Strukturni paterni

1. *Adapter*

   U klasi “Narudzba” i “Korpa”, posjedujemo atribut koji vraća listu artikala. Mogli bismo napraviti adapter koji bi nam umjesto liste vraćao artikle koji su uređeni npr. po veličini, cijeni i sličnom.

2. *Facade*

   U našem sistemu će biti iskorištene metode kao što su sortiranje i filtriranje. Međutim kako bi iskoristili ovaj paterni, mogli bismo ove metode smjestiti u jednu klasu, class Fasadu, s obzirom na njihovu usku povezanost. Kako se ovaj patern oslanja na to da nas ne zanima šta je “ispod haube”, nas neće zanimati algoritam kojim će se iskoristiti za sortiranje ili filtriranje. U klasu Fasada bi mogli smjestiti metode kao što su filtriraj/sortiraj po veličini, cijeni, određenoj kategoriji i sličnom.

3. *Decorator*

   Kad bi htjeli iskoristiti ovaj strukturni patern, mogli bismo dodati neke funkcionalnosti našem administratoru koji vodi računa o izgledu stranice. Npr. mogli bismo mu dodati neke funkcije kao što je ispis cijene zavisno od valute date države, ispis veličine zavisno od kontinenta, ispis datuma po evropskom ili američkom standardu.

4. *Bridge*

   Možemo primijetiti da je ovaj patern iskorišten u klasi “PlacanjeRacuna”. Imamo dvije mogućnosti plaćanja, a to je kartično i popreuzeću. Ako se osvrnemo na kartično plaćanje, ono ima doticaja s vanjskim uređajem kao što je banka. Znamo da sve banke nemaju isti način uplate i skidanja novca, pa bi mogli naparviti intefejs “Bridge” koji će definisati apstrakciju i ima različite implementacije za svaku banku.

5. *Proxy*

   Budući da se najčešće ovaj strukturni patern koristi za čuvanje podataka, neku autentifikaciju, dobili smo ideju da bi mogli u našem sistemu implementirati klasu „AutentificationProxy“ koja nasljeđuje interfejs sa metodom za log in (ona bi primala korisnicko ime i lozinku), koja bi odobravala ili odbijala pristupanje korisničkom računu ukoliko neki podatak nije tačan, istu stvar bi mogli iskoristiti ukoliko bi se u sistemu tražilo da se upiše lozinka ukoliko se želi obrisati korisnički račun.

6. *Composite*

Kada bi imali neki view na kojem bi administrator zajedno želio prikazati PlacanjePoPreuzecu i KarticnoPlacanje na nekoj listi, pošto su to dva različita objekta njihov prikaz bi bio realizovan na različit način, tako da KarticnoPlacanje imamo bankovni racun korisnika, a za PlacanjePoPreuzecu imamo datum isporuke. Iz tog razloga smo napravili interfejs „IPrikaz“ te klasu „ListaZaPrikazComposite“ koja u sebi drzi listu objekata tipa „IPrikaz“ te kao i KarticnoPlacanje i PlacanjePoPreuezecu implementira istoimeni interfejs. Sada prilikom poziva metode „Prikazi“ za listu objekata „IPrikaz“ dobijati ce se odgovarajuća „Prikazi“ metoda.

7. *Flyweight*

Pošto se ovaj strukturni patern koristi kada bi neke naše modelske klase imale neku osobinu koja se naknadno postavlja (neobavezna je za rad same aplikacije) te se toj osobini dodjeljuje neka default vrijednost koja se nalazi na samom sistemu radi smanjenog korištenja memorije. Pošto smo dizajnirali naš sistem tako da koristimo samo podatke (atribute) koji su osnovni i potrebni u aplikaciji, nemamo veliku potrebu za ovim paternom. Jedino gdje bi mogli iskoristiti ovaj patern je sa slikama kada Administrator dodaje neki novi proizvod (iako je neophodno da svaki artikl ima sliku), ako ne doda slike za odredjeni artikl da on dobije neku defaultnu vrijednost (neke slike na nasem sistemu).