

# Software Ingeniaritza



## 4. Gaia: Inplementazioa

### 4.1 Interfaze grafikoak: SWING/AWT

**A. Goñi, J. Ibáñez, J. Iturrioz, J.A. Vadillo**



informatika  
fakultatea



facultad de  
informática



Universidad  
del País Vasco

eman ta zabal zazu  
Euskal Herriko  
Unibertsitatea

# Aurkibidea

- Sarrera
  - Helburuak
  - AWT klase hierarkia: COMPOSITE diseinu patroia
- Osagai nagusiak
  - AWT/Swing edukitzaileak
  - AWT/Swing osagai nagusiak
  - Diseinuaren kudeatzaileak
- Gertaeren kudeaketa
  - Behe-mailako eta goi-mailako gertaerak
  - Listener-ak

# Sarrera

- Erabiltzailearen interfaze grafikoak (GUI) eraikitzeko baliabideak behar ditugu.
- Java-k aukera ematen du lortzeko:
  - AWT klaseak (Abstract Window Toolkit).
  - Swing klaseak.
    - Swing AWTren ondokoa da eta osagai gehiago ditu.
  - Interfazeen diseinu eta programazio erraza eta azkarra.
  - Web-erako Applet-en diseinua.

# Helburuak

- Erabiltzailearen interfazeak eraikitzeko Java-n diseinatutako klaseen hierarkia ulertzea.
- Gertaeren kudeaketa nola egiten den ulertzea.
- Ez da gure helburua:  
klase, listener, etab... guztien izenak ikastea.
  - Interfazeak Javaren garapenerako tresnak erabiliz eraiki daitezke (adib: Eclipse, NetBeans).

# AWT klase hierarkia

Elementu Sinpleak

Label

Button

CheckBox

ScrollBar

Choice

List

Edukiontziaik(Container)

Panel

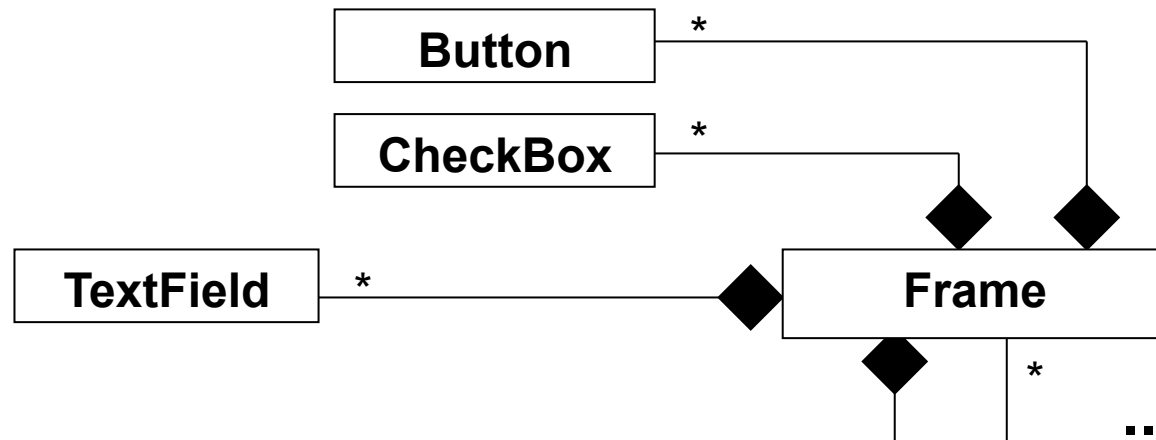
Frame

Dialog

Applet

Nola antolatu edukiontziaik eta elementu sinpleak?

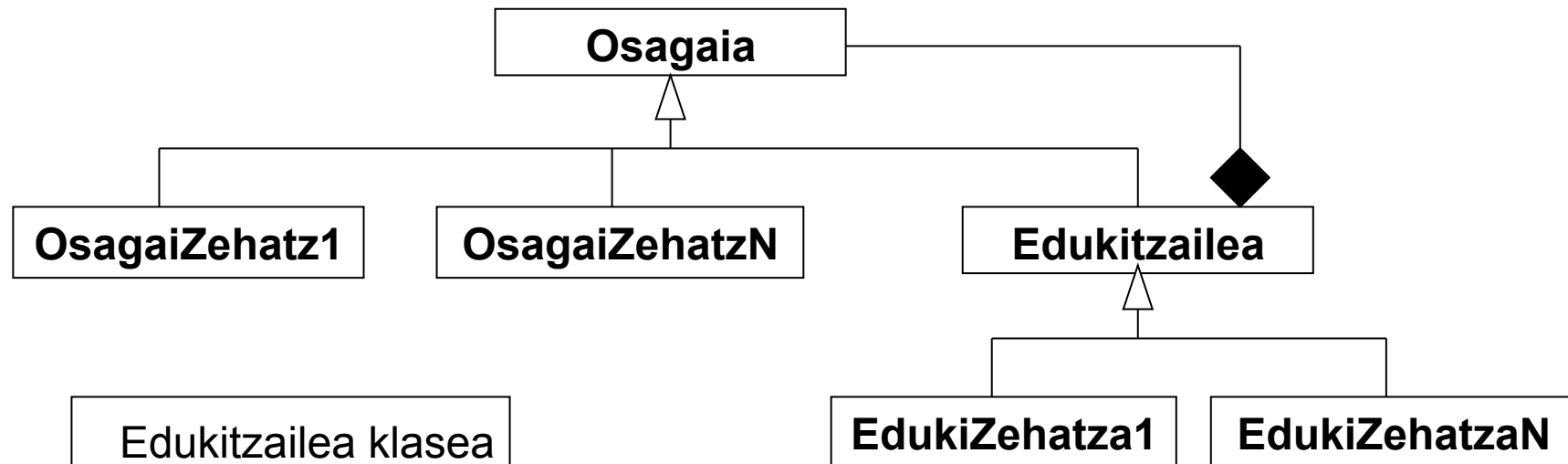
# Antolaketa desegokia



## Desabantailak:

- Frame klasean `addButton`, `addCheckBox`, `addTextField`, `AddFrame`,... metodoak beharko lirateke.
- Gainera diagrama ez dago batere osatua zeren eta **Button** izan daiteke **Panel**, **Dialog**,... baten osagaia.
- XXX osagai berria gehitu nahiko balitz, **Frame** klasea aldatu beharko litzateke eta `addXXX` metodoa gehitu (ebazpen hau ez litzateke batere hedagarria izango).

# Klase hierarkia: COMPOSITE diseinu patroia



Edukitzailea klasea
add (Osagaia o)
--add metodoak edukitzaileari --osagaiak gehitzen dizkio

**Edukitzaile** bat hainbat **osagaiez** osatuta dago, hauek osagai zehatzak edo edukitzaileak izanik. Edukitzaile hauek aldi berean hainbat osagai izan ditzakete.

