Strukturalni paterni

1. Uvod

Naš postojeći sistem obuhvata entitete poput korisnika, termina, medicinskih usluga i nalaza, ali kako bismo postigli veću fleksibilnost i ponovnu iskoristivost koda, poboljšali modularnost, smanjili međuzavisnost klasa, uvodimo dva strukturalna paterna:

- Facade - koristi se kada sistem ima više podsistema (subsystems) pri čemu su apstrakcije i implementacije podsistema usko povezane
- Decorator - omogućava dinamičko dodavanje novih elemenata i ponašanja (funkcionalnosti) postojećim objektima

2. Potreba za strukturalnim paternima

Dizajn softverskih sistema u domenu zdravstva često podrazumijeva veliki broj povezanih klasa koje zajedno implementiraju poslovne procese kao što su zakazivanje termina, generisanje nalaza, upravljanje korisnicima i uslugama. S obzirom na to da zdravstveni informacioni sistem mora biti modularan, fleksibilan i lako proširiv, uočena su dva ključna nedostatka trenutnog rješenja:

a) Problem kompleksnosti

U našem postojećem sistemu, klase kao što su Termin, MedicinskeUsluge, Korisnik i druge su direktno povezane i zavise jedna od druge.

Na primjer:

Klasa Termin direktno koristi atribute i metode iz Korisnik, Lokacija i MedicinskeUsluge.

Za zakazivanje jednog termina klijent mora upravljati nizom klasa i znati tačan redoslijed interakcija.

Ovakav dizajn uzrokuje:

- Teško održavanje i testiranje sistema.
- Veliku šansu za greške prilikom izmjena.
- Visoku međuzavisnost između klasa.

Rješenje – **Facade patern**:
Uvođenjem Facade paterna kreira se jedinstveni interfejs (ZakazivanjeFacade) koji predstavlja složenu logiku zakazivanja termina. Klijenti sistema sada koriste samo

zakaziTermin(...) metodu bez potrebe za razumijevanjem svih unutrašnjih veza i logike. To pojednostavljuje korištenje i povećava modularnost.

b) Problem ponovne upotrebe i proširivosti

U stvarnim uslovima rada, često se javlja potreba da se medicinskom nalazu dodaju nove funkcionalnosti kao što su:

Potpis doktora

Komentar doktora

Prevod nalaza na drugi jezik

Tradicionalni pristup bi zahtijevao izmjene unutar HistorijaGenerisanihNalaza klase ili stvaranje brojnih podklasa za svaku novu funkcionalnost.

Ovo dovodi do:

- Širenja hijerarhije klasa
- Dupliranja koda

Rješenje – **Decorator patern**:
Implementacijom Decorator paterna uvodi se interfejs NalazComponent koji definiše zajedničku metodu generisi(). Zatim se koristi osnovna klasa OsnovniNalaz, a dodatne funkcionalnosti (potpis, komentar) se dodaju kroz dekoratore: NalazSaPotpisom, NalazSaKomentarom.

Ovi dekoratori se mogu kombinovati po potrebi bez promjene postojećeg koda. Time se omogućava:

- Proširivost bez izmjena postojećih klasa
- Dinamičko dodavanje funkcionalnosti
- Fleksibilno sastavljanje ponašanja nalaza

3. Primjena paterna

3.1 Facade Pattern

- Lokacija primjene:

Kombinacija više podsistema za zakazivanje termina (Korisnik, Termin, MedicinskeUsluge, Lokacija) u jedinstven interfejs: ZakazivanjeFacade.

- Klase:

ZakazivanjeFacade: pruža jednostavne metode poput zakaziTermin() i otkaziTermin().

Interno koristi klase Termin, MedicinskeUsluge, Korisnik, Lokacija.

- Prednosti:

Klijent ne mora poznavati kompleksnost zakazivanja termina.

Pojednostavljuje interakciju sa sistemom.

3.2 Decorator Pattern

- Lokacija primjene:

Dodavanje dodatnih funkcionalnosti rezultatima nalaza (HistorijaGenerisanihNalaza)

npr. dodavanje digitalnog potpisa, dodatnih komentara itd.

- Klase:

NalazComponent (interfejs)

OsnovniNalaz (implementira NalazComponent)

NalazSaPotpisom, NalazSaKomentarom

- Prednosti:

Fleksibilno dodavanje novih ponašanja bez mijenjanja postojeće logike.

Poštuje Open/Closed princip.

4. Zaključak

Primjena strukturalnih paterna omogućava:

- Redukciju kompleksnosti i bolju organizaciju sistema (Facade)
- Modularno i fleksibilno proširivanje funkcionalnosti (Decorator)

Ovakav pristup povećava kvalitet dizajna, smanjuje tehnički dug i čini sistem spremnim za buduće zahtjeve bez potrebe za velikim izmjenama postojećeg koda.

Uvođenjem Facade i Decorator paterna sistem postaje modularniji i lakši za održavanje. Ovo je posebno važno za našu web stranicu zdravstva gdje su česte izmjene zahtjeva i

funkcionalnosti. Na ovaj način krajnji korisnici (doktori, administratori i pacijenti) ostvaruju brži, sigurniji i intuitivniji pristup funkcionalnostima sistema.