PATERNI PONAŠANJA

Strategy patern:

Ukoliko bi se javila potreba da sortiramo naše usluge, to bismo mogli uraditi na više načina tj. sa različitim algoritmima za sortiranje (QuickSort, MergeSort, ShellSort). Uočavamo da za to trebamo koristiti Strategy patern, jer on omogućava klijentu (doktoru) izbor jednog od algoritama iz familije algoritama za sortiranje stomatoloških usluga.

State patern:

Ukoliko bismo imali dva tipa stomatoloških kartona – otvoren i zatvoren, tada bismo imali dva stanja. Ta dva stanja su : stomatološki karton je otvoren i stomatološki karton je zatvoren. Kod State paterna se način ponašanja objekta mijenja u zavisnosti od trenutnog stanja objekta. Ukoliko je karton u otvorenom stanju, omogućene su nam metode koje već imamo, dok ukoliko je u zatvorenom stanju tada ne bismo mogli vršiti promjene tj. pozivati metode koje mijenjaju podatke koji se nalaze u kartonu. Objekat (Stomatološki karton) bi sam mijenjao stanje zavisno od interakcije klijenta sa kontekstom.

TemplateMethod patern:

U našoj aplikaciji trenutno nemamo sortiranje termina. Ukoliko bi to bilo potrebno izvesti, mogli bismo sortirati termine ili prema ID-u termina ili prema vremenu termina. Tako vidimo da zapravo možemo da poredimo dva termina prema ta dva atributa. Uočavamo da je najbolje koristiti TemplateMethod patern jer se kod njega može poređenje izdvojiti u drugu klasu jer se ono može razlikovati po varijabli po kojoj se poredi, upravo ono što nama treba. Na taj način bismo mogli implementirati taj dio operacije sortiranja (poređenja) na dva različita načina.

Observer patern:

Prilikom registracije pacijenta na našu web aplikaciju, pacijent bi mogao da odabere opciju da bude obavještavan za sve termine koje je zakazao, time osiguravajući da neće zaboraviti na termin te da će sigurno doći na isti. Stanje koje bi moglo okidati ovaj događaj jeste vrijeme koje je preostalo do termina. Mogli bismo porediti trenutno vrijeme sa vremenom termina i staviti da obavještenja budu na 7 dana, 1 dan i sat vremena prije termina.

Iterator patern:

Ovaj patern bismo mogli iskoristiti za prolazak kroz sve stomatološke kartone. Time bismo izdvojili prolazak kroz kolekciju u poseban objekat koji zovemo iteratorom. Također, objekat enkapsulira sve detalje prolaska kroz kolekciju kao i trenutnu poziciju i broj elemenata do kraja kolekcije. Tako više iteratora može prolaziti kroz istu kolekciju istovremeno, neovisno jedni od drugih.

Chain of responsibility patern:

Kod metode za postavljanje RTG snimka vilice, postoji mnogo provjera koje bi mogle biti stavljene u posebne objekte – handlere i povezane u lanac. Prvo trebamo provjeriti da li je snimak ispravnog formata, ukoliko jeste onda dolazimo do drugog handlera. U njemu provjeravamo da li je ispravna rezolucija. Ukoliko je i to uredu, dolazimo do trećeg handlera koji provjerava da li zauzima previše memorije, odnosno da li je veličina fajla prevelika, te nakon toga dolazimo do četvrtog handlera koji provjerava da li je ime snimka u ispravnom formatu (imePacijenta_datumSnimanja). Pri svakom zadovoljavanju trenutnog handlera se zahtjev za postavljanjem snimka prenosi na sljedeći handler. Ukoliko nije zadovoljena jedna od provjera, onda se zaustavlja zahtjev za postavljanjem snimka te ne dolazi do daljnje provjere handlera koji se nalaze iza istog.

Mediator patern:

Ukoliko bismo željeli da smanjimo zavisnost između objekata onda koristimo ovaj patern. U našoj aplikaciji bismo mogli da smanjimo, odnosno ukinemo zavisnost između klase Doktor i klasa kojima on pristupa poput klasa Stomatološki Karton, Stomatološka Usluga, Adresar i Cjenovnik. Napravili bismo specijalni mediator objekat kroz koji bismo obavljali svu komunikaciju između Doktora i ostalih navedenih klasa. Time bismo postigli da klase Stomatološki Karton, Stomatološka Usluga, Adresar i Cjenovnik ovise samo od tog mediator objekta, a ne od klase Doktor.