Paterni Ponašanja

1. Observer (Promatrač)

Observer pattern omogućava obavještavanje donora krvi o potrebama klinika za određenim tipovima krvi. Kada zaliha krvi padne ispod kritične linije, sistem (subjekat) šalje obavijest svim donorima koji su promatrači (observers) i pripadaju odgovarajućoj krvnoj grupi. Donori tada mogu odlučiti da li žele donirati krv. Prednosti ovog pristupa uključuju rasterećenje sistema od direktnog kontaktiranja svakog donora i omogućavanje donorima da samostalno odluče kako će reagirati na obavijesti. Ovo također povećava fleksibilnost i modularnost sistema jer se novi promatrači mogu dodavati ili uklanjati bez utjecaja na ostatak sistema.

2. Strategy (Strategija)

Strategy pattern može se koristiti za definisanje različitih strategija obrade podataka i interakcije s korisnicima. Na primjer, za izračunavanje kritične linije zaliha krvi možemo imati različite strategije: jednu koja se bazira na historijskim podacima o potrošnji krvi, drugu koja koristi trenutne statističke modele i treću koja se oslanja na sezonske varijacije. Svaka strategija će biti implementirana kao zasebna klasa koja implementira zajednički interfejs za izračunavanje kritične linije. Ovo omogućava lako mijenjanje strategija bez potrebe za izmjenom osnovne logike aplikacije.

3. Iterator (Iterator)

Iterator pattern omogućava korisnicima prolazak kroz listu donora, zaliha krvi ili rezerviranih termina bez otkrivanja unutrašnje strukture tih kolekcija. Na primjer, donori mogu koristiti iterator za pregled svojih prošlih donacija, dok zaposlenici mogu koristiti iterator za prolazak kroz listu trenutnih zaliha krvi ili predstojećih termina za donaciju. Iterator omogućava standardiziran način prolaska kroz podatke, čineći aplikaciju fleksibilnijom i lakšom za održavanje.

4. Memento (Memento)

Memento pattern se može koristiti za snimanje trenutnog stanja profila donora ili stanja zaliha krvi i vraćanje tog stanja u budućnosti. Na primjer, prilikom promjene korisničkih podataka, možemo snimiti trenutno stanje korisničkog profila kao memento objekat. Ako korisnik kasnije poželi vratiti prethodno stanje, sistem može koristiti memento objekat za vraćanje podataka. Ovo omogućava korisnicima sigurnost da mogu vratiti prethodno stanje u slučaju greške ili neželjenih promjena.

5. Chain of Responsibility (Lanac odgovornosti)

Chain of Responsibility patern može se koristiti za validaciju podataka prilikom registracije donora ili prilikom obrade zahtjeva za donaciju. Svaki korak u procesu validacije (npr. provjera ispravnosti formata broja telefona, provjera kompatibilnosti krvne grupe s trenutnim potrebama, provjera ispravnosti unesenih osobnih podataka) može biti zaseban handler u lancu odgovornosti. Ako jedan handler ne može obraditi zahtjev, on ga prosljeđuje sljedećem handleru u lancu. Ovo omogućava modularnu i fleksibilnu obradu zahtjeva.

6. Mediator (Posrednik)

Mediator pattern može se koristiti za olakšavanje komunikacije između različitih komponenti sistema, kao što su modul za registraciju donora, modul za upravljanje zalihama krvi i modul za slanje notifikacija. Posrednik upravlja interakcijama između ovih komponenti, smanjujući direktne zavisnosti među njima. Na primjer, kada se novi donor registruje, modul za registraciju obavještava posrednika, koji zatim obavještava modul za slanje notifikacija i modul za upravljanje zalihama kako bi ažurirali svoje podatke i poslali relevantne obavijesti.

7. Template Method (Template metoda)

Template Method pattern može se koristiti za definiranje koraka u procesu doniranja krvi, gdje se zajednička struktura algoritma nalazi u apstraktnoj klasi, a specifični koraci se implementiraju u podklasama. Na primjer, proces rezervacije termina za donaciju krvi može imati sljedeće korake: izbor termina, unos podataka, potvrda rezervacije i slanje obavijesti. Apstraktna klasa može definirati template metodu RezervacijaTermina(), dok će specifične metode kao što su IzborTermina(), UnosPodataka() i SlanjeObavijesti() biti implementirane u podklasama. Ovo omogućava lako prilagođavanje i proširenje procesa rezervacije bez mijenjanja osnovne strukture algoritma.

Navedeni Paterni Ponašanja omogućavaju efikasno i modularno upravljanje funkcionalnostima unutar sistema "VitalFlow", poboljšavajući njegovu fleksibilnost, održivost i korisničko iskustvo.

8. Interpreter Pattern (Tumač)

Interpreter pattern se može koristiti za tumačenje ili analiziranje jezika specifičnog za domenu unutar aplikacije. U slučaju aplikacije VitalFlow, ovaj patern bi se mogao koristiti za tumačenje i izvršavanje složenih pravila ili upita koji se odnose na upravljanje zalihama krvi, filtriranje donacija, ili generisanje izvještaja. Na primjer, možemo imati jednostavan jezik koji omogućava administratorima da definišu pravila za filtriranje donacija krvi na osnovu različitih kriterija kao što su krvna grupa, datum donacije, ili status obrade donacije. Ovaj jezik može imati sintaksu kao što je: "krvna_grupa = 'A+' AND datum > '2023-01-01' AND status = 'prihvaćeno'".

9. Command (Komanda)

Command pattern je pattern ponašanja koji pretvara zahtjeve ili jednostavne operacije u objekte. Ovaj pattern omogućava parametrizaciju klijenata sa različitim zahtjevima, redoslijedima ili zapisima operacija i podršku za operacije poništavanja. U kontekstu aplikacije VitalFlow, Command pattern se može koristiti za implementaciju operacija kao što su registracija donora, rezervacija termina za donaciju, ili obavještavanje korisnika. Na primjer, svaka operacija može biti enkapsulirana kao objekat komande, koji sadrži sve potrebne informacije za izvršenje te operacije.

10. Visitor (Posjetioc)

Visitor pattern je pattern ponašanja koji omogućava dodavanje novih operacija na objekte bez promjene tih objekata. To se postiže definisanjem operacija u zasebnoj klasi koja se naziva Visitor (posjetioc). Objekti koje treba posjetiti (elementi) prihvataju posjetioca i omogućavaju mu da izvrši operacije na njima. U kontekstu aplikacije VitalFlow, Visitor pattern se može koristiti za izvršavanje različitih operacija nad podacima donora krvi bez promjene klasa donora. Na primjer, možemo imati različite posjetioce za operacije kao što su generisanje izveštaja, verifikacija podataka ili ažuriranje statistika.

11. State (Stanje)

State pattern je pattern ponašanja koji omogućava objektu da promijeni svoje ponašanje kada mu se promijeni unutrašnje stanje. Objekt će izgledati kao da je promijenio svoju klasu. Ovo je korisno kada objekt treba da se ponaša različito u zavisnosti od svog stanja. U kontekstu aplikacije VitalFlow, State pattern se može koristiti za upravljanje različitim stanjima donacija krvi, kao što su "primljena", "obrađena", "odobrena" i "odbijena". Svako od ovih stanja može imati različito ponašanje i reakcije na događaje.