Paterni ponasanja

1. Strategy patern

Uloga ovog patterna je da se izvrsavaju odredjeni algoritmi u ovisnosti od odabira korisnika. U nasoj aplikaciji ovaj patter bi mogli iskoristi kada admin pregleda zahtjev od strane vlasnika terena za dodavanje ili promjenu nekog terena, admin moze selektovati koji ce se algoritam izvrsiti da li to bio algoritam koji ce odbiti zahtjev I obavjestit korisnika ili algoritam koji ce prihvatiti isti I dodati ga u bazu podataka ukoliko je novi teren ili mijenjanje istog ukoliko je zahtjev bio promjena podataka terena.

2. State patern

Uloga state paterna je da se odredi ponasanje koje klase na osnovu nekog stanja (state-a) unutar klase. U nasoj aplikaciji ovaj patter bi mogli iskorisniti na sljedecim mjestima:

Pri verifikaciji trasnakcije za rezervaciju termina, imamo 2 stanja: Uspjesno i NijeUspjesno i na osnovu toga bi se radnja aplikacije izvrsavala na razlicite nacine, ukoliko je uspjesna obavjestava se korisnik o uspjesnosti i dodaje se rezervacija u sistem dok ako nije uspjesna obavjestava se da nije i daje korisniku mogucnost da opet unese podatke za trasakciju.

Ukoliko je rezervacija otvorena i zatvorena odnosno ako je proslo vrijeme termina, Rezervacija se uklanja iz liste trenutnih rezervacija i dodaje se u arhivu svih rezervacija.

3. Template Method Patern

Uloga Template method patterna je da se odvoji instanca poredjena od same klase i ukoliko trebamo da sortiramo instance neke klase mozemo do raditi po razlicitim atributima bez da mijenjamo implementaciju klase za poredjenje. U nasoj aplikaciji ovaj pattern bi se mogao koristiti pri filtriranju terena za prikaz korisniku, mozemo koristiti jednu klasu poredjenja I u zavisnosti od odabira korisnika tenere mozemo filtrirati preko lokacije, sporta, cijene, slobodnih termini idt.

4. Observer Patern

Observer patter radi na principu kreiranja one-to-many zavisnosti izmedju objekata I ukoliko se stanje glavnog objekta promjeni svi njegovi zavisni objekti ce primiti obavjest I adaptirati se na osnovu te promjene. U nasoj aplikaciji ovaj pattern mozemo iskoristit ukoliko vlasnik terena mora iz nekog razloga da zatvori svoj tener, obavjestavaju se svi korisnici koji imaju rezervacije za taj dan da su te rezervacije uklonjene I samim time mijenja se UI izgled svih rezervacija za svakog korisnika koji je imao rezervaciju za taj teren.

5. Iterator Patern

Uloga ovog paterna je da se kreira nacin za prolazak kroz kolekcije kompleksnih objekata bez da se prikazuje njihova interna struktura. U nasoj aplikaciji ovaj pattern bi se mogao iskoristit na iducim mjestima:

Za prolazak kroz sve rezervacije za odredjenog korisnika ili vlasnika terena, to bi uradili tako sto bi na osnovu Id-a razlikovali odredjene rezervacije i prikazivali bi se detalji samo potrebni za tu radnju, samim time bi sakrili sve privatne podatke/atribute klase od korisnika

Na nacin kao za rezervacije mogli bi isto koristiti ovaj patter za prolazak kroz sve transakcije bez prikaza senzibilnih detalja. Za prolazak kroz listu terena za odredjenog vlasnika terena itd. Ovaj pattern enkapsulira sve detalje prolaska kroz kolekciju kao i trenutnu poziciju i broj elemenata nakon iste, tako da vise iteratora moze da prolazi kroz istu kolekciju istovremeno, neovisno jedni od drugih

6. Chain of Responsibility pattern

Ovaj pattern sluzi ra razlozivanje odredjene radnje odnosno algoritma na vise razlicih klasa. U nasoj aplikaciji ovaj pattern bi se mogao realizirati na sljedeci nacin:

Pri zahtjevu za dodavanje novog tenera, prvo provjeravamo da li je naziv i opis terena pristojan, ukoliko jeste prelazimo na drugi handler koji ce provjeravati pristojnost I rezoluciju slika za teren, nakon toga se prelazi na drugi handler koji ce provjeravati lokaciju terena. Svaki handler nakon uspjesne validacije prenosi objekat na iduci handler I ukoliko se desi da jedan od handlera ne uspije verifikaciju, zaustavlja se zahtjev I ne dolazi do dalje provjere handlera koji se nalaze iza istog.

7. Mediator patern

Ovaj pattern se koristi za smanjenje zavisnosti klasa jednih od drugih u nasoj aplikaciji ovaj pattern bi mogli koristiti na svim mjestima gdje imamo neku formu (log in,sign up kao korisnik, sign up kao vlasnik itd.). Umjesto da svaki objekat tre forme bude povezan sa Dialogom koji ce izbacivati da li su sve potrebne informacije popunjenje, koristimo hub koji ce biti povezan sa svim tim objektima I ukoliko korisnik pritisne na dugme za validaciju, ako hub detektuje da bilo koji od objekata forme nije ispunjen zaustavlja se I obavjestava korisnika porukom na ekranu da nisu uneseni svi potrebni podaci ili da neki od unesenih podataka nisu ispravni.