

## **Factory – Kreacijski Patern**

### **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 attribute i ponašanja. Bez Factory Patterna, instanciranje ovih različitih tipova korisnika može postati kompleksno i neorganizirano, naročito 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, postizemo 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, visoke vjerovatnoće greške i teškog održavanja.

### **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, postićemo 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 ovoj aplikaciji ovaj patern bi mogli primjeniti za kloniranje instanci donor, zaposlenik ili administrator, ukoliko bi one sadržavale više kompleksnih atributa.

### **Primjena**

Budući da su naše klase još uvijek dovoljno jednostavne primjena ovog paternu nije previše neophodna.

# 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 imamo resurse i stanja koja treba da budu jedinstvena kroz cijelu aplikaciju, kao što je stanje zaliha i sistem za notifikacije.

## 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, postizemo 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.