KREACIJSKI PATTERNI

1. Prototype pattern

Uloga Prototype paterna je da kreira nove objekte klonirajući jednu od postojećih prototip instanci (postojeći objekat). Ako je trošak kreiranja novog objekta velik i kreiranje objekta je resursno zahtjevno tada se vrši kloniranje već postojećeg objekata. Prototype dizajn patern dozvoljava da se kreiraju prilagođeni objekti bez poznavanja njihove klase ili detalja kako je objekat kreiran.

U našem sistemu ovaj pattern možemo iskoristiti za klasu Stol. Recimo kada se kreira nova Rezervacija, ili se Stol dodaje u određeni lokal i zatreba nam identičan objekat klase Stol možemo samo pozvati metodu clone().

2. Factory Method

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.

U našem sistemu ovaj pattern možemo iskoristiti za podklase PlacanjeUzivo i KarticnoPlacanje. Klijent može da izabere da plati gotovinom i u tom slučaju se šalje obavijest uposleniku koji je zadužen za njegov sto. Ako klijent izabere da plati putem kreditne kartice, otvaraju mu se polja za unos broja kreditne kartice, datuma isteka i CVV-a.

3. Singleton patern

Singleton pattern se koristi kako bi se osiguralo da se klasa može instanicarati samo jednom i da osigura globalni pristup kreiranoj instanci klase.

Bez dodavanja korisničkih zahtjeva ovaj patern u našem sistemu ne bismo mogli iskoristiti jer ne postoji klasa koja bi se samo jednom trebala instacirati. Međutim, ako bismo dodali klasu koja bi simulirala bazu podataka te koja bi se kreirala samo jednom tada bismo mogli kreirati Singleton pattern za naš sistem.

4. Abstract Factory pattern

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 pattern nećemo primijeniti jer naša aplikacija nema veliki broj srodnih objekata/klasa.

5. Builder pattern

Uloga Builder patterna je odvajanje specifikacije kompleksnih objekata od njihove stvarne konstrukcije. Upotreba Builder paterna se često može naći u aplikacijama u kojima se kreiraju kompleksne strukture. 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.

U našem sistemu ovaj patter možemo primijeniti kod mušterije koja pri potrazi za rezervacijom može tražiti slobodne lokale u određenim ulicama ili naseljima. Napravili bismo klasu koja bi generisala slobodne lokale na tim mjestima.