Univerzitet u Sarajevu

Elektrotehnički Fakultet

Objektno Orijentisana Analiza i Dizajn

STRUKTURALNI PATERNI

Strukturalni paterni se bave kompozicijom i predstavljaju načine za definisanje odnosa među objektima. Strukturalni paterni daju mogućnost ukoliko je potrebna promjena u jednom dijelu sistema da se ne mora ostatak sistema mijenjati. Grupu strukturalnih paterna sačinjava sljedećih 7 paterna koji slijede u nastavku:

1. Adapter pattern

Osnovna namjena Adapter paterna je da omogući širu upotrebu već postojećih klasa. Koristi se u situacijama kada je potreban drugačiji interfejs već postojeće klase, a ne želimo mijenjati postojeću klasu. Njegova uloga je da kreira novu adapter klasu koja služi kao posrednik između originalne klase i interfejsa, te tako dobijamo željenu funkcionalnost bez izmjena na originalnoj klasi i bez ugrožavanja integriteta cijele aplikacije.

Moguca primjena Adapter paterna u sistemu:

Adapter patern bi u našem sistemu mogli iskoristiti za slike. Svaki alat ce imati svoju sliku.U sistemu trenutni tip atributa Slika je string, navedeni tip bi mogli zamijeniti uz pomoć metoda konverzije Adapter klase i omogućiti prihvatanje raznih tipova formata slika što neće ugroziti integritet cijele aplikacije.

2. Decorator pattern

Osnovna namjena Decorator paterna je da omogući dinamičko dodavanje novih elemenata i funkcionalnosti postojećim objektima. Objekat pri tome ne zna da je urađena dekoracija što je veoma korisno za ponovnu upotrebu komponenti softverskog sistema.

Moguca primjena Decorator paterna u sistemu:

Decorater patern bi u našem sistemu mogli iskoristiti također za slike. Time bi dodali mogućnost uređivanja slika kao što je promjena veličine, rezanje, rotacija, povezivanje, izoštravanje.. U sistem bi dodali interfejs ISlika koji bi imao metode za uređivanje slike. Pošto su slike jedan od najbitnijih komponenti nekog oglasa samim tim bi trebale biti i omogućene neke aktivnosti nad slikama kako bi se osigurao kvalitetan prikaz korisnicima sistema.

3. Proxy pattern (primijenjen u sistemu)

Osnovna namjena Proxy paterna je da omogući pristup i kontrolu pristupa stvarnim objektima. Omogućava se kontrola pristupa objektima, te se i onemogućava manipulacija objektima ukoliko neki uslov nije ispunjen ili ukoliko korisnik nema prava pristupa traženom objektu. Proxy patern rješava probleme kada se objekt ne može instancirati direktno (npr. zbog restrikcije pristupa).

Primjena Proxy paterna u sistemu:

Kako je Proxy pattern vezan za kontrolu pristupa, mi smo na našem dijagramu dodali klasu Proxy koja implementira interfejs IRestrictionOnRating. Zamislili smo da svaki registrovani korisnik ima opciju da ostavi ocjenu na alat, međutim to mogu uraditi samo oni korisnici koji su vec iznajmljivali taj alat, jer za druge nema smisla posto ga nisu koristili.Tako da ćemo putem metode pristup iz Proxy klase moći da zaključimo da li je registrovani korisnik koristio alat ili nije. Ukoliko nije koristio njemu neće biti dozvoljeno da ostavi ocjenu,a ako jeste koristio bice u mogucnosti da,ukoliko zeli,ostavi ocjenu.

4. Composite pattern(Primijenjen u sistemu)

Osnovna namjena Composite paterna je da omogući formiranje strukture stabla pomoću klasa, u kojoj se individualni objekti i kompozicije individualnih objekata jednako tretiraju. Koristi se za kreiranje hijerahije objekata, tačnije kada svi objekti imaju različite implementacije nekih metoda kojima je potrebno pristupati na isti način.

