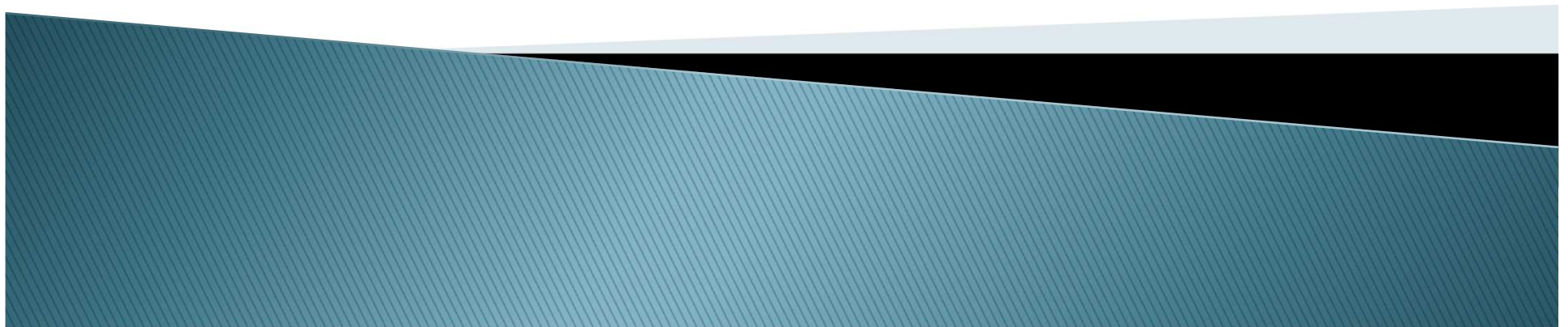


Gertaerei bideratutako programazioa

SOFTWARE INGENIARITZA



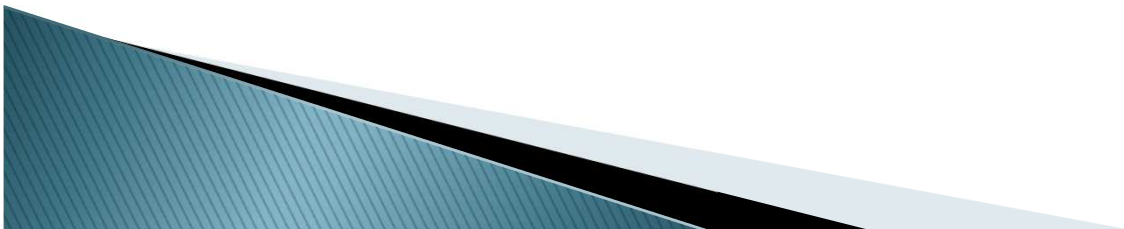
Gertaerei bideratutako programazioa

- ▶ Bai programaren egitura, bai exekuzioa sisteman gertatzen diren ebentoen edo gertaeren menpe daude.
- ▶ Gertaerak erabiltzaileak edo programak berak sor ditzake.
- ▶ Programan definitu beharrekoa exekuzioa kudeatzen duten gertaerak, eta gertaera horiek agertzean burutuko diren akzioak dira.



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 irabazi, testu-eremu baten balioa aldatu, menu bateko item bat hautatu



