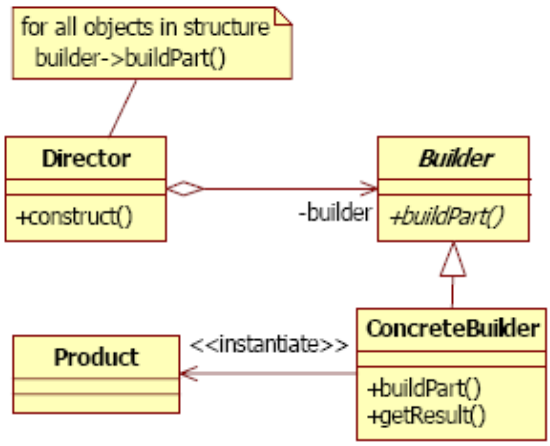
Design Pattern – analiza primera

1. Builder



Aplikacija za pregled studentskih ocena: Aplikacija radi u različitim režimima (zavisno od ulogovanog korisnika)

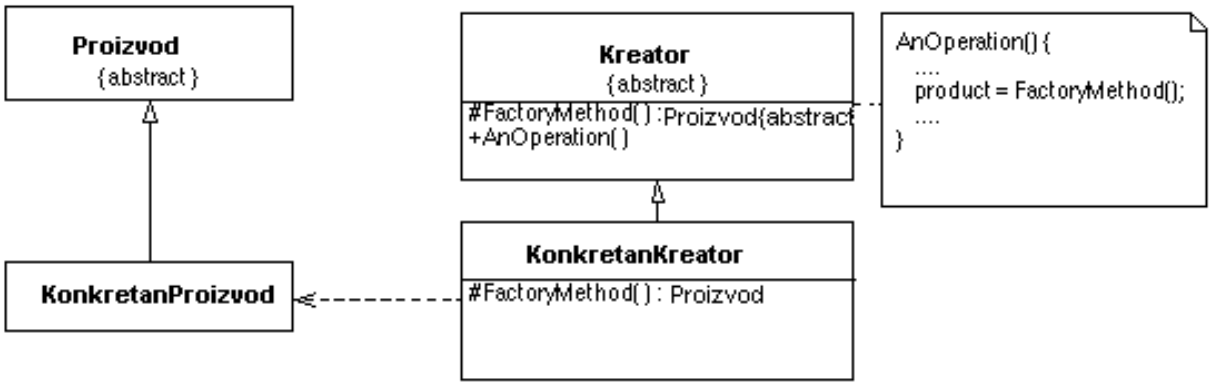
Builder - interface za „izgradnju“ forme koja zavisi od informacija dobijenih nakon logovanja na aplikaciju

ConcreteBuilder - specifične forme za različit tip korisnika (jer je specifičan i skup informacija koje se prikazuju, kao i dopustive informacije, tj. neka polja, dugmad ,... nisu vidljiva studentima, a jesu nastavnicima, administratorima,...)
ConcreteBuilderAdministratora -
ConcreteBuilderNastavnika -
ConcreteBuilderStudenta -

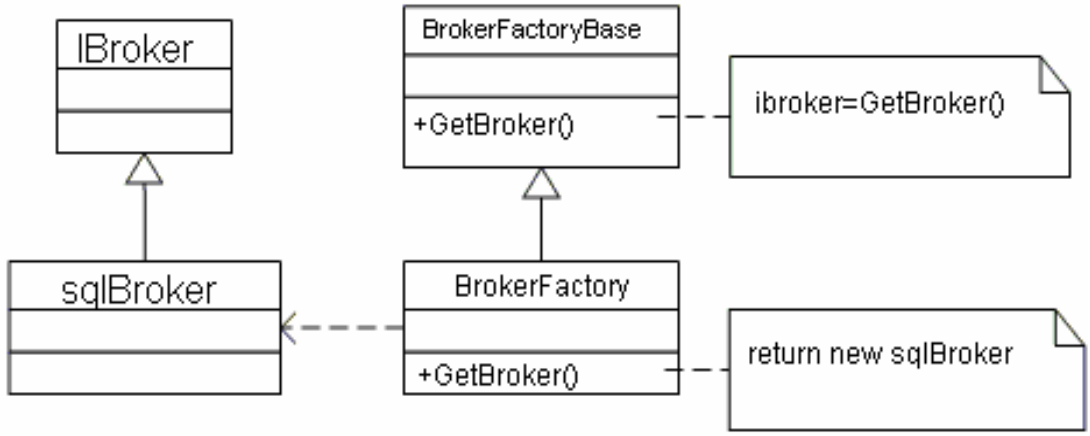
Product - finalna forma koju koristi aplikacija za konkretan slučaj
Director - aplikacija koja zahteva specifičnu formu (zavisno od login informacija)

