*Objektno Orijentisana Analiza i Dizajn*

Strukturalni paterni

1. **Adapter pattern**

Koristi se za prilagođavanje postojećih klasa ili interfejsa novim zahtevima aplikacije. Adapter koristimo da se prilagodi različitim tipovima fitnes uređaja ili servisa za plaćanje.

Aplikacija za fitnes može imati različite tipove fitnes uređaja kao što su trake za trčanje, bicikli... Svaki od ovih uređaja može imati svoj specifičan interfejs za kontrolu, kao što su komande za početak, zaustavljanje, podešavanje brzine, i slično. Adapter patternom možemo prilagoditi svim različitim tipovima uredjaja bez potrebe za pisanjem novog koda.

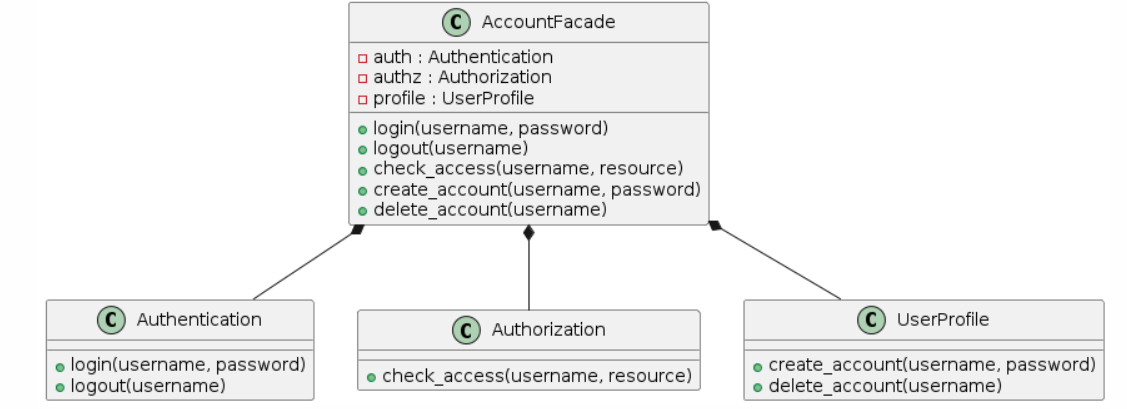
Takodje mozemo je prilagoditi različitim tipovima plaćanjima kao sto su Paypal, bankovni transferi.. Možemo za svaki odraditi jedinstven dizajn za plaćanje preko Adapter patterna.

1. **Facade pattern**

Prilagođava složene podsisteme jednostavnijem interfejsu. Facade pružita jednostavan interfejs za upravljanje korisničkim nalozima, što pozadinski upravlja složenim operacijama autentifikacije i autorizacije.

U aplikaciji za fitnes, složeni podsistemi mogu obuhvatati operacije kao što su autentifikacija, autorizacija, upravljanje korisničkim profilima, kontrola pristupa, kreiranje novih naloga, brisanje naloga, promena lozinki i slično. Svaki od ovih podsistema može zahtevati različite korake i procese kako bi se izvršile odgovarajuće operacije.

Facade pattern omogućava da se ovaj kompleksan skup operacija pojednostavi kroz jednostavan interfejs. Na primer, Facade klasa može biti kreirana koja pruža jednostavne metode poput **login**, **logout**, **changePassword**, **deleteAccount**, koje sakrivaju složene pozadinske operacije autentifikacije, autorizacije i upravljanja nalozima.



*Objektno Orijentisana Analiza i Dizajn*

1. **Decorator pattern**

Omogućava dinamičko dodavanje novih funkcionalnosti ili atributa objektima. Decorator koristimo za dodavanje opcija filtriranja ili sortiranja u pregledima treninga.

* Prvo, definišite osnovni objekat koji predstavlja pregled treninga. Ovaj objekat može sadržati osnovne funkcije kao što su prikazivanje lista treninga, detaljne informacije o svakom treningu.
* Dekorator za filtriranje treninga po vrsti vežbi, po trajanju treninga, po težini vežbi ili bilo kojem drugom kriterijumu .

1. **Bridge pattern**

Bridge pattern omogućava razdvajanje apstrakcije i implementacije klase tako da klasa može imati više apstrakcija i više implementacija za apstrakcije.

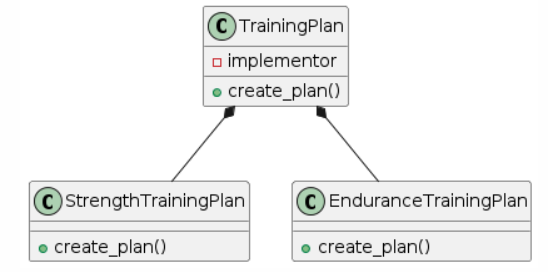
Na primer, u aplikaciji za fitnes, apstraktna struktura može biti "Trening Plan" koja definiše osnovne karakteristike trening plana kao što su lista vežbi, trajanje treninga, intenzitet, itd. Sa druge strane, implementacije mogu biti različiti tipovi trening planova kao što su plan za snagu, plan za izdržljivost, plan za mršavljenje...

1. **Composite pattern**

Koristi se za organizaciju objekata u hijerarhijsku strukturu tako da se mogu tretirati kao jedinstven objekat. Composite koristimo za grupisanje vežbi u trening planove .

Trening plan može biti složen objekat koji sadrži različite elemente kao što su vežbe, pauze, opisi vežbi, slike ili video materijali za vežbe, broj ponavljanja i serija, intenzitet i slično. Svaki od ovih elemenata može biti predstavljen kao zaseban objekat.

Jedna od ključnih prednosti Composite patterna je što omogućava da se grupisane vežbe tretiraju kao jedinstven objekat. To znači da admin može dodavati, uklanjati ili menjati grupisane vežbe u trening planu kao da su to jednostavni objekti, iako su zapravo kompleksne strukture koje sadrže druge objekte.



1. **Proxy pattern**

Omogućava kontrolu pristupa ili dodatne operacije nad objektom bez direktnog pristupa objektu. Proxy koristimo ograničavanje pristupa određenim funkcijama aplikacije.

Aplikacija može imati različite nivoe pristupa za admina i korisnike. Proxy može poslužiti kao posrednik koji proverava korisnikove privilegije pre nego što omogući pristup određenim funkcijama ili podacima. Na primer, admin može imati pristup svim funkcijama aplikacije dok se korisnicima može ograničiti pristup samo određenim funkcionalnostima koje su relevantne za njihovu ulogu.

**7. Flyweight pattern**

Flyweight pattern je dizajn pattern koji se koristi kako bi se smanjila potrošnja memorije tako što se deljeni objekti koriste umesto stvaranja novih objekata za svaku instancu. Ovaj pattern je posebno koristan kada aplikacija ima veliki broj sličnih objekata ili kada objekti imaju veliki uticaj na potrošnju memorije.

U aplikaciji za fitnes, deljeni objekti su informacije o vežbama koje se često koriste u različitim treninzima.

Umesto da se za svaku vežbu u trening planu kreira nova instanca sa svim podacima, može se koristiti deljeni objekat koji sadrži osnovne informacije o vežbi, dok se specifični podaci (kao što su težina, broj ponavljanja itd.) čuvaju samo u instanci konkretnog trening plana.

*Objektno Orijentisana Analiza i Dizajn*