STRUKTURALNI PATERNI

1. **ADAPTER PATTERN**

Adapter patern se koristi kada je potreban drugačiji interfejs već postojeće klase bez mijenjanja postojeće klase. Tako dobijamo željenu funkcionalnost bez izmjena u originalnoj klasi.

Naš sistem bismo mogli nadograditi tako da dermatologu i kozmetičaru omogućimo prikaz sortiranih produkata, po cijeni i ocjeni.

1. **DECORATOR PATTERN**

Decorator patern služi za omogućavanje različitih nadogradnji objektima koji imaju istu osnovu. Umjesto definisanje velikog broja izvedenih klasa, dovoljno je omogućiti različito dekoriranje objekata, pa se tako pojednostavljuje i rukovanje objektima klijentima i samo implementiranje modela objekata. Decorator patern se koristi u slučajevima kada želimo dodijeliti dodatna ponašanja objektu tokom izvođenja programa bez da mijenjamo kod koji je na neki način u interakciji sa datim objektom.

U našem sistemu, korisnik ima atribut slika, pa bismo mogli omogućiti manipulacije istom, npr. rotacija, rezanje itd.

1. **BRIDGE PATTERN**

Osnovna namjena Bridge paterna je da omogući odvajanje apstrakcije i implementacije neke klase tako da ta klasa može posjedovati više različitih apstrakcija i više različitih implementacija za pojedine apstrakcije. Moguće je implementirati i sistem za razmjenu poruka primjenom Bridge paterna.

U našem sistemu mogli bismo primjeniti ovaj patern za razmjenu poruka između dermatologa i kozmetičara, tj. zahtjev za promjenu i odgovor da li je odobren ili odbijen.

1. **PROXY PATTERN**

Namjena Proxy paterna je da omogući pristup i kontrolu pristupa stvarnim objektima. Proxy je obično mali javni surogat objekat koji predstavlja kompleksni objekat čija aktivizacija se postiže na osnovu postavljenih pravila. Proxy patern rješava probleme kada se objekt ne može instancirati direktno (npr. zbog restrikcije pristupa).

U našem projektu možemo ga koristiti za klasu Admin i Dermatolog jer oni imaju pristup bazi podataka. Admin ima mogućnost brisanja korisnika, dok dermatolog uređuje/dodaje produkte, znaci da ne može svako imati pristup tim kalsama npr. Korisnik sa računom.

1. **COMPOSITE PATTERN**

Osnovna namjena kompozitnog paterna je omogućavanje formiranja strukture stabla pomoću klasa, u kojoj se kompozicije individualnih objekata (korijeni stabla) i individualni objekti (listovi stabla) ravnopravno tretiraju, odnosno moguće je pozvati zajedničku metodu nad svim klasama.

Omogućava nam uređenu hijerarhijsku strukturu koja zadržava uniformnost tako što je moguće istu metodu primjeniti nad različitim implementacijama.

1. **FLYWEIGHT PATTERN**

Primarna funkcija Flyweight paterna je da dozvoli da više različitih objekata dijele isto glavno stanje, a da im sporedna stanja budu različita.

Koristimo ga jer pomoću njega postižemo manju i racionalniju upotrebu resursa kao što je radna memorija i znatno brže izvršavanje programa.

1. **FAÇADE PATTERN**

Uloga facade paterna je osiguravanje više pogleda visokog nivoa na podsisteme čija je implementacija skrivena od korisnika.