



Akademija tehničko-vaspitačkih studija odsek NIŠ

Savremene računarske tehnologije SRT

OBJEKTNO ORIJENTISANO PROGRAMIRANJE - OOP

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

2019/2020.



Webinar OOP

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

OBJEKTNO ORIJENTISANO PROGRAMIRANJE - OOP

Grafičko programiranje u Javi

(13)

Sadržaj

▶ GRAFIČKI KORISNIČKI INTERFEJS – GUI U JAVI

▶ Paket `java.awt`

- ▶ Hijerarhija klasa za prozore
- ▶ Prozor tipa `Frame` u apletu
- ▶ `Jframe` klasa za funkcionalni prozor
- ▶ `JFrame` prozor u Eklipsu
 - ▶ Dimenzije ekrana

▶ KOMPONENTE I KONTEJNERI

- ▶ Veličina i pozicija komponenti
- ▶ Vizuelne karakteristike komponenti

▶ SWING KOMPONENTE

- ▶ Dugme
- ▶ Meni
- ▶ Tekst

▶ UPOTREBA KONTEJNERA

- ▶ Menadžeri rasporeda
- ▶ Metoda `setLayout()`
 - ▶ Primer: raspored `Flow`
- ▶ Dodavanje menija u prozoru

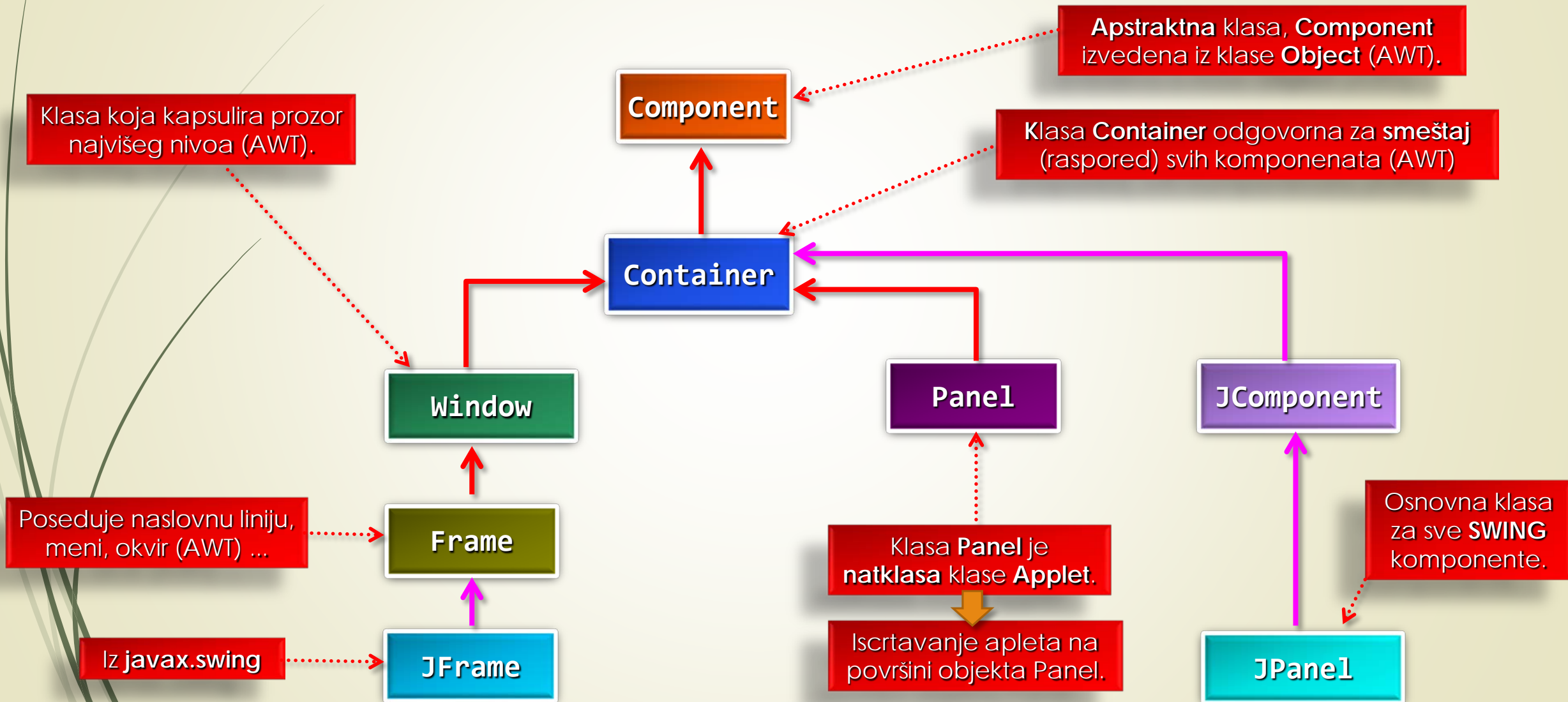
▶ WINDOWBUILDER I ECLIPSE

- ▶ Kalkulator
 - ▶ Izvorni kod

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 obezbeđuje **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 windowHeight = 400;           // Širina prozora u pikselima, param.
    int windowWidth = 150;           // Visina prozora u pikselima , param.
    aWindow.setBounds(50, 100, windowHeight, windowWidth);
                                        // 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

The screenshot displays the Eclipse IDE interface. The main editor window shows the source code for `TrayWindow.java`. The code defines a class `TrayWindow` that extends `JFrame` and implements a `main` method to create and display a window.

```
import javax.swing.*;
public class TrayWindow {
    static JFrame aWindow = new JFrame("Naslov prozora");

    public static void main(String[] args) {
        int windowWidth = 400;
        int windowHeight = 150;

        aWindow.setBounds(50, 100, windowWidth, windowHeight);
        aWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        aWindow.setVisible(true);
    }
}
```

Below the code editor, a small window titled "Naslov prozora" is visible, representing the output of the program. The window is empty and has a standard Windows-style title bar with minimize, maximize, and close buttons.

The Package Explorer on the left shows the project structure: `JFrame_primer` (default package) containing `TrayWindow.java`. The Task List on the right is empty. The Outline on the right shows the class structure: `import declarations`, `javax.swing.*`, `TrayWindow`, `aWindow : JFrame`, and `main(String[]) : void`.

The Console at the bottom shows the execution path: `TrayWindow [Java Application] C:\Program Files\Java\jre1.6.0\bin\javaw.exe (2014-09-24 13:52:31)`.

