Kreacijski paterni

1. Singleton patern

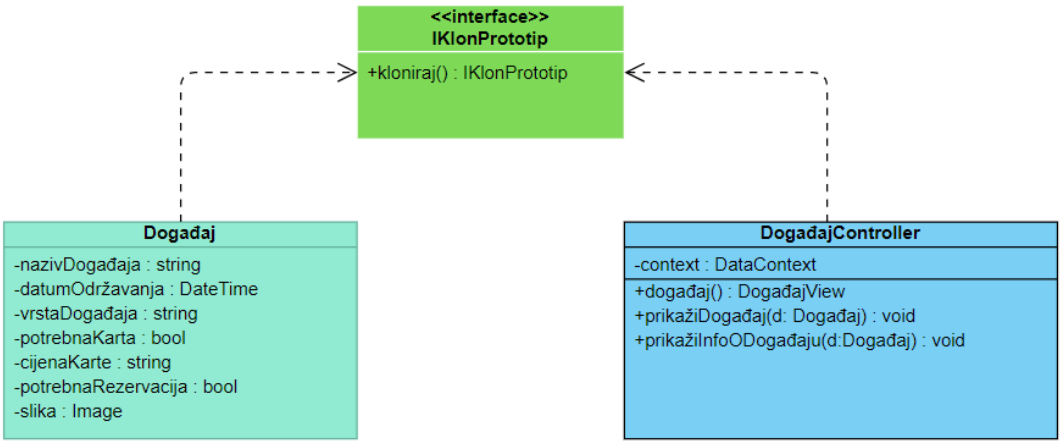
Uloga singleton paterna je da osigura da se klasa može instancirati samo jednom i da osigura globalni pristup kreiranoj instanci klase. Postoji više objekata koje je potrebno samo jednom instancirati i nad kojim je potrebna jedinstvena kontrola pristupa.

U našem sistemu, Singleton patern se može iskoristiti prilikom registracije korisnika. Dakle, klasa Registracija bi imala samo jednu instancu kojoj bi sve ostale klase mogle pristupiti nakon što se korisnik prijavi na sistem i na taj način dobiti sve informacije o korisniku (aktivnost na aplikaciji, broj kupljenih karata, napisane recenzije...).

2. Prototype patern

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

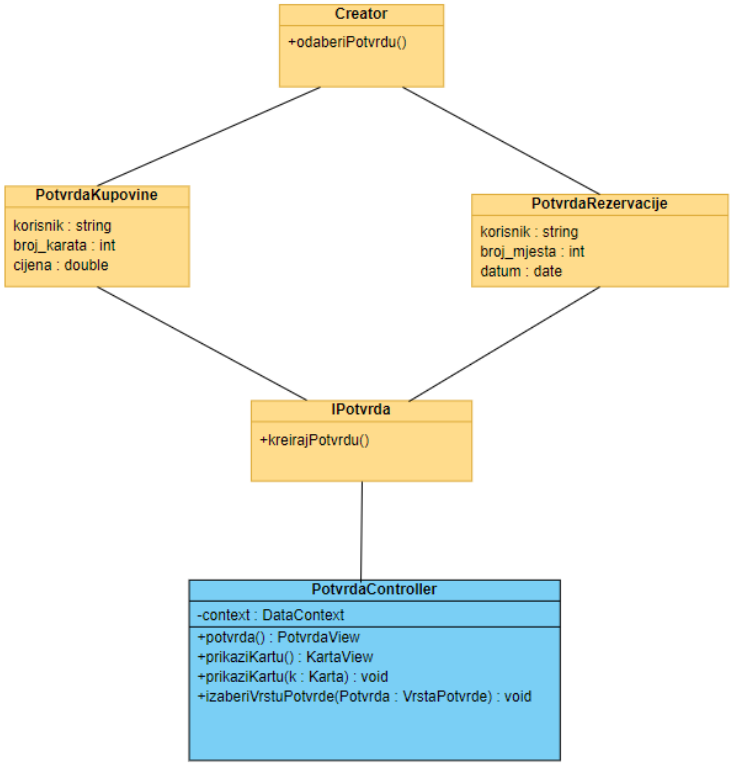
U našem sistemu ovaj patern se može primijeniti nad klasom Događaj. Ukoliko Poslovni korisnik želi dodati novi događaj kao što je utakmica, koncert ili film mnogo je lakše klonirati događaj koji već postoji i promijeniti samo neke informacije. Npr. ako u sistem želimo dodati događaj Koncert Dine Merlina koji se održava u BKC, a već je bio koncert Marije Šerifović, najlakše ćemo to uraditi ako se koncert klonira i onda se promijene neki detalji npr. vrijeme i cijena karte.



3. Factory Method patern

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

U našem sistemu možemo iskoristiti ovaj patern kada administrator šalje potvrdu. On tada ima mogućnost da bira između dvije vrste potvrda(PotvrdaKupovine i PotvrdaRezervacije). Dodali smo interfejs IPotvrda, te klasu Creator koja ima metodu +odaberiPotvrdu() koja odlučuje koju od ove dvije klase instancirati.



4. Abstract Factory patern

Abstract Factory patern omogućava da se kreiraju familije povezanih objekata. Na osnovu apstraktne familije produkata kreiraju se konkretne fabrike produkata različitih tipova i različitih kombinacija. Patern odvaja definiciju klase produkata od klijenta. Zbog toga se familije produkata mogu jednostavno izmjenjivati ili ažurirati bez narušavanja strukture klijenta.

Ovaj patern bi mogli iskoristiti kada bi u našem sistemu postojala mogućnost rezervacije putem sistema i plaćanja na licu mjesta/ u objektu. Kada bi korisnik rezervisao kartu putem našeg sistema, a istu kupiti na ulazu u sami objekat, nakon završene rezervacije korisnik bi dobio univerzalan kod koji bi pokazao na pri samom plaćanju. Za korisnike koji imaju 100+ bodova(i mogućnost na popust) kodovi bi se razlikovali od korisnika koji ne posjeduju popust. Kod bi bila neka apstraktna klasa, a naslijeđeni objekti bi bili npr popust_kod i kod_bez_popusta.

5. Builder patern

Uloga Builder paterna je odvajanje specifikacije kompleksnih objekata od njihove stvarne konstrukcije. Isti konstrukcijski proces može kreirati različite reprezentacije.

Ovaj patern u našem sistemu mogli bismo realizirati tako što bismo dodali interfejs IDogađajBuilder sa različitim metodama kao npr. dajDječijeDogađaje, dajDogađajeZaPenzionere, dajPovoljneDogađaje, dajPopularneDogađaje itd. Ove metode bi vraćale liste događaja i uzimale bi u obzir razne faktore (starosnu dob, cijenu karte, recenzije...). Zatim bi dodali klase DječijiDogađajiBuilder, DogađajZaPenzionereBuilder koje bi u sebi imale kao atribut listu svih dostupnih događaja u sistemu. Ove metode bi mogle u ovisnosti od nekih faktora preporučivati događaje koji su najbolji za korisnika.