Akademija tehničko-vaspitačkih strukovnih studija
odsek NIŠ

Savremene računarske tehnologije **SRT**

OBJEKTNO ORIJENTISANO PROGRAMIRANJE - OOP

Prof. dr Zoran Veličković, dipl. inž. el.

2019/2020.

Prof. dr Zoran Veličković, dipl. inž. el.

OBJEKTNO ORIJENTISANO PROGRAMIRANJE - OOP

Grafičko programiranje u Javi

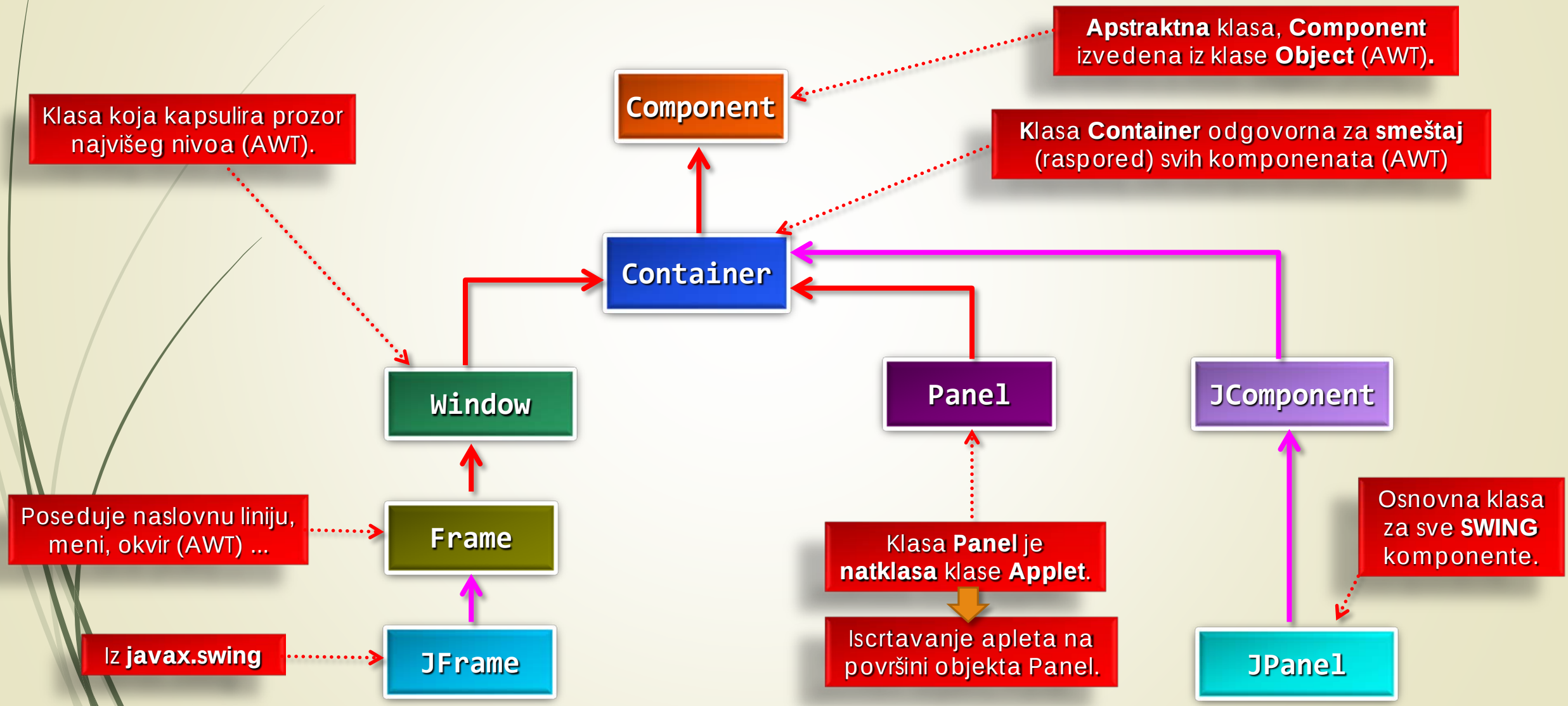
(13)

Sadržaj

Grafički korisnički interfejs - GUI u Javi

- Već je pokazano da se podrška za **GUI** (engl. *Graphic User Interface*) u Javi nalazi u paketima `java.awt.* i javax.swing.*`.
- **`java.awt`** (engl. *Abstract Window Toolkit*) je paket u Javi koji obezbjeđuje **MINIMALNI SKUP KOMPONENTI** grafičkog interfejsa koje postoje za operativni sistem na kome se program izvršava.
- Primena klasa iz ove biblioteke ima za posledicu **RAZLIČITI IZGLED** grafičkog interfejsa na različitim platformama! (ozbiljan problem!)
- **`javax.swing`** je paket koji se nalazi u Javi od verzije 1.2 i sastoji se od KLASA koje su **NEZAVISNE OD PLATFORME** i **OPERATIVNOG SISTEMA** na kojoj se Java izvršava.
- Paket **`javax.swing`** nije jednostavna zamena za **`java.awt`** i često se oba paketa koriste zajedno.
- Za **RAD SA PROZORIMA**, koji su osnovne komponente u grafičkom programiranju, koriste se klase **`Panel`** (za aplete) i **`Frame`** (za standardne prozore kakve poznajete iz Windows-a).

Hijerarhija klasa za prozore



Prozor tipa Frame u apletu (1)

```
// Kreiranje prozora unutar apleta
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

class SampleFrame extends Frame {
  SampleFrame (String title) {
    super(title);
// kreiranje objekta koji upravlja događajima prozora
    MyWindowAdapter adapter = new MyWindowAdapter(this);
// Registrovanje za prijem tih događaja
    addWindowListener(adapter);
  }
  public void paint(Graphics g) {
    g.drawString("This is in frame window", 10, 40);
  }
}
class MyWindowAdapter extends WindowAdapter {
  SampleFrame sampleFrame;
  public MyWindowAdapter(SampleFrame sampleFrame) {
    this.sampleFrame = sampleFrame;
  }
```

```
/*
  <applet code="AppletFrame" width=300 height=50>
  </applet>
*/
```

Konstruktor

Metoda paint

Prozor tipa Frame u apletu (2)

```
public void windowClosing(WindowEvent we) {
    sampleFrame.setVisible(false);
  }
}
// Kreiranje apleta - prozora
public class AppletFrame extends Applet {
  Frame f;
  public void init() {
    f = new SampleFrame("A Frame Window");
        f.setSize(250, 250);
       f.setVisible(true);
  }
  public void start() {
    f.setVisible(true);
  }
  public void stop() {
    f.setVisible(false);
  }
  public void paint(Graphics g) {
    g.drawString("This is in applet window", 10, 20);
  }
}
```

`Jframe` klasa za funkcionalni prozor

- Za formiranje prozora u Javi gotovo se **NIKAD NE KORISTE** objekti klase **`Window`** jer **NE POSEDUJU** liniju naslova ni granične linije!
- Klasa **`Jframe`** obezbeđuje sve potrebne komponente za formiranje i upravljanje prozorima i može sadržavati DRUGE KOMPONENTE.
- Pored ostalog, klase **`Window`** i **`Container`** omogućavaju RUKOVANJE DOGAĐAJIMA koji nastaju interakcijom korisnika sa prozorom.
- Prozor Java aplikacije se formira kroz TRI KORAKA:
  1. Kreiranje objekta tipa **`JFrame`**,
  2. Poziv metode objekta za podešavanje DIMENZIJA PROZORA,
  3. Poziv metode za PRIKAZ prozora.
- Na ovaj način se dobija potpuno operativan **PROZOR APIKACIJE** (ovako kreiran prozor se može minimizirati, zatvoriti - x ili mu se može izmeniti veličina).
- Kroz nekoliko primera biće DODATA POTPUNA FUNKCIONALNOST ovako formiranog prozora dodavanjem dugmadi, menija, ...

JFrame prozor u Javi

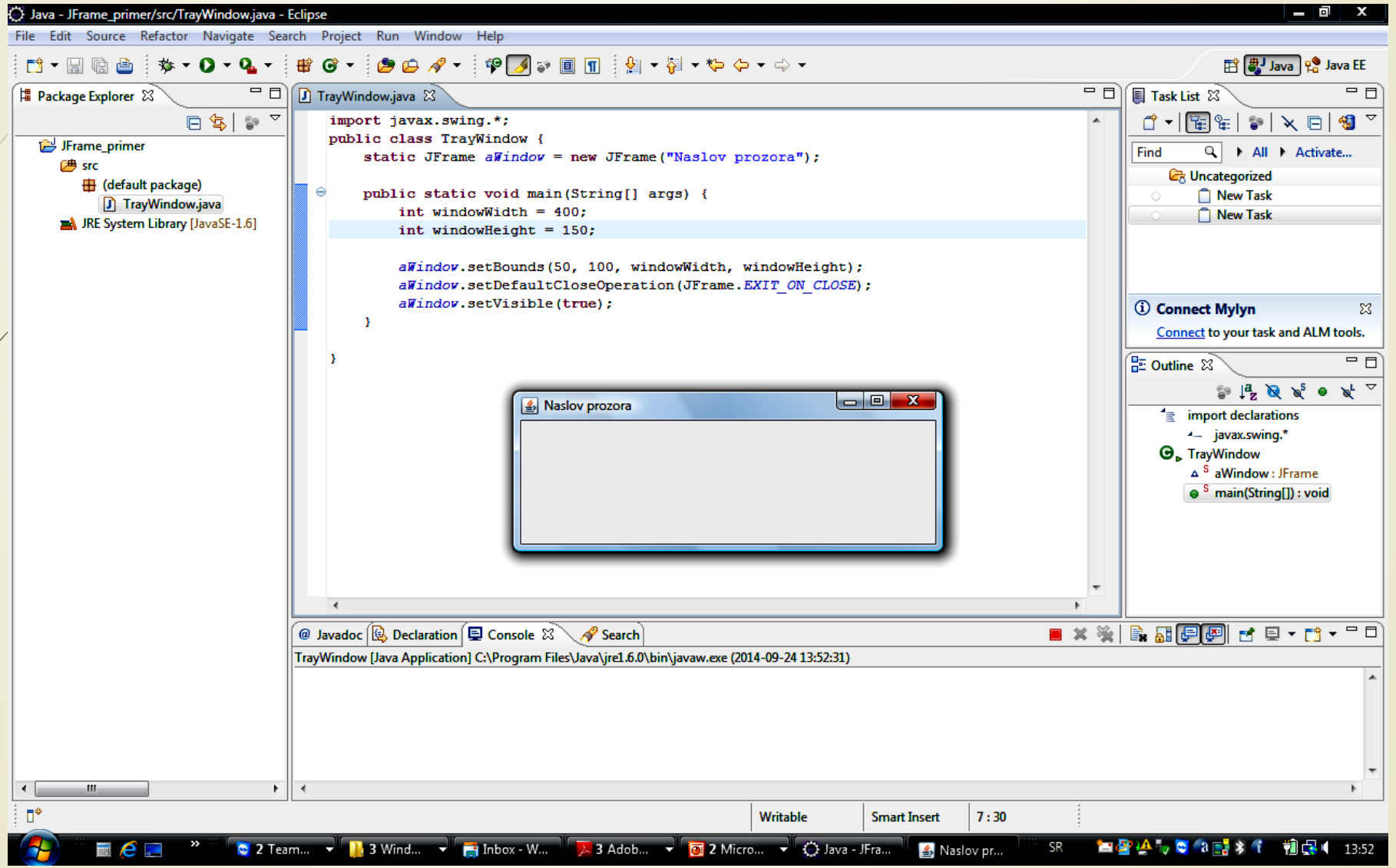
```
import javax.swing.JFrame;
public class TryWindow {
// object tipa prozor
static JFrame aWindow = new JFrame("Naslov prozora");

public static void main(String[] args) {
   int windowWidth = 400;                    // Širina prozora u pikselima, param.
   int windowHeight = 150;                   // Visina prozora u pikselima , param.
   aWindow.setBounds(50, 100, windowWidth, windowHeight);
                                             // Podešavanje pozicije i veličine prozora
   aWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);     //zatv. Pr.
   aWindow.setVisible(true);                 // Učini vidljivim formirani prozor
   }
}
```