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 **Dimension** 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)

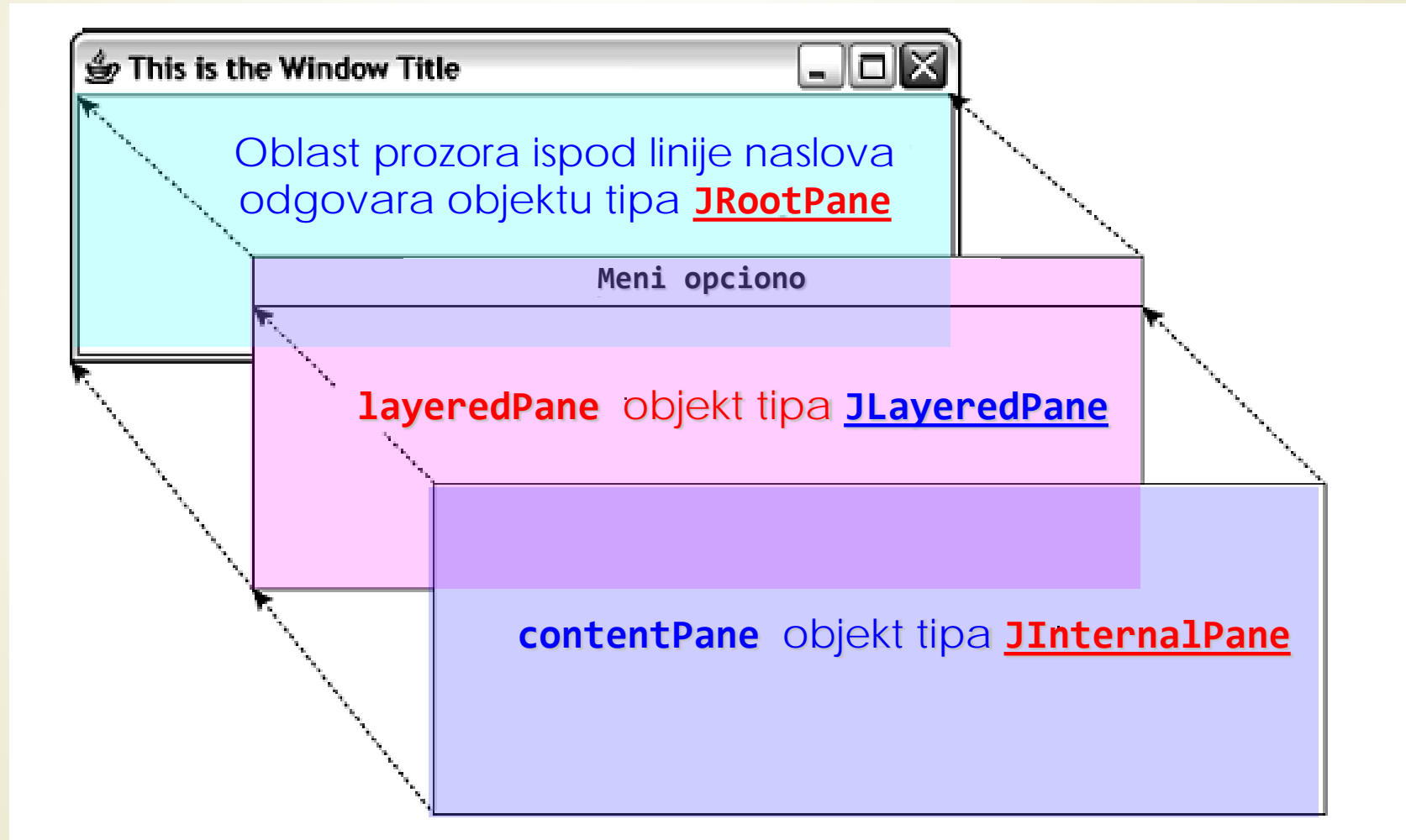
- ▶ U Javi se pod **KOMPONENTOM** smatra grafički entitet koji se može prikazati u prozoru, a dobijen je iz klase **Component** (na osnovu ove definicije i **JFrame** je takođe komponenta).
- ▶ Neke od podklasa klase **Component** su (pogledaj prethodno datu hijerarhiju klasa):
 - ▶ **JFrame**, koristi 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 applete;
 - ▶ **JComponent**, definiše niz standardnih komponenti 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 **JLayeredPane** (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 **JGlassPane** 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
<code>void setBounds(int x, int y, int width, int height)</code>	Postavlja poziciju objekta na željene koordinate i širinu i visinu objekta na zahtevane vrednosti.
<code>void setBounds(Rectangle rect)</code>	Postavlja poziciju i veličinu objekta na vrednost <code>rect</code> objekta Rectangle .
<code>void setSize(Dimension d)</code>	Postavlja širinu i visinu objekta na vrednosti postavljene u članovima objekta <code>d</code> .
<code>setLocation(int x, int y)</code>	Postavlja poziciju komponente na tačku određenu vrednostima <code>(x,y)</code> .
<code>setLocation(Point p)</code>	Postavlja poziciju komponente na tačku <code>p</code> .
<code>void setMinimumSize(Dimension d)</code>	Postavlja minimalnu veličinu objekta određene sa <code>d</code> .
<code>void setPreferredSize(Dimension d)</code>	Postavlja maksimalnu veličinu objekta određene sa <code>d</code> .
<code>void setMaximumSize(Dimension d)</code>	Postavlja željenu veličinu objekta određene sa <code>d</code> .

Vizuelne karakteristike komponenti

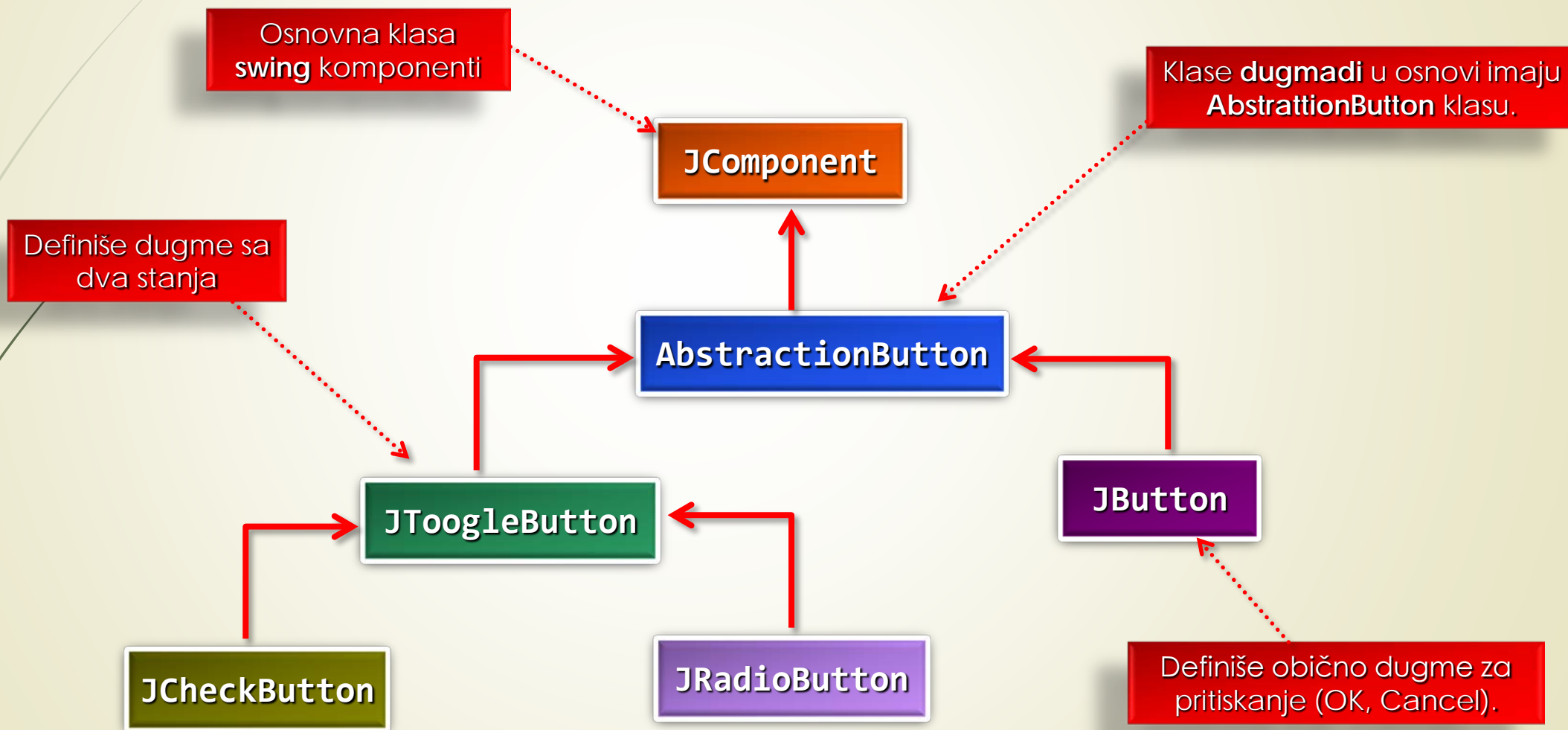
- Selektovane metode koje poseduje klasa **Component** za zadavanje VIZUELNIH KARAKTERISTIKA komponentama su prikazane u tabeli.

