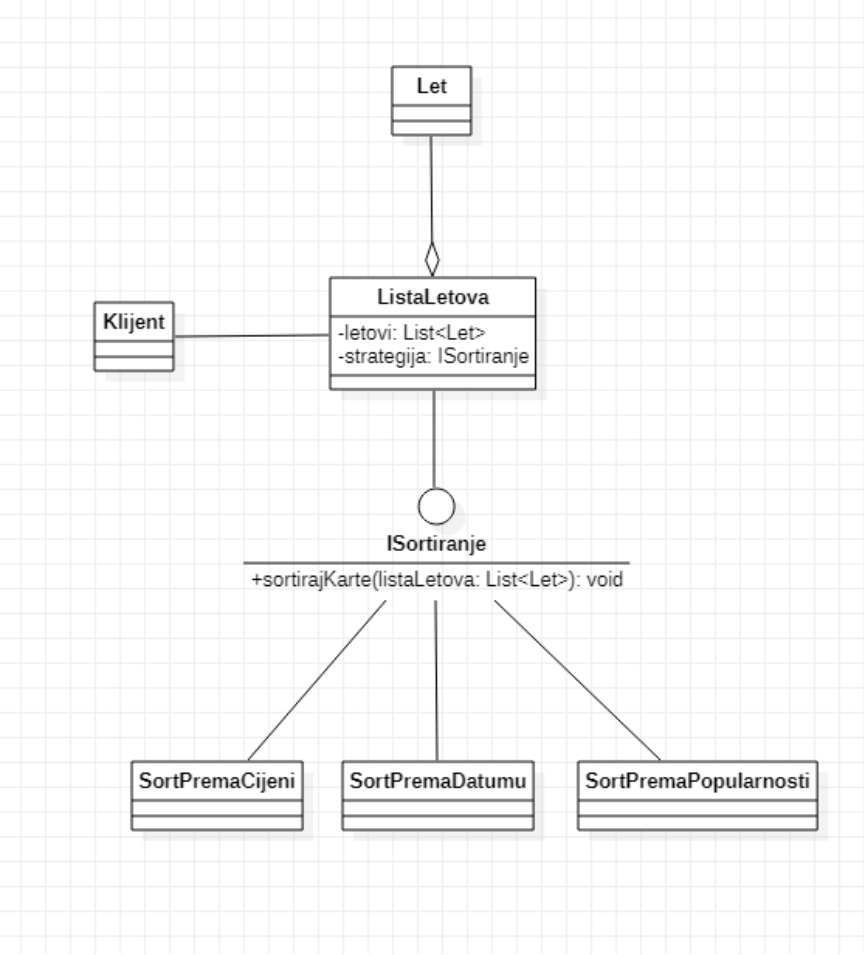
PATERNI PONASANJA

1) **Strategy patern**

Uloga strategy patern jeste da izdvaja algoritam iz matične klase i uključuje ga u posebne klase.
Pogodan je kada postoje različiti primjenjivi algoritmi (strategije) za neki problem.
Strategy patern omogućava klijentu izbor jednog od algoritma iz familije algoritama za korištenje.
Algoritmi su neovisni od klijenata koji ih koriste.
Podržava *open-closed* princip.

U nasem slucaju, strategy patern bi se mogao iskoristiti prilikom odabira nacina sortiranja letova (bilo to po popularnosti, cijeni karte, vremenu polijetanja itd.), gdje bi nam u Strategy podklasama bile metode razlicitih sortiranja, koje bi bile dio IStrategy interfacea.



2) **State patern**

State Pattern je **dinamička verzija Strategy paterna**.
Objekat **mijenja način ponašanja na osnovu trenutnog stanja**.
Postiže se **promjenom podklase unutar hijerarhije klasa**.