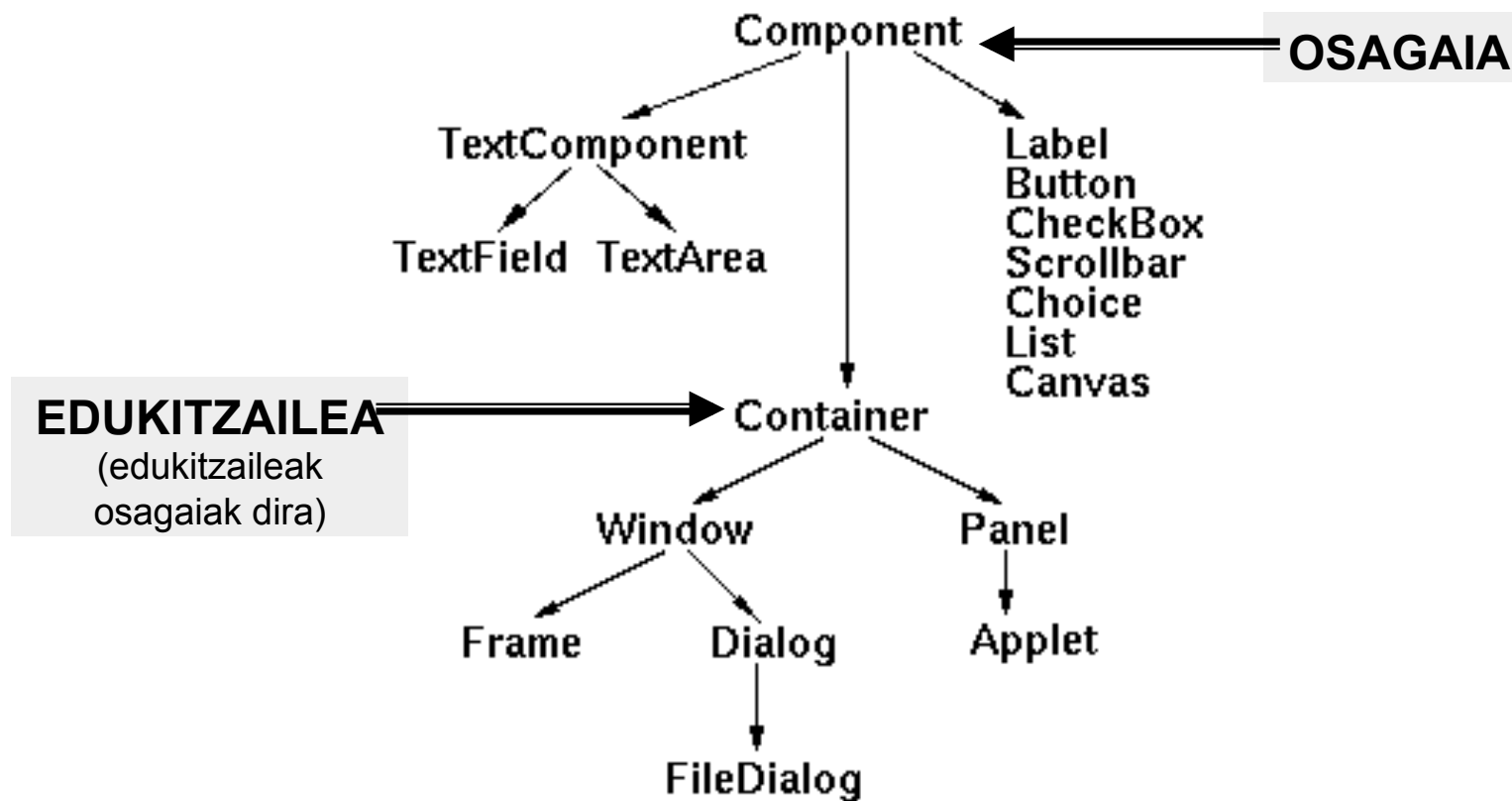
# Aurkibidea

- Sarrera
  - Helburuak
  - AWT klase hierarkia: COMPOSITE diseinu patroia
- Osagai nagusiak
  - AWT/Swing edukitzaileak
  - AWT/Swing osagai nagusiak
  - Diseinuaren kudeatzaileak
- Gertaeren kudeaketa
  - Behe-mailako eta goi-mailako gertaerak
  - Listener-ak

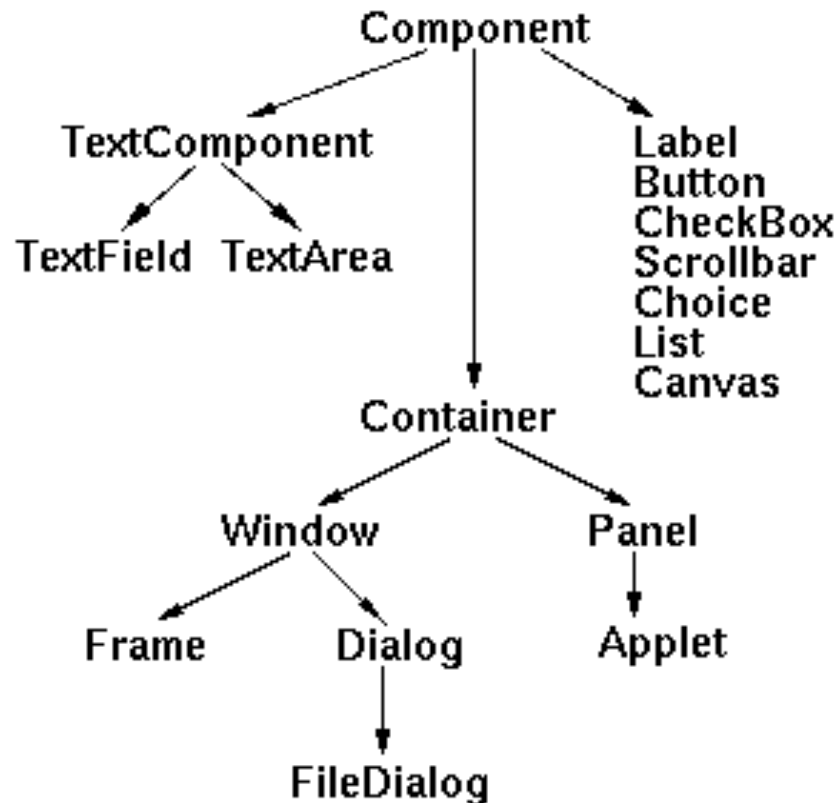


# AWT: osagai nagusiak

- AWT klaseen hierarkia



# AWT klase hierarkia: COMPOSITE diseinu patroia jarraitzen du

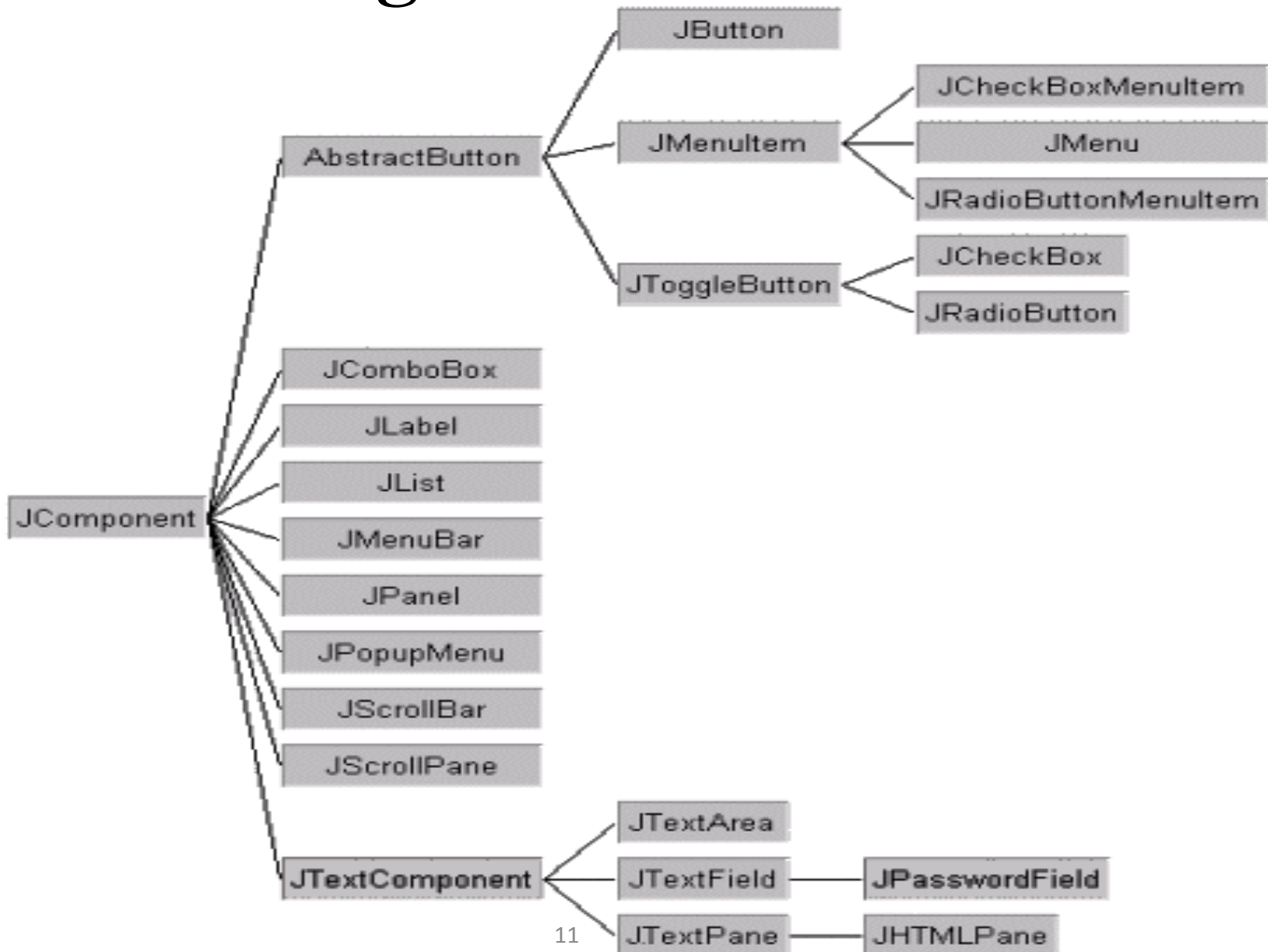


Aukera infinitoak dira:

- Leiho bat 2 testu-koadro, 2 testu-area, 3 botoi, eta panel bat dituen 5 kontrol-kutxa eta goitibehera-koadro bat dituen.
- Leiho bat 2 etiketa, 2 testu-area eta botoi bat dituen.
- ...

Gainera, edukitzaile eta osagai mota berriak gehitzea ez litzateke oso garestia izango (Ebazpen hedagarria da).

# Swing hierarkia



# AWT/Swing: edukitzaileak

Edukitzaileak {  
• **Frame/JFrame** ← garrantzitsuena  
• **Panel/JPanel**  
• **Dialog/JDialog**

- **Frame/JFrame** klaseak
  - Leiho laua, maximizatu, minimizatu eta itxi ikonoak eskeintzen dituena.
  - Menuak sartu daitezken edukitzaile bakarra da.
- **Panel/JPanel** klaseak
  - Osagaiak elkartzeko edukitzaile generikoak.
  - Edukitzaileak beste edukitzaile batzuen barnean sartzeko erabiltzen den klasea (panelen barruan azpi-panelak).
  - Swing-eko JFrame klasearen barnean JPanel bat sartzen da modu automatikoan. AWT<sup>1,2</sup>-klasean ez.