2. Factory

Aplikacija koja može biti nezavisna od izbora DBMS

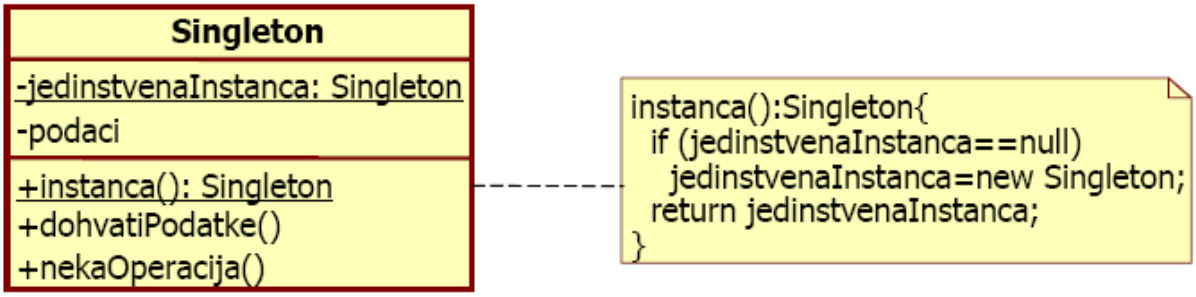


Svaki DBMS (IBM DB2, SQLServer, MySQL, Oracle) poseduje svoju sopstvenu implementaciju brokera koji nasleđuje interfejs Ibroker **(Proizvod)**. Prilikom izvršavanja aplikacije, biznis objekat ne poziva direktno SQL brokera, već klasu BrokerFactory koja na osnovu konfiguracionog



parametra vraća odgovarajuću instancu brokera za trenutno odabran tip baze podataka.

3. Singleton



Primer - Troslojna Web aplikacija: Da bi se prikazla korisniku željena HTML stranica rezultata, najpre treba aplikacija da pristupi bazi podataka, potom izvrši upit, preuzme podatke, izgeneriše HTML i tako gotovu stranicu pošalje nazad do korisnika.
Kako se svaka stranica sastoji od opštih elemenata koji su identični na svim stranama (naslov, logo, footer…) moguće je te elemente keširati i na taj način smanjiti broj poziva ka bazi podataka kad god trebamo te elemente. Za keširanje elemenata kojima se često pristupa može se koristiti

Singleton patern (npr. objekat CacheCollection i operacija getCache). Kada se instancira Cache klasa (tj. kada se učita vrednost elementa), Singleton će čuvati vrednoti elementa (u nekoj strukturi podataka, npr. hash tabeli). Tako da vrednosti kada su potrebne, više se ne dobavljaju iz baze, već iz keša. Time se značajno ubrzava rad aplikacije.

```
public class CacheCollection
{
private CacheCollection Cache;
public CacheCollection(){}
public static CacheCollection getCache()
{  if (Cache==null){ Cache=new CacheCollection();}
    return Cache;
}

private static Hashtable cacheElements = new Hashtable();
public static void Add(string key, object currentValue)
{
  if (!cacheElements.Contains(key))
   {  CacheElement newElement = new CacheElement(key,currentValue);
      cleanUp();
      cacheElements.Add(newElement.Key, newElement);
       number ++;
   }
 }
}
```

Zadaci:

1. KonzolnaAplikacija: Izrada pice
   public static void main(String[] args) {kreacija pica različitog sastava}

   Vrste pica: ItalijanskaPica, SrpskaPica
   Metodi: kreirajItalijanskuPicu, kreirajSrpskuPicu
   DeloviPice: Testo, Kora
    VrsteTesta: ItalijanskoTesto (brasno400, voda, so, jaja), SrpskoTesto (brasno500, mleko, secer)
    VrsteKora: ItalijanskaKora (sunka, pizzarela), SrpskaKora (kulen, pirotskiKackavalj)

2. KonzolnaAplikacija: IzradaImenika
   public static void main(String[] args) {kreacija pica različitog sastava}

   TipImenika: Beograd, Novi Sad
   Metodi: kreirajImenikBeograda, kreirajImenikNovogSada
   DeloviImenika: Adresa, TelefonskiBroj
    TipAdrese: NoviSad (21000+xxx), Beograd (11000+xxx)
    TipTelefonskogBroja: NoviSad(021-...), Beograd (011-...)

