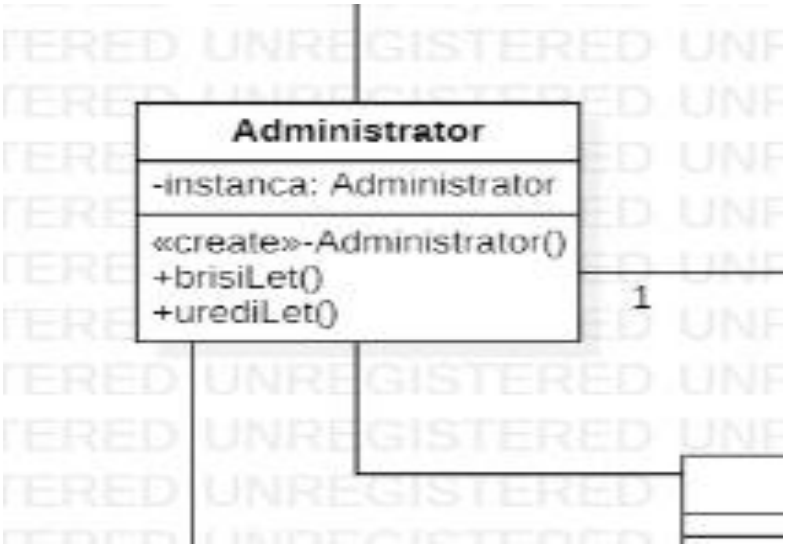
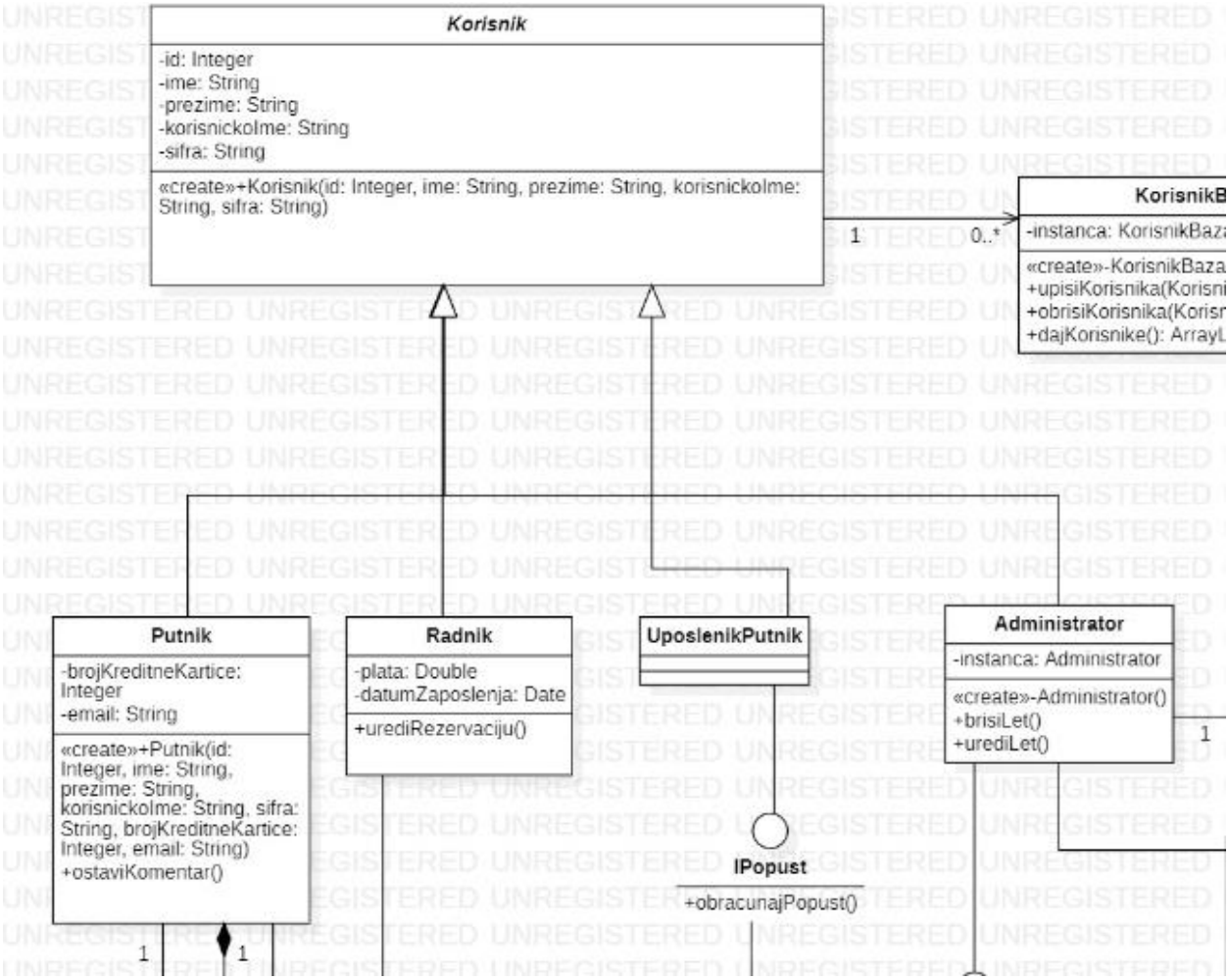
Kreacijski paterni

