УНИВЕРЗИТЕТ У БЕОГРАДУ

ФАКУЛТЕТ ОРГАНИЗАЦИОНИХ НАУКА

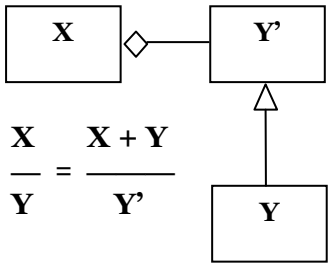
Катедра за софтверско инжењерство

СОФТВЕРСКИ ПАТЕРНИ

задаци 6. и 7. предавање

Аутор:

Др Синиша Влајић ванр.проф.



Београд - 2014.

Задатак Z-AD1:

1. Прилагодити клијенту методу **PrikaziStara()** са методом **PrikaziNova()**.

Додати програм на месту где су дате три тачке.

```
interface Target
{ ...
}

class Adapter ...
{ Adaptee adaptee;
  ...
}

class  Adaptee
{
 void prikaziStara(){System.out.println("Danas je lep dan.");}
}


class Client
{  ...
  Client(...){tar=tar1;}

  public static void main(String args[])
            {  ...
              ad.prikaziNova();
            }
}
```

Задатак Z-AD2: Додати програм на месту три тачке како би се добила порука: Danas je lep dan.

```
class Server
{
  void Prikazi(...) {c.PrikaziPoruku();}
}


class Client1
{
   Server s;
   Client1(Server s1) {...}

   public static void main(String args[])
             {
               ...
               Client1 c = new Client1(s);
               c.Prikazi();
             }

  void Prikazi(){s.Prikazi(...);}

  void PrikaziPoruku(){System.out.println("Danas je lep dan.");}
}
```

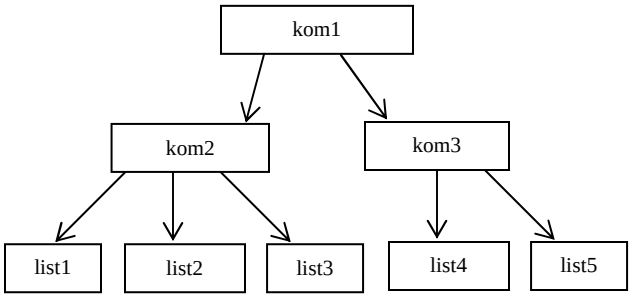
Задатак Z-BR1: Додати програм на месту три тачке како би се добиле поруке:
Sutra ce biti jos lepsi dan.
Danas je lep dan.

```
class Client2
   {
    Abstraction a;
    Client2 (Abstraction a1) {a=a1;}

     public static void main(String args[])
              {   ConcreteImplementorA ciA = new ConcreteImplementorA();
                  ConcreteImplementorB ciB = new ConcreteImplementorB();
                  RefinedAbstraction1 ra = new RefinedAbstraction1();
                  Client2 c = new Client2(...);
                  c.Prikazi(...); c.Prikazi(...);  }

     void Prikazi(Implementor i){a.Prikazi(i);}
    }

   abstract class  Abstraction { ... }

   class RefinedAbstraction1 extends Abstraction
   { void Prikazi(...){i.Prikazi();}
   }

   abstract class Implementor { ...}

   class ConcreteImplementorA extends Implementor
   { void Prikazi(){System.out.println("Danas je lep dan.");}
   }

   class ConcreteImplementorB extends Implementor
   { void Prikazi(){System.out.println("Sutra ce biti jos lepsi dan.");}
}
```

Задатак Z-BR2: Направити структуру класа где се било која класа из неког скупа класа A може повезати са било којом класом из неког скупа класа B.

Задатак Z – CO1: Додати програм на месту три тачке како би се на стандардном излазу добило:
Треба да се добије оваква структура објеката:

Kompozicija kom1 ima decu:
1.Kompozicija kom2 ima decu:
1.List1
2.List2
3. List3
2.Kompozicija kom3 ima decu:
1.List4
2.List5




```
class Client3
{
  public static void main(String arg[])
    {
      Kompozicija kom1 = new Kompozicija("kom1");  Kompozicija kom2 = new Kompozicija("kom2");
      Kompozicija kom3 = new Kompozicija("kom3");
      kom1.Dodaj(kom2);
   ...
      List list1 = new List("list1");  List list2 = new List("list2");  List list3 = new List("list3");
      ...
      List list4 = new List("list4"); List list5 = new List("list5");
    ...
      kom1.Prikazi();
    }
}
```

```
abstract class Komponenta
{ String Naziv;
 ...
 abstract void Prikazi();
 void Dodaj(Komponenta kom){}
}
```

```
class List extends Komponenta
{  List(String Naziv) {super(Naziv);}

   void Prikazi(){... }
}
```

```
class Kompozicija extends Komponenta
{ Komponenta deca[];   int brojDece;   static int  nivo = 0;

 Kompozicija(String Naziv) {super(Naziv); deca = new Komponenta[5];brojDece=0;}

 void Prikazi()
   {  System.out.println("Kompozicija " + Naziv + " ima decu:");
      for(int i=0;i<brojDece; i++)   {  nivo ++;  Pomeraj(i);     ...    nivo--;     }
   }

 void Dodaj(Komponenta kom){deca[brojDece]=kom; brojDece++;}
 void Pomeraj(int i) { for (int j=0;j<nivo;j++)  System.out.print("\t");   System.out.print((i + 1) + ".");  }
}
```

Задатак Z – CO2: Додати програм на месту три тачке како би програм могао да претражи компоненте композитне структуре преко назива компоненте и да прикаже поруку о томе.

```
class Client4
{
  public static void main(String arg[])
    {
      Kompozicija kom1 = new Kompozicija("kom1");    Kompozicija kom2 = new Kompozicija("kom2");
      Kompozicija kom3 = new Kompozicija("kom3");     kom1.Dodaj(kom2);     kom1.Dodaj(kom3);
      List list1 = new List("list1");  List list2 = new List("list2");   List list3 = new List("list3");
      kom2.Dodaj(list1);  kom2.Dodaj(list2);    kom2.Dodaj(list3);   List list4 = new List("list4");  List list5
= new List("list5");
      kom3.Dodaj(list4);  kom3.Dodaj(list5);

      System.out.println("Unesi naziv komponente:");
      Scanner in = new Scanner(System.in);
      String nazivKomp = in.next();

      if (kom1.Nadji(...))
          System.out.println("Komponenta:" + nazivKomp + " postoji!");
      else
          System.out.println("Komponenta:" + nazivKomp + " ne postoji!");
       }
}
abstract class Komponenta
{ String Naziv;
 Komponenta(String Naziv1){Naziv = Naziv1;}
 void Dodaj(Komponenta kom){}
 boolean Nadji(String Naziv1){return Naziv.equals(Naziv1);}
}

class List extends Komponenta
    { List(String Naziv) {super(Naziv);}}

class Kompozicija extends Komponenta
{ Komponenta deca[];
 int brojDece;

 Kompozicija(String Naziv) {super(Naziv); deca = new Komponenta[5];brojDece=0;}
 void Dodaj(Komponenta kom){deca[brojDece]=kom; brojDece++;}

 boolean Nadji(String Naziv1)
   { if (...) return true;
     for(int i=0;i<brojDece; i++)
       {  if (deca[i].Nadji(Naziv1))       ...      }
     ...
   }

 }
```

Задатак Z – CO3: Додати програм на месту три тачке како би програм могао да претражи компоненте композитне структуре преко назива компоненте и да прикаже сву децу тражене компоненте уколико она постоји.

```
import java.util.Scanner;

class Client5
{
  public static void main(String arg[])
    {
     Kompozicija kom1 = new Kompozicija("kom1");  Kompozicija kom2 = new Kompozicija("kom2");
     Kompozicija kom3 = new Kompozicija("kom3");
     kom1.Dodaj(kom2);  kom1.Dodaj(kom3);
     List list1 = new List("list1"); List list2 = new List("list2");   List list3 = new List("list3");
     kom2.Dodaj(list1); kom2.Dodaj(list2);  kom2.Dodaj(list3);
     List list4 = new List("list4");  List list5 = new List("list5");
     kom3.Dodaj(list4);   kom3.Dodaj(list5);
     System.out.println("Unesi naziv komponente:");
     Scanner in = new Scanner(System.in);
     String nazivKomp = in.next();

    Komponenta pom;
    pom = kom1.Nadji(nazivKomp);
    if (...)
       pom.Prikazi();
    else
       System.out.println("Ne postoji trazena komponenta!");
    }
 }
```

```
class List extends Komponenta
{  List(String Naziv) {super(Naziv);}
   void Prikazi(){System.out.println("Nema dece.");}
}
```

```
abstract class Komponenta
{ String Naziv;
  Komponenta(String Naziv1){Naziv = Naziv1;}
  void Dodaj(Komponenta kom){}
  Komponenta Nadji(String Naziv1){if (Naziv.equals(Naziv1)) return ...; return null;}
  String vratiNaziv() {return Naziv;}
  abstract void Prikazi();
}

class Kompozicija extends Komponenta
{ Komponenta deca[];
  int brojDece;
  Kompozicija(String Naziv) {super(Naziv); deca = new Komponenta[5];brojDece=0;}
  void Dodaj(Komponenta kom){deca[brojDece]=kom; brojDece++;}

Komponenta Nadji(String Naziv1)
   {   Komponenta pom = super.Nadji(Naziv1);
     if (pom!=null) ...;
     for(int i=0;i<brojDece; i++)    {   ...    }
     return null;
   }

  void Prikazi()
   { System.out.println("Deca od komponente "+ this.Naziv + " su:") ;
     for (int i=0;i<brojDece;i++)  {  System.out.println((i+1) + ". dete " + deca[i].vratiNaziv()); } }}
```