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

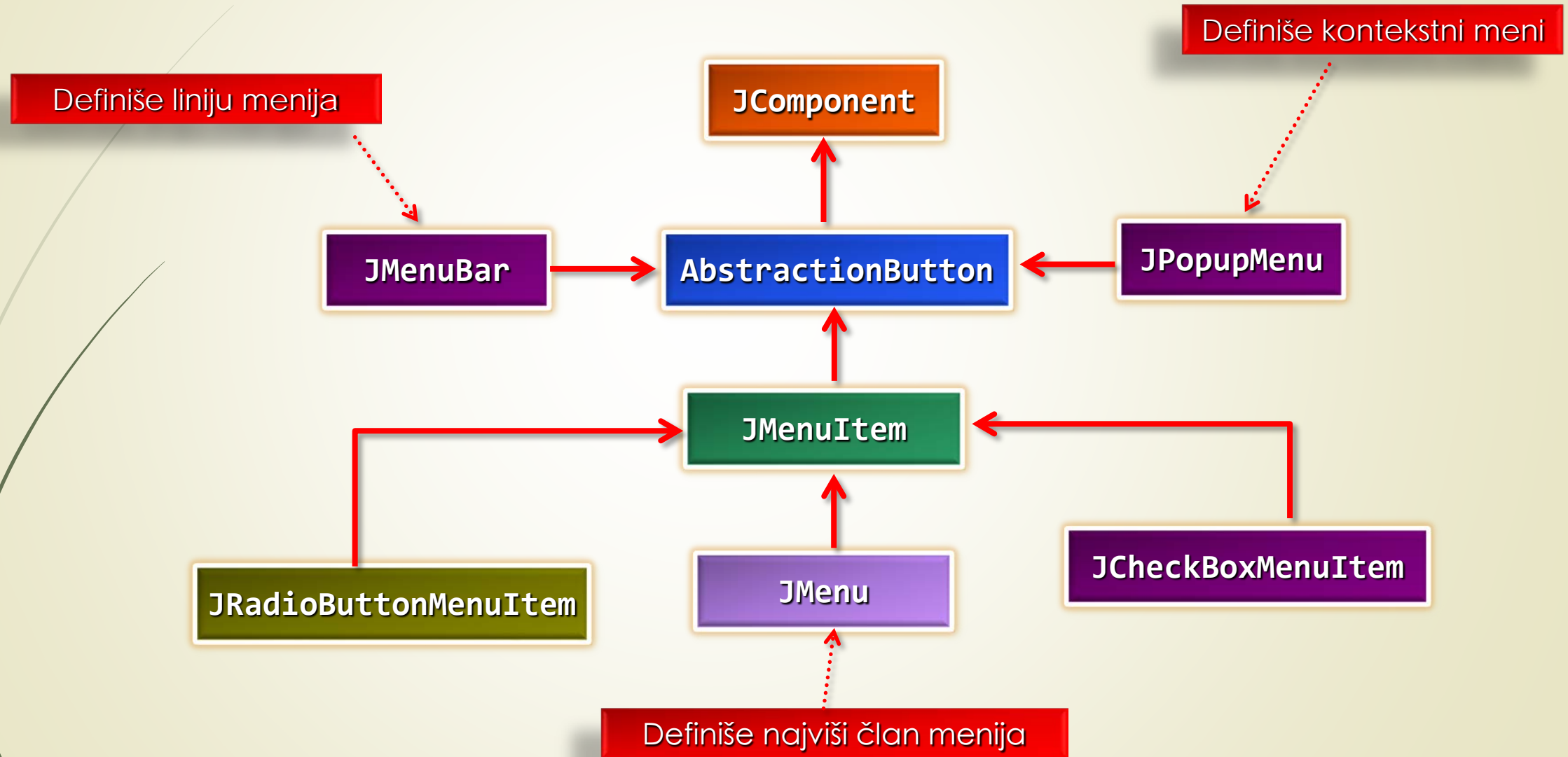
SWING komponente

- Sve klase **SWING** komponenti definisane su u paketu **javax.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



