

BookMyStyle- kreacijski paterni

U sistemu BookMyStyle, koji omogućava zakazivanje termina u frizerskim salonima, te pružanje usluga tog salona, dali smo uvid gdje se mogu implementirati kreacijski paterni radi optimizacije procesa kreiranja objekata te povećanja fleksibilnosti i skalabilnosti sistema. Korištenjem kreacijskih paterna Factory Methoda i Buildera, postizemo modularnost, smanjenje zavisnosti između klasa, te olakšanu integraciju budućih proširenja.

Factory Method Pattern

Zašto smo izabrali:

Primjena Factory Method paterna omogućava nam jednostavno i kontrolisano kreiranje različitih tipova obavijesti u sistemu bez direktnog instanciranja konkretnih klasa. Ovaj patern smanjuje povezanost između klijenta i konkretnih implementacija obavijesti, olakšavajući dodavanje novih tipova obavijesti u budućnosti. Omogućili smo kreiranje objekata na način da podklase odluče koju klasu instancirati, te na taj način razdvojili zahtijevanje objekata od kreiranja objekata.

Factory podklasa ima punu kontrolu nad kreacijskim procesom i vraća traženi podtip. Samim tim propagira Open/Closed princip. Uvodimo nove obavijesti koje zavise od tipa i mogu se implementirati u nekoj budućoj verziji projekta.

Primjena u sistemu:

Korištena je apstraktna klasa „CreatorObavijest“ kao centralna tačka za kreiranje različitih tipova obavijesti. Implementirane su konkretne klase „CreatorEmailObavijest“ i „CreatorQRObavijest“ koje realiziraju kreiranje specifičnih tipova obavijesti.

Primjer implementacije u sistemu i dijagramu:

- Abstract Class: CreatorObavijest
- Concrete Creators: CreatorEmailObavijest, CreatorQRObavijest
- Product Interface: IObavijest
- Concrete Products: EmailObavijest, QRObavijest

Builder Pattern

Zašto smo izabrali:

Builder pattern odabran je zbog potrebe složenog kreiranja termina koji uključuju različite opcije kao što su usluga, vrijeme, datum i frizer. Ovaj pattern pojednostavljuje kreaciju složenih objekata kroz korake, omogućavajući jednostavniju i jasniju inicijalizaciju objekta bez potrebe za velikim brojem konstruktora. Odvajamo specifikacije kompleksnijih objekata od njihove stvarne konstrukcije. Na ovaj način dobili smo različite skupove objekata koji se na neki način sastavljaju od istih dijelova. Neke od prednosti su da se objekti grade korak po korak ili rekurzivno. Propagira se Single Responsibility Princip. Nedostatak je što se složenost koda povećava zbog kreiranja novih klasa.

Primjena u sistemu:

Builder pattern implementiramo za klasu „TerminBuilder“ koja omogućava da se kreiranje termina izvodi kroz jasno definirane korake, kao što su definiranje usluge, frizera, vremena i datuma.

Primjer implementacije:

- Director: TerminDirector
- Builder: ITerminBuilder
- Concrete Builder: KonkretniTerminBuilder
- Product: Termin

Potencijalna primjena drugih kreacijskih paterna:

- **Singleton Pattern:** Može se koristiti za klasu koja upravlja konekcijom na bazu podataka ili centralnim upravljanjem konfiguracijama, gdje je potreban samo jedan instanca objekta kroz cijeli sistem.
- **Prototype Pattern:** Prikladan je za klasu „Termin“, gdje postoji potreba za kreiranjem velikog broja sličnih objekata termina, a gdje bi se smanjila količina redundantnog koda kloniranjem postojećih objekata.

- **Abstract Factory Pattern:** Može se implementirati za upravljanje različitim grupama povezanih objekata, kao što su različiti setovi UI komponenti za web aplikaciju ili mobilnu aplikaciju.

Zaključak:

Uvođenjem Factory i Builder kreacijskih paterna značajno poboljšavamo strukturu, modularnost i skalabilnost sistema BookMyStyle. Potencijalnim korištenjem Singleton, Prototype i Abstract Factory paterna dodatno se povećava fleksibilnost za buduće proširenje funkcionalnosti te se olakšava održavanje koda.