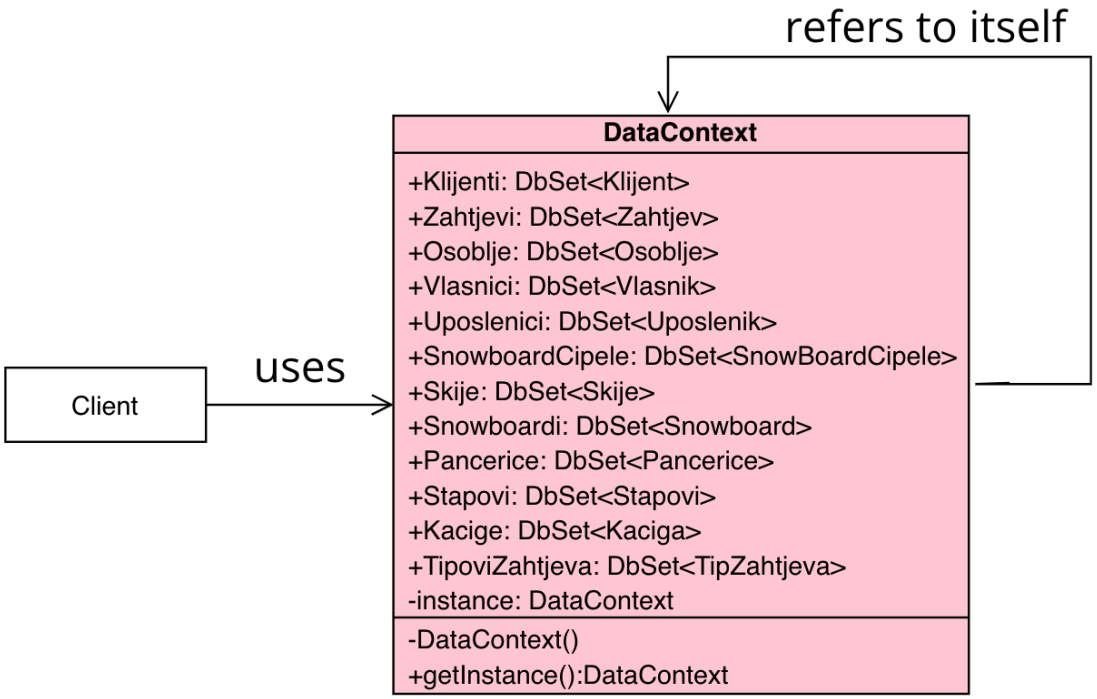
Kreacijski design patterni

Za implementaciju odabrani su Singleton i Prototype design patterni.

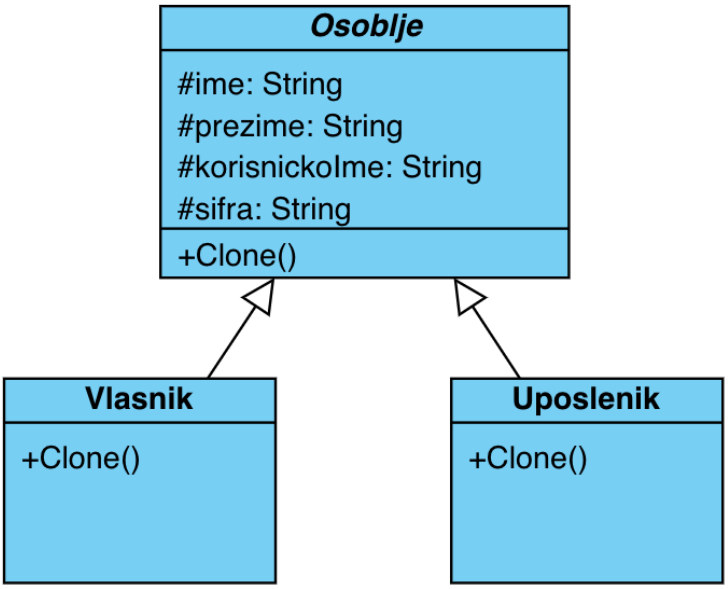
1. Singleton

Uloga Singleton patterna je da osigura da se klasa može instancirati samo jednom i da osigura globalni pristup kreiranoj instanci klase. U našem sistemu bismo mogli iskoristiti ovaj pattern pri pristupu bazi podataka, kako bi riješili problem mogućeg istovremenog pristupa od strane vlasnika i uposlenika. Klasa koja bi nam omogućila pristup bazi bi bila singleton.



2. Prototype

Uloga Prototype paterna je da kreira nove objekte klonirajući jednu od postojećih prototip instanci. Ako je trošak kreiranja novog objekta velik i kreiranje objekta je resursno zahtjevno tada se vrši kloniranje već postojećeg objekata. U našem sistemu ovaj pattern možemo iskoristiti nad klasom Osoblje. Mogli bismo iskoristiti prvu instancu klase, klonirati je a zatim promijeniti podatke koji se razlikuju. Klase implementiraju IPrototype interfejs i metodu Clone.



3. Builder

Uloga Builder paterna je odvajanje specifikacije kompleksnih objekata od njihove stvarne konstrukcije. Isti konstrukcijski proces može kreirati različite reprezentacije. U našem sistemu možemo ovaj pattern primijeniti za konstrukciju klase Zahtjev. Napravili bi „Direktor“ klasu koja bi definirala redoslijed pozivanja koraka konstrukcije. Također, trebali bi dodati builder interfejs koji bi sadržavao metode za dodavanje potrebnih informacija i builder klase koje mi implementirale ovaj interfejs.

4. Abstract factory

Abstract Factory patern omogućava da se kreiraju familije povezanih objekata/produkata. Ukoliko postoji više tipova istih objekata te različite klase koriste različite podtipove, te klase postaju fabrike za kreiranje objekata zadanog podtipa bez potrebe za specificiranjem pojedinačnih objekata. Na ovaj način se, korištenjem nasljeđivanja, ukida potreba za postojanjem if-else uslova jer određeni tip fabrike sadrži određene tipove objekata i zna se tačno koju podklasu će instancirati. U našem sistemu ovaj pattern moguće je iskoristiti za klasu Oprema gdje bi to bila Abstract Factory a izvedene klase bi bile Concrete Factories.

5. Factory method

Uloga Factory Method paterna je da omogući kreiranje objekata na način da podklase odluče koju klasu instancirati. Različite podklase mogu na različite načine implementirati interfejs. Factory Method instancira odgovarajuću podklasu(izvedenu klasu) preko posebne metode na osnovu informacije od strane klijenta ili na osnovu tekućeg stanja. U našem sistemu je ovaj pattern moguće iskoristiti kod sistema personaliziranih preporuka gdje samo na osnovu informacije o visini klijenta instanciraju se odgovarajući objekti.