Kreacijski paterni

Merjem Bećirović

Bakir Bajrović

Stefani Kecman

1. Singleton pattern

Mi smo predvidjeli da u našem sistemu postoje osobe koje će se brinuti o funkcionalnostima i podacima za prikaz. To su administrator i zaposlenik sistema. Iako zaposlenika može biti više, samo će jedna osoba obnašati funkciju administratora. Ako bismo to prikazivali na dijagramu, imali bismo klasu Administrator, koja bi bila singleton, zbog već objašnjenih razloga.

2. Prototype pattern

Imamo dvije ideje za implementaciju prototype patterna u našem programu.

1. Kada se kreira bankovni račun, koji je potrebna stavka svakog korisničkog računa, možemo kao prototip uzimati prethodno kreirani bankovni račun, s tim da ćemo njegov broj povećati za jedan u odnosu na prethodnog, a stanje na računu inicijaliziramo nekom početnom vrijednosti koja će biti ista za sve tek kreirane račune. Pošto su nam samo ta dva atributa od interesa za ovaj projekt, mi bismo svaki račun generisali na osnovu prototipa, uz već pomenutu izmjenu broja.

2. Pri kreiranju nove karte, za jednu utrku, provjeravamo da li je već ranije instancirana karta za istu utrku i u istoj kategoriji. Ukoliko jeste, znači da cijena, kategorija i utrka ostaju isti na novokreiranoj karti, u poređenju sa ranije instanciranom. U tom slučaju, mi ćemo iskorisititi ranije instanciranu klasu kao prototip. Jedino što će se promijeniti jeste redni broj karte koji će biti jedinstven za svaku novu kartu, bez obzira što je iz iste kategorije.

Na našem dijagramu smo implementirali drugu ideju protoype patterna.

3. Factory pattern

Ideja primjene ovog paterna u našem projektu se sastoji od sljedećeg. U aplikaciji imamo potrebu prikazivanja statistika i tabela za različite stvari. Uzet ćemo da je osnova toga apstraktna klasa Leaderboard. Ona može instancirati dva različita leaderboarda, leaderboard vozača i leaderboard ekipa. Pored toga, imamo i Statistiku, koja može biti statistika vozača i statistika ekipa. Imajući to u vidu, metoda „prikaži leaderboard“ će, u zavisnosti od potrebe, generirati drugačije instance, nekad će to biti leaderboard vozača, nekad leaderboard ekipa.

4. Abstract Factory pattern

Za ovaj patern, uzeli smo da korisnici, u zavisnosti od toga da li su premium ili obični registrovani korisnici, mogu kupovati karte samo one kategorije koja je predviđena za njihov status. Ovo nećemo implementirati, već samo dajemo ideju kako bismo izmijenili sistem da se navedeni pattern može primijeniti. Dakle, klasa RegistrovaniKorisnik ima svoje konkretne instance, korisnike, kao i klasa PremiumKorisnik. Klasa Karta bi, za potrebe ove implementacije, imala „podklase“ ili naslijeđene klase 1Sektor, 2Sektor, 3Sektor. Instance 1Sektora i 3Sektora su pozicionirane na startnoj liniji i na kraju staze, dok su sjedišta drugog sektora u sredini, samim tim i manje atraktivni. Na osnovu ovog patterna, sve instance PremiumKorisnika kupuju karte 1Sektor i 3Sektor klasa, ali ne i 2Sektor. 2Sektor kupuju samo instance RegistrovaniKorisnik. Ovo će se omogućiti dodavanjem nekog atributa, koji će biti inicijaliziran na navedeni način, odmah prilikom kreiranja instance.

5. Builder pattern

Ovaj patern bi omogućio korisniku koji se prvi put registruje na sistem da bira između dva načina registrovanja. On bi mogao odabrati opciju za automatsko generiranje username i password polja ili, ukoliko to želi, da ih sam unese. Dva načina registracije su predviđena samo za RegistrovaniKorisnik, ali ne i za PremiumKorisnik, pošto premium korisnici ostvaruju posebne pogodnosti i samim tim su nam potrebni njihovi tačni podaci.

U sklopu class dijagrama smo implementirali i builder pattern.