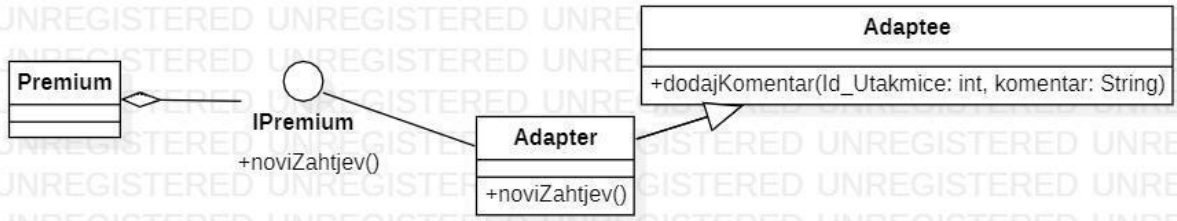
STRUKTURALNI PATERNI

1. ADAPTER PATERN

Adapter patern kreira novu adapter klasu koja služi kao posrednik između originalne klase i željenog interfejsa. Tim postupkom se dobija željena funkcionalnost bez izmjena na originalnoj klasi i bez ugrožavanja integriteta cijele aplikacije.

Naš sistem nema potrebu za implementacijom Adapter paterna. U slučaju dodavanja novih funkcionalnosti u neku klasu, ne bismo mijenjali strukturu klase nego bismo napravili *adapter klasu* i interfejs sa metodama koje su potrebne za ispunjenje novih funkcionalnih zahtjeva.

Naprimjer, ukoliko bismo željeli da dodamo novu funkcionalnost u klasu *Premium* i to da Premium korisnik ima mogućnost dodavanja komentara ispod utakmice onda bi imalo smisla dodati Adapter patern:



2. FASADNI PATERN

Osnovna namjena Facade paterna je da osigura više pogleda visokog nivoa na podsisteme (implementacija podsistema skrivena od korisnika). Ovaj patern olakšava korištenje aplikacije za korisnike koji ne moraju znati na koji način je izvedena implementacija funkcionalnosti koje koriste.

Ovaj patern je već implementiran u našem sistemu. Primjer upotrebe ovog paterna su klase Sudija i klasa Utakmica, gdje sudija unosi rezultat, kartone, golove i asistencije za klasu Utakmica bez ikakvih saznanja o načinu implementacije klase Utakmica.

//treba dodati metodu unosAsistencije u klasu Sudija, a iz parametara u toj klasi izbaciti klase Rezultat i Igrač, a staviti parametre int, string itd.

Još jedan primjer implementacije u našem sistemu je prikaz top strijelaca i asistencija direktno korisniku. Obzirom da korisnik ne treba da poziva pojedinačno objekte (igrače) te tražiti koji od

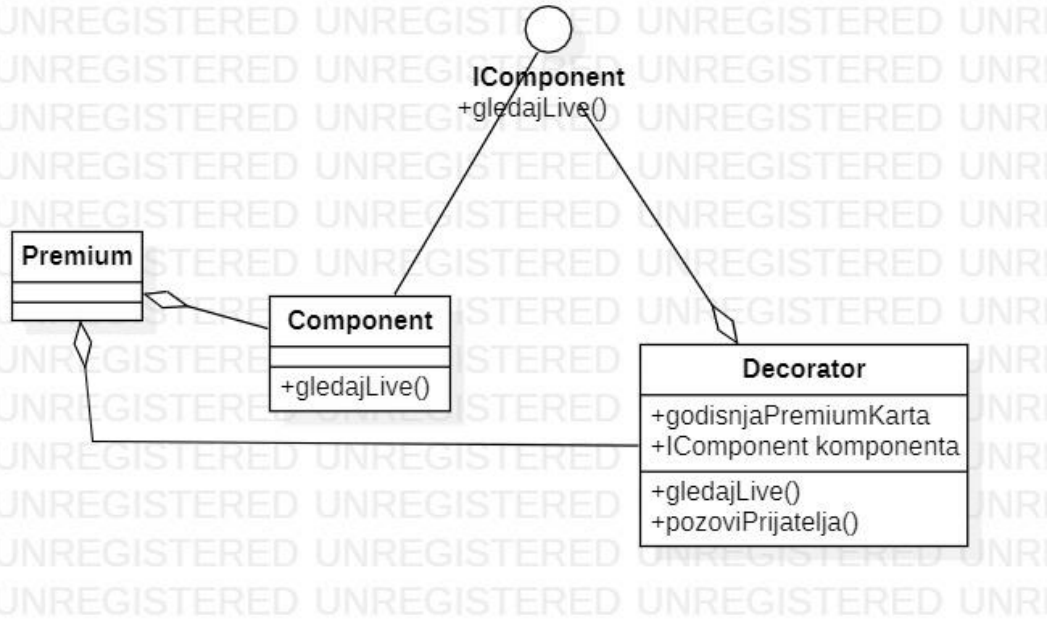
njih je postigao najveći broj golova, koristeći ovaj patern se to uspješno izbjegava pritiskom korisnika na button “top strijelci” pored tabele rezultata.

