# Интернет програмирање предавање 15

Проф. др Мирослав Лутовац

mlutovac@viser.edu.rs

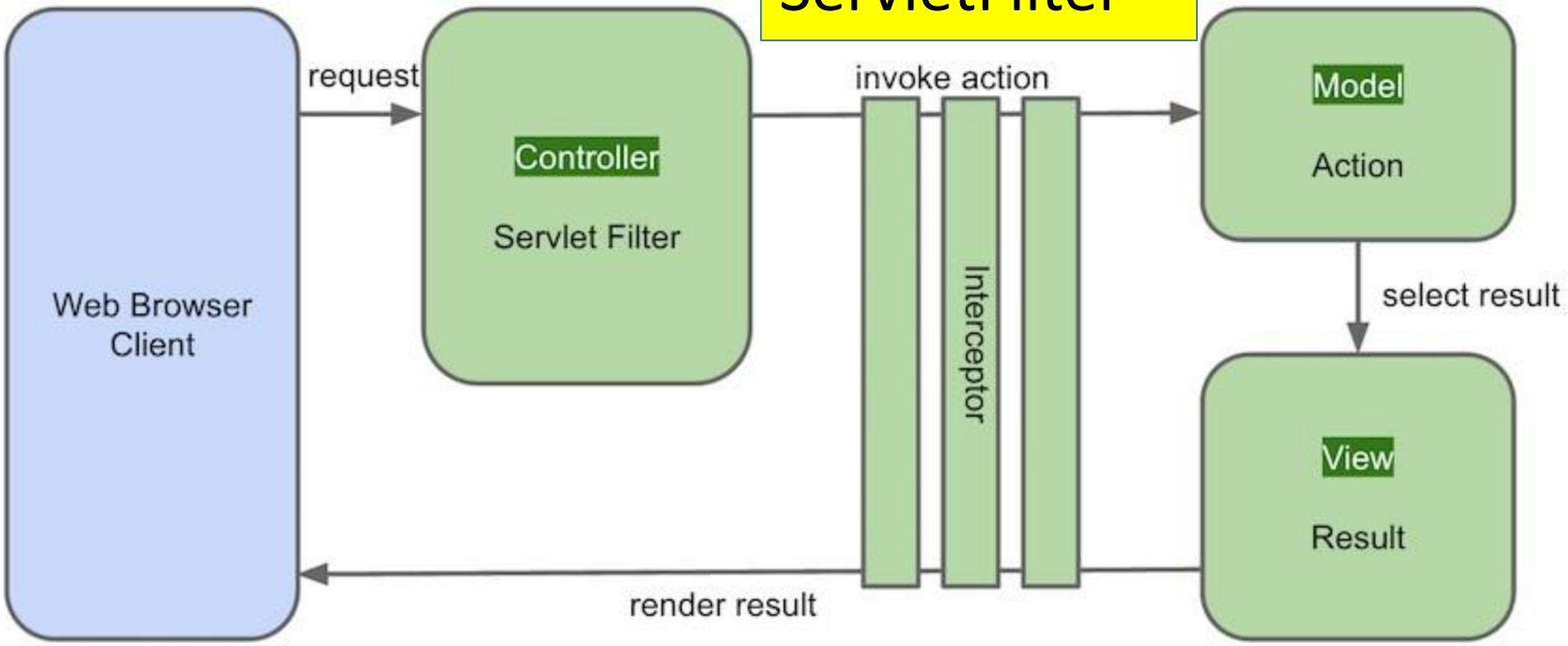
# Literatura

Boško Nikolić, Internet programiranje 1,
VISER Beograd,
ISBN: 978-86-7982-031-0

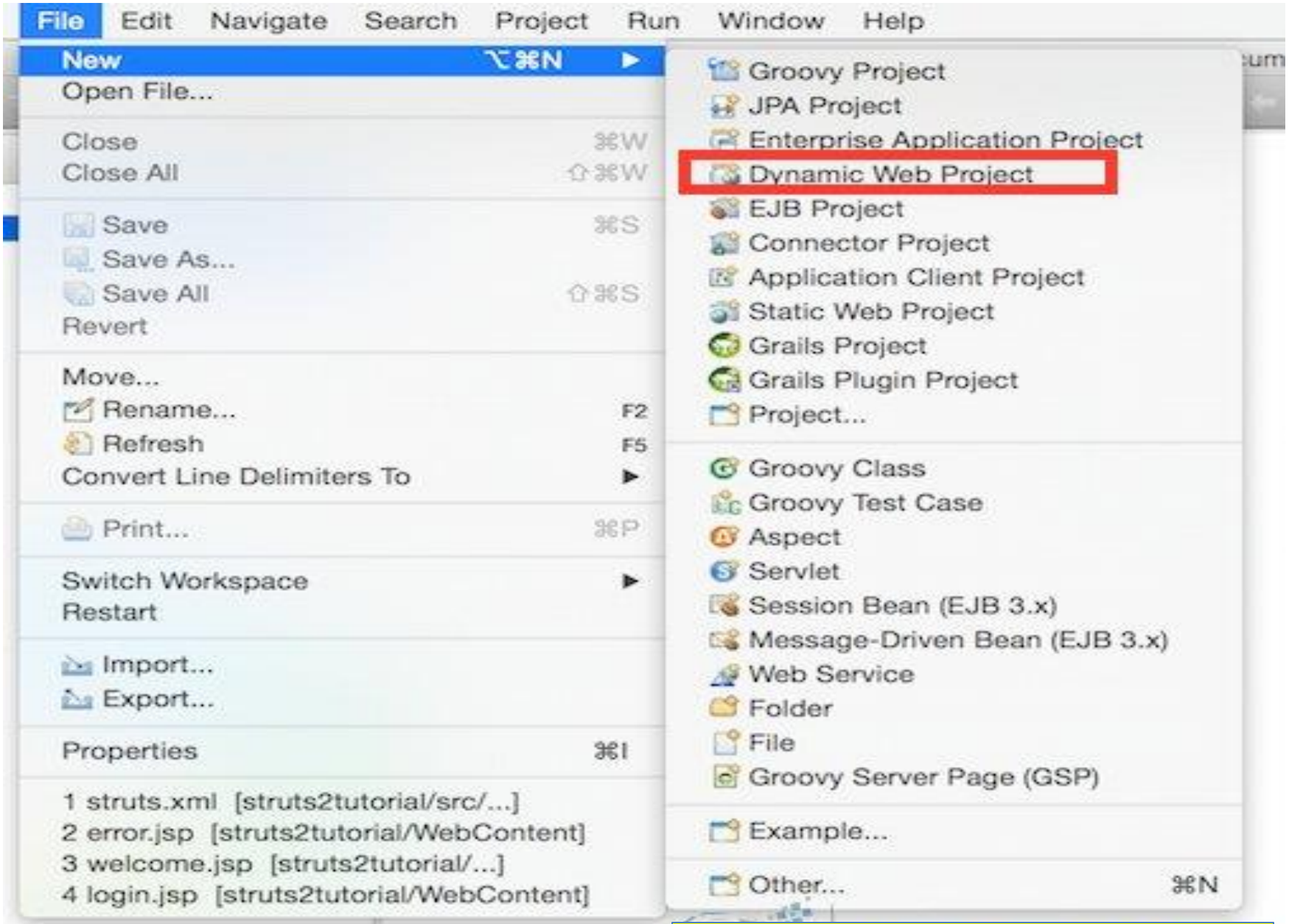
# Java Struts

- Struts je jedno od najpopularnijih okruženja za razvoj Java baziranih aplikacija
- Apache Software Foundation
- Zasnovan na **Model View Controller (MVC)** arhitekturi
- **Model**, interakcija sa standardnim tehnologijama za pristup podacima (JDBC, EJB , Hibernate , iBATIS , Object Relational Bridge
- **View** , radi sa JavaServer Pages , JSTL i JSF , Velocity Templates , XSLT  i drugim sistemima za prezentaciju

