GUI

Programski jezici II

Uvod

- korisnički interfejs je upravljan događajima (Event Driven)
- u verzijama 1.0 i 1.1 standard je AWT biblioteka (Abstract Window Toolkit) – biblioteka koja se zasniva na korišćenju komponenti korisničkog interfejsa koje su dostupne na platformi na kojoj se program pokreće
- implementacija AWT komponenti – različita za svaki operativni sistem
- izgled aplikacija – kao da su pisane u bilo kom drugom jeziku na datoj platformi
- od verzije 1.2 standard je Swing biblioteka

Abstract Window Toolkit

- AWT: biblioteka koja obezbjeđuje upotrebu minimalnog skupa komponenti grafičkog interfejsa, a kojeg posjeduju sve platforme koje podržavaju Javu
- izgleda “podjednako osrednje” na svim platformama
- paketi:

```
java.awt
java.awt.event
java.awt.image
java.awt.datatransfer
```

Abstract Window Toolkit

- klase podijeljene u 3 kategorije: grafika, komponente i layout manager-i
- grafičke klase: manipulacija oblicima, bojama, fontovima...
- komponente: dugme, meni, dijalog...
- layout manager-i: smještanje grafike i komponenti na predefinisane pozicije

JFC

- Java Foundation Classes
  - izgradnja grafičkog korisničkog interfejsa
  - dodaju bogate grafičke funkcionalnosti i interaktivnost Java aplikacijama
- karakteristike JFC-a
  - Swing GUI komponente
  - Look-and-Feel podrška
  - Accessibility API
  - Java 2D API
  - internacionalizacija

Swing

- napisan “od nule”, u Javi
- ne oslanja se na operativni sistem, komponente se samostalno iscrtavaju na ekranu
- jednako izgleda i radi na svim platformama
- veliki broj komponenti, tj. nema ograničenja na broj i tip GUI komponenti koje će ući u biblioteku
- Look-and-feel: Windows, Motif, Metal, Mac
- Drag&Drop, rad sa clipboard-om
- kompletna podrška za Unicode kodnu stranu
- nazivi klasa počinju sa Jxxx

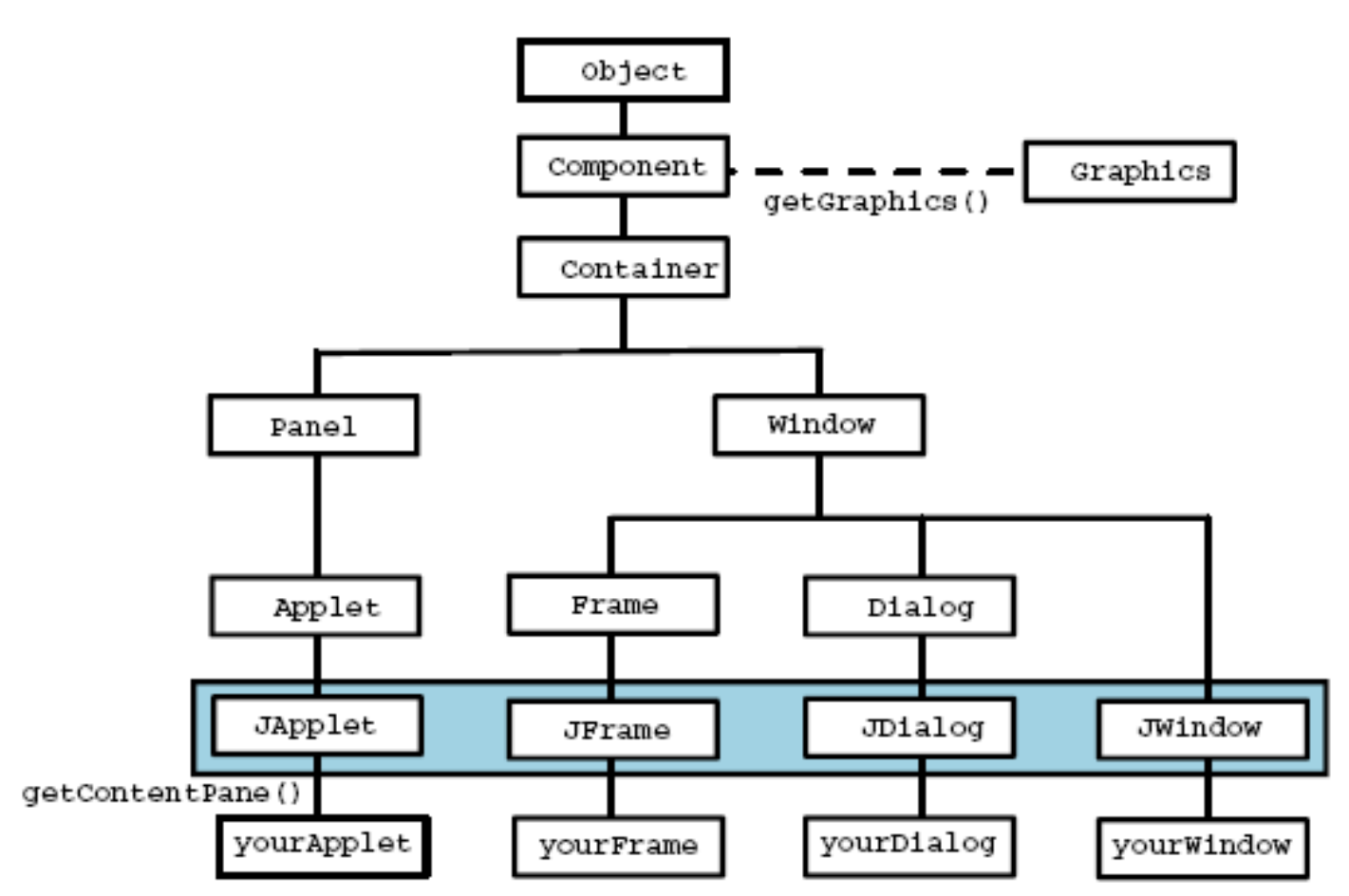
Swing

- Swing API – “moćan”, fleksibilan i ogroman
- paketi:
  - javax.accessibility
  - javax.swing.plaf
  - javax.swing.text
  - javax.swing
  - javax.swing.plaf.basic
  - javax.swing.text.html
  - javax.swing.border
  - javax.swing.plaf.metal
  - javax.swing.text.html.parser
  - javax.swing.colorchooser
  - javax.swing.plaf.multi
  - javax.swing.text.rtf
  - javax.swing.event
  - javax.swing.plaf.synth
  - javax.swing.tree
  - javax.swing.filechooser
  - javax.swing.table
  - javax.swing.undo

