# Single Responsibility Principle

***KLASA BI TREBALA IMATI SAMO JEDAN RAZLOG ZA PROMJENU***

Klasa User sadrži sve osnovne podatke o korisniku te metode koje izračunavaju BMI i TargetCalories. Ove metode služe samo klasi User.

Klasa DailyMealPlan sadrži podatke za dnevni plan ishrane. Pojedinačni podaci o nutrijentima tog dnevnog plana i podacima o jelima se nalaze u odvojenim klasama tako da plan ne ovisi o promjeni tih klasa.

Klasa Raiting opisuje jedan raiting dnevnog plana ishrane. U klasi UserRaiting se nalazi svaki raiting za svaki dnevni plan zajedno sa static metodama koje ne ovise o drugim klasama.

Klasa Person opisuje svaku osobu. Ostle 3 klase Employee, Administrator i User dodaju opise na postojeću baznu klasu i ne zavise međusobno jedna od druge.

# Open Closed Principle

***ENTITETI SOFTVERA (KLASE, MODULI, FUNKCIJE) TREBALI BI BITI OTVORENI ZA***

***NADOGRADNJU, ALI ZATVORENI ZA MODIFIKACIJE.***

Ovaj princip je ispoštovan u svim klasama osim u klasi User jer ova klasa koristi Enumeration tipove podataka. Tako da naprijmjer dodavanje nove stavke u enum tip bi za posljedicu imalo promjenu logike metoda u klasi User.

Svi ovi pobrojani tipovi ili zavise od api-a tako da se ne mogu automatski postaviti jer se mora ručno ući u dokumentaciju api-a i vidjeti koje sve pobrajene tipove nude, ili se neće mijenjati u budućnosti.

# Liskov Substitution Principle

***PODTIPOVI MORAJU BITI ZAMJENJIVI NJIHOVIM OSNOVNIM TIPOVIMA***

Klase Administrator, Employee i User su zamjenljive njihovim podtipom Person jer sve 3 klase predstavljaju osobu.

Dodatno, sve klase koje implementiraju interfejs **ApiCalls**, mogu se koristiti na mjestima gdje se očekuje tip ApiCalls odsnosno u **ApiWrapper** klasi.

# Interface Segregation Principle

***KLIJENTI NE TREBA DA OVISE O METODAMA KOJE NEĆE UPOTREBLJAVATI***

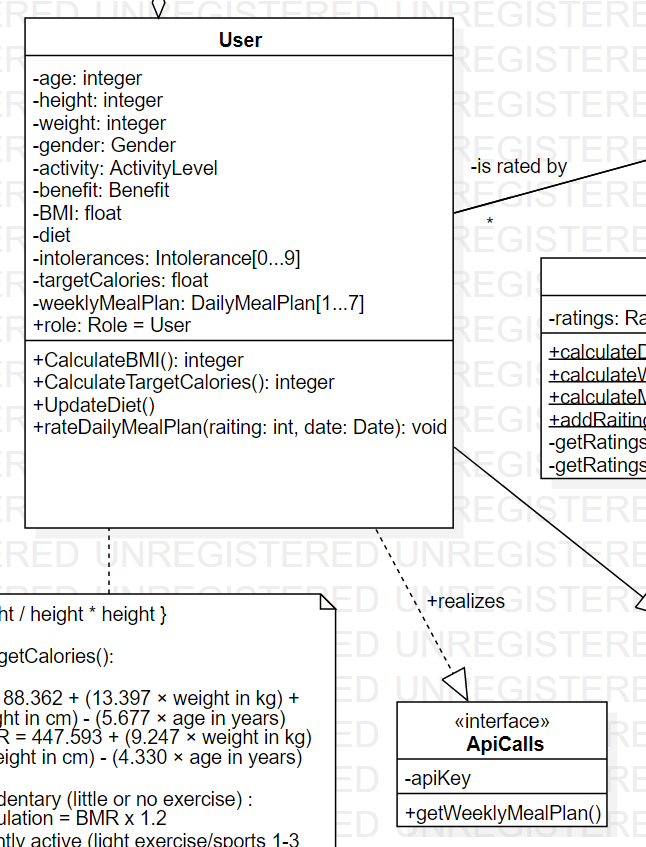
Klasa SpoonacularApi i svaka druga klasa koja možda bude dodana za drugi API implementira interfejs ApiCalls. Ova klasa koristi sve metode definisane u interfejsu tako da ga nije potrebno razdvajati na manje interfejse.

# Dependency Inversion Principle

***A. MODULI VISOKOG NIVOA NE BI TREBALI OVISITI OD MODULA NISKOG NIVOA. OBA BI TREBALO DA OVISE OD APSTRAKCIJA.***

***B. MODULI NE BI TREBALI OVISITI OD DETALJA. DETALJI BI TREBALI BITI OVISNI OD APSTRAKCIJA.***

Ovaj princip nije bio ispoštovan. Ako bi naša aplikacija prestala koristit Spoonacular API i počela koristiti neki drugi, morali bi promijeniti implementaciju interfejsa ApiCalls što je naš high level kod.



Rješenje je da koristimo neki wrapper koje će predstavljati posrednika između klase User i API-a.

Taj wrapper može biti interface koji će koristiti posebne Api klase gdje će svaka klasa imati iste metode ali implementirane na različite načine, kao na slici ispod.

