DIZAJN PATERNI

- MOGUĆNOST IMPLEMENTACIJE U PROJEKTU „AV-MAU AZIL“ -

-

Composite patern

Composite patern omogućava grupama objekata koje predstavljaju hijerarhiju da budu tretirane na isti način kao jedna instanca objekta. Jednostavnije rečeno, funkcija ovog paterna je da se jedna komponenta i druga komponenta mogu tretirati na isti način. Tipične operacija nad komponentama uključuju add, remove, display, find i group. Aplikacije za puštanje muzike (iTunes, Winamp..) ili aplikacije za kreiranje albuma sa slikama (npr. Flickr) koriste ovaj patern. Stavke se smještaju u velike liste, koje se zatim na određeni način struktuiraju.

- **PRIMJER PRIMJENE U PROJEKTU “AV-MAU AZIL”:** Ako posmatramo PetPhoto odjeljak u našoj aplikaciji, vidjećemo da je u planu da postoje različiti načini na koje možemo prikazivati slike koje su u njega unijete: hronološki ili na osnovu imena pod kojim su unete (npr. Ljeto2918uAzilu). Jedna fotografija se može pojaviti u okviru više albuma. Kreiranje albuma stvara kompozitni objekat, koji ne uključuje stvarno kopiranje fotografija na više lokacija.

Iterator patern

Iterator patern omogućuje kretanje kroz kolekciju na formalizovan način. Konkretnije, omogućava sekvencijalni pristup elementima u kolekciji a da se pri tome ne mora znati njena struktura. Pored toga, patern omogućava filtriranje elemenata na različite načine. Uz koncept iternatora definira se i koncept enumeratora koji su odgovorni za dobavljanje sljedećeg elementa u sekvenci koju definiše određeni kriterij. Za takvu sekvencu se kaže da je „enumerable“.Primjeri mogu biti sljedeći cijeli broj, ili sljedeći naručen proizvod sortiran po datumu.

- **PRIMJER PRIMJENE U PROJEKTU “AV-MAU AZIL”:** Iterator patern će naći odličnu primjenu u našem projektu za filtriranje Petbook-a, odnosno za prolazak kroz kolekcijuživotinja prilikom traženje životinje za udomljavanje od strane korisnika.

Memento patern

Memento patern omogućava vraćanje objekta u prethodno stanje, bez narušavanja principa enkapsulacije i privatnosti objekta. Primjer primjene memento paterna su „checkpoints“ sačuvani u video igrici gdje je igrač u mogućnosti da se vrati na nivoe koje je već osvojio. Drugi primjer je un-do operacija u aplikacijama koje procesiraju riječi. Memento koristi Originator klasu, koja je objekat koji će biti sačuvan i „restauriran“ kasnije. Memento klasa čuva „historijsku“ informaciju Originatora u nekom trenutku.

- **PRIMJER PRIMJENE U PROJEKTU "AV-MAU AZIL":** Memento patern će biti koristan da se obezbijedi "undo" mehanizam u našoj aplikaciji, kada interno stanje objekta treba povratiti ili obnoviti u nekom kasnijem stadijumu. Primjer je klasa Korisnika kojoj možemo sačuvati njene originalne vrijednosti, kako bi kasnije mogli izvršiti un-do na bilo kakve izmjene.

Prototype patern

Prototype patern je kreacijski patern koji se odnosi na kloniranje objekta. On ima dvije prednosti: ubrzava instanciranje veoma velikih klasa koje se dinamički učitavaju (kada je kopiranje objekata brže), i čuva znanje o sastavnim dijelovima velikih struktura podataka koje se mogu kopirati bez da je potrebno znati podklase od kojih su kreirane. Suština kod ovog paterna jeste da program kreira objekat željenog tipa, ne tako što će da kreira novu instancu, već tako što će da kopira praznu instancu date klase.

