Kreacijski paterni

1. Singleton pattern

Singleton pattern se koristi za ograničavanje instanciranja klase na samo jedan objekt. To znači da klasa ima samo jednu instancu kojoj se omogućuje globalni pristup.

Ovaj pattern možemo iskoristiti za stvaranje i upravljanje rezervacijama vozila. Jedna instanca klase će se koristiti za sve rezervacije, odnosno preko iste instance singleton klase ćemo upravljati svim rezervacijama vozila. Pri kreiranju nove rezervacije vozila informacije o rezervaciji se dodaju u singleton instancu. Za pristup tim informacijama može se koristiti singleton instanca da se dohvati rezervacija ili da bi se izvršila neka druga operacija nad rezervacijom.

2. Prototype pattern

Prototype pattern se koristi za kreiranje novih objekata na temelju već postojećih objekata. Umjesto da se koristi klasično instanciranje objekata (new operator), Prototype pattern koristi kopiranje (cloning) postojećeg objekta kako bi se stvorila nova instanca.Glavna ideja iza Prototype patterna je da se stvori objekt koji će služiti kao prototip (uzorak) i zatim se koristi taj prototip za kreiranje novih objekata kroz kopiranje. Na taj način, izbjegava se direktno instanciranje objekata.

Ovaj pattern možemo koristiti kada korisnik specificira željene karakteristike vozila, poput modela, godišta, mjenjača i drugih opcija, odnosno koristeći ovaj pattern kreirat će se nova instanca vozila koja odgovara željenim specifikacijama.

3. Factory Method pattern

Factory Method pattern (Patern fabričke metode) je dizajn obrazac (design pattern) koji se koristi za kreiranje objekata, ali prepušta konkretne detalje kreiranja podklasama. Factory Method pattern definiše apstraktnu klasu ili interfejs koji sadrži apstraktnu metodu koju konkretne podklase implementiraju kako bi kreirale objekte.Glavna ideja iza Factory Method patterna je da se odvoji proces kreiranja objekata od koda koji koristi te objekte. Umjesto da direktno instancirate objekte, koristite Factory Method koji abstrahuje detalje kreiranja i pruža fleksibilnost da se odaberu različite implementacije objekata.

Ovaj pattern mozemo koristiti za implementaciju discounta za određenu rezervaciju. Kada se primjenjuje discount za određenu rezervaciju, poziva se Factory Method kako bi dohvatili instancu discounta. Tako možemo koristiti tu instancu za izračunavanje konačne cijene rezervacije nekog vozila.

4. Abstract factory pattern

Abstract Factory pattern pruža interface za stvaranje porodice srodno povezanih objekata bez eksplicitnog specificiranja njihovih konkretnih klasa. Ovaj obrazac omogućava kreiranje objekata koji su međusobno kompatibilni i rade zajedno, bez da se detaljno zna o njihovim konkretnim implementacijama.Glavna ideja iza Abstract Factory patterna je da se korisniku pruži intefrace koji definiše niz metoda za kreiranje objekata. Svaki konkretsn factory implementira taj interface i pruža implementacije za kreiranje objekata koji su međusobno kompatibilni. Na taj način, korisnik može koristiti apstraktni factory da kreira objekte bez da zna o konkretnu implementaciju.

I ovaj pattern mozemo iskoristiti za implementaciju discounta za određenu rezerevaciju. U ovom slučaju kada koristimo instancu apstraktnog factory-a da bi dobili objekat discounta. Zatim tu instancu koristimo za izračunavanje konačne cijene rezervacije.

5. Builder pattern

Builder pattern se koristi za konstrukciju složenih objekata korak po korak. Omogućava kreiranje različitih reprezentacija istog objekta koristeći isti konstrukcijski kod.Glavna ideja iza Builder patterna je da se odvoji proces konstrukcije objekta od njegove reprezentacije. Koristi se kada želimo konstruirati objekte koji su složeni i imaju različite varijacije, ali želimo da se konstrukcija tih objekata izvršava na sistematičan način.

Builder pattern se može koristiti za kreiranje i konfiguriranje rezervacije vozila. Glavni dio Builder patterna je Builder klasa koja sadrži metode za postavljanje različitih atributa rezervacije, kao što su model vozila, godište, mjenjač, period rezervacije, cijena itd. Builder klasa omogućuje korisniku da postavlja samo one atribute koji su potrebni za rezervaciju, a ostatak se može postaviti na neku defaultnu vrijednost. Nakon što svi atributi budu proslijeđeni poziva se metoda build() na Builder objektu, koja lreira konačan objekt rezervacije vozila.