Analiza primjene kreacijskih obrazaca dizajna za Sistem upravljanja univerzitetom

1. Singleton patern

Uloga

Osigurava da postoji samo jedna instanca određene klase i omogućava globalni pristup toj instanci.

Primjena u sistemu upravljanja univerzitetom

Na osnovu zadanih klasa (`Korisnik`, `Predmet`, `Izvjestaj`, `Notifikacija`) i enumeracija (`UlogaKorisnika`, `Semestar`), **nema očitih kandidata** za primjenu Singleton obrasca. Navedene klase tipično predstavljaju entitete kojih sistem treba imati više instanci (npr. više korisnika, više predmeta). Singleton obrazac bi bio primjenjiv na klase koje upravljaju globalnim, jedinstvenim resursima ili stanjem, a takve klase nisu specificirane u ovom kontekstu.

2. Prototype patern

Uloga

Omogućava brzo kloniranje već postojećih objekata bez potrebe da se kreira novi objekt od nule, što je korisno kada je kreiranje objekta skupo ili složeno.

Primjena u sistemu upravljanja univerzitetom

- **Klasa:** `Predmet` – Mogao bi se koristiti za kreiranje nove instance predmeta za narednu akademsku godinu koja je veoma slična postojećoj (npr. isti naziv, ECTS bodovi, pripadajući semestar iz enumeracije `Semestar`), uz minimalne izmjene. Kloniranje postojećeg objekta `Predmet` i zatim modificiranje potrebnih atributa može biti efikasnije.
- **Klasa:** `Notifikacija` – Mogao bi biti koristan za kreiranje više sličnih notifikacija gdje se mijenja samo primalac ili manji dio sadržaja, koristeći postojeću notifikaciju kao prototip.

Klasa `Izvjestaj` obično ne bi bila kandidat za Prototype, jer se izvještaji generiraju na osnovu trenutnih podataka.

3. Factory Method patern

Uloga

Definira interfejs za kreiranje objekta, ali omogućava izvedenim klasama da odluče koju konkretnu klasu instancirati. Klasična primjena ovog obrasca oslanja se na postojanje hijerarhije klasa gdje tvornička metoda delegira instanciranje odgovarajućoj izvedenoj klasi.

Primjena u sistemu upravljanja univerzitetom

- **Klasa:** `Korisnik` **i enumeracija** `UlogaKorisnika` – S obzirom da je `Korisnik` jedna konkretna klasa koja obuhvata sve tipove korisnika (gdje se specifična uloga korisnika određuje putem enumeracije `UlogaKorisnika` koja se postavlja kao atribut), **klasični Factory Method obrazac nije direktno primjenjiv** za kreiranje različitih *tipova (izvedenih klasa)* korisnika. Ne postoji hijerarhija izvedenih klasa od `Korisnik` (npr. `Student extends Korisnik`) za koje bi tvornička metoda odlučivala koju instancirati. Kreiranje objekata `Korisnik` bi se oslanjalo na njegov konstruktor, koji bi primao enumeraciju `UlogaKorisnika` kao parametar za internu konfiguraciju instance. Eventualna statička pomoćna metoda koja bi enkapsulirala poziv konstruktora i možda neku dodatnu logiku ovisno o ulozi bila bi bliža "Simple Factory" idiomu, ali ne i punom Factory Method obrascu.

- **Klasa:** `Notifikacija` – Slično i ovdje, `Notifikacija` je jedna konkretna klasa, Factory Method ne bi bio primjenjiv u svom klasičnom smislu. Primjena bi imala smisla ako bi postojali različiti tipovi notifikacija implementirani kao **izvedene klase** od `Notifikacija`.

- **Klasa:** `Izvjestaj` – Isto vrijedi i za klasu `Izvjestaj`. Ako je to jedna konkretna klasa, Factory Method nije direktno primjenjiv. Bio bi relevantan ako bi postojali različiti tipovi izvještaja kao **izvedene klase** od `Izvjestaj`.

Zaključno, za navedene konkretne klase, bez uvođenja hijerarhije nasljeđivanja, Factory Method obrazac gubi svoju primarnu svrhu delegiranja instanciranja različitim izvedenim klasama.

4. Abstract Factory patern

Uloga

Omogućava kreiranje familija povezanih ili ovisnih objekata bez navođenja njihovih konkretnih klasa.

Primjena u sistemu upravljanja univerzitetom

S obzirom na strogo definirani skup klasa (`Korisnik`, `Predmet`, `Izvjestaj`, `Notifikacija`) i enumeracija (`UlogaKorisnika`, `Semestar`), gdje su klase poput `Korisnik` konkretne, **nema očitih scenarija** za primjenu Abstract Factory obrasca. Ovaj obrazac se koristi za kreiranje familija *različitih, ali povezanih* tipova objekata. Za primjenu Abstract Factory obrasca, bilo bi potrebno definirati više skupova povezanih produkata (npr. različiti tipovi korisničkih interfejsa ili alata koji ovise o platformi ili kontekstu) i odgovarajuće tvornice za svaki skup, što nije implicirano trenutnim opisom klasa.

5. Builder patern

Uloga

Odvaja konstrukciju kompleksnog objekta od njegove reprezentacije, tako da isti proces konstrukcije može kreirati različite reprezentacije. Koristan je kada objekt ima mnogo atributa.

Primjena u sistemu upravljanja univerzitetom

- **Klasa:** `Korisnik` – Ova klasa vjerovatno ima veći broj atributa (npr. ID, ime, prezime, email, korisničko ime, lozinka, atribut uloge baziran na enumeraciji `UlogaKorisnika`, datum rođenja, kontakt informacije itd.). Korištenje konstruktora s mnogo parametara je nepraktično. Builder obrazac bi omogućio čitljivije i fleksibilnije kreiranje `Korisnik` objekata, gdje se atributi postavljaju postepeno.

- **Klasa:** `Predmet` – Klasa `Predmet` također može imati više atributa (npr. naziv, šifra, ECTS bodovi, opis, atribut semestra baziran na enumeraciji `Semestar`, reference na profesora/korisnika itd.). Builder bi olakšao kreiranje i konfiguraciju ovih objekata.

- **Klasa:** `Notifikacija` – Ako objekti `Notifikacija` imaju više konfigurabilnih dijelova (npr. primalac, pošiljalac, naslov, tijelo poruke, tip, prioritet, vrijeme slanja), Builder bi mogao biti koristan za njihovu konstrukciju.

- **Klasa:** `Izvjestaj` – Ako je proces kreiranja objekta `Izvjestaj` složen i uključuje postavljanje više parametara ili filtera prije samog generiranja sadržaja, Builder bi mogao pojednostaviti ovaj proces.

Builder obrazac je posebno koristan kada objekti imaju opcionalne atribute ili kada je redoslijed inicijalizacije bitan, a želimo izbjeći "teleskopske konstruktore".