- **Singleton patern** - uloga Singleton paterna je da osigura da se klasa može instancirati samo jednom i da osigura globalni pristup kreiranoj instanci klase. Instanciranjem više od jednog objekta mogu nastati problemi. Singleton klasa sadrži mehanizam za jedinstveno instanciranje same sebe. Unutar klase je private static varijabla (uniqueInstance) koja čuva jednu/jedinstvenu instancu klase, static metoda (getInstance) preko koje se pristupa Singleton klasi. Važan dio Singleton paterna je inicijalizacija resursa u Singleton konstruktoru. U našem sistemu Singleton patern je implementiran u klasi Administrator tako što smo dodali private static konstruktor i private static varijablu (jedinstvenaInstanca) koja čuva jednu/jedinstvenu instancu klase. Također sve klase za rad sa bazama podataka su singleton, u suprotnom bi imali problema sa drajverima na bazu.



- **Abstract factory patern** – ovaj patern služi kako bi se izbjeglo korištenje velikog broja if-else uslova pri kreiranju različitih hijerarhija objekata. 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. Ovaj patern smo implementirali kod klasa Korisnik, Putnik, Radnik i Administrator.

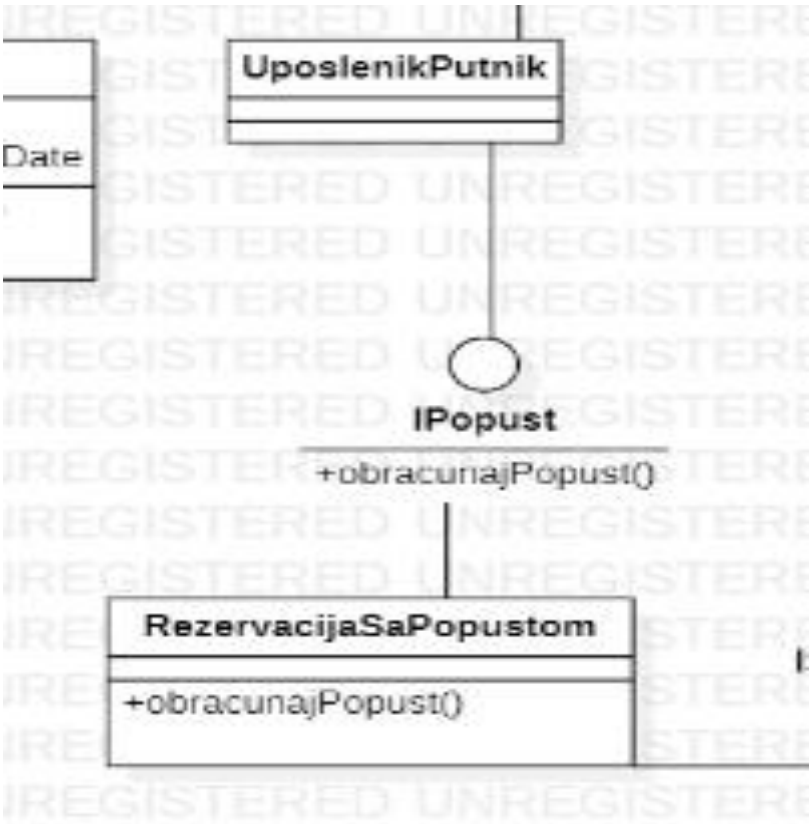


- **Prototype patern** - uloga Prototype paterna je da kreira nove objekte klonirajući jednu od postojećih prototip instanci. Time smanjujemo kompleksnost kreiranja novog objekta, posebno kada objekti sadrže veliki broj atributa koji su za većinu instanci isti. Ovaj patern nismo primijenili. Ali bismo to mogli uraditi za kloniranje rezervacije leta uvođenjem interfejsa IPrototip, koji sadrži metodu clone(). Ova metoda će kreirati novu rezervaciju sa istim atributima kao i rezervacija nad kojom je pozvana ova metoda.

