STRUKTURALNI PATTERNI ZA KUPIKARTU

1) **ADAPTER PATTERN**

Adapter patern omogućava rad dviju nespojivih interfejsa. U našem slučaju, možemo ga koristiti za integraciju različitih API-ja za plaćanje ili različitih baza podataka.

Npr. PlaćanjeAdapter: omogućava integraciju različitih metoda plaćanja (PayPal, Stripe, bankovni transferi) koristeći jedinstveni interfejs za plaćanje

2) **FACADE PATTERN**

Facade patern pruža jednostavniji interfejs za kompleksan sistem. Može se koristiti za pojednostavljenje interakcija s podsistemima kao što su upravljanje korisnicima, upravljanje manifestacijama, i sl.

Npr. RezervacijskiSistem: Fasada koja objedinjuje funkcionalnosti upravljanja korisnicima, manifestacijama i kartama.

3) **DECORATOR PATTERN**

Decorator patern se koristi za dodavanje dodatnih funkcionalnosti objektima dinamički, bez promjene njihovog koda. Ovaj patern možete koristiti za proširenje funkcionalnosti karata.

Npr. Karta: Može se dekorirati dodatnim funkcionalnostima poput popusta, dodatnih pogodnosti, ili VIP statusa.

4) **BRIDGE PATTERN**

Bridge patern razdvaja apstrakciju od njene implementacije, omogućavajući nezavisnu promjenu obje strane. Ovo je korisno kada trebamo proširiti obje strane hijerarhije (apstrakciju i implementaciju) bez promjene postojeće logike.

Npr. Karta: Može imati različite vrste (VIP, standardna) koje se mogu implementirati pomoću različitih platformi za isporuku (fizička karta, digitalna karta).

5) **PROXY PATTERN**

Proxy patern koristi se za kontrolu pristupa objektu. Može se koristiti za implementaciju sigurnosnih provjera ili keširanja.

Npr. KartaProxy: Kontroliše pristup stvarnoj klasi Karta, omogućavajući provjeru valjanosti ili autorizacije prije izvršenja operacija.

6) COMPOSITE PATTERN

Composite patern omogućava tretiranje pojedinačnih objekata i njihovih kolekcija uniformno. Koristan je za prikaz složenih hijerarhija podataka.

Npr. Sjediste: Može se organizirati kao kompozitna struktura koja omogućava tretiranje pojedinačnih sjedišta i sekcija sa sjedištima na isti način.

7) FLYWEIGHT PATTERN

Flyweight patern smanjuje korištenje memorije dijeljenjem što je moguće više stanja između objekata. Korisno je kada imamo veliki broj sličnih objekata.

Npr. Mjesto: Možemo koristiti Flyweight patern za upravljanje sjedištima u dvorani, gdje svako sjedište može dijeliti zajedničke podatke poput reda i kolone.