Primijenjeni Strukturalni Paterni

Bridge Pattern

Primijenićemo ovaj patern između klase Rezervacije i interfejsa Rezervabilno kako bismo elegantnije mogli podržati neovisni razvoj i jedne i druge klase, jer bi nekad u budućnosti mogli očekivati da ćemo morati podržati rezervacije različitog tipa, da ne spominjem da je i čitava poenta Rezervabilnog interfejsa to da se podrži širok broj proizvoda u budućnosti.

Composite Pattern

U naš sistem ćemo uvesti mogućnost i da imamo ponudu ponuda (rezervabilni objekat koji agregira same ponude), tj. pakete ponuda koji bi mogli modelirati recimo turneje kroz niz gradova, gdje svaki grad ima svoj set isplaniranih usluga. U tom trenutku otvara se mogućnost kreiranja strukture podataka stablo. Kako bi smo elegantno mogli obrađivati istu (npr. obračunatu ukupnu cijenu) implementiraćemo patern Composite.

Preostali Strukturalni Paterni

Adapter

Postoji nekoliko situacija gdje bi ovaj patern mogao biti iskorišten i gotovo sve su usko vezane uz dodavanje facade paterna. Spomenućemo neke ideje za API-je, pa shodno tome za očekivati je da će se pojaviti potreba za formatiranjem određenih klasa u format prikladan i razumljiv datom API-ju, pa je stoga u tu svrhu mudro iskoristiti adapter patern.

Facade

Naša aplikacija sadrži određene komponente namijenjene za komunikaciju, autentifikaciju, finansijsku analizu – sve mogućnosti koje je moguće dodatno proširivati upotrebom vanjskih API-ja. Sada je vrlo vjerovatno da nama ne bi trebale te mogućnosti u svom punom opsegu pa je stoga facade patern logičan. Moguće bi bilo objediniti nekoliko različitih API-ja u jednu apstraktnu fasadu, a onda pružiti i nekoliko specifičnijih implementacija istog poštujući princip segregacije interfejsa.

Dekorator

Upotreba ovog paterna bi mogla poslužiti za ekstenziju dijela sistema vezanog za komunikaciju, uvodeći mogućnosti da se poruka ili obavijest obavi i izvan našeg sistema. Primjeri su da se npr. emituje mail pretplanicima mailing liste neke agencije kada ova odluči promovisati / izdvojiti neki proizvod. Možda čak i SMS obavijesti te neka integracija sa društvenim mrežama. Sve ove primjene bi mogle iziskivati upotrebu dekorator paterna nad klasom Poruka i odgovarajućeg dijela sistema.

Proxy

Operacije sa bazom podataka mogu biti dosta spore što je problem s aspekta skalabilnosti. Možda bi bilo efikasnije ukoliko bi smo takve operacije (npr. registraciju ili brisanje profila) vršili rjeđe a s više podataka nego često s relativno manje podataka. U tome bi nam mogao pomoći proxy patern koji bi

kontrolisao pristup bazi podataka, upravljao keširanjem potrebnih promjena, a zatim u odgovarajuće vrijeme keširane promjene u zasebnom threadu oslikao i na bazu podataka.

Flyweight

Pošto je u ovom paternu riječ o memorijskoj optimizaciji, otvara se mogućnost optimizacije segmenata sistema koji rade sa ponudama, korisnicima ili transakcija gdje bi se eventualno među njima dijeljeni podaci prepoznavali i keširali upotrebom flyweight paterna.