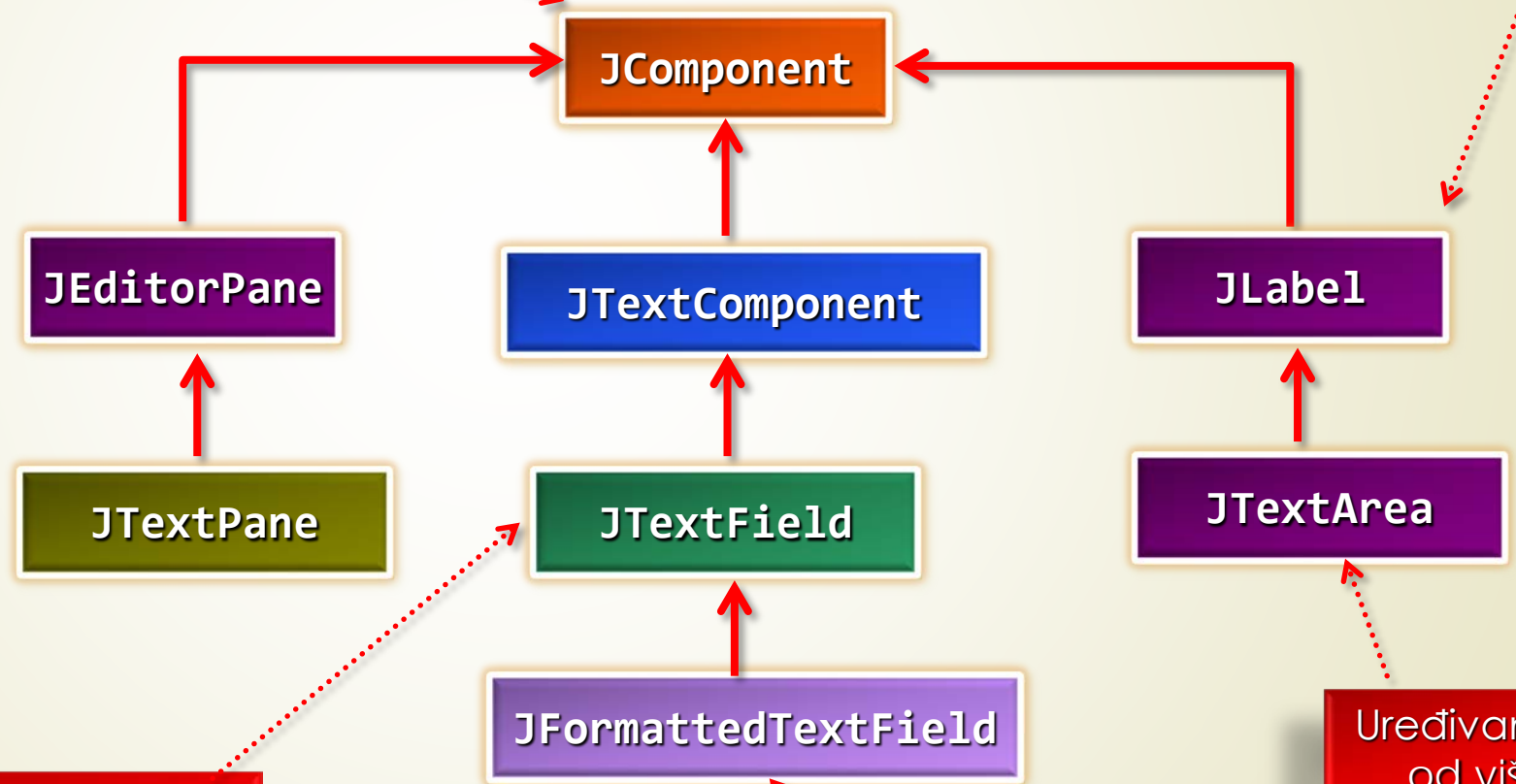
Komponenta: meni swing



Komponenta: swing tekst

Osnovna klasa swing komponenti

Osnovna tekst komponenta



Prikaz jedne linije teksta koj se može editovati

Formatira podatke

Uređivanje teksta od više linija

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 preklapljene 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 rasporeda 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 četiri 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,  
    wndSize.width/2, wndSize.height/2);
```

```
aWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
FlowLayout flow = new FlowLayout();
```

```
Container content = aWindow.getContentPane();
```

```
content.setLayout(flow);
```

```
// Dodaj 6 taster komponenti kroz for petlju!
```

```
for(int i = 1; i <= 6; i++)
```

```
    content.add(new JButton("Press " + i));
```

```
    aWindow.setVisible(true);
```

```
}
```

```
}
```

```
// Pozicija
```

```
// Veličina
```

```
// Kreiraj layout menadžer
```

```
// Uzmi sad. "pane"
```

```
// Postavi container layout mgr.
```

```
// Dodaj taster. na "pane"
```

```
// Prikaži prozor
```

Eclipse: Raspored Flow (1)

The screenshot displays the Eclipse IDE interface with the following components:

- Package Explorer:** Shows the project structure for 'Raspore_Flow', including a 'src' folder, a '(default package)', and the file 'TrayFlowLayout.java'. The 'JRE System Library [JavaSE-1.6]' is also visible.
- TrayFlowLayout.java:** The main code editor showing the following code:

```
import javax.swing.*;
import java.awt.*;

public class TrayFlowLayout {
    // objekt prozora
    static JFrame aWindow = new JFrame("Ovo je Flow raspored komponenti");

    public static void main(String[] args) {
        // Učitaj komponente alatki za prozor
        Toolkit theKit = aWindow.getToolkit();
        Dimension wndSize = theKit.getScreenSize(); // Učitaj veličinu prozora

        // Postavi u centar i pola od veličine ekrana
        aWindow.setBounds(wndSize.width/4, wndSize.height/4, wndSize.width/2, wndSize.height/2);

        aWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        FlowLayout flow = new FlowLayout(); // Napravi menadžer rasporeda
        Container content = aWindow.getContentPane(); // Učitaj okno sadržaja
        content.setLayout(flow); // Postavi menadžer rasporeda za kontejner

        // Dodavanje 6 tastera - dugmeta u kontejner
        for (int i = 1; i <= 6; i++)
            content.add(new JButton("Pritisni dirku" + i)); // Dodaj dugme u okno sadržaja
        aWindow.setVisible(true);
    }
}
```
- Task List:** Shows a 'Categorized' view with two 'New Task' items.
- Connect Mylyn:** A panel with a 'Connect' button to link tasks and ALM tools.
- Outline:** Shows the class hierarchy: 'import declarations' (javax.swing.*, java.awt.*), 'TrayFlowLayout' (with fields 'aWindow: JFrame' and 'main(String[]): void').
- Bottom Panel:** Includes 'Javadoc', 'Declaration', 'Console', and 'Search' tabs. The console shows the command: 'TrayFlowLayout [Java Application] C:\Program Files\Java\jre1.6.0\bin\javaw.exe (2014-09-24 19:51:47)'. The status bar at the bottom indicates 'Writable', 'Smart Insert', and '30:1'.

Eclipse: Raspored Flow (2)

The screenshot displays the Eclipse IDE interface. The main editor window shows the source code for `TrayFlowLayout.java`. The code defines a `JFrame` window titled "Ovo je Flow raspored komponenti" and contains a `main` method that creates a `Toolkit` and a `Dimension` object. A dialog window titled "Ovo je Flow raspored komponenti" is overlaid on the IDE, showing a `FlowLayout` arrangement of six buttons: "Pritisni dirku1", "Pritisni dirku2", "Pritisni dirku3", "Pritisni dirku4", "Pritisni dirku5", and "Pritisni dirku6". The buttons are arranged in two rows: the first row contains buttons 1 through 5, and the second row contains button 6. A red rectangle highlights the buttons in the first row. The IDE's Package Explorer on the left shows the project structure, and the Outline view on the right shows the class and method structure.

```
public class TrayFlowLayout {  
    // objekt prozora  
    static JFrame aWindow = new JFrame("Ovo je Flow raspored komponenti");  
  
    public static void main(String[] args) {  
        // Učitaj komponente alatki za prozor  
        Toolkit theKit = aWindow.getToolkit();  
        Dimension wndSize = theKit.getScreenSize(); // Učitaj veličinu prozora  
    }  
}
```

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 **addSeparator()**.

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");
        Toolkit theKit = window.getToolkit();
        Dimension wndSize = theKit.getScreenSize();

        // Prozor aplikacije
        // Učitaj alat za prozor
        // 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

The screenshot displays the Eclipse IDE interface for a Java project named "Gradenje_menija". The main editor window shows the source code for "Sketcher.java". The code defines a "Sketcher" class with a "main" method that creates a "SketcherFrame" window. The window is titled "Sketcher" and is positioned at the center of the screen with a size of 250x250 pixels. The code is as follows:

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

public class Sketcher{

    public static void main(String[] args) {
        window = new SketcherFrame("Sketcher"); // Napravi prozor aplikacije
        Toolkit theKit = window.getToolkit(); // Učitaj komplet alatki za prozor
        Dimension wndSize = theKit.getScreenSize(); // Učitaj veličinu 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);
    }
}
```

The "SketcherFrame" class is defined in "SketcherFrame.java" and is currently empty. The "Package Explorer" on the left shows the project structure, including the "src" folder and the "Sketcher.java" and "SketcherFrame.java" files. The "Task List" on the right shows a calendar view for the week of September 29, 2014. The "Outline" on the right shows the class structure, including the "Sketcher" class and its "main" method. A red circle highlights the "File Elements" menu item in the "SketcherFrame" window, which is currently empty.