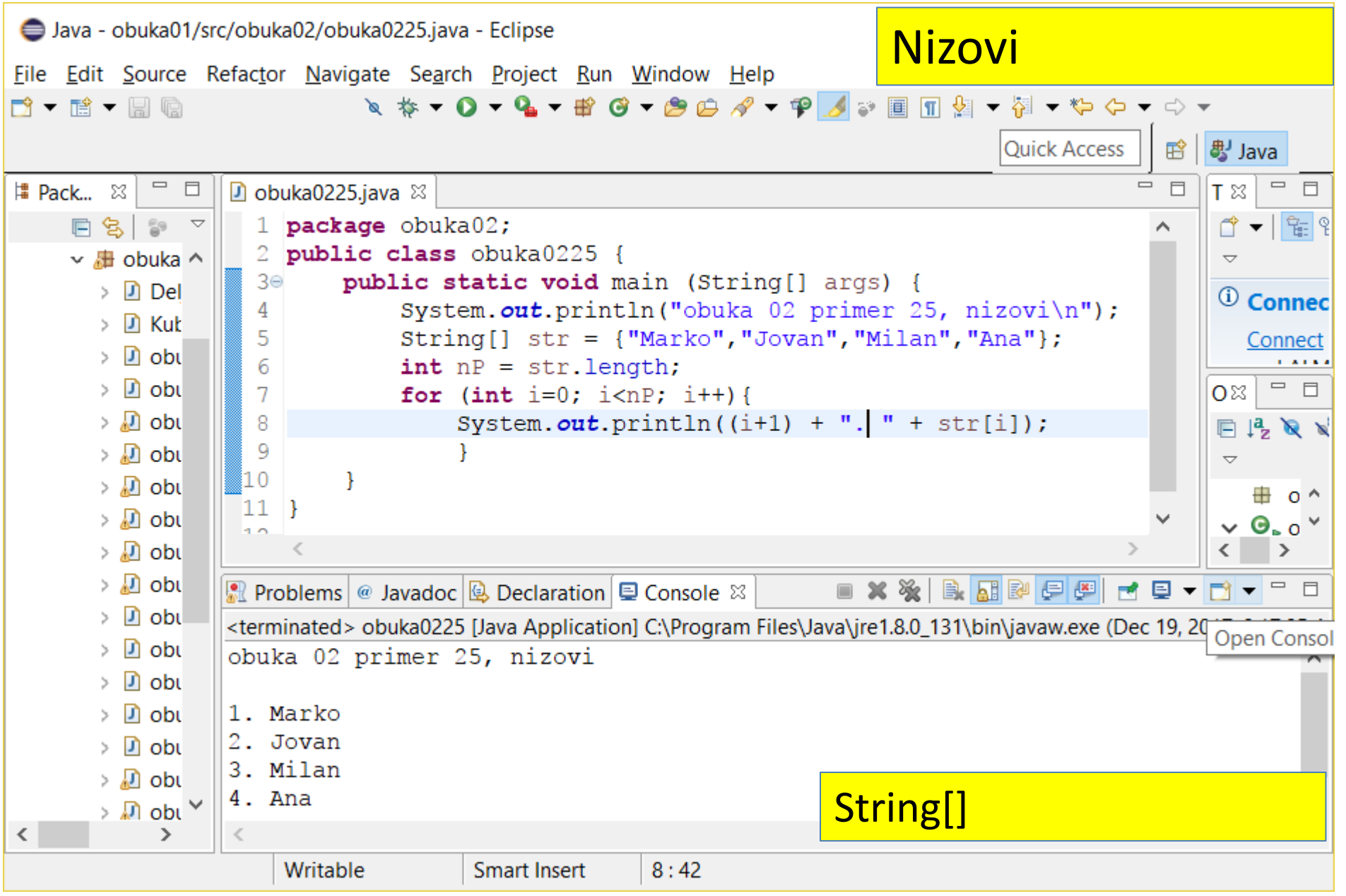
Actions
Results
ServletFilter




- ✓ The Struts Application Framework obezbeđuje set mehanizama koiji omogućava razvoj veb aplikacija
- ✓ Kombinacijom Java klasa, JSP i mapiranjima XML konfiguracionih fajlova, obezbeđuje se modularan pristup koji se lako održavaju kao veb aplikacije




Dinamičke web strane
web application on Tomcat


Java - obuka01/src/obuka02/obuka0225.java - Eclipse
Nizovi
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Pack...
obuka0225.java
1 package obuka02;
2 public class obuka0225 {
3 public static void main (String[] args) {
4 System.out.println("obuka 02 primer 25, nizovi\n");
5 String[] str = {"Marko","Jovan","Milan","Ana"};
6 int nP = str.length;
7 for (int i=0; i<nP; i++){
8 System.out.println((i+1) + ". " + str[i]);
9 }
10 }
11 }
Problems
Javadoc
Declaration
Console
<terminated> obuka0225 [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe
obuka 02 primer 25, nizovi
1. Marko
2. Jovan
3. Milan
4. Ana
String[]
Writable
Smart Insert
8 : 42

Java - obuka01/src/obuka02/obuka0226.java - Eclipse

**tipPodatka[] imeNiza;**

File Edit Source Refactor Navigate Search Project Run Window

obuka0226.java

```
1  package obuka02;
2  public class obuka0226 {
3      public static void main (String[] args) {
4          System.out.println("obuka 02 primer 26, nizovi\n");
5          int[] god = {23,41,33,19};
6          String[] ime = {"Marko","Jovan","Milan","Ana"};
7          int nP = ime.length;
8          for (int i=0; i<nP; i++){
9              System.out.println((i+1) +
10                     ". " + ime[i] +
11                     ", god. " + god[i]);
12             }
13     }
14 }
```

Problems | Javadoc | Declaration | Console

<terminated> obuka0226 [Java Application]

```
obuka 02 primer 26, nizovi

1. Marko, god. 23
2. Jovan, god. 41
3. Milan, god. 33
4. Ana, god. 19
```

**element niza** je istog tipa
int[] god
String[] ime
Indeks 0 do (ime.length-1)

Java - obuka01/src/obuka02/obuka0227.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window

