KREACIJSKI PATTERNI

1. Singleton pattern

Uloga ovog paterna je da osigura da se klasa moze instancirati samo jednom, te da osigura globalni pristup kreiranoj instanci klase. Neki od objekata koje je potrebno samo jednom instancirati su: objekti koji se koriste za login,driveri za razne uredjaje, objekti za upravljanje setovanjem registara itd. Ovaj patern se najcesce koristi za uspostavljanje jedinstvene konekcije na bazu podataka, te ubrzavanje pristupa koristenjem kes memorije.

Primjer primjene:

Singleton patern bi smo mogli implementirati za komuniciranje s vanjskim servisima, jedna singleton klasa bi brinula o integraciji tih servisa. Takodjer bi smo mogli implementirati singleton klasu koja predstavlja kalendar dogadjaja, s obzirom da cemo koristiti Calendly za odredjivanje pocetka I kraja perioda iznajmljivanja alata.

2. Prototype pattern

Ovaj patern sluzi za kreiranje novih objekata klonirajuci vec postojeci objekat. Tako se dozvoljava kreiranje prilagodjenih objekata bez poznavanja klase ili detalja o kreiranju objekta. Koristi se za sakrivanje konkretne klase od klijenta, editovanje klasa za vrijeme izvrsavanja, te da broj klasa u sistemu bude u minimumu.

Primjer primjene:

S obzirom da imamo sistem s registrovanim korisnicima, mogli bi smo dodati pocetne postavke za korisnika, te implementirati prototip objekte za korisničke profile. Kada se novi korisnik registrira, možemo klonirati prototip objekta korisničkog profila i prilagoditi ga za novog korisnika.

3. Factory method pattern

Uloga ovog patterna je da omoguci kreiranje objekata na nacin da podklase odluce koju ce klasu instancirati. Razlicite podklase mogu na razlicit nacin implementirat interfejs. Odgovarajuca podklasa se instancira preko posebne metode na osnovu informacija od strane klijenta.

Primjer primjene:

Da smo se odlucili da imamo vise klasa za razlicite tipove alata, mogli bi smo implementirati factory class koji će instancirati objekte alata na temelju korisničkog zahtjeva. Na taj način bi izbjegli da svugdje u kodu moramo eksplicitno instancirati konkretne klase alata.

4. Abstract factory pattern

Ovaj pattern omogucava kreiranje familije povezanih objekata. Na osnovu njih se kreiraju konkretne fabrike produkata razlicitih tipova I kombinacija. Patern odvaja definicju produkata od klijenta. Sistem postaje neovisan od toga kako su produkti kreirani, sastavljeni I implementiran.

Primjena:

Ovaj pattern smo vec iskorisitili kod klase User, jer sve nasljedjene klase iz User klase ce moci koristiti metode te klase.

5. Builder pattern

Ovaj pattern sluzi za odvajanje specifikacije kompleksnih objekata od njihove stvarne konstrukcije. Koristi se kada je neovisan algoritam za kreiranje pojedinacnih dijelova, kada je potrebna kontrola procesa konstrukcije, kada se vise objekata na razlicit nacin sastavlja od istih dijelova.

Primjena:

Builder pattern cemo iskoristiti za filtriranje alata prilikom pretrage. Na klasu Filter cemo dodati atribute za sortiranje, za odabir najvise I najnize cijene, za odabir lokacije tj. Store-a itd. S obzirom da ne moraju svi atributi biti iskoristeni kod filtriranja, na ovaj nacin cemo izbjeci bespotrebnu inicijalizaciju.