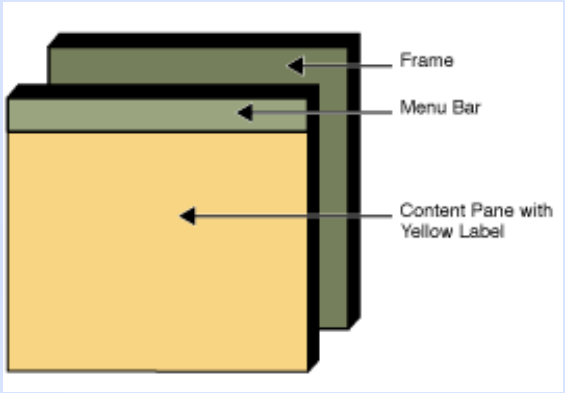
Top-level kontejneri

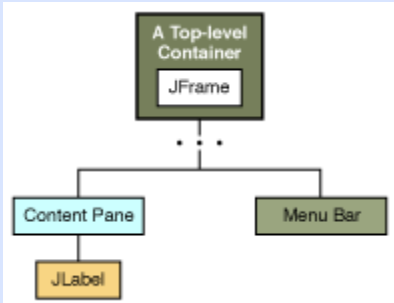
- 4 top-level Swing kontejnera:
  - JApplet, JFrame, JDialog, JWindow
- Apleti - nasljeđuju JApplet
- GUI aplikacija može naslijediti JFrame
- JDialog se obično koristi sa roditeljskim JFrame ili Dialog klasama
- JWindow nema titlebox, niti window management dugmad
- svi kontejneri obezbjeđuju getContentPane() metodu
- svi imaju podršku za meni - JMenuBar

Top-level kontejneri



Top-level kontejneri





- pri korišćenju ovih klasa bitno je znati:
  - da bi se mogla prikazati, svaka GUI komponenta mora biti dio sadržavajuće hijerarhije – sadržavajuća hijerarhija je stablo komponenti koje ima top-level kontejner kao root
  - svaka GUI komponenta može biti sadržana samo jednom – ako je komponenta dodata u jedan kontejner, i ako se nakon toga dodaje u drugi, biće izbačena iz prvog
  - svaki top-level kontejner ima content pane koji sadrži (direktno ili indirektno) vidljive komponente
  - menu bar se može dodati u top-level kontejner (opciono) – po konvenciji, smješta se unutar top-level kontejnera, ali izvan content pane-a

Top-level kontejneri

- svaki program koji koristi Swing komponente ima bar jedan top-level kontejner i bar jednu sadržavajuću hijerarhiju
- primjer
  - aplikacija ima JFrame i 2 dijaloga – ima 3 top-level kontejnera i 3 sadržavajuće hijerarhije

Top-level kontejneri

- dodavanje komponente u content pane

```
topLevelContainer.getContentPane().add(someComponent
   , BorderLayout.CENTER);
```

  - podrazumijevani content pane je intermediate kontejner koji nasljeđuje JComponent i koristi BorderLayout kao layout manager
  - kreiranje content pane-a i postavljanje u top-level kontejner

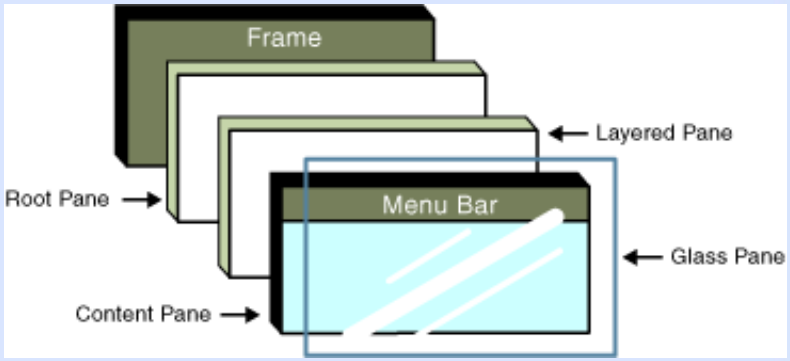
```
//kreiraj panel i dodaj komponente na njega
JPanel contentPane = new JPanel(new BorderLayout());
contentPane.setBorder(someBorder);
contentPane.add(someComponent, BorderLayout.CENTER);
contentPane.add(anotherComponent,
BorderLayout.PAGE_END);
topLevelContainer.setContentPane(contentPane);
```

Top-level kontejneri

- dodavanje menu bar-a

```
frame.setJMenuBar(greenMenuBar);
```

- svaki top-level kontejner oslanja se na intermediate kontejner – root pane



- pored content pane-a i menu bar-a root pane dodaje i layered pane i glass pane
- layered pane sadrži menu bar i content pane
- glass pane se koristi za presretanje događaja koji se dešavaju nad top-level kontejnerom – može se koristiti i za iscrtavanje komponenti

Intermediate kontejneri

- Swing obezbjeđuje dodatne klase koje se koriste kao intermediate kontejneri – sadrže druge komponente
- Intermediate kontejneri opšte namjene:
  - JPanel
  - JScrollPane
  - JSplitPane
  - JTabbedPane
  - JToolBar
- Specijalizovani:
  - JInternalFrame
  - JLayeredPane
  - JRootPane

JComponent

- osim top-level kontejnera, sve Swing komponente čiji naziv počinje sa "J" nasljeđuju JComponent klasu
- primjer:
  - JPanel, JScrollPane, JButton, JTable
  - JFrame i JDialog ne nasljeđuju JComponent jer one implementiraju top-level kontejnere
- JComponent klasa nasljeđuje Container klasu koja nasljeđuje Component klasu
- Component klasa pruža različite funkcionalnosti klasama koje je nasljeđuju

JComponent

- funkcionalnosti koje JComponent klasa obezbjeđuje klasama nasljednicama:
  - tool tip-ovi
    - setToolTipText metoda – obezbjeđuje pomoć korisnicima komponente
  - iscrtavanje i ivice
    - setBorder metoda obezbjeđuje iscrtavanje ivice komponente
    - paintComponent metoda za iscrtavanje unutar komponente
  - look and feel aplikacije
    - svaki JComponent objekat ima odgovarajući ComponentUI objekat koji vrši iscrtavanje, event handling, određivanje veličine i dr. za JComponent – koji tačno ComponentUI objekat se koristi zavisi od trenutnog look and feel-a, koji se postavlja pomoću UIManager.setLookAndFeel metode

JComponent

- funkcionalnosti koje JComponent klasa obezbjeđuje klasama nasljednicama:
  - custom properties – moguće je povezati jedan ili više property-ja (par ime-objekat) sa svakom JComponent-om – metode putClientProperty i getClientProperty
  - podrška za layout – JComponent klasa dodaje setter metode – setMinimumSize, setMaximumSize, setAlignmentX i setAlignmentY
  - podrška za dostupnost – JComponent klasa obezbjeđuje API i osnovne funkcionalnosti koje pomažu tehnologijama kao što su čitači ekrana da dobiju informacije od Swing komponente

JComponent

- funkcionalnosti koje JComponent klasa obezbjeđuje klasama nasljednicama:
  - podrška za drag and drop – JComponent klasa obezbjeđuje API za postavljanje transfer handler-a komponente, koji je osnova za Swing drag and drop podršku
  - dvostruko baferovanje – podrška iscrtavanju
  - key bindings – ova osobina omogućava da komponenta reaguje na pritisak tastera na tastaturi

