УНИВЕРЗИТЕТ У БЕОГРАДУ

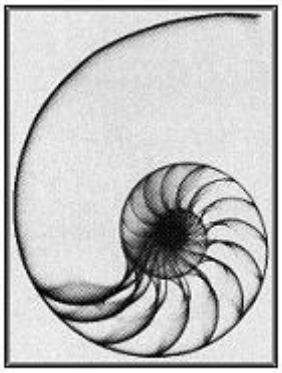
ФАКУЛТЕТ ОРГАНИЗАЦИОНИХ НАУКА

Катедра за софтверско инжењерство –
Лабораторија за софтверско инжењерство

СОФТВЕРСКИ ПАТЕРНИ

ПИТАЊА И ЗАДАЦИ

Аутор: Др Синиша Влајић доц.



Београд - 2017.

Питања:

А) Основе и анализа патерна

А1. Шта је патерн?

А2. Шта су тронивојски патерни?

А3. Која је улога патерна у развоју софтвера?

А4. Шта треба урадити да би пројектовани софтверски систем био примењив као решење скупа различитих проблема?

А5. Када треба користити софтверске патерне?

А6. Објаснити однос између патерна, реда и хаоса.

А7. Дати пример патерна у свакодневном животу.

А8. Објаснити општи GOF облик патерна пројектовања.

А9. Објаснити структуру проблема код патерна.

А10. Објаснити структуру решења код патерна.

А11. Објаснити дефиницију GOF патерна преко симетрије.

Б) Програмски концепти патерна пројектовања

Б1. Објаснити наслеђивање и компатибилност објектних типова. Дати пример.

Б2. Објаснити разлику између раног и касног повезивања објекта са методом. Дати пример.

Б3. Објаснити шта су апстрактне класе и интерфејси. Дати пример.

Б4. Објаснити како се брокер базе података преко структуре решења код патерна повезује са доменским класама.

Б5. Дати имплементацију општег облика ГОФ патерна пројектовања.

Б6. Како се праве генеричке методе. Дати пример.

Б7. Које се правило користи код прављења генеричких метода?

Б8. Објаснити како се преко структуре решења код патерна може решити повезивање Јава програма са било којим системом за управљање базом података.

Ц) Патерни за креирање објеката

Ц1. На сопственом примеру дефинисати: а) кориснички захтев за Abstract Factory, Builder и Factory method патерн. б) пример структуре X патерна сходно дефинисаном корисничком захтеву. ц) програмски код клијент класе код X патерна сходно дефинисаном корисничком захтеву д) програмски код серверске класе код X патерна сходно дефинисаном корисничком захтеву X може бити Abstract Factory, Builder и Factory method патерн.
Ц2. Објаснити разлику између Abstract Factory, Builder и Factory method патерна.
Ц3. Надоградити Abstract Factory патерн тако да не зависи од броја производа који чине финални производ.
Ц4. Дати сопствени пример за Prototype патерн.
Ц5. Дати сопствени пример за Singleton патерн.

Задаци:

Z1. Шта је потребно урадити у програму да би се на стандардном излазу добила порука "Sutra ce biti lep dan." . Наредбе програма не смеју да се бришу. Наредбе програма могу да се додају на местима где има три тачке.

```
abstract class Z
  {  abstract Prikazi ();
  }

class X  …
{    Y y;
      X() { y = new Y(); }
      void op(Z z) { z.Prikazi(); }
      void Prikazi(){System.out.println("Sutra ce biti lep dan");}
}

class Y …
{   void Prikazi() {System.out.println("Danas je lep dan");}
}


class Z1
{  public static void main(String arg[])
      { X x = new X();
        …
        x.op(…);
     }
}
```

Z2. Шта је потребно урадити у програму да би се на стандардном излазу добила порука: "Danas je lep dan." . Наредбе програма не смеју да се бришу. Наредбе програма могу да се додају на местима где има три тачке.

```
interface Z
{  abstract void Prikazi ();
}

class X …
{
     public void Prikazi(){System.out.println("Sutra ce biti lep dan");}
}

class Y …
{
      public void Prikazi() {System.out.println("Danas je lep dan");}
}

class Z2
{  public static void main(String arg[])
     { X x = new X();
       Y y = new Y();
        Z z = …;
        z.Prikazi();
    }
}
```

Z3. Шта је потребно урадити у програму да би на стандардном излазу добили поруке “Danas je lep dan. Sutra ce biti lep dan.” у наведеном редоследу. Наредбе програма не смеју да се бришу. Наредбе програма могу да се додају на местима где има три тачке.

```
interface Z
{  abstract void Prikazi ();
}

class X …
{    public void Prikazi(){System.out.println(“Sutra ce biti lep dan”);}
}

class Y …
{    public void Prikazi() {System.out.println(“Danas je lep dan”);}
}


class Z3
{  public static void main(String arg[])
     { X x = new X();
       Y y = new Y();
        Z z ;
        …
        z.Prikazi();
        …
        z.Prikazi();
    }
}
```

