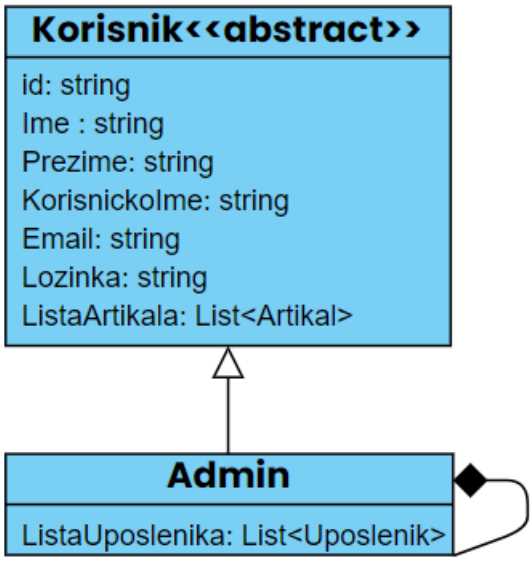
Kreacijski paterni

Singleton pattern

Uloga Singleton paterna je da osigura da se klasa može instancirati samo jednom i da osigura globalni pristup kreiranoj instanci klase. Postoji više objekata koje je potrebno samo jednom instancirati i nad kojim je potrebna jedinstvena kontrola pristupa. Neki od njih su: thread pools (skupina niti), objekti koji upravljaju setovanjem registara, objekti koji se koriste za logiranje, objekti koji se koriste kao drajveri za razne uređaje kao što su printeri i grafičke kartice. Instanciranje više nego jednom navedenih objekata mogu se prouzrokovati problemi kao što su nekorektno ponašanje programa, neadekvantno korištenje resursa ili nekonzistentan rezultat. Najčešće se koristi za uspostavljanje jedinstvene konekcije na bazu podataka, te ubrzava pristup korištenjem keš memorije.

Primjena:

U našem sistemu klasu Administrator proglasavamo kao Singleton. Administrator je jedan i nema potrebe da se često instancira.



Abstract factory pattern

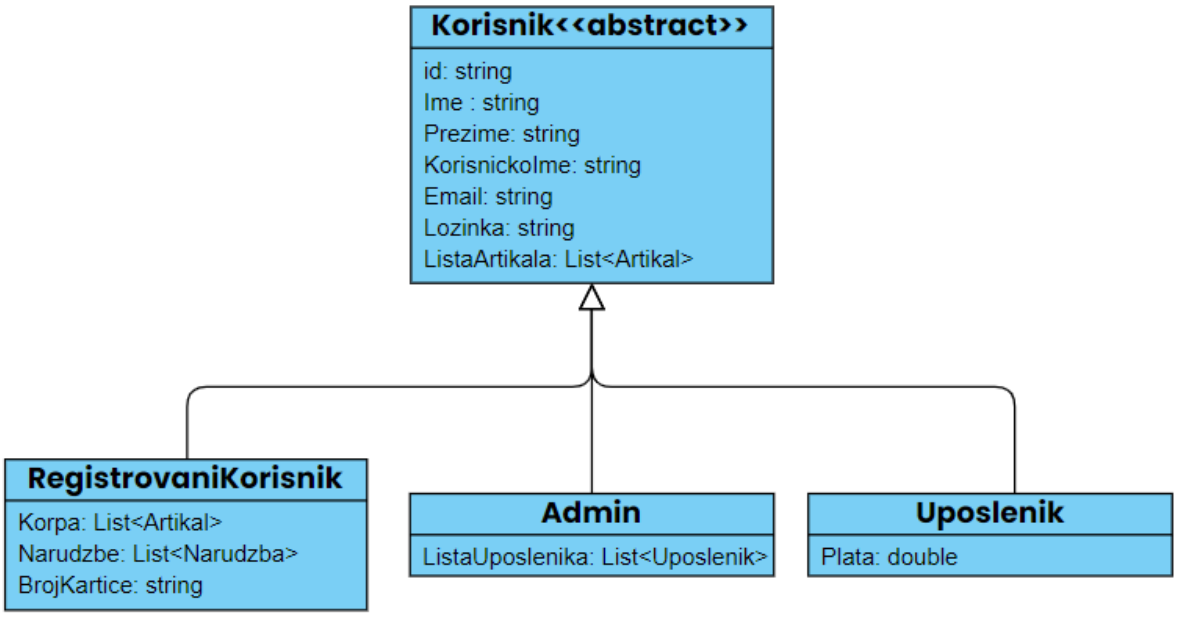
Abstract Factory patern omogućava da se kreiraju familije povezanih objekata/produkata. Na osnovu apstraktne familije produkata kreiraju se konkretne fabrike (factories) produkata različitih tipova i