JComponent

- metode za prilagođenje prikaza komponente:
  - void setBorder(Border)
  - Border getBorder()
  - void setForeground(Color)
  - void setBackground(Color)
  - Color getForeground()
  - Color getBackground()
  - void setOpaque(boolean)
  - boolean isOpaque()
  - void setFont(Font)
  - Font getFont()
  - void setCursor(Cursor)
  - Cursor getCursor()

JComponent

- metode za postavljanje i dobijanje stanja komponente:
  - void setComponentPopupMenu(JPopupMenu)
  - void setTransferHandler(TransferHandler)
  - TransferHandler getTransferHandler()
  - void setToolTipText(String)
  - void setName(String)
  - String getName()
  - boolean isShowing()
  - void setEnabled(boolean)
  - boolean isEnabled()
  - void setVisible(boolean)
  - boolean isVisible()

JComponent

- metode za event handling:
  - void addHierarchyListener(hierarchyListener l)
  - void removeHierarchyListener(hierarchyListener l)
  - void addMouseListener(MouseListener)
  - void removeMouseListener(MouseListener)
  - void addMouseMotionListener(MouseMotionListener)
  - void removeMouseMotionListener(MouseMotionListener)
  - void addKeyListener(KeyListener)
  - void removeKeyListener(KeyListener)
  - void addComponentListener(ComponentListener)
  - void removeComponentListener(ComponentListener)
  - boolean contains(int, int)
  - boolean contains(Point)
  - Component getComponentAt(int, int)
  - Component getComponentAt(Point)
  - Component setComponentZOrder(component comp, int index)
  - Component getComponentZOrder(component comp)

JComponent

- metode za iscrtavanje komponenti:
  - void repaint() – iz klase Component
  - void repaint(int, int, int, int)
  - void repaint(Rectangle)
  - void revalidate()
  - void paintComponent(Graphics)

JComponent

- metode za rad sa sadržavajućom hijerarhijom:
  - Component add(Component)
  - Component add(Component, int)
  - void add(Component, Object)
  - void remove(int)
  - void remove(Component)
  - void removeAll()
  - JRootPane getRootPane()
  - Container getTopLevelAncestor()
  - Container getParent()
  - int getComponentCount()
  - Component getComponent(int)
  - Component[] getComponents()
  - Component getComponentZOrder(int)
  - Component[] getComponentZOrder()

JComponent

- metode za prilagođavanje komponenti:
  - void setPreferredSize(Dimension)
  - void setMaximumSize(Dimension)
  - void setMinimumSize(Dimension)
  - Dimension getPreferredSize()
  - Dimension getMaximumSize()
  - Dimension getMinimumSize()
  - void setAlignmentX(float)
  - void setAlignmentY(float)
  - float getAlignmentX()
  - float getAlignmentY()
  - void setLayout(LayoutManager)
  - LayoutManager getLayout()
  - void applyComponentOrientation(ComponentOrientation)
  - void setComponentOrientation(ComponentOrientation)

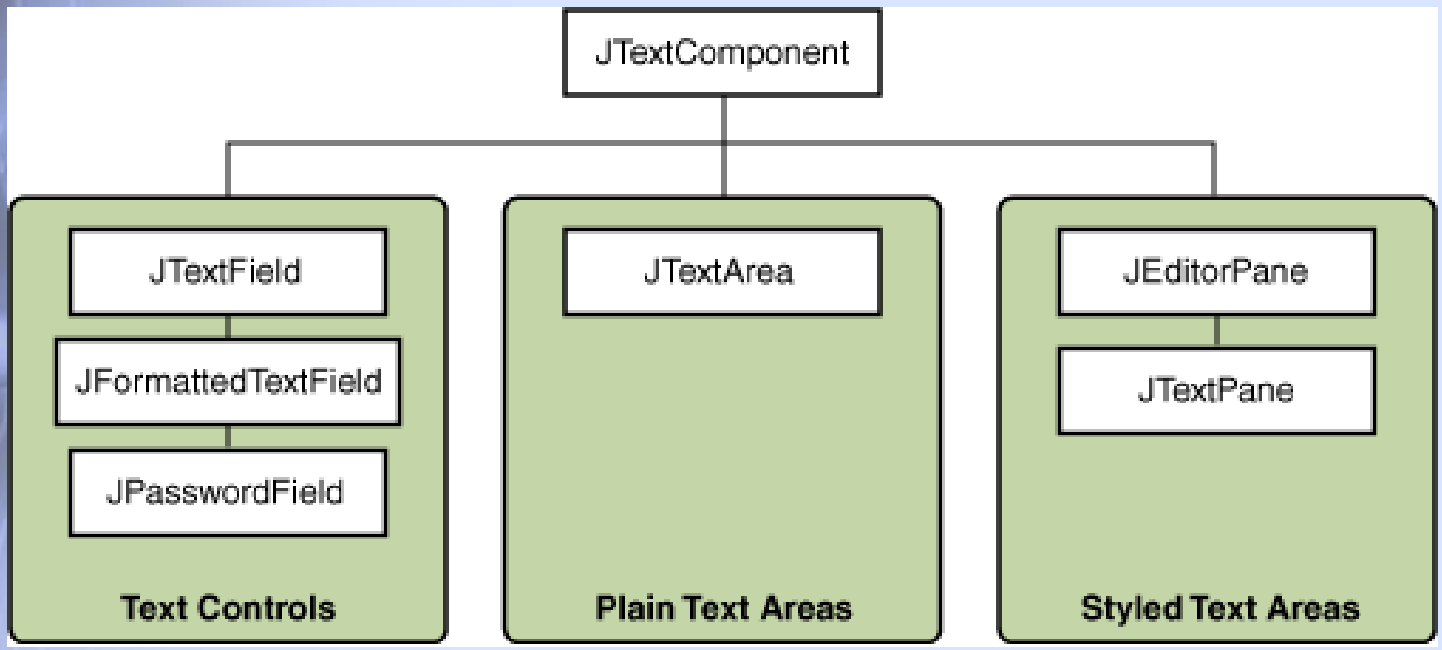
JComponent

- metode za dobijanje informacija o veličini i poziciji:
  - int getWidth()
  - int getHeight()
  - Dimension getSize()
  - Dimension getSize(Dimension)
  - int getX()
  - int getY()
  - Rectangle getBounds()
  - Rectangle getBounds(Rectangle)
  - Point getLocation()
  - Point getLocation(Point)
  - Point getLocationOnScreen()
  - Insets getInsets()

JComponent

- metode za specificiranje apsolutne veličine i pozicije:
  - void setLocation(int, int)
  - void setLocation(Point)
  - void setSize(int, int)
  - void setSize(Dimension)
  - void setBounds(int, int, int, int)
  - void setBounds(Rectangle)

Tekst komponente



JTextComponent klasa

- JTextComponent klasa obezbjeđuje klasama nasljednicama:
  - model – poznat kao dokument koji upravlja sadržajem komponente
  - pogled (view) – koji prikazuje komponente na ekranu
  - kontroler – poznat kao editor kit koji čita i upisuje tekst i implementira mogućnosti editovanja sa akcijama
  - podršku za undo i redo
  - podršku za caret change listeners-e i navigacione filtere

