1. Singleton patern: (implementacija)

Singleton patern se može primijeniti na klase poput Korisnik, Admin i LeaderBoard kako bi se osiguralo da postoji samo jedna instanca tih klasa u cijeloj aplikaciji. To bi moglo biti korisno kada želimo imati jedinstvenu instancu klase koju svi dijele. Na primjer, Admin klasa može biti implementirana kao singleton kako bi se osiguralo da postoji samo jedna instanca administratora sistema. Ovaj patern ćemo implementirati kako bi osigurali da za svaki grad postoji samo jedna instanca LeaderBorada.

2. Builder patern: (implementacija)

Builder patern se koristi kada želimo olakšati proces izgradnje složenih objekata koristeći korak-po-korak pristup. Ovaj patern omogućava konfiguraciju objekta s različitim atributima prije nego što se objekt konačno izgradi. Implementirat ćemo klasu "RegistrovaniKorisnik" preko builder paterna koji omogućava postavljanje različitih atributa korisnika, kao što su korisničko ime, lozinka, e-mail itd., prije nego što se konačno izgradi objekt korisnika.

3. Abstract Factory patern:

Abstract Factory patern se može koristiti kada želimo stvoriti skup srodnih objekata iste familije. Abstract Factory definiše interfejs za stvaranje tih objekata, dok Concrete Factory implementiraju taj interfejs i pružaju konkretne implementacije objekata. Na primjer, možemo imati "PlanIshraneFactory" koja definiše metode za stvaranje objekata "PlanIshrane", "Recept" i "Nutricionist". Zatim, možemo imati konkretne klase poput "JednodnevniPlanFactory" i "SedmodnevniPlanFactory" koje implementiraju abstract factory i pružaju konkretne implementacije tih metoda.

4. Factory Method patern:

Factory Method patern se može koristiti kada želimo delegirati proces kreiranja objekata podklasama. Apstraktna metoda u roditeljskoj klasi definiše interfejs za kreiranje objekata, dok svaka podklasa implementira tu metodu na svoj način kako bi kreirala specifičan objekt. Na primjer, klasa "PlanIshrane" može biti apstraktna klasa koja definiše apstraktnu metodu "kreirajPlanIshrane()". Zatim, podklase poput "JednodnevniPlan" i "SedmodnevniPlan" nasljeđuju "PlanIshrane" i pružaju konkretne implementacije metode "kreirajPlanIshrane()".

5. Prototype patern:

Prototype patern se može koristiti kada želimo stvoriti nove objekte kloniranjem postojećih objekata umjesto da ih stvaramo direktno. Objekti se mogu klonirati tako da implementiraju metodu "clone()" koja stvara kopiju objekta. Na primjer, klasa "Korisnik" može implementirati

mogućnost kloniranja kako bi se lako stvorile nove instance korisnika s istim atributima kao i postojeći korisnik.