U ovom primeru koriste se dva dizajn paterna:

1. Strategy Pattern

- **Opis**: Strategy patern omogućava definisanje različitih algoritama ili ponašanja (strategija) i njihovu dinamičku primenu bez menjanja osnovne klase.
- **Kako se koristi u ovom primeru**:
    - Različite vrste korisnika (StandardUserStrategy, VIPUserStrategy, EntrepreneurStrategy) implementiraju zajednički interfejs UserTypeStrategy.
    - Klasa BankUser koristi instancu UserTypeStrategy za određivanje ponašanja korisnika (maksimalno povlačenje, specijalne usluge, pristup računima).
    - Ovaj pristup omogućava lako dodavanje novih tipova korisnika (novih strategija) bez menjanja klase BankUser.

---

2. Factory Method Pattern

- **Opis**: Factory Method patern omogućava kreiranje objekata bez direktnog instanciranja njihovih klasa, delegiranjem odgovornosti za kreiranje na posebnu metodu ili klasu.
- **Kako se koristi u ovom primeru**:
    - Klasa UserFactory sadrži metodu createUser koja na osnovu unetog tipa korisnika (standard, vip, entrepreneur) kreira odgovarajući objekat BankUser sa odgovarajućom strategijom.
    - Ovaj patern omogućava centralizovano upravljanje kreiranjem objekata, što olakšava proširenja i održavanje koda.

Prednosti ovog pristupa:

1. **Ekstenzibilnost**:
    - Dodavanje novih tipova korisnika moguće je bez modifikacije postojeće logike, već samo dodavanjem nove strategije i dopunom fabrike.
2. **Održavanje**:
    - Strategije i fabrika drže kôd modularnim, čineći ga lakšim za razumevanje i promene.
3. **Otvorenost za proširenja**:

- Kôd sledi **OCP (Open/Closed Principle)**: otvoren za proširenja, zatvoren za modifikacije.