različitih kombinacija. Patern odvaja definiciju (klase) produkata od klijenta. Zbog toga se familije produkata mogu jednostavno mijenjati ili ažurirati bez narušavanja strukture klijenta. Važan aspekt Abstract Factory paterna je da se cijela familija (fabrika) produkata može promijeniti za vrijeme izvršavanja aplikacije zbog toga što produkti implementiraju isti apstraktni interfejs. Interfejsi operacija za pojedine fabrike su isti ali njihova implementacija se razlikuje. To omogućava da sistem bude neovisan od toga kako su produkti kreirani, sastavljeni i implementirani

Primjena:

U našem sistemu Abstract factory pattern implementiran je nad klasom Korisnik gdje imamo nasljeđivanja u smislu da klase koje nasljeđuju klasu Korisnik predstavljaju razlicite tipove korisnika. Imamo klase Registrovanikorisnik – korisnik koji ima mogućnost kupovine i postavljanj feedbacka, klasa Uposlenik – ima mogućnosti dodavanje i editovanja artikala, klasa Admin – ima mogućnost dodavanje novih uposlenika

Sve navedene klase imaju mogućnost pregled, filtriranja artikala.



Prototype pattern

Uloga Prototype paterna je da kreira nove objekte klonirajući jednu od postojećih prototip instanci (postojeći objekat). Ako je trošak kreiranja novog objekta velik i kreiranje objekta je resursno zahtjevno tada se vrši kloniranje već postojećeg objekata. Prototype dizajn patern dozvoljava da se kreiraju prilagođeni objekti bez poznavanja njihove klase ili detalja kako je objekat kreiran.Prototype patern se koristi kada je potrebno da se sakriju konkretne klase od klijenta, dodaju ili izbrišu nove klase za vrijeme izvršavanja, da se broj klasa u sistemu održi na minimumu, kada je potrebna promjena strukture podataka za vrijeme izvršavanja. Composite i Decorator paterni često imaju prednosti od Prototype paterna. Prototype patern je često koristan i prilikom višestrukog korištenja podataka iz baze. Ako je

potrebno uraditi i druge analize nad istim skupom podataka nije dobra ideja ponovo pristupati bazi podataka, čitati podatke i kreirati objekat za njihovo smještanje.

Primjena:

Prilikom pregleda artikala moglo bit doci do znatnog usporavanja aplikacije izazvanim umanjenjem performansi pri dobivanju informacija iz baze podataka. Navedeni problem mogli bi riješiti primjenom prototip paterna koji će omogućiti da već učitane podatke recikliramo, odnosno da ih jednostavno kloniramo te ponovno iskoristimo.

Factory method patern

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

Primjena:

Factory method pattern mogli bi primijeniti kod plaćanja tako sto bi omogucili vise nacina placanja (karticno, paypal ...). U klasi Naplata trebali bi imat metodu koja će odlučivati koju klasu instancirati. (trebali bi dodati odgovarajuce klase za sve nacine naplate).

Builder patern

Uloga Builder paterna je odvajanje specifikacije kompleksnih objekata od njihove stvarne konstrukcije. Isti konstrukcijski proces može kreirati različite reprezentacije. Upotreba Builder paterna se često može naći u aplikacijama u kojima se kreiraju kompleksne strukture. Koristi se kada je neovisan algoritam za kreiranje pojedinačnih dijelova, kada je potrebna kontrola procesa konstrukcije, kada se više objekata na različit način sastavlja od istih dijelova. Omogućuje konstrukciju istog objekta primjenom razlicitih postupaka.

Primjena:

Kada bi u sistemu imali pakete artikala, odnosno vise artikala grupisanih kao jedan (npr. hot wheels staza + 2 autica), mogli bi uraditi da imamo dva način kreiranja narudžbi: automatski (predefinirani paket) i mogućnost manuelnog sastavljanja narudžbu (izbor autica).