1

2

3

JFrame prozor u Eklipsu



Dimenzije ekrana

- Iz već prikazane hijerarhije klasa za kreiranje prozora može se zaključiti da **JFrame** nasljeđuje brojne metode od klasa **Window** i **Container**.
- Metode za pozicioniranje prozora na ekranu su **setSize()** i **setLocation()** i zahtevaju poznavanje **REZOLUCIJE** ekrana koja se dobija putem metoda **getDefaultToolkit()** iz klase **java.awt.Toolkit**.
- U paketu java.awt takođe postoji klasa **Dimesion** koja služi za reprezentaciju **ŠIRINE** i **VISINE** ekrana.
- Programski kod koji koristi pomenute klase za ovu operaciju je sledeći:

```
Toolkit kit = Toolkit.getDefaultToolkit();        // Statička metoda

Dimension screen = kit.getScreenSize();

int height = screen.height;                       // Javne promjenljive članice

int width = screen.width;
```

Komponente i kontejneri (1)

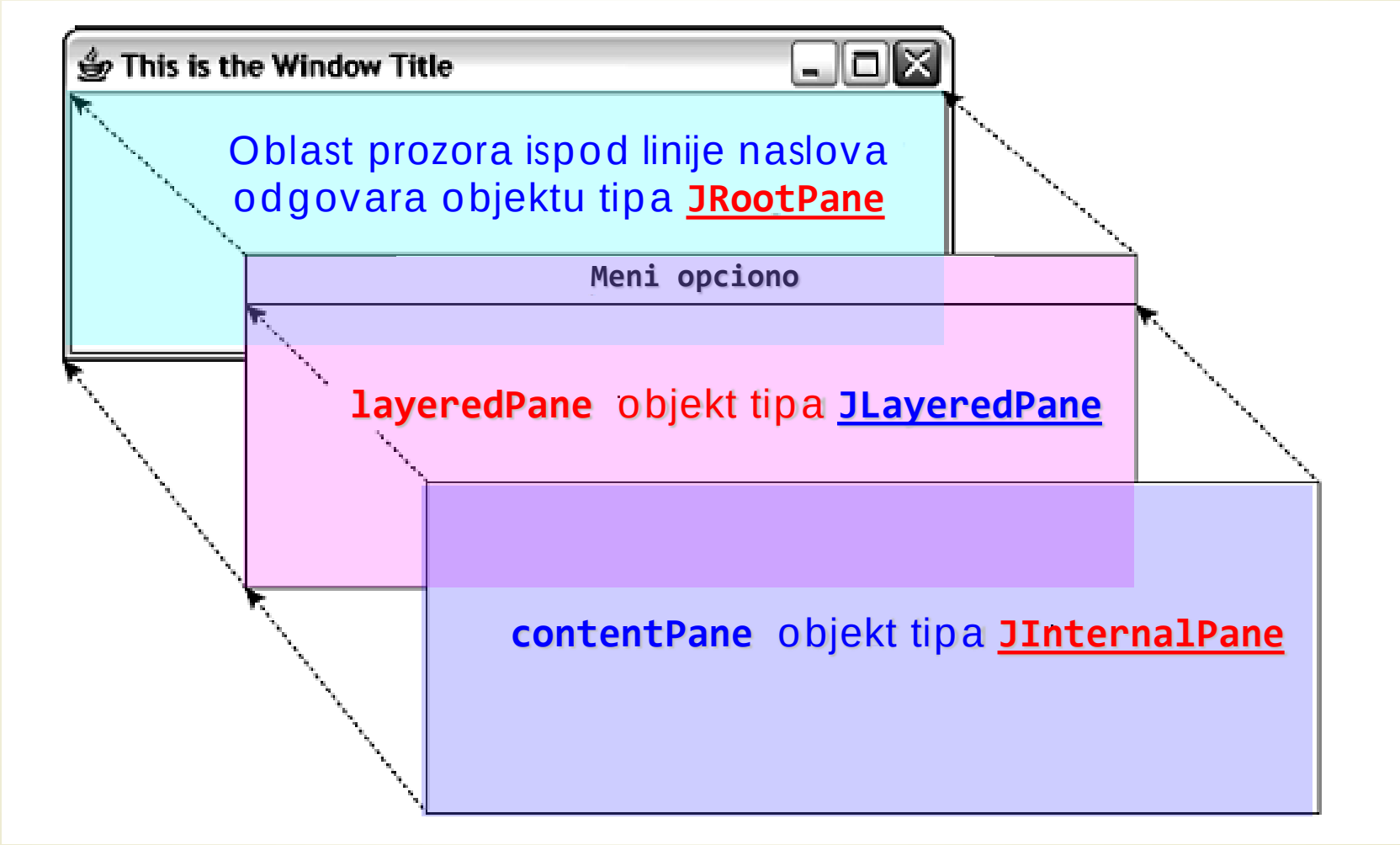
- U Javi se pod **KOMPONENTOM** smatra grafički entitet koji se može prikazati u prozoru, a dobijen je iz klase **`Component`** (na osnovu ove definicije i **`JFrame`** je takođe komponenta).
- Neke od podklasa klase **`Component`** su (pogledaj prethodno datu hijerarhiju klasa):
  - **`JFrame`**, koristi se za osnovni Java prozor aplikacije;
  - **`JWindow`**, prozor bez naslovne linije i ikonice za upravljanje prozorom;
  - **`JDialog`**, definiše prozor dijaloga za unos podataka u program;
  - **`JApplet`**, osnovna klasa za java 2 aplete;
  - **`JComponent`**, definiše niz standardnih komponeti kao što su meniji, dugmadi, ...
- Sve klase izvedene iz **`Container`** klase mogu takođe da sadže druge objekte ove klase i nazivaju se **KONTEJNERI** (kontejneri mogu sadržavati druge kontejnere osim klase **`Window`**).

Komponente i kontejneri (2)

- Kada se dodaju GUI komponente **`JFrame`** objektu, zapravo se dodaju komponente **PROZORSKOM OKNU** kojim upravlja **`JFrame`** objekt.
- **PROZORSKA OKNA** su KONTEJNERI koji predstavljaju **POVRŠINU PROZORA** (ima ih nekoliko tipova), a najčešće se koristi okno sadržaja.
- Objekt **`layeredPane`** (engl. *pane* – srb. *umetak*) obezbeđuje upravljanje **GRUPOM KOMPONENTI** u odvojenim slojevima koji se PREKLAPAJU UNUTAR OKNA (slojevi se prikazuju od nazad ka napred, tako da će komponente u prednjem sloju biti prikazane ispred onih u slojevima koji su pozadi).
- Postoji i objekt **`glassPane`** koji pokriva **`JRootPane`** oblast i prikazuje se PREKO SVIH DRUGIH OBJEKATA (koristi se za potrebe prikaza stalno prisutnih komponenti – primer meni).
- Metode koje pružaju reference na pojedina okna su: **`getRootPane()`, `getLayeredPane()`, `getContentPane()`** i **`getGlassPane()`**.

Prozor i okna aplikacije

Prozor tipa **Jframe**



Veličina i pozicija komponenti

- Selektovane metode koje poseduje klasa **Component** za zadavanje POZICIJE i RASPON VARIJACIJA komponenti su prikazane u tabeli.

METOD	ZNAČENJE
`void setBounds(int x, int y, int width, int height)`	Postavlja poziciju objekta na željene koordinate i širinu i visinu objekta na zahtevane vrednosti.
`void setBounds(Rectangle rect)`	Postavlja poziciju i veličinu objekta na vrednost **rect** objekta **Rectangle**.
`void setSize(Dimension d)`	Postavlja širinu i visinu objekta na vrednosti postavljene u članovima objekta **d**.
`setLocation(int x, int y)`	Postavlja poziciju komponente na tačku određenu vrednostima **(x,y)**.
`setLocation(Point p)`	Postavlja poziciju komponente na tačku **p**.
`void setMinimumSize(Dimension d)`	Postavlja minimalnu veličinu objekta određene sa **d**.
`void setPreferredSize(Dimension d)`	Postavlja maksimalnu veličinu objekta određene sa **d**.
`void setMaximumSize(Dimension d)`	Postavlja željenu veličinu objekta određene sa **d**.

