Strukturalni paterni

Adapter pattern

1. Namjena ovog paterna:

Osnovna namjena Adapter paterna je da omoguci širu upotrebu vec postojecih klasa. U situacijama kada je potreban drugaciji interfejs vec postojece klase, a ne želimo mijenjati postojecu klasu koristi se adapter patern. Adapter patern kreira novu adapter klasu koja služi kao posrednik izmedu originalne klase i željenog interfejsa. Tim postupkom se dobija željena funkcionalnost bez izmjena na originalnoj klasi i bez ugrožavanja integriteta cijele aplikacije. razlikujemo dvije vrste implementacije adapter patterna: pomocu nasljedjivanja i pomocu asocijacije/agregacije.

Adapter patern ustvari mapira interfejs jedne klase u drugu tako da mogu raditi zajedno.

2. Kako bismo mi mogli primijeniti i korisiti ovaj patern u nasem sistemu

Ovaj patern se vec koristi u nasem sistemu, jer imamo interfejs Rezervacija kojim prosirujemo mogucnosti klase Agencija. Na ovaj smo izbjegli kreiranje dodatne klase Rezervacija, a prosirili mogucnosti koje instanca klase Agencija posjeduje.

3. Kako koristenje ovog paterna poboljsava funkcionisanje ovog sistema?

Izbjegli kreiranje dodatne klase Rezervacija i smanjili smo kompleksnost programa(Keep it simple).

Fasadni patern

Namjena ovog paterna:

Fasadni patern sluzi da osigura vise pogleda visokog nivoa na podsisteme. Facade patern se koristi kada sistem ima više identificiranih podsistema pri čemu su apstrakcije i implementacije podsistema usko povezane.

2. Kako bismo mi mogli primijeniti i korisiti ovaj patern u nasem sistemu

Primjenili smo ovaj patern na nacin da imamo rezervisiPutovnje i otkazi putovanje u interfejsu Rezervacija. Sav teski posao upravljanja rezervacijama sto ukljucuje dodavanje i brisanje rezervacija kao i pretraga rezervacija po korisnicima i putovanjima pri cemu moramo paziti na nacin na koji su podaci

snimljeni u bazi je sada sakriven iza dvije metode rezervisiPutovnje i otkaziPutovanje i mozemo dodati jos dvije metode dajrezervacijezakorisnika i dajrezervacijezaputovanje.

3. Kako koristenje ovog paterna poboljsava funkcionisanje ovog sistema?

Olaksava upravljanje podacima na nacin da programer ne mora na vise od jednog mjesta brinuti o nacinu cuvanja podataka i relacijama podataka. Dvije pogodnosti toga su : citljiviji kod koji je lakse na vise mjesta koristiti(DRY princip), a druga pogodnost je laksi standardizovani pristup podacima koji su sada logicki grupisani.

Dekorator patern

Namjena ovog paterna:

Osnovna namjena Decorator paterna je da omogući dinamičko dodavanje novih elemenata i ponašanja (funkcionalnosti) postojećim objektima. Objekat pri tome ne zna da je urađena dekoracija što je veoma korisno za ponovnu upotrebu komponenti softverskog sistema.

Kako bismo mi mogli primijeniti i korisiti ovaj patern u nasem sistemu?

Decorator pattern koristimo da pruzimo dodatne funkcionalnost izvedenoj klasi a da se pri tome ne vidi razlika od bazne klase. Mi u nasoj implementaciji imamo registrovanog i neregistrovanog korisnika(tj. gosta i klijenta), oba objekta pretrazuju putovanja ali samo registrovani korisnik ima dodatnu mogucnost za dodavanje komentara na putovanja na kojima je bio.

Kako koristenje ovog paterna poboljsava funkcionisanje ovog sistema?

Ovim paternom je olaksano manipulisanjem podataka i lakse upravljanje tom mogucnosti komentarisanja putovanja u nasem sistemu.

Kompozitni patern

1. Namjena ovog paterna:

Osnovna namjena Composite paterna (kompozitni patern) je da omogući formiranje strukture stabla pomoću klasa, u kojoj se individualni objekti (listovi stabla) i kompozicije individualnih objekata (korijeni stabla) jednako tretiraju .

2. Kako bismo mi mogli primijeniti i korisiti ovaj patern u nasem sistemu?

Ovaj pattern nije iskoristen u nasem projektu, ali bi se mogao iskoristiti ako bismo recimo zeljeli

Definisati skup metoda za obracun placanja osiguranja za putovanje sa pojedinacnim putnicima, ali i sa grupama putnika na isti nacin.

3. Kako koristenje ovog paterna poboljsava funkcionisanje ovog sistema?

Koristenjem ovog paterna aplikacija je u stanju da vrsi opracija nad hijerarhijom raznih objekata kako jednostavnim tako i komplikovanih struktura. I ogranicava da se svaki put provjerava tacan tip objekta pa da se onda tek odlucuje o operaciji.