# AWT/Swing: edukitzaileak

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

public class FrameaPanelekin extends JFrame
{
    // ...
    // Hemen erazagutu genuke Framearen
    // osagaiak: menu-barra, File aukera, ...

    public FrameaPanelekin() {
        this.getContentPane().setLayout(null);
        this.setSize(new Dimension(400, 300));
        this.setTitle("Panela duen Frame baten adibidea");
        this.setBackground(Color.black);
    }

    public static void main(String[] args) {
        JFrame frame = new FrameaPanelekin();
        frame.setVisible(true);
    }
}
```



JFramereri asoziatutako JPanel-a atzitzeko

# **AWT/Swing: osagai nagusiak**

- **TextField/JTextField** klaseak
  - Lerro bateko testu sarrerarako eremua.
- **TextArea/JTextArea** klaseak
  - Testu lerro anitzak sartzeko.
  - Testua erakusteko ere erabili daiteke.
  - TextArea-k testu-eremu bakoitzerako scroll bertikal eta horizontal bat inplementatzen du.
  - JTextArea-k ez du scroll-ik automatikoki gehitzen. Horretarako JTextArea-ri JScrollPane bat gehitu behar zaio.

# AWT/Swing: osagai nagusiak

```
public class TestuEremuak extends JFrame {
    private JPanel jContentPane=new JPanel();
    private JLabel jLabel1=new JLabel();
    private JTextField jTextField1=new JTextField();
    private JLabel jLabel2=new JLabel();
    private JTextField jTextField2=new JTextField();

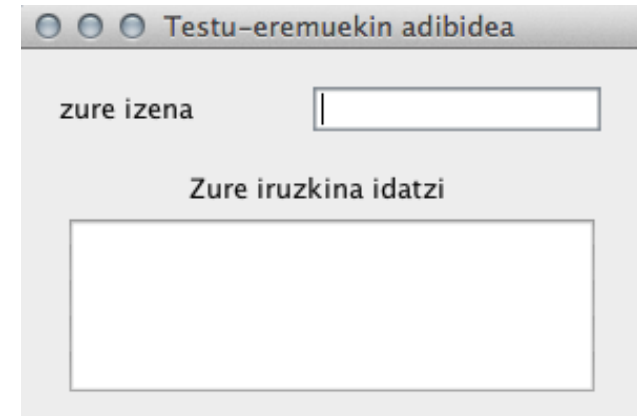
    public TestuEremuak() {
        this.setSize(300, 200);
        this.setTitle("Testu-eremuekin adibidea");
        jLabel1.setBounds(new Rectangle(15, 19, 100, 19));
        jLabel1.setText(" zure izena");
        jLabel2.setBounds(new Rectangle(81, 57, 138, 19));
        jLabel2.setText("Zure iruzkina idatzi");

        this.setContentPane(jContentPane);
        jContentPane.setLayout(null);

        jContentPane.add(jLabel1, null);
        jTextField1.setBounds(new Rectangle(137, 18, 143, 21));
        jContentPane.add(jTextField1, null);

        jContentPane.add(jLabel2, null);
        jTextField2.setBounds(new Rectangle(21, 78, 256, 88));
        jContentPane.add(jTextField2, null);
        this.setContentPane(jContentPane);
    }

    public static void main(String[] args) {
        TestuEremuak framea = new TestuEremuak();
        framea.setVisible(true);
    }
}
```



# *AWT/Swing: osagai nagusiak*

- **Button/JButton** klaseak
  - Botoiak sortzeko.
  - Botoi bat sakatzean ekintzaren bat egikarituko da.
- **Label/JLabel** klaseak
  - Informazioa azaltzeko.
  - Testu-eremuen ondoan erabiltzen dira.



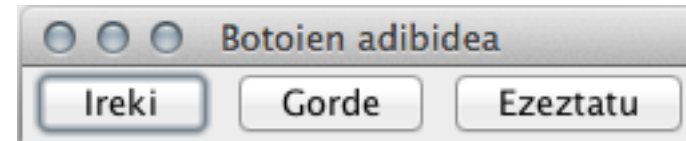
# AWT/Swing: osagai nagusiak

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

public class Botoiak extends JFrame
{
    JButton jButton1 = new JButton();
    JButton jButton2 = new JButton();
    JButton jButton3 = new JButton();

    public Botoiak() {
        this.setTitle("Botoien adibidea");
        jButton1.setText("Ireki");
        jButton2.setText("Gorde");
        jButton3.setText("Ezeztatu");
        this.getContentPane().add(jButton3, BorderLayout.EAST);
        this.getContentPane().add(jButton2, BorderLayout.CENTER);
        this.getContentPane().add(jButton1, BorderLayout.WEST);
        pack();
    }

    public static void main(String[] args) {
        JFrame frame = new Botoiak();
        frame.setVisible(true);
    }
}
```



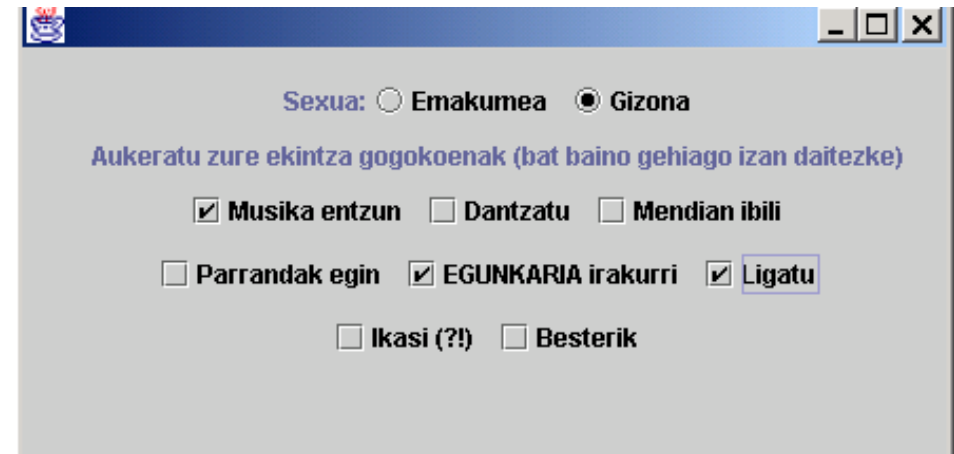
# AWT/Swing: osagai nagusiak

- **CheckBox/JCheckBox/JRadioButton** klaseak
  - Kontrol-laukiek aukerak aktibatzeke eta desaktibatzeke eskeintzen dute.
  - JRadioButton gehituko dira ButtonGroup batean, **aukera bakar bat** hautatu behar denean (ButtonGroup ez da osagai bisuala).
  - AWT-n gauza berdina lortzeko CheckBox-ak CheckboxGroup batean definitu beharko dira.

# AWT/Swing: osagai nagusiak

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

public class Aukerak extends JFrame{
    JPanel jPanel1 = new JPanel();
    JLabel jLabel1 = new JLabel();
    JRadioButton jButton1 = new JRadioButton();
    JRadioButton jButton2 = new JRadioButton();
    ButtonGroup g = new ButtonGroup();
    JLabel jLabel2 = new JLabel();
    JCheckBox jCheckBox1 = new JCheckBox();
    JCheckBox jCheckBox2 = new JCheckBox(); //Besteak eraiki
    public Aukerak() {
        jLabel1.setText("Sexua:");
        jButton1.setText("Gizona");
        jButton2.setText("Emakumea");
        jLabel2.setText("Aukeratu zure ekintza gogokoenak (bat baino gehiago izan daitezke)");
        jCheckBox1.setText("Musika entzun");
        jCheckBox2.setText("Dantzatu");//Besteenak testuak gehitu
        jPanel1.add(jLabel1, null);
        jPanel1.add(jButton2, null);
        jPanel1.add(jButton1, null);
        jPanel1.add(jLabel2, null);
        jPanel1.add(jCheckBox1, null); //Beste JCheckBox-ak gehitu
        g.add(jButton1);
        g.add(jButton2);
        this.getContentPane().add(jPanel1, null);
    }
}
```

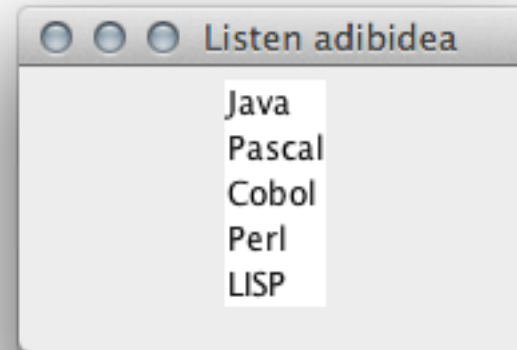


# AWT/Swing: osagai nagusiak

- **List/JList** klaseak
  - Goitibehera zerrenda bat desplegatzen duen widget bat da. Erabiltzaileak elementu bat aukeratu dezake zerrendatik.
  - AWT-ren List-ean scroll bat dago, agertzen diren listako elementuak List-ari esleitutako espazioan sartzen ez direnerako.
  - Swing-eko JList-ek ez du scroll-ik. Horretarako JScrollPane batean gehitu beharko da.

# AWT/Swing: osagai nagusiak