Vizuelne karakteristike komponenti

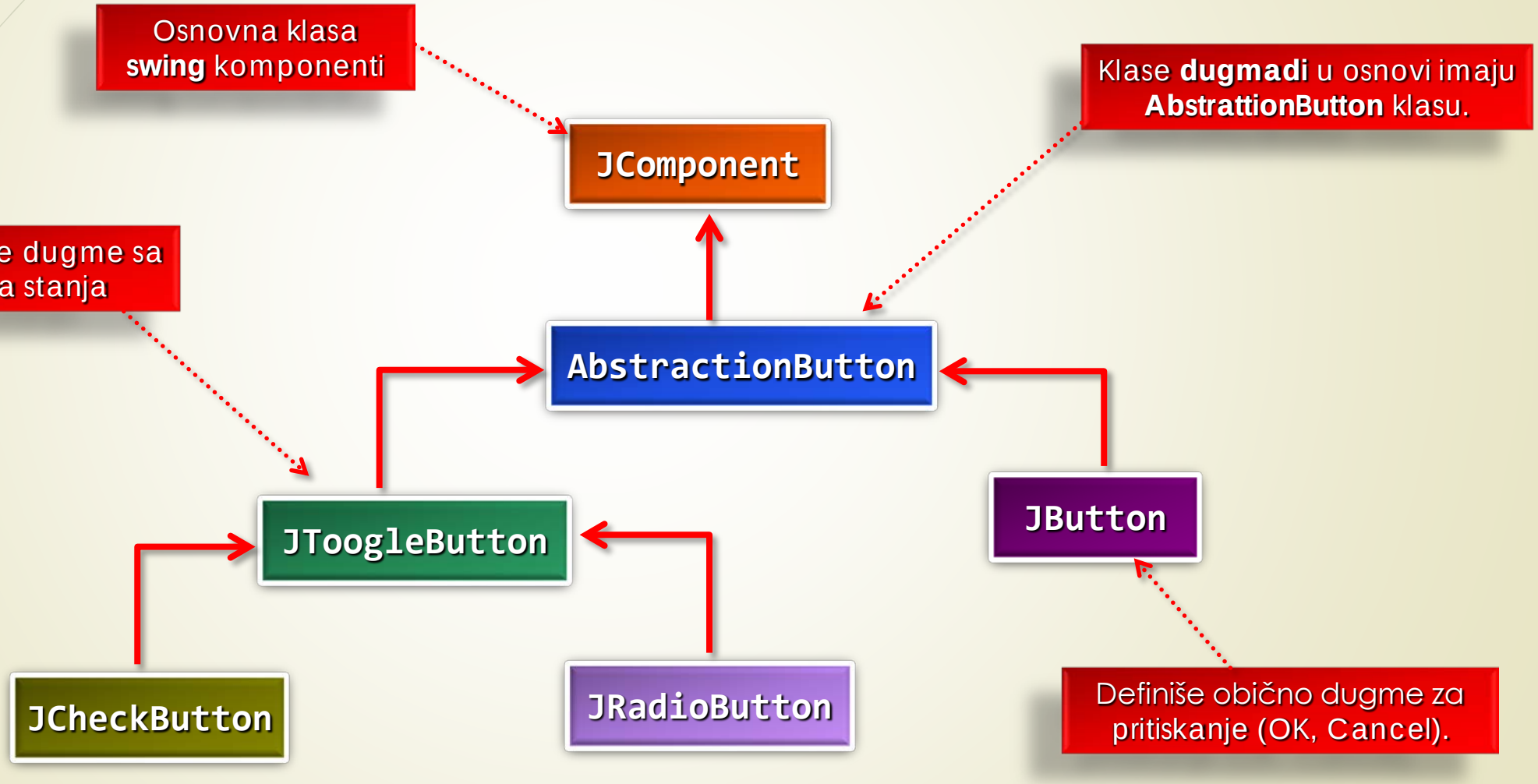
- Selektovane metode koje poseduje klasa **Component** za zadavanje VIZUELNIH KARAKTERISTIKA komponentama su prikazane u tabeli.

METOD	ZNAČENJE
`void setBackground(Color aColor)`	Postavlja boju pozadine na **aColor**.
`Color getBackground()`	Vraća aktuelnu boju pozadine.
`void setForeground(Color bColor)`	Postavlja boju prednje strane na ..
`Color getForeground()`	Vraća aktuelnu boju prednje strane .
`void setCursor(Cursor aCursor)`	Postavlja kursor komponente na **aCursor**.
`void setFont(Font aFont)`	Postavlja font za objekt .
`Font getFont()`	Vraća ojekt Font koji koristi komponenta.

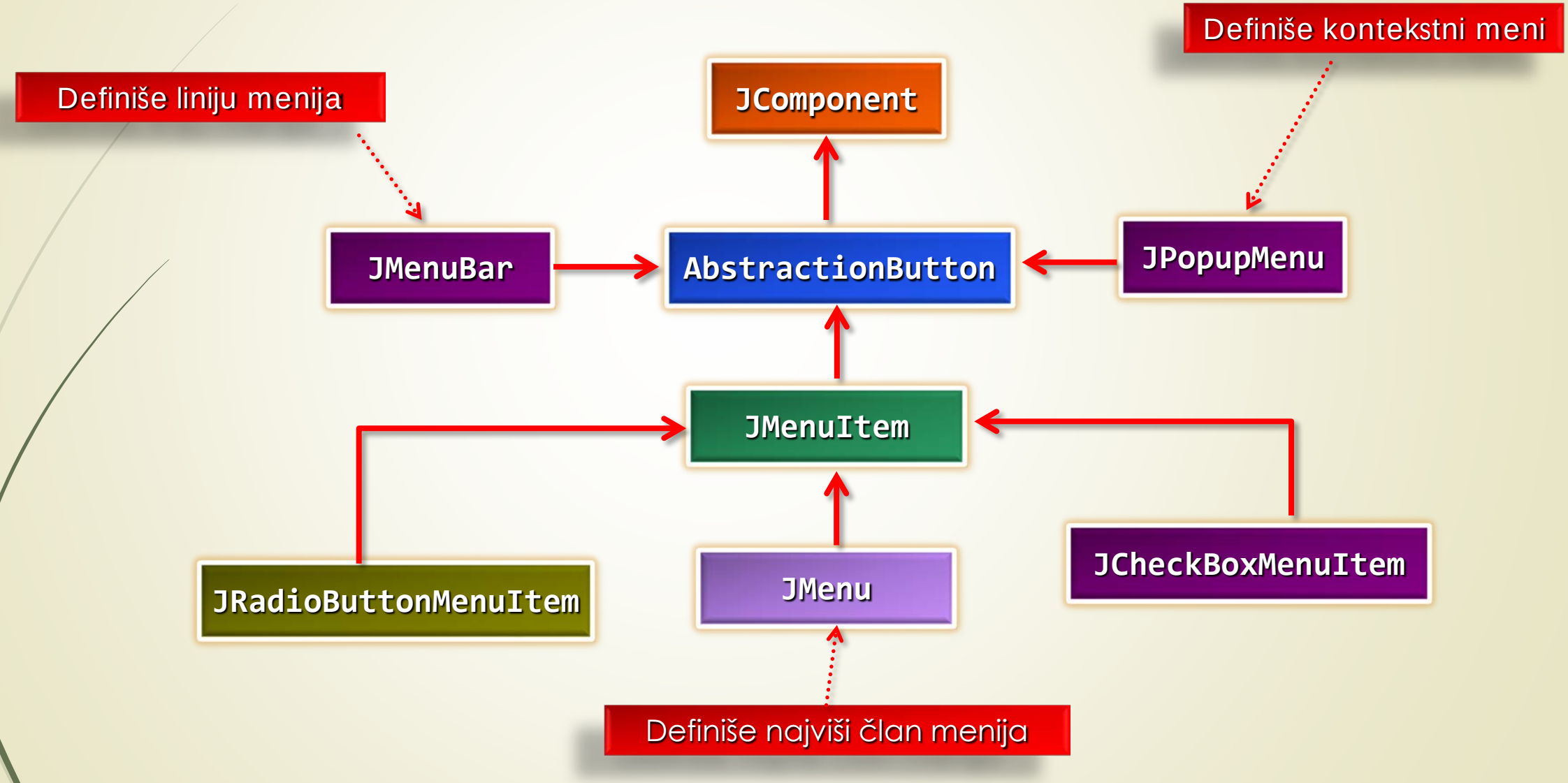
SWING komponente

- Sve klase **SWING** komponenti definisane su u paketu **jawax.swing**.
- Sve **SWING** komponente imaju **JComponent** klasu za osnovu koja je i sama klasa naslednica klase **Component**.
- **SWING KOMPONENTE** podržavaju PROŠIRIV UTISAK I IZGLED (primer: `Metal`, `Motif`, ...)
- **SWING KOMPONENTE** podržavaju SAVETE o alatima (poruke koje opisuju svrhu komponente kada se kursor miša zadrži na njoj).
- **SWING KOMPONENTE** podržavaju automatsko POMERANJE SADRŽAJA u listama, tabelama i stablima.
- **SWING KOMPONENTE** imaju podrška za otklanjanje grafičkih GREŠAKA.
- **SWING KOMPONENTE** se lako proširuju prilikom formiranja sopstvenih komponenti.
- Sva imena klasa iz paketa **javax.swing** počinju velikim slovom **J**.
- Hijerarhija klasa **DUGMADI** u **swingu** je prikazana na sledećoj slici.