Primjena Composite paterna u sistemu:

Klase Tool I Rent nemaju mnogo toga zajednickog.Mozemo reci da su to dvije potpuno razlicite klase.Medjutim,za obje klase nam treba metoda koja ce nam omoguciti da uporedjujemo objekte ovih klasa na nacin koji nam je potreban.To ce biti metoda CompareTo.Imacemo interface Comparable sa metodom CompareTo,koja ce biti naslijedjena u klasama Tool I Rent,ali drugacije implementirana.Naknadno,ako bude potrebno,mozemo I nekim drugim klasama omoguciti da naslijede ovaj Comparable interface I da na njima specifican nacin implementiraju metodu.

5. Facade pattern

Facade patern se koristi kada sistem ima više identificiranih podsistema pri čemu su apstrakcije i implementacije podsistema usko povezane. Osnovna namjena Facade paterna je da osigura više pogleda visokog nivoa na podsisteme. Operacije koje su potrebne određenoj korisničkoj perspektivi mogu biti sastavljene od različitih dijelova podsistema. Može se više fasada postaviti oko postojećeg skupa podsistema i na taj način formirati više prilagođenih pogleda na sistem. Facade patern služi kako bi se klijentima pojednostavilo korištenje kompleksnih sistema. Klijenti vide samo krajnji izgled objekta, dok je njegova unutrašnja struktura skrivena. Na ovaj način se smanjuje mogućnost pojavljivanja grešaka jer klijenti ne moraju dobro poznavati sistem da bi ga koristili.

Moguca primjena Facade paterna u sistemu:

Ovaj pattern ne bismo mogli upotrijebiti u našem sistemu iz razloga što nemamo više podsistema koje bi na neki način mogli instancirati u nekoj novoj klasi Fasada koja bi imala neke smislene metode. Kada bismo imali tu klasu Fasada, korisnik bi mogao da koristi smislene metode samo primijenjene na

različitim primjercima nekih drugih klasa, recimo u nekom drugom sistemu bi imala metode IscrtajLiniju, IsctajKrug, ObojiLiniju, ObojiKrug i slično. Mi ne možemo da kreiramo takvu klasu u našem sistemu jer kada bismo je keirali imala bi skup nesmislenih metoda koje bi logički bolje išle u drugim klasama nego u jednoj zajedničkoj.

6. Flyweight pattern

Flyweight patern se koristi kako bi se onemogućilo bespotrebno stvaranje velikog broja instanci objekata koji svi u suštini predstavljaju jedan objekat. Osnovna namjena Flyweight paterna je upravo da se omogući da više različitih objekata dijele isto glavno stanje, a imaju različito sporedno stanje. Samo ukoliko postoji potreba za kreiranjem specifičnog objekta sa jedinstvenim karakteristikama, vrši se njegova instantacija, dok se u svim ostalim slučajevima koristi postojeća opća instance objekta.

Moguca primjena Flyweight patterna u sistemu:

Flywight pattern bismo mogli implementirati u sistemu da olaksamo korisniku da dodje do zeljenog tipa alata.I to na nacin da bismo koristili njegovu historiju rentanja(RentHistory) I izabrali dvije ili tri vrste alata koje je on najvise puta iznajmio I alate te vrste mu prikazali na pocetnoj stranici kao defaultne.

7. Bridge pattern

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 je pogodan kada se implementira nova verzija softvera, a postojeća mora ostati u funkciji.

Moguca primjena Bridge paterna u sistemu:

Ovaj pattern bismo mogli primijeniti za metodu kloja bi racunala poseban koeficijent koji bi sluzio za racunanje popusta.Preko interfacea IDiscountCoeff koji bi imao tu metodu,koja bi bila naslijedjena od strane klasa Tool I ToolType,pa bismo imali racunanje koeficijenta u odnosu na alat kao jedinku I alat ako tip.