Paterni ponašanja

Bave se organizacijom algoritama, dodjeljivanja odgovornosti i komunikaciju između objekata.

Strategy patern

Izdvaja algoritam iz matične klase i uključuje ga u posebne klase.

Ovaj patern bismo u našem sistemu mogli iskoristiti ukoliko bismo imali mogućnost izbora različitog prevoznog sredstva. Tada bismo dodali klasu PrevoznoSredstvo preko koje bi odlučivali koju 'strategiju' ćemo koristiti, tj na koje prevozno sredstvo je odabrano. Implementirali bismo interfejs IprevoznoSredstvo koji bi imao metodu izračunajCijenu i njega bi naslijedile klase Autobus i Avion koje bi na različit načine računale cijenu aranžmana, tj. na osnovnu cijenu bi dodavale različite iznose. Bilo bi moguće promijeniti 'strategiju', tj. prevozno sredstvo pri čemu bi se cijenja mijenjala shodno algoritmu implementiranom u određenoj klasi.

Na ovaj način bismo smanjili kompleksnost metoda time što ne bismo koristi mnogo if ili switch-case izraza.

State patern

Mijenja način ponašanja objekata na osnovu trenutnog stanja

U našem primjeru ovaj patern bismo mogli implementirati tako što klasa ... šalje obavijesti klasi Context do kojeg stepena rezervacije putovanja je korisnik stigao. Početno stanje bi bilo 'nemaRezervacije'. Na akciju klika na dugme Rezervisi, klasa Context mijenja svoje stanje na 'rezervisano', dok bi klasa Rezervacija izvršila rezervaciju putovanja za tog korisnika. Ukoliko bi klijent izabrao elektronski način plačanja, tada bi sljedeće akcija bila uplata amortizacije, pa bi se stanje promijenilo na 'uplacenaAmortizacija', dok bi klasa Amortizacija vršila isplatu vrijednosti amortizacije sa računa klijenta na račun agencije. Posljednje stanje bi bilo 'uplacenCijeliIznos', a da bi se moglo doći do tog stanja, bilo bi potrebno uvesti opciju 'Plati cijeli iznos' u naš sistem, i tada bi klasa CijeliIznos vršila isplatu ostatka cijene putovanja sa računa klijenta na račun agencije.

TemplateMethod patern

Omogućava algoritmima da izdvoje pojedine korake u podklase.

Ovaj patern nismo primijenili, ali bismo mogli ukoliko dodamo da klijent može biti obični i VIP i ukoliko bismo imali bolje razrađenu akciju plaćanja. Tada bismo algoritam za plaćanje putovanja mogli razdvojiti na operaciju plaćanja akontacije i plaćanje pune cijene (metode

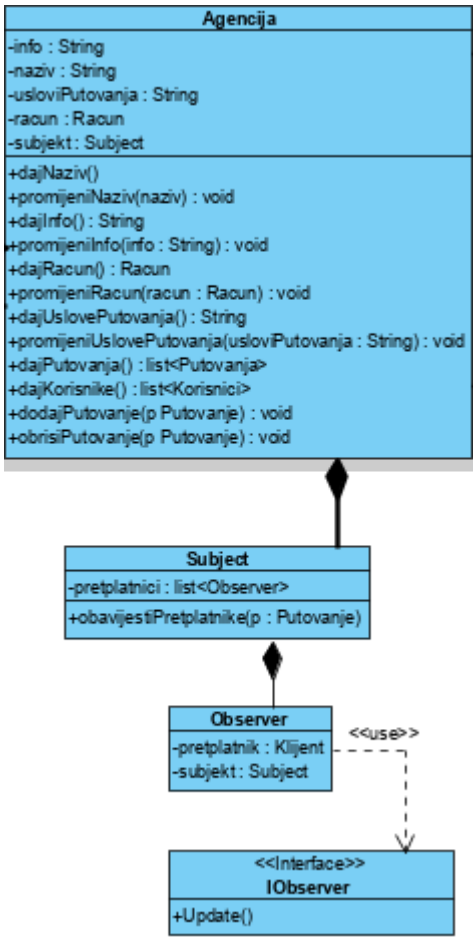
placanjeAkontacije i placanjePuneCijene). Klasa ObicniKlijent implementirala bi ove operacije na način da se plati puna akontacija, a zatim razlika pune cijene i akontacije, dok npr. klasa VIPKlijent akontaciju ne bi plaćala, a na punu cijenu bi još bio obračunat popust od 10%.

Observer patern

Uspostavlja relaciju između objekata takvu da kad se stanje jednog objekta promijeni svi vezani objekti dobiju informaciju

Svim putnicima koji su rezervisali putovanje šalje se obavijest putem mail-a ukoliko se putovanje odgodi.

Kada se u klasi Agencija pozove metoda obrisiPutovanje tada se prije svega svim klijentima iz liste onih koji su rezervisali to putovanje šalje obavijest preko klase Subject. Pozivom njene metode obavijestiPretplatnike, kojoj se kao parametar šalje lista korisnika koji su rezervisali to putovanje, poziva se event mehanizam Notify koji poziva Update metodu svakog Observer objekta koji kao svoj atribut ima klijenta koji je izvršio rezervaciju, te se tim klijentima šalje mail.



Iterator patern

Omogućava pristup elementima kolekcije sekvencijalno bez poznavanja interne strukture kolekcije

U našem primjeru u klasi Collection držimo kolekciju svih putovanja, koju je moguće proširiti sa dodatnim putovanjima, te je moguće izbrisati određeno putovanje. Klasa Agencija ima atribut tipa Ienumerable preko kojeg pristupa toj kolekciji i ta klasa, tačnije njene metode mogu iterirari kroz tu kolekciju ne vodeći računa na koji način je struktuirana ta kolekcija. Klasa Collection implementria interfejs IEnumerable koji ima metodu getEnumeration() koja je već uključena u okviru System.Collections. Zahvaljujući toj metodi kroz kolekciju možemo iterirati pomoću foreach iskaza, što nam je vrlo pogodno jer ćemo često pristupati kolekciji putovanja I izdvajati putovanja po određenom kriteriju, te za to možemo koristiti stream-ove.