JTextComponent klasa

- metode za postavljanje atributa:
  - void setEditable(boolean)
  - boolean isEditable()
  - void setDragEnabled(boolean)
  - boolean getDragEnabled()
  - void setDisabledTextColor(Color)
  - Color getDisabledTextColor()
  - void setMargin(Insets)
  - Insets getMargin()

JTextComponent klasa

- metode za manipulaciju selektovanim tekstom:
  - String getSelectedText()
  - void selectAll()
  - void select(int, int)
  - void setSelectionStart(int)
  - void setSelectionEnd(int)
  - int getSelectionStart()
  - int getSelectionEnd()
  - void setSelectedTextColor(Color)
  - Color getSelectedTextColor()
  - void setSelectionColor(Color)
  - Color getSelectionColor()

Tekst komponente

- klase i metode za izmjenu teksta:
  - void cut()
  - void copy()
  - void paste()
  - void replaceSelection(String)
    (in JTextComponent)

  - EditorKit
  - DefaultEditorKit
  - StyledEditorKit
  - String xxxxAction
    (in DefaultEditorKit)

  - BeepAction
  - CopyAction
  - CutAction
  - DefaultKeyTypedAction
  - InsertBreakAction
  - InsertContentAction
  - InsertTabAction
  - PasteAction
    (in DefaultEditorKit)

Tekst komponente

- klase i metode za izmjenu teksta:
  - AlignmentAction
  - BoldAction
  - FontFamilyAction
  - FontSizeAction
  - ForegroundAction
  - ItalicAction
  - StyledTextAction
  - UnderlineAction
    (in StyledEditorKit)
  - Action[] getActions()
    (in JTextComponent)
  - InputMap getInputMap()
    (in JComponent)
  - void put(KeyStroke, Object)
    (in InputMap)

Tekst komponente

- klase i interfejsi koji predstavljaju dokumente:
  - Document
  - AbstractDocument
  - PlainDocument
  - StyledDocument
  - DefaultStyledDocument

Tekst komponente

- metode za rad sa dokumentima:
  - DocumentFilter
  - void setDocumentFilter(DocumentFilter)
    (in AbstractDocument)
  - void setDocument(Document)
  - Document getDocument()
    (in JTextComponent)
  - Document createDefaultModel()
    (in JTextField)
  - void addDocumentListener(DocumentListener)
  - void removeDocumentListener(DocumentListener)
    (in Document)
  - void addUndoableEditListener(UndoableEditListener)
  - void removeUndoableEditListener(UndoableEditlistener)
    (in Document)
  - int getLength()
  - Position getStartPosition()
  - Position getEndPosition()
  - String getText(int, int)
    (in Document)
  - Object getProperty(Object)
  - void putProperty(Object, Object)
    (in Document)
  - void setDocumentProperties(Dictionary)
  - Dictionary getDocumentProperties()
    (in AbstractDocument)

Tekst komponente

- metode za čitanje i upis teksta:
  - void read(Reader, Object)
  - void write(Writer)
    (in JTextComponent)
  - void read(Reader, Document, int)
  - void read(InputStream, Document, int)
    (in EditorKit)
  - void write(Writer, Document, int, int)
  - void write(OutputStream, Document, int, int)
    (in EditorKit)

Komponente

JButton	Dugme
JTextField	Jednolinijsko Edit polje
JTextArea	Višelinisjko Edit polje (Memo)
JLabel	Labela
JCheckBox	CheckBox
ButtonGroup	Kontejner za Radio Button-e, nije vidljiva komponenta
JRadioButton	Radio Button
JComboBox	ComboBox
JList	ListBox
JTabbedPane	Kartice
JOptionPane	MessageBox
JMenu	Meniji
JMenuItem	Stavka menija
JCheckBoxMenuItem	Stavka menija sa check box-om
JRadioButtonMenuItem	Stavka menija sa check radio button-om
JDialog	DialogBox
JFrame	Prozor
JToggleButton	Toggle button
JColorChooser	Paleta za izbor boje
JFileChooser	Dijalog za izbor datoteke
JPasswordField	Password polje
JProgressBar	Komponenta za prikaz napretka određene akcije
JTable	Tabela

Event Driven model

- korisnički interfejs kod GUI aplikacija je upravljan događajima
- pišu se metode (obrađivači događaja) koje se izvršavaju po pojavi nekog događaja korisničkog interfejsa - klik mišem, pritisak tastera i sl.
- program ima inicijalizacioni blok i blokove koda koji reaguju na događaje korisničkog interfejsa (obrađivače događaja)
- program se ne izvršava linearno (“od gore prema dole”), nego u određenim vremenskim intervalima:
  - kada se vrši pokretanje aplikacije (inicijalizacija)
  - reakcije na događaje (kada se vrši obrada događaja)

Event Driven model

- inicijalizacioni blok se u Java GUI programima izvršava počevši od metode main()
- događaji se opisuju xxxEvent klasama – za svaku vrstu događaja definisana je posebna klasa
- pritisak tastera – KeyEvent klasa
- pomjeranje miša – MouseEvent klasa
- sve xxxEvent klase nasljeđuju klasu Event (sličnost s izuzecima!!!)

Event Driven model

- svaka akcija nad komponentama korisničkog interfejsa izaziva generisanje objekata klasa nasljednica Event klase
- ti objekti se prosljeđuju objektima klasa nasljednica EventListener klase, koje “osluškuju” događaje
- “osluškivači” – instance neke od xxxListener klasa, npr.:
  - KeyListener – “osluškivač” za KeyEvent događaj
- EventListener-i uvedeni od verzije 1.1.

Osnovna struktura GUI aplikacije

- izvršavanje programa počinje sa main() metodom, kao i svaki drugi Java program
- najčešće se u main() metodi inicijalizuje i glavni prozor aplikacije, koji se potom i prikaže na ekranu

