KREACIJSKI PATERNI

U kreacijske paterne spadaju : Singleton, Prototype, Factory Method, Abstract Factory i Builder.

- SINGLETON PATERN

Uloga Singleton paterna je da osigura da se klasa može instancirati samo jednom i da osigura globalni pristup kreiranoj instanci klase.Ovaj patern smo iskoristili u klasi Zubex gdje smo dodali privatni static konstruktor, static read.only objekat koji se instancira korištenjem privatnog konstruktora i metodu koja pristupa privatnom objektu i instancira ga ako objekat ne postoji-static getInstance().

- PROTOTYPE PATERN

Uloga Prototype paterna je da kreira nove objekte klonirajući jednu od postojećih prototip instanci (postojeći objekat). U našem slučaju moguće je kreirati više pregleda sa istom dijagnozom, u isto vrijeme. Nakon što se kreiraju ti pregledi moguće je napraviti minorne promjene u izgledu svakog pojedinačnog pregleda, kao što su naziv doktora i pacijenta.

- FACTORY METHOD PATERN

Uloga Factory Method paterna je da omogući kreiranje objekata na način da podklase odluče koju klasu instancirati. Različite podklase mogu na različite načine implementirati interfejs. Factory Method instancira odgovarajuću podklasu(izvedenu klasu) preko posebne metode na osnovu informacije od strane klijenta ili na osnovu tekućeg stanja. Ovaj patern se u našem diajgramu mogao primijeniti kod klase Zaposlenik, ona sadrži metode koje nasljeđuje svaka naslijeđena klasa, samo što bi u klasu Zaposlenik trebalo dodati factory metodu, kreirati interfejs IKorisnikFactory, koji će zamijeniti metode za kreiranje pojedinačnog uposlenika, a taj interfejs naslijediti klase Doktor, Direktor, MedSestra.

- ABSTRACT FACTORY PATERN

Abstract Factory patern omogućava da se kreiraju familije povezanih objekata/produkata.Na osnovu apstraktne familije produkata kreiraju se konkretne fabrike produkata različitih tipva i različitih kombinacija.Patern odvaja definiciju(klase) produkta od klijenta.Zbog toga se familije produkata mogu jednostavno izmjenjivati ili ažurirati bez narzušavanja strukture klijenta.Ovaj patern bi se mogao iskoristiti u klasi Zaposlenik, jer se iz zaposlenika stvaraju klase Doktor,Medicinska Sestra i Direktor.

- BUILDER PATERN
  Uloga Builder paterna je odvajanje specifikacije kompleksnih objekata od njihove stvarne konstrukcije. Isti konstrukcijski proces može kreirati različite reprezentacije. Builder pattern smo iskoristili ovdje da bi napravili interfejs Builder filtriranje dvije klase koje smo spojili s pregledom i to ce nam omoguciti da dobijemo spisak svih pregleda i

spisak pregleda koje ce biti filtrirane po nekom kriteriju koji mi zadamo i to se na kraju spoji sa Zubex gdje se zadaju pregledi.

STRUKTURNI PATERNI

U strukturne paterne spadaju : Adapter, Facade, Decorator, Bridge, Composite, Proxy i Flyweight.

- ADAPTER PATERN

Osnovna namjena Adapter paterna je da omogući širu upotrebu već postojećih klasa.U situacijama kada je potreban drugačiji interfejs već postojeće klase, a ne želimo mijenjati ppstojeću klasu koristi se Adapter patern.Adapter patern krera novu adapter klasu koja služi kao posrednik između orginalne klase i željenog interfejsa.Tim postupkom se dobija željena funkcionalnost bez izmjena na orginalnoj klasi i bez ugrožavanja integriteta cijele aplikacija.Ovaj patern bi se mogao iskorititi ukoliko bismo mi imali mogućnost nekog plaćanja gdje bi pomoću njega omogućili pretvaranje para u određene valute, tj. imali bi interfejs u kojem postoji metoda za konverziju para i klasu Adapter sa istom metodom.