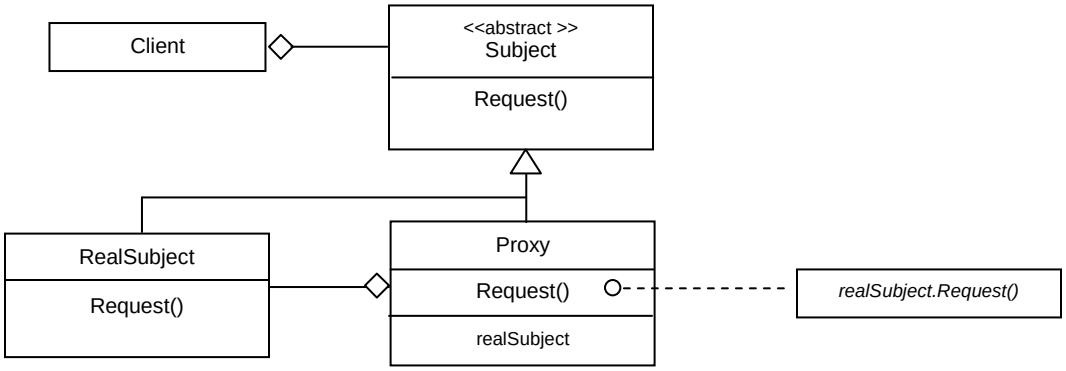
Задатак Z – DE1: Додати програм на месту три тачке како би се добиле поруке:
Juce je bilo oblacno vreme.
Danas je lep dan.
Sutra ce biti jos lepsi dan.

```
interface Komponenta
{ void prikazi();}

class  Dekorator implements Komponenta
{  ...
   Dekorator(Komponenta komp1) {...}
   public void prikazi(){komp.prikazi();}
}

class KonkretniDekoratorA extends Dekorator
{  KonkretniDekoratorA(Komponenta komp1) {super(komp1); }
   public void prikazi(){super.prikazi(); System.out.println("Danas je lep dan.");}
}
class KonkretniDekoratorB  extends Dekorator
{  KonkretniDekoratorB(Komponenta komp1) {super(komp1);}
   public void prikazi(){super.prikazi(); ... }
}

class KonkretnaKomponenta implements Komponenta
{ public void prikazi() { ... }
}

class Client6
{
    public static void main(String args[])
    {   KonkretnaKomponenta kk = new KonkretnaKomponenta();
        ...

        kdb.prikazi();
    }
}
```

Задатак Z-FA1: Додати програм на месту три тачке како би се добиле поруке:
Juce je bilo oblacno vreme.
Danas je lep dan.
Sutra ce biti jos lepsi dan.

```
class Client7
{

  public static void main(String args[])
  {  Fasada f = new Fasada();
    ...
    }
}


class Fasada
{ ...
}



class Podsistem1
{   void prikazipod1() {System.out.println("Juce je bilo oblacno vreme.");}}


class Podsistem2
{  void prikazipod2() {System.out.println("Danas je lep dan.");}}


class Podsistem3
{  void prikazipod3() {System.out.println("Sutra ce biti jos lepsi dan.");} }
```

Задатак Z-PR1: Осмислити пример на основу структуре Proxy патерна.



Задатак Z-PAT-REK1: Направити програм који ће да рачуна факторијел произвољног броја коришћењем рекурзије.

Задатак Z-PAT-REK2: Направити програм који ће да рачуна степен произвољног броја коришћењем рекурзије.

Задатак Z-PAT-REK3: Направити програм који ће да рачуна суму произвољног броја коришћењем рекурзије.

ZACOR1: Направити програм у коме такмичари одговарају на постављено питање. Када неко од такмичара тачно одговори на питање прекида се давање одговора осталих такмичара који нису одговорили на то питање (Обрада се прекида када први handler обради захтев).