

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. 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.

- *Unutar aplikacije CryptoInvest, Singleton patern bi se mogao iskoristiti da smo umjesto klase Valuta imale klase za svaku valutu pojednicano, te bi time izbjegli nepotrebn pristup bazi više puta, s obzirom da se u valuti cuvaju podaci koji su zajednicki za svakog korisnika, i time bismo izbjegli probleme i uštedili na vremenu. Kako bi navedeno bilo moguće potreban je privatni atribut tipa određene valute, privatni konstruktor bez parametara, te metoda getInstance().*

PROTOTYPE PATTERN

Uloga Prototype patterna 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 objekta. Prototype dizajn pattern dozvoljava da se kreiraju prilagođeni objekti bez poznavanja njihove klase ili detalja kako je objekat kreiran.

- *Unutar aplikacije CryptoInvest, Prototype pattern bi se mogao iskoristiti u okviru statistike. S obzirom na to da sama aplikacija zahtjeva jako puno statistike, kao naprimjer dnevni profit, sedmični profit, itd. Nema potrebe da se svaki put iznova ažuriraju navedeni podaci, te je samim time moguće iskoristiti Prototype pattern pomoću kojeg bismo klonirali već postojeći objekat te naknadno vršili potrebne izmjene kao i nadogradnje.*

FACTORY METHOD PATTERN

Uloga Factory Method patterna je da omogući kreiranje objekata na način da podklase odluče koju klasu instancirati. Različite pod klase 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.

- *Unutar aplikacije CryptoInvest, Factory Method pattern bi mogao biti iskorišten unutar pristupa kursevima. Uz upotrebu Factory Method patterna, imali bismo interfejs IKurs, te klase koje bismo mogli naslijediti kao naprimjer Beginner, Medium, Hard i sl. Na taj način bismo izbjegli korištenje različitih metoda za kreiranje različitih kurseva.*

ABSTRACT FACTORY PATTERN

Abstract Factory pattern omogućava da se kreiraju familije povezanih objekata/produkata. Na osnovu apstraktne familije produkata kreiraju se konkretne fabrike (factories) produkata različitih tipova i različitih kombinacija. Pattern odvaja definiciju (klase) produkata od klijenta. Zbog toga se familije produkata mogu jednostavno izmjenjivati ili ažurirati bez narušavanja strukture klijenta.

- *Unutar aplikacije CryptoInvest, Abstract Factory pattern bi mogao biti iskorišten u slučaju kada bi imali mnogo više kriptovaluta u ponudi. Kako bismo omogućili da korisnik brzo i jednostavno pronađe valute koje mu odgovaraju postojali bi određeni filteri kao naprimjer filtriranje po cijeni, ili popularnosti određene valute. U tom slučaju kreirali bismo interfejs IFactory kako bi ostvarili jednostavniju pretragu određenih proizvoda.*

BUILDER PATTERN

Uloga Builder patterna je odvajanje specifikacije kompleksnih objekata od njihove stvarne konstrukcije. Isti konstrukcijski proces može kreirati različite reprezentacije.

- *Unutar aplikacije CryptoInvest, Builder pattern bi se mogao iskoristiti u slučaju većeg broja ponuđenih kriptovaluta na način da kada korisnik odabere željenu kriptovalutu prikazuje se stranica sa detaljnim specifikacijama određene valute. Prema tome, postojao bi interfejs IBuilder te klasa ValutaBuilder.*