**Potreba za strukturalnim paterni u postojećem sistemu**

**1. Uvod**

Rastuća složenost sistema za upravljanje korisnicima, predmetima i notifikacijama uvodi niz izazova:

* **Čvrsta povezanost** (tight coupling) između klasa onemogućava jednostavnu zamjenu implementacija.
* Dupliranje logike slanja notifikacija u različitim dijelovima koda otežava održavanje.
* Različiti kanali slanja (e-mail, SMS, push) zahtijevaju promjene na više mjesta u kodnoj bazi.

Da bismo riješili ove probleme i poboljšali arhitekturu, uvodimo dva ključna strukturna paterna:

1. **Bridge**
2. **Facade**

**2. Bridge patern**

**2.1. Motivacija**

* Odvajanje apstrakcije (EmailNotification) od konkretne implementacije slanja (SmtpSender).
* Smanjenje direktne zavisnosti između klasa notifikacija i servisa za slanje.
* Omogućavanje lake zamjene ili proširenja "sendera" bez mijenjanja klasa notifikacije.

**2.2. Implementacija**

1. **Interfejs**
2. <<interface>> INotificationSender
3. + deliver(poruka: String): void
4. **Konkretni sender**
5. SmtpSender
6. – implementira INotificationSender
7. + deliver(poruka: String): void
8. **Abstrakcija notifikacije**
9. EmailNotification
10. – sender: INotificationSender «aggregation»
11. – poruka: String
12. + EmailNotification(sender: INotificationSender, poruka: String)
13. + send(): void
14. **Veze i kardinalnosti**
    * EmailNotification —agregacija→ INotificationSender (1 ⟶ 1)
    * SmtpSender —realizacija→ INotificationSender (bez kardinalnosti)
    * Postojeća klasa Notifikacija dobija metodu +posaljiNotifikaciju(): void koja delegira logiku slanja na EmailNotification.

**3. Facade patern**

**3.1. Motivacija**

* Kreiranje **jednostavnog ulaznog API-ja** za cijeli sistem: registracija, prijava, upis na predmet i slanje notifikacije.
* Sakrivanje detalja poziva više servisa i entiteta unutar jedne centralne tačke.
* Izolovanje klijenata od promjena u unutrašnjim klasama i servisima.

**3.2. Implementacija**

1. **Fasada-klasa**
2. SystemFacade
3. + registerUser(korisnickoIme: String, lozinka: String, …): Korisnik
4. + loginUser(korisnickoIme: String, lozinka: String): boolean
5. + subscribeToCourse(userId: int, courseId: int): void
6. + sendNotificationToUser(userId: int, poruka: String, kanal: String): void
7. **Veze i kardinalnosti**
   * SystemFacade —dependency→ Korisnik (1 ⟶ \*)
   * SystemFacade —dependency→ KorisnikPredmet (1 ⟶ \*)
   * SystemFacade —dependency→ Notifikacija (1 ⟶ \*)

**4. Zaključak**

Primjenom **Bridge** i **Facade** paterna:

* Održavamo **nisku povezanost** i **visoku koheziju**.
* Jednostavno **dodajemo nove kanale notifikacije** ili izmjene u API-ju bez velikih promjena.
* Povećavamo čitljivost i održivost koda, pripremajući sistem za buduće proširenje.