Dodavanje članoa u meniu (1)

// Formiranje file DROP-DOWN meni

```
newItem = fileMenu.add("New");
```

```
openItem = fileMenu.add("Open");
```

```
closeItem = fileMenu.add ("Close");
```

```
    fileMenu.addSeparator();
```

```
saveItem = fileMenu.add("Save");
```

```
saveAsItem = fileMenu.add("Save As...");
```

```
    fileMenu.addSeparator();
```

```
printItem = fileMenu.add("Print");
```

// Dodaj novi item

// Dodaj Open item

// Dodaj Close item

// Dodaj separator

// Dodaj Save item

// Dodaj Save As item

// Dodaj separator

// Dodaj Print item

Dodavanje članoa u meniu (2)

// Formiranje elemenata drop-down menija

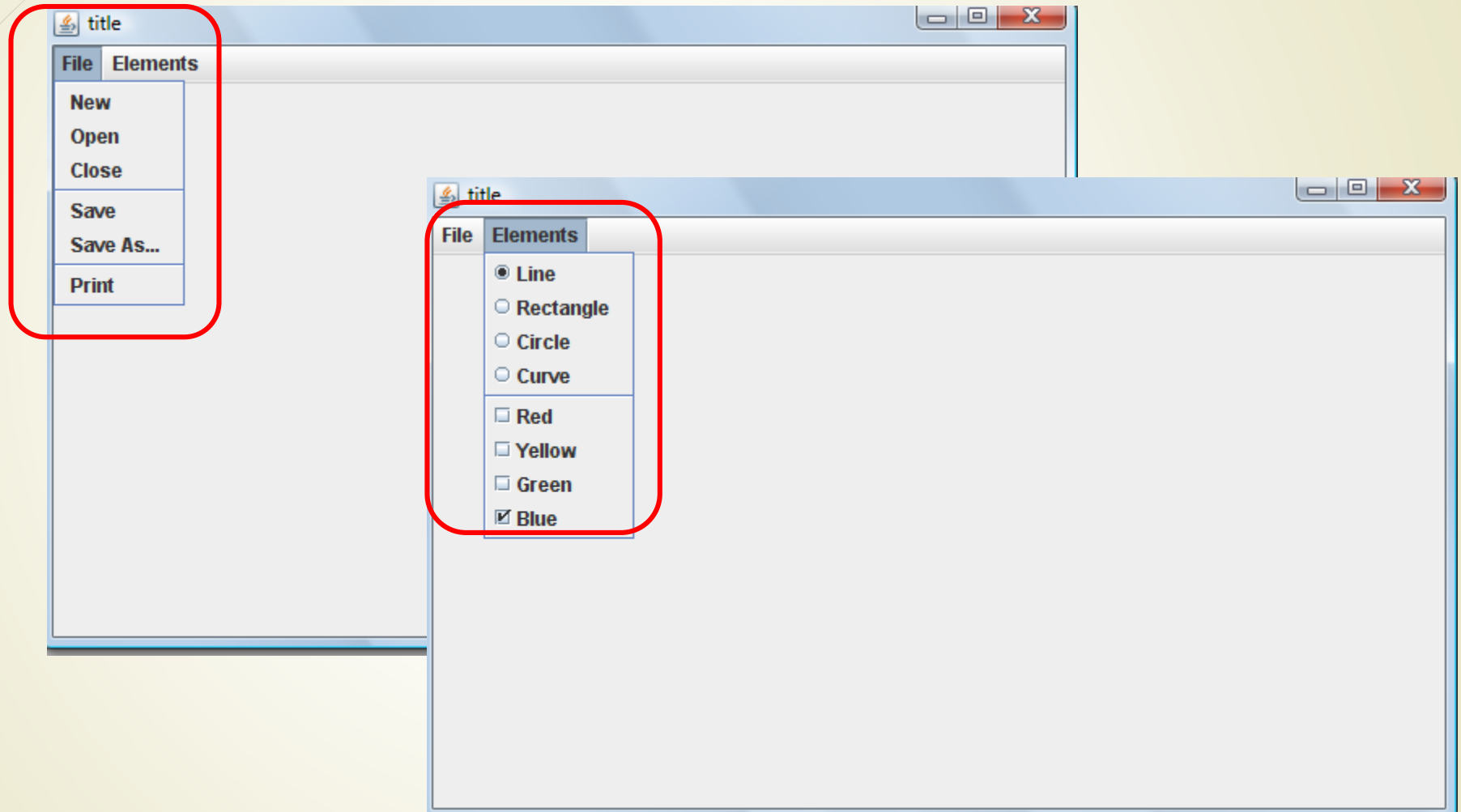
```
elementMenu.add(lineItem = new JRadioButtonMenuItem("Line", true));
elementMenu.add(rectangleItem = new JRadioButtonMenuItem("Rectangle", false));
elementMenu.add(circleItem = new JRadioButtonMenuItem("Circle", false));
elementMenu.add(curveItem = new JRadioButtonMenuItem("Curve", false));

ButtonGroup types = new ButtonGroup();
types.add(lineItem);
types.add(rectangleItem);
types.add(circleItem);
types.add(curveItem);

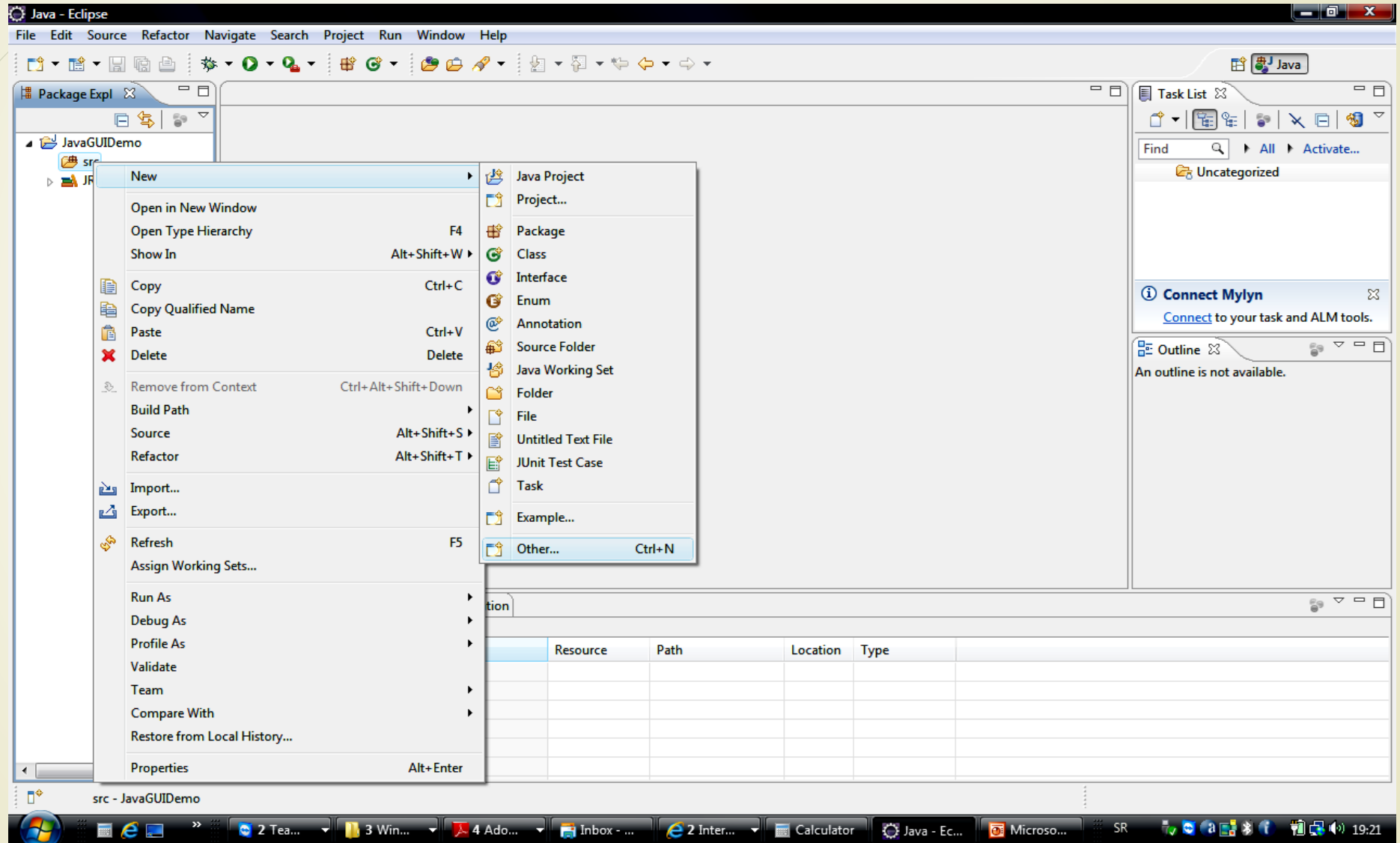
elementMenu.addSeparator();

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));
```

