Univerzitet u Sarajevu
Elektrotehnički fakultet
Sarajevo

Deseti projektni zadatak

Objektno orijentisana analiza i dizajn

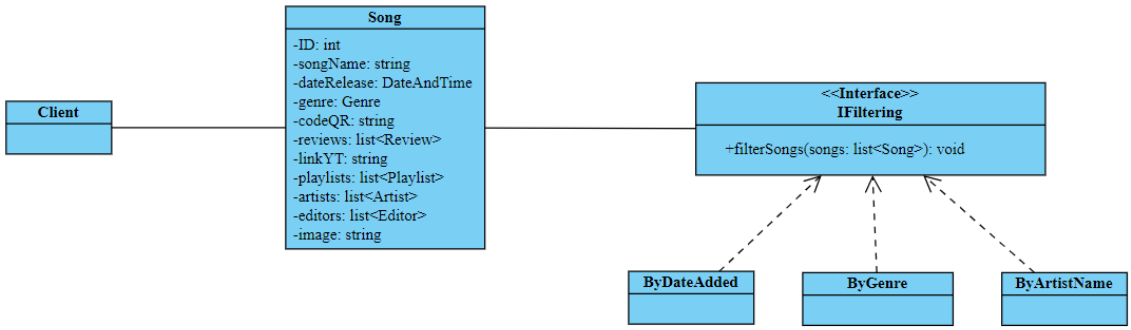
Naziv grupe: Spotifive
Članovi grupe: Nadina Miralem 18937
Amina Hromić 19084
Nerma Kadrić 19030
Amila Kukić 19065
Una Hodžić 19044

Paterni ponašanja

U nastavku slijedi opis paterna ponašanja koji će biti primijenjeni u sistemu.

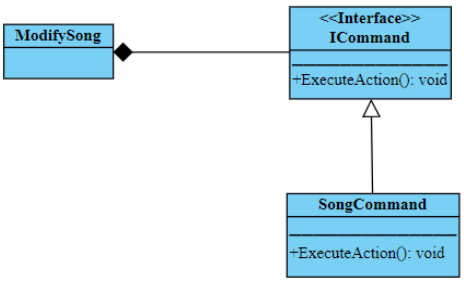
Strategy

Omogućava nam definisanje skupa algoritama koji mogu biti iskorišteni pri rješavanju nekog problema. U našem sistemu ovaj patern možemo iskoristiti prilikom filtriranja pjesama prema određenom parametru (prema žanru, nazivu izvođača, datumu kada je dodana i slično).



Command

Ovaj patern bismo mogli iskoristiti prilikom modificiranja informacija o pjesmi ili izvođaču. Za primjer ćemo uzeti klasu Song. Kreirat ćemo ICommand interface koji će imati metodu ExecuteAction(). Zatim ćemo dodati SongCommand klasu koja implementira ICommand interface, te klasu ModifySong koja će "upravljati" zahtjevima koji su vezani za neku pjesmu. Ta klasa može raditi sa bilo kojom klasom koja implementira ICommand interface.



U nastavku slijedi opis ostalih paterna ponašanja, te načina na koje bi se oni mogli implementirati u našem sistemu.

Interpreter

Interpreter klasa sadrži logiku koja će prevesti nešto što želimo u određeni format. Na primjer, kada bismo implementirali ovaj patern, iskoristili bismo ga za različite interpretacije formata datuma - bilo da je to format MM-DD-YY, DD-MM-YY ili neki drugi.

Template Method

Template method patern definiše kostur algoritma koji je potreban za izvršenje neke operacije, prebacujući neke korake u subklase. On omogućava subklasama da definišu te korake algoritma bez mijenjanja strukture samog algoritma. Ovaj patern bismo mogli implementirati ukoliko bismo u naš sistem uveli još jednog aktera - Premium korisnika koji bi plaćao aplikaciju, te imao dodatne funkcionalnosti koje nemaju drugi korisnici. Kako bi se pri registraciji morale, između ostalog, provjeriti i informacije o stanju bankovnog računa Premium korisnika, bilo bi pogodno iskoristiti Template method patern.

Chain of Responsibility

Ovaj patern nam pruža mogućnost proslijeđivanja nekog zahtjeva duž lanca handlera. Nakon primanja zahtjeva, svaki handler odlučuje o tome da li će obraditi zahtjev ili ga proslijediti idućem handleru u nizu. Prilikom same registracije korisnika, potrebna je validacija registracije na više nivoa. Na tom mjestu bismo mogli primijeniti Chain of Responsibility patern.

Flyweight

Pomoću ovog paterna više različitih objekata može koristiti isto glavno stanje. Primjer korištenja ovog paterna u našem sistemu bio bi vezan za postavljanje profilne slike korisnika. Za sve korisnike koji ne žele postaviti profilnu sliku bila bi iskorištena unaprijed zadana slika. Dakle, da bi korisnici mogli zadržati tu default-nu sliku, bilo bi potrebno implementirati Flyweight patern kako bi svi ti korisnici koristili jedan, zajednički resurs.

Iterator

Ovaj patern omogućava pristup elementima kolekcije, bez da se otkriva koja je njena stvarna struktura (stablo, lista, stack…). U našem sistemu postoje liste pjesama, izvođača… Svaka od tih listi kao element sadrži kompleksne objekte. Kako bismo lakše prolazili kroz navedene liste, mogli bismo implementirati Iterator patern.

Mediator

Mediator patern nam pomaže da smanjimo zavisnost između objekata. Ovaj patern bismo mogli implementirati na način da napravimo “bota” koji bi analizirao korisnikove preslušane pjesme i playlist a zatim bi slao notifikacije korisniku sa prijedlozima pjesama.

Memento

Omogućava nam da sačuvamo i restauriramo prethodna stanja nekog objekta. Zadatak ovog paterna je da pamti prethodno stanje objekta te da ga odvoji od trenutnog. Ukoliko se trenutno stanje iz bilo kojeg razloga izgubi, prethodno stanje objekta uvijek može biti vraćeno. Primjer korištenja bi bio implementacija jednog vida Undo operacije. Kako ne postoji potreba za tom operacijom u našem sistemu, ovaj patern nećemo implementirati.

Observer

Ukoliko bi naša aplikacija Registrovanim korisnicima pružala mogućnost "praćenja" Artista (Follow opcija), mogli bismo implementirati Observer patern jer bi tada korisnici primali notifikacije svaki put kada bude objavljena nova pjesma od tog Artist-a.

State

Ovaj patern omogućava objektu da promijeni svoje ponašanje kada se njegovo unutrašnje stanje promijeni, te se tada čini kao da je objekat promijenio klasu. Kada bi Administrator u našem sistemu odobravao promjene koje Editor uradi (dodavanje pjesme, brisanje pjesme i sl.), tada bismo mogli implementirati State patern. Nakon što Editor pošalje zahtjev, omogućeno mu je da prati stanje tog zahtjeva - da li je u stanju obrade, na čekanju ili je već obrađeno.

Visitor

Ovaj patern nam omogućava definisanje novih operacija bez mijenjanja klasa elemenata nad kojim operira. Drugim riječima, omogućava dodavanje novih ponašanja u postojeće klase, bez mijenjanja već postojećeg koda. Visitor patern možemo implementirati ukoliko nam je potreban prolazak kroz neku listu, pri čemu je potrebno izvršiti istu operaciju nad svakim elementom. Kako nam to nije potrebno, ovaj patern nećemo implementirati.