Gertaeren kudeaketa

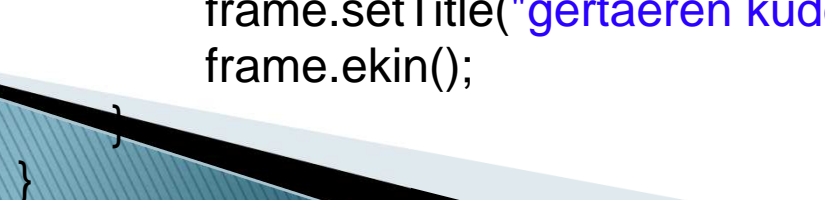
```
import javax.swing.*;

public class SimpleGUI extends JFrame {

    JButton button;

    public void ekin(){
        button = new JButton("sakatu hemen");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        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



Zerbait egin nahi badugu botoia sakatzen denean:

- 1) Metodo bat programatu beharko dugu, gertaera kudeatzeko
- 2) Gertaera noiz sortzen den jakin beharko dugu.

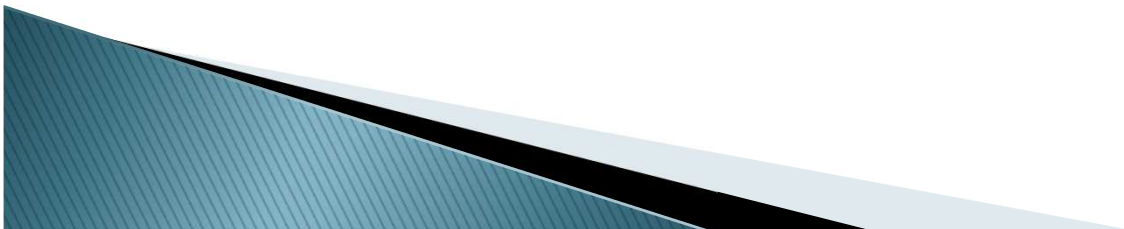
Gertaeren kudeaketa

1

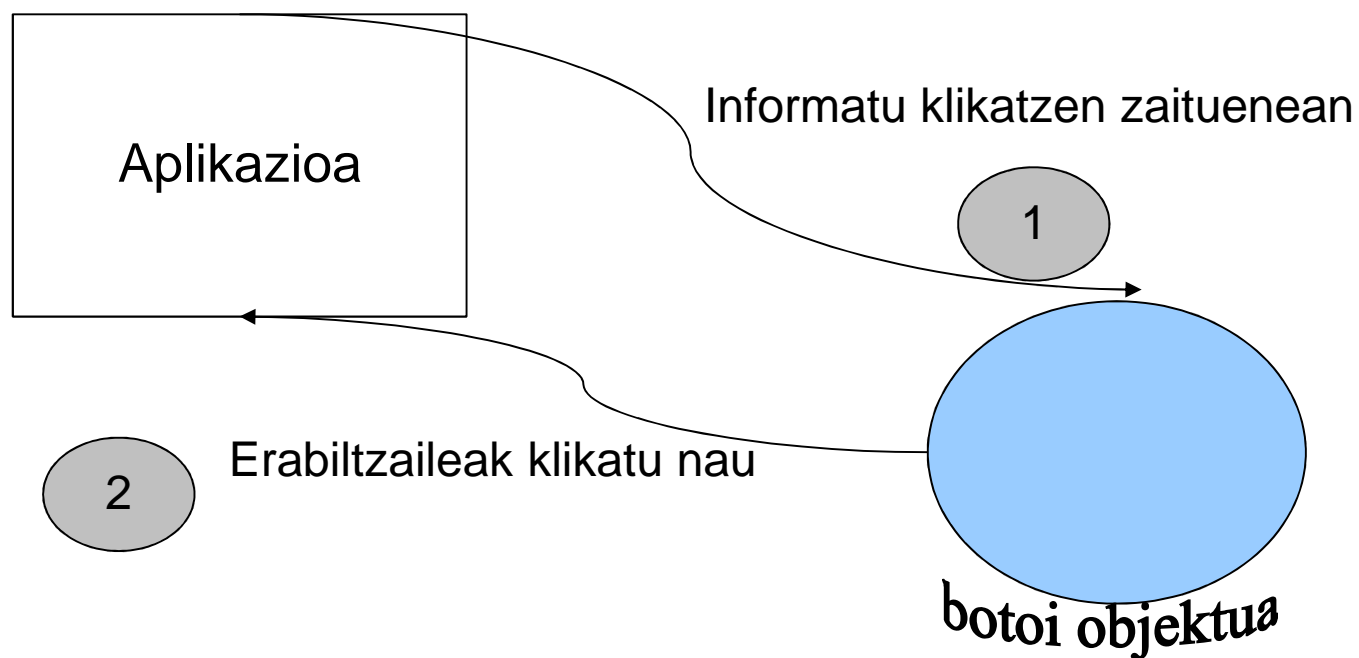
```
public void textuaAldatu(){  
    button.setText("ok! botoia sakatu duzu");  
}
```

2

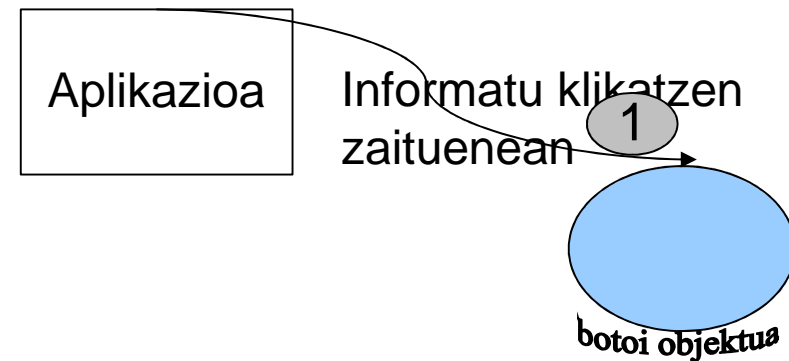
Baina, nola jakin daiteke erabiltzaileak botoia **noiz** sakatu duen?



Gertaeren kudeaketa (*event-handling*)

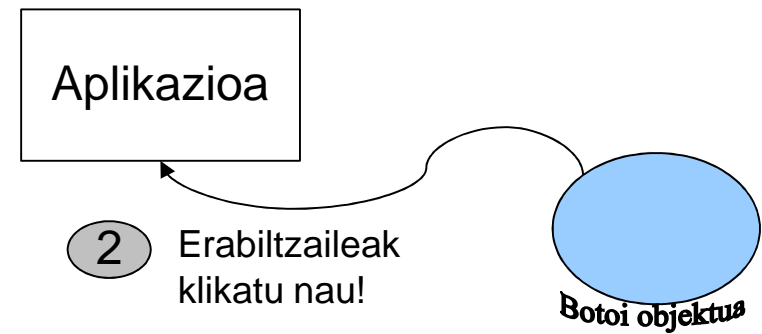


Gertaeren kudeaketa (*event-handling*)



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


Gertaeren kudeaketa (*event-handling*)

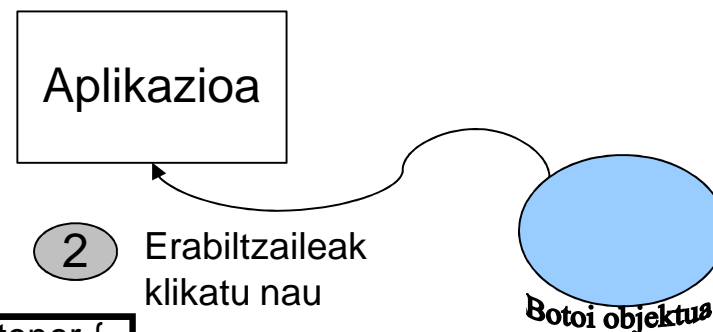


Gertaeren kudeaketa

<<interface>>
ActionListener

actionPerformed(ActionEvent ev)

Botoi baten gertaerei buruz informatuta egon nahi badugu,
ActionListener interfazea implementatu behar dugu



