stanje bazirano na interakciji klijenta.

Ovaj patern bi u našem sistemu bio koristan u kontroli stanja artikala. Različita stanja koja zavise od interakcije korisnika su:

-artikal ubačen u korpu
-artikal validiran i proslijeđen u narudzbu
-artikal plaćen

U zavisnosti od ovih stanja mijenja se način ponašanja za artikal. Ako je korisnik potvrdio narudžbu i izvršio plaćanje mijenja se količina artikala na stanju.
S druge strane je artikal dodan u korpu i u tom stanju u međuvremenu nestane zaliha za isti artikal (količina padne na 0) artikal se treba ukloniti iz korpe.

Ovim paternom se ne narušava Single responsibility princip jer dio koda koji se odnosi na pojedinačna stanja je u posebnim klasama niti Open/Closed princip jer uvodimo nova stanja bez promjene postojećih stanja klasa.

TemplateMethod

Omogućava izdvajanje određenih koraka algoritma u odvojene podklase.
Struktura algoritma se ne mijenja - mali dijelovi operacija se izdvajaju i ti se dijelovi mogu implementirati različito.

Ovaj patern se našem sistemu može implementirati u procesu postavljanja novog artikla. To je algoritam koji ima ima uvijek iste korake: validacija osnovnih podataka, validacija specifičnih podataka, postavljanje u bazu podataka.

Validacija osnovnih podataka je zajednička za sve naše artikle (naziv, cijena, opis), Validacija specifičnih podataka ovisi o vrsti artikla - za određene vrste odjeće postoje različite stavke koje se moraju navesti u specifikacijama. Zbog toga bismo primjenom TemplateMetode ovu operaciju izdvojili iz algortima u odredjene podklase gdje se implementira različito. Isti slučaj je i sa metodom koja postavlja u bazu podataka jer može biti razlike u specifikacijama odredjenih artikala u zavisnosti od vrste i samim tim se ne spašavaju jednako u bazu.

Ovim bismo dobili jednostavniju strukturu, a pojedine dijelove implementacije bi bilo jednostavnije za mijenjati.

Chain of responsibility

– Omogućava lančano obrađivanje zahtjeva. Predstavlja listu objekata, ukoliko objekat ne može da odgovori prosljeđuje zahtjev narednom u nizu

Moguća primjena u našem sistemu moze biti preko validacije narudžbe. Tokom narudzbe, korisnik unosi svoje ime, prezime, adresu i podatke o placanju. Da imamo još mogućnosti kao npr. dostave trebalo bi napraviti i validaciju tih podataka prilikom narudžbe. Mogli napraviti apstarktnu metodu za validaciju narudžbe a zatim konkretne klase za validaciju adrese, podataka o placanju i podataka o dostavi. Svaka od od ovih provjera može biti dio lanca odgovornosti koji vrši specifičnu provjerui ako je sve u redu, prosljeđuje dalje. Na ovaj nacin se obezbjeduje modularna i prosirivu validaciju bez „if-else" konsturkcije. Također omogućava se da se dodaju nove validacije bez izmjene postojećeg koda.

6. Iterator

- Iterator patern omogućava pristup elementima kolekcije bez poznavanja kako je kolekcija strukturirana. Navigacija kroz kolekciju podataka provodi se koristeći zajednički interfejs bez znanja o njihovoj temeljnoj implementaciji.

Primjeri u našem sistemu:

Lista artikala - kada se korisnik nalazi na pogledu Artikli (tu može i izabrati određeni filter za sortiranje), iterator omogućava prolazak kroz sve proizvode, stranicu po stranicu, bez obzira koliko ima elemenata u bazi i kako su spremljeni.

Admin Panel - Administrator može proći kroz sve registrovane radnike ili sve pending narudžbe, bez obzira kako su podaci organizovani u pozadini.

Korpa - Jednostavan prolazak kroz sve stavke u korpi ili listi želja, bez brige o tome kako su te stavke interno spremljene.

7. Mediator – enkapsulira protokol za komunikaciju među objektima dozvoljavajući da objekti komuniciraju bez međusobnog poznavanja interne strukture objekta
Ovaj patern bi mogao poslužiti za komunikaciju između narudžbe i korisniškog profila. Ako se narudžba uspješno izvrši, mediator brine o postavljanju obavijesti korisniku i dodavanju narudžbe u listu narudžbi korisnika, bez da se metode za obavijesti i dodavanje pozivaju što smanjuje zavisnost klasa jednim o drugima i čini kod preglednijim i jednostavnijim.

Observer
Uspostavlja relaciju između objekata takvu da kad se stanje jednog objekta promijeni, svi vezani objekti dobiju informaciju.

Ovaj patern omogućava da jedan objekat (korpa) obavještava druge objekte (npr. Notifikacije) automatski o promjenama svog stanja, bez da ih direktno povezuje. U klasi Korpa - kada kupac doda/ukloni artikal iz korpe, automatski se ažurira prikaz ukupne cijene, broja stavki i šalje se notifikacija korisnicima,
Kada korisnik klikne "Naruči" sačuvaju se podaci o trenutnoj cijeni i ID_narudžbe koji iz tog kontrolera prosljeđuju kao informacija kontroleru za plaćanje te se vrši direkcija na PlaćanjeContoller.

Takodjer u klasa Narudzba - kada se promijeni status narudžbe, automatski se šalju notifikacije kupcu i te informacije koristi ArticalController da obradi artikle u skladu sa stanjem narudžbe.

9. Visitor – omogućava definiranje i izvršavanje nove operacije nad elementima postojeće strukture ne mijenjajući samu strukturu

Može se koristiti za dodavanje različitih operacija nad artiklima, poput računanja popusta ili generisanja izvještaja o artiklima.

Ovaj patern bismo mogli iskoristiti u situaciji kada želimo izvrsiti različite statisticke analize nad korisnicima i proizvodima - npr. broj narudzbi po korisniku, ukupan prihod po kategoriji odjeće, prosječna cijena proizvoda itd. (Sve bez promjene same strukture klasa)

10. Interpreter – podržava interpretaciju instrukcija napisanih za određenu upotrebu. U našoj aplikaciji nisu podržani pisani upiti. Primjena ovog paterna u budućnosti bi mogla biti sljedeća. Korisnik bi mogao napisati neki složeniji upit za filtriranje artikala, a interpreter bi to protumačio i vratio odgovarajuće rezultate. Umjesto da korisnik mora kliknuti 10 različitih filtera, želi se omogućiti da korisnik unese:

- "crvene haljine veličina M do 100 KM popust više od 20%".

11. Memento – omogućava spašavanje internog stanja objekta i kasniji povrat prethodnog stanja objekta bez narušavanja enkapsulacije. Ovaj patern se u online sistemu odjeće može koristiti za spašavanje artikala koji više nisu dostupni. Kada artikal postane nedostupan, sistem može sačuvati njegovu konfiguraciju (veličina, boja, cijena) i automatski ga vratiti u ponudu kada se zalihe obnove. Također mogli bismo ga iskoristiti za funkcionalnost „poništi promjenu" (undo) kod uredivanja korisničkog profila.