Univerzitet u Sarajevu
Elektrotehnički fakultet u Sarajevu
Odsjek za računarstvo i informatiku

Kreacijski paterni

Objektno orijentisana analiza i dizajn

Članovi tima:
Nađa Alijagić
Hana Biševac
Anida Nezović

1. Singleton patern

Uloga ovog 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. Neki od njih su: thread pools (skupina niti), objekti koji upravljaju setovanjem registara, objekti koji se koriste za logiranje, objekti koji se koriste kao drajveri za razne uređaje kao što su printeri i grafičke kartice. U našem sistemu, ovaj patern bismo mogli primjeniti tako što bi klasu Administrator proglasili Singleton. Administrator je jedan i nema potrebe da se instancira često, također pošto je Administrator odgovoran za mnoge aktivnosti koje vrše korisnici, vodiči i zaposlenici poželjno je da pristup Administratoru imamo na globalnom nivou.

2. Prototype patern

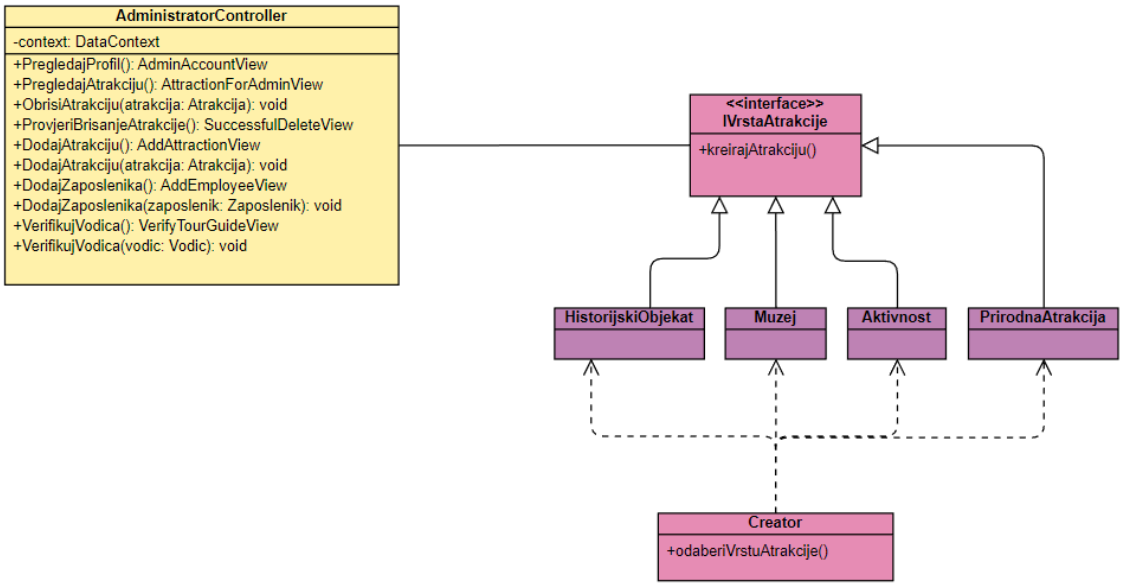
Uloga Prototype paterna je da kreira nove objekte klonirajući jednu od postojećih prototipa instanci. Prototype dizajn patern dozvoljava da se kreiraju prilagođeni objekti bez poznavanja njihove klase ili detalja kako je objekat kreiran.

U našem sistemu, mogli bismo ovaj patern primijeniti tokom kreiranja novog objekta klase RegistrovaniKorisnik, te tako dobiti cijeli jedan objekat kojem možemo izmijeniti neke atribute. To bi nam bilo korisno ako bismo imali korisnike sa istom lokacijom, te bismo tada mogli klonirati jedan objekat i izmijeniti ime, prezime, email i password. Naravno, ako bi novi korisnik imao isto ime ili prezime kao klonirani, ne bi čak ni bilo potrebe za ikakvim promjenama.

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.

U našem sistemu, imamo više vrsta atrakcija i tu bismo mogli primijeniti ovaj patern. Kreiranjem interfejsa *IVrstaAtrakcije* i podklasa *HistorijskiObjekat*, *Muzej*, *PrirodnaAtrakcija* i *Aktivnost*, Creator posjeduje metodu kojom će se odlučiti koja od ovih podklasa treba bit instancirana.



4. Abstract factory patern

Abstract factory patern omogućava da se kreiraju familije povezanih objekata/produkata. Na osnovu apstraktne familije produkata kreiraju se konkretne fabrike (factories) produkata različitih tipova i različitih kombinacija. Važan aspekt Abstract Factory paterna je da se cijela familija (fabrika) produkata može promijeniti za vrijeme izvršavanja aplikacije zbog toga što produkti implementiraju isti apstraktni interfejs. Mi trenutno u našem sistemu nemamo potrebu za implementacijom abstract factory paterna. Ukoliko bismo naš sistem nadogradili, naprimjer, sa funkcionalnosti da osim slika, korisnik može postavljati i video zapise. Slike i video zapisi su slični s obzirom da je za oboje potrebna kamera i imju još neke slične osobine.

5. Builder patern

Uloga Builder paterna je odvajanje specifikacije kompleksnih objekata od njihove stvarne konstrukcije. Isti konstrukcijski proces može kreirati različite reprezentacije. Upotreba Builder paterna se često može naći u aplikacijama u kojima se kreiraju kompleksne strukture. Mi smo ovaj patern u našem sistemu primjenili na klasi Tura. U našem sistemu Vodič može kreirati dvije vrste True (ako to želi) a to su mala tura (koja je unutar jednog grada) i velika tura (koja obuhvata nekoliko gradova).