Paterni ponašanja

Merjem Bećirović

Bakir Bajrović

Stefani Kecman

1. Strategy patern

Zaposlenik u našem sistemu unosi podatke, nakon čega poziva algoritam za njihovo sortiranje te ažurirane podatke šalje na sajt. Svjesni smo da postoje različiti algoritmi sortiranja i da im se performanse mogu porediti u odnosu na specifičnosti strukture seta podataka koje sortiraju. Samim tim, omogučivši zaposleniku da bira između različitih algoritama sortiranja onaj koji je najpogodniji za način unosa podataka koji izvršava (to može biti unos pojedinačnih informacija, unos čitave tabele novih podataka i slično), mi dobivamo najbolje performanse za datu situaciju. Ovo je poželjno ako želimo stvoriti efekt real-time ažuriranja podataka na stranici.

2. State pattern

Naš sistem poznaje dvije vrste registrovanih korisnika, obične i premium. Međutim, premium korisnik sve dok ne unese svoj premium kod i dok mu se on ne verficira, dobiva tretman običnog registrovanog korisnika. To bi značilo da premium korisnik ne može ostvarivati popust pri kupovini karata, niti birati sektore VIP sjedišta sve dok se putem premium koda ne provjeri njegov status. Stanje (state) bi u ovom slučaju bila provjera da li je unesen validan premium kod, te od toga zavise korisnikove mogućnosti. Ovakav vid state patterna smo implementirali u našem sistemu.

3. Template method

Template metoda u našoj aplikaciji bi bio LogIn interface. Bez obzira da li se osoba registrira kao premium ili obični korisnik, u svakom slučaju se verificira broj bankovnog računa, da li je korisničko ime već zauzeto, da li je password

prema propisima aplikacije i slično. Međutim, ukoliko se osoba registruje kao premium korisnik, potrebno je da se provjeri, pored ostalih stvari, da li ima na računu dovoljno novca da odmah plati naknadu za aktivaciju korisničkog računa. Dakle, sve provjere prije novčane se izvršavaju kroz template login (koji je jednak loginu običnog registroanog korisnika) uz dodatnu provjeru iznosa za premium korisnika.

4. **Observer pattern**

Aplikacija šalje svim registrovanim korisnicima notifikaciju ukoliko, uslijed unosa novih rezultata, dođe do promjene stanja top 3 mjesta na ljestvici. Ta promjena predstavlja događaj koji je okidač slanja notifikacije, a observeri su svi registrovani korisnici. S obzirom da smo ovu funkcionalnost svakako mislili uključiti u projekt, implementirat ćemo je kroz observer pattern.

5. **Iterator pattern**

Kada zaposlenik unosi pojedinačne rezultate za vozače, može birati hoće li ih unositi po abecednom poretku ili po poretku prethodnog stanja na tabeli. Odabir vrši u odnosu na listu podataka (rezultata) koje ima. S obzirom da obe varijante zahtijevaju drugačije algoritme iteriranja kroz vozače, ovo bi bila primjena iterator patterna.

6. **Chain of Responsibility**

Glavna interakcija sistema sa korisnikom je prilikom kupovine karata. To je proces koji zahtijeva redoslijed različitih obrada. Kada se kupuje karta, sistem izvršava algoritme obrade količine karata, sektora sjedišta, cijene karata, validacije transakcije te evenutalnog unosa premium koda i njegove provjere. To se najbolje može iskazati kroz chain of responsibility pattern.

7. **Mediator pattern**

Već je napomenuto da se premium korisnik tretira kao obični pri kupovini ulaznica sve dok ne unese svoj premium kod. Međutim, nije dovoljno samo unijeti kod. Ključni korak je validacija korisnika, koja ne podrazumijeva samo da li je uneseni kod ispravan, već i da li je premium korisnik izmirio sve troškove i

time očuvao svoj premium status. To kontroliše sistem kroz mediator pattern. Tek kada premium korisnik prođe potrebne kontrole kod mediatora, onda mu mediator daje posebne pogodnosti i time utiče na njegovo iskustvo na sajtu.