3. DEKORACIJSKI PATERN

Osnovna namjena Dekoracijskog paterna je da omogući dinamičko dodavanje novih elemenata i ponašanja (funkcionalnosti) postojećim objektima. Dakle, ukoliko imamo objekte koji imaju istu osnovu i razlikuju se u samo nekim pojedinostima, umjesto da pravimo niz izvedenih klasa, upotrijebit ćemo Dekoracijski patern.

U našem sistemu nije iskorišten ovaj patern.

Primjer kako bismo ovaj patern iskoristili na našem sistemu:



Kada bismo željeli da razdvojimo Premium korisnike i da korisnici koji uzmu godišnju pretplatu imaju neke dodatne pogodnosti kao što je, naprimjer, mogućnost da pozovu prijatelja da proba aplikaciju besplatno na mjesec dana, onda bismo koristili Decorator patern umjesto što bismo nasljeđivali nove klase iz klase Premium.

4. FLYWEIGHT PATERN

Ovo je strukturalni dizajn patern koji nam omogućava “natrpati” više objekata u slobodni prostor RAM-a tako što se dijele zajednički atributi između više objekata umjesto da se čuvaju svi podaci svakog objekta.

U našem sistemu se može iskoristiti na način da ako se učita ekipa da se pamti više informacija o njoj, ili kada se učita liga da nam učita/čuva podatke o više rezultata/utakmica između timova.

Međutim, za broj pobjeda odabrane ekipe, broj poraza, broj bodova i slično, ti podaci mogu se ujediniti u bazi pomoću tabele da se svi ti podaci drže na jednom mjestu te se iz te klasične tabele bodova mogu izvući ti podaci za pojedinačne ekipe obzirom da svi timovi imaju zajedničke atribute.

Problem u našem sistemu jeste da se bilježe podaci svake odigrane utakmice te strijelci, asistenti i kartoni na svakoj utakmici, međutim ti isti podaci se bilježe i pojedinačno za svakog igrača te se ne može to baš lahko ujediniti iz razloga što je drugačiji broj pogodaka za igrače, te se i sami igrači po se bi razlikuju jer to nisu iste osobe. Ukoliko korisnik želi da uvidi u rezultate i podatke željene utakmice, mi ne možemo držati atribut u klasi igrač vezano za datum nekog pogotka, a sa druge strane ukoliko korisnik želi da ima uvid u golove i asistencije jedinstvenog igrača, moguće je proći kroz svaku utakmicu od početka sezone i prebrojati te podatke, ali mnogo je zgodnije imate te podatke u klasi igrač zbog mnogo bržeg pristupa korisnika.

Jedan od načina na koji je moguće iskoristiti ovaj patern u našem sistemu je da recimo, ukoliko želimo funkcionalnost prikaza najboljeg tima u tekućoj godini, da vratimo tim koji je ubjedljivo bio najbolji prethodnih godina. Na taj način bismo uštedili vrijeme i memoriju.

5. BRIDGE PATERN

Ovaj patern odvaja kompleksnu klasu na dvije hijerarhije apstrakcija i implementacija tako da se mogu pojedinačne instance odvojeno implementirati u zavisnosti od njihove kompleksne strukture koja je “neovisna” od bazne klase. Ovaj patern je veoma važan jer omogućava ispunjavanje Open-Closed Solid principa, tj. nadogradnju novih potencijalnih klasa.

Primjer kako bi se implementirao ovaj patern bi bio da uvežemo lige iz neke države. Naprimjer, država BiH i sistem lige u Federaciji nije isti kao sistem lige u RS-u te tako možemo razgraničiti sistem izvlačenja parova na utakmicama između ta dva sistema lige.