**tipPodatka imeNiza[];**

obuka0227.java

```
1  package obuka02;
2  public class obuka0227 {
3      public static void main (String[] args) {
4          System.out.println("obuka 02 primer 27, nizovi\n");
5          int god[] = {23,41,33,19};
6          String ime[] = {"Marko","Jovan","Milan","Ana"};
7          int nP = ime.length;
8          for (int i=(nP-1); i>=0; i--){
9              System.out.println((nP-i) +
10                     ". " + ime[i] +
11                     ", god. " + god[i]);
12             }
13     }
14 }
```

Problems | Javadoc | Declaration | Console

<terminated> obuka0227 [Java Application]

```
obuka 02 primer 27, nizovi

1. Ana, god. 19
2. Milan, god. 33
3. Jovan, god. 41
4. Marko, god. 23
```

**element niza** je istog tipa
int god[]
String ime[]
indeks (ime.length-1) do 0



Writable | Smart Insert | 9 : 37

**tipPodatka[] imeNiza = new tipPodatka[broj];**

Java - obuka01/src/obuka02/obuka0228.java

File Edit Source Refactor Navigate Search

obuka0228.java

```
1  package obuka02;
2  public class obuka0228 {
3      public static void main (String[] args) {
4          System.out.println("obuka 02 primer 28, nizovi\n");
5          int[] god = new int[4];
6          god[0] = 23; god[1] = 41; god[2] = 33; god[3] = 19;
7          String[] ime = new String[4];
8          ime[0] = "Marko"; ime[1] = "Jovan";
9          ime[2] = "Milan"; ime[3] = "Ana";
10         int nP = ime.length;
11         for (int i=(nP-1); i>=0; i--){
12             System.out.println((nP-i) +
13                     ". " + ime[i] +
14                     ", god. " + god[i]);
15             }
16     }
17 }
```

Problems Javadoc Declaration Console

<terminated> obuka0228 [Java Application] C:\Prog

```
obuka 02 primer 28, nizovi

1. Ana, god. 19
2. Milan, god. 33
3. Jovan, god. 41
4. Marko, god. 23
```

Writable Smart Insert 1 : 17

**Rezervisanje memorije**

int[] imeNiza = new int[4];

String[] imeNiza = new String[4];

# BubbleSort

obuka0229.java

```
1  package obuka02;
2  public class obuka0229 {
3      public static void main (String[] args) {
4          System.out.println("obuka 02 primer 29, nizovi\n");
5          int[] god = new int[4];
6          god[0] = 23; god[1] = 41; god[2] = 33; god[3] = 19;
7          String[] ime = new String[4];
8          ime[0] = "Marko"; ime[1] = "Jovan";
9          ime[2] = "Milan"; ime[3] = "Ana";
10         int nP = ime.length;
11         int tmpGod = god[0];
12         String tmpIme = ime[0];
13         for (int j=1;j<nP;j++){
14             for (int i=1;i<nP;i++){
15                 if (god[i-1]>god[i]){
16                     tmpGod = god[i-1];
17                     god[i-1] = god[i];
18                     god[i] = tmpGod;
19                     tmpIme = ime[i-1];
20                     ime[i-1] = ime[i];
21                     ime[i] = tmpIme;
22                 }
23             }
24         }
25         for (int i=0; i<nP; i++){
26             System.out.println((i+1) +
27                     ". " + ime[i] +
28                     ", god. " + god[i]);
29             }
30     }
31 }
```

Problems | Javadoc | Declaration | Console

<terminated> obuka0229 [Java Application] C:\Program Files\Java\jre1

```
obuka 02 primer 29, nizovi

1. Ana, god. 19
2. Marko, god. 23
3. Milan, god. 33
4. Jovan, god. 41
```

Zamenjuje mesta dva uzastopna člana po rastućem poretku

# 2D niz

obuka0230.java

```java
package obuka02;
public class obuka0230 {
    public static void main (String[] args) {
        System.out.println("obuka 02 primer 30, 2D niz");
        int[][] niz2D = new int[3][];
        int[] b = {23, 41, 47, 19};
        int[] c = {31, 32, 33, 34};
        int[] d = {95, 96, 97, 98};
        niz2D[0] = b;
        niz2D[1] = c;
        niz2D[2] = d;
        int nP1 = niz2D[0].length;
        int nP2 = niz2D.length;
        System.out.println(nP1 + ", " + nP2);
        for (int j=0;j<nP1;j++){
            for (int i=0;i<nP2;i++){
                int tmp = niz2D[0][1];
                System.out.println((i) + " " + (j) +
                        ", xij " + niz2D[i][j]);
            }
        }
    }
}
```

```
obuka 02 primer 30, 2D niz
4, 3
0 0, xij 23
1 0, xij 31
2 0, xij 95
0 1, xij 41
1 1, xij 32
2 1, xij 96
0 2, xij 47
1 2, xij 33
2 2, xij 97
0 3, xij 19
1 3, xij 34
2 3, xij 98
```

# 2D niz

obuka0231.java

```
1  package obuka02;
2  public class obuka0231 {
3      public static void main (String[] args) {
4          System.out.println("obuka 02 primer 31, 2D niz");
5          int[][] niz2D = {
6                  {23, 41, 47, 19},
7                  {31, 32, 33, 34},
8                  {95, 96, 97, 98}};
9          int nP1 = niz2D[0].length;
10         int nP2 = niz2D.length;
11         System.out.println(nP1 + ", " + nP2);
12         for (int j=0;j<nP1;j++){
13             for (int i=0;i<nP2;i++){
14                 int tmp = niz2D[0][1];
15                 System.out.println((i) + " " + (j) +
16                         ", xij " + niz2D[i][j]);
17             }
18         }
19     }
20 }
21
```

Problems | Javadoc | Declaration | Console

<terminated> obuka0231 [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe

```
obuka 02 primer 31, 2D niz
4, 3
0 0, xij 23
1 0, xij 31
2 0, xij 95
0 1, xij 41
1 1, xij 32
2 1, xij 96
0 2, xij 47
1 2, xij 33
2 2, xij 97
0 3, xij 19
1 3, xij 34
2 3, xij 98
```

**Citaj.java**

**Funkcija, procedura, koja se koristi u drugoj klasi**

```
1  package obuka02;
2  //Citaj.java - Citanje podataka standardnih tipova iz ulaznog toka,
3  //             iz datoteke i s glavnog ulaza.
4
5  import  java.io.*;
6
7  public class Citaj {
8
9  private InputStream ut;     // Ulazni tok iz kojeg se cita.
10 private char c;              // Poslednji procitani znak.
11 private boolean eos;         // Indikator kraja toka.
12
13 public Citaj(InputStream uut) { ut = uut; }  // Inicijalizacija
14                                               //   ulaznim tokom.
15 public Citaj(String ime) throws FileNotFoundException  // Otvaranje
16  { ut = new FileInputStream(ime); }                    //   datoteke.
17
18 public boolean endS() { return eos; }         // Da li je kraj toka?
19
20 public char getChS() {      // Dohvatanje sledeceg znaka.
21  try { int i = ut.read(); return c = (eos = i == -1) ? ' ' : (char)i; }
22    catch (Exception g) { eos = true; return c = ' '; }
23 }
```

```
56 public int    IntS     ()  // Citanje jednog podatka tipa int.
57  { String s = StringS (); return !eos ? Integer.parseInt(s) : 0; }
58
```

```
83 public static int     Int     () { return gl.IntS     (); }
```

