Kreacijski paterni

1. Uvod

Naš sistem Mediplan obuhvata osnovne entitete kao što su korisnici, termini, medicinske usluge i nalazi, što omogućava osnovno funkcionisanje klinike. Ali kako bi naš sistem bio skalabilniji, lakši za održavanje i pripremljen za moguće izmjene, postoji potreba za dodatnim unapređenjima na nivou konstrukcije objekata. Zbog toga uvodimo **kreacijske paterne**, koji nam omogućavaju kontrolisano, fleksibilno i modularno kreiranje instanci objekata.

Oni su posebno korisni u sistemima kao što je Mediplan, gdje se veliki broj objekata kreira dinamički – poput termina, nalaza, paketa usluga i korisničkih naloga. Korištenjem kreacijskih paterna dobijamo veću fleksibilnost, bolju ponovnu iskoristivost koda, smanjenje međuzavisnosti između komponenti i jednostavnije testiranje pojedinačnih dijelova sistema.

U nastavku ćemo opisati slijedeće kreacijske paterne:

- **Singleton** – za jedinstvene resurse u sistemu
- **Prototype** – za brzo kloniranje postojećih šablona
- **Factory Method** – za dinamičko kreiranje objekata prema tipu
- **Abstract Factory** – za stvaranje povezanih grupa objekata
- **Builder** – za postepeno kreiranje kompleksnih objekata

2. Potreba za dodavanjem kreacijskih paterna

Kreacijski paterni omogućavaju odvajanje logike stvaranja objekata od njihovog korištenja. Na taj način se postiže bolja modularnost sistema, izbjegava dupliciranje koda, olakšava održavanje i omogućava lakše dodavanje novih tipova objekata bez izmjena postojećeg koda.

2.1. Singleton

Ograničava klasu na jednu jedinu instancu i omogućava globalni pristup toj instanci.

Primjena:

- DatabaseConnectionPool – Centralna tačka za pristup bazi podataka. Umjesto da svaka komponenta pravi vlastitu konekciju, koristi se Singleton koji održava jednu konekciju (ili pool konekcija). To smanjuje opterećenje i omogućava kontrolu pristupa bazi.

- EmailGateway – Klasa zadužena za slanje mailova (npr. potvrde termina, rezultati nalaza). Ima konfiguraciju SMTP servera koja treba da bude jedinstvena i dostupna globalno.

- TerminScheduler – Jedan centralni scheduler kontroliše sve zakazane termine i sprječava kolizije (npr. da dva pacijenta ne budu zakazani kod istog doktora u isto vrijeme).

Prednosti: Jedinstveni resursi se ne dupliciraju, čime se smanjuje rizik od grešaka i povećava efikasnost sistema.

2. Prototype

Omogućava kreiranje novih objekata kloniranjem postojećih, što je korisno kada je kreiranje složenih objekata skupo ili se često ponavlja isti obrazac.

Primjena:

- TerminTemplate – Doktori i recepcioneri često koriste iste vremenske slotove i tipove usluga za više pacijenata. Kreiranjem jednog šablona termina (npr. "Kardiološki pregled – 30 minuta, EKG + konsultacije"), on se može kopirati i samo promijeniti datum ili pacijent.

- PaketUslugaTemplate – Sistematski pregledi često uključuju isti set laboratorijskih analiza i dijagnostičkih usluga. Umjesto da se svaki put ponovo definiše lista usluga, koristi se prototip koji se klonira i prilagodi po potrebi.

Prednosti: Smanjuje se potreba za ponovnim unosom istih podataka i omogućava brže kreiranje kompleksnih objekata.

3. Factory Method

Definiše interfejs za kreiranje objekta, ali podklase odlučuju koji objekat će biti kreiran.

Primjena:

- KorisnikFactory

  Apstraktna fabrika definiše interfejs kreirajKorisnika(...)

  Konkrete implementacije: PacijentFactory, DoktorFactory, AdministratorFactory

  Kada novi korisnik pristupi sistemu ili se registruje, na osnovu tipa se poziva odgovarajuća fabrika da kreira instancu odgovarajuće klase bez potrebe za if-else granama.

- UslugaFactory

Kreira odgovarajuće instance usluga: LaboratorijskaUsluga, SpecijalistickaUsluga, DijagnostickaUsluga, itd.

Omogućava dinamičko proširenje bez izmjene postojećeg koda.

Prednosti: Eliminacija if-else i switch logike, lakše dodavanje novih tipova korisnika i usluga bez izmjene postojećeg koda. Omogućava fleksibilno kreiranje objekata bez poznavanja konkretne implementacije – olakšava održavanje i testiranje.

4. Abstract Factory

Omogućava kreiranje "porodice povezanih objekata" bez potrebe za navođenjem njihovih konkretnih klasa.

Primjena:

- KlinikaFactory (interfejs)

  Definiše metode: kreirajTermin(), kreirajNalaz(), kreirajUsluge()

- Implementacije:
  - KardiologijaFactory – kreira termine kod kardiologa, EKG uslugu i šablon kardiološkog nalaza.
  - DermatologijaFactory – kreira termin, dermatološke usluge i nalaz s dermatoskopskim opisom.
  - LaboratorijaFactory – kreira termin, set laboratorijskih analiza i obrazac nalaza.

Prednosti: Omogućava da se cijeli paket funkcionalnosti zamijeni promjenom fabrike (npr. nova klinika ili odjel), bez potrebe da se mijenja klijentski kod.

5. Builder

Patern koji omogućava postepeno, fleksibilno kreiranje složenih objekata pomoću više metoda umjesto jednog velikog konstruktora.

Primjena:

TerminBuilder

- Metode:
  - zaPacijenta(pacijentId)
  - kodDoktora(doktorId)
  - naDatum(datum)
  - saUslugama(listaUsluga)

    - build(): Termin
- Koristi se kad korisnik (pacijent ili recepcija) unosi djelimične informacije i kasnije ih dopunjava. Omogućava fleksibilnost u formama i API pozivima.

NalazBuilder

- Sekvencijalno dodaje sekcije nalaza:
    - dodajLaboratorijskeNalaze(...)
    - dodajZakljucak(...)
    - build(): Nalaz
- Prikladno za različite tipove doktora koji imaju različite nalaze.

Prednosti: Poboljšana fleksibilnost i čitljivost koda, lakše testiranje i održavanje.

3. Zaključak

Uvođenjem kreacijskih paterna u naš sistem Mediplan postiže se značajno poboljšanje fleksibilnosti, modularnosti i održivosti arhitekture. Svaki patern ima jasno definisanu ulogu:

- Singleton uvodi jedinstvene tačke kontrole nad ključnim resursima,
- Prototype omogućava brzo i efikasno kloniranje šablona,
- Factory Method i Abstract Factory obezbjeđuju dinamičko i organizovano kreiranje objekata različitih tipova,
- Builder pojednostavljuje gradnju kompleksnih entiteta kroz fleksibilan i čitljiv pristup.

Upotrebom ovih paterna, sistem Mediplan postaje skalabilan, pregledan i spreman za buduće faze razvoja i proširenja, čime se olakšava i rad programera i korisnika.