STRUKTURALNI PATERNI

1. *Adapter*

Adapter pattern možemo iskoristiti prilikom kreiranja liste premium korisnika. Ona se pojavljuje kao atribut u klasama Nutritionist i Leaderboard, ali nam je u ovisnosti od klase potreban drugačiji poredak, tačnije nutricionista će vidjeti listu sortiranu po imenima i prezimenima, dok je u leaderboard-u potreban poredak po broju ostvarenih poena. U svrhu različitog prikaza liste korisnika kreirat ćemo adapter koji vrši promjenu iz jednog u drugi poredak.

2. ***Facade***

U našem sistemu ćemo imati mogućnost sortiranja/filtriranja podataka po različitim kriterijima. Kako su svi oni usko povezani, a algoritmi koji to implementiraju su u drugom planu, možemo sve metode sortiranja objediniti u jednu tzv. Facade klasu u koju smještamo, na primjer, sortiraj po imenu, godinama, bodovima ili bilo kojoj drugoj kategoriji.

3. *Decorator*

Decorator pattern možemo iskoristiti za bilo kakvo buduće proširivanje klase Person. Na primjer, ako dodamo nove atribute kao što su način života (active/sedentary), alergije, klima, restrikcije u prehrani ili neki drugi podatak relevantan za plan ishrane, nećemo morati izvršiti nikakve drastične izmjene. Uvest ćemo interfejs ExtraInformation() sa metodom edit(), tako da će svaki korisnik imati mogućnost pružanja dodatnih informacija u svrhu generisanja bolje prilagođenog plana ishrane.

4. *Bridge*

Bridge pattern možemo iskoristiti za funkcionalnost kartičnog plaćanja. Kartično plaćanje se veže za banku, a kako različite banke imaju svoje načine uplate i povlačenja sredstava sa računa tako će i u aplikaciji biti potrebno razdvojiti ove načine u posebne metode. Ovo se može implementirati na način da uvedemo interfejs Bank() sa metodom payment(). Interfejs kasnije poziva potrebnu metodu u zavisnosti od banke koja nudi karticu.

5. *Composite*

Composite pattern možemo iskoristiti prilikom dodjeljivanja premium korisnika nutricionisti. Moguće je jednom nutricionisti dodijeliti jednog ili više korisnika. Operacija dodjele se može izvršiti nad jednim korisnikom ili nad više korisnika istovremeno. U svrhu različite implementacije suštinski iste metode(dodjela jednog i dodjela više korisnika) koristimo composite pattern.

6. ***Proxy***

Proxy pattern ćemo iskoristit prilikom LogIn-a na način da ćemo implementirati klasu AutentificationProxy. U njoj ćemo definisati ISubject interfejs, te Subject klasu koja implementira stvarni interfejs. Iz ISubject interfejsa nasljeđujemo ProtetctionProxy klasu koja će zatražiti password od korisnika prije njegovog pristupanja bilo kakvim funkcionalnostima. Također je potrebno izvršiti autentifikaciju podataka koje proslijedi korisnik.

7. *Flyweight*

Možemo proširiti naš sistem sa opcijom za dodavanje profilne slike. Korisnik bi mogao odabrati neku od ponuđenih slika ili upload-ati neku od slika sa vlastitog uređaja. Naravno, on to ne mora uraditi te bi mu se u tom slučaju dodijelila unaprijed zadana slika. Pošto je moguće da se više korisnika odluči upravo za zadanu sliku potrebno je iskoristiti flyweight patern iz razloga što svi koriste isti resurs.

*Boldirani su patterni koje ćemo implementirati u našem projektu