- **PRIMJER PRIMJENE U PROJEKTU "AV-MAU AZIL":** Kao primjer upotrebe ovog paterna, možemo posmatrati fiktivno "AvMauAzil" proširenu aplikaciju koja bi radila sa slikama i koja bi čuvala grupe fotografija. Sa takvom aplikacijom u određenom momentu, mi možemo poželeti da arhiviramo jednu grupu fotografija, tako što ćemo ih kopirati u drugi album. U ovom slučaju, arhiva postaje kontejner za prototipove koji se može kopirati kada god je to neophodno. Objekti se obično instanciraju od klasa koje predstavljaju dio programa. Prototype patern predstavlja alternativu ovom pristupu jer se objekti kreiraju na osnovu prototipova.

Proxy pattern

Namjena Proxy paterna je da omogući pristup i kontrolu pristupa stvarnim objektima. Proxy je obično mali javni surogat objekat koji predstavlja kompleksni objekat čija se aktivacija postiže na osnovu postavljenih pravila. Proxy patern rješava probleme kada se objekt ne može instancirati direktno (npr. zbog restrikcije pristupa). Struktura Proxy patterna je sastavljena od klasa:

1. Subject (ISubject) zajednički interfejs za realne/stvarne subjekte i proksije-surogate (proxies) koji omogućava da se oni koriste naizmjenično.
2. RealSubject je glavni objekat kojeg "predstavlja" proxy.
3. Proxy - implementira isti interfejs kao RealSubject tako da se Proxy može koristiti umjesto RealSubject objekta. Proxy vrši kontrolu pristupa RealSubject objektu - može kreirati i brisati taj objekat.

PRIMJER UPOTREBE U PROJEKTU „AV-MAU AZIL": Ukoliko bi u budućnosti u Azilu bio razvijan sistem naplate karticom (na primjer za kupovinu hrane ili drugih rekvizita za ljubimce), svaki put kada se vrši naplata sa kartice, bio bi korišten proxy pattern. Kartica korisnika je ustvari proxy reprezentacija bankovnog računa. Proxy omogućuje naplatu sa kartice klijenta bez pristupa svim podacima računa

korisnika (koji je u pozadini, i koji je implementiran „skupljom“ strukurom). Na ovaj način, Proxy obezbeđuje skladište za kompleksniju klasu čije je instanciranje „skupo“.

Decorator pattern

Osnovna namjena Decorator paterna je da omogući dinamičko dodavanje novih elemenata i ponašanja (funkcionalnosti) postojećim objektima. Može se naprimjer koristi za implementaciju različitih kompresija videa, simultano prevođenje. Ključna tačka implementacije kod ovog paterna jeste da "dekorator" objekti u isto vreme i implementira isti interfejs kao i originalna klasa i sadrži instancu njegove implementacije (originalnu klasu).

Decorator pattern čine sljedeće klase:
1. Component – Originalna klasa (koja sadrži interfejs koji se može mijenjati ili mu se mogu dinamički dodati operacije)

2. IComponent – interfejs koji identificira klase objekata koji trebaju biti dekorisani
3. Decorator – klasa koja odgovara IComponent interfejsu i implementira dinamički prošireni interfejs.

PRIMJER PRIMJENE U PROJEKTU “AV-MAU AZIL”: Recimo da AvMAU Azil planira uvesti malu biblioteku i videoteku u svoju instituciju, u svrhe podizanja svijesti o životinjama lutalicama. Dakle, kreirali bi projekat u okviru kojeg bi postojećim klasama koje predstavljaju knjige i filmove, dodali funkcionalnost da se mogu izdavati. Ukratko, dodali bi dvije klase Knjiga I Video.Nakon dodavanja osnovnih klasa dodali bi I klasu Iznajmljivo, koja predstavalja dekorator klasu koja će dodati nove funkcionalnosti postojećim klasama. Na kraju u okviru klase Program mijenjamo kod kako bi main metoda kreirala originalne i dekorisane(Iznajmljive) objekte.

Facade pattern

Facade patern se koristi kada sistem ima više identificiranih podsistema (subsystems) pri čemu su apstrakcije i implementacije podsistema usko povezaneOsnovna namjena Facade paterna je da osigura više pogleda visokog nivoa na podsisteme (implementacija podsistema skrivena od korisnika).Može se više fasada postaviti oko postojećeg skupa podsistema i na taj način formirati više prilagođenih pogleda na sistem.