# Unosenje niza Korišćenjem ...

obuka0232.java | Citaj.java

```
1  package obuka02;
2  public class obuka0232 {
3      public static void main (String[] args) {
4          System.out.println("obuka 02 primer 32, metode");
5          System.out.println("   unesi broj clanova niza");
6          int u = Citaj.Int();
7          int i;
8          int a[] = new int[u];
9          int b= a.length;
10         for (i=0; i<b;i++){
11             System.out.println("   unesi za niz a njegov " + i + ". clan ");
12             a[i] = Citaj.Int();
13         }
14         for (i=0;i<b;i++){
15                 System.out.println("a[" + i + "] = " + a[i]);
16         }
17     }
18 }
```

Problems | Javadoc | Declaration | Console

<terminated> obuka0232 [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe (Dec 24, 2017, 1:13:57 PM)

```
obuka 02 primer 32, metode
   unesi broj clanova niza
3
   unesi za niz a njegov 0. clan
4567
   unesi za niz a njegov 1. clan
89
   unesi za niz a njegov 2. clan
21
a[0] = 4567
a[1] = 89
a[2] = 21
```

# Unosenje niza

Java - obuka01/src/obuka02/obuka0232.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

obuka0232.java | Citaj.java

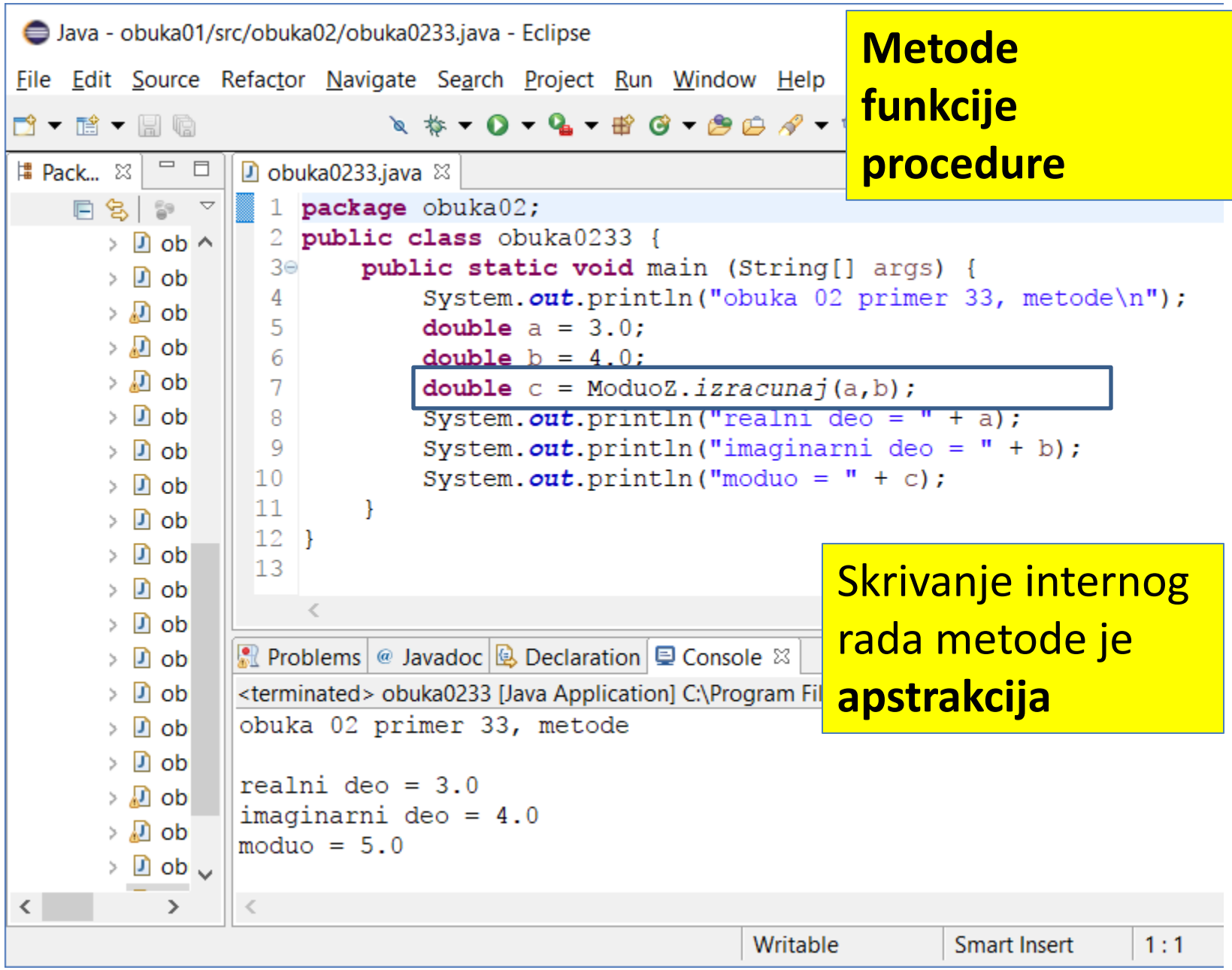
```
1  package obuka02;
2  public class obuka0232 {
3      public static void main (String[] args) {
4          System.out.println("obuka 02 primer 32, metode");
5          System.out.println("   unesi broj clanova niza");
6          int u = Citaj.Int();
7          int i;
8          int a[] = new int[u];
9          int b= a.length;
10         for (i=0; i<b;i++){
11             System.out.println("   unesi za niz a njegov " + i + ". clan ");
12             a[i] = Citaj.Int();
13         }
14         for (i=0;i<b;i++){
15                 System.out.println("a[" + i + "] = " + a[i]);
16         }
17     }
18 }
```

Stop rada

Problems | Javadoc | Declaration | Console

<terminated> obuka0232 [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe (Dec 24, 2017, 1:16:11 PM)

```
obuka 02 primer 32, metode
   unesi broj clanova niza
456
   unesi za niz a njegov 0. clan
11
   unesi za niz a njegov 1. clan
23
   unesi za niz a njegov 2. clan
```




**Metode funkcije procedure**

Skrivanje internog rada metode je **apstrakcija**

# Metode

Java - obuka01/src/obuka02/obuka0234.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

obuka0234.java

```
1  package obuka02;
2  public class obuka0234 {
3      double absValue(double x){
4          if (x<0)
5          return -x;
6          else
7              return x;
8      }
9      public static void main (String[] args) {
10         obuka0234 objekat = new obuka0234();
11         System.out.println("obuka 02 primer 34, metode\n");
12         double broj = -3.0;
13         System.out.println("negativan broj = " + broj);
14         double novi = objekat.absValue(broj);
15         System.out.println("abs broja = " + novi);
16     }
17 }
```