- FACADE PATERN

Facade patern se koristi kada sistem ima više identificiranih podsistema (subsystems) pri čemu su apstrakcije i implementacije podsistema usko povezane. Osnovna namjena Facade paterna je da osigura više pogleda visokog nivoa na podsisteme (implementacija podsistema skrivena od korisnika). Ovaj patern se u našem dijagramu može povezati sa klasom Zaposlenik, jer je to apstraktna klasa za više klasa. Svaka klasa koja naslijeđuje klasu Zaposlenik je specifična na svoj način, a sve posjeduju nekoliko zajedničkih atributa. Mislimo da je ovaj patern implementiran pomoći klase Zubex, na način da u sebi sadrži druge klase kao atribute. Pozivom jedne metode iz ove klase pozivamo više različitih metoda iz različitih klasa, te se time izvršava kompleksniji proces u koji korisnik nema uvid.

- DECORATOR PATERN

Osnovna namjena Decorator paterna je da omogući dinamičko dodavanje novih elemenata i ponašanje(funkcionalnost) postojećim objektima.Objekat pri tome ne zna da je urađena dekoracija što je veoma korisno za ponovnu upotrebu komponenti softverskog sistema.Ovaj patern se možda mogao iskoristiti ukoliko bi npr. postojali neki kuponi i na osnovu njih bi cijene pojedinih usluga bile drugačije.Tada bi trebalo napraviti interfejs sa metodom koja računa novu cijenu i nove klase npr klasa za računanje cijene usluge s kuponom(koja bi sadržala vrijednost popusta na određenu uslugu) i klasa za računanje cijene usluge bez kupona(klasa sa popustom koji je jedak 0).Interfejs se spaja sa klasom Usluga.

- BRIDGE PATERN

Osnovna namjena Bridge paterna je da omogući odvajanje apstrakcije i implementacije neke klase tako da ta klasa može posjedovati više različitih apstrakcija i više različitih implementacija za pojedine apstrakcije. Bridge patern pogodan je kada se implementira nova verzija softvera a postojeća mora ostati u funkciji. Mislimo da u našoj aplikaciji nema potrebe za Bridge paternom, odnosno nemamo mogućnost za njegovu primjenu.

- COMPOSITE PATERN
  Osnovna namjena Composite paterna (kompozitni patern) je da omogući formiranje strukture stabla pomoću klasa, u kojoj se individualni objekti (listovi stabla) i kompozicije individualnih objekata (korijeni stabla) jednako tretiraju.Composite patern se koristi kada imamo nekoliko klasa koje u susitini trebaju da imaju iste metode,ali u nasem primjeru nema smisla da postoji neka zajednicka nadklasa pa cemo umjesto npr. Klase Osoba napraviti interfejs Iosoba koja ima funkciju npr. Izracunaj platu i to cim naslijede zaposlenik i direktor odmah je moraju implementirati.
- PROXY PATERN

Namjena Proxy paterna je da omogući pristup i kontrolu stvarnim objektima.Proxy je obično mali javni surogat objekat koji predstavlja kompleksi objekat čija aktivizacija se postiže na osovu postavljenih pravila.Proxy patern rješava probleme kada se objekat ne može instancirati direktno.Ovaj patern je iskorišten kod klase Zaposlenik gdje smo omogućil pristup pregleda Osoblja ordinacije.Dodana je klasa Proxy u kojoj se nalazi metodu na osnovu koje se postavlja pristup na broj 1 ukoliko je direktor(jer samo on može dobiti pregled svih Doktora i Medicinskih Sestara), u ostalim slučajevima(Doktor ili Medicninska Sestra) pristup se postavlja na 0.

- FLYWEIGHT PATERN
  Postoje situacije u kojima je potrebno da se omogući razlikovanje dijela klase koji je uvijek isti za sve određene objekte te klase (tzv. glavno stanje (engl. intrinsic state)) od dijela klase koji nije uvijek isti za sve određene objekte te klase (tzv. sporedno stanje (engl. extrinsic state)). Osnovna namjena Flyweight paterna je upravo da se omogući da više različitih objekata dijele isto glavno stanje, a imaju različito sporedno stanje .Mislimo da u našoj aplikaciji nema potrebe za Flyweight paternom, odnosno nemamo mogućnost za njegovu primjenu.