- Struktura Facade patterna sačinjena je od sljedećih klasa:

    1. Facade – definira i implementira jedinstven interfejs za skup operacija nekog podsistema
    2. SubsystemClassN – definira i implementira N-ti interfejs u skupu interfejsa nekog podsistema

- Implementacija podsistema je u okviru odvojenih imenovanih prostora (ili biblioteka), u kojima se sa specifikatorima pristupa može postaviti željena vidljivost klase.
-

PRIMJER PRIMJENE U PROJEKTU "AV-MAU AZIL": Facade patern omogućuje jednostavan interfejs i kontroliše pristup nizu komplikovanih interfejsa podsistema. Tipičan primjer gdje bi se ovaj patern mogao upotrebiti u okviru našeg projekta jese za eventualno naručivanje namirnica za kućne ljubimce preko interneta. Ukoliko vlasnik ljubimca često naručuje hranu za jubimca preko našeg sajta, on može sačuvati podatke o svojooj kreditnoj kartici I o adresi gde namirnice trebaju biti dostavljene. Na ovaj način bi ubrzali proces naručivanja robe. Ono što razlikuje Façade patern od Adapter paterna ili Decorator paterna je to što su interfejsi koji se kreiraju u potpunosti novi (nisu povezani sa postojećim zahtjevima i ne moraju se podudarati sa postojećim interfejsima).

Singleton patern (unikat)

Singleton pattern ograničava instanciranje klase i osigurava da samo jedna instanca date klase postoji i pruža globalnu tačku pristupa ka toj instanci. Patern osigurava da je klasa instancirana samo jednom i da su svi zahtevi upućeni ka tom jednom i samo jednom objektu. Tipičan primjer su globalni objekti (kao što je neka Configuration klasa) ili deljeni resursi (kao što je event queue).

PRIMJER PRIMJENE U PROJEKTU "AV-MAU AZIL": Singleton patern bi koristili u kombinaciji sa Configuration klasom koja po svojim svojstvima i namjeni odgovora upotrebi sa singleton patternom. Configuration klasa predstavlja konfiguraciju, računara, .Net klijent aplikacija, ASP.Net aplikacija, Web direktorija i dr. Ova konfiguracijska klasa će u projektu biti instanciarana samo jednom i imati globalnu tačku pristupa.

Observer patern

Observer patern definiše način na koji jedna ili više klasa treba da budu upozorene na promjene u drugoj klasi, odnosno obaviještene da su nastale promjene u drugoj klasi. Observer pattern se koristi ukoliko postoje jedna-na-više veze između objekata takve da ako se jedan objekat modifikuje, ostali će biti obavješteni automatski. Patern koristi tri klase aktera: subjekat, observer i objekat. Observer nadzire subjekat i svaki put kad se on promijeni obavještavaju se objekti.

PRIMJER PRIMJENE U PROJEKTU "AV-MAU AZIL": U okviru projekta "AvMau Azil" dobar primjer upotrebe Observer paterna bi bio prilikom slanja notifikacija o novo-pristigloj životinji u Azil. Vozač po preuzimanju životinje sa terena, šalje notifikaciju ostalim uposlenicima Azila. Nakon toga Veterinar zna da treba obaviti pregled, Upravitelj da će biti registrirana nova životinja u azil, Higijeničar treba da spremi boks itd..

Interpreter patern

Interpreter patern spada u patene ponašanja i određuje kako evaluirati/interpretirati rečenice ili izraze u jeziku. Ovaj patern implementira Expression interfejs koji govori kako da se interpretira određeni kontekst. Često se koristi u kompajlerima i parserima.

PRIMJER PRIMJENE U PROJEKTU "AV-MAU AZIL": Ovaj patern se može koristiti u slučaju procjene validnosti šifre admina ili korisnika prilikom korištenja aplikacije. Kontrola paterna se odnosi na provjeru da li je unesena šifra po traženom standardu.