**ADAPTER PATTERN**

**Adapter** omogućava da klase s nekompatibilnim interfejsima međusobno sarađuju. Koristan je kada se želi omogućiti saradnja među postojećim klasama bez mijenjanja njihovog koda.

**Primjena u našem sistemu:**  
Adapter pattern se može koristiti za usklađivanje različitih sistema slanja notifikacija. Na primjer, jedna komponenta koristi e-mail notifikacije, druga koristi aplikacijske poruke. Adapter omogućava jedinstven interfejs za različite tipove obavještenja, čime se postiže interoperabilnost među podsistemima.

**FACADE PATTERN**

**Facade** pruža pojednostavljen interfejs prema složenom podsistemu, omogućavajući korisnicima da koriste kompleksnu funkcionalnost putem jedne „ulazne tačke“.

**Primjena u našem sistemu:**  
Možemo kreirati fasadu za rukovanje ispitima i izvještajima. Na primjer, jedna klasa IspitFacade može objediniti kreiranje ispita, validaciju datuma, dodjelu predmeta i automatsko generisanje izvještaja o ispitu. To omogućava jednostavno korištenje kompleksnih procesa u samo jednom pozivu.

**DECORATOR PATTERN**

**Decorator** omogućava dinamičko dodavanje dodatnog ponašanja objektima bez izmjene njihovog osnovnog koda.

**Primjena u našem sistemu:**  
Kod slanja notifikacija, možemo koristiti dekoratore za dodavanje dodatnih podataka: vremenske oznake, tip korisnika, potvrde o prijemu itd. Na primjer, bazna notifikacija se može dekorisati Timestamp-om ili oznakom prioriteta. Takođe, dekoratore možemo primijeniti pri prikazu predmeta ili ispita — dodajući im dodatne informacije (npr. broj studenata prijavljenih za ispit, broj ECTS bodova, prosječne ocjene...).

**🧱 BRIDGE PATTERN**

**Bridge** razdvaja apstrakciju od implementacije, omogućavajući nezavisno variranje obje strane.

**Primjena u našem sistemu:**  
Kod korisnika (student, profesor, dekan), ako želimo podržati više načina autentifikacije (lozinka, verifikacioni kod, prijava putem fakultetskog naloga...), bridge pattern omogućava povezivanje korisničkih tipova sa različitim strategijama autentifikacije bez kombinovanja svega u velik broj podklasa.

**PROXY PATTERN**

**Proxy** omogućava da se umjesto pravog objekta koristi zamjenski objekat koji može dodati logiku kao što su autorizacija, evidentiranje pristupa ili keširanje.

**Primjena u našem sistemu:**  
Pristup izvještajima ili ispitima može biti zaštićen. ProxyIzvjestaj ili ProxyIspit klasa može provjeravati da li korisnik ima pravo da vidi izvještaj ili učestvuje na ispitu (npr. samo studenti koji su prijavljeni na predmet mogu vidjeti detalje ispita ili izvještaja).

**COMPOSITE PATTERN**

**Composite** omogućava rad sa hijerarhijskim strukturama objekata tako da se pojedinačni i grupisani objekti tretiraju na isti način.

**Primjena u našem sistemu:**  
Ispiti ili izvještaji mogu biti organizovani u kolekcije po semestrima, predmetima ili tipovima ispita (kolokvij, završni, usmeni...). Kreiranjem interfejsa IIzvjestajElement, možemo omogućiti rad sa pojedinačnim izvještajima i kolekcijama izvještaja kao da su isti objekti — npr. prikaz svih izvještaja za jedan predmet u jednom semestru.