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.
Ovaj patern bismo mogli iskoristiti kada bi imali neku kontejnersku klasu Macke koja bi sadržavala spisak svih mačaka i nalazila se kao atribut u nekim drugim klasama. Nju bi trebalo instancirati samo jednom jer bi inače bilo moguće da dođe do konflikta dok admin i korisnik uređuju npr profil mačke u isto vrijeme. Trebalo bi dodati privatni static atribut Macke, privatni konstruktor Macke koji ne prima ništa, te public static metodu getInstance().

Prototype patern

Uloga Prototype paterna je da kreira nove objekte klonirajući jednu od postojećih prototip instanci (postojeći objekat). Prototype dizajn patern dozvoljava da se kreiraju prilagođeni objekti bez poznavanja njihove klase ili detalja kako je objekat kreiran.
Ovaj patern bismo mogli iskoristiti kada bi sa statistikom glasova na anketama nešto radili npr. uzimali statistiku glasova i analizirali ih na osnovu perioda, pola, dobi itd. Pri prvoj analizi bismo kopirali podatke iz baze, enkapsulirali ih i kreirali objekat za njihovo spašavanje. Sljedeći put kada bi bilo potrebno vršiti neke nove analize nad istim skupom podataka ne bismo pristupali bazi opet nego koristili taj klonirani objekat.

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.
Ovaj patern bismo mogli iskoristiti kada bismo pored mačaka željeli imati registraciju i drugih životinja. Sistem bi se lahko nadogradio zato što bi samo

napravili klasu za novu životinju dok bi registracija ostala ista. Patern bi omogućio da se instancira klasa(vrsta životinje) koja je potrebna pri toj registraciji. Bio bi nam potreban interfejs Izivotinja te klase Macka i npr. Pas koje bi bile naslijeđene iz nje, a admin bi jednom metodom mogao instancirati bilo koju od tih klasa.

Abstract Factory patern

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 izmjenjivati ili ažurirati bez narušavanja strukture klijenta.
Ovaj patern bi bio iskoristiv kada bismo željeli ubaciti filtere na pretragu destinacija(po udaljenosti,..). Bilo bi potrebno kreirati interfejs IFactory koji bi na jednostavniji način omogućio dobivanje instance Destinacije na osnovu datih filtera.

Builder patern

Uloga Builder paterna je odvajanje specifikacije kompleksnih objekata od njihove stvarne konstrukcije. Isti konstrukcijski proces može kreirati različite reprezentacije. 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.
Ovaj patern bi se mogao iskoristiti kada bi dodali još neke vrste životinja. Pri pristupu bi se tražio odabir životinje te na osnovu te informacije bi se prikazala stranica sa novostima i anketama prilagođena toj vrsti životinje. Dodali bismo interfejs IBuilder i klasu VrstaZivotinjeBuilder.