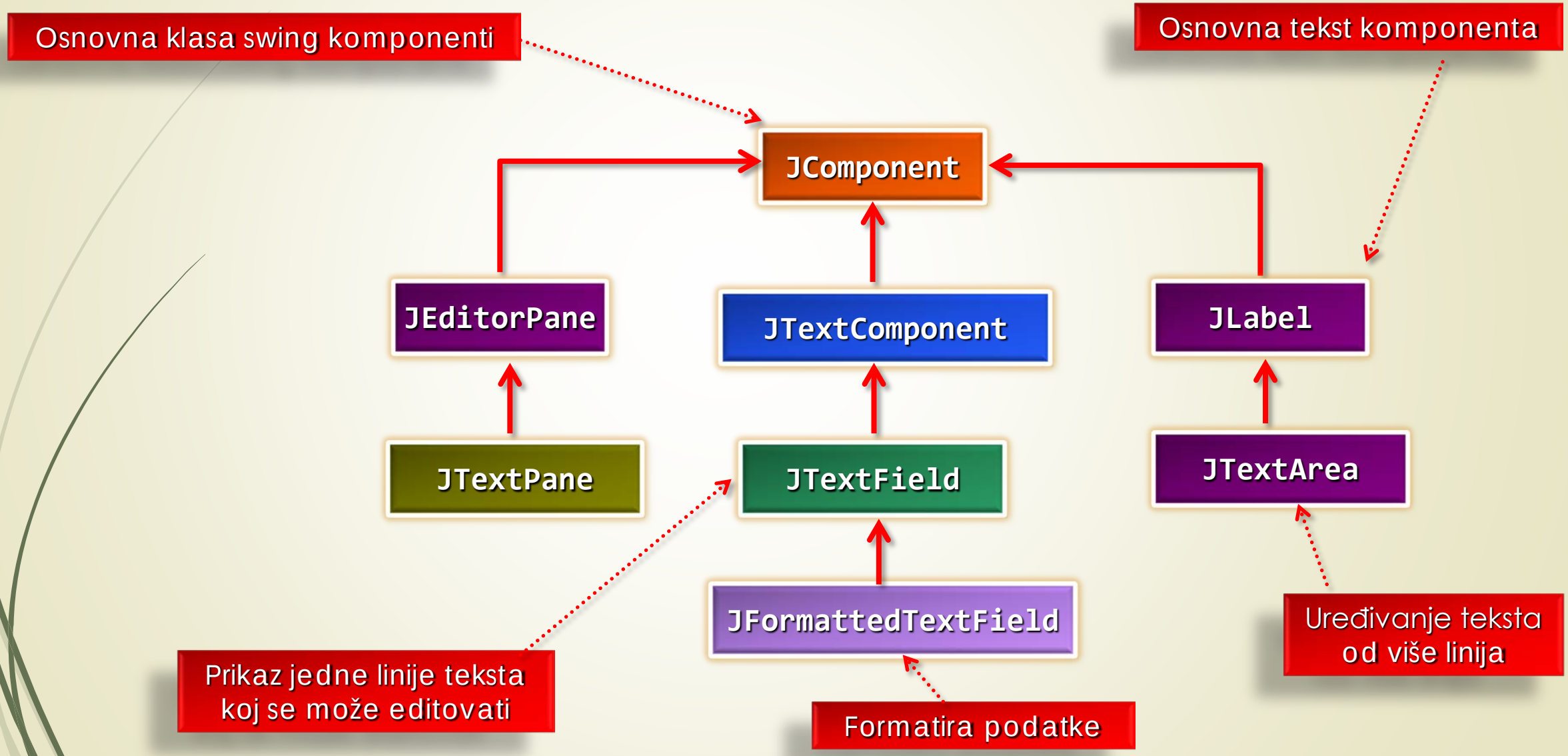
Komponenta: dugme swing



Komponenta: meni swing



Komponenta: swing tekst



Upotreba kontejnera

- **KONTEJNER** je komponenta koja može sadržati DRUGE KOMPONENTE (sve swing komponente su kontejneri).
- Komponente u kontejneru se prikazuju UNUTAR OBLASTI koja je na ekranu zauzeta za kontejner.
- Kontejner kontroliše **RASPORED** svojih komponenti **MENADŽEROM RASPOREDA** (eng. Layout manager).
- Komponente smeštene u kontejner zapisuju se u **NIZ** unutar objekta **`Container`**, koji se **ADAPTIRA** da prihvati željeni broj komponenti.
- Da bi se dodala komponenta u kontejner koristi se preklopljene metode **`add()`**:
  - **`Component add(Component c),`**
  - **`Component add(Component c, int index),`**
  - **`void add(Component c, Object constraints),`**
  - **`void add(Component c, Object constraints, int index).`**

Menadžeri rasporeda (1)

- Način na koji su RASPOREĐENE KOMPONENTE u kontejneru određuje objekat **MENADŽER RASPOREDA**.
- Sve klase koje definišu menadžer rasporeda IMPLEMENTIRAJU INTERFEJS `LayoutManager`.
- Postoje **ŠEST KLASA** menadžera rasporeda: `FlowLayout, BorderLayout, CardLayout, GridLayout, GridBagLayout, BoxLayout i SpringLayout`.
- Opis ovih menadžera raporeda je dat u sledećoj tabeli, a oni se nalaze u paketima `javax.swing` i `java.awt`.
- Iako bi se komponente mogle postaviti na određeno mesto u kontejneru, to se **NE RADI** iz praktičnih razloga jer treba osigurati ISTOVETAN PRIKAZ u svakom mogućem Java okruženju
- Menadžeri rasporeda **AUTOMATSKI** podešavaju komponente tako da se mogu **UKLOPITI** u raspoloživ prostor.
- Metod `setLayout()` služi za podešavanje MENADŽERA RASPOREDA.

Menadžeri rasporeda (2)

MENADŽER RASPOREDA	ZNAČENJE
FlowLayout	Postavlja komponente u sukcesivne vrste kontejnera, uklapajući najveći mogući broj komponenti u svaku vrstu.
BorderLayout	Postavlja komponente uz bilo koju od četri granice kontejnera.
CardLayout	Postavlja komponente jedne preko druge u kontejner.
GridLayout	Postavlja komponente u pravougaonu mrežu sa brojem željenim brojem vrsta i kolona.
GridBagLayout	Postavlja komponente u vrste i kolone različitih dužina.
BoxLayout	Postavlja komponente u vrste i kolone sa odsecanjem.
SpringLayout	Omogućava komponentama da imaju poziciju definisanu oprugama (engl. springs) ili podupiračima učvršćenim za ivicu kontejnera ili neku drugu komponentu.

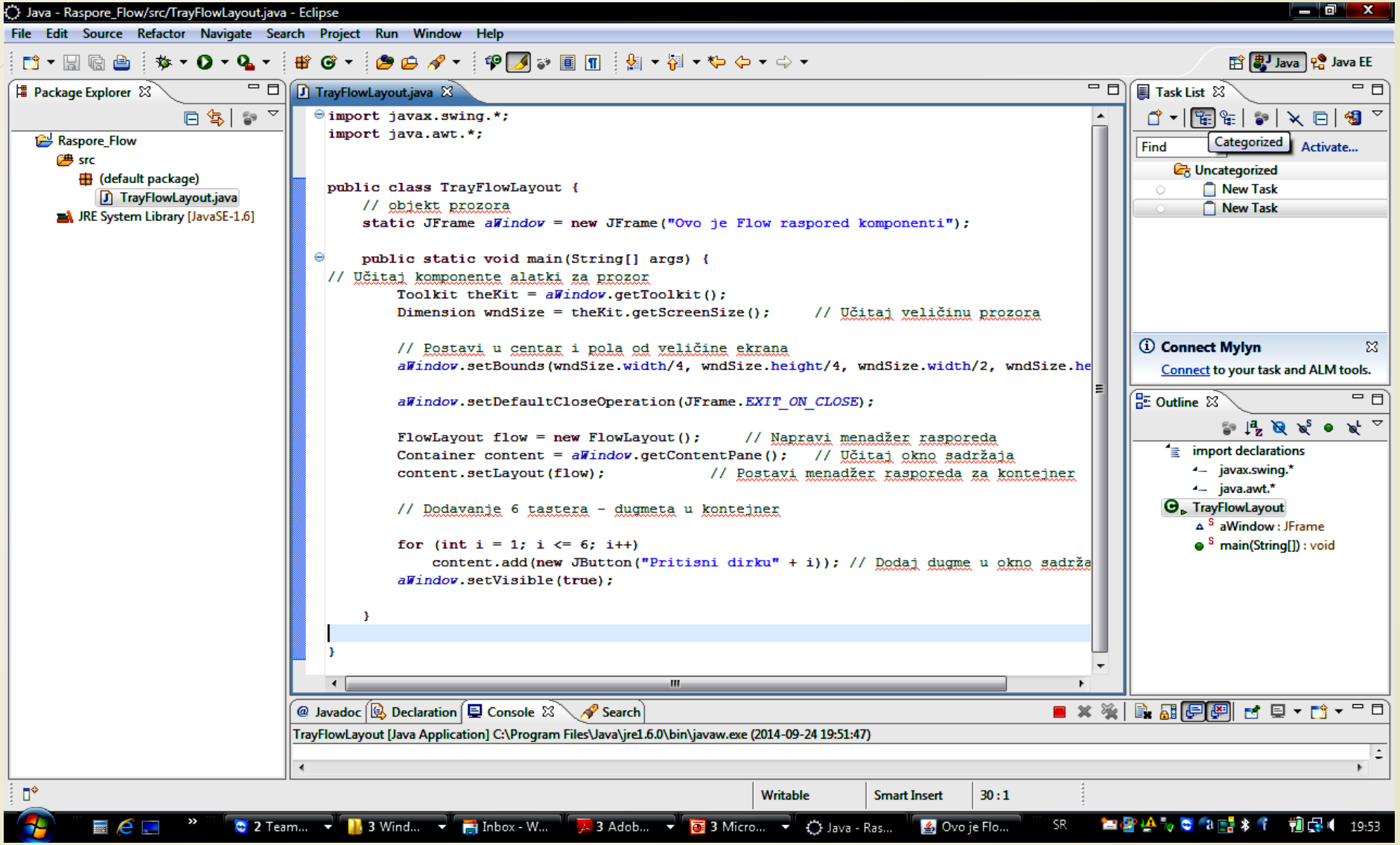
Primer: raspored Flow (1)

```
import javax.swing.JFrame;
import javax.swing.JButton;
import java.awt.Toolkit;
import java.awt.Dimension;
import java.awt.Container;
import java.awt.FlowLayout;
public class TryFlowLayout {
// Objekt prozora
static JFrame aWindow = new JFrame("Ovo je Flow raspored komp.");
public static void main(String[] args) {
   Toolkit theKit = aWindow.getToolkit();    // Get the window toolkit
   Dimension wndSize = theKit.getScreenSize();  // Get screen size
```