```
import javax.swing.*;
import java.awt.*;
import java.util.*;
public class Listak extends JFrame {
    JList jList1; //new gero egingo da
    Vector elementuak = new Vector();
    JPanel jPanel1 = new JPanel();
    public Listak(){
        this.setTitle("Listen adibidea");
        elementuak.addElement("Java");
        elementuak.addElement("Pascal");
        elementuak.addElement("Cobol");
        elementuak.addElement("Perl");
        jList1 = new JList(elementuak);
        jPanel1.add(jList1, null);
        this.getContentPane().add(jPanel1, null);
        elementuak.addElement("LISP"); //elementua
        aurkezten da oraindik framea aurkeztu ez delako
    }
    public static void main(String[] args){
        JFrame frame = new Listak();
        frame.setVisible(true);
    }
}
```



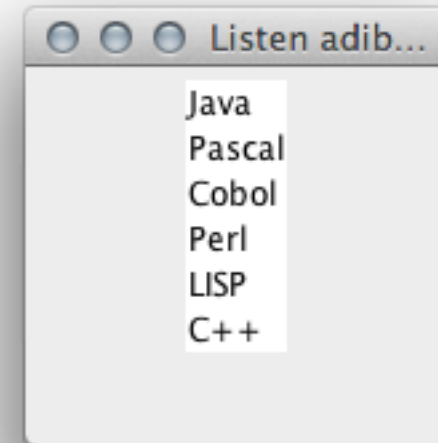
**frame.elementuak.addElement("C++");**  
ez da aurkeztuko

**Vector-ek ez du egitura "dinamikoa" kudeatzen.**

# AWT/Swing: osagai nagusiak

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

public class Listak extends JFrame {
    JList jList1; //new gero egingo da
    DefaultListModel elementuak = new DefaultListModel();
    JPanel jPanel1 = new JPanel();
    public Listak(){
        this.setTitle("Listen adibidea");
        elementuak.addElement("Java");
        elementuak.addElement("Pascal");
        elementuak.addElement("Cobol");
        elementuak.addElement("Perl");
        jList1 = new JList(elementuak);
        jPanel1.add(jList1, null);
        this.getContentPane().add(jPanel1, null);
        elementuak.addElement("LISP"); //Vector-en
            gehitze, JLIST-a aldatzen da!!
    }
    public static void main(String[] args){
        JFrame frame = new Listak();
        frame.setVisible(true);
        frame.elementuak.addElement("C++");
        frame.setVisible(true);
    }
}
```

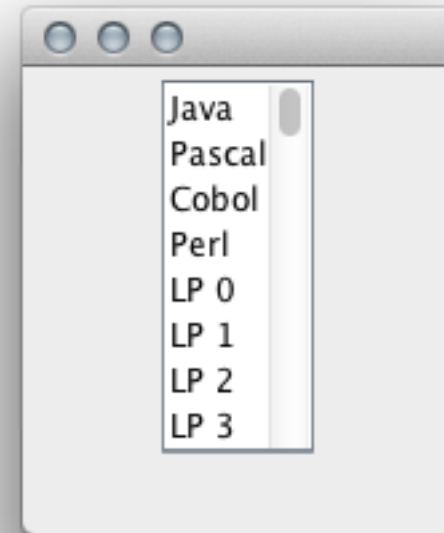


**javax.Swing DefaultListModel  
klasea Vector klasearen metodo  
berdinak eskaintzen ditu.**

**DefaultListModel erabili zerrenda "dinamikoa" denean**

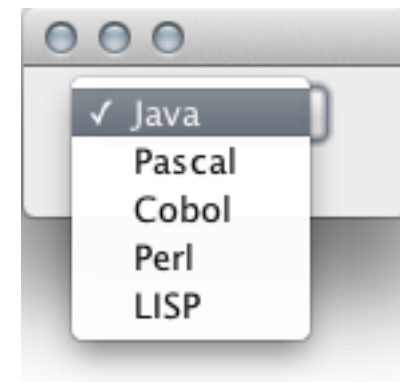
# AWT/Swing: osagai nagusiak

```
import javax.swing.*;
import java.awt.*;
import java.util.*;
public class ListakScrollekin extends JFrame {
    JPanel jPanel1 = new JPanel();
    JList jList1;
    Vector elementuak = new Vector();
    public ListakScrollekin() {
        this.getContentPane().add(jPanel1, null);
        elementuak.addElement("Java");
        elementuak.addElement("Pascal");
        elementuak.addElement("Cobol");
        elementuak.addElement("Perl");
        jList1 = new JList(elementuak);
        JScrollPane j = new JScrollPane(jList1);
        //Lista sartzen dugu scroll-a duen panel batean
        jPanel1.add(j,null);
        for (int i=0;i<50;i++) elementuak.addElement("LP " +
        i);
        pack();
    }
    public static void main(String[] args) {
        JFrame frame = new ListakScrollekin();
        frame.setVisible(true);
    }
}
```



# AWT/Swing: osagai nagusiak

- **Choice/JComboBox** klaseak
  - Aukeren menuak sortzeko.
  - Honen abantaila goitibeherako-menuak ez dute espazioa asko okupatzen pantailan.

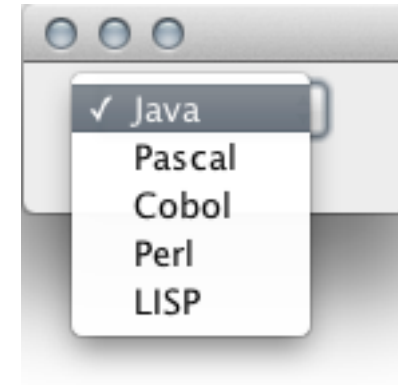




# AWT/Swing: osagai nagusiak

```
import javax.swing.*;
import java.awt.*;
import java.util.*;
public class ComboBoxak extends JFrame {
    DefaultComboBoxModel elementuak = new DefaultComboBoxModel ();
    JPanel jPanel1 = new JPanel();
    JComboBox jComboBox1;//new gero egingo da

    public ComboBoxak(){
        elementuak.addElement("Java");
        elementuak.addElement("Pascal");
        elementuak.addElement("Cobol");
        elementuak.addElement("Perl");
        jComboBox1 = new JComboBox(elementuak);
        jPanel1.add(jComboBox1, null);
        this.getContentPane().add(jPanel1, null);
        elementuak.addElement("LISP");
        pack();
    }
    public static void main(String[] args){
        Frame frame = new ComboBoxak();
        frame.setVisible(true);
    }
}
```



# AWT/Swing: osagai nagusiak

- **Menuen sorkuntza**

- **Swing-en**

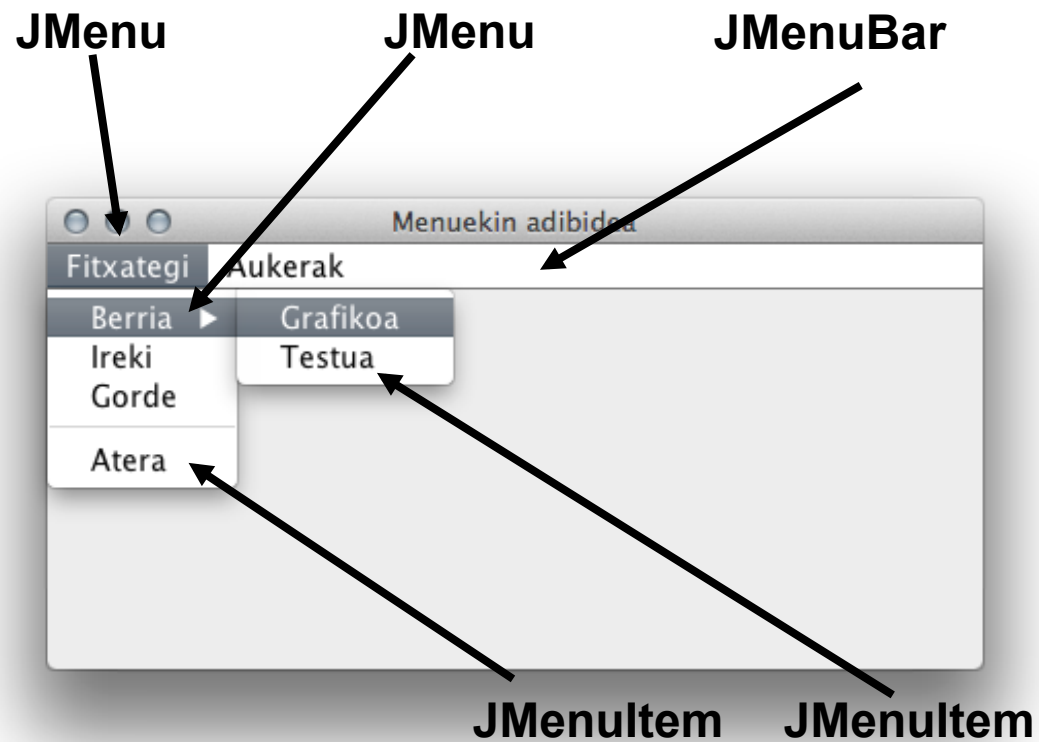
- **JFrame** klasea da menu-barra bat eduki dezakeen edukitzaile bakarra.
    - **JMenuBar** klaseak menuak sartuko diren menu barra sortzen du.
    - **JMenu** klaseak menuak sortzen ditu. Menu hauek izen bat dute eta hainbat elementu azaltzen dituzte goitibeherako-leiho batean.
    - Menuaren elementuak izan daitezke **JMenuItem**-eko objektuak **edo** **JMenu**-koak (menuak kaskadan sortzeko).

