STRUKTURALNI PATERNI

ADAPTER PATTERN

Namjena: Osnovna namjena Adapter paterna je da omogući 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.

- *Unutar aplikacije CryptoInvest, Adapter patern bi mogao biti iskorišten u okviru klase Novcanik, gdje bismo omogućili da metoda getValute(), vraća valute u sortiranom redoslijedu, pri čemu bi navedeno sortiranje mogli ostvariti ili po cijeni, ili po nazivu. Na taj način klasa Novcanik bi bila nadograđena novom funkcionalnosti, pri čemu postoji mogućnost daljeg nadograđivanja. Važno je napomenuti da navedena klasa nije promjenjena, već samo nadograđena.*

FACADE PATTERN

Namjena: Façade patern se koristi kada sistem ima više identificiranih podsistema (subsystems) pri čemu su apstrakcije i implementacije podsistema usko povezane. Osnovna namjena Facade paterna je da osigura više pogleda visokog nivoa na podsisteme (implementacija podsistema skrivena od korisnika).

- *Unutar aplikacije CryptoInvest, ne postoji naročita potreba za primjenom Facade paterna, zbog toga što je interakcija između korisnika i sistema jasno definisana. U slučaju da ipak želimo primjeniti navedeni patern, jedna od mogućnosti primjene bi bila eventualno na rangiranje, jer to predstavlja jedan od procesa koji nisu bliski korisniku.*

DECORATOR PATTERN

Namjena: Osnovna namjena Decorator paterna je da omogući dinamičko dodavanje novih elemenata i ponašanja (funkcionalnosti) postojećim objektima. Objekat pri tome ne zna da je urađena dekoracija što je veoma korisno za ponovnu upotrebu komponenti softverskog sistema.

- *Unutar aplikacije CryptoInvest, primjena Decorator paterna bi mogla biti ostvariva u okviru pristupa Kursevima odnosno Testiranju znanja, gdje bi korisnik mogao postavljati fotografije urađenih testova, te bi tom prilikom imao mogućnost rotiranja te rezanja fotografija.*

PROXY PATTERN

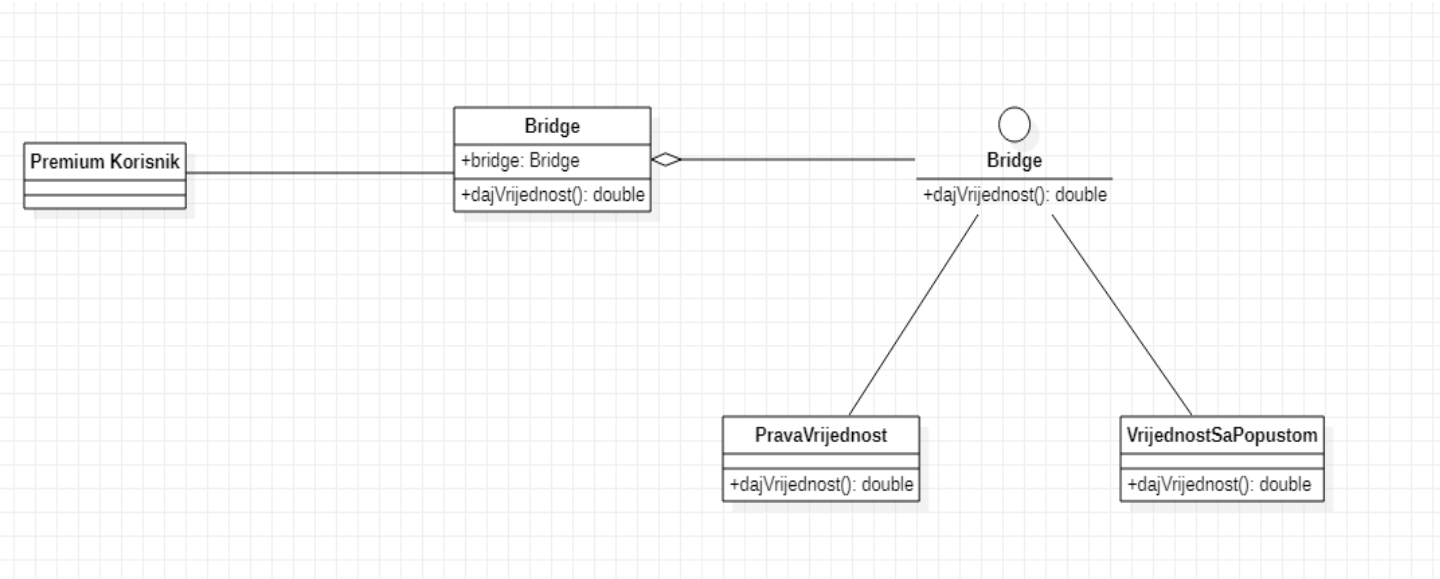
Namjena: Namjena Proxy paterna je da omogući pristup i kontrolu pristupa stvarnim objektima. Proxy je obično mali javni surogat objekat koji predstavlja kompleksni objekat čija aktivizacija se postiže na osnovu postavljenih pravila. Proxy patern rješava probleme kada se objekat ne može instancirati direktno (npr. zbog restrikcije prava pristupa).