Bridge patern

1. Namjena ovog paterna:

Osnovna namjena Bridge paterna je da omogući odvajanje apstrakcije i implementacije neke klase tako da ta klasa može posjedovati više različitih apstrakcija i više različitih implementacija za pojedine apstrakcije. Bridge patern pogodan je kada se implementira nova verzija softvera a postojeća mora ostati u funkciji. Moguće je implementirati i sistem za razmjenu poruka primjenom Bridge paterna.

2. Kako bismo mi mogli primijeniti i korisiti ovaj patern u nasem sistemu?

Posto u nasem sistemu postoji klasa koja cesto varira u svojoj implementaciji i aplikacija moze imati vise razlicitih implementacija za apstrakciju.

Mozemo koristiti ovaj pater tako sto mozemo kreitati interfejs pogodnosti koji implementiraju sve klase koje predstavljaju pogodnost putovanja kao sto su klase PorodicniPopust, PutnickoOsiguranje, PraznicniPopust itd. na taj nacin ukoliko zelimo da dodamo novu pogodnost, potrebno je samo napraviti logiku za tu pogodnost (klasu sa odgovarajucim metodama) a ostatak sistema vec zna kako da koristi novu pogodnost.

3. Kako koristenje ovog paterna poboljsava funkcionisanje ovog sistema?

Koristenjem ovog paterna smanjujemo kompleksnost sistema. I dodajemo mogucnost za implementiranjem nove verzije softvera pri cemu stara verzija treba moci nastaviti da radi.

Proxy patern

1. Namjena ovog paterna:

Namjena Proxy paterna je da omogući pristup i kontrolu pristupa stvarnim objektima. Proxy je obično mali javni surogat objekat koji predstavlja kompleksni objekat čija aktivizacija se postiže na osnovu postavljenih

pravila. Proxy patern rješava probleme kada se objekt ne može instancirati direktno (npr. Zbog restrikcije pristupa).

2. Kako bismo mi mogli primijeniti i korisiti ovaj patern u nasem sistemu

Mogli bismo napraviti proxy klasu za Kodove, koja bi služila kao maska između klase Kod i tabele kodovi. Posto se kodovi rijetko mijenjaju, a cesto koriste, nema potreba da se svaki put kad se zatrazi lista kodova salje novi upit u bazu i time trose dargocijeni resursi. Moze se raditi *event based caching*, tako što će se nakon prvog upita u bazi u proxyKod klasi sacuvati lista svih kodova, i svaki put kada se zatrazi lista svih kodova, ne mora se raditi upit prema bazi vec se moze vratiti ta lista. Dodatno, ukoliko se dodaje novi kod u bazu ili mijenja postojeci, istovremeno se mijenja i lista kodova da se ocuva integritet podataka.

Klasa proxyKod treba da ima sve iste metode kao i klasa pomoću koje program inače komunicira sa bazom, tj. Gdje god se moze primijeniti orginal klasa, moze i proxy.

3. Kako koristenje ovog paterna poboljsava funkcionisanje ovog sistema?

Koristenje ovog paterna bi osiguralo objekte od zlonamjerne ili pogresne upotrebe. Primjenom ovog paterna omogućava se kontrola pristupa objektima, te se onemogućava manipulacija objektima ukoliko neki uslov nije ispunjen, odnosno ukoliko korisnik nema prava pristupa traženom objektu. U nasem slucaju ako bismo imali odredjene restrikcije sprijecilo bi mnoge konflikte ili neke nesporazume unutar uredjenja sistema.

Na ovaj nacin bismo ustedili vrijeme pristupa bazi, a posto kodovi nisu memorijski veliki, ne bi bili veliki teret aplikaciji.

Flyweight patern

1. Namjena ovog paterna:

Postoje situacije u kojima je potrebno da se omogući razlikovanje dijela klase koji je uvijek isti za sve određene objekte te klase (tzv. Glavno stanje) od dijela klase koji nije uvijek isti za sve određene objekte te klase (tzv. Sporedno stanje). Osnovna namjena Flyweight paterna je upravo da se omogući da više različitih objekata dijele isto glavno stanje, a imaju različito sporedno stanje.

2. Kako bismo mi mogli primijeniti i korisiti ovaj patern u nasem sistemu?

Ovaj patern bismo mogli iskoristiti u nasem sistemu ako bismo iz klase Putovanje zeljeli nasljediti klase paket poude1 i ponuda2 ako bismo imali potrebu da razdvojimo odredjena putovanja(npr. prema VipKLijentima ili ponude za „obicne klijente“ ili prema placanju putnickog osiguranja za individualne putnike ili za grupe putnika...)

3. Kako koristenje ovog paterna poboljsava funkcionisanje ovog sistema?

Ovim paternom štedimo memoriju sistema jer imamo potrebu za manjim brojem objekata i povecavamo preformanse jer veliki broj objekata treba ucinkovito podrzavamo.