3. Implementirajte odgovarajući C# primer za konzolnu aplikacija
http://poincare.matf.bg.ac.rs/~jelenagr//rs2/DPjhp/example/Fabrika/RealanSlucaj.java

4. Omogućite rešenju zadatka 3 da može da radi i sa dokumentima tipa PrikazSoftvera (SekcijaIstorije, SekcijaSlika, SekcijaFunkcionalnost, SekcijaTest). Objasniti mehanizam dodavanja bilo kog novog proizvoda, tj. novog tipa dokumenta.

5. Analizirati primer primene Prototype obrasca i konstruisati primer primene prototype u C# jeziku. (Ovaj primer prikazuje obrazac Prototype pomoću koga se kreiraju nove instance klase Color kopiranjem prethodno postojećih objekata istog tipa koje je korisnik već definisao).

Izlaz

```
Cloning color RGB: 255,  0,  0
Cloning color RGB: 128,211,128
Cloning color RGB: 211, 34, 20
```

6. Aplikacija: analiza prodaje podataka iz baze
Rešenje: Kopirati informacije iz baze, izvršiti enkapsulaciju u objekte i obaviti analizu

Aplikacija 2: Potrebna je i potpuna drukčija analiza podataka iz iste baze
Rešenje: Kopirati informacije iz baze, izvršiti enkapsulaciju u objekte i obaviti analizu. Skupo!!!
Skupo je ponovno čitanje baze i kreiranje novih objekata.
Rešenje: upotreba Prototype – objasnite zašto? Ko će biti Client? Ko će biti ConcretePrototype?

7. Na programskom jeziku java kreirati klasu klijenta koji klonira Prototip objekte, tj. poziva kloniranu metodu public static Resource getResource(ResourceType type).
Ta metoda treba da vrati novu instancu resursa korišćenjem plitkog kopiranja. Tip resursa može biti DBResource i NetworkResource. U klasi klijenta realizovati i klasično instanciranje resursa (opet tipom DBResource i NetworkResource). Ispisati na standardni izlaz broj poziva inicijalizacija pri klasičnom instanciranju i instanciranju prototipom.
Izvršite program i uporedite broj inicijalizacija u ova dva režima instanciranja.
Na primer, izlaz:
Klasično instanciranje
Broj poziva dugih inicijalizacija = 1001
Prototype instanciranje
Broj poziva dugih inicijalizacija = 0

8. Potrebno je konstruisati formu korisničkog interfejsa koristeći *layout manager* `BiloGdeLayout`. Raspored komponenti određuje klasa `BiloGdeLayoutConstraints` koja definiše 7 parametara/atributa koji utiču na raspored komponenti (uvesti neka generička imena za atribute). Komponenta se dodaje na layout upotrebom metode `add(Component, BiloGdeLayoutConstraints)`. Upotrebom *prototype* dizajn šablona pojednostaviti dodavanje 10 komponenti ukoliko se zna da se svake dve susedne komponente (npr. 1 i 2, 4 i 5 itd.) razlikuju samo u po jednom parametru/atributu. `BiloGdeLayout` treba da implementira metodu `layout` koja će ispisati sve komponente i njihova ograničenja (svih sedam atributa svake komponente). **Napomena: Ne koristiti Swing klase niti bilo koje druge klase za rad sa grafičkim korisničkim interfejsima. Koristiti samo konzolu.**

9. Na programskom jeziku java kreirati konvertor formata dokumenata. Delovi dokumenta mogu biti sledećeg tipa:

- naslov
- podnaslov
- pasus
- nabrajanje
- definicija

Direktor kreira delove dokumenta na proizvoljan način pri čemu dokument može da sadrži više instanci istog tipa dela dokumenta. Napraviti simulaciju konverzije u npr. HTML, wiki i DocBook format. Posle kreiranja dokumenta Direktor ispisuje dokument na konzoli. Direktor ne sme da

sadrži referencu na konkretan Builder.