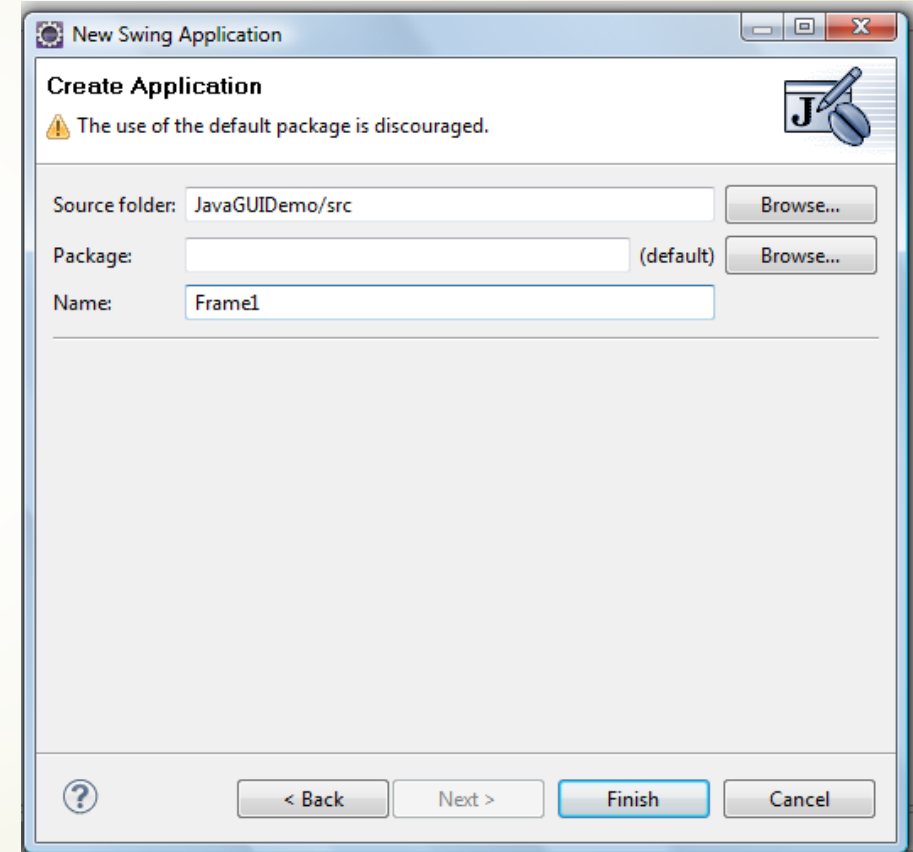
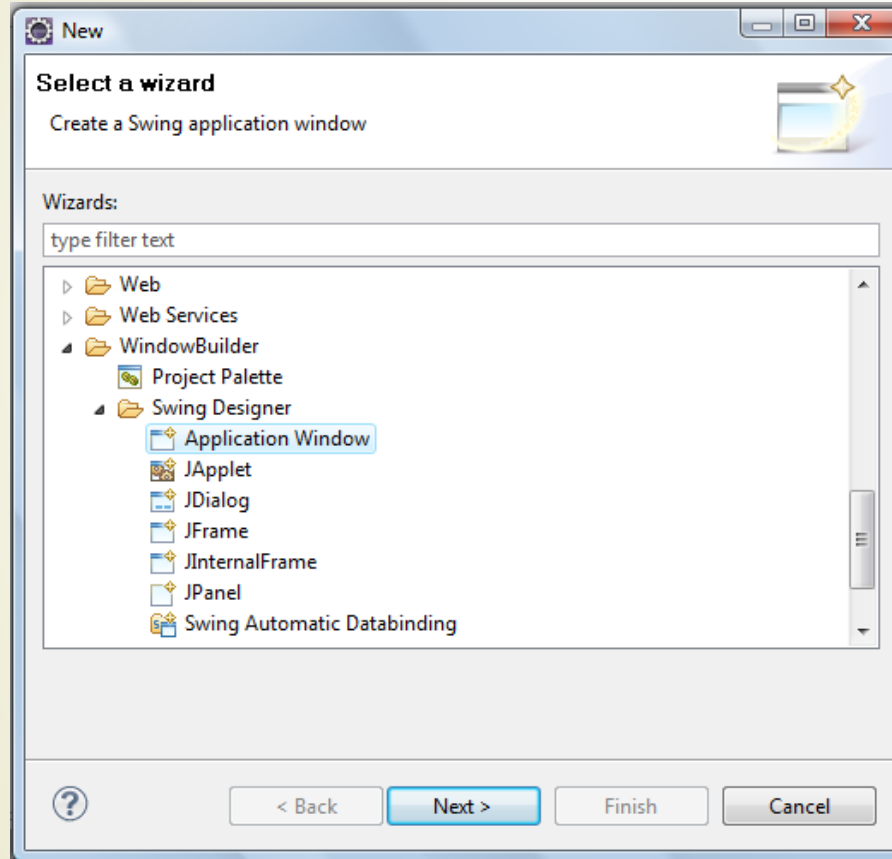
Dodavanje članova u meniu (2)