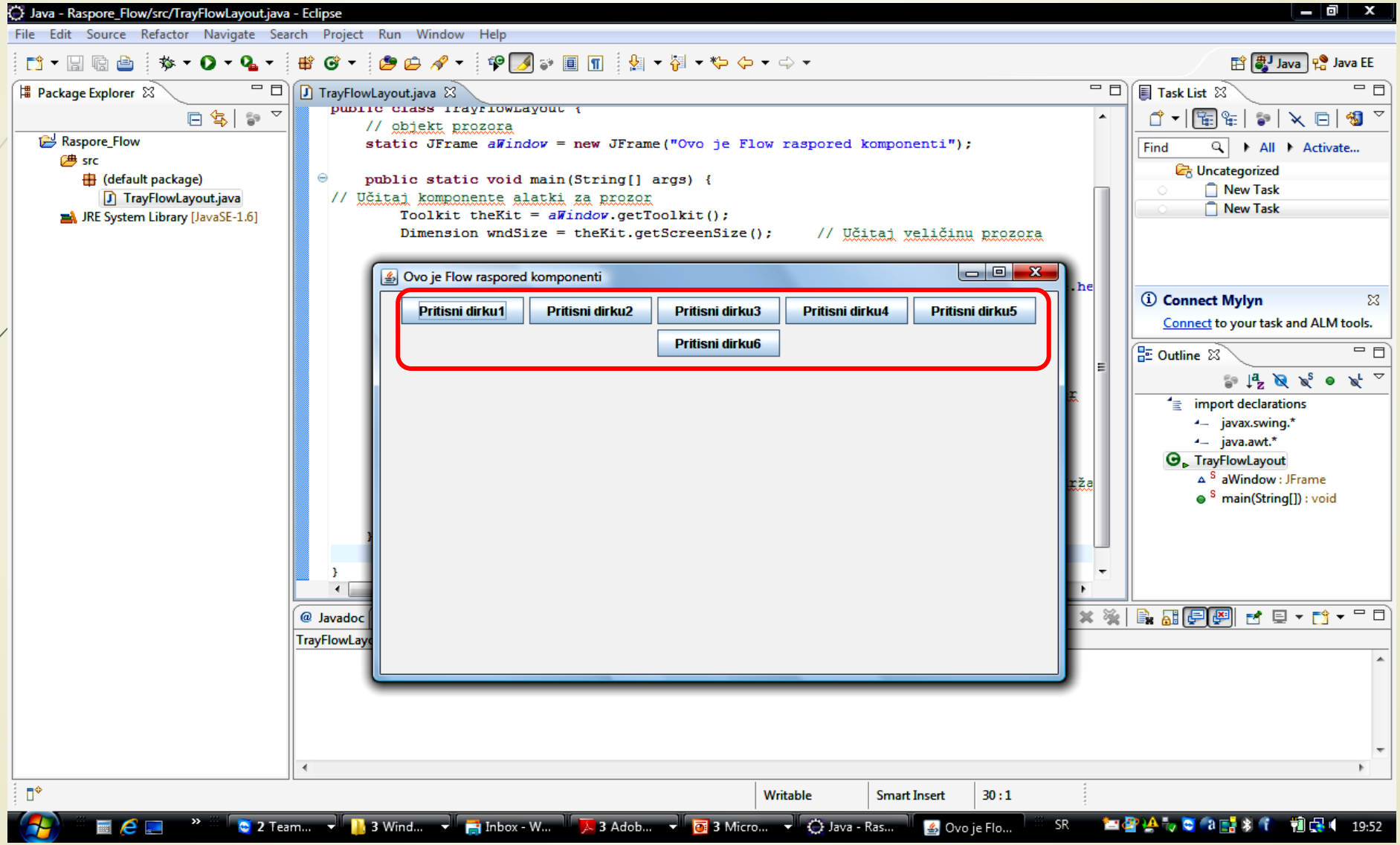
Primer: raspored Flow (2)

```
// Postavi poziciju na ekranu: center & veličina na ½ ekrana
    aWindow.setBounds(wndSize.width/4, wndSize.height/4,        // Pozicija
                      wndSize.width/2, wndSize.height/2);       // Veličina
    aWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    FlowLayout flow = new FlowLayout();                         // Kreiraj layout menadžer
    Container content = aWindow.getContentPane();               // Uzmi sad. "pane"
    content.setLayout(flow);                                    // Postavi container layout mgr.

// Dodaj 6 taster komponenti kroz for petlju!
    for(int i = 1; i <= 6; i++)
     content.add(new JButton("Press " + i));                    // Dodaj taster. na "pane"
     aWindow.setVisible(true);                                  // Prikaži prozor
    }
}
```

Eclipse: Raspored Flow (1)



Eclipse: Raspored Flow (2)



Dodavanje menija u prozoru (1)

- Objekt **JMenuBar** predstavlja **LINIJU MENIJA** koja se postavlja na **VRH PROZORA**.
- U objekt **JMenuBar** mogu se dodati objekti JMenu, JMenuItem i biće prikazani na liniji menija.
- Objekt **JMenu** je član menija sa nazivom koji može da prikaže **PADAJUĆI MENI** kada se klikne na njega.
- Objekat **JMenuItem** predstavlja običan član čijim pritiskom se obavlja neka programska akcija.
- Rad sa članovima menija se odvija preko sledećih metoda:
  - **void setEnabled(boolean b)**,
  - **void setText(String label) i**
  - **String getText()**.
- U meni se mogu dodati i SEPARATORI pomoću metode **addSEpartor()**.

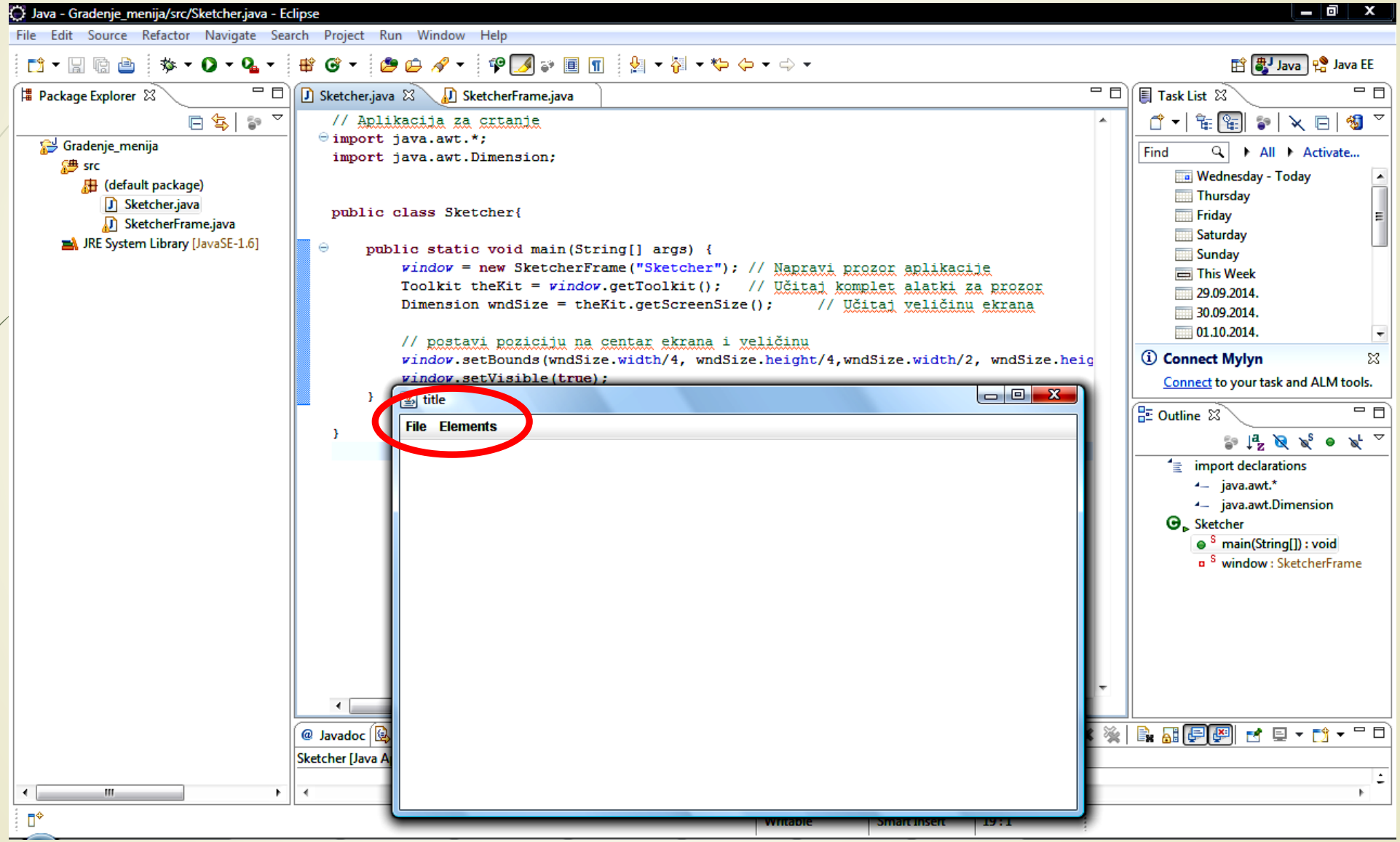
Dodavanje menija u prozoru (2)

```
// Aplikacija za crtanje
import java.awt.*;
import java.awt.Dimension;

public class Sketcher {
   public static void main(String[] args) {
   window = new SketcherFrame("Sketcher");                    // Prozor aplikacije
   Toolkit theKit = window.getToolkit();                      // Učitaj alat za prozor
   Dimension wndSize = theKit.getScreenSize();                // Učitaj vel. ekrana

// postavi poziciju na centar ekrana i veličinu
   window.setBounds(wndSize.width/4, wndSize.height/4,
                    wndSize.width/2, wndSize.height/2);
       window.setVisible(true);
   }
   private static SketcherFrame window;                       // Aplikacioni prozor
}
```

Dodavanje menija u prozoru (3)

```
import javax.swing.*;
public class SketcherFrame extends JFrame {

// Konstruktor
     public SketcherFrame (String title) {
     setTitle("title");
     setDefaultCloseOperation(EXIT_ON_CLOSE);
     setJMenuBar(menuBar);

     JMenu fileMenu = new JMenu("File");
     JMenu elementMenu = new JMenu("Elements");
     menuBar.add(fileMenu);
     menuBar.add(elementMenu);
    }
    private JMenuBar menuBar = new JMenuBar();
 }
}
```

