Strukturalni patterni

*paterni koji će biti implementirani sigurno su boldirani

1. Adapter pattern

Ovaj pattern možemo iskoristiti kada pretražujemo proizvode – pretraga se radi po nazivu, cijeni i vrsti cvijeća. Korisnik bira kako želi da mu rezultati budu sortirani. Koristimo ovaj pattern jer nam treba drugačiji interfejs za postojeću klasu *Proizvod*, ali ne želimo da mijenjamo samu klasu. Zato pravimo *Adapter* klasu koja služi kao posrednik i omogućava da dodamo funkcionalnost pretrage bez diranja u postojeći kod. Tačnije, riječ je o *Class Adapter* patternu – pravimo interfejs *iSearch* s metodom *sortProducts()*, koja vraća sortiranu listu proizvoda. Zatim se kreira klasa *ProductAdapter* koja implementira taj interfejs.

2. Facade pattern

Facade pattern koristimo za prikaz izvještaja koji je namijenjen zaposlenicima cvjećare. Sistem automatski generiše izvještaj pomoću određenog algoritma, a zaposlenik može da izabere koji tip izvještaja želi, pogleda ga i preuzme u obliku Excel tabele. Ovi izvještaji se odnose na poslovanje i nisu namijenjeni krajnjim korisnicima. Algoritam koji pravi izvještaj koristi metode iz više različitih klasa jer je prilično sveobuhvatan, ali zaposlenike zanima samo gotov izvještaj u tabeli. Tu dolazi *Facade pattern*, koji sve to pojednostavljuje i daje samo ono što im treba.

3. Decorator pattern

Zaposlenik ima mogućnost da mijenja detalje proizvoda – kao što su naziv, cijena, kategorija, podkategorija, opis, slika i boja. Te promjene možemo podijeliti u tri grupe: *EditAll*, koja omogućava uređivanje svih informacija o proizvodu. *EditNameAndPrice*, koja daje pristup samo nazivu i cijeni – jer su to najbitniji podaci koji se najčešće koriste u različitim funkcijama i algoritmima, i metoda *GetAllProducts*, koja prikazuje sve proizvode. Za ovu svrhu koristimo Decorator pattern. Uvodimo dekorator klasu koja preko agregacije sarađuje s *IProizvod* interfejsom i može sadržavati jedan ili više *IProizvod* objekata. Na ovaj način možemo fleksibilno dodavati nove mogućnosti bez mijenjanja osnovnog koda.

4. Bridge pattern

Neprijavljeni korisnici mogu pregledati sve proizvode, njihove detalje, kao i best sellere iz naše ponude – isto kao i registrovani korisnici. Ova funkcionalnost je dostupna svima i radi bez problema. Međutim, za dodavanje proizvoda u korpu i kupovinu, korisnik mora biti registrovan i prijavljen. Ako nije, prikazuje mu se opcija da se prijavi ili registruje kako bi mogao nastaviti s kupovinom.

5. Composite pattern

Ovaj obrazac može se koristiti za izračunavanje ukupne cijene koju korisnik plaća prilikom narudžbe. Izračun se vrši svaki put prilikom plaćanja. Ako kupac ima pravo na popust, cijena se automatski koriguje. Klase će imati interfejs koji omogućava ovu funkcionalnost.

6. Proxy pattern

Prilikom plaćanja narudžbe, kupac ima mogućnost unosa koda za popust. Kod za popust koji je dobio putem email-a je potrebno verifikovati, odnosno provjeriti njegovu ispravnost u smislu tačnosti koda I provjere da li kod važi u datom trenutku zbog ograničenog trajanja koda. Dakle, za verifikaciju koda I odobravanje popusta na cijenu narudžbe koristimo Proxy pattern.