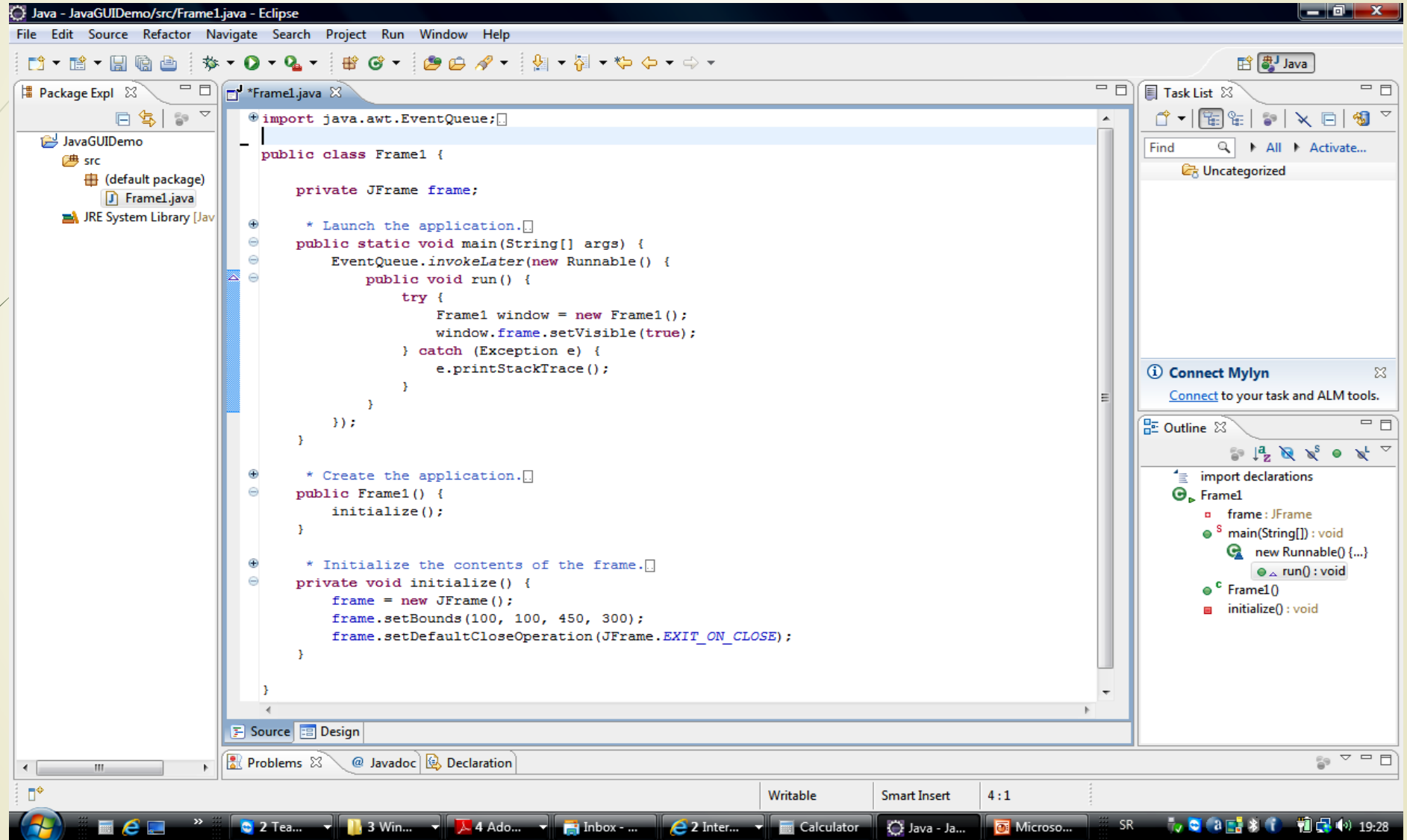
WindowBuilder i Eclipse (1)



WindowBuilder i Eclipse (2)