- **AWT-en**

- Antzeko klaseak daude.

# AWT/Swing: osagai nagusiak

- **Menuen sorkuntza Swing-en**



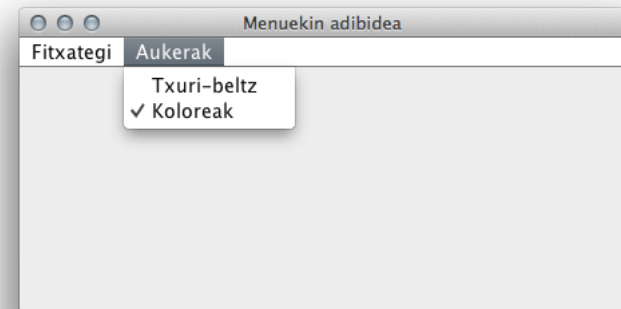
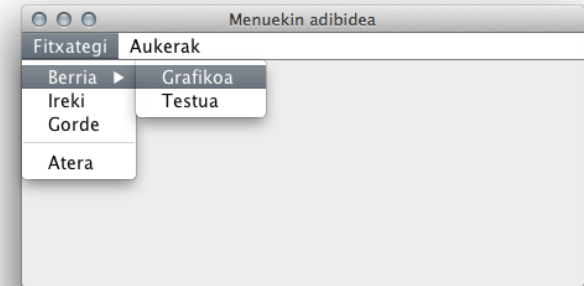
# AWT/Swing: osagai nagusiak

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

public class Menuak extends JFrame {
    JMenuBar menuBarra = new JMenuBar();
    JMenu fitxategi = new JMenu(); JMenu aukerak = new JMenu();
    JMenu berria = new JMenu(); JMenuItem ireki = new JMenuItem();
    JMenuItem gorde = new JMenuItem(); JMenuItem atera = new JMenuItem();
    JMenuItem koloreak = new JRadioButtonMenuItem();
    JMenuItem txuriBeltz = new JRadioButtonMenuItem();
    ButtonGroup bg = new ButtonGroup();
    JMenuItem testua = new JMenuItem(); JMenuItem grafikoa = new JMenuItem();

    public Menuak() {
        this.setJMenuBar(menuBarra); this.setTitle("Menuekin adibidea");
        fitxategi.setText("Fitxategi"); aukerak.setText("Aukerak");
        berria.setText("Berria"); grafikoa.setText("Grafikoa");
        testua.setText("Testua"); ireki.setText("Ireki");
        gorde.setText("Gorde"); atera.setText("Atera");
        koloreak.setText("Koloreak"); txuriBeltz.setText("Txuri-beltz");
        testua.setText("Testua");
        berria.add(grafikoa); berria.add(testua); berria.add(testua)
        fitxategi.add(berria); fitxategi.add(ireki); fitxategi.add(gorde)
        fitxategi.addSeparator(); fitxategi.add(atera);
        menuBarra.add(fitxategi);
        aukerak.add(txuriBeltz); aukerak.add(koloreak);
        bg.add(txuriBeltz); bg.add(koloreak);
        menuBarra.add(aukerak);
    }

    public static void main(String[] args){
        Frame frame = new Menuak();
        frame.setVisible(true);
    }
}
```



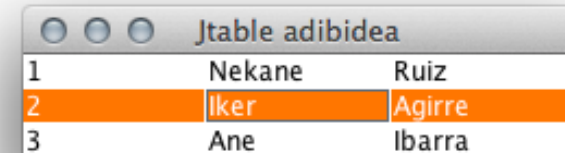
# JTable

## JTable datuak editatu eta aurkezteko

```
import java.awt.BorderLayout;
import javax.swing.JFrame;
import javax.swing.JTable;

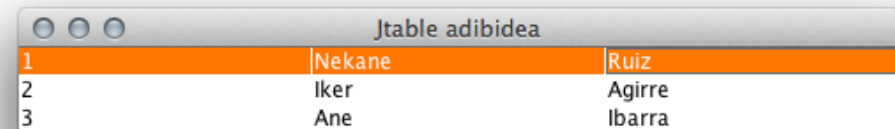
public class TaulaSinple extends JFrame{
    private JTable jTable1;
    final String[] goikoPartea={"KODEA", "IZENA", "ABIZENA"};
    private Object[][] datuak={
        {1, "Nekane","Ruiz"},
        {2, "Iker","Agirre"},
        {3, "Ane","Ibarra"};
    };
    public TaulaSinple(){
        this.setTitle("Jtable adibidea");
        jTable1=new JTable(datuak, goikoPartea);
        this.getContentPane().setLayout(new BorderLayout());

        this.getContentPane().add(jTable1, BorderLayout.NORTH);
        pack();
    }
    public static void main(String args[]){
        TaulaSinple ts=new TaulaSinple();
        ts.setVisible(true);
    }
}
```



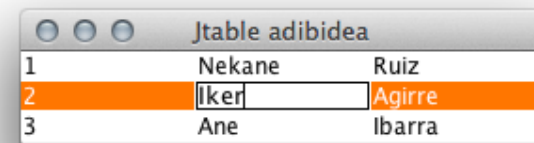
1	Nekane	Ruiz
2	Iker	Agirre
3	Ane	Ibarra

**haunditu daitezke**



1	Nekane	Ruiz
2	Iker	Agirre
3	Ane	Ibarra

**Elementuen balioak aldatu**



1	Nekane	Ruiz
2	Iker	Agirre
3	Ane	Ibarra

# AWT/Swing: osagai nagusiak

- **Dialog/JDialog** klaseak
  - Erabiltzailearen datuak irakurtzeko leihoa.
  - MODAL ezaugarria jartzen badiogu, aktiboa dagoen bitartean ezin izango da beste leiho batetara aldatu.
- **FileDialog** klasea (soilik AWT-n)
  - Fitxategiak ireki eta gordetzeko leihoa (FileDialog.LOAD eta FileDialog.SAVE moduak).
  - Javaren Makina Birtuala egikaritzen ari den Sistema Eragileko leiho bera erabiltzen da.
  - Funtzionalitate hau eskeintzen du: Fitxategi baten gainean idazten saiatzean ez da birprogramtu behar (alerta-leiho bat erakusten du).

# JDialog

```
class DialogPass extends JDialog {
    private JPasswordField jPasswordField1 = new JPasswordField();
    private JLabel jLabel1=new JLabel();
    private JButton jButton1=new JButton();

    public DialogPass (JFrame parent, String title, boolean modal){
        this.setSize(new Dimension(400,135));
        this.getContentPane().setLayout(null);
        jPasswordField1.setText("jPasswordField1");
        jPasswordField1.setBounds(new Rectangle(195,30,130,25));

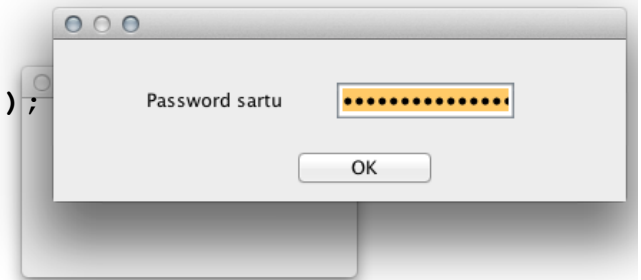
        jLabel1.setText("Password sartu");
        jLabel1.setBounds(new Rectangle(65,35,110,15));

        jButton1.setText("OK");
        jButton1.setBounds(new Rectangle(165,75,105,30));

        this.getContentPane().add(jButton1,null);
        this.getContentPane().add(jLabel1,null);

        this.getContentPane().add(jPasswordField1,BorderLayout.CENTER);
    }

    public static void main (String args[]){
        JFrame a = new JFrame();
        JDialog dp=new DialogPass (a, "Password frogaketa", true);
        a.setVisible(true);
        dp.setVisible(true);
    }
}
```



Modal lehioa, itxi arte ezin da aurreko lehiora joan.

# Aurkibidea

- Sarrera
  - Helburuak
  - AWT klase hierarkia: COMPOSITE diseinu patroia
- Osagai nagusiak
  - AWT/Swing edukitzaileak
  - AWT/Swing osagai nagusiak
  - Diseinuaren kudeatzaileak
- Gertaeren kudeaketa
  - Behe-mailako eta goi-mailako gertaerak
  - Listener-ak



# Diseinuaren kudeatzaileak: Layout

- Edukitzaile barruan osagai bat non kokatzeko erabiltzen dira.

