Kreacijski 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.

Postoji više objekata koje je potrebno samo jednom instancirati i nad kojim je potrebna jedinstvena kontrola pristupa.

To se postiže tako što se konstruktor proglasi privatnim (jer se samim činom pozivanja konstruktora stvara novi objekat i mijenjanje toga je van naše kontrole), napravi se privatni staticki atribut tipa same te klase (u kontekstu C#-a to je referenca) i napravi se javna statička kreaciona metoda. Ta metoda radi 2 stvari: ako je gore spomenuti atribut/referenca null kreira se novi objekat tipa same te klase i referenca se postavi da "pokazuje" na njega; u zadnjem koraku vraća objekat na koji atribut "pokazuje".

Singleton pattern bi se u našem projektu mogao iskoristiti ako bi se odlučili implementovati neku vrstu globalnog tajmera koji bi se ponašao kao logger. Biljezio bi kada svaka vožnja započne, kada se završi, koliko je dugo trebalo da se dođe na određenu lokaciju, koliko je korisnik bio aktivan na aplikaciji u toku dana…Pošto se vrijeme računa od registracije korisnika, nema smisla da ovih tajmera bude više, već samo jedan.

2. Prototype patern

Uloga Prototype patterna je da kreira nove objekte klonirajući jednu od postojećih prototip instanci (postojeći objekat). Ako je trošak kreiranja novog objekta velik i kreiranje objekta je resursno zahtjevno tada se vrši kloniranje već postojećeg objekata.

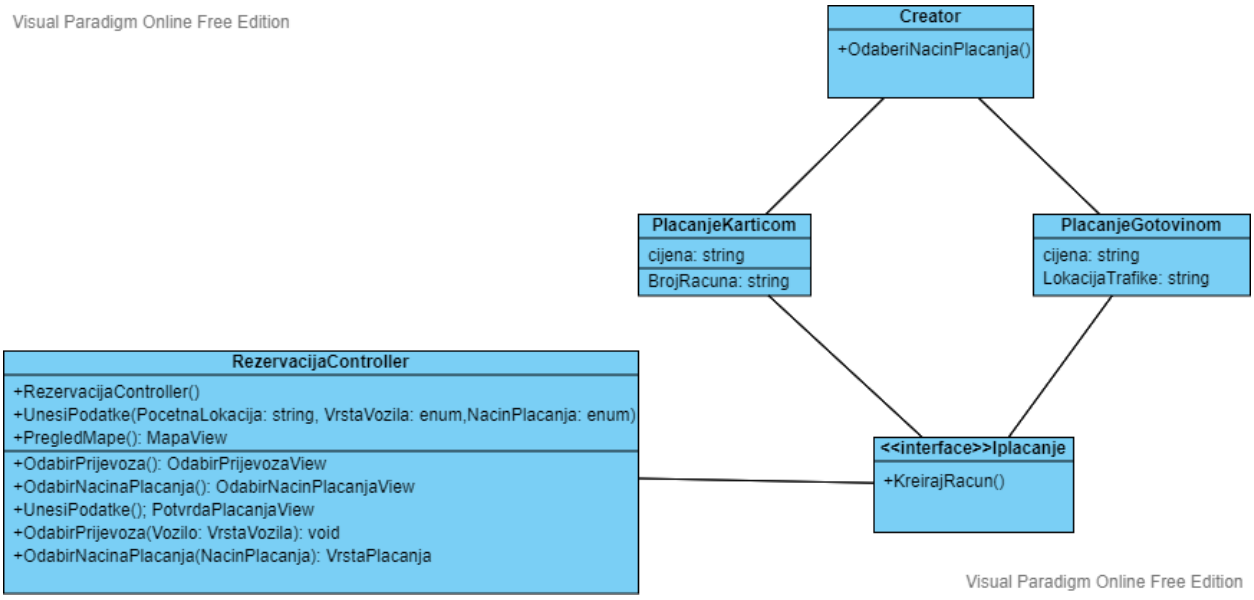
Prototype dizajn patern dozvoljava da se kreiraju prilagođeni objekti bez poznavanja njihove klase ili detalja kako je objekat kreiran.

U našem sistemu, ovaj patern bi mogli primijeniti nad klasom Vozilo. Ukoliko recimo želimo kao administrator dodati neko novo vozilo, mnogo je lakše klonirati jedno vozilo i mijenjati samo neke informacije. (naprimjer: pojavilo se novo vozilo koje želimo dodati u sistem, lakše je da se ono klonira i zatim promijene recimo brzina, cijena, naziv).