```
public class SimpleGUI extends JFrame implements ActionListener {
```

```
    JButton button;
```

```
    public void ekin(){  
        button = new JButton("sakatu hemen");  
        button.addActionListener(this);  
    }
```

```
    public void actionPerformed(ActionEvent e) {  
        button.setText("ados! botoia sakatu duzu!"); //this.textuaAldatu();  
    }  
}
```

Listener interfazeak

- ▶ Gertaerak kudeatzeko Javak Listener interfaze “entzuleak” ematen ditu, programatzaileak implementatu behar dituen metodoekin.
- ▶ Metodo bakoitzaren implementazioak gertaera bakoitzari erantzun egokia emango dio.
- ▶ Objektu grafiko baten gaineko hainbat gertaera kontrolatu nahi baditugu, objektuari listener bat esleitzen diogu:
`objGraf.addXXXListener(objListener)`



Gertaeren kudeaketa

```
<<interfaze>>  
ActionListener
```

```
actionPerformed(ActionEvent ev)
```

```
button.addActionListener(this)
```

Aplikazioa



Listener

```
actionPerformed(gertaera)
```

Gertaera
iturria

Gertaeren kudeaketa

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

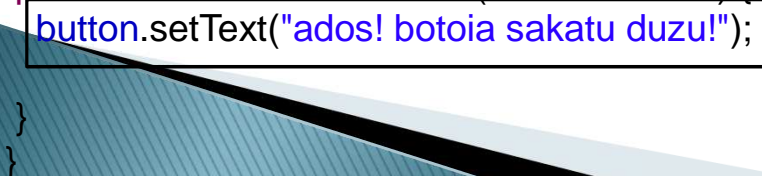
```
public class SimpleGUI extends JFrame implements ActionListener {
```