```
edukitzailea.add(osagaia);
```

- **FlowLayout** (Panel-erako defektuzko kudeatzailea)

- Osagaiak gehitzen doa lerro batean. Lerroa bukatzean hurrengora pasatzen da.

- **BorderLayout** (Frame eta Dialog-en kudeatzailea)

- Osagaiak 5 eremuetan gehitzen ditu: ipar, hego, ekialde, mendebalde eta erdia.

- Defektuzko Kudeatzailea aldatu daiteke:

```
edukitzailea.setLayout(new BorderLayout());
```

# Diseinuaren kudeatzaileak: Layout

- Osagaia koordenatu zehatz batzuetan jartzeko osagaiaren kudeatzailea kendu behar da.

```
osagaia.setLayout (null) ;  
// edo  
// setLayout (new XYLayout ()) ;
```

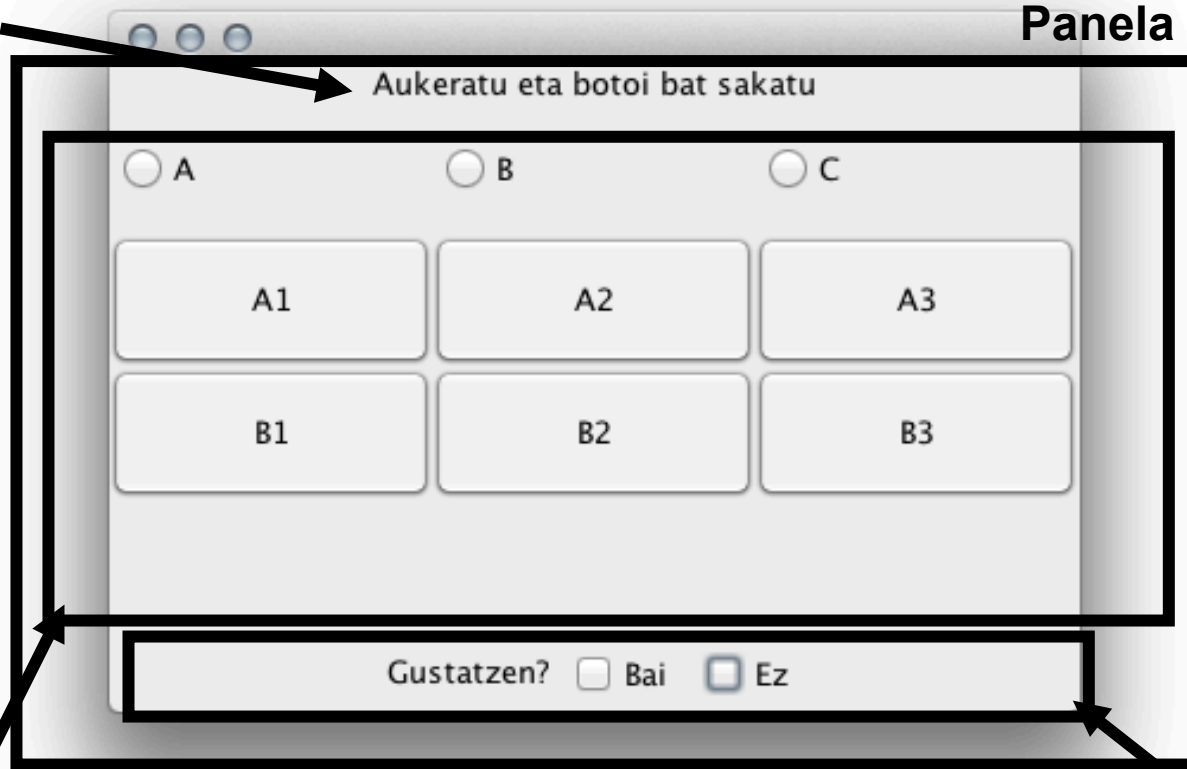
- Bitez `this` edukitzaile bat eta `textField1` bere osagai bat:

```
this.setLayout (null) ;  
textField1.setBounds (15 , 20 , 50 , 60) ;
```

# Diseinuaren kudeatzaileak: Layout

BorderLayout.NORTH-en dagoen  
Labela

BorderLayout duen  
Panela



BorderLayout.CENTER-en dagoen  
Panela GridLayout(3,3)-rekin

BorderLayout.SOUTH-en dagoen  
Panela FlowLayout-ekin

# Diseinuaren kudeatzaileak: Layout

```
import java.awt.BorderLayout;
import java.awt.GridLayout;

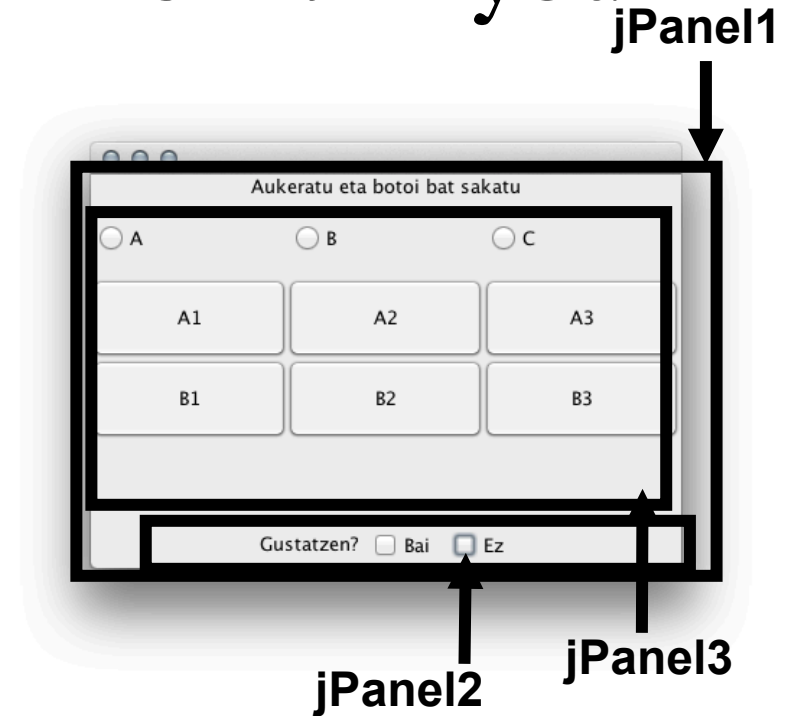
import javax.swing.*;

public class Layoutak extends JFrame {
    JPanel jPanel1 = new JPanel();
    JPanel jPanel2 = new JPanel();
    JPanel jPanel3 = new JPanel();
    BorderLayout borderLayout1 = new BorderLayout();
    GridLayout gridLayout1 = new GridLayout(3,3);
    JLabel jLabel2 = new JLabel();
    JLabel jLabel1 = new JLabel();

    JButton jButton1 = new JButton("A1");
    JButton jButton2 = new JButton("A2");
    JButton jButton3 = new JButton("A3");
    JButton jButton4 = new JButton("B1");
    JButton jButton5 = new JButton("B2");
    JButton jButton6 = new JButton("B3");

    JCheckBox jCheckBox1=new JCheckBox();
    JCheckBox jCheckBox2=new JCheckBox();

    JRadioButton jRadioButton1=new JRadioButton("A");
    JRadioButton jRadioButton2=new JRadioButton("B");
    JRadioButton jRadioButton3=new JRadioButton("C");
```



# Diseinuaren kudeatzaileak: Layout

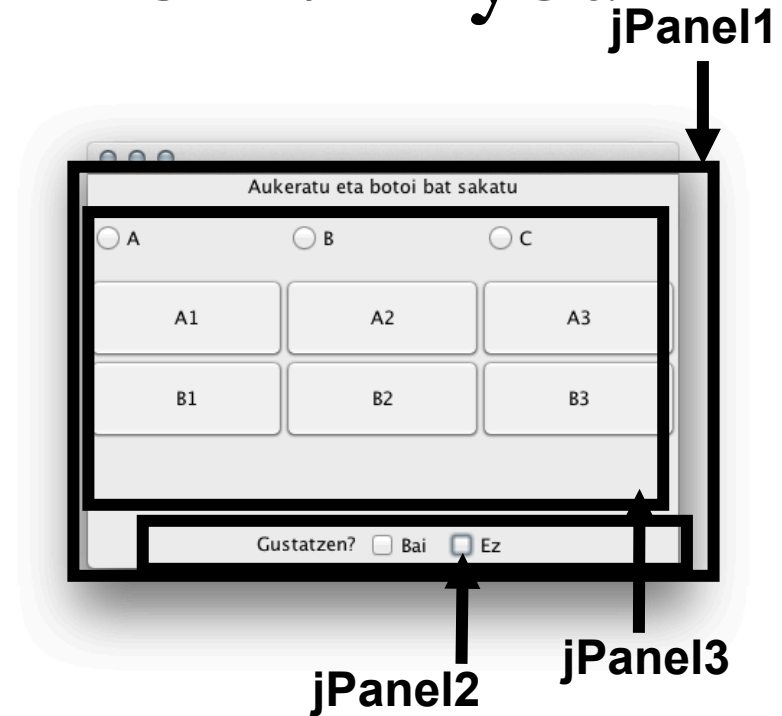
```
public Layoutak() {
    jPanel1.setLayout(borderLayout1);
    jLabel1.setText("Gustatzen?");
    jCheckBox1.setText("Bai");
    jCheckBox2.setText("Ez");
    jLabel2.setText("Aukeratu eta botoi bat sakatu");
    jLabel2.setHorizontalAlignment(SwingConstants.CENTER);
    jPanel3.setLayout(gridLayout1);

    jPanel3.add(jRadioButton1, null);
    jPanel3.add(jRadioButton2, null);
    jPanel3.add(jRadioButton3, null);

    jPanel3.add(jButton1, null);
    jPanel3.add(jButton2, null);
    jPanel3.add(jButton3, null);
    jPanel3.add(jButton4, null);
    jPanel3.add(jButton5, null);
    jPanel3.add(jButton6, null);