Z4. Променити наведени програм који је тежак за одржавање и надоградњу у одрживи програм у складу са концептом патерна.

```
class Y1
{    void Prikazi(){System.out.println("Sutra ce biti lep dan");}
}

class Y2
{    void Prikazi() {System.out.println("Danas je lep dan");}
}

class X
  {  Y1 y1;
     Y2 y2;
      X() { y1 = new Y1(); y2 = new Y2(); }
      void op(int y)
           {  if (y == 1)
                y1.Prikazi();
              if (y == 2)
                y2.Prikazi();
           }
  }


class Z4
{  public static void main(String arg[])
      {  X x = new X();
         x.op(1);
         x.op(2);
     }
}
```

Z5. Довршити следећи програмски код, како би се на стандардном излазу добила следећа порука: *Збир два броја је: 8*
У постојећем програму не сме се избацити или ставити под коментар ниједна линија програма. Користити објекте из главног програма за добијање жељеног резултата. Додати наредбе на местима где су дате три тачке (...).

```
class Izraz
{ int clan1;   char operator;   int clan2;
  Izraz(int clan1p,int clan2p, char operatorp){clan1 = clan1p; clan2 = clan2p; operator = operatorp;}
}

interface KRacunaj { int Racunaj(Izraz iz);}
class Saberi implements KRacunaj {    public int Racunaj(Izraz iz) {return iz.clan1 + iz.clan2;}}
class Oduzmi implements KRacunaj {  public int Racunaj(Izraz iz) {return iz.clan1 - iz.clan2;}}

class Z5
{  public static void main(String str[])
   {    KRacunaj rac = null;
        Saberi sa = new Saberi();
        Oduzmi od = new Oduzmi();
        Izraz iz = new Izraz(5,3,'+');
        if (iz.operator == '+')
              {...}
        else
             if (iz.operator == '-')  {...}
             else {}
       System.out.println(rac.Racunaj(iz));
  }
}
```

Z6. Довршити следећи програмски код, како би се на стандардном излазу добила следећа порука: *Збир два броја је: 8*
У постојећем програму не сме се избацити или ставити под коментар ниједна линија програма. Користити објекте из главног програма за добијање жељеног резултата. Додати **једну** наредбу на месту где су дате три тачке (...)

```
class Izraz
{ int clan1;
  KRacunaj operator;
  int clan2;
  Izraz(int clan1p,int clan2p, KRacunaj operatorp){clan1 = clan1p; clan2 = clan2p; operator = operatorp;}
  int Racunaj(){return operator.Racunaj(this);}
  int vratiClan1(){return clan1;}
  int vratiClan2(){return clan2;}
}

interface KRacunaj
{ int Racunaj(Izraz iz);}

class Saberi implements KRacunaj
{public int Racunaj(Izraz iz) {System.out.print("Zbir dva broja je:"); return iz.vratiClan1() + iz.vratiClan2();}}

class Oduzmi implements KRacunaj
{public int Racunaj(Izraz iz) {System.out.print("Razlika dva broja je:"); return iz.vratiClan1() - iz.vratiClan2();}}

class Z6
{
  public static void main(String str[])
  { Saberi sa = new Saberi();
    Oduzmi od = new Oduzmi();
    Izraz iz = new Izraz(5,3,sa);
     ...
  }
}
```

Z7. Довршити следећи програмски код, како би се на стандардном излазу приказао низ бројева у растућем редоследу.
У постојећем програму не сме се избацити или ставити под коментар ниједна линија програма. Додати наредбу на месту где су дате три тачке (...).

```
class Niz
{   int niz[];
    Niz(int niz1[]) {niz=niz1;}
    void Sortiraj(OperatorPoredjenja op)
      { int pom = 0;
        for(int i=0; i<niz.length-1;i++)
              for(int j=i+1; j<niz.length;j++)
                  if( ...)
                    { pom = niz[i]; niz[i]=niz[j];  niz[j]=pom; }
               }
              void prikaziNiz()
                { System.out.println("Elementi niza su:");
                 for(int i=0;i < niz.length;i++)    {System.out.println(niz[i]);}
                 }
}

interface OperatorPoredjenja { boolean Poredi(int x,int y);}

class Vece implements OperatorPoredjenja
{ public boolean Poredi(int x,int y) {...}}

class Manje implements OperatorPoredjenja
{ public boolean Poredi(int x,int y) {...}}

class Z7
{ public static void main(String str[])
              { int n[] = {7,3,9,2,1,4,8};
                 Niz niz = new Niz(n);
                 Vece ve = new Vece();
                 Manje ma = new Manje();
                 niz.prikaziNiz();
                 niz.Sortiraj(...);
                 niz.prikaziNiz();
              }
}
```

