Factory – Kreacijski

Opis

Factory patern je kreacijski patern koji omogućava kreiranje objekata bez navođenja tačnih klasa objekata koji se kreiraju. Koristimo ga ukoliko nismo sigurni tačno sa kakvim tipom objekata će naš kod raditi. On definiše interfejs za stvaranje objekata, ali dozvoljava podklasama da promijene tip objekta koji će biti kreiran.

Problemi koje rješava

U aplikaciji imamo različite vrste korisnika (Donor, Zaposlenik, Admin) koji dijele zajedničke osobine, ali također imaju specifične atribute i ponašanja. Bez Factory Patterna, instanciranje ovih različitih tipova korisnika može postati kompleksno i neorganizirano, naručito kada se doda potreba za inicijalizacijom specifičnih atributa.

Primjena

Korištenjem Factory Patterna za kreiranje korisničkih objekata u aplikaciji VitalFlow, postiže se centralizacija i enkapsulacija logike za instanciranje različitih tipova korisnika. Ovo čini kôd čistijim, lakšim za održavanje i proširenje, te smanjuje mogućnost grešaka pri kreiranju objekata.

Abstract Factory – Kreacijski Patern

Opis

Abstract Factory Pattern je kreacijski patern koji pruža interfejs za kreiranje familija povezanih ili zavisnih objekata bez navođenja njihovih konkretnih klasa. Omogućava grupisanje logike za kreiranje srodnih objekata u okviru jedne fabrike. Ovaj pattern prati open-closed i single-responsibility principe. Slično kao i prethodni pattern, ovaj metod također centralizira kreiranje objekata na jedno mjesto u aplikaciji.

Problemi koje rješava

U aplikaciji VitalFlow potrebno je upravljati raznovrsnim resursima kao što su zalihe krvi i notifikacije za donore i klinike. Bez odgovarajuće strukture, kreiranje i upravljanje ovim resursima može postati kompleksno i teško za održavanje.

Primjena

Korištenjem ovog paterna u aplikaciji VitalFlow za upravljanje resursima kao što su zalihe krvi i notifikacije, postižemo bolju organizaciju i strukturu koda. Ovo omogućava lakše održavanje i proširenje aplikacije, osigurava dosljednost u kreiranju objekata i smanjuje mogućnost grešaka. Abstract Factory Pattern je idealan kada imamo potrebu za kreiranjem kompleksnih struktura povezanih objekata, što je čest slučaj u složenim aplikacijama kao što je ova.

Builder – Kreacijski Patern

Opis

Builder Pattern je kreacijski patern koji omogućava konstruisanje složenih objekata korak po korak. Omogućava kreiranje različitih reprezentacija objekta koristeći istu konstrukciju.

Problemi koje rješava

U aplikaciji VitalFlow imamo različite tipove korisnika (Admin, Donor, Zaposlenik) s mnogo atributa. Kreiranje ovih objekata korištenjem velikih konstruktora može dovesti do:

- **Nepreglednog koda**: Gigantsko dugi konstruktori s mnogo parametara postaju teško čitljivi i održivi.
- **Visoke vjerovatnoće grešaka**: Povećana mogućnost grešaka zbog nepravilnog redoslijeda ili zaboravljenih parametara prilikom instanciranja objekata.
- **Teškog održavanja**: Dodavanje ili izmjena parametara zahtijeva promjenu svih poziva konstruktora.

Primjena

Koristićemo Builder Pattern za kreiranje različitih tipova korisnika (Admin, Donor, Zaposlenik) u našoj aplikaciji. Korištenjem Builder Patterna u aplikaciji VitalFlow za kreiranje korisničkih objekata, postižemo bolju organizaciju i strukturu koda. Ovo omogućava lakše kreiranje složenih objekata korak po korak, poboljšava čitljivost i održavanje koda, te smanjuje mogućnost grešaka prilikom instanciranja objekata. Builder Pattern je posebno koristan kada imate objekte sa mnogo opcionalnih parametara ili kompleksnim inicijalizacijama.

Prototype – Kreacijski Patern

Opis

Prototype Pattern je kreacijski patern koji omogućava kloniranje postojećih objekata umjesto kreiranja novih objekata od nule. Omogućava kreiranje novih objekata na osnovu šablona, čime se smanjuje kompleksnost i vrijeme potrebno za instanciranje.

Problemi koje rješava

U aplikaciji VitalFlow, Prototype Pattern bi se mogao primjeniti za kloniranje instanci korisnika kao što su Donor, Zaposlenik ili Administrator, posebno ukoliko ove instance sadrže više kompleksnih atributa i trebaju se često kreirati sa sličnim početnim stanjima.

Primjena

Iako su naše trenutne klase dovoljno jednostavne i ne zahtijevaju hitnu primjenu ovog paterna, Prototype Pattern može postati koristan kako se aplikacija bude razvijala i kako se budu dodavali složeniji atributi i ponašanja korisničkim objektima. Korištenje Prototype Patterna omogućilo bi jednostavnije i brže kreiranje kopija postojećih objekata, što bi poboljšalo efikasnost i održavanje koda.

Singleton – Kreacijski Patern

Opis

Singleton Pattern je kreacijski patern koji osigurava da klasa ima samo jednu instancu i pruža globalnu tačku pristupa toj instanci. Ovo je korisno za upravljanje resursima koje je potrebno centralizovati.

Problemi koje rješava

U aplikaciji VitalFlow postoje resursi i stanja koja treba da budu jednistvena kroz cijelu aplikaciju kao što su:

- **Stanje zaliha krvi**: Praćenje zaliha krvi mora biti centralizovano kako bi se osigurala tačnost podataka i efikasno upravljanje donacijama i potrebama.
- **Sistem za notifikacije**: Slanje notifikacija korisnicima treba biti centralizovano kako bi se osiguralo konzistentno obavještavanje svih korisnika.

Primjena

Koristićemo Singleton Pattern za upravljanje stanjem zaliha krvi i sistemom za notifikacije. Korištenjem Singleton Patterna u aplikaciji za upravljanje globalnim stanjima i resursima kao što su stanje zaliha krvi i sistem za notifikacije, postižemo centralizovanu kontrolu, smanjujemo mogućnost grešaka i poboljšavate efikasnost koda. Singleton Pattern je idealan za resurse koji trebaju biti jedinstveni kroz cijelu aplikaciju, omogućavajući jednostavan i kontrolisan pristup.