### **Uvod**

S obzirom na dinamiku korisničkih akcija i potrebnu fleksibilnost, uvođenje paterna ponašanja (Behavioral Patterns) predstavlja optimalan pristup za povećanje skalabilnosti, održavanja i organizacije softverskog koda.

### **1. Strategy Pattern – za prilagodbu rangiranja i statistike**

* **Mjesto u sistemu:** Rangiranje korisnika, dodjela postignuća
* **Zašto:** Algoritmi za bodovanje, rangiranje i dodjelu priznanja mogu zavisiti od više faktora: vrste sporta, nivoa takmičenja, broja mečeva, ponašanja korisnika itd. Budući da se ovi kriteriji mogu vremenom mijenjati ili prilagođavati po sportu, neophodna je fleksibilna struktura.
* **Korist:** Omogućava da lako dodajemo nove algoritme bez izmjene postojećih klasa i logike, zadržavajući pritom konzistentan interfejs za pozivanje algoritama.

### **2. Observer Pattern – za obavijesti i notifikacije**

* **Mjesto u sistemu:** Slanje zahtjeva, prihvatanje/odbijanje, održavanje meča
* **Zašto:** Korisnici u aplikaciji često međusobno utiču jedni na druge. Na primjer, kada neko pošalje zahtjev za igru, kreator oglasa mora odmah primiti notifikaciju. Umjesto direktnog povezivanja objekata, bolje je da se posmatrači automatski obavještavaju o promjenama.
* **Korist:** Povećava fleksibilnost i modularnost – sistem reaguje na promjene bez strogih zavisnosti između klasa. Novi tipovi notifikacija se mogu dodati bez izmjena u postojećoj logici.

### **3. Command Pattern – za administraciju i undo radnje**

* **Mjesto u sistemu:** Akcije administratora (brisanje korisnika, slanje opomena, prihvatanje prijava)
* **Zašto:** Administratorske radnje moraju biti zapisane, moguće ih je sekvencirati, a u nekim slučajevima i poništiti. Korištenjem komandnog paterna svaka radnja se predstavlja kao objekat koji sadrži sve potrebne informacije za izvršenje, poništavanje.
* **Korist:** Centralizovano rukovanje akcijama, jednostavna implementacija undo/redo mehanizama, te mogućnost automatizovanog zapisivanja i analiziranja administratorskih aktivnosti.

### **4. State Pattern – za praćenje stanja meča**

* **Mjesto u sistemu:** Mečevi (zakazan, održan, otkazan, prijateljski, neuspješan)
* **Zašto:** Svaki meč tokom svog životnog ciklusa može da prelazi kroz različita stanja, od kojih svako ima vlastito ponašanje i pravila.
* **Korist:** Svako stanje implementira zasebnu logiku, bez upotrebe velikih i neodrživih if-else blokova. Dodavanje novih stanja postaje jednostavno i lokalizovano.

### **5. Template Method – za validaciju i kreiranje korisničkih akcija**

* **Mjesto u sistemu:** Registracija, prijava, postavljanje oglasa
* **Zašto:** Ove akcije imaju zajedničku strukturu toka – unos podataka, validacija, izvršenje akcije, povratna poruka. Međutim, detalji validacije i izvršenja se razlikuju.
* **Korist:** Omogućava definisanje osnovnog toka rada, dok se pojedinačni koraci lako mogu redefinisati u podklasama. Time se smanjuje dupliciranje koda i greške u logici.

### **6. Mediator Pattern – za upravljanje interakcijom korisnik–meč–recenzija**

* **Mjesto u sistemu:** Komunikacija između klasa: Korisnik, Meč, Recenzija, Oglas
* **Zašto:** Direktna komunikacija između više povezanih klasa može stvoriti visoku međuovisnost, što otežava proširenje sistema i testiranje.
* **Korist:** Mediator centralizuje komunikaciju i omogućava koordinaciju složenih interakcija bez da klase direktno zavise jedna od druge. Ovo pojednostavljuje kod i povećava fleksibilnost.