Kreacijski paterni

Singleton patern

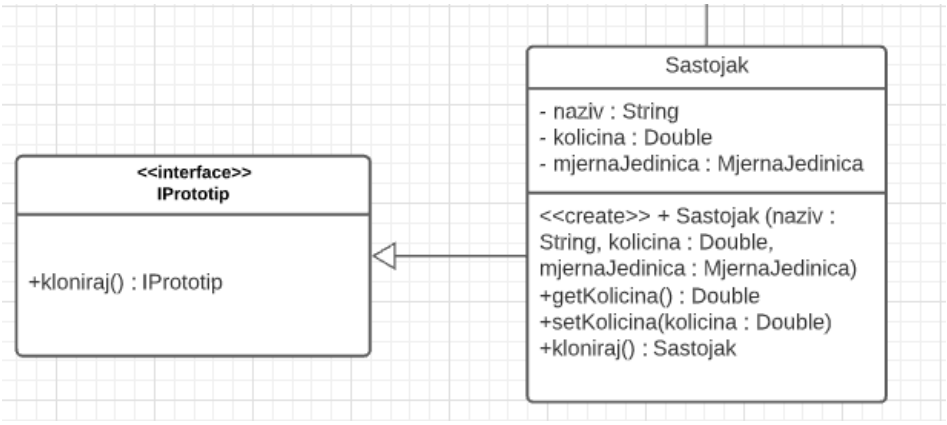
Uloga Singleton paterna je da osigura da se klasa može instancirati samo jednom i da osigura globalni pristup kreiranoj instanci klase.

U našem projektu imamo jednu Singleton klasu, a to je klasa BazaSastojaka. Ova klasa sadrži listu postojećih sastojaka u sistemu, te smo je dodali pri implementaciji Flyweight paterna. Obzirom da nema potrebe za više instanci ove klase, pošto nam je lista svih sastojaka svakako potrebna na jednom mjestu i imamo potrebu da pristupimo ovoj listi (indirektno) bez obzira o kom korisniku, odnosno receptu se radilo. Ovaj patern je implementiran tako da klasa BazaSastojaka sadrži interno kao privatni atribut instancu same sebe, a to je upravo 'statički' atribut kom se pristupa pozivom metode getInstance(). Konstruktor ove klase mora biti privatan i poziva se samo jednom.

Prototype patern

Uloga Prototype paterna je da kreira nove objekte klonirajući jednu od postojećih prototip instanci (postojeći objekat).

Ovaj patern ćemo iskoristiti za kreiranje plitkih kopija Sastojak prilikom upotrebe Flyweight paterna. Dakle, ukoliko pozivom metode dodjaUBazuSastojaka ustanovimo da već postoji, suštinski isti, sastojak u listi sastojaka, onda će se kreirati plitka kopija tog sastojka i proslijediti kao povratna vrijednost. Na taj način dobit će se potrebne vrijednosti atributa koji se neće mijenjati (mjerna jedinica i naziv), a naknadno korisnik može izmijeniti atribut kolicina. Kako bi implementirali ovaj patern potrebno je kreirati interface IPrototip koji sadrži metodu kloniraj(). Ovaj interface će implementirati klasa Sastojak, te će se unutar te klase ova metoda implementirati na način da se kopiraju vrijednosti svih atributa u novo-kreiranu instancu Sastojak.



Factory Method patern

Uloga Factory Method paterna je da omogući kreiranje objekata na način da podklase odluče koju klasu instancirati.

Recimo da dopuštamo korisniku da prilikom registracije odabere da li je vegan ili ne, te na osnovu toga sve recepte koje kreira KorisnikVegan označavamo kao veganske, u suprotnom, ako ih je kreirao KorisnikNeVegan, oni neće biti označeni kao veganski. Za implementaciju ove funkcionalnosti bi se mogli poslužiti Factory Method paternom. Nove klase KorisnikVegan i KorisnikNeVegan bi nasljeđivale kreator klasu odnosno Korisnik klasu (koja bi u ovom slučaju bila apstraktna, jer sa naslijeđenim klasama bi se pokrivale sve vrste registrovanih korisnika). Dodali bi i interface IRecept koji bi nasljeđivali konkretni produkti odnosno klase VeganskiRecept i NeveganskiRecept. Tako bi kreatorska klasa prilikom dodavanja/kreiranja novog recepta, kreirala instancu odgovarajuće klase recepata, na osnovu klase korisnika koji upravo kreira taj recept.

Abstract Factory patern

Abstract Factory patern omogućava da se kreiraju familije povezanih objekata/produkata. Na osnovu apstraktne familije produkata kreiraju se konkretne fabrike produkata različitih tipova i različitih kombinacija.

Kada bi imali već implementiran prethodno opisan Factory Method patern, mogli bi ga proširiti u Abstract Factory patern dodavanjem novih vrsta korisnika baziranih na preferiranom tipu ishrane, te bi ujedno i dodali nove klase za recepte koji se uklapaju u te tipove ishrane (npr. Keto, Pesketerijanizam, Dijabeticarski itd.). Osim toga mogli bi klasificirati i recepte na osnovu kuhinja kojima pripadaju (npr. Italijanska, Francuska, Grčka itd.), te bi ponudili korisnicima da prilikom registraciju odaberu i ove preference. Prilikom kreiranja recepata automatski bi se na osnovu korisnikovih preferenci tj. klase korisnika kreirale odgovarajuće klase recepata.

Builder patern

Uloga Builder paterna je odvajanje specifikacije kompleksnih objekata od njihove stvarne konstrukcije. Isti konstrukcijski proces može kreirati različite reprezentacije.

Obzirom da u našoj aplikaciji nema izuzetno kompleksnih objekata, nismo vidjeli potrebu za implementacijom ovog paterna. Međutim, jedan od načina njegove implementacije bio bi da kreiramo Builder interface koji bi služio za 'izgradnju' jednog recepta. Ponovo, imali bi različite klase za recepte na osnovu vrste jela i kuhinji kojoj pripadaju (FrancuskoSlano, FrancuskoSlatko, GrckoSlatko, GrckoSlano itd.) i za svaku od tih klasa imali bi i odgovarajuću Builder klasu koja će izgraditi tu vrstu recepta. Takve recepte bi onda mogli kategorisati i prikazati u različitim tabovima/kategorijama na Home pogledu.