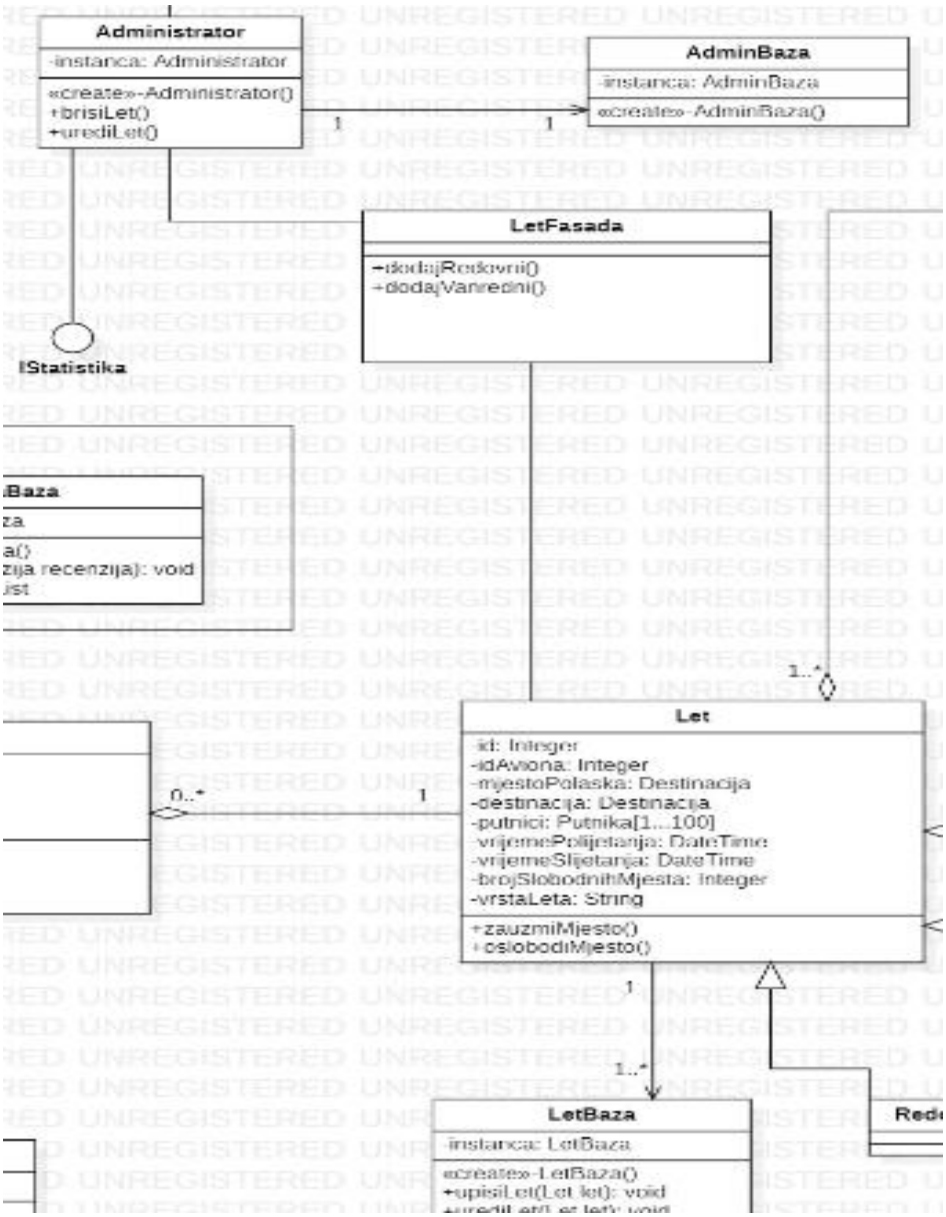
Strukturalni paterni

- **Adapter patern** – osnovna uloga Adapter paterna je da omoguči širu upotrebu već postojećih klasa. U situacijama kada je potreban drugačiji interfejs već postojeće klase, a ne želimo mijenjati postojeću klasu koristi se Adapter patern. Adapter patern kreira novu adapter klasu koja služi kao posrednik između originalne klase i željenog interfejsa. Tim postupkom se dobija željena funkcionalnost bez izmjena na originalnoj klasi i bez

ugrožavanja integriteta cijele aplikacije. Adapter patern, u našoj aplikaciji smo implemetirali kreiranjem interfejsa IPopust sa metodom obracunajPopust() i dodavanjem klasa UposlenikPutnik i RezervacijaSaPopustom. Tako omogućavamo radniku aviokompanije pravo na popust bez izmjene postojećeg objekta.



- **Fasade patern** - fasadni patern služi kako bi se klijentima pojednostavilo korištenje kompleksnih sistema. Klijenti vide samo fasadu, odnosno krajnji izgled objekta, dok je