PATERNI PONAŠANJA

1. STRATEGY

Strategy patern služi za ukidanje velikog broja if-else blokova.

U našem sistemu smo ovaj patern implementirali kod filtriranja majstora, ukoliko želimo da se filtriranje može vršiti na različite načine, na primjer po profesiji majstora, recenzijama i cijenama usluga. U tu svrhu uvodimo interfejs IFiltrirajStrategy, i tri nove klase za pojedinačne vrste filtriranja.

2. STATE

State patern se koristi za izdvajanje koda koji je smješten u velike if-else blokove ovisno o vrijednosti neke varijable.

Nismo predvidjeli implementaciju ovog paterna, ali bi se on mogao iskoristiti tako što bi se provjeravalo stanje korisnika na aplikaciji 24 sata prije termina, kada treba da im stigne obavijest o dogovorenom terminu. Ukoliko su korisnik i majstor online, notifikacija će stići regularno preko aplikacije, a ako su offline, podsjetnik za termin bi im mogao stići putem maila. Imali bi novi interfejs IState koji određuje kako će obavijest biti poslana, i odvojene klase za online i offline stanje.

3. TEMPLATE METHOD

Ovaj patern omogućava promjenu samo dijela algoritma i korištenje preostalog dijela koda.

U našem sistemu ovaj patern bi mogli iskoristiti također kod slanja notifikacije za dogovoreni termin. Template notifikacije je isti i za majstora i za korisnika usluga, s tim da se neke informacije razlikuju. Majstor dodatno ima prikazano na koju adresu ide i da li je termin već uplaćen.

4. OBSERVER

Observer patern omogućava propagiranje promjena sa jednog objekta na druge objekte.

Ovaj patern smo implementirali u sistem, tako da korisnici, koji su zainteresovani za rad određenog majstora, dobiju obavijest kada se datom majstoru oslobodi neki termin. Dodali smo interfejs IObavijestObserver koji šalje obavijesti, kojeg bi implementovala naša klasa RegistrovaniKorisnik.

5. ITERATOR

Iterator patern omogućava prolazak kroz kolekciju objekata koja nije niz niti lista prema željenoj logici.

U našem sistemu bi se mogao primijeniti za prolazak kroz listu recenzija određenog majstora, na primjer prikaz recenzija od najboljih, najlošijih ili nekim slučajnim redoslijedom. Imali bi dva interfejsa, IKreatorIterator, koji bi kreirao iterator, i IIterator koji bi vraćao sljedeću recenziju, kao i klase RecenzijeOdNajboljih, RecenzijeOdNajlošijih, RecenzijeSlučajniIzbor.

6. CHAIN OF RESPONSIBILITY

Kod ovog paterna je važno definisati korake i redoslijed procesa. On služi za razdvajanje dijelova procesa tako da se mogu neovisno mijenjati.

Ovim paternom bi mogli predstaviti korake pri rezervaciji termina. Prvo korisnik odabere termin, nakon čega ga majstor odobri, zatim korisnik uplaćuje termin i dobija potvrdu o uplaćenom i rezervisanom terminu.

7. MEDIATOR

Mediator patern se koristi za smanjivanje direktnih veza između klasa i kreiranje centralizovanog sistema.

Ukoliko bi željeli, ovaj patern bi mogli primijeniti u svrhu provjere komentara koje korisnici pišu majstorima. Preko interfejsa IKomentarMediator vršila bi se provjera recenzije koju korisnik želi ostaviti, jer nije dozvoljeno pisati neprimjerene komentare i komentare uvredljivog sadržaja.