Osnovna struktura GUI aplikacije

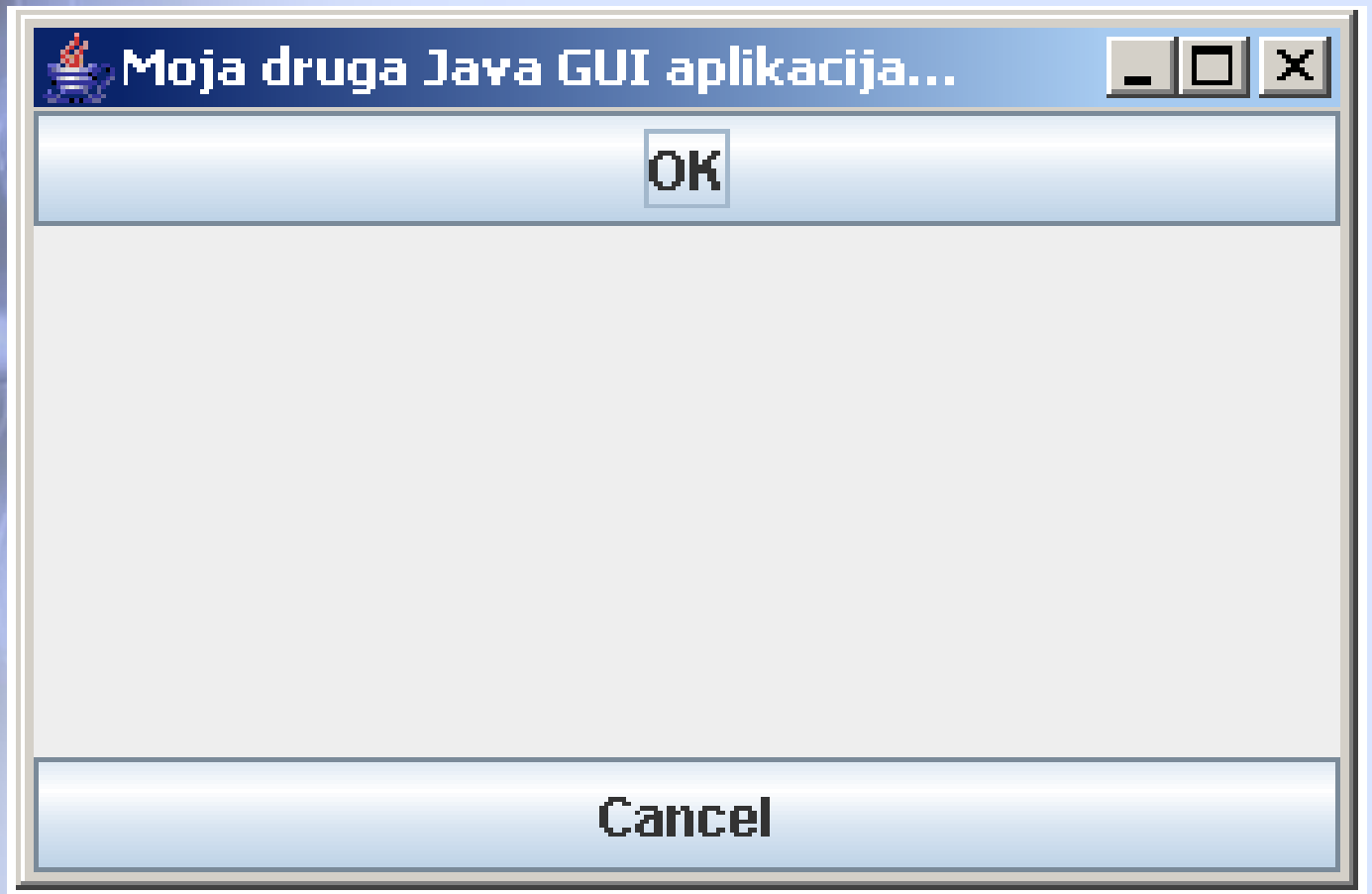
```
import javax.swing.*;
public class MainFrame extends JFrame {
  public MainFrame() {
    setSize(300, 200);
    setTitle("Moja prva Java GUI aplikacija…");
  }
}

public class MyApp {
  public static void main(String[] args) {
    MainFrame mf = new MainFrame();
    mf.setVisible(true);
  }
}
```

Dodavanje komponenti u prozor

- prilikom konstrukcije objekta klase naslednice JFrame dodaju se komponente na formu u određenom redosledu

```
import javax.swing.*;
public class MainFrame extends JFrame {
  public MainFrame() {
    setSize(300, 200);
    setTitle("Moja druga Java GUI aplikacija…");
   // dodajemo komponente na formu:
    getContentPane().add(bOK,
  BorderLayout.NORTH);
    getContentPane().add(bCancel,
  BorderLayout.SOUTH);
  }
 // Elementi na formi su najčešće privatni
  atributi klase
  private JButton bOK = new JButton("OK");
  private JButton bCancel = new
  JButton("Cancel");
```

Dodavanje komponenti u prozor

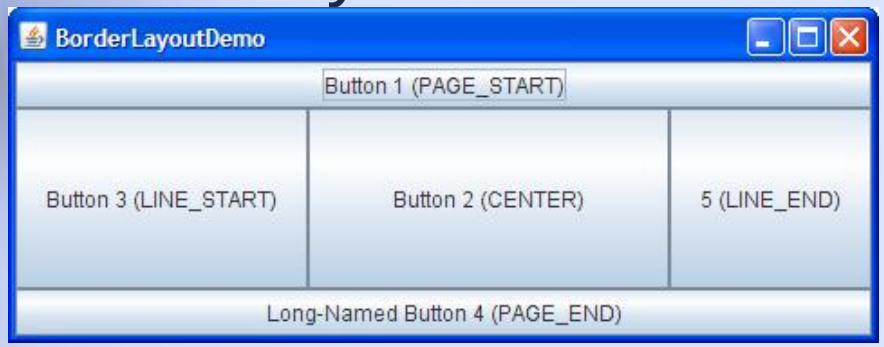


Prostorni raspored komponenti

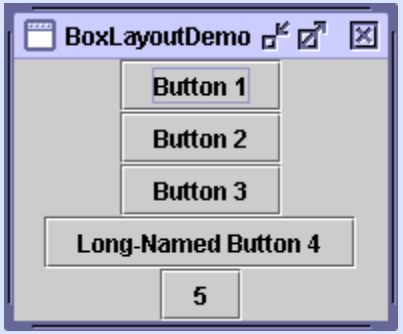
- koriste se layout manager-i
- instanca neke od xxxLayout klasa
- različiti layout manager-i iste komponente raspoređuju na različit način
- svaki kontejner ima sebi asociran layout manager
- standardni layout manager-i:
- BorderLayout, CardLayout, FlowLayout, GridLayout, GridBagLayout, BoxLayout, GroupLayout, SpringLayout
- korisnički definisani layout manager-i

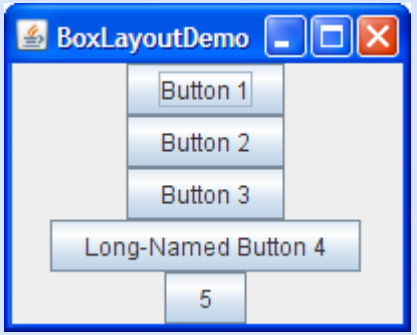
Prostorni raspored komponenti

- Layout manager-i se koriste kako u Swing tako i u AWT biblioteci (zato imamo import java.awt.*)
- BorderLayout:



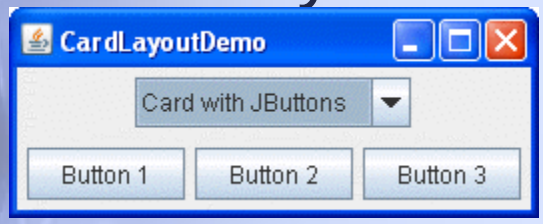
- BoxLayout:

