**Adapter patern**

Namjena adapter paterna jeste da omoguci širu upotrebu već postojećih klasa. Kada nam je potreban drugačiji interface postojeće klase, ali ne želimo mijenjati postojeću klasu, trebamo koristiti ovaj patern.

Kreirana klasa služi kao posrednik između orginalne klase i željenog interface-a.

Ovaj patern u našoj aplikaciji možemo koristiti u situacijama se želi kupiti karta. Metoda kupiKartu obračunava cijenu u skladu sa kartom, ali možemo uvesti i metodu koja obračunava popust ili podiže cijenu u skladu sa prtljagom koji putnik posjeduje.

**Strategy pattern**

Ovaj pattern je pattern ponašanja i izdvaja algoritam iz matične klase i uključuje ga u posebne klase. Koristi se kada postoje različiti algoritmi za neki problem. Omogućava klijentu izbor jednog od algoritma iz familije algoritama za korištenje. Ovaj pattern možemo koristiti kada putnik vrsi plaćanje karte, bilo da je plaća karticom ili kešom.

**Prototype patern**

Uloga ovog paterna je da kreira nove objekte klonirajući postojeći objekat. Mogu se kreirati prilagođeni objekti bez poznavanja klase ili informacija o tome kako je objekat kreiran.Ovaj patern se koristi kada je potrebno sakriti konkretne klase od klijenta, dadati ili izbrisati nove klase za vrijeme izvršavanja, da se broj klasa u sistemu održi na minimumu, kada je potrebna promjena strukture podataka za vrijeme izvršavanja. Ovaj pattern možemo koristiti ukoliko npr. Neki od registorvanih korisnika postaje zaposlenik firme, svi podaci ostaju isti, samo se doda podatak o radnom mjestu. Potrebno je dodati metodu Clone koja vrši kloniranje pomoću metode MemberwiseClone(). Takođe se ovaj patern može koristiti za vršenje analize nad nekim skupom podataka iz baze, isti objekat može klonirati vise puta i koristit za vise analiza(npr. Podaci o narudzbama)