6. COMPOSITE PATERN

Patern omogućava kompoziciju objekata strukture drveta što nudi mogućnost obrade podataka kao individualnih objekata. Koristi se kada svi objekti imaju različite implementacije nekih metoda, ali svakoj od njih se pristupa na isti način, pa Composite patern pojednostavljuje njihovu implementaciju.

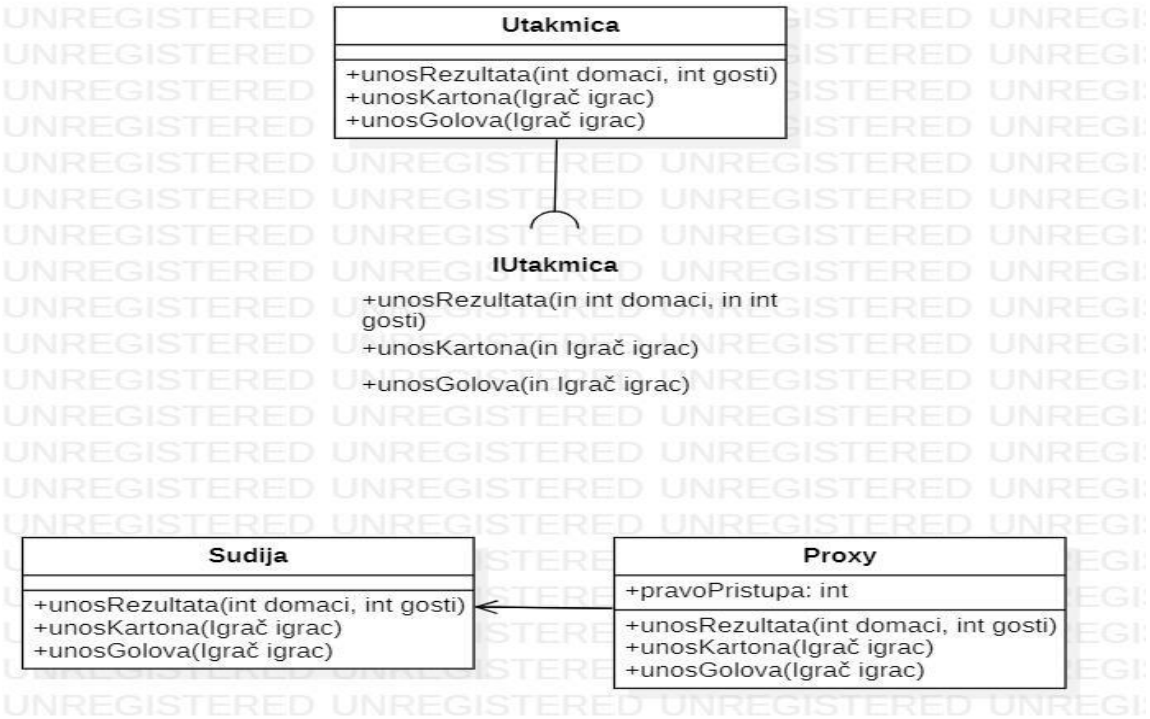
Primjer ovog paterna nije iskorišten na našem sistemu, ali navest ćemo primjer kako bi se mogao iskoristiti. Recimo, ukoliko bismo u klasu Igrač i u klasu Tim dodali metodu dajUkupnoKartona(), ona bi se isto pozivala ali bi se njena implementacija razlikovala s obzirom da se u klasi Igrač računa za samo jednog igrača, a u klasi Tim se računa za cijelu listu Igrača.

7. PROXY PATERN

Ovaj patern predstavlja kontrolu pristupa originalnom objektu i koristi se ukoliko je potrebno dodatno osiguranje objekata od pogrešne i zlonamjerne upotrebe. Često se koristi i za klase koje imaju osjetljive podatke ili spore operacije.

U našem sistemu postoji klasa Premium koja sadrži osjetljive podatke (broj kreditne kartice) i ovdje bi Proxy patern pronašao svoju ulogu u hashiranju podataka.

Također, klasa Sudija treba imati ograničeno pravo pristupa jer sudija nema pravo pristupati podacima utakmice koju ne sudi.



(Kreiranje kopije podataka iz baze da se ne radi direktno nad njima). Ovo se može iskoristiti za prikaz rezultata cijele lige jer se svakako vrši samo prikaz, a nije potrebna nikakva modifikacija istih.

Ovaj patern se također može upotrijebiti i pri plaćanju/odobravanju premium paketa.