Metoda koja će se koristiti za objekat u klasi

objekat u klasi

objekat u klasi
Koji koristi metodu

Problems | Javadoc | Declaration

<terminated> obuka0234 [Java Application

```
obuka 02 primer 34, metode

negativan broj = -3.0
abs broja = 3.0
```

**tip ime_metode argument_metode**
double absValue (double x)
tip je kao izlaz koji metoda vraća

Writable | Smart Insert | 18 : 1

# Metode

Java - obuka01/src/obuka02/obuka0235.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

obuka0235.java

```
1  package obuka02;
2  public class obuka0235 {
3      double sqrt(double x){
4          double epsilon = 1e-15;
5          double t = x;
6          while (Math.abs(t - x/t) > epsilon
7              t = (x/t + t) / 2.0;
8          }
9          return t;
10     }
11     public static void main (String[] args) {
12         obuka0235 objekat = new obuka0235();
13         System.out.println("obuka 02 primer 35, metode\n");
14         double broj = 2.0;
15         System.out.println("zadati broj = " + broj);
16         double novi = objekat.sqrt(broj);
17         System.out.println("kvadratni koren broja = " + novi);
18         double novi2 = Math.sqrt(broj);
19         System.out.println("koriscenjem Math.sqrt = " + novi2);
20     }
21 }
```

Metoda koja koristiti algoritam za izračunavanje sqrt()

U metodi se koristi **void** ako se ništa ne vraća, već. na primer. prikazuje rezultat na konzoli

Problems Javadoc Declaration

<terminated> obuka0235 [Java Application]

```
obuka 02 primer 35, metode

zadati broj = 2.0
kvadratni koren broja = 1.414213562373095
koriscenjem Math.sqrt = 1.4142135623730951
```



Writable Smart Insert 19 : 50

# Java objekti i klase 1

- **Tipovi podataka**, int, double, boolean, String
- **Promenljive**, pamte vrednosti određenog tipa
- **Nizovi**, pamte više vrednosti istog tipa
- **Kontrolne strukture**, grananje i petlje
- **Metode**, delovi koda kojima mogu da se proslede argumenti i da nešto izračunaju, odrade

# Java objekti i klase 2

- U nekim slučajevima se podaci različitog tipa pamte zajedno **(String[] imena, int[] godine)**
- Metode mogu da vrate samo rezultat jednog tipa
- Objekti su alati za primenu apstrakcije
- Koje detalje treba sakriti, a koje prikazati?

# Java objekti i klase 3

- Stanje, osobine objekta, promenljive koje pamtimo u okviru objekta (field, stste)
- Ponašanje, nešto što objekat može da uradi, metoda koja se definiše u objektu (methods)

# Java objekti i klase 4

- U realnom životu, automobil je objekt
- Automobil ima stanja, osobine (težina, boja), i ponašanje, metode (kreni i stani)
- Svi automobili imaju iste osobine, ali se osobine razlikuju od auta do auta
- Svi automobili imaju iste metode, ali se metode izvršavaju u različito vreme

# Java objekti i klase 5

| **Objekat, Object** | **Stanje, osobina, Properties** | **Ponašanje, Methods** |
|---|---|---|
| auto | ime, model, boja | kreni, vozi |

auto.ime = Fiat

auto.model = 500L

auto.tezina = 850kg

auto.boja = crvena

auto.kreni()

auto.vozi()

auto.koci()

auto.stani()

**Modularnost, sakrivanje informacija, Važan je interfejs**

# Java klase 1

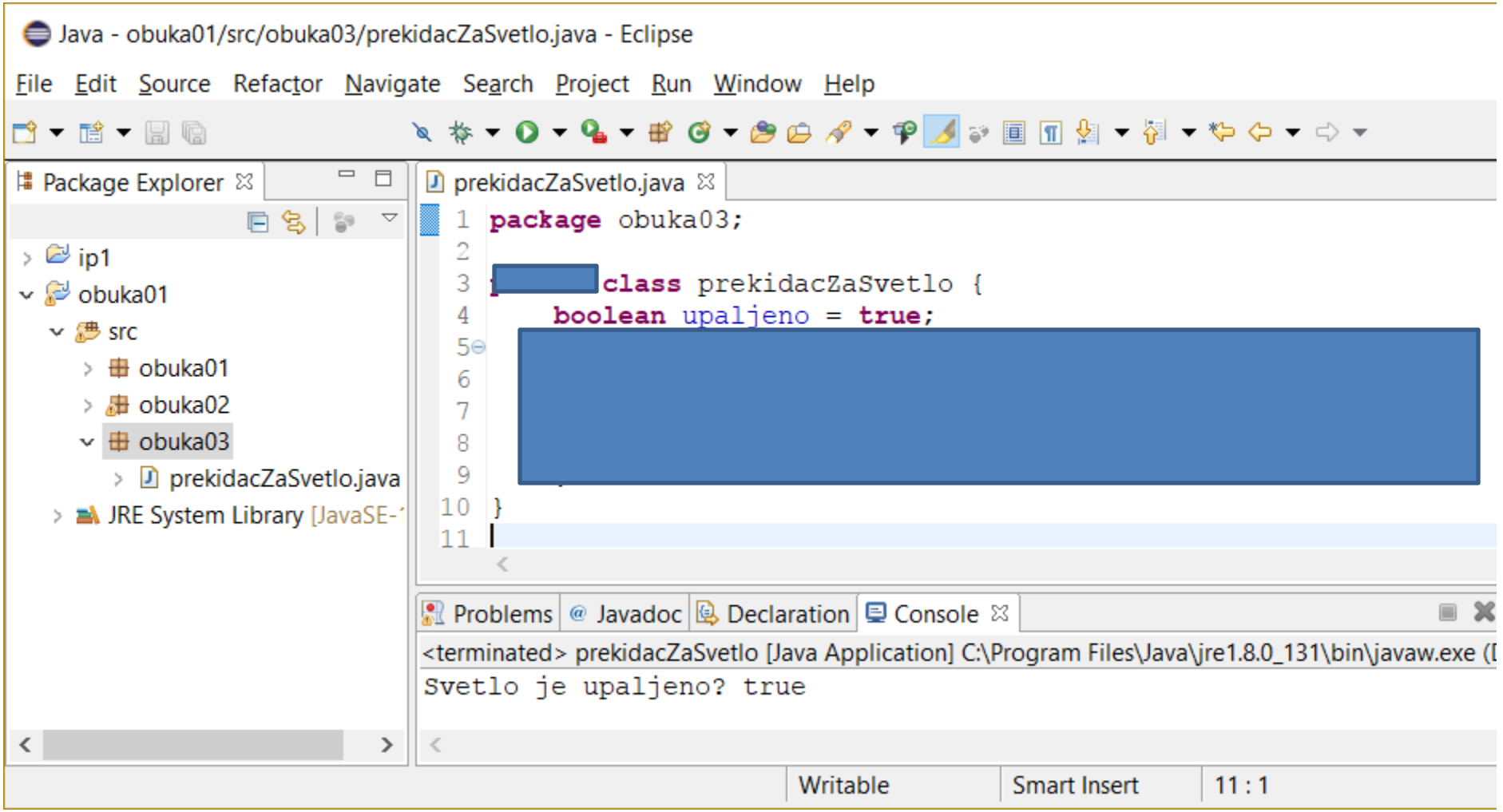
- Klasa je neophodna da bi u okviru nje definisali objekat
- Klasa je uzorak, kalup da bi se napravio objekat
- Klasa sadrži
  - Članove podataka (osobine, karakteristike objekta, klase)
  - Metode (ponašanje objekta klase)
  - Konstruktore (specijalne metode)

