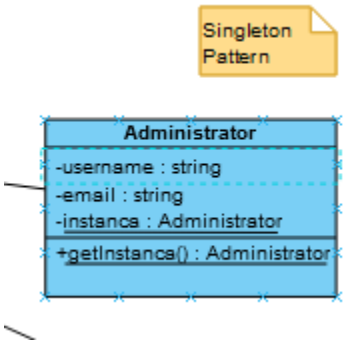
Kreacijski patterni

1. Singleton patern

Singleton patern se koristi kada želimo uvijek imati jedinstveni objekat neke klase. Obično se koristi pri kreiranju jedinstvene konekcije na bazu podataka. Implementacija ovog patterna obično rezultira većom brzinom pristupa, korištenjem *cache* memorije.

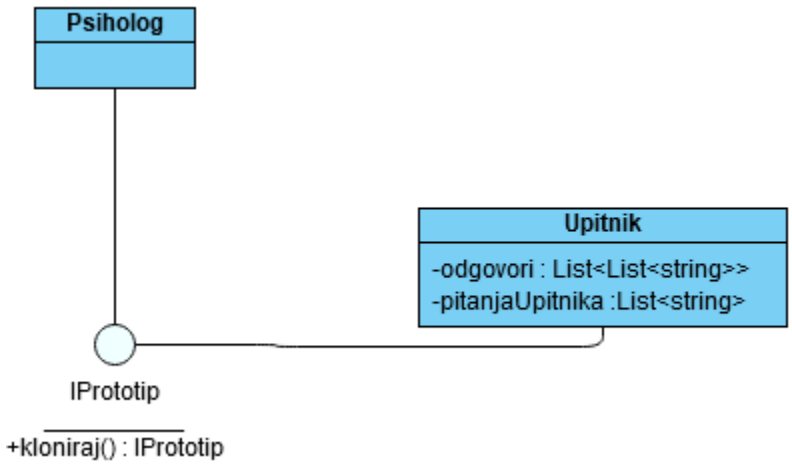
Ovaj pattern možemo implementirati u skladu sa klasom *Administrator*, pošto u našem sistemu može postojati samo jedna instanca te klase (jedan administrator čitavog sistema):



2. Prototip

Ovaj pattern se koristi za olakšavanje procesa kreiranja sličnih instanci, pravljenje konfiguracija za kloniranje objekata koje posjeduju privatne atribute.

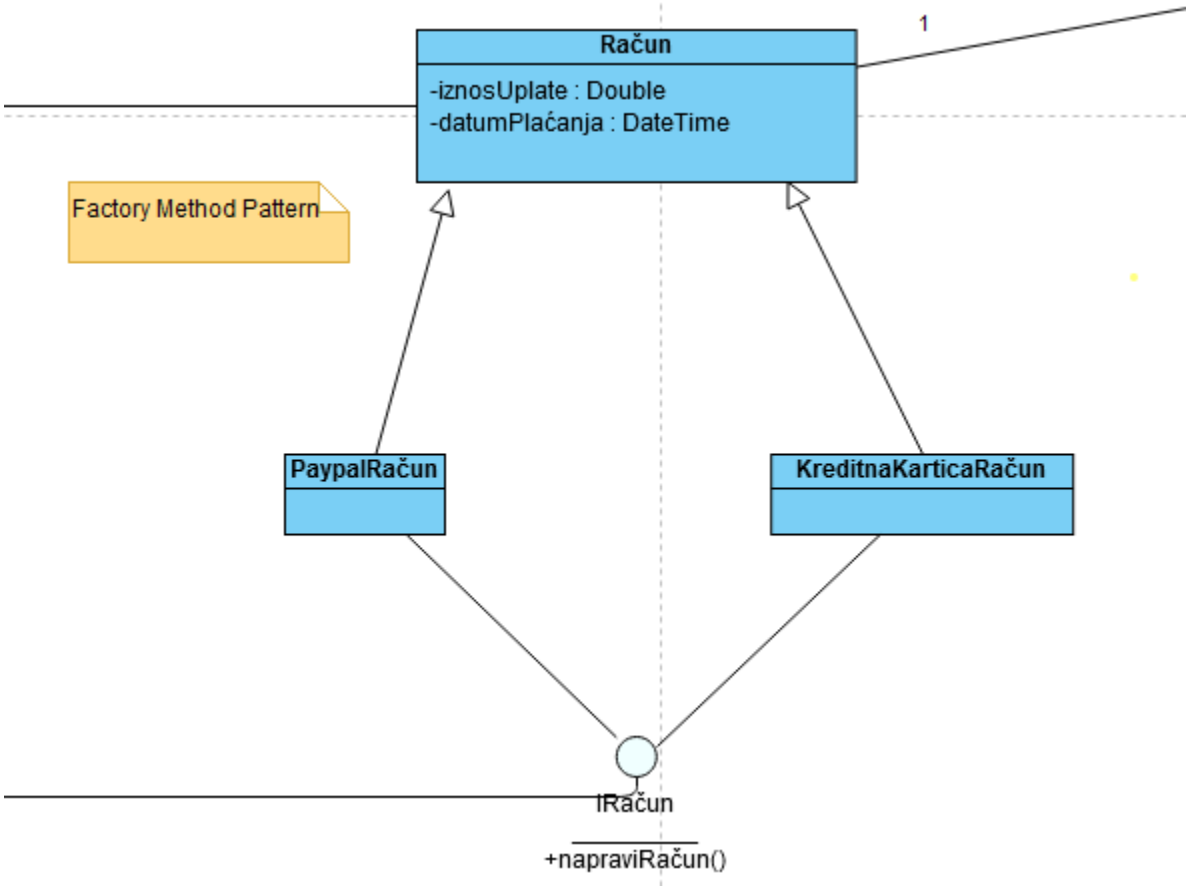
Prototip pattern nije predviđen za implementaciju, ali bi se mogao dodati bez većih poteškoća. Na primjer, ako bi psiholog imao mogućnost kreiranja više različitih upitnika, to bi se moglo izvesti putem interfejsa *IPrototip* koji bi omogućavao kreiranje različitih upitnika za različite dijagnoze, koji bi sadržavali različit broj pitanja, ali koji bi u osnovi bili Upitnici.



