Strukturalni paterni

Uvod

U cilju smanjenja kompleksnosti, povećanja fleksibilnosti i ponovne iskoristivosti koda u zdravstvenom informacionom sistemu, u **projektu smo koristili dva ključna paterna**:

- **Facade** – za objedinjavanje kompleksne logike zakazivanja termina.
- **Decorator** – za dinamičko proširenje funkcionalnosti nalaza bez izmjene postojećih klasa.

1. **Facade Pattern**

Opis: Pojednostavljuje korištenje više povezanih klasa preko jednog interfejsa.

Primjena u projektu: Zakazivanje termina kroz klasu ZakazivanjeFacade koja interno koristi Termin, Korisnik, Lokacija, MedicinskeUsluge.

Primjer: zakaziTermin(korisnik, usluga, vrijeme) – klijent ne mora znati kako klase međusobno komuniciraju.

Prednosti: Jednostavno korištenje, manje grešaka, bolja modularnost.

2. **Decorator Pattern**

- **Opis**: Dodaje nove funkcionalnosti objektima bez mijenjanja njihove osnovne strukture.
- **Primjena u projektu**: Dodavanje potpisa, komentara i prevoda nalazima.
- **Primjer**:
  - OsnovniNalaz – osnovni rezultat.
  - NalazSaPotpisom, NalazSaKomentarom – dodatne funkcionalnosti kao dekoratori.
- **Prednosti**: Poštuje princip otvoreno/zatvoreno, olakšava proširenje bez izmjena postojećeg koda.

3. **Adapter Pattern**

- **Opis**: Omogućava povezivanje nekompatibilnih interfejsa.
- **Primjena**: Spajanje OAuth autentifikacije sa klasom Korisnik preko AuthAdapter.
- **Prednosti**: Integracija bez mijenjanja postojećeg sistema.

4. Bridge Pattern

- **Opis**: Razdvaja apstrakciju i implementaciju.
- **Primjena**: Prikaz nalaza u više formata (PDF, HTML).
- **Prednosti**: Veća fleksibilnost, olakšano održavanje.

5. Proxy Pattern

- **Opis**: Kontrola pristupa stvarnim objektima.
- **Primjena**: NalazProxy provjerava prava korisnika prije prikaza nalaza.
- **Prednosti**: Povećava sigurnost bez izmjene osnovnog objekta.

6. Composite Pattern

Opis: Tretira pojedinačne i složene objekte isto.

Primjena: Kreiranje paketa usluga (npr. Sistematski pregled = EKG + ultrazvuk…).

Prednosti: Jedinstveno upravljanje složenim strukturama.

7. Flyweight Pattern

Opis: Dijeli zajedničke objekte kako bi se smanjila memorijska potrošnja.

Primjena: LokacijaFactory vraća iste instance za iste adrese.

Prednosti: Optimizacija memorije kod velikog broja ponavljanja.

Zaključak

Primjena strukturalnih paterna poboljšava dizajn softverskog sistema:

Facade i Decorator (korišteni u projektu): Povećavaju modularnost i proširivost.