njegova unutrašnja struktura skrivena. Na ovaj način smanjuje se mogućnost pojavljivanja grešaka jer klijenti ne moraju dobro poznavati sistem kako bi ga mogli koristiti. U našem dijagramu smo ovaj patern implementirali kod kreiranja leta u administratorskoj klasi. Klasa administrator je klijent I ona putem metoda kreirajRedovniLet() i kreirajVandredniLet() iz klase LetFasada dodaje letove u bazu.


Administrator
-instanca: Administrator
«create»-Administrator()
+brisiLet()
+urediLet()
AdminBaza
-instanca: AdminBaza
«create»-AdminBaza()
1
1
LetFasada
+dodajRedovni()
+dodajVanredni()
IStatistika
Baza
za
a()
zija recenzija): void
ist
1..*
Let
-id: Integer
-idAviona: Integer
-mjestoPolaska: Destinacija
-destinacija: Destinacija
-putnici: Putnika[1...100]
-vrijemePolijetanja: DateTime
-vrijemeSlijetanja: DateTime
-brojSlobodnihMjesta: Integer
-vrstaLeta: String
+zauzmiMjesto()
+oslobodiMjesto()
0..*
1
1
1..*
LetBaza
-instanca: LetBaza
«create»-LetBaza()
+upisiLet(Let let): void
Redo