- *Unutar aplikacije CryptoInvest, Proxy patern je korišten na način da postoje restrikcije u okviru prijave na profil. U trenutku kada korisnik unese neispravnu lozinku, pristup profilu je onemogućen. U slučaju da je lozinka ispravno unesena, pristup profilu je odobren.*

BRIDGE PATTERN

Namjena: Osnovna namjena Bridge paterna je da omogući odvajanje apstrakcije i implementacije neke klase tako da ta klasa može posjedovati više različitih apstrakcija i više različitih implementacija za pojedine apstrakcije. Bridge patern pogodan je kada se implementira nova verzija softvera, a postojeća mora ostati u funkciji.

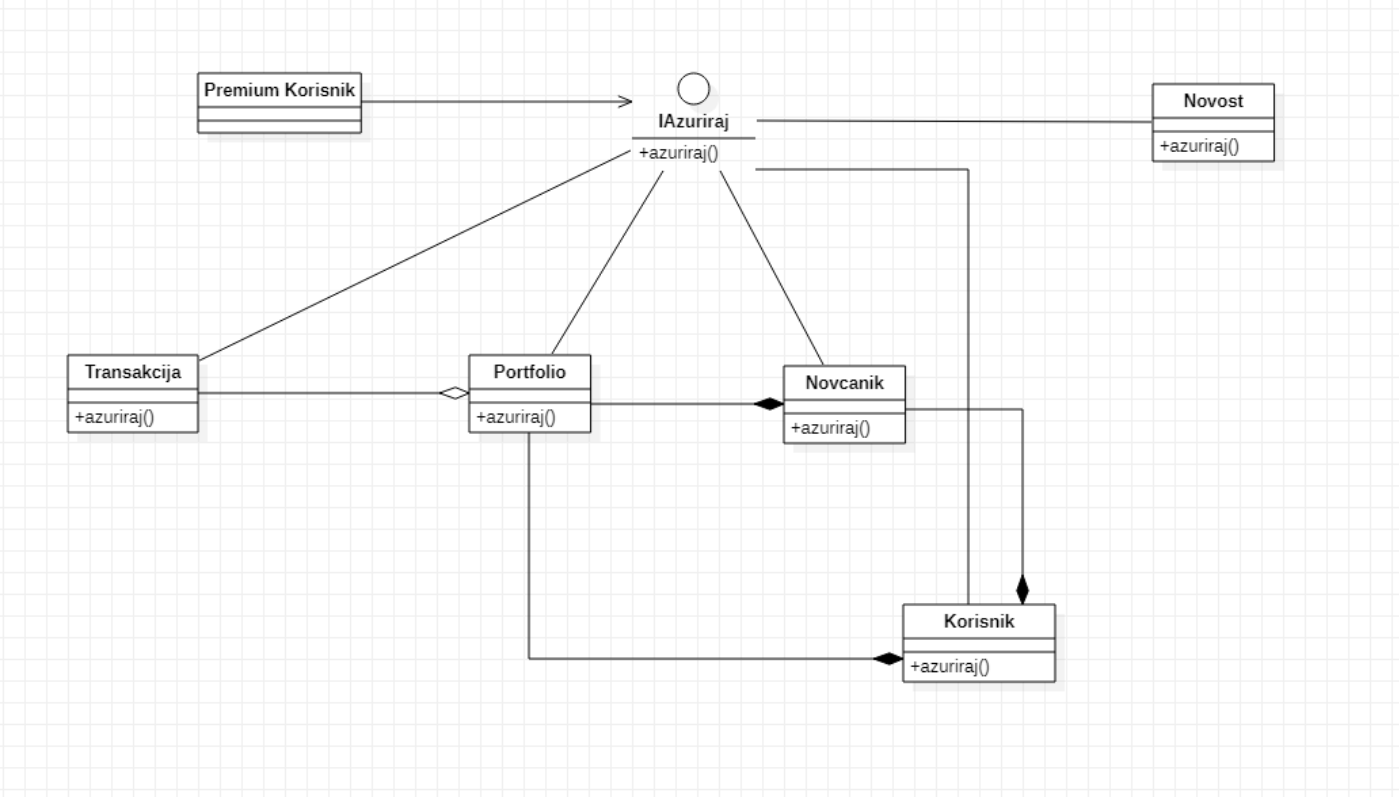
- *U našoj aplikaciji Bridge patern bi se mogao implementirati u slučaju obračunavanja popusta za Premium korisnike, te ćemo tu implementaciju i napraviti. Na taj način ćemo odvojiti regularno plaćanje od plaćanja sa popustom, pri čemu smo ostvarili implementaciju nove verzije plaćanja, dok postojeća ostaje u funkciji. Ono što ćemo uraditi jeste dodati interfejs Bridge, koji će imati metodu dajVrijednost(), koja predstavlja trenutnu vrijednost. Prva implementacija posjeduje metodu dajVrijednost() koja će vraćati regularnu vrijednost paketa, dok će druga implementacija imati metodu dajVrijednost() ali će ta metoda vraćati vrijednost sa obračunatim popustom.*


Premium Korisnik
Bridge