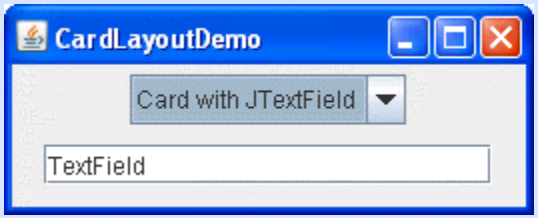




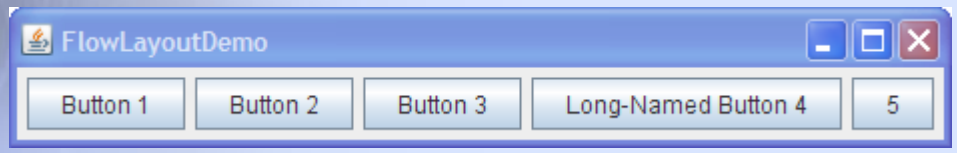
Prostorni raspored komponenti

- CardLayout:

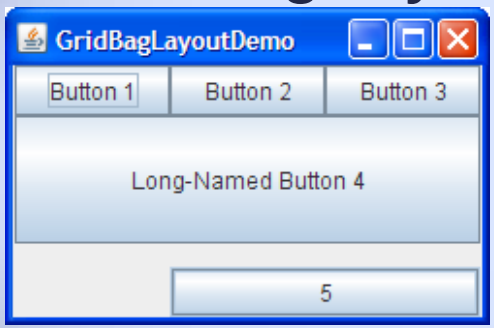


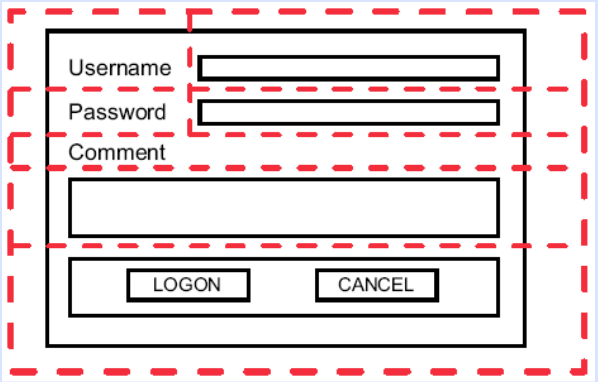


- FlowLayout:



- GridBagLayout:

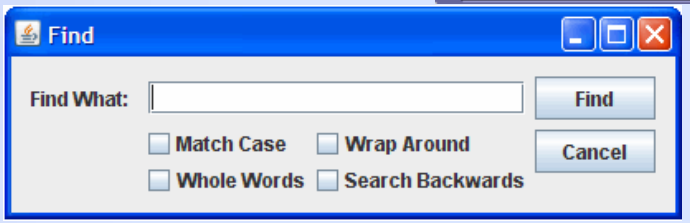




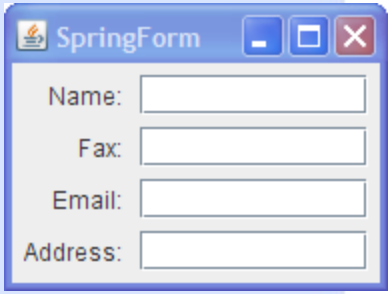
Prostorni raspored komponenti

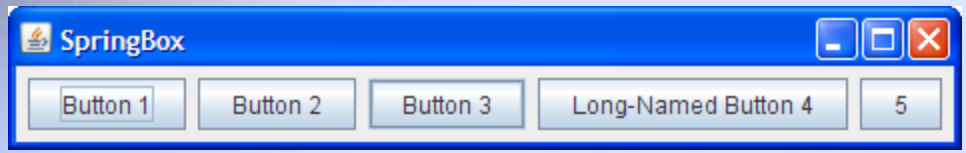
- GridLayout:

- Group Layout:



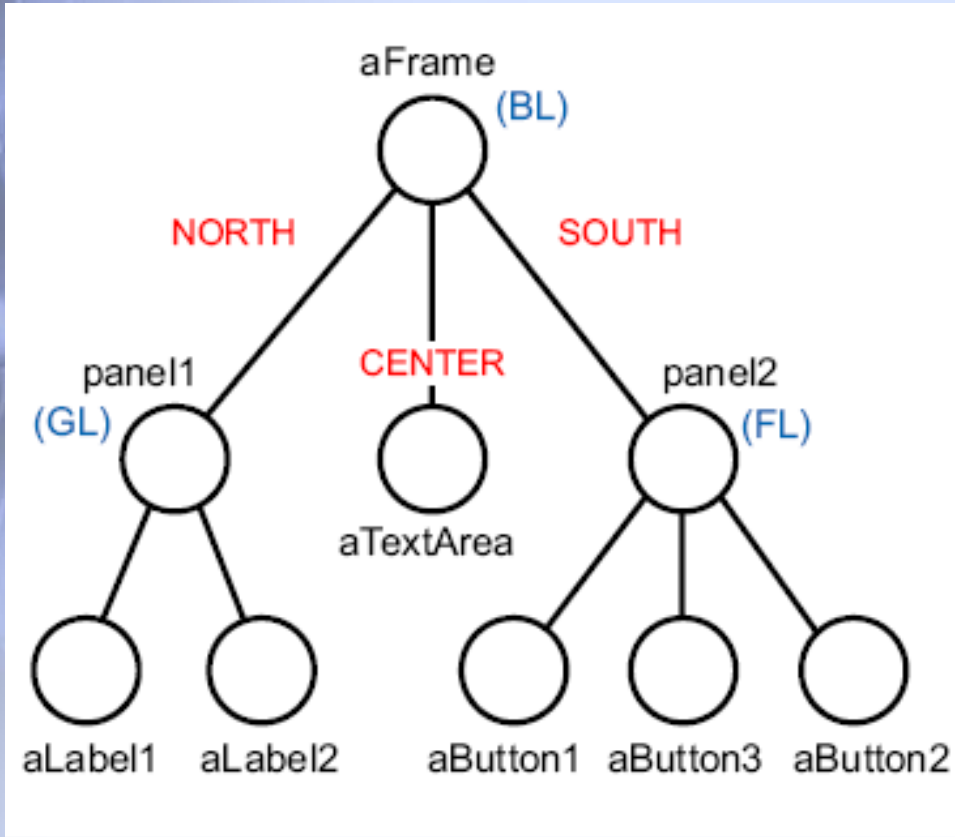
- SpringLayout:

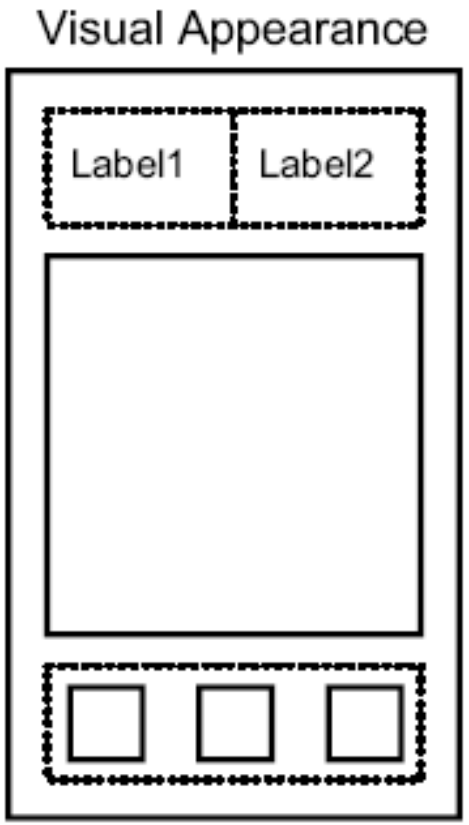




- osnovni skup layout manager-a ne obuhvata nijedan koji omogućava postavljanje komponente na tačno određeno mjesto na prozoru (mjesto određeno koordinatama u pikselima)
- *third/party*:
  - XYLayout
  - MIGLayout

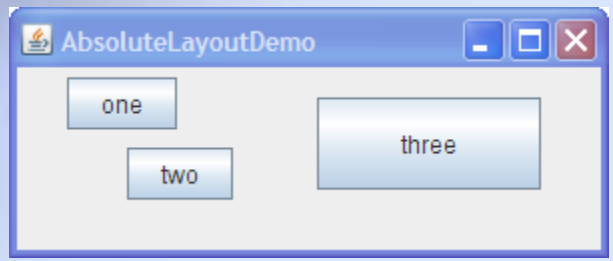
Prostorni raspored komponenti





Prostorni raspored komponenti

- raspored komponenti bez layout manager-a – moguće
- preporuka – koristiti layout manager kad god je to moguće
- kreiranje kontejnera bez layout manager-a uključuje sljedeće korake:
  - postaviti layout manager kontejnera na null pozivom setLayout(null)
  - pozvati metodu setBounds klase Component za svaku komponentu u kontejneru
  - pozvati metodu repaint klase Component



- kreiranje kontejnera bez layout manager-a može izazvati probleme, kao npr. pri promjeni veličine prozora

Look and Feel

- Look – odnosi se na pojavljivanje GUI widget-a (JComponent-i)
- Feel – odnosi se na način na koji se widget-i ponašaju
- platforme sa odgovarajućim look and feel-ovima:
  - Solaris, Linux sa GTK+ 2.2 ili kasnijim - GTK+
  - Solaris, Linux – Motif Classic Windows – Windows
  - Windows XP – Windows XP
  - Windows Vista – Windows Vista
  - Macintosh – Macintosh*
  - IBM UNIX – IBM*
  - HP UX – HP*

Look and Feel

- programsko kreiranje Look and Feel-a

```
public static void main(String[] args) {
      try {
        UIManager.setLookAndFeel(
        UIManager.getCrossPlatformLookAndFeelClassName());
      }catch (UnsupportedLookAndFeelException e)
      {
        // handle exception
      } catch (ClassNotFoundException e) {
        // handle exception
      } catch (InstantiationException e) {
        // handle exception
      } catch (IllegalAccessException e) {
        // handle exception
      }
      new SwingApplication();
  }
  -------
UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");
  ---------
java -Dswing.defaultlaf=com.sun.java.swing.plaf.gtk.GTKLookAndFeel MyApp
```

Rukovanje događajima

- događaji su predstavljeni objektima xxxEvent klasa, a osluškivači su odgovarajući xxxListener objekti
- xxxListener-i su interfejsi => “osluškivač” mora biti instanca neke klase koja implementira odgovarajući interfejs

```
import java.awt.*;
import java.awt.event.*;

public class MyListener implements
  ActionListener {
  public void actionPerformed(ActionEvent
  ev) {
    System.exit(0);
  }
}
```

Dodavanje listener-a

- komponenta koja obrađuje neki događaj se prijavljuje pozivom addxxxListener

```
public class MainFrame extends JFrame {
  public MainFrame() {
    ...
    getContentPane().add(bOK);
    getContentPane().add(bCancel);
    // dodajemo reakcije na događaje dugmadima
    bOK.addActionListener(new MyListener());
    bCancel.addActionListener(new MyListener());
  }

  // Elementi na formi su najčešće privatni
  atributi klase
  private JButton bOK = new JButton("OK");
  private JButton bCancel = new
  JButton("Cancel");
}
```

Osluškivači kao unutrašnje klase

- prozori mogu biti pretrpani komponentama koje obrađuju više događaja – definicija velikog broja Listener klasa – nepregledan program
- varijanta 2:

```
JButton b = new JButton("Pritisni me");
ActionListener al = new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    ...
  }
 b.addActionListener(al);
```

Ime klase koja implementira interfejs nigdje nije navedeno !!!

- varijanta 3: // najčešća – koristi je Jbuilder, Eclipse VE i dr.

```
JButton b = new JButton("Pritisni me");
b.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    ...
  });
```

Reakcije na događaje

Tip događaja	Dodijeljeni Listener
ActionEvent	ActionListener
AdjustmentEvent	AdjustmentListener
AncestorEvent	AncestorListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener
KeyEvent	KeyListener
HierarchyEvent	HierarchyListener
MouseEvent	MouseListener
WindowEvent	WindowListener
ItemEvent	ItemListener
TextEvent	TextListener

Osluškivači, događaji i metode

Listener Interface	Event	Listener Methods
ActionListener	ActionEvent	actionPerformed()
AdjustmentListener	AdjustmentEvent	adjustmentValueChanged()
ComponentListener	ComponentEvent	componentHidden() componentMoved() componentResized() componentShown()
ContainerListener	ContainerEvent	componentAdded() componentRemoved()
FocusListener	FocusEvent	focusGained() focusLost()

Osluškivači, događaji i metode

Listener Interface	Event	Listener Methods
KeyListener	KeyEvent	keyPressed() keyReleased() keyTyped()
MouseListener	MouseEvent	mouseClicked() mouseEntered() mouseExited() mousePressed() mouseReleased()
MouseMotionListener	MouseEvent	mouseDragged() mouseMoved()

Korisnički definisane komponente

- koriste se u situacijama kada standardne komponente ne mogu da obezbjede zadovoljavajuću funkcionalnost
- konstruišu se nasljeđivanjem neke od standardnih komponenti korisničkog interfejsa
- za vizuelne komponente potrebno je redefinisati paint() metodu za iscrtavanje
- ako nijedna komponenta nije adekvatna za nasljeđivanje, može se koristiti generička komponenta JComponent

JavaBeans

- standard za kreiranje softverskih komponenti koje imaju svoje osobine i ponašanje i koje se mogu koristiti u okviru RAD alata
- svaka JavaBean komponenta ima svojstva (properties) i događaje (events)
- JavaBean je svaka Java klasa za koju važi:
  - da ima podrazumijevani konstruktor
  - za svaki property xxx moraju da postoje public metode setXxx() i getXxx();
  - Za svaki event xxxEvent moraju da postoje public metode addXxxListener(XxxListener) i removeXxxListener(XxxListener)