3. *Factory method*

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

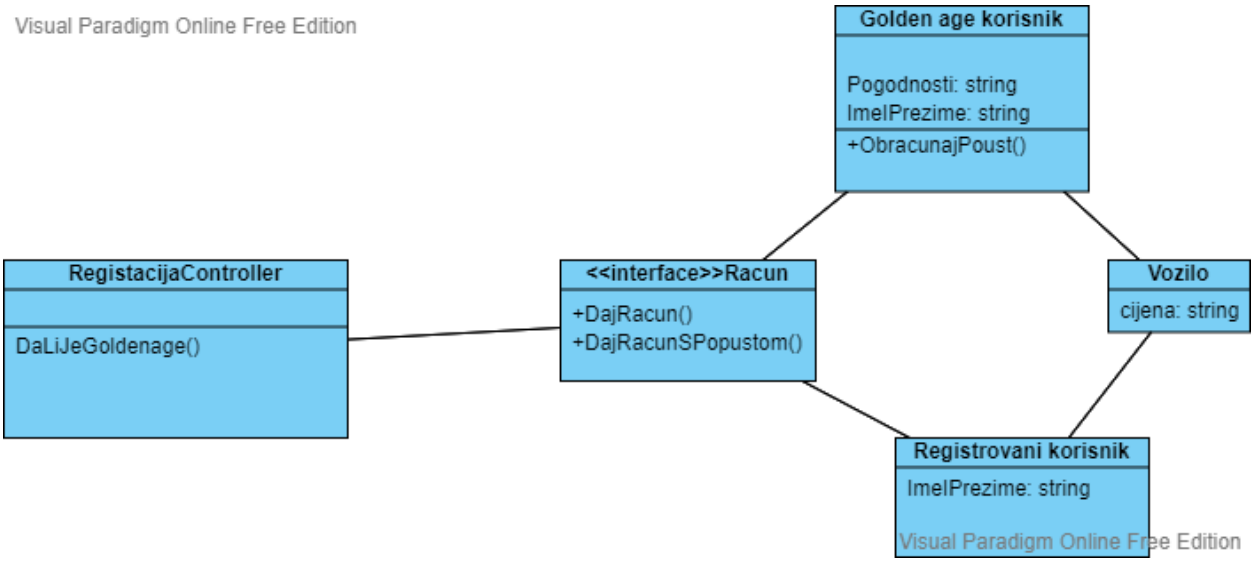
U našem sistemu možemo iskoristiti ovaj patern kod plaćanja vožnje s obzirom da imamo više načina plaćanja, npr. PlaćanjeKarticom, PlaćanjeGotovinom. Također, trebali bi imati interfejs Iplaćanje. Pored toga, trebamo napraviti i klasu Creator koja bi imala metodu FactoryMethod koja će odlučiti koju klasu instancirati.



4. *Abstract factory*

Abstract Factory patern nam omogućava da se kreiraju familije povezanih objekata. Na osnovu apstraktne familije produkata kreiraju se konkretne fabrike produkata različitih tipova i različitih kombinacija.

Rezervacija je omogućena I registrovanim i Golden age korisnicima. Međutim, samo Golden age korisnici mogu dobiti popust pri vožnji. Možemo napraviti apstraktnu factory klasu iz koje ćemo naslijediti dvije factory klase, za registrovane i Golden age korisnike. Shodno tome kakav korisnik je u pitanju, overrideana metoda koja kreira Racun bi u jednom slučaju vraćala instancu Racun sa popustom, a u drugom Racun bez popusta. Ako bismo u budućnosti željeli omogućiti specifičan pogodnosti nekoj trećoj kategoriji korisnika, to bi nam sada bilo znatno olakšano.



5. Builder patern

Uloga Builder paterna je odvajanje specifikacije kompleksnih objekata od njihove stvarne konstrukcije. Isti konstrukcijski proces može kreirati različite reprezentacije. Koristi se kada je neovisan algoritam za kreiranje pojedinačnih dijelova, kada je potrebna kontrola procesa konstrukcije, kada se više objekata na različit način sastavlja od istih dijelova.

Ovaj patern bi se mogao iskoristiti kada bi dodali još neke vrste vozila. Pri pristupu bi se tražio odabir vozila te na osnovu te informacije bi se prikazala stranica sa novostima i specifikacijama prilagođena toj vrsti vozila. Dodali bismo interfejs IBuilder i klasu VrstaVozilaBuilder.