STRUKTURALNI PATERNI U SISTEMU

ZA UPRAVLJANJE APOTEKARSKIM PROIZVODIMA

UVOD

Radi unapređenja postojećeg sistema za upravljanje apotekarskim proizvodima, analizirani su i primijenjeni strukturalni dizajn paterni. Njihova uloga je da obezbijede veću modularnost, proširivost i jednostavnost održavanja sistema. U nastavku se opisuju relevantni strukturalni paterni, sa posebnim fokusom na dva implementirana u klasnom dijagramu.

1) **Adapter Pattern**
   Adapter patern omogućava integraciju klasa koje imaju nespojive interfejse. U našem sistemu koristi se za povezivanje korisnika sa različitim načinima plaćanja (npr. kreditna kartica, PayPal) bez izmjene postojećeg koda. Korištenjem adaptera, klasa Korisnik ne zavisi od konkretne implementacije naplate, već koristi zajednički interfejs INaplata.
2) **Composite Pattern**
   Composite patern omogućava kreiranje hijerarhija u kojima se pojedinačni proizvodi i grupe proizvoda tretiraju na isti način. Uveden je interfejs IProizvod, koji implementiraju klase JedinstveniProizvod i ProizvodGrupa. Na taj način, moguće je raditi sa složenim paketima proizvoda kao da se radi o jednom proizvodu.
3) **Facade Pattern (nije implementiran)**
   Facade patern omogućava korisnicima interakciju sa kompleksnim podsistemima kroz pojednostavljen interfejs. U okviru apotekarskog sistema, ovaj patern bi se mogao koristiti za upravljanje aktivnostima administratora, kao što su upravljanje nalozima, narudžbama i fakturama, omogućujući objedinjeni pristup svim tim funkcijama.
4) **Decorator Pattern (nije implementiran)**
   Decorator patern omogućava dinamičko dodavanje funkcionalnosti objektima bez izmjene njihovog izvornog koda. U sistemu apoteke, ovaj patern bi se mogao koristiti za prikaz dodatnih informacija o proizvodima (npr. popusti, promotivne oznake), proširujući osnovnu klasu Proizvod sa dekoratorima.
5) **Bridge Pattern (nije implementiran)**
   Bridge patern odvaja apstrakciju od implementacije. U našem slučaju, mogao bi se koristiti za odvajanje načina prikaza proizvoda (osnovni vs. detaljni prikaz) od

logike pretrage i upravljanja proizvodima, čime se omogućava veća fleksibilnost kod prilagođavanja korisničkih interfejsa.

6) **Proxy Pattern (nije implementiran)**

Proxy patern upravlja pristupom stvarnim objektima. U sistemu za apoteke, Proxy bi mogao biti koristan kod kontrole pristupa osjetljivim operacijama, kao što su uređivanje podataka o proizvodima ili narudžbama, gdje bi proxy objekat provjeravao ovlaštenja korisnika prije delegiranja stvarnoj operacij

ZAKLJUČAK

Primjena strukturalnih paterna u ovom sistemu omogućila je veću fleksibilnost i skalabilnost:
Adapter omogućava laku integraciju više metoda plaćanja bez izmjena postojećih klasa.
Composite omogućava rad sa složenim strukturama proizvoda, bez dodatnog opterećenja u klijentskom kodu.
Ovakav pristup osigurava dugoročnu održivost sistema i olakšava razvoj dodatnih funkcionalnosti u budućnosti.