Strukturalni paterni

Adapter pattern

Osnovna namjena Adapter paterna je da omogući širu upotrebu već postojećih klasa. U situacijama kada je potreban drugačiji interfejs već postojeće klase, a ne želimo mijenjati postojeću klasu koristi se Adapter patern. Adapter patern kreira novu adapter klasu koja služi kao posrednik između orginalne klase i željenog interfejsa.

Facade pattern

Kako mu i sam naziv govori, ovaj patern se korisiti s ciljem da osigura više pogleda visokog nivoa na podsisteme (implementacija podsistema skrivena od korisnika). Ni ovaj patern nismo uspjeli implementirati, ali bi se mogao primjeniti kod izrade različitih vrsta izvještaja koji mogu biti zatraženi iz baze podataka.

Decorator pattern

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. U našem sistemu konkretno ne možemo navesti primjer ovog paterna.

Bridge pattern

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.

Composite pattern

Composite patern služi za kreiranje hijerarhije objekata. Koristi se kada svi objekti imaju različite implementacije nekih metoda, no potrebno im je svima pristupati na isti način, te se na taj način pojednostavljuje njihova implementacija. Primjena ovog paterna unutar sistema nije moguća, jer odstupa od generalnog uzorka po kojem bi trebala funkcionisati aplikacija.

Proxy pattern

Proxy patern služi za dodatno osiguravanje objekata od pogrešne ili zlonamjerne 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.

Flyweight pattern

Flyweight patern koristi se kako bi se onemogućilo bespotrebno stvaranje velikog broja instanci objekata koji svi u suštini predstavljaju jedan objekat. Samo ukoliko postoji potreba za kreiranjem specifičnog objekta sa jedinstvenim karakteristikama (tzv. specifično stanje), vrši se njegova instantacija, dok se u svim ostalim slučajevima koristi postojeća opća instance objekta (tzv. bezlično stanje). Primjena ovog paterna unutar sistema nije moguća, jer odstupa od generalnog uzorka po kojem bi trebala funkcionisati aplikacija.