Kreacijski paterni

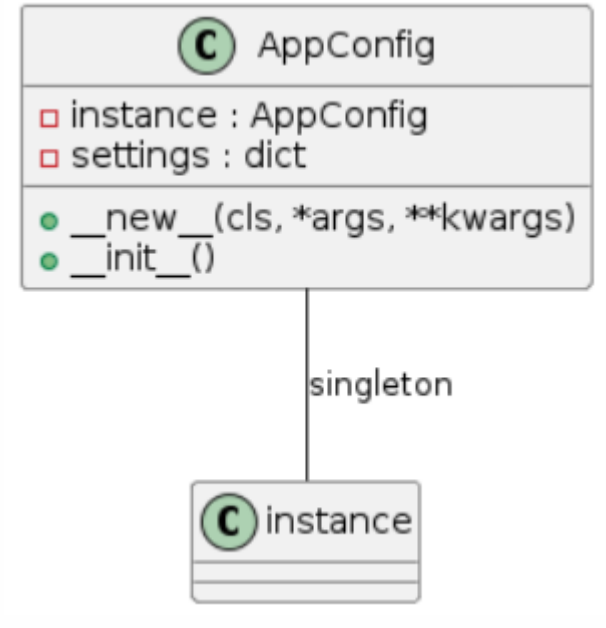
Potreba za Kreacijskim Paternima:

Kreacijski paterni omogućavaju fleksibilno stvaranje objekata, čime se olakšava proširenje i održavanje koda. Oni pomažu da se izbjegne direktna upotreba konkretnih klasa i umjesto toga koriste apstrakcije za kreiranje objekata, čime se povećava fleksibilnost i proširivost sistema.

1. Singleton pattern

Singleton patern osigurava da klasa ima samo jednu instancu i pruža globalnu tačku pristupa toj instanci.

U aplikaciji za praćenje fitnes planova, klasa koja upravlja konfiguracijom aplikacije može koristiti Singleton patern. Na primjer, klasa AppConfig može sadržavati postavke aplikacije koje trebaju biti dostupne kroz cijeli sistem.



2. Prototype pattern

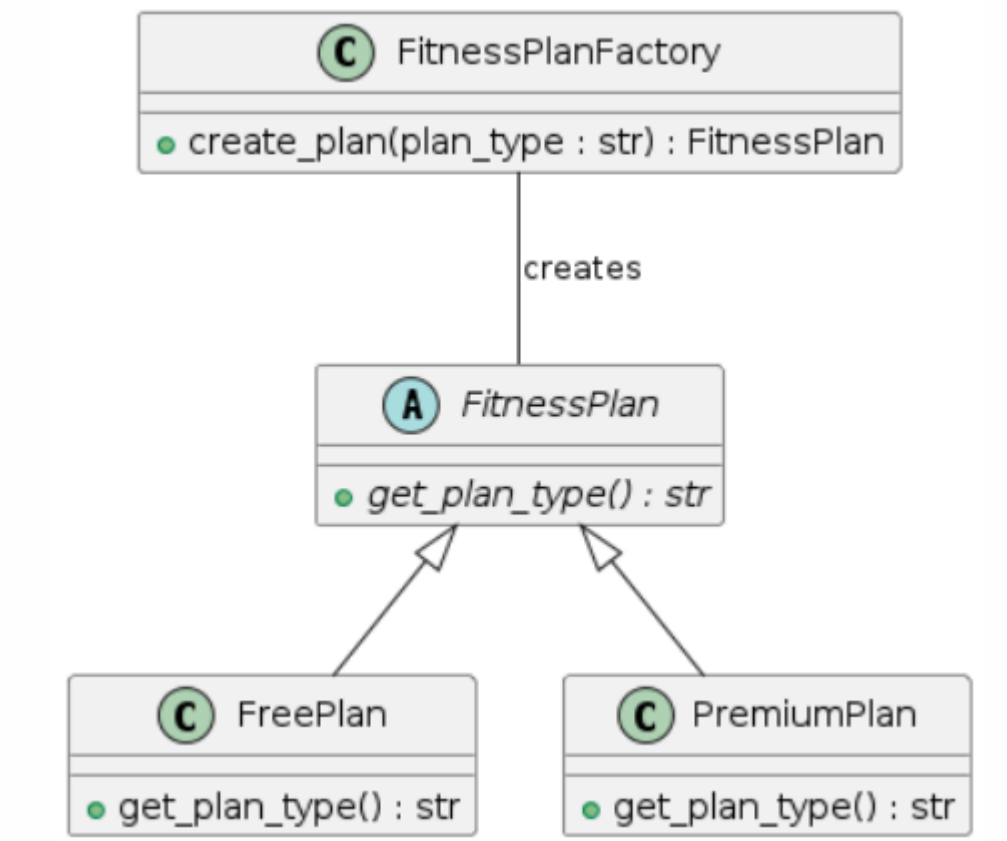
Prototype patern omogućava kreiranje novih objekata kloniranjem postojećih instanci.

Koristan je kada korisnici često kopiraju postojeće fitnes planove i prilagođavaju ih svojim potrebama. Na primjer, korisnici mogu klonirati postojeći plan i napraviti male izmjene.

3. Factory Method pattern

Factory Method patern omogućava klasi da delegira stvaranje objekata podklasama, čime se omogućava jednostavno proširenje i modifikacija bez izmjene osnovne klase.

U aplikaciji imamo različite tipove fitnes planova (npr. BesplatniPlan, PremiumPlan). Umjesto direktnog stvaranja objekata, koristit ćemo Factory Method za kreiranje fitnes planova.



4. Abstract factory pattern

Abstract Factory patern pruža sučelje za stvaranje porodica povezanih objekata bez specificiranja njihovih konkretnih klasa. To omogućava stvaranje grupa povezanih objekata koji se mogu koristiti zajedno, a da se ne mora znati njihove konkretne implementacije.

U našoj aplikaciji, možemo imati različite tipove korisničkih interfejsa za različite platforme. Na primjer, možemo imati WebUIFactory, MobileUIFactory i DesktopUIFactory koje generiraju UI komponente specifične za web, mobilne i desktop aplikacije.

5. Builder pattern

Builder patern odjeljuje konstrukciju kompleksnog objekta od njegovog predstavljanja, omogućavajući isti proces konstrukcije za različite reprezentacije.

Idealno za stvaranje složenih fitnes planova koji zahtijevaju mnogo parametara. Builder može pomoći u stvaranju prilagođenih fitnes planova sa specifičnim ciljevima, vježbama i rasporedom.

6. Object Pool

Object Pool patern koristi skup inicijaliziranih objekata koji su spremni za korištenje,umjesto stvaranja i uništavanja objekata na zahtjev.

Efikasan za upravljanje konekcijama prema bazi podataka ili drugim resursima koji se često koriste. Na primjer, DatabaseConnectionPool može upravljati skupom konekcija prema bazi podataka.

Zaključak:

Uvođenje kreacijskih paterna u naš sistem donijet će mnoge prednosti, uključujući bolju organizaciju koda, lakše održavanje i veću fleksibilnost pri dodavanju novih funkcionalnosti. Implementacija Singleton i Factory Method paterna značajno će unaprijediti trenutno stanje sistema, dok će ostali paterni pružiti mogućnosti za daljnje poboljšanje u specifičnim situacijama.