+bridge: Bridge
+dajVrijednost(): double
Bridge
+dajVrijednost(): double
PravaVrijednost
+dajVrijednost(): double
VrijednostSaPopustom
+dajVrijednost(): double

COMPOSITE PATTERN

Namjena: Osnovna namjena Composite paterna (kompozitni patern) je da omogući formiranje strukture stabla pomoću klasa, u kojoj se individualni objekti (listovi stabla) i kompozicije individualnih objekata (korijeni stabla) jednako tretiraju.

- *S obzirom na to da u našoj aplikaciji postoji više metoda unutar različitih klasa koje vrše ažuriranje određenih stavki, implementirati ćemo Composite patern kako bismo optimizovali navedene metode, odnosno kako bismo postigli da se navedene metode izvršavaju unutar zajedničkog interfejsa, umjesto da se izvršava svaka metoda zasebno. Navedeni interfejs će sadržavati metodu azuriraj() koja se može primjeniti nad objektima različitih kompleksnosti.*



FLYWEIGHT PATTERN

Namjena: Postoje situacije u kojima je potrebno da se omogući razlikovanje dijela klase koji je uvijek isti za sve određene objekte te klase (tzv. glavno stanje (eng. intristic state)) od dijela klase koji nije uvijek isti za sve određene objekte te klase (tzv. sporedno stanje (eng. extrinsic state)). Osnovna namjena Flyweight paterna je upravo da se omogući da više različitih objekata dijele isto glavno stanje, a imaju različito sporedno stanje,