Univerzitet u Sarajevu
Elektrotehnički fakultet
Sarajevo

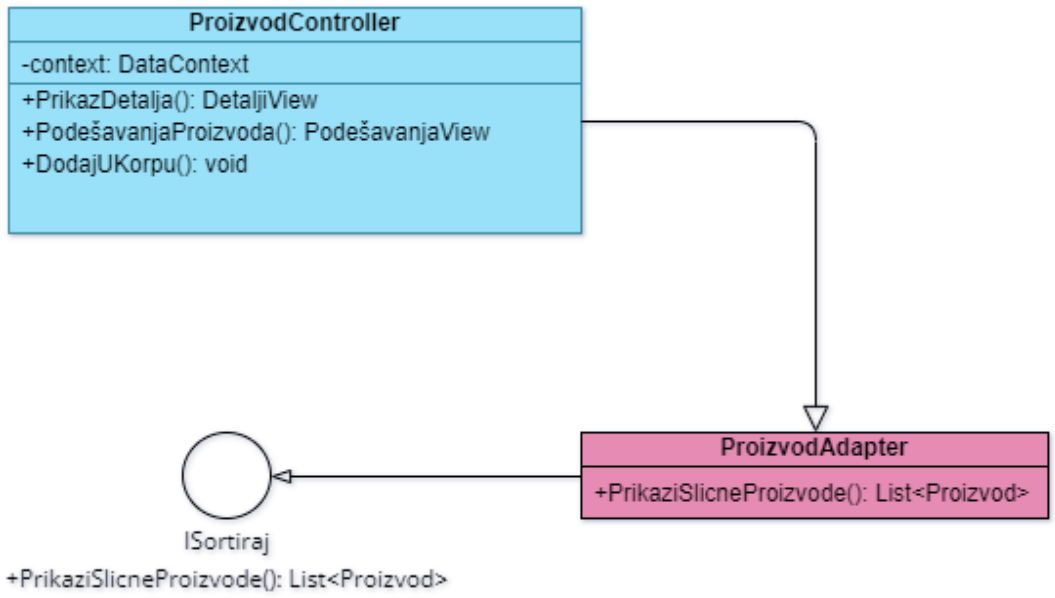
STRUKTURALNI PATERNI

-AMA Cosmetics-

1. ADAPTER PATERN

Adapter patern služi da omogući da se interfejs već postojeće klase koristi kao drugi interfejs, ali često omogućava i to da postojeće klase rade sa drugim bez izmjene njihovog izvornog koda. Osnovna namjena je da omogući širu upotrebu već postojećih klasa. On kreira novu adapter klasu koja povezuje originalnu klasu i željeni interfejs. Na taj način se dobija željena funkcionalost bez ikakvih izmjena na originalnoj klasi.

Ovaj patern u našoj aplikaciji možemo realizovati tako što ćemo prvo implementirati interfejs sa metodom PrikaziSlicneProizvode() koja će omogućiti prikaz sličnih proizvoda nakon otvaranja detalja određenog proizvoda. Nakon toga definišemo klasu *ProizvodAdapter* koja implementira interfejs ISortiraj sa navedenom metodom.



2. FACADE PATERN

Facade patern služi kao prednji interfejs koji maskira složeniji kod tj. skriva složenost većeg sistema i klijentu pruža jednostavniji intefejs. Ovaj patern se koristi kada se želi prekriti neka komplikovana implementacija podsistema i pružiti jednostavno pozivanje inače komplikovanih operacija.

Ovaj patern u našoj aplikaciji se može iskoristiti za jednostavno pozivanje funkcija, filtriranja proizvoda ili sortiranja po određenom kriteriju, a da njihova implementacija u pozadini bude komplikovanija. Definisali bismo jednu klasu u koju bismo smjestili metode npr. filtriraj po kategoriji, brendu, sortiraj po cijeni (od veće ka manjoj ili obrnuto)…

