# KREACIJSKI PATERNI

## 1. SINGLETON

Singleton patern ima ulogu da osigura da se klasa može instancirati samo jednom i da osigura globalni pristup toj instanci klase.

U našem sistemu implementirat ćemo klasu Administrator kao singleton klasu tako što ćemo njen konstruktor proglasiti privatnim. Također, dodat ćemo statičku metodu getInstance koja se ponaša kao konstruktor i vraća uvijek isti objekat, a ne pravi novi. Dodat ćemo i statički atribut tipa Administrator.

## 2. PROTOTYPE

Prototip patern nam omogućava da kloniramo postojeće objekte bez da ovisimo o njihovim klasama. Koristimo ga kada je trošak kreiranja novog objekta prevelik.

Naš sistem ne predviđa ovaj patern, ali bi se on mogao implementirati tako što bi dodali interfejs IPrototip koji bi klonirao objekat tipa Majstor, preuzevši tako sve detalje iz te klase, a onda bi naknadno mijenjali podatke koji se razlikuju kod majstora (npr. lični podaci).

## 3. FACTORY METHOD

FactoryMethod patern nam omogućava kreiranje objekata tako da podklase odluče koju klasu instancirati. FactoryMethod instancira odgovarajuću podklasu preko posebne metode na osnovu informacije od strane klijenta ili na osnovu tekućeg stanja.

Naš sistem ne predviđa ovaj patern, ali bi se on mogao implementirati ako bismo imali posebne klase za različite tipove majstora. Tada bismo imali klase tipa Stolar, Zidar, Vodoinstalater itd, a koje bi interpretirale interfejs IMajstor. Potrebna nam je i klasa Creator koja posjeduje metodu FactoryMethod() koja odlučuje koju klasu će da instancira na osnovu profesije majstora.

## 4. ABSTRACT FACTORY

AbstractFactory  patern nam omogućava kreiranje familije povezanih objekata. Na osnovu apstraktne familije produkata kreiraju se konkretne fabrike produkata različitih tipova i različitih kombinacija.

Ovaj patern bi mogli primijeniti ukoliko bi, na primjer, postojale dvije vrste formulara za prijavljivanje grešaka, jedan za majstore i jedan za korisnike usluga. Imali bi klasu Formular, iz koje bi naslijedili klase FormularMajstor i FormularKorisnik, koje bi povezali sa odgovarajućim AbstractFactory klasama.

## 5. BUILDER PATERN

Builder patern koristimo kada odvajamo specifikacije kompleksnih objekata od njihove stvarne konstrukcije. Isti konstrukcijski proces može kreirati različite reprezentacije.

U našem sistemu ovaj patern iskoristit ćemo kod ograničenja rada majstora na određeni prostor. Kada korisnik prilikom filtriranja majstora odabere majstore u blizini (tj. majstori koji su ograničili lokaciju rada), onda se pomoću njegove lokacije (koju je naveo kod registracije) pomoću OgraničenjeBuilder klase, koja implementira IOgraničenjeBuilder interfejs, filtriraju majstori koji su ograničili rad na određenoj lokaciji, i prikazuju na stranici.