U nasem slucaju, mozemo iskoristiti state patern kada unosimo jedinstveni kod leta u pretragu, te ukoliko je status leta da je u zraku, metoda pretrage otvara mapu te omogucuje korisniku da prati taj let, dok sa druge strane ukoliko je let u bilo kojem drugom stanju, metoda bi samo ispisala to stanje. Preduslov za ovaj patern bi bio da napravimo podklase aviona za sva njegova stanja, te da ih povezemo sa odredjenim interfejsom.

3) **TemplateMethod patern**

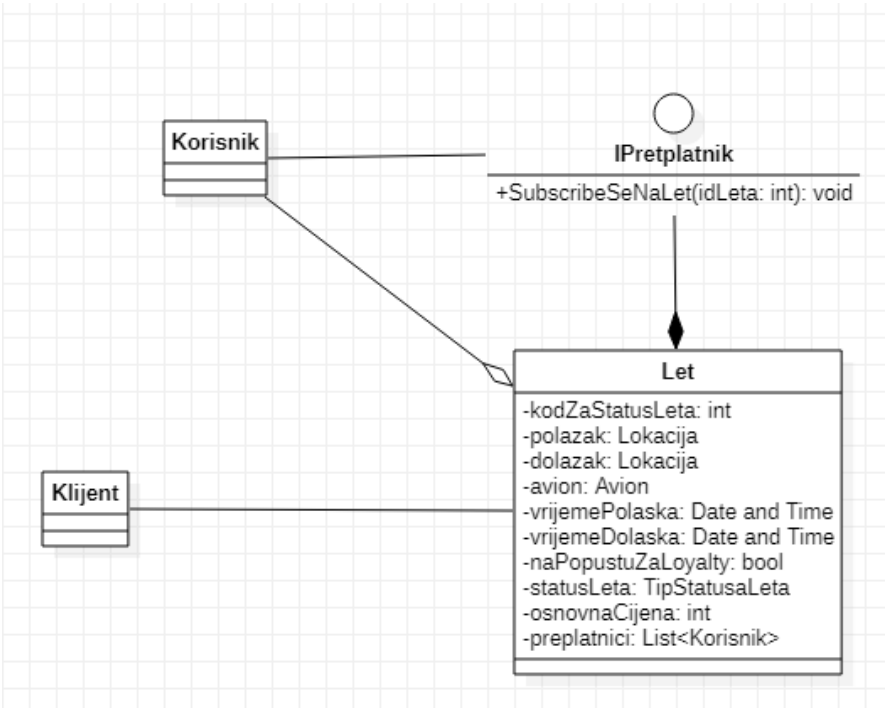
Omogućava **izdvajanje određenih koraka algoritma u odvojene podklase**.
Struktura algoritma se ne mijenja - mali dijelovi operacija se izdvajaju i ti se dijelovi mogu implementirati različito.

U nasem slucaju, templateMethod patern bi se mogao iskoristiti prilikom kupovine karte, gdje u zavisnosti da li je korisnik prijavljen ili ne, prilikom kupovine karte uracanuvamo popust.
Tok transakcije, prikaz karata i druge mogucnosti ce ostati jednake, jedina razlika ce se ogledati u mogucnosti unosenja koda za popust, zavisno od podklase korisnika.

4) **Observer patern**

Uloga **Observer paterna** je da **uspostavi relaciju između objekata tako kada jedan objekat promijeni stanje drugi zainteresirani objekti se obavještavaju**.

U nasem slucaju, observer patern bi mogli iskoristiti u slucaju kada bi omogucili korisnicima da se „subsc ribeaju“ na odredjeni let, te da im dolazi notifikacija kada god taj let promijeni svoje stanje.



5) **Iterator patern**

Iterator patern omogućava sekvencijalni pristup elementima kolekcije **bez poznavanja kako je kolekcija strukturirana**.

U nasem slucaju, iterator patern bi mogli iskoristiti prilikom korisnikove „posjete“ web shopu. Korisnik bi mogao da bira nacin listanja artikala, bilo kroz *„shuffle“* mode (slucajni redoslijed artikala), *„cijena“* (ide kroz artikle prema nekom algoritmu koji uzima u obzir cijenu) itd.