Prozor sa Menijem



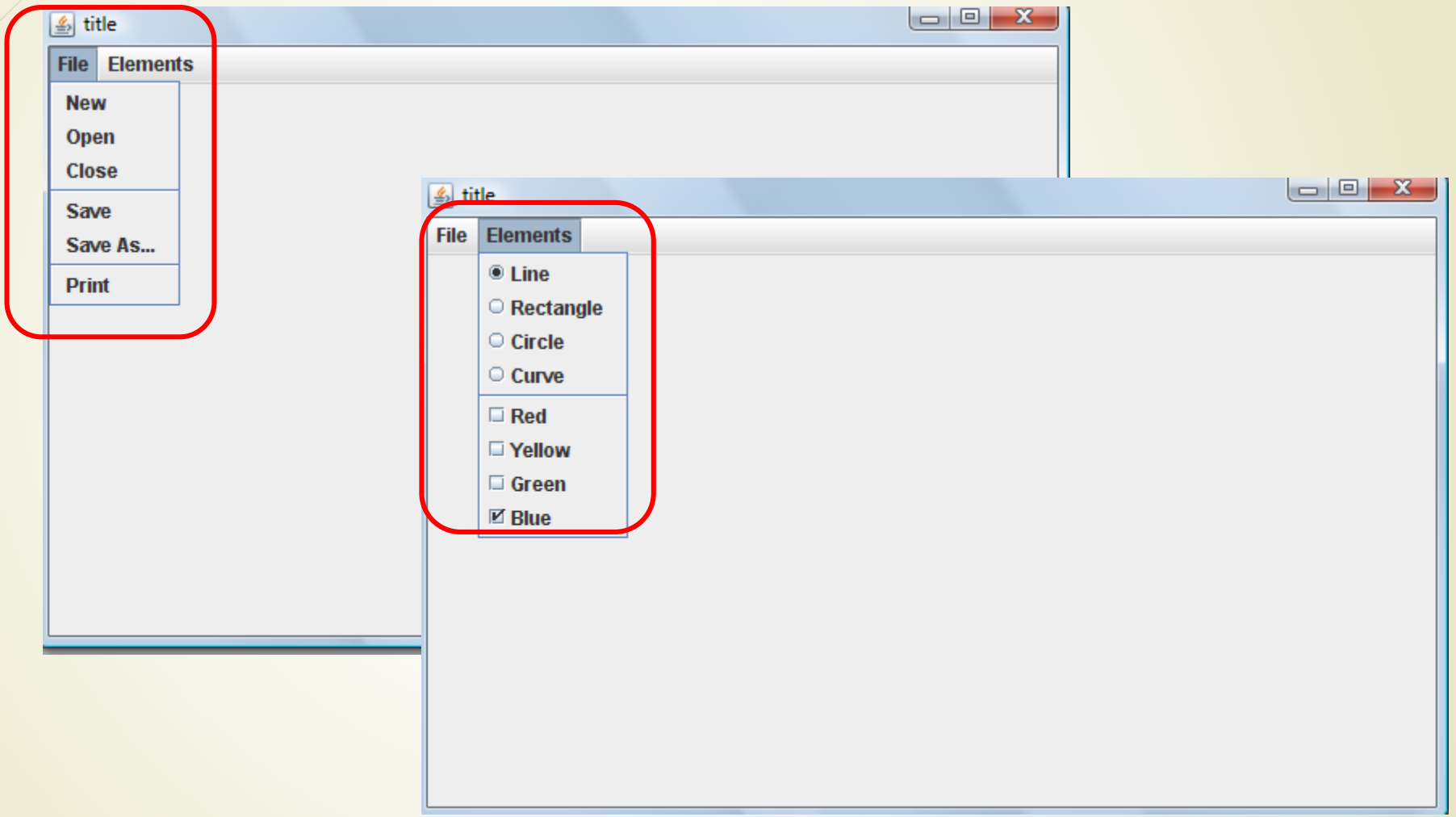
Dodavanje članova u meniu (1)

```
// Formiranje file DROP-DOWN meni


newItem = fileMenu.add("New");                  // Dodaj novi item
openItem = fileMenu.add("Open");                // Dodaj Open item
closeItem = fileMenu.add ("Close");             // Dodaj Close item
   fileMenu.addSeparator();                     // Dodaj separator
saveItem = fileMenu.add("Save");                // Dodaj Save item
saveAsItem = fileMenu.add("Save As...");        // Dodaj Save As item
   fileMenu.addSeparator();                     // Dodaj separator
printItem = fileMenu.add("Print");              // Dodaj Print item
```

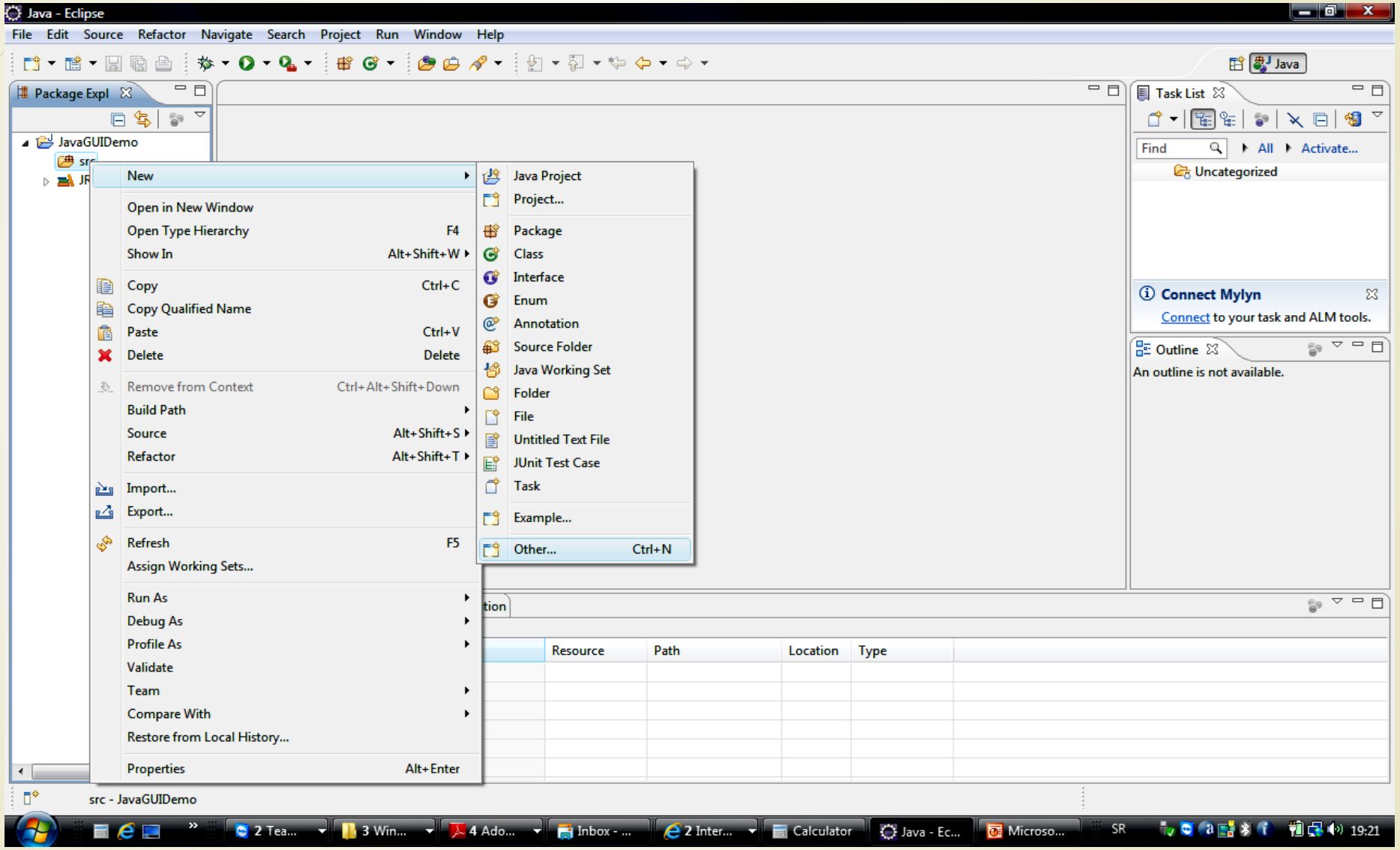
Dodavanje članova u meniu (2)

```
// Formiranje elemenata drop-down menija
elementMenu.add(lineItem = new JRadioButtonMenuItem("Line", true));
elementMenu.add(rectangleItem = new JRadioButtonMenuItem("Rectangle", false));
elementMenu.add(circleItem = new JRadioButtonMenuItem("Circle", false));
elementMenu add(curveItem = new JRadioButtonMenuItem("Curve", false));
ButtonGroup types = new ButtonGroup();
types add(lineItem);
types add(rectangleItem);
types add(circleItem);
types add(curveItem);
elementMenu addSeparator();
elementMenu add(redItem = new JCheckBoxMenuItem("Red", false));
elementMenu add(yellowItem = new JCheckBoxMenuItem("Yellow", false));
elementMenu add(greenItem = new JCheckBoxMenuItem("Green", false));
elementMenu add(blueItem = new JCheckBoxMenuItem("Blue", true));
```

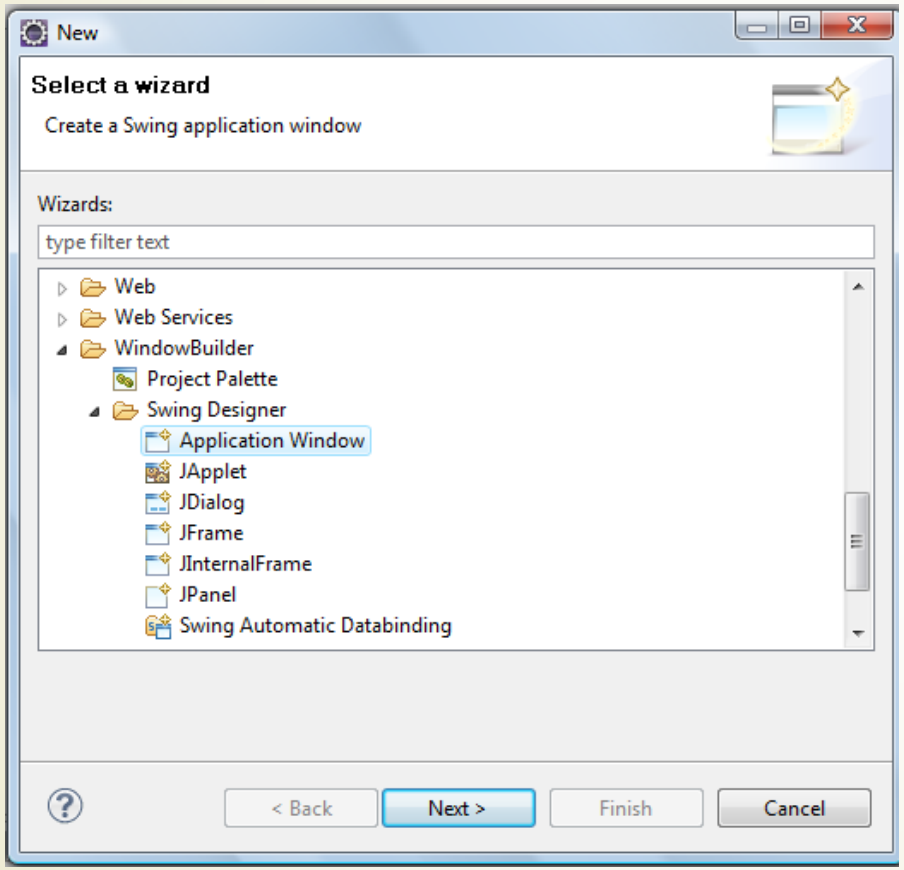
Dodavanje članova u meniu (2)