3. Factory method pattern

Ovaj patern se koristi za instanciranje različitih objekata koji nude iste mogućnosti koristeći istu metodu. On također omogućava interfejs za kreiranje objekata u nadklasi, ali dozvoljavajući da podklase mijenjaju tip objekata koji će se stvoriti.

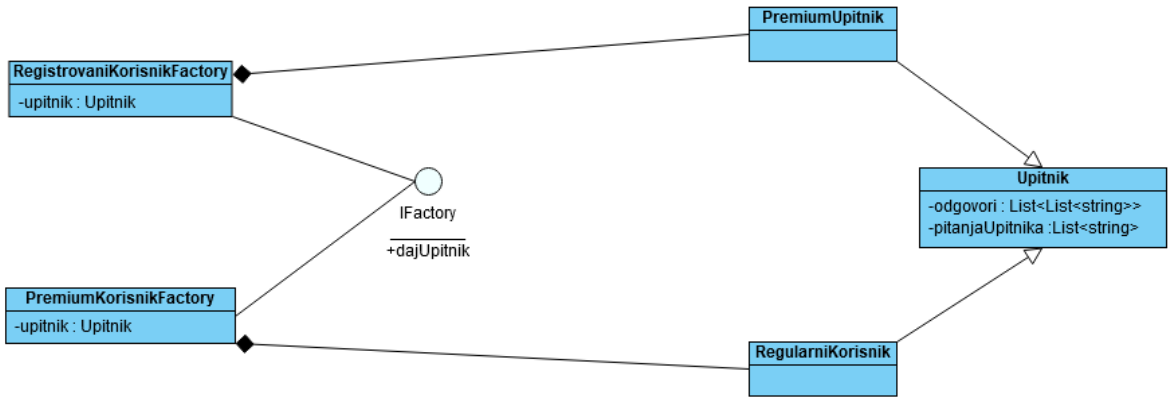
U našoj aplikaciji, ovaj pattern je najpogodnije bilo implementirati kod klase *Račun*. Pošto postoje dva načina plaćanja računa, iz klase Račun smo naslijedili klase *PaypalRačun* i *KreditnaKarticaRačun*, pri čemu kontrolerska klasa *TerapeutController* implementira interfejs *IRačun*.



4. Abstract factory pattern

Abstract factory pattern je najpogodnije koristiti za kreiranje više familija objekata sa mogućnošću dodavanja novih u budućnosti Sve familije objekata nude iste mogućnosti, ali različite instance. To u suštini predstavlja stvaranje srodnih objekata, bez navođenja njihovih konkretnih klasa.

Naš sistem ne predviđa implementaciju ovog patterna, ali bi se on mogao izvesti na sljedeći način. Ukoliko bi postojale dvije vrste upitnika, jedan za registrovanog korisnika, a drugi za premium korisnika, oba bismo mogli naslijediti iz klase *Upitnik* te povezati sa prikladnim Factory klasama.



5. Builder pattern

Omogućavanje primene različitih postupaka za konstrukciju istog objekta. Koristi se i za pravljenje predefinisanih konfiguracija za objekte. On nam dozvoljava izradu različitih tipova i zastupanja objekata koristeći isti programski kod.

Naš sistem implementira ovaj pattern na sljedeći način: