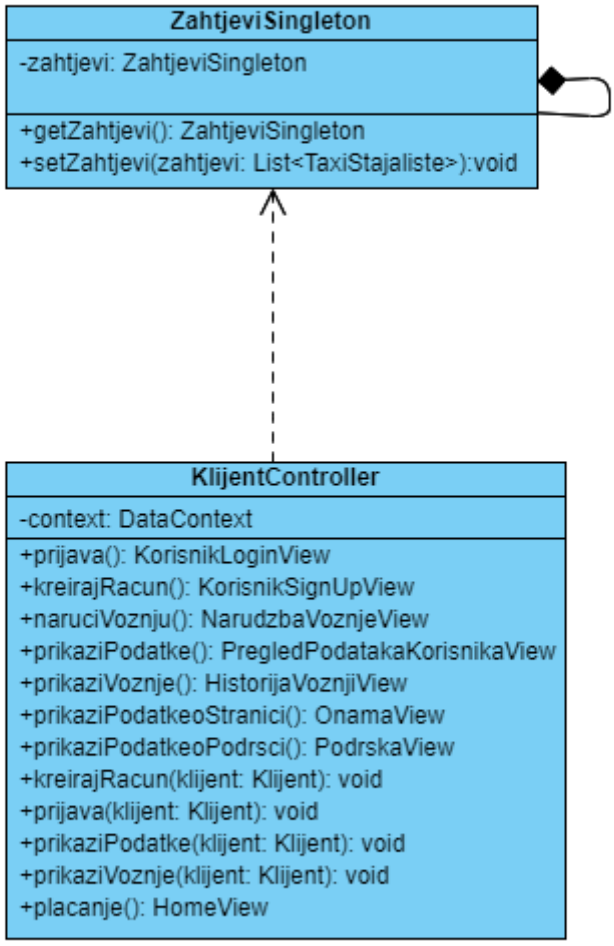
KREACIJSKI DIZAJN PATERNI

1. SINGLETON PATERN

Uloga Singleton paterna je da osigura da se klasa može instancirati samo jednom i da osigura globalni pristup kreiranoj instanci klase.

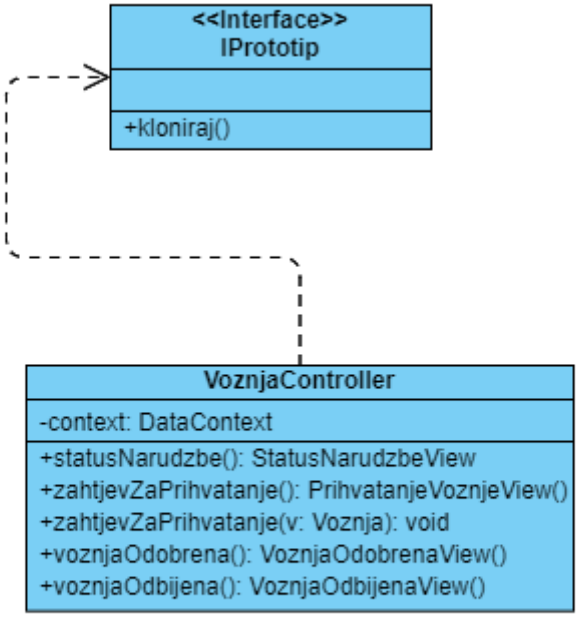
Ovaj patern možemo iskoristiti pri instanciranju klase TaxiStajaliste. Ovu klasu ćemo instancirati samo jednom i bit osiguran globalni pristup kreiranoj instanci klase. Za implementaciju ovog paterna uvest ćemo klasu ZahtjeviSingleton. Naši klijenti mogu da vide sva taxi stajališta i oni će koristiti ovu klasu ZahtjeviSingleton da šalju zahtjev za prikaz svih stajališta.



2. PROTOTYPE PATERN

Prototype dizajn patern dozvoljava da se kreiraju prilagođeni objekti bez poznavanja njihove klase ili detalja kako je objekat kreiran. Prototype obezbjeđuje interfejs za konstrukciju objekata kloniranjem jedne od postojećih prototip instanci i modifikacijom te kopije.

Ovaj patern ćemo upotrijebiti prilikom kreiranja vožnji. Kada kreiramo vožnju možemo klonirati već postojeće vožnje i samo promijeniti odgovarajuće podatke.



3. FACTORY METHOD PATERN

Uloga Factory Method paterna je da omogući kreiranje objekata na način da podklase odluče koju klasu instancirati. Factory Method instancira odgovarajuću podklasu (izvedenu klasu) preko posebne metode na osnovu informacije od strane klijenta ili na osnovu tekućeg stanja.

Ovaj pattern bi mogli iskoristiti pri kreiranju instanci izvedenih klasa iz klase Osoba. Na osnovu informacije od strane korisnika vršit ćemo instanciranje ili klase Vozač ili klase Klijent. Kada bi korisnik aplikacije započeo korištenje iste, dobio bi mogućnost da bira da li će se prijaviti kao vozač ili kao klijent i na osnovu toga bi vršili instanciranje.

4. ABSTRACT FACTORY METHOD PATERN

Abstract Factory patern omogućava da se kreiraju familije povezanih objekata/produkata bez specificiranja konkretnih klasa. Na osnovu apstraktne familije produkata kreiraju se konkretne fabrike (factories) produkata različitih tipova i različitih kombinacija.

Recimo da imamo dvije vrste klijenata StandardKlijent i VipKlijenti i recimo da nakon odredjenog broja vožnji, recimo 50, oba klijenta dobiju odgovarajući univerzalni kod koji bi označio da su prešli taj broj vožnji i koji bi se razlikovao u zavisnosti od vrste klijenta. Za VipKlijente taj bi kod omogućavao da dobiju odredjeni popust nakon što ga dobiju. Taj kod bi bila neka apstraktna klasa a recimo naslijeđeni objekti bi bili vipKod i standardKod.

5. BUILDER PATERN

Uloga Builder paterna je odvajanje specifikacije kompleksnih objekata od njihove stvarne konstrukcije.

Ovaj patern bi mogli iskoristiti kod postavljanja korisničkih podataka. Korisnik može mijenjati svoje podatke neovisno o odobrenju administratora. Recimo da kod klase Klijent umjesto konstruktora sa puno parametara koristimo KlijentBuilder koji bi sadrđžavao iste atribute kao i klasa Klijent. Klasa "KlijentBuilder" bi imala metode za postavljanje svakog pojedinog atributa klijenta. Na primjer, "postaviIme(String ime)" postavlja ime klijenta.

Zatim, korisnik može koristiti KorisnikBuilder objekt da bi postupno postavio svoje podatke. KlijentBuilder objekt omogućava klijentu da postavi samo one atribute koji su mu potrebni ili koje želi promijeniti. Na primjer, klijent može samo ažurirati svoju mail adresu.

Na ovaj način, korisnik može mijenjati svoje podatke neovisno o odobrenju administratora, jer ima kontrolu nad postavljanjem i ažuriranjem svojih podataka putem Builder objekta.