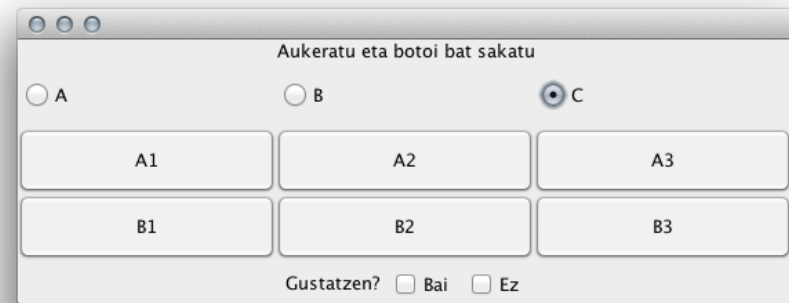
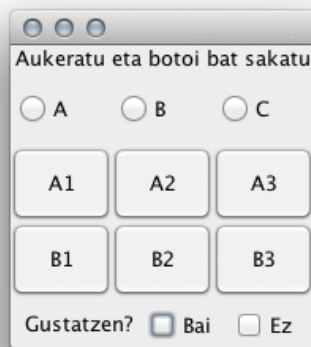
    jPanel2.add(jLabel1, null);
    jPanel2.add(jCheckBox1, null);
    jPanel2.add(jCheckBox2, null);
    jPanel1.add(jLabel2, BorderLayout.NORTH);
    jPanel1.add(jPanel3, BorderLayout.CENTER);
    jPanel1.add(jPanel2, BorderLayout.SOUTH);
    this.getContentPane().add(jPanel1, null);
}

public static void main(String args[]){
    JFrame layoutak=new Layoutak();
    layoutak.setVisible(true);
}
}
```



# Diseinuaren kudeatzaileak: Layout

- Diseinuaren kudeatzaileekin definitzearen abantaila:
  - leihoaren tamaina aldatuz osagaiak automatikoki egokitzen dira.



# Diseinuaren kudeatzaileak: Layout

- Diseinuaren kudeatzaileak gabe osagai guztien koordenatuak ematen dira
  - tresna bisuala erabiltzen erraza da.

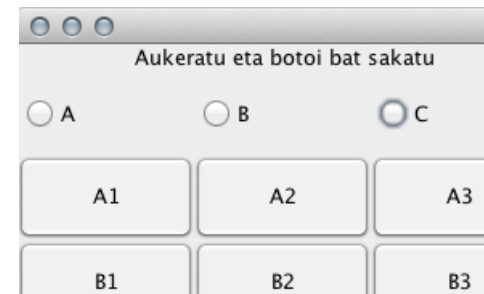
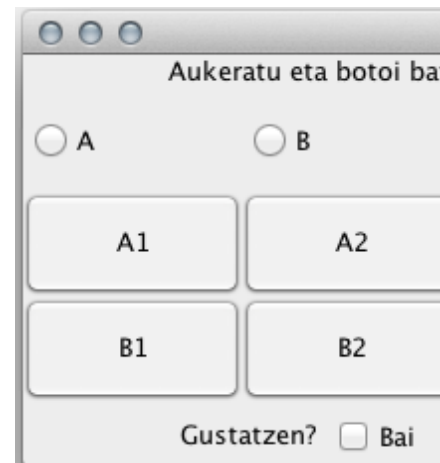
```
jPanel1.setBounds(new Rectangle(3, 0, 333, 170));
jPanel1.setLayout(null);
jPanel2.setLayout(null);
jPanel2.setBounds(new Rectangle(7, 121, 289, 43));
jLabel1.setText("Gustatzen");
jLabel1.setBounds(new Rectangle(66, 18, 62, 16));
jCheckBox1.setText("Bai");
jCheckBox1.setBounds(new Rectangle(142, 17, 48, 19));

jCheckBox2.setText("Ez");
jCheckBox2.setBounds(new Rectangle(201, 17, 44, 21));
jPanel3.setLayout(null);
jPanel3.setBounds(new Rectangle(2, 23, 333, 110));
jRadioButton1.setText("A");
jRadioButton1.setBounds(new Rectangle(14, 10, 49, 23));
jRadioButton2.setText("B");
jRadioButton2.setBounds(new Rectangle(14, 40, 58, 23));
jRadioButton3.setText("C");
jRadioButton3.setBounds(new Rectangle(14, 70, 55, 23));
jButton1.setText("A1");
jButton1.setBounds(new Rectangle(139, 10, 80, 23));
//...
```



# Diseinuaren kudeatzaileak: Layout

- Diseinuaren kudeatzaileak gabe desabantaila:
  - Framea berdimentsionatzean osagaiak zeuden lekuan geratzen dira.





# Aurkibidea

- Sarrera
  - Helburuak
  - AWT klase hierarkia: COMPOSITE diseinu patroia
- Osagai nagusiak
  - AWT/Swing edukitzaileak
  - AWT/Swing osagai nagusiak
  - Diseinuaren kudeatzaileak
- Gertaeren kudeaketa
  - Behe-mailako eta goi-mailako gertaerak
  - Listener-ak

# Gertaeren kudeaketa

- Interfaze grafiko bat diseinatzean kontutan hartu behar da erabiltzailearen ekintzen ondorioz hainbat **gertaera** emango direla.
- Erabiltzailearen gertaerak prozesatzeko hainbat **ekintza** programatu beharko dira.
- Gertaera bat:
  - erabiltzailearen ekintza batek sortzen du.
  - interfazearen osagaien batekin lotuta dago.
  - Adibideak:
    - tekla bat sakatu, xagua mugitu, leiho baten formatua aldatu, leiho bat itxi, leiho bat minimizatu, botoi bat sakatu, osagai baten fokoa galdu edo lortu, testu-eremu baten balioa aldatu, menu bateko item bat hautatu.

# Gertaeren kudeaketa

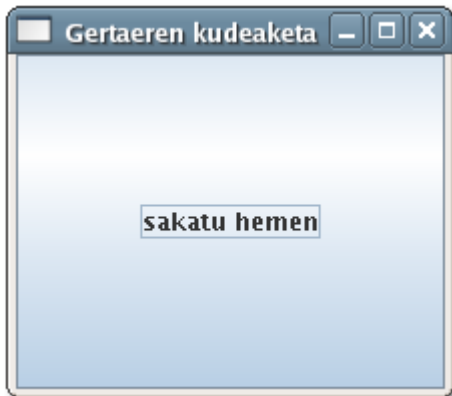
```
import javax.swing.*;
public class SimpleGUI extends JFrame {

    JButton button;

    public void ekin(){
        button = new JButton("sakatu hemen");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.getContentPane().add(button);
        setSize(300,300);
        setVisible(true);
    }

    public static void main(String[] args){
        SimpleGUI frame = new SimpleGUI();
        frame.setTitle("gertaeren kudeaketa");
        frame.ekin();
    }
}
```

# Gertaeren kudeaketa

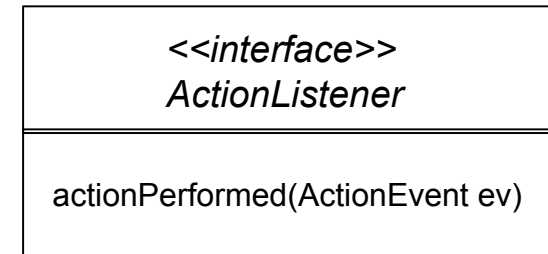


Erabiltzaileak botoia noiz sakatzen duen jakin nahi dugu. Botoia sakatzerakoan gertaera bat sortuko da.

Nola kudeatu "sakatu hemen" botoia klikatzen den gertaera?:

- 1) Klase berezi bat inplementatu beharko dugu, gertaerari erantzuna emateko. Klase horrek interfaze berezi bat inplementatu beharko du.
- 2) Botoian gertaera bat sortzen denean, aurreko puntuan definitutako klase horretako objektu bateri deitzeko erazagutu beharko dugu.

# Gertaeren kudeaketa



GertaeraEntzulea klase berezi bat definitzen da. Klase honek botoi gertaerak kudeatzen duen klase bat da (*ActionListener* interfazea implementatzen du). Botoi-ean gertaera bat ematen denean `actionPerformed` metodoari deituko zaio.