Z8. У ниже наведеном примеру написати програмски код који неће мењати main() методу ако се дода нови геометријски облик (npr. Kvadrat) за који ће моћи да се види површина. Преко main() методе звати методу Povrsina() за било који геометријски облик.

```
import java.io.*;

class Z8
{   public static void main(String str[]) throws IOException
     {  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        Krug kr = new Krug(7);
        Pravugaonik pr = new Pravugaonik(5,2);
        System.out.println("Izaberi geometrijski oblik za koji se racuna povrsina (1-krug, 2-pravugaonik):");
        String izbor = br.readLine();
             if (izbor.equals("1")==true)
                 kr.Povrsina();
             if (izbor.equals("2")==true)
                 pr.Povrsina();
     }
}

 class Krug
  { int r;
    Krug(int r1) {r=r1;}
    void Povrsina(){System.out.println("Povrsina kruga je:" + r*r*3.14);}
  }

 class Pravugaonik
   { int a,b;
     Pravugaonik(int a1,int b1) {a=a1;b=b1;}
     void Povrsina(){System.out.println("Povrsina pravugaonika je:" + a*b);}
  }
```

Z9: Довршити следећи програмски код, како би се на стандардном излазу добила следећа порука: Povrsina kruga je: 100.94
У постојећем програму не сме се избацити или ставити под коментар ниједна линија програма. Додати наредбу на месту где су дате три тачке (...)

```
class Z9
{
 public static void main(String str[])
   { GeometrijskiOblik go = new ...
     System.out.println("Povrsina kruga je:" + go.Povrsina());
   }
}

interface GeometrijskiOblik
{ ...
}

class Krug ...
{ ...
  Krug(double r1){r=r1;}
  public double Povrsina() {return r*r*3.14;}
}
```

Z10: Довршити следећи програмски код, како би се на стандардном излазу добила следећа порука: *Povrsina kruga je: 100.94*
У постојећем програму не сме се избацити или ставити под коментар ниједна линија програма. Додати наредбу на месту где су дате три тачке (...)

```
class Z10
{ public static void main(String str[])
  { GeometrijskiOblik go = ...
    System.out.println("Povrsina kruga je:" + go.Povrsina());
  }
}

abstract class GeometrijskiOblik
{...
}

class Krug ...
{ …
  Krug(double r1){r=r1;}
  public double Povrsina() {return r*r*3.14;}
}
```

Z11: Довршити следећи програмски код, како би се на стандардном излазу добила следећа порука: *Konkretni server2 je obradio poziv*
У постојећем програму не сме се избацити или ставити под коментар ниједна линија програма. Додати наредбу на месту где су дате три тачке (...)

```
class Z11
{
  public static void main(String arg[])
    {  ...              ks =  ...
       Klijent kl = new Klijent(ks);
       kl.obradi();
    }
}

class Klijent
{ ApstraktniServer as;
  Klijent(ApstraktniServer as1) {as = as1;}
  void obradi(){...}
}


interface ApstraktniServer
{ void obradi();
}

class KonkretniServer1 ...
{  public void obradi(){...}
}

class KonkretniServer2 ...
{ public void obradi(){...}
}
```

Z12: Довршити следећи програмски код како би био задовољен следећи кориснички захтев: Направити сопствену класу Изузетак која ће обрадити следеће изузетке који се дешавају над низом:
а) Прекорачење броја елемената низа.
б) Убацивање негативне вредности.
Над низом су дефинисане две операције: убаци и избаци. Користити лифо принцип код убацивања и избацивања елемената низа.

```
import java.io.*;

class Izuzetak extends Exception
{  String poruka;
    Izuzetak(String poruka1) { ...}
  public ...    toString(){return poruka;}
}

class Z12
{ int n[];
  int brojElemenataNiza=0;
  void ubaci(int novi) throws Izuzetak
    {    if (novi < 0)  { ... }
         if (brojElemenataNiza >= n.length)
                  { ... }
         for(int i=brojElemenataNiza; i>0; i--)
               { ... }
          n[0]=novi;
          brojElemenataNiza++;
    }

  int izbaci()
     { int pom;
       pom = n[0];
       for(int i=0; i<brojElemenataNiza-1; i++)  {...}
       ...
       return pom;
     }

  void prikaziElemente()
    { System.out.println("Elementi niza:");
     for(int i=0; i<brojElemenataNiza; i++) {System.out.println("Element:" + n[i]);}
    }

    public static void main(String arg[]) throws IOException
     {
      Z12 z12 = new Z12();
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      System.out.println("Ubaci broj elemenata niza:");
      int brelniza = Integer.parseInt(br.readLine());
      z12.n = new int[brelniza];
      System.out.println("Ubaci elemente niza:");
      while(true)
       { try { int pom = Integer.parseInt(br.readLine());
                  if (pom == 99)  break;
                  z12.ubaci(pom);
             } catch (Izuzetak iz) {System.out.println(iz);}
       }
       z12.izbaci();
       z12.prikaziElemente();
     }
}
```