**УНИВЕРЗИТЕТ У БЕОГРАДУ**

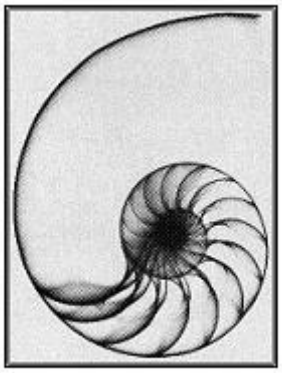
**ФАКУЛТЕТ ОРГАНИЗАЦИОНИХ НАУКА**

Катедра за софтверско инжењерство –
**Лабораторија за софтверско инжењерство**

# *СОФТВЕРСКИ ПАТЕРНИ*

## *ЗАДАЦИ (други тест)*

**Аутор: Др Синиша Влајић доц.**



Београд - 2017.

**Adapter патерн**

Задатак **Z-AD1**: Прилагодити клијенту методу PrikaziStara() са методом PrikaziNova() како би се добила порука:

Danas je lep dan.

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 **ZAD2**: 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.");}
}
```

**Bridge патерн**

Решење задатака **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
{ @Override
  void Prikazi(...){i.Prikazi();}}

abstract class Implementor { ...}

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

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

**Decorator патерн**

Задатак **ZDE1:** 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) {...}
    @Override
    public void prikazi(){komp.prikazi();}
 }

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

class KonkretniDekoratorB  extends Dekorator
{  KonkretniDekoratorB(Komponenta komp1) {super(komp1);}
  @Override
  public void prikazi(){super.prikazi(); ...}
}

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

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

**Facade патерн**

Задатак **ZFA1:** 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
 {   Podsistem1 pod1;
    ...
    Fasada() {pod1 = new Podsistem1();...}
    void prikazipod1(){pod1.prikazipod1();}
    ...
}

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

class Podsistem2 {   void prikazipod2() {... }

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

Кориснички захтев **RZPX1:** Додати програм на месту три тачке како би се добила порука:
Danas je lep dan!!!

```
class Client9
{  ...
   Client9(Subject sub1){sub = sub1;}

    public static void main(String[] args) {
        RealSubject rs = new RealSubject();
        ...
        Client9 cl = new Client9(pr);
        ...
    }

    void Request(){sub.Request();}
}



abstract class  Subject
{
   ...
}

class Proxy extends Subject
{
  RealSubject rs;
  Proxy(RealSubject rs1){rs=rs1;}
  @Override
  void Request(){rs.Request();}
}

class RealSubject extends ...
{@Override
 void Request(){...}
}
```

**Chain of responsibility патерн**

Задатак **ZCOR1**: Додати програм на месту три тачке како би се добила порука:

Juce je bio oblacan dan.

Danas je lep dan.

Sutra ce biti jos lepsi dan.

```
class Client
{public static void main(String[] args) {
      ConcreteHandler1 ch1 = new ConcreteHandler1(null);
      ...
      ConcreteHandler3 ch3 = new ConcreteHandler3(ch2);
      ...
   }
}
class Handler
{ Handler successor;
  Handler(Handler successor1){successor=successor1;}
  void HandleRequest()
    {if (...)
         successor.HandleRequest();
    }
}

class ConcreteHandler1 extends Handler
{  ConcreteHandler1(Handler successor) {...}
  @Override
  void HandleRequest() { super.HandleRequest(); System.out.println("Danas je lep dan."); } }

class ConcreteHandler2 extends Handler
{  ConcreteHandler2(Handler successor) {super(successor);}
  @Override
  void HandleRequest() { super.HandleRequest(); System.out.println("Sutra ce biti jos lepsi dan."); } }

class ConcreteHandler3 extends Handler
{  ConcreteHandler3(Handler successor) {super(successor);}
  @Override
  void HandleRequest() { ... super.HandleRequest();  }  }
```

**Command патерн**

Задатак **ZCOMM1**: Додати програм на месту три тачке како би се добила порука:

Danas je lep dan.

```
class Client
{   Receiver rec;
    Command cm;
    Client(){rec = new Receiver(); ...}
    Command getCommand(){return cm;}

    public static void main(String[] args) {
        Client cl = new Client();
        Invoker inv = new Invoker(...);
        inv.Execute();
    }
}

class Invoker
{ Command com;
  Invoker(Command com1){com=com1;}
  void Execute(){...}
}

interface Command
{
  void Execute();
}

class ConcreteCommand implements Command
{ Receiver rec;
  ConcreteCommand(Receiver rec1){rec = rec1;}
  @Override
  public void Execute(){...}
}

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

## Observer патерн

Кориснички захтев **ZPOBS1:** Осмислити програм који ће да реализује Observer патерн, у коме постоје Професор, Асистент и Студенти.

**State патерн**

Задатак **ZPST1:** Документ може да се нађе у три стања: Необрађен, Обрађен и Сторниран. Уколико је документ необрађен, он се може обрадити или сторнирати. Уколико је документ обрађен, он се може сторнирати али се не може поново обрадити. Уколико је документ сторниран он се не може обрадити нити поново сторнирати.

Додати програм на месту три тачке како би се добиле поруке:

Ponuda je obradjena!!!

Ne moze da se obradi ponuda, jer je vec obradjena!!!

Ponuda je stornirana!!!

Ne moze da se stornira ponuda, koja je vec stornirana!!!

Ne moze da se obradi stornirana ponuda!!!

```
abstract class Stanje
{abstract Stanje obradi();
 abstract Stanje storniraj();
}

class Obradjen extends Stanje
{ @Override
  Stanje obradi() {System.out.println(...); return this;}
  @Override
  Stanje storniraj() {System.out.println("Ponuda je stornirana!!!"); return ...;}
}

class Neobradjen extends Stanje
{@Override
 Stanje obradi() {System.out.println("Ponuda je obradjena!!!"); return ...;}
 @Override
 Stanje storniraj() {System.out.println(...); return new Storniran();}
}

class Storniran extends Stanje
{ @Override
  Stanje obradi() {System.out.println(...); return this;}
  @Override
   Stanje storniraj() {System.out.println("Ne moze da se stornira ponuda, koja je vec stornirana!!!");
return ...;}}
```

```
class Dokument
{ Stanje stanjeDokumenta;
  Dokument(){stanjeDokumenta = new Neobradjen();}
  void obradi(){stanjeDokumenta = stanjeDokumenta.obradi();}
  void storniraj(){stanjeDokumenta = stanjeDokumenta.storniraj();}
}

class ZPST1 {
    public static void main(String[] args) {
       Dokument d = new Dokument();
       d.obradi();
       d.obradi();
       ...
       d.storniraj();
      ...
   }}
```

### Template method патерн

Кориснички захтев **ZPTM1:** Написати програм помоћу template method патерна којим се описује радни дан студената на Факултету.