```
public class GertaeraEntzulea implements ActionListener {
```

```
    public void actionPerformed(ActionEvent e) {
```

```
        System.out.println("ados! botoia sakatu duzu!"); //edo  
        this.textuaAldatu();
```

```
    }  
}
```

# Gertaeren kudeaketa (event-handling)

Botoian gertaera bat dagoenean, GertaeraEntzulea klaseko objektu bateri deitzeko erazagutzen da.

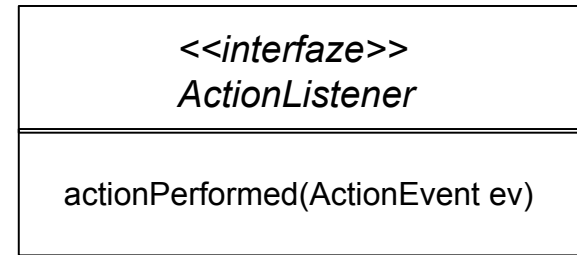
```
public class SimpleGUI extends JFrame{  
  
    JButton button;  
  
    public void ekin(){  
        button = new JButton("sakatu hemen");  
        button.addActionListener(new GertaeraEntzulea());  
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        getContentPane().add(button);  
        setSize(300,300);  
        setVisible(true);  
    }  
}
```

# Listener interfazeak

- Gertaerak kudeatzeko Javak **Listener** interfaze “entzuleak” ematen ditu, programatzaileak inplementatu behar dituen metodoekin.
- Metodo bakoitzaren inplementazioak gertaera bakoitzari erantzun egokia emango dio.
- Objektu grafiko batean hainbat gertaera kontrolatu nahi baditugu, objektuari listener bat esleitzen diogu:

**objGraf.addXXXListener(objListener)**

# Listener-ak



`button.addActionListener(aplikazio objektua)`

Aplikazioa



Listener

`actionPerformed(event)`

Gertaera iturria



# Listener-ak

## Listener (entzule) bat izanda:

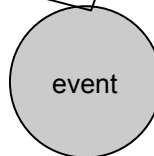
- 1) Interfaze bat inplementatu behar dut
- 2) Botoiaren gertaerak entzuteko, botoian listener bezala erregistratu behar naiz.
- 3) Gertaerari erantzuteko, metodo bat eskaini behar dut.

## Gertaera iturri bezala:

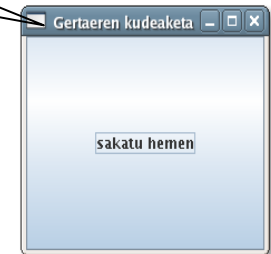
- 1) listener-en erregistroa onartu behar dut.
- 2) Erabiltzaileen akzioak onartu behar ditut.
- 3) Erabiltzailearen akzio bat jasotzen dudanean, gertaera hori listener-ei notifikatu behar diet.

Aplikazioa

Ey! eta nik zer?



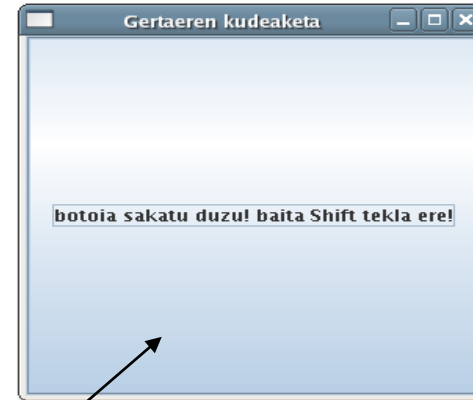
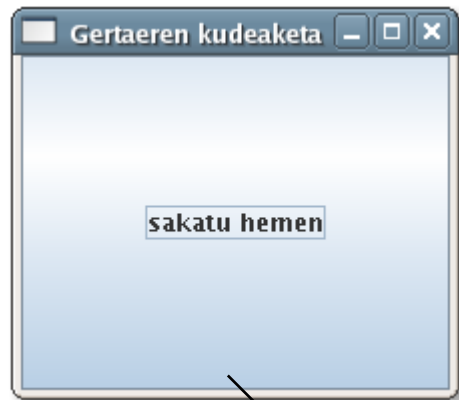
```
public void actionPerformed(ActionEvent e) {
```



Gertaera iturria,  
gertaera bidaliko du

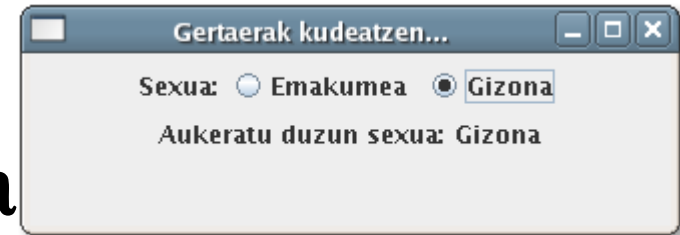
Listener-ak gertaera jasoko du

# Gertaeren kudeaketa



```
public void actionPerformed(ActionEvent e) {  
    button.setText("botoia sakatu duzu!");  
    if ( (e.getModifiers() & ActionEvent.SHIFT_MASK) != 0)  
        button.setText(button.getText() + " baita Shift tekla ere!");  
}
```

# Gertaeren kudeaketa



```
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
```

```
public class Aukerak2 extends JFrame{
```

```
    JLabel jLabel1 = new JLabel("Sexua:");
    JLabel jLabel2 = new JLabel("Aukeratu duzun sexua:");
    JLabel emaitza = new JLabel();
    JRadioButton emakumea = new JRadioButton("Emakumea", true);
    JRadioButton gizona = new JRadioButton("Gizona", false);
    ButtonGroup bg = new ButtonGroup();
```

```
public Aukerak2() {
    super("Gertaerak kudeatzen...");
}
```

```
public void go(){
    bg.add(emakumea);
    bg.add(gizona);
    emakumea.addActionListener(new GertaeraKudeatzaile());
    gizona.addActionListener(new GertaeraKudeatzaile());
    this.getContentPane().setLayout(new FlowLayout());
    getContentPane().add(jLabel1,null);
    getContentPane().add(emakumea,null);
    getContentPane().add(gizona,null);
    getContentPane().add(jLabel2,null);
    getContentPane().add(emaitza,null);
    setSize(300,200);
    setVisible(true);
}
```

```
public class GertaeraKudeatzaile implements ActionListener {

    public void actionPerformed(ActionEvent e) {
        emaitza.setText(e.getActionCommand());
    }
}
```

```
public static void main(String[] args){
    Aukerak2 proba = new Aukerak2();
    proba.go();
    proba.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
}
```

# Gertaeren kudeaketa

```
public void go(){
    bg.add(emakumea);
    bg.add(gizona);
    emakumea.addActionListener(new GertaeraKudeatzaile());
    gizona.addActionListener(new GertaeraKudeatzaile());
    this.getContentPane().setLayout(new FlowLayout());
    getContentPane().add(jLabel1,null);
    getContentPane().add(emakumea,null);
    getContentPane().add(gizona,null);
    getContentPane().add(jLabel2,null);
    getContentPane().add(emitza,null);
    setSize(300,200);
    setVisible(true);
}

public class GertaeraKudeatzaile implements ActionListener {

    public void actionPerformed(ActionEvent e) {
        emitza.setText(e.getActionCommand());
    }
}
```

```
public void go(){
    bg.add(emakumea);
    bg.add(gizona);
    emakumea.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e) {
            emitza.setText(e.getActionCommand());
        }
    });

    gizona.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e) {
            emitza.setText(e.getActionCommand());
        }
    });
    this.getContentPane().setLayout(new FlowLayout());
    getContentPane().add(jLabel1,null);
    getContentPane().add(emakumea,null);
    getContentPane().add(gizona,null);
    getContentPane().add(jLabel2,null);
    getContentPane().add(emitza,null);
    setSize(300,200);
    setVisible(true);
}
```

**Kasu honetan, GUI gertaeren kudeaketa “klase anonimoak” definituz erazagutu daiteke, hau da, ActionListener interfazea implementatzen duen objektu bat sortzen da, bere implementazioa erazagutuz.**

# Gertaerak eta listener interfazeak

**obj1.addXXXListener(obj2)**

**obj1:**

- objektu grafiko baten erreferentzia du (botoia, leihoa, lista, checkbox,...)
- bere ganean gertaeren jokabidea definitu nahi da.

**obj2:**

- XXXListener interfazea implementatzen duen klase bateko objektu baten erreferentzia du.

```
public interface XXXListener
{
    .....
    void actionYYY(AWTEvent e);
}
```

---

Exekuzio denboran **obj1** objektuari dagokion gertaera bat ematen da.

Gertaera objektu bat (AWTEvent) sortzen da eta objektu entzuleari (**obj2**) pasatzen zaio kontrola

Objektu entzuleak ekintzari dagokion metodoa (adib: **akzioYYY**) exekutatu du, parametro bezala sortu den gertaera objektua erabiliz.

# Osagai grafikoak eta listener-ak

- Osagai grafiko bakoitzari esan behar zaio zein Listener interfazea implementatu nahi diogun

## **addXXXListener()**

Adibidez:     - addActionListener(actionListener entzulea)  
                  - addItemListener(ItemListener entzulea)

- Listener batzuk hainbat metodo dituzte, baina agian ez ditugu denak implementatu nahi (ez dizkiegulako gertaera mota posible guztiei erantzun eman nahi). Hala ere Javak interfaze bateko metodo guztiak implementatzera behartzen du.  
Honetarako Javak, Adapter klase egokitzailleak eskeintzen ditu, interfazeen metodo guztiak implementatzen dituztenak (hutsik uzten ditu) .
- Garapenerako tresna bisual bat erabiltzen badugu azkeneko hau ez da arazo bat, eta gainera ez da beharrezkoa Listener mota ezberdinak ikastea.