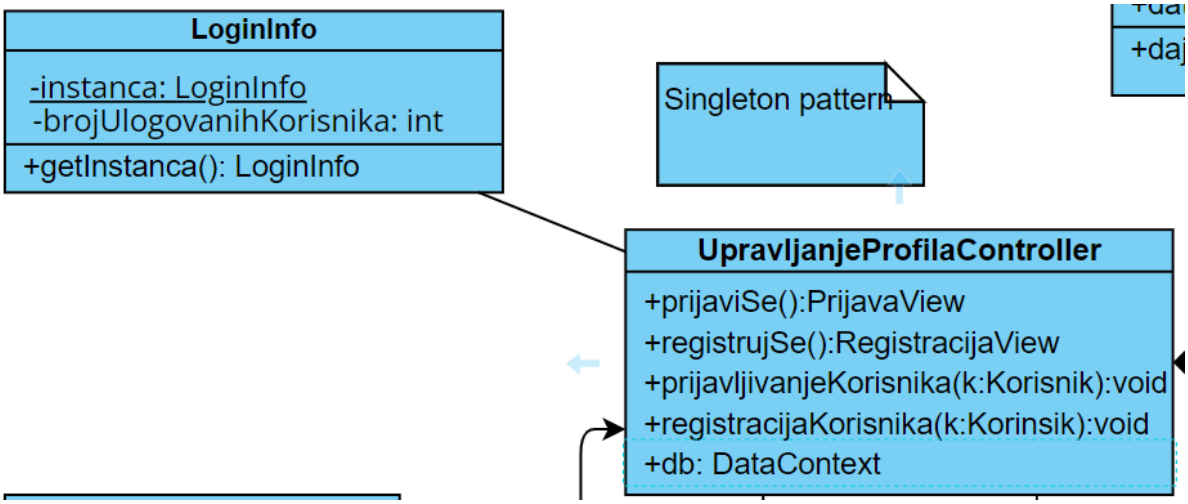
KREACIJSKI PATERNI

1. Singleton patern

U našem sistemu dodali smo klasu LogInInfo koja je realizovana kao singleton klasa.

Primjenom ovog paterna izbjegavamo instanciranje više istih objekata. Time smo dobili uštedu memorije i prirodniji pogled na sistem.

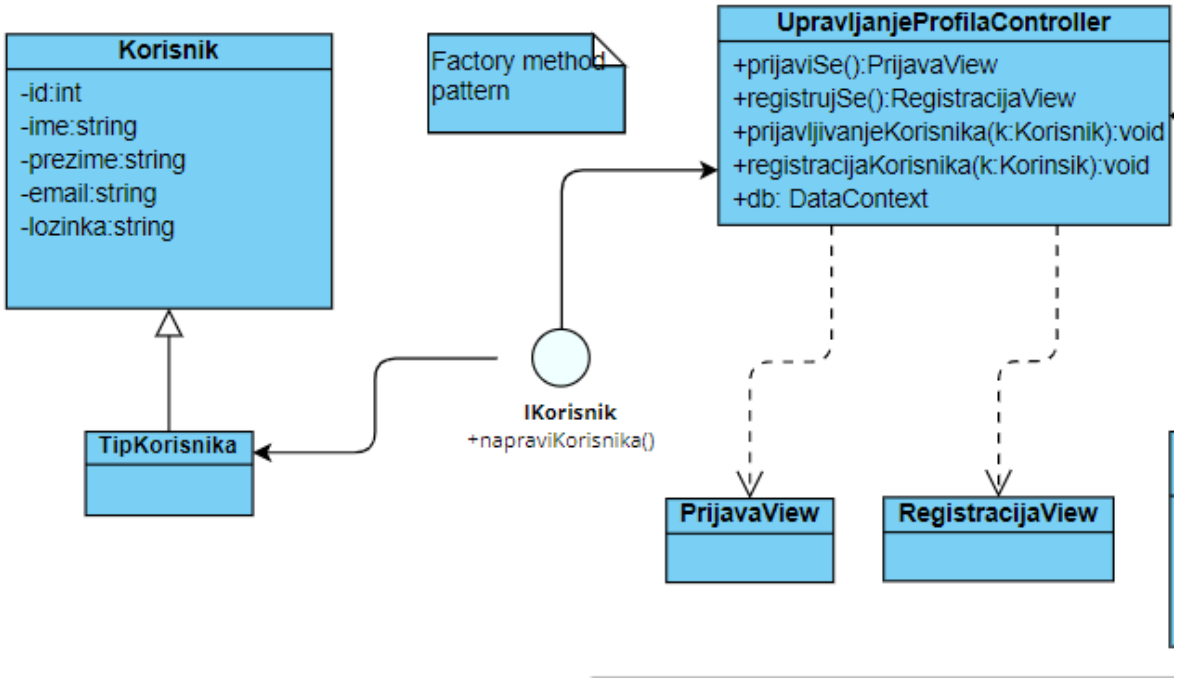
Dodan je static atribut instanca tipa LogInInfo kao i metoda getInstanca().



2. Factory method pattern

Factory method pattern je u našem sistemu iskorišten za različite tipove klase Korisnika.

Na naš dijagram klasa se dodaje interfejs koji smo nazvali IKorisnik. Zatim tu imamo već postojeće klase Korisnik i TipKorisnika. Klasa TipKorisnika implementira taj interfejs. Metoda napraviKorisnika() će se različito implementirati za različite vrste korisnika.



3. Abstract factory pattern

Ovaj patern se koristi pri radu sa familijama sličnih produkata, kao i ukoliko postoji više tipova istih objekata te različite klase koriste različite podtipove. Kako mi u našem sistemu ne posjedujemo familije sličnih produkata jer ne postoji nikakva ponuda koja vodi u tom smislu, ovaj patern nije moguće primijeniti.

Ukoliko bi postojale više vrsta knjiga, neke koje su dostupne samo za Korisnika, a druge za SuperKorisnika sve bismo mogli naslijediti iz klase Knjiga te povezati sa prikladnim Factory klasama.

4. Prototype pattern

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

Jedan od slučajeva u kojem bismo mogli upotrijebiti ovaj patern jeste slučaj kada bi željeli da podržimo novo izdanje istog autorskog djela. Bilo bi potrebno napraviti novu instancu klase, koja bi imala iste podatke kao i već objavljeno autorsko djelo. Naknadno bismo ažurirale atribut datum.

5. Builder pattern

Builder patern služi za apstrakciju procesa konstrukcije objekta kako bi se kao rezultat mogle dobiti različite specifikacije objekta koristeći isti proces konstrukcije.

Kada bi u našem sistemu imali različite vrste knjiga mogli bismo iskoristiti ovaj patern da spriječimo pretjerano nasljeđivanje. Vrste knjiga koje bismo imale su: biografija (dodatni atribut bi bilo ime osobe), naučni rad (dodatni atribut je naučna oblast) i rječnik (dodatni atribut je jezik).

Dodale bismo Ibuilder interfejs unutar kojeg bi se nalazile metode dodajiImeOsobe(ime: String), dodajOblast(oblast: String), dodajJezik(jezik: String).

Zatim bismo napravili builder klase: BuilderBiografija, BuilderNaucniRad i BuilderRjecnik koje bi u sebi imale atribut Knjiga i pri kreiranju nove knjige kombinovali bi razne pozive ovih metoda (neki bi pozvali samo određene).