Java apleti

- Java programi namjenjeni za ugrađivanje u HTML stranice, tj. Java programi koji se izvršavaju unutar Web browsera
- u okviru stranice dobija određenu pravougaonu površinu – dimenzije u pikselima – slično slikama
- web čitač pristupa stranici koja sadrži aplet – automatski se preuzima i programski kod apleta, pokreće JVM i počinje izvršavati aplet
- aplet ima na raspolaganju gotovo sve mogućnosti klasičnih Java aplikacija, osim 2 bitna ograničenja uvedena iz bezbjednosnih razloga

Java apleti

- ograničenja:
  - apleti ne mogu pristupati fajl sistemu računara na kome se izvršavaju
  - apleti ne mogu uspostaviti mrežnu konekciju sa bilo kojim računarom, osim sa Web serverom sa koga su preuzeti
- aplet je klasa koja nasljeđuje klasu Applet (AWT) ili JApplet (Swing)
- pisanje apleta – nasljeđivanje klase JApplet i redefinisanje odgovarajućih metoda
- Najvažnije metode:
  - init()
  - start()
  - stop()
  - paint()
  - destroy()

Java aplet – primjer

AppletTest.java

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

public class AppletTest extends JApplet {
  public void init() {
    getContentPane().add(new
  JLabel("Applet!"));
  }
}
```

HTML stranica sa apletom

- AppletTest.html

```
...
<applet code = "AppletTest”
  width = 200 height = 200>
</applet>
...
```

standardna HTML oznaka; koristi se browserova ugrađena JVM

- index.html

```
...
<OBJECT
  codebase="http://java.sun.com/update/1.6.0
  /jinstall-6-windows-i586.cab"
  classid="clsid:5852F5ED-8BF4-11D4-A245-
  0080C6F74284" height=400 width=400>
...
</OBJECT>
```

poziv Java 1.6 Plug-In dodatka

Java aplet – primjer

```
JButtonTest.java
import java.awt.*;
import javax.swing.*;
public class JButtonTest extends JApplet {
  JButton b;
  public void init() {
    b = new JButton("Pritisni me");
    Container cp = getContentPane();
    cp.add(b);
  }
}
```

- postavljanje komponenti na površinu apleta ne razlikuje se ni po čemu od postavljanja komponenti na prozor klasične aplikacije

Aplet i aplikacija – 2 in 1

- aplet – osnovna klasa mora naslijediti Japplet klasu
- aplikacija – mora imati main()
- 2 in 1:
  - osnovna klasa nasljeđuje JApplet klasu i sadrži main() metodu

Java Web Start

- tehnologija namijenjena pokretanju Java aplikacija preko računarske mreže, prvenstveno Interneta
- korištenjem ove tehnologije moguće je preuzeti aplikaciju sa udaljene lokacije i automatski je pokrenuti, zaobilazeći proces instalacije aplikacije
  - ovako preuzeta aplikacija se izvršava na lokalnoj JVM
- JWS aplikacija se obično pokreće putem odgovarajućeg linka ugrađenog u web stranicu
  - ovaj link pokazuje na JNLP (*Java Network Launch Protocol*) datoteku koja daje instrukcije JWS softveru da preuzme i pokrene aplikaciju sa udaljene lokacije

Java Web Start

- prednosti u pogledu korištenja ovog tipa aplikacija. U ove prednosti se ubrajaju:
  - mogućnost smještanja Java aplikacije na web server odakle može biti preuzeta i pokrenuta na različitim platformama
  - JWS softver podržava višestruke verzije JRE-a. JWS može preuzeti i instalirati odgovarajuću verziju JRE-a, ako je to neophodno za izvršavanje aplikacije.
  - mogućnost kreiranja prečice na fajl sistemu računara, kojom se pokreće JWS aplikacija
  - JWS aplikacija se izvršava unutar lokalne JVM sa odgovarajućim sigurnosnim ograničenjima, kao što su ograničen pristup fajl sistemu računara na kome se izvršavaju i ograničen pristup mrežnim resursima.
  - preuzete JWS aplikacije se lokalno keširaju. Na ovaj način poboljšavaju se performanse.
  - nove verzije JWS aplikacije se automatski preuzimaju pri pokretanju aplikacije.
  - JWS softver neophodan za izvršavanje JWS aplikacija ugrađen je u JRE, tako da korisnici ne moraju preduzimati bilo kakve dodatne akcije kako bi mogli izvršavati JWS aplikacije.

Java Web Start

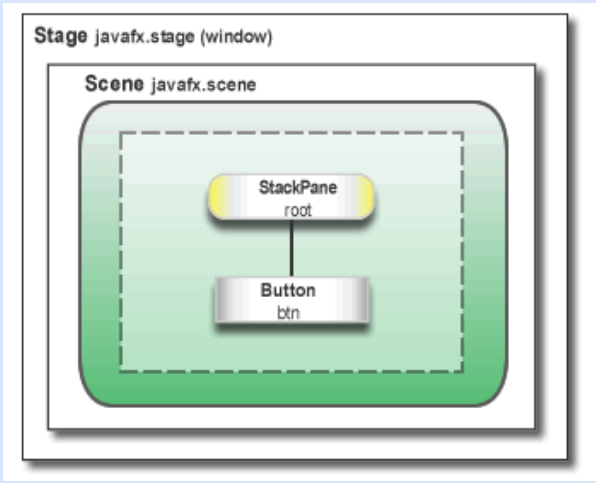
- razvoj i pokretanje JWS aplikacije:
  - razvija na isti način kao i druge GUI aplikacije
  - kompajlirani kod se smješta u JAR arhivu
  - JWS aplikacija se pokreće korištenjem JNLP protokola
  - preuzima se sa web servera
  - mora biti opisana u odgovarajućoj JNLP datoteci
    - JNLP datoteka je XML bazirana datoteka

Java Web Start

```
<?xml version="1.0" encoding="UTF-8"?>
<jnlp spec="1.0+" codebase="http://127.0.0.1:8080/ETF_JAVA_12_JWS/"
   href="visual.jnlp">
   <information>
      <title>Visual</title>
      <vendor>ETF BL</vendor>
   </information>
   <resources>
      <j2se version="1.6+"
        href="http://java.sun.com/products/autodl/j2se"/>
      <jar href="visual.jar" main="true" />
   </resources>
   <application-desc main-class="net.etfbl.visual.MyApp">
   </application-desc>
   <update check="background"/>
</jnlp>
```

JavaFX

- Osnovna struktura JavaFX aplikacije
  - Osnovna klasa JavaFX aplikacije nasljeđuje javafx.application.Application klasu. Metoda start() je osnvni „entry point“ za sve JavaFX aplikacije.
  - JavaFX aplikaciji definiše UI kontejner koristeći *stage* i *scene. Stage* klasa je top-level JavaFX kontejner. JavaFX *Scene* klasa je kontejer za sav sadržaj.
  - Sadržaj scene je predstavljen kao graf



  - Metoda main() nije potrebna za JavaFX aplikacije u slučaju kada je JAR datoteka kreirana pomoću JavaFX Packager alata koji ugrađuje JavaFX Launcher u JAR datoteku. Korisno ju imati main() metodu kako bi se JAR datoteke mogle pokretati bez JavaFX Launcher-a.