Kreacijski paterni

- SINGLETON PATTERN

Uloga Singleton paterna je osiguravanje instanciranja klasa samo jedanput, te osiguravanje globalnog pristupa kreiranoj instanci klase.
Singleton patern moguće bi bilo iskoristiti za klasu Admin koja se neće mijenjati.
S obzirom na činjenicu da imamo samo jednog admina, a potrebna nam je globalna veza sa svim ostalim klasama u sistemu. Za ovo nam je potreban privatni statički atribut, privatni konstruktor i getInstance().

- PROTOTYPE PATTERN

Uloga Prototype paterna je kreiranje novih objekata klonirajući već postojeći objekat. Ovime postižemo mogućnost kreiranja prilagođenih objekata uz smanjenje kompleksnosti, te mogućnost repliciranja objekta više puta uz naknadne izmjene karakteristika. Ovo je korisno za objekte sa više instanci koje imaju većinu istih atributa.
U našem sistemu Prototype pattern bi recimo mogao naći primjenu kod objekta klase Obrazac gdje bi recimo postavili neke “tipične” odgovore u obrascu koji bi mogli naknadno biti izmijenjeni. Ili recimo za poruke zbog potencijalno velikog broja poruka koje će biti razmijenjene između dva spojena korisnika. Za to bi nam bio dovoljan interfejs sa metodom kloniraj(). Ovime bismo izbjegli više bespotrebnih pristupanja bazi.

- FACTORY METHOD PATTERN

Uloga Factory Method paterna je kreiranje novih objekata na način da podklase odluče koju klasu instancirati. Za ovo se koristi factory metoda koja odlučuje o programskoj logici. Podklase pri tome koriste isti interfejs, te korisnik ne treba znati razliku između rezultata koje različite subklase vraćaju.
Ovo bi se moglo iskoristiti u našem programu za kreiranje Korisnika. Bio bi nam neophodan interfejs ICreate sa metodom NoviKorisnik koja bi vraćala korisnika. Na ovaj način mogli bismo uvesti nove vrste korisnika recimo botove ili premium korisnike.

- ## ABSTRACT FACTORY PATTERN

Uloga Abstract Factory paterna je kreiranje hijerarhija objekata pri čemu se izbjegavaju if-else uslovi nasljeđivanjem. Ako postoji više tipova istih objekata i ako različite klase koriste različite podtipove, te klase onda postaju fabrike za kreiranje objekata zadanog podtipa bez potrebe za specificiranjem pojedinačnih objekata.
Ovaj patern dobro bi bilo iskoristiti za omogućavanje korisnicima teme izgleda aplikacije ili samo chata. Svaka tema bi bila zaseban factory sa metodama koje mijenjaju izgled ovisno o tome koja je tema u pitanju. Mogao bi se dodati veliki broj različitih tema za korisnika da bira, a sve bi jednostavno nasljeđivale apstraktnu klasu Factory.

- ## BUILDER PATTERN

Uloga Builder paterna je razdvajanje specifikacija kompleksnih objekata od njihove stvarne konstrukcije. Kao rezultat dobijamo različite specifikacije objelata koristeći isti proces konstrukcije. Različite dijelove konstrukcije objekata izdvajamo u zasebne metode koji se mogu pozivati različitim redoslijedom ili čak izostavljati, kako bi se dobili različiti podtipovi objekta.
Ovaj patern bi bio iskoristiv za potrebe dodavanja informacija na korisnički račun. Za primjer možemo uzeti izmjenu bračnog statusa, dodavanje slike, visine, težine, lokacije i ostalih nenužnih informacija koje nisu postavljene pri inicijalnom kreiranju naloga. Najkompleksniji primjer bi bio ispunjavanje obrasca. U svrhu nadograđivanja informacija o korisniku postojali bi različiti Builderi za svaku dodatnu informaciju. Svi oni bi pripadali interfejsu IBuilder.