Laboratorijska vježba 9 – upotreba kreacijskih paterna

1. Singleton – singleton je jako koristan za klase za koje nema smisla da postoje u velikom broju, međutim većina naših klasa je tipa Termin, Sala, Korisnik, gdje očekujemo veliki broj. Mjesto za potencijalnu upotrebu se može naći u bivšoj kontejnerskoj klasi Kino, koju smo ukinuli kad nam je rečeno da dijagram klasa ne smije sadržavati kontejnerske klase. Ta klasa je inicijalno sadržavala spisak filmova, sala, aktivnih termina i sl. S tim na umu, da smo uzeli drugi put razvoja aplikacije, mogli bismo vratiti klasu Kino na dijagram uz postulat singleton paterna.
2. Protoype je potreban za hitro kloniranje neke klase, uz idealnu verziju upotrebe za nekoliko različitih izvedenih klasa koje se mogu brzo kopirati u nove instance. Za taj idealni princip primjene nemamo potencijala, ALI protoype možemo iskoristiti uz klasu Film, jer su svi Filmovi u principu jako slični jedan drugom, i svaki film u istom žanru se može stvoriti kopijom nekog pretka istog žanra i slično. S tim bi radnik pri dodavanju filma na nivou koda kopirao stari film i mijenjao mu neke sitne detalje.
3. Factory Method – Ovaj šablon se mogao iskoristiti za pružanje termina korisnicima na uniforman način. Kako ovaj patern radi u principu jeste da klijent zahtijeva uslugu, ali dobavljač usluge bira na koji će se način usluga ispuniti. Tom filozofijom mogli smo raširiti termin na dvije podklase: termin za fizička i termin za pravna lica, gdje bi klasa sa FactoryMethod funkcijom birala da li će klijent dobiti termin za fizičko ili termin za pravno lice, zavisno od tipa klijenta.
4. Abstract Factory – na svim primjerima ovaj patern barata velikim brojem klasa slične namjene ili ideje. Takvih klasa u našem programu baš i nema da se dijele, ali postoji način koji bi tražio reorganizaciju samog načina rada aplikacije. Trenutno je fokus aplikacije na Terminima, koji čuvaju skoro sve informacije za jednu projekciju, ali ukoliko bismo stavili fokus na Rezervaciju tako da sadrži sve informacije i o Terminu i o Korisniku, onda bi Abstract Factory mogao biti sistem kojim bi se izgradila Rezervacija, sa jednom verzijom za fizičko lice sa svojim vrstama termina i svojim vrstama bilježenja zauzetih sjedišta i drugom verzijom za pravno lice čije razervacije rade na istom kosturu ali na sasvim drugi način.
5. Builder – za razliku od onog spomenutog neposredno iznad, ovaj se može iskoristiti na Rezervacijama u stanju u kakvom ih trenutno zatičemo: umjesto da rezervacija builda sama sebe, builder bi je mogao sagraditi, uz proslijeđene parametre za id korisnika, id termina i potencijalno sjedišta.