WindowBuilder i Eclipse (3)



```
import java.awt.EventQueue;

public class Frame1 {

    private JFrame frame;

    * Launch the application.
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Frame1 window = new Frame1();
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    * Create the application.
    public Frame1() {
        initialize();
    }

    * Initialize the contents of the frame.
    private void initialize() {
        frame = new JFrame();
        frame.setBounds(100, 100, 450, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

WindowBuilder i Eclipse (4)

The screenshot displays the Eclipse IDE interface with the WindowBuilder design view active for a Java Swing window. The main workspace shows a simple rectangular window with a title bar. The left sidebar contains the Package Explorer, showing the project structure with 'Frame1.java' selected. The Structure view shows the 'frame' component and its 'getContentPane()' method. The Palette view is open, displaying various Java Swing components and layouts. The Properties view shows the 'Layout' property set to 'java.awt.BorderLayout'. The Outline view shows the class structure of 'Frame1', including 'main(String[]) : void' and 'initialize() : void'. The bottom status bar shows the current line and column position as '37 : 17'.

Java - JavaGUIDemo/src/Frame1.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer: JavaGUIDemo, src, (default package), Frame1.java, Frame1, mair, fram, Fram, initia, JRE System Library

Structure: Components, frame, getContentPane()

Palette: Containers (JPanel, JSplitPane, JToolBar, JDesktopPane, JScrollPane, JTabbedPane, JLayeredPane, JInternalFrame), Layouts (Absolute layout, BorderLayout, GridBagLayout, BoxLayout, GroupLayout, FormLayout, FlowLayout, GridLayout, CardLayout, SpringLayout, MigLayout), Struts & Springs, Components (JLabel, JComboBox, JCheckBox, JToggleButton, JFormattedTextFi..., JTextPane, JSpinner, JTable, JProgressBar, JTextField, JButton, JRadioButton, JTextArea, JPasswordField, JEditorPane, JList, JTree, JScrollBar)

Properties: Layout (java.awt.BorderLa...), Class (java.awt.Container), background (240,240,240), enabled (true), font (Tahoma 11), foreground (0,0,0), tab order

Outline: import declarations, Frame1, frame : JFrame, main(String[]) : void, new Runnable() {...}, Frame1(), initialize() : void

Task List: Find, All, Activate..., Uncategorized

Connect Mylyn: Connect to your task and ALM tools.

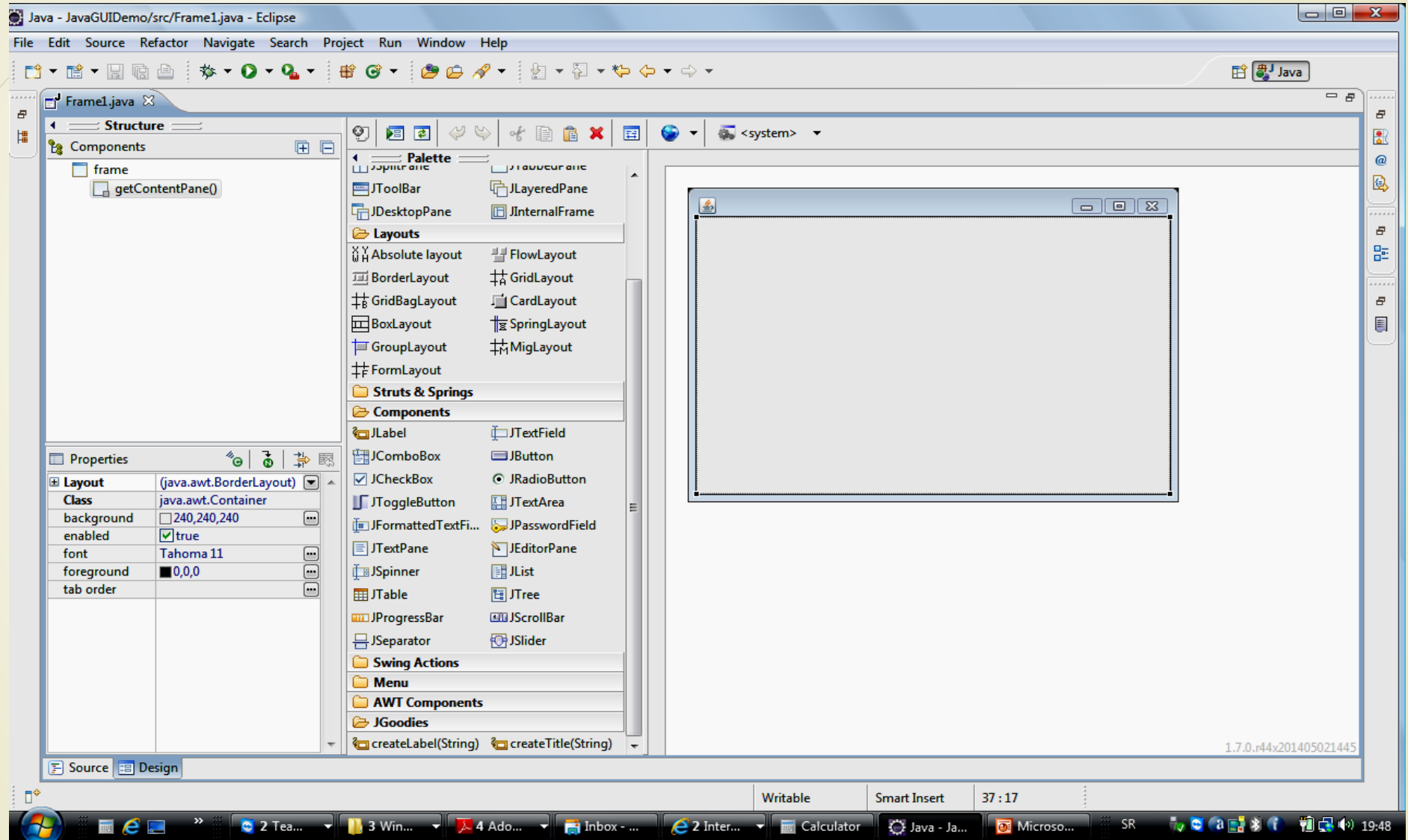
Problems: 0 items

Bottom Bar: Writable, Smart Insert, 37 : 17

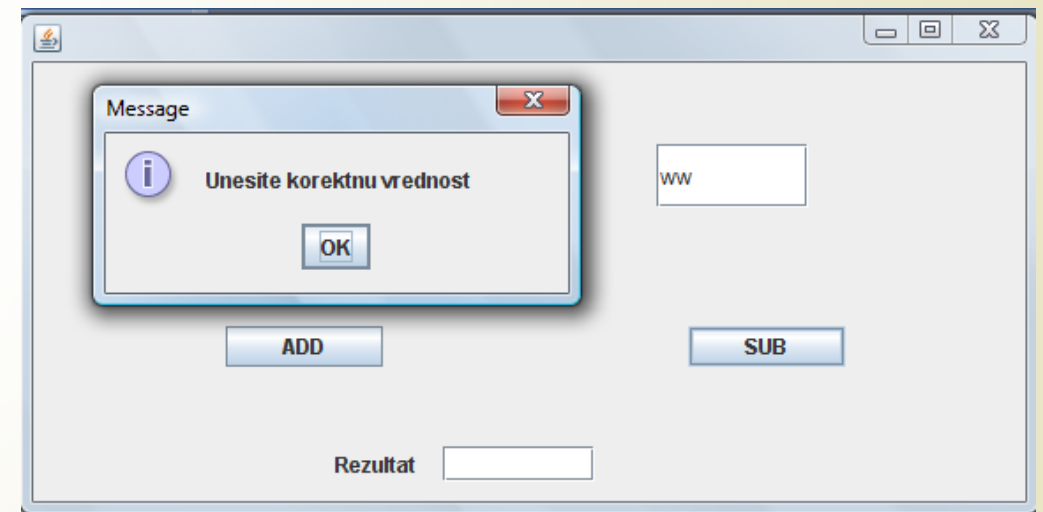
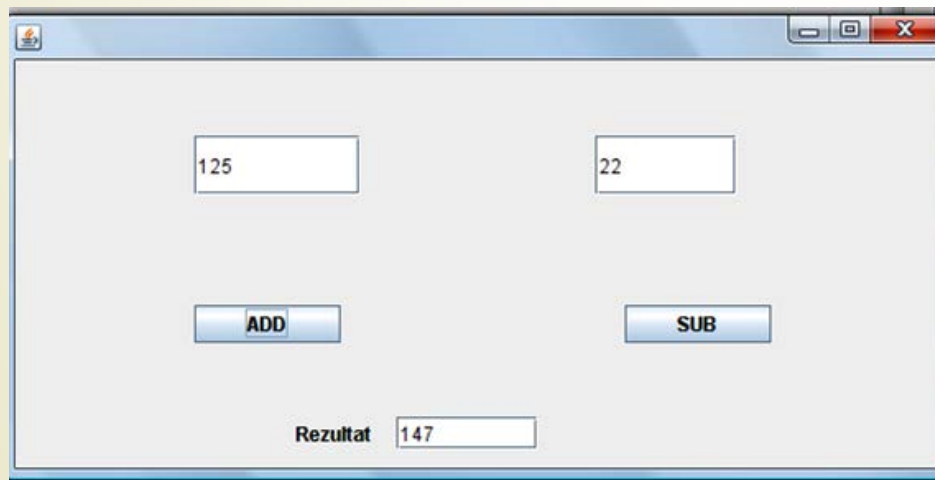
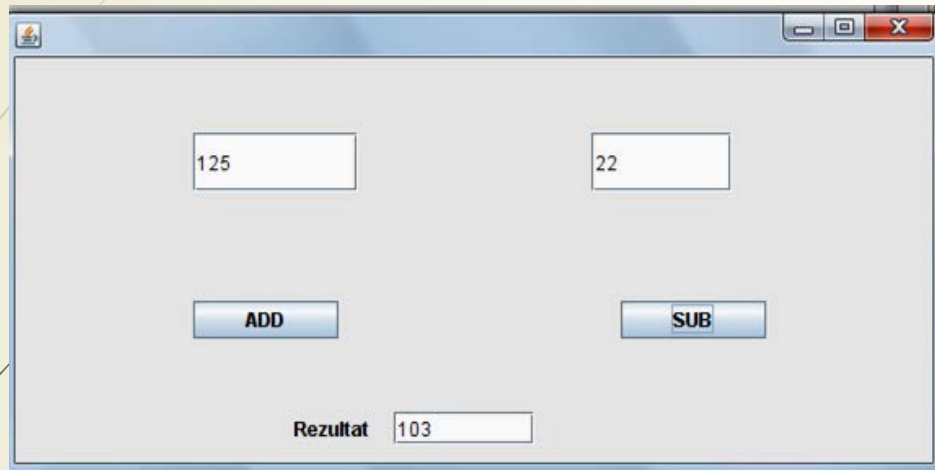
Windows: 2 Tea..., 3 Win..., 4 Ado..., Inbox - ..., 2 Inter..., Calculator, Java - Ja..., Microso...

System Tray: SR, 19:41

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);
}
```