Kreacijski paterni

1. *Singleton* patern je jedan od kreacijskih paterna i koristi se kada se neka klasa instancira samo jednom.

   Što se tiče naše aplikacije, klasa administrator može biti singleton klasa s obzirom na to da je administrator jedna osoba koja se veoma rijetko mijenja, a ima interakciju sa svim podacima u bazi.
2. *Prototype* patern omogućava smanjenje kompleksnosti kreiranja novog objekta tako što se uvodi operacija kloniranja. Na taj način prave se prototipi objekata koje je moguće replicirati više puta a zatim naknadno promijeniti jednu ili više karakteristika, bez potrebe za kreiranjem novog objekta nanovo od početka.

   Prototype patern bit će iskorišten pri kreiranju više termina predavanja/vježbi od strane profesora.
3. *Factory method* patern služi za omogućavanje instanciranje različitih vrsta podklasa koristeći factory metodu koja odlučuje koja će se podklasa instancirati i koja programska logika izvršiti. Pogodnu primjenu Factory method paterna u našem projektu nismo pronašli.
4. *Abstract factory* patern služi kako bi se izbjeglo korištenje velikog broja if-else uslova pri kreiranju različitih hijerarhija objekata. Ukoliko postoji više tipova istih objekata te različite klase koriste različite podtipove, te klase postaju fabrike za kreiranje objekata zadanog podtipa bez potrebe za specificiranjem pojedinačnih objekata.

   Pomenuti patern nije iskorišten u našoj aplikaciji.
5. *Builder* patern služi za apstrakciju procesa konstrukcije objekta, kako bi se kao rezultat mogle dobiti različite specifikacije objekta koristeći isti proces konstrukcije. Ovaj patern koristi se kako bi se izbjeglo kreiranje kompleksne hijerarhije klasa te kako bi se izbjegao kompleksni programski kod konstruktora jedne klase koja može imati različite konfiguracije atributa.

   Builder patern neće biti iskorišten u našem projektu. Hipotetički, mogli bismo ga iskoristiti u slučaju da napravimo klase izvedene iz klase Termin, tj. klase Predavanje i Tutorijal.