# Kreacijski paterni

## *Singleton patern*

Singleton patern služi kako bi se neka klasa mogla instancirati samo jednom. Na ovaj način može se omogućiti i instantacija klase tek onda kada se to prvi put traži. Osim toga, osigurava se i globalni pristup jedinstvenoj instanci te iste klase

Što se tiče singleton paterna idealna klasa na kojoj se može primijeniti je "KlasaZaBazuPodataka" jer nam treba samo jedna takva klasa, odnosno stvaranje više instanci te klase je suvišno, jer jednom kad izvršimo konekciju sa bazom, ne moramo to više raditi.

Sljedeća slika prikazuje kako bi se implementirao ovaj patern u našem projektu:


## *Prototype patern*

Jednostavno rečeno Prototype patern omogućava smanjenje kompleksnosti kreiranja novog objekta tako što se uvodi operacija kloniranja.

U našem trenutnom stanju dijagrama klasa, ne dolazi do potrebe da se išta treba klonirati, odnosno ne treba da dođe do potrebe da se prave identične duboke kopije drugih objekata. Eventualno u slučaju da se uvede neka dodatna funkcionalnost, kao što je na primjer ponavljanje kviza sa istim pitanjima. U tom slučaju bi se mogao klonirati početni kviz i na toj kopiji izvršiti premještanje pitanja i odgovora u pitanjima.

## *Factory Method patern*

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.

Jedina klasa koja je upitna kada dolazi do ovog paterna je klasa “Korisnik”, jer se jedino u njoj dešava nasljeđivanje. Jedine klase koje koriste korisnika su ljestvica i kurs, ali one imaju identičnu implementaciju za sve klase. Možda bi se moglo diskutovati da provjeravanje pristupa nekom kursu može da se riješi ovim paternom, ali to je posao proxy paterna. Eventualno da program treba da različito funkcioniše za različite klase, na primjer da obični korisnici mogu da uzmu određen broj kurseva dostupnih samo fakultetskim korisnicima pa da se  nekako treba računati broj ‘premium’ kurseva koje mogu uzeti.

## *Abstract Factory patern*

Abstract Factory patern se koristi 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.

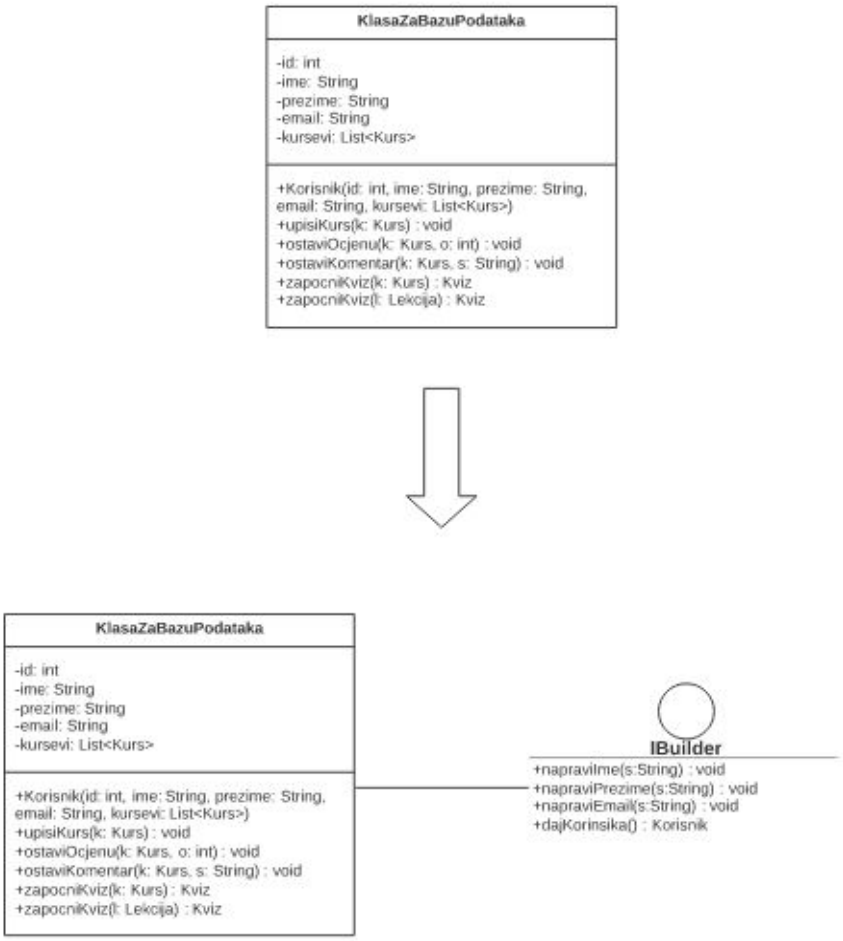
Opet i ovaj problem zbog nedostatka uslova za provjeru vrsta instanci nije potreban. Promjena u dijagramu kada bi mogao biti primjenjen ovaj patern je da klasa “Kviz” ima dvije podklase, za obični kviz i za ispit, pri čemu studenti mogu samo raditi ispite, a obični korisnici samo obične kvizove.

## *Builder patern*

Builder patern služi za apstrakciju procesa konstrukcije objekta, kako bi se kao rezultat mogle dobiti različite specifikacije objekta koristeći isti proces konstrukcije.

Ovaj patern bi se mogao koristiti kod klase “Korisnik” koja ima konstruktor koji prima veći broj parametara, a primjenom ovog paterna ne mora da obavi dodjeljivanje u konstruktoru odjednom.
Još jedna olakšavajuća stvar je što olakšava situacije kada npr. korisnik unese pogrešnu e-mail adresu, primjenom ovog paterna ne mora ponovo da definiše ime, prezime i slično, već samo definiše ponovo e-mail. Ovo također olakšava postupak dodavanja novih atributa ovoj istoj klasi.

Sljedeća slika prikazuje kako bi se implementirao ovaj patern u našem projektu:


KlasaZaBazuPodataka
-id: int
-ime: String
-prezime: String
-email: String
-kursevi: List<Kurs>
+Korisnik(id: int, ime: String, prezime: String, email: String, kursevi: List<Kurs>)
+upisiKurs(k: Kurs) : void
+ostaviOcjenu(k: Kurs, o: int) : void
+ostaviKomentar(k: Kurs, s: String) : void
+zapocniKviz(k: Kurs) : Kviz
+zapocniKviz(l: Lekcija) : Kviz
KlasaZaBazuPodataka
-id: int
-ime: String
-prezime: String
-email: String
-kursevi: List<Kurs>
+Korisnik(id: int, ime: String, prezime: String, email: String, kursevi: List<Kurs>)
+upisiKurs(k: Kurs) : void
+ostaviOcjenu(k: Kurs, o: int) : void
+ostaviKomentar(k: Kurs, s: String) : void
+zapocniKviz(k: Kurs) : Kviz
+zapocniKviz(l: Lekcija) : Kviz
IBuilder
+napraviIme(s:String) : void
+napraviPrezime(s:String) : void
+napraviEmail(s:String) : void
+dajKorinsika() : Korisnik