3. DECORATOR PATERN

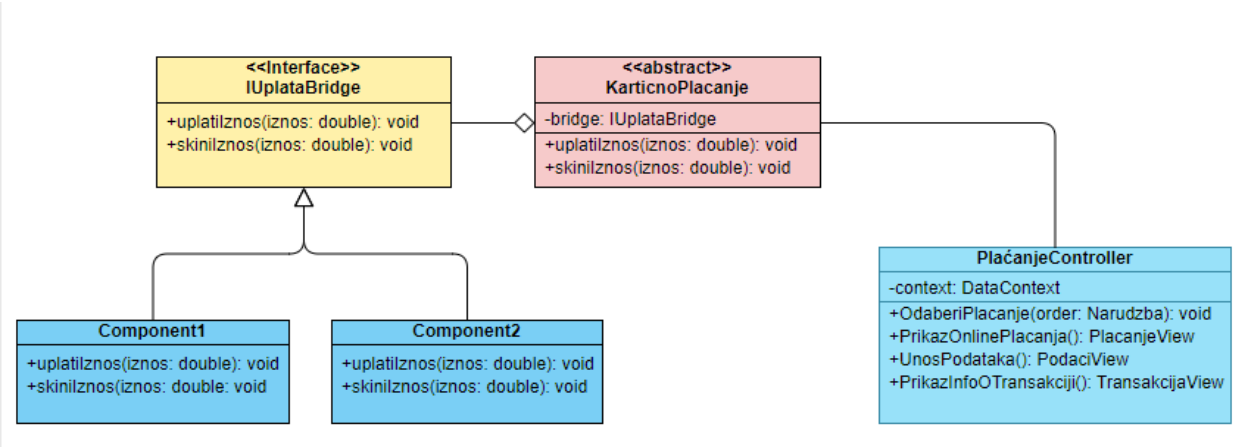
Decorator patern služi da omogući dinamičko dodavanje novih elemenata i ponašanja (funkcionalnosti) postojećim objektima. Također omogućava i da se funkcionalnost podijeli između klasa u zavisnosti od područja interesa. Ovaj patern nasljeđuje originalnu klasu, ali se ne oslanja na nasljeđivanje prilikom dodavanja novih atributa i objekata, već kreira nove.

Ovaj patern u našoj aplikaciji se može iskoristiti ukoliko želimo da korisniku damo mogućnost promjene njegovih podataka, kao što je npr. promjena email-a ili drugih podataka.

4. BRIDGE PATERN

Bridge patern služi da omogući odvajanje apstrakcije i implementacije neke klase tako da ta klasa može posjedovati više različitih apstrakcija i više različitih implementacija za pojedine apstrakcije. Ima mogućnost korištenja naslijeđivanja kako bi razdvojio odgovornosti u različite klase. Ovaj patern je koristan kada imamo razliku između klase i onoga što ta klasa radi i u tom slučaju klasu možemo posmatrati kao apstrakciju, a ono što ona radi kao implemetaciju.

Ovaj patern u našoj aplikaciji se može iskoristiti kod plaćanja računa. U našoj aplikaciji imamo 2 mogućnosti plaćanja računa: kartično i pouzećem. Ako se osvrnemo na kartični način plaćanja računa, ono ima doticaja s vanjskim uređajem a to je banka. Sve banke nemaju isti način uplate i skidanja novca sa računa, pa ćemo napraviti interfejs IUplataBridge koji će definisati apstrakciju i ima različite implementacije za svaku banku.



5. PROXY PATERN

Proxy patern služi da omogući kontrolu pristupa stvarnim objektima. Pored toga on se koristi i kao omotač kojeg klijent uvijek može pozvati da bi pistupio stvarnom objektu koji se nalazi „iza scene“, kao i kada klase imaju osjeltjive podatke ili spore operacije.

Ovaj patern u našoj aplikaciji možemo iskoristi npr. kod prijave korisnika u sistem. Prilikom unosa podataka isti ti podaci se provjeravaju, da li su ispravno uneseni i određuje se vrsta aktera našeg sistema. U zavisnosti od vrste aktera postoje različite funkcionalnosti koje su im namijenjene. Isto tako, klijent je jedini akter koji se ne može prijaviti u sistem bez prethodne registracije. Kao još jedan primjer možemo navesti onemogućavanje liste želja korisnicima koji nisu premium.

6. COMPOSITE PATERN

Composite patern služi da opisuje neku grupu objekata koji su tretirani na isti način i dio su klase, te da omogući kreiranje strukture stabla pomoću klasa. Ovaj patern još omogućava klijentima da pojedinačne objekte i njihove kompozicije tretira ujednačeno odnosno na isti način.

Ovaj patern u našoj aplikaciji možemo iskoristiti kod izračunavanja konačne cijene narudžbe, jer cijena narudžbe registrovanog i premium korisnika se može razlikovati ukoliko premium korisnik iskoristi neki popust koji je ostvario.

7. FLYWEIGHT PATERN

Flyweight patern služi kako bi različiti objekti imali isto glavno stanje, a svaki objekat ima drugačije sporedno stanje, te na taj način postižemo manje zauzimanje memorije. Ovaj patern još omogućava i lakšu implementaciju, testiranje, promjenu i ponovnu upotrebu.

Ovaj patern u našoj aplikaciji možemo iskoristiti na način da se omogući korisnicima da imaju svoju profilnu sliku. Ukoliko neko ne želi postaviti svoju sliku onda je potrebno koristiti neku defaultnu sliku. S obzirom da onda više korisnika može imati istu defaultnu sliku, potrebno je implementirati ovaj patern kako bi korisnici koristili jedan zajednički resurs.