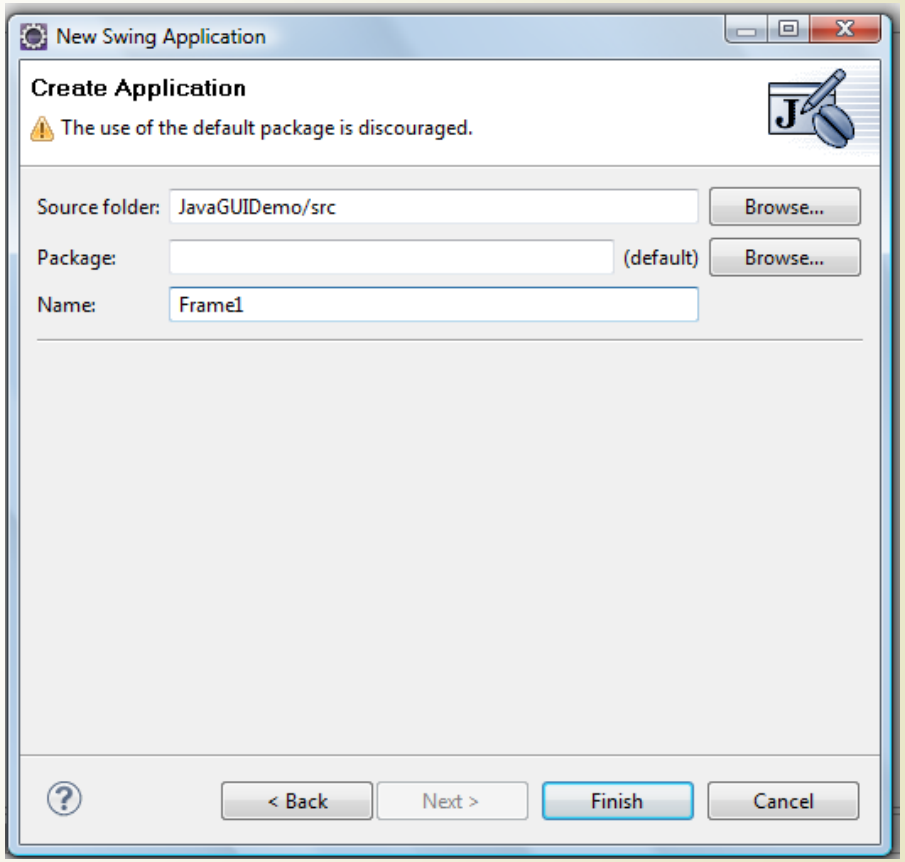


WindowBuilder i Eclipse (1)

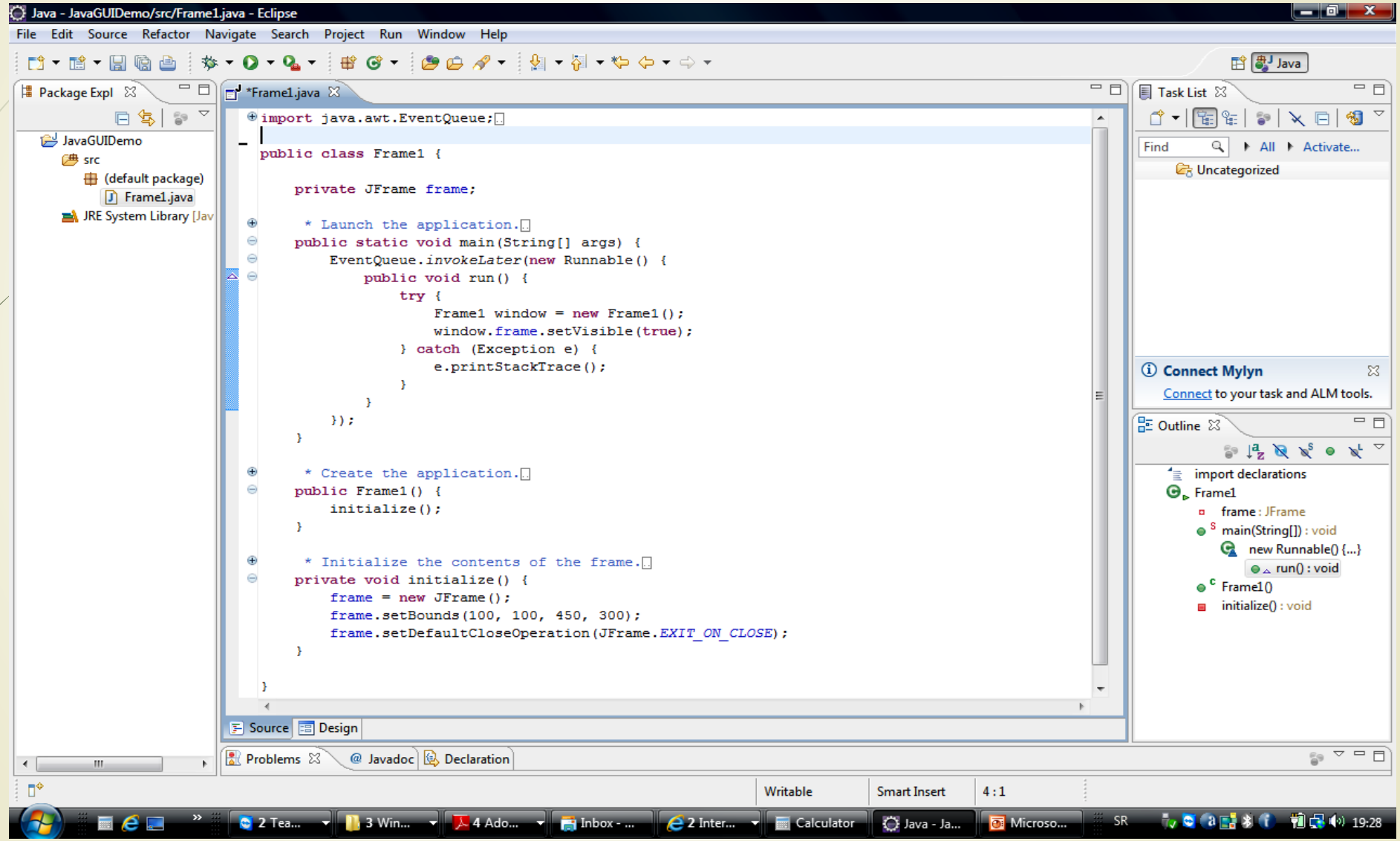


WindowBuilder i Eclipse (2)

