Univerzitet u Sarajevu
Elektrotehnički fakultet u Sarajevu
Odsjek za računarstvo i informatiku

Kreacijski paterni
eDostava

Objektno orijentisana analiza i dizajn

Naziv grupe: eDostava
Članovi: Ina-Bedra Mujaković
Ema Kalmar
Rijad Zejnilović

1. Singleton patern

Uloga Singleton patern-a je da osigura da se klasa može instancirati samo jednom i da osigura globalni pristup kreiranoj instanci klase. Postoji više objekata koje je potrebno samo jednom instancirati i nad kojim je potrebna jedinstvena kontrola pristupa.

U našem sistemu možemo iskoristiti ovaj patern kod registracije korisnika (ali i izrade profila). To bi se vršilo tako što bi se obavila konekcija na bazu sistema i preko jedne klase bi se vršili svi upiti i tako dobavljali podaci iz baze. Klasu bi nazvali Login i ona će nam osigurati singleton konekciju na bazu i globalni pristup na nju. To znači da će svaka klasa moći pristupiti instanci te klase i dobiti sve neophodne informacije.

2. Prototype patern

Uloga Prototype patterna 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 patern bi mogli primijeniti nad klasom Prozivod. Ukoliko recimo želimo kao administrator dodati neki novi prozivod u restoran ili prodavnicu, mnogo je lakše klonirati jedan proizvod i mijenjati samo neke informacije. (naprimjer: u nekom restoranu se pojavila nova pizza koju želimo dodati u sistem, lakše je da se ona klonira i zatim promijene recimo sastojci, lokacija i cijena).

3. Factory Method patern

Uloga Factory Method patern-a je da omogući kreiranje objekata na način da podklase odluče koju klasu instancirati. Factory Method instancira odgovarajuću podklasu(izvedenu klasu) preko posebne metode na osnovu informacije od strane klijenta ili na osnovu tekućeg stanja.

U našem sistemu možemo iskoristiti ovaj patern kod plaćanja narudžbe s obzirom da imamo više načina plaćanja, npr. PlaćanjeKarticom, PlaćanjePriPreuzimanju, PlaćanjePayPal. Također, trebali bi imati interfejs Iplaćanje. Pored toga, trebamo napraviti i klasu Creator koja bi imala metodu FactoryMethod koja će odlučiti koju klasu instancirati.

4. Abstract Factory patern

Abstract Factory patern nam omogućava da se kreiraju familije povezanih objekata. Na osnovu apstraktne familije produkata kreiraju se konkretne fabrike produkata različitih tipova i različitih kombinacija.

Kod nas ovaj patern možemo primijeniti kod filtriranja jela u restoranima tokom pretrage. Na osnovu nekog filtera, kreirala bi se fabrika produkata različitih jela i tako bi se njihov prikaz učinio mnogo efikasnijim jer ovaj patern upravlja familijama objekata i čuva sve informacije o njima.

5. Builder patern

Uloga Builder patern-a je odvajanje specifikacije kompleksnih objekata od njihove stvarne konstrukcije. Isti konstrukcijski proces može kreirati različite reprezentacije.

U našem sistemu možemo iskoristiti ovaj patern na dva mjesta, u klasi Korisnik i u klasi Narudžba. Obe klase imaju konstruktore s velikim brojem parametara i ako iskoristimo ovaj patern, kreiranje novih objekata bi se obavljalo u klasi KorisnikBuilder ili NarudžbaBuilder, pa bi se i validacija podataka koja bi se inače odvijala u glavnim klasama odvijala u ovim novim. Na taj način bi olakšali i dodavanje novih atributa u početne klase.