# Java klase 2

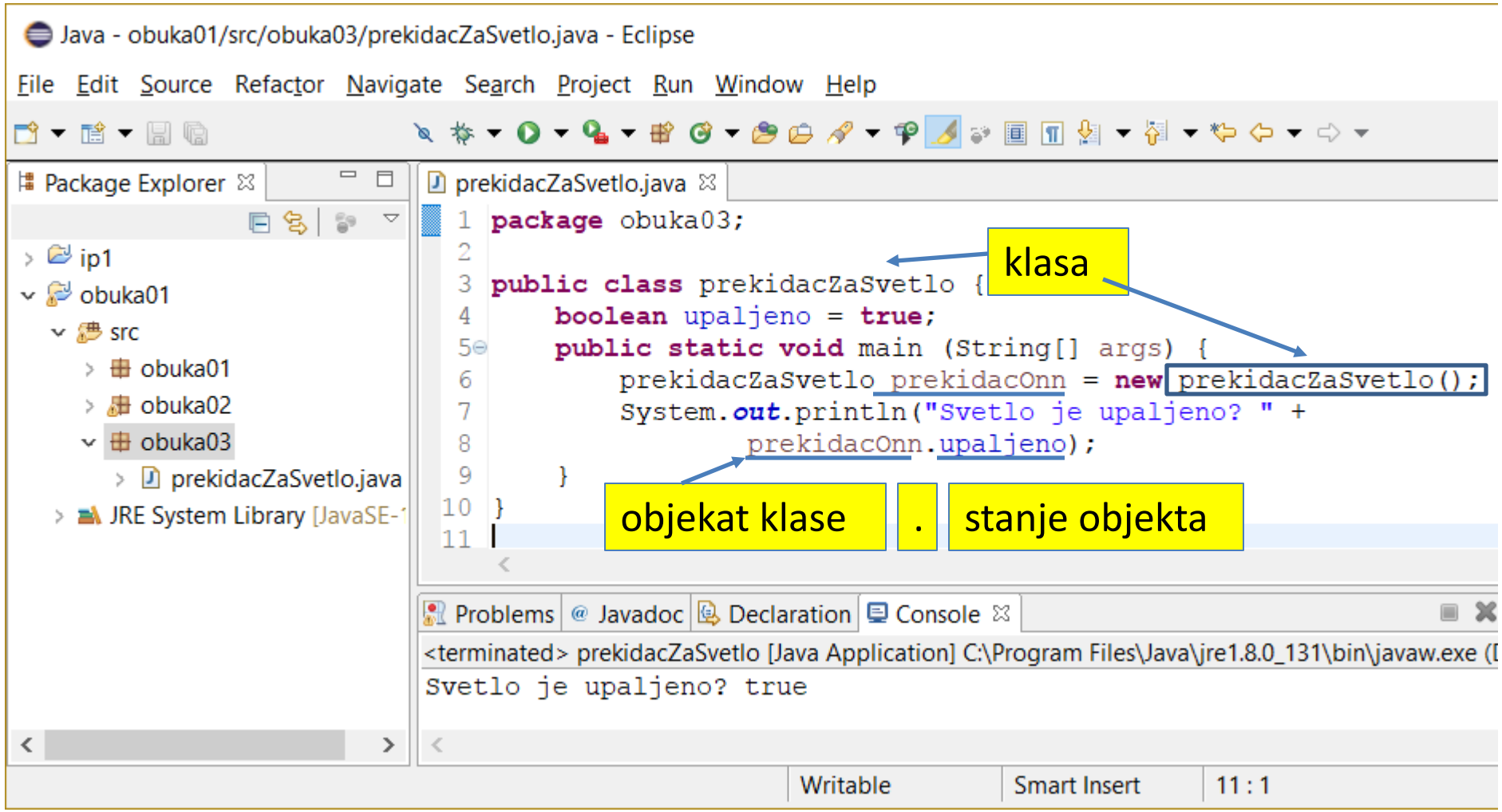
- Anatomija klase
  - Članove podataka (osobine, karakteristike objekta, klase)
  - Metode (ponašanje objekta klase)
  - Konstruktore (specijalne metode)

```
public class imeKlase {
Članovi podataka
Konstruktori
Metode
}
```

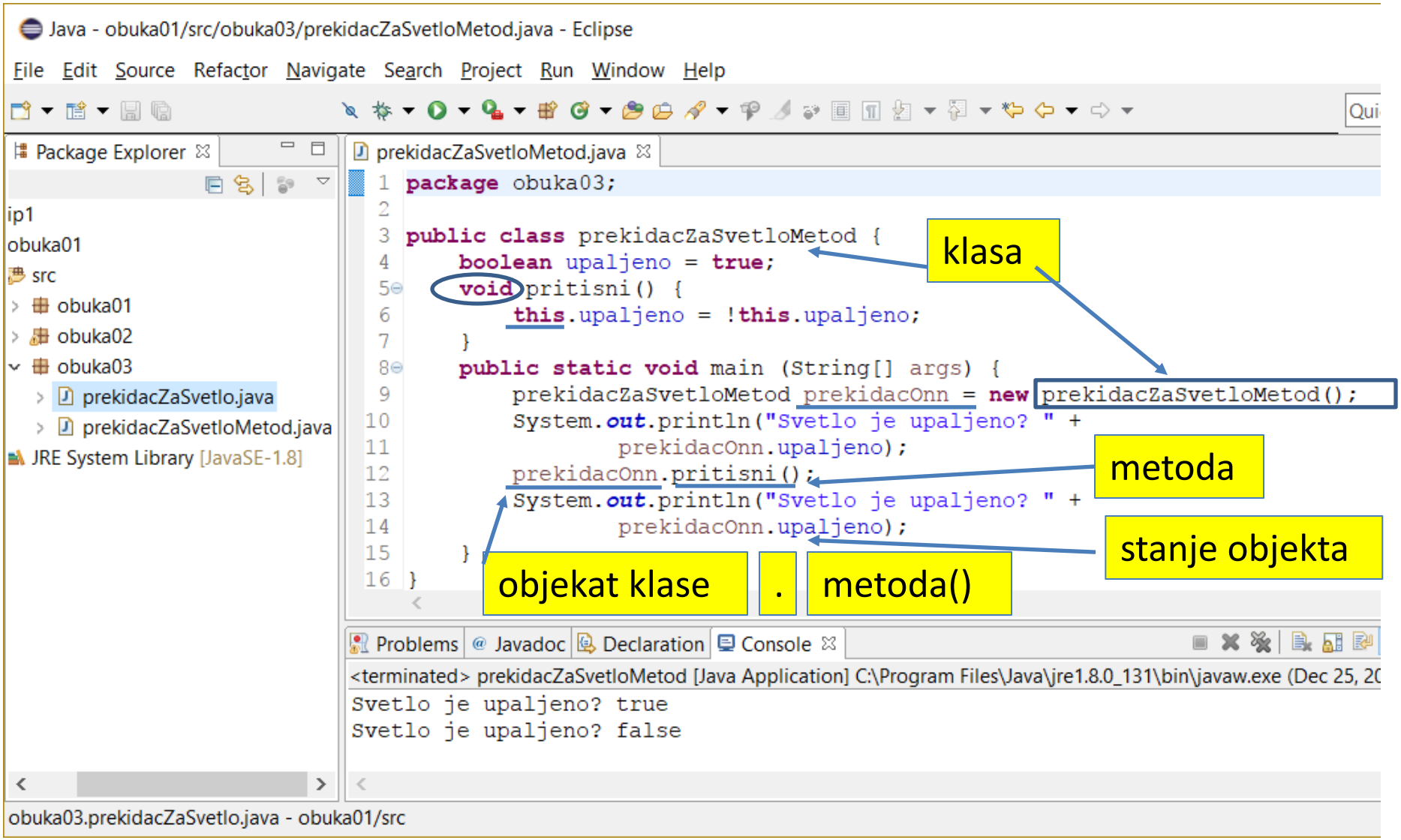
# Primer jednostavne klase



# Primer jednostavne klase, prikaz rezultata, stanje



# Primer jednostavne klase, stanje i metoda



# Primer jednostavne klase, stanje, metoda i konstruktor

Java - obuka01/src/obuka03/prekidacSvetlaKonst.java - Eclipse

File Edit Source Re

prekidacSvetlaKonst.java

```
1  package obuka03;
2
3  public class prekidacSvetlaKonst {
4      boolean upaljeno;
5      void pritisni() {
6          this.upaljeno = !this.upaljeno;
7      }
8      prekidacSvetlaKonst (boolean pocetnoStanje){
9          this.upaljeno = pocetnoStanje;
10     }
11     public static void main (String[] args) {
12         boolean pocetnoStanje=false;
13         prekidacSvetlaKonst prekidacOnn = new prekidacSvetlaKonst(pocetnoStanje);
14         System.out.println("Svetlo je upaljeno? " +
15                 prekidacOnn.upaljeno);
16         prekidacOnn.pritisni();
17         System.out.println("Svetlo je upaljeno? " +
18                 prekidacOnn.upaljeno);
19     }
20 }
21
```

Početno stanje

Objekat klase ima stanje, metodu i početno stanje koje može da postavi stanje na početku, a zatim se stanje menja metodom

Problems | Javadoc | Declaratic

<terminated> prekidacZaSvetloMetod

```
Svetlo je upaljeno? true
Svetlo je upaljeno? false
```

Ako se ne specificira da li je polje ili metod public ili private, podrazumeva se da je public

# Primer jednostavne klase, stanje, metoda, private

Polje upaljeno je definisano kao private
Ako se iz druge klase pristupa ovom polju, desiće se greška

```
1  package obuka03;
2
3  class prekidacSvetlaPriv {
4      private boolean upaljeno = true;
5      public boolean daLiJeUpaljeno(){
6          return upaljeno;
7      }
8      void pritisni() {
9          upaljeno =  !upaljeno;
10     }
11     public static void main (String[] args) {
12         prekidacSvetlaPriv prekidacOnn = new prekidacSvetlaPriv();
13         System.out.println("Svetlo je upaljeno? " +
14                 prekidacOnn.upaljeno);
15         prekidacOnn.pritisni();
16         System.out.println("Svetlo je upaljeno? " +
17                 prekidacOnn.upaljeno);
18     }
19 }
```

```
<terminated> prekidacSvetlaPriv [Java
Svetlo je upaljeno? true
Svetlo je upaljeno? false
```

Ne sme se pristupati private polju druge klase!

**Operator == za objekte,** kao za byte, short, int, long, double, float, boolean, char,

Java - obuka01/src/obuka03/svetlaObjekt.java - Eclipse

File Edit Source Refactor Navigate Search

svetlaObjekt.java

```
2
3  class svetlaObjekt {
4      private boolean upaljeno = true;
5      public boolean daLiJeUpaljeno(){
6          return upaljeno;
7      }
8      void pritisni() {
9          upaljeno =  !upaljeno;
10     }
11     public static void main (String[] args) {
12         svetlaObjekt svetlo1 = new svetlaObjekt();
13         svetlaObjekt svetlo2 = new svetlaObjekt();
14         svetlaObjekt svetlo3 = svetlo1;
15         System.out.println("svetlo1 ==  svetlo2?  " +
16                 (svetlo1 ==  svetlo2));
17         System.out.println("svetlo1 ==  svetlo3?  " +
18                 (svetlo1 ==  svetlo3));
19     }
20 }
```

Problems | Javadoc | Declaration

<terminated> svetlaObjekt [Java Applicat

```
svetlo1 ==  svetlo2?  false
svetlo1 ==  svetlo3?  true
```

obuka03.svetlaObjekt.java - obuka01/src

Da li su dva objekta ista?
Objekti se kreiraju sa ključnom reči **new**
Kada se instanca poveže sa promenljivom, promenljiva sadrži referencu, vezu, sa objektom
Prva dva objekta su različita iako imaju identična stanja; treći objekat je referenca na isti objekat

# Objektno orijentisani jezici

- Svakoj apstrakciji problema odgovara jedna klasa
- **Klasa** predstavlja prototip iz koga se kreira instanca
- **Objekat** predstavlja instancu klase, instanciranjem
- **Apstrakcija** – zapažanje osobina i ponašanje objekta
- **Enkapsulacija** - implementacija koja dovodi do ponašanja
- Enkapsulacija, da sakrije osobine klase koje nisu bitne, da sakrije strukturu klase

# Objektno orijentisani jezici

- Klasa ima
  1. **Interfejs**, spoljašnji izgled klase, sredstvo preko koga se ustanovljava da li klasa ima željeno ponašanje
  2. **Implementacija**, realizacija svih mehanizama koji dovode do željenog ponašanja
- Implementacija u programskim jezicima obuhvata kreiranje klase koje su identifikovane apstrakcijom
- Svaka klasa ima članove klase, podatke i funkcije

# Objektno orijentisani jezici

- Iz klase se pomoću šablona, uzorka, kalupa, proizvode instance – objekti – sa kojima se funkcionalnost složenog sistema preko interfejsa objekata (komunikacija)
- Uloga enkapsulacije da razdvoji javni i privatni deo klase
- **Javni deo** klase čini interfejs ka okolini preko koga se pristupa instancama klase
- **Privatni delovi** čine implementacione detalje i podatke koji za korisnika klase nisu važni pa ne mora ni da zna (vozač auta zna da auto ima pogonski deo, ali ne i kako radi)

# Objektno orijentisani jezici

- **Modularnost**, generisanje modula koji mogu da se kompajliraju odvojeno, ali imaju dobro definisane veze sa drugim modulima
- Za module je važno uočiti logičke celine
- Ugneždavanje, podpaketi, hijerarhija (klasa, objekata)
- **Hijerarhija** se ostvaruje preko nasleđivanja (inheritance)
- Jedna apstrakcija može biti vrsta druge apstrakcije sa dodatim osobinama
- Dodavanje osobenosti – autonomni entiteti

# Objektno orijentisani jezici

- **Tipiziranje**, svaka klasa definiše novi tip
- Instanciranjem klase dolazi se do objekta te klase
- **Vezivanje**
  - **Statičko**, objekti odgovaraju na poruke prema tipovima u vreme kompajliranja
  - **Dinamičko**, objekti odgovaraju na poruke prema svom tipu u vreme izvršavanja programa

# Priroda objekata

- Objekat je entitet koji ima stanja, ponašanja, identitet
- Strukture i ponašanja objekata se definišu preko klasa
- Objakat – opiplivo, vidljivo i **misaono (opaziti)**
- Stanje objekta su osobine i trenutne vrednosti osobina
- Stanje se izražava preko promenljivih definisanih u klasi
- Akcija nad objektom = šalje mu se poruka, menja stanje
- Stanje objekta je kumulativni rezultat ponašanja
- Mtoda objekta – poziv neke funkcije koja je definisana u klasi objekta

# Priroda objekata

- Vrste metoda
  - Modifikatori, menjaju stanje objekta
  - Selektori, pristupaju stanju bez promene stanja
  - Iteratori, pristupaju svim delovima objekta na definisan način
- Konstruktori – kreiranje objekta
- Destruktori – uništavanje objekta i oslobađanje prostora koji je objekat zauzimao

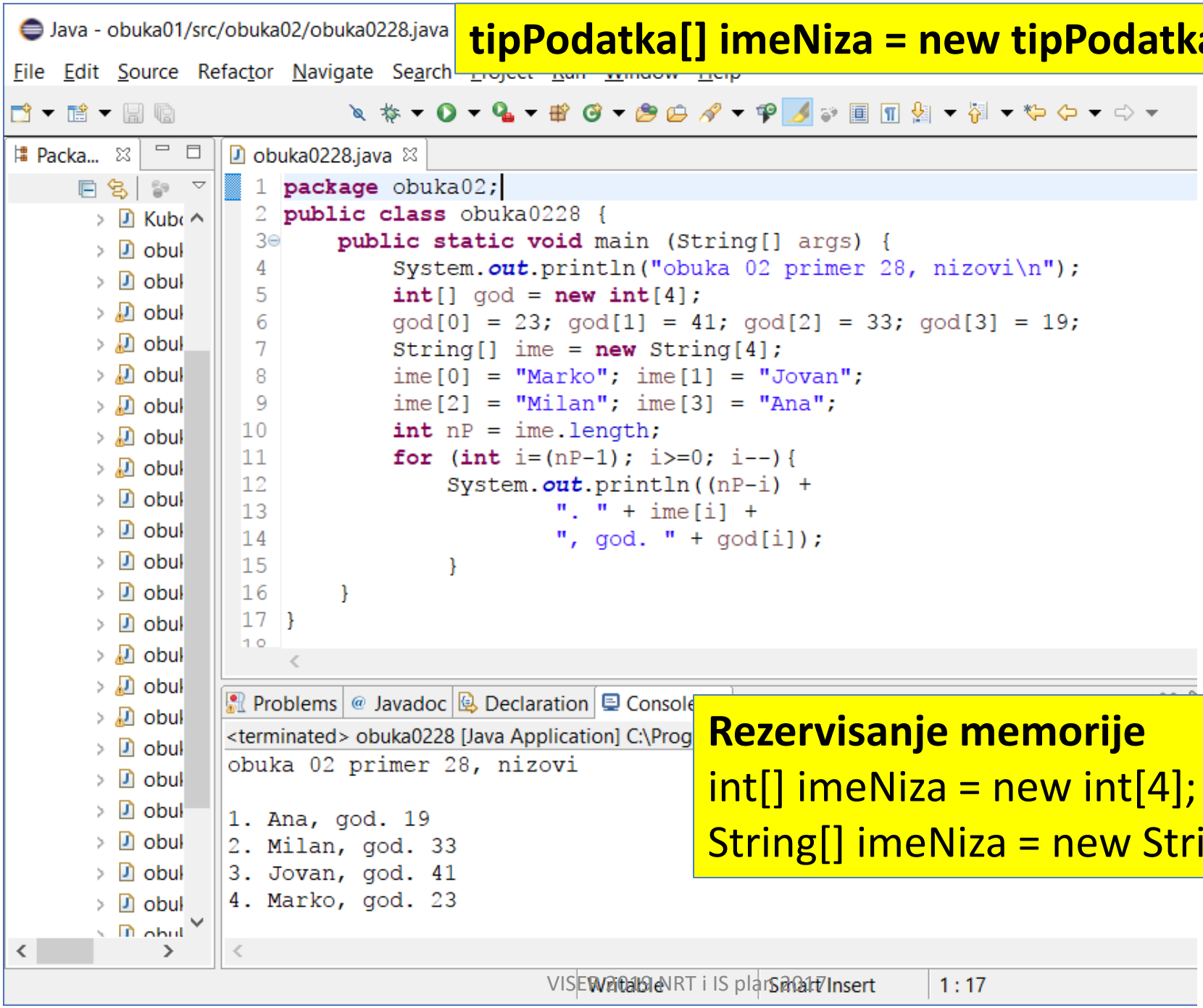
# Veze između objekata

- Komunikacijom između objektima se postiže funkcionalnost
- Linkovi – fizičke i konceptualne veze između objekata
  - Aktivni objekti, operišu nad drugima, ne nad njima, objekat čiji se metod poziva
  - Serveri, ne operišu nad drugima, operiše se na njima, objekat koji se predaje kao argument metode
  - Agenti, operišu nad drugima, i drugi nad njima objekat čiji se metod poziva
- Da objekat bude vidljiv drugom objektu

# Veze između objekata

- Vidljivost objekata
  - Globalan za klijenta
  - Parametar klijentovih operacija
  - Deo klijentskog objekta
  - Lokalno deklarisan u operaciji klijenta
- Agregacij – jedan objekat je deo drugog, utiče na sveukupno stanje
- Agregat, kontejner je objekat koji sadrži druge objekte
- Objekat koji sadrži druge kao atribute

**tipPodatka[] imeNiza = new tipPodatka[broj];**




**Rezervisanje memorije**

int[] imeNiza = new int[4];

String[] imeNiza = new String[4];