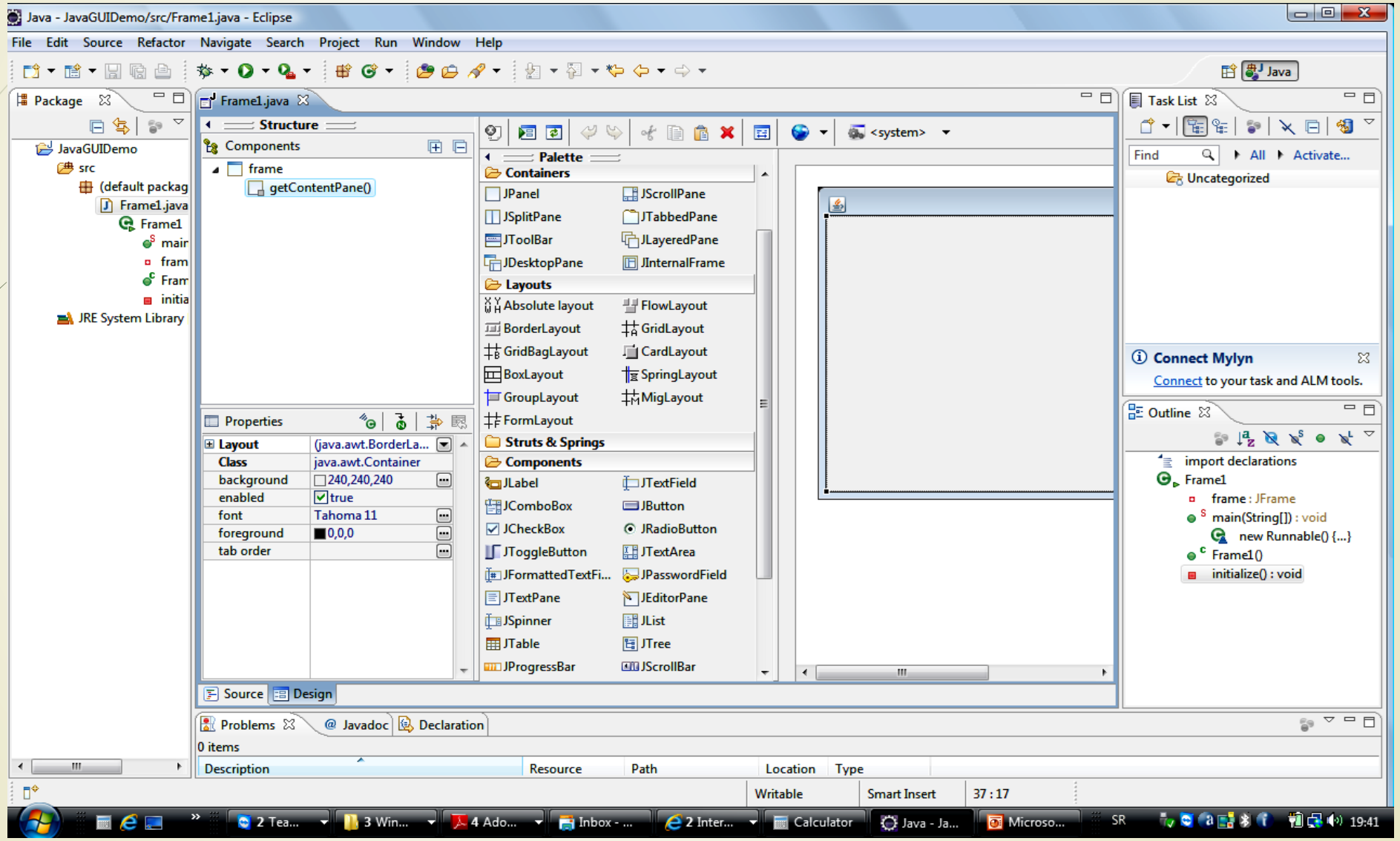




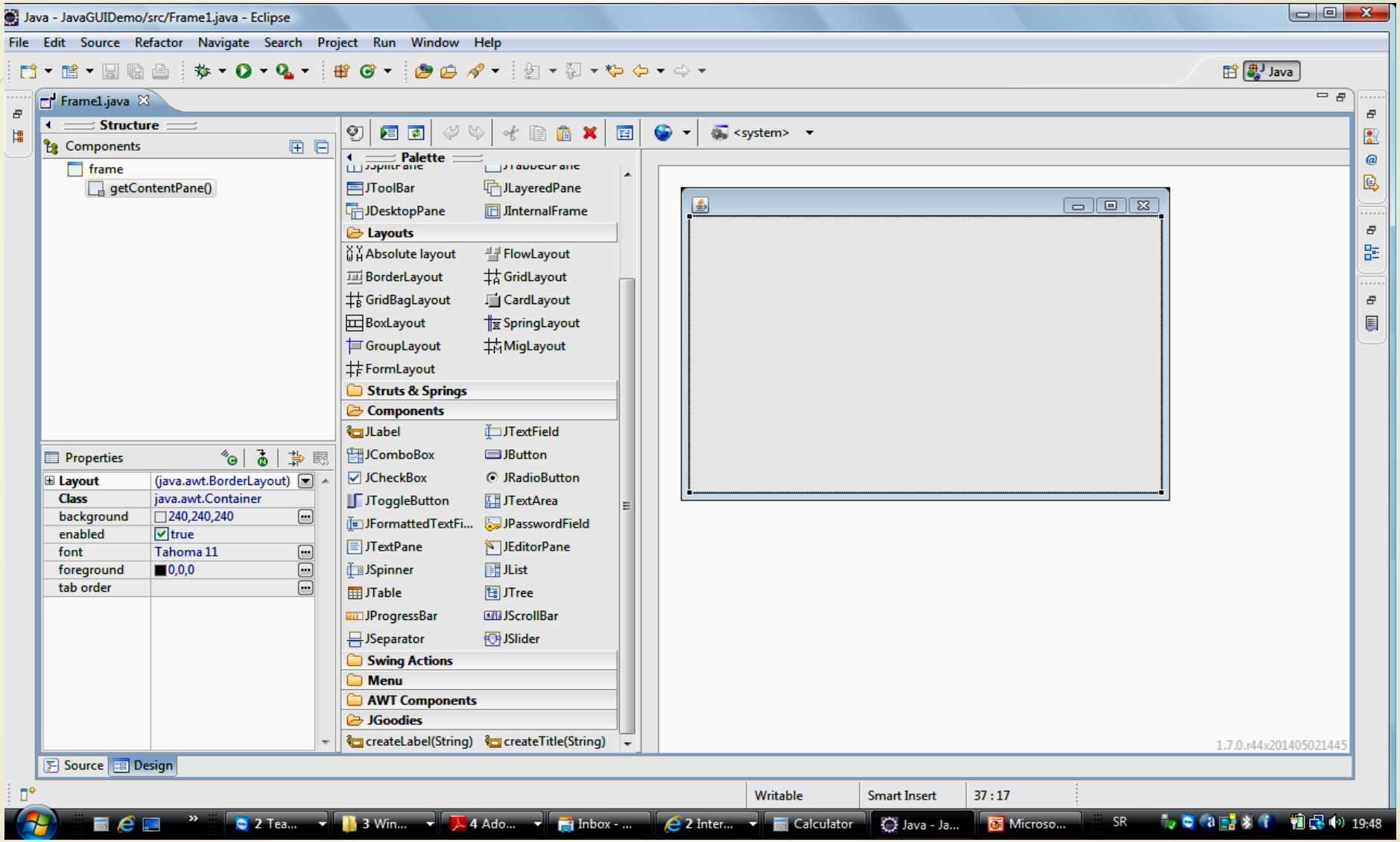
WindowBuilder i Eclipse (3)



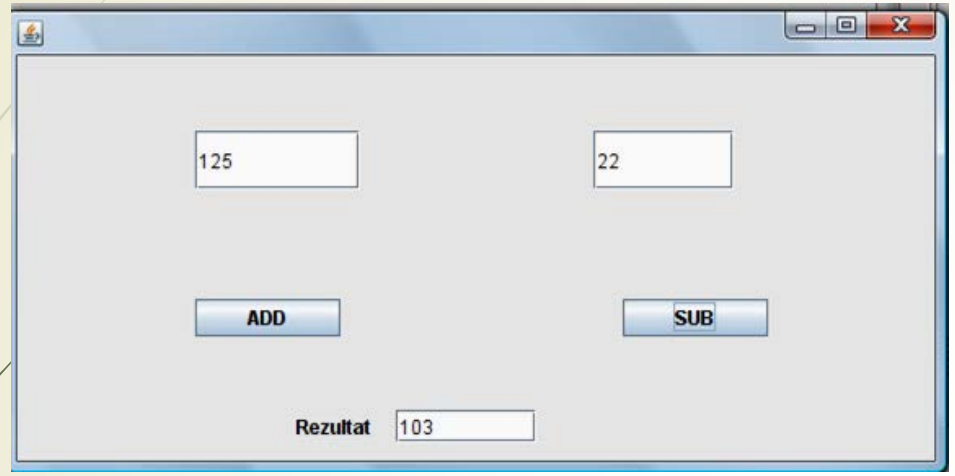
WindowBuilder i Eclipse (4)

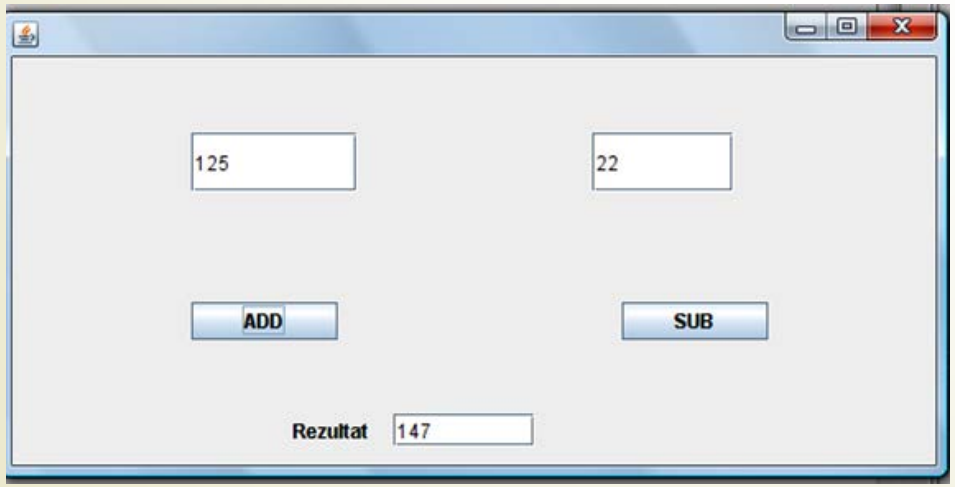


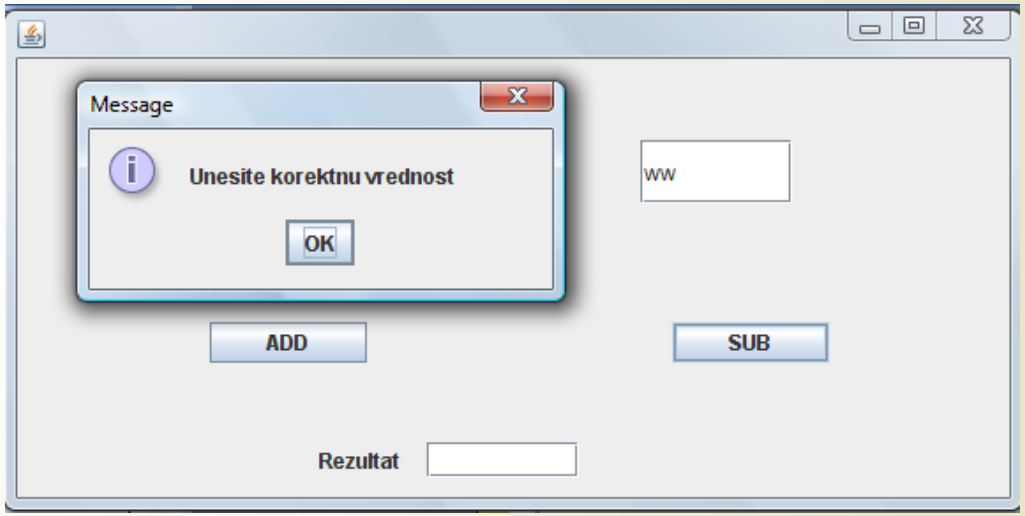
WindowBuilder i Eclipse (5)



Kalkulator i WindowBuilder







Kompletan izvorni kod (1)

```
import java.awt.EventQueue; import javax.swing.JFrame; import javax.swing.JOptionPane;
import javax.swing.JTextField; import javax.swing.JButton; import java.awt.event.ActionListener; import
    java.awt.event.ActionEvent; import javax.swing.JLabel;


public class frame {
    private JFrame frame;
    private JTextField textNum_1;
    private JTextField textNum_2;
    private JTextField textFieldREZ;
     public static void main(String[] args) {
     EventQueue.invokeLater(new Runnable() {
     public void run() {
     try {
          frame window = new frame();
          window.frame.setVisible(true);
     } catch (Exception e) {
          e.printStackTrace();
     }
         }
    });
 }
```

Kompletan izvorni kod (2)

```
public frame() {
    initialize();
}
    private void initialize() {
     frame = new JFrame();
     frame.setBounds(100, 100, 578, 284);
     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     frame.getContentPane().setLayout(null);
     textNum_1 = new JTextField();
     textNum_1.setBounds(109, 46, 101, 36);
     frame.getContentPane().add(textNum_1);
     textNum_1.setColumns(10);

     textNum_2 = new JTextField();
     textNum_2.setBounds(353, 46, 86, 36);
     frame.getContentPane().add(textNum_2);
     textNum_2.setColumns(10);
     JButton btnNewButton = new JButton("ADD");
     btnNewButton.addActionListener(new ActionListener() {
```

Kompletan izvorni kod (3)

```
public void actionPerformed(ActionEvent arg0) {
    int num1, num2, ans;
    try{
     num1=Integer.parseInt(textNum_1.getText());
     num2=Integer.parseInt(textNum_2.getText());
     ans = num1 + num2;
     textFieldREZ.setText(Integer.toString(ans));
    }catch(Exception e){
JOptionPane.showMessageDialog(null, "Unesite korektnu vrednost");
} } });
    btnNewButton.setBounds(109, 149, 89, 23);
    frame.getContentPane().add(btnNewButton);
    JButton btnSub = new JButton("SUB");
    btnSub.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
    int num1, num2, ans;
    try{
     num1=Integer.parseInt(textNum_1.getText());
     num2=Integer.parseInt(textNum_2.getText());
     ans = num1 - num2;
     textFieldREZ.setText(Integer.toString(ans));
```

Kompletan izvorni kod (4)

```
public void actionPerformed(ActionEvent arg0) {
}catch(Exception e){
    JOptionPane.showMessageDialog(null, "Unesite korektnu vrednost");
   }
   }
});

   btnSub.setBounds(371, 149, 89, 23);
   frame.getContentPane().add(btnSub);

   textFieldREZ = new JTextField();
   textFieldREZ.setBounds(232, 217, 86, 20);
   frame.getContentPane().add(textFieldREZ);
   textFieldREZ.setColumns(10);

   JLabel lblNewLabel = new JLabel("Rezultat");
   lblNewLabel.setBounds(171, 220, 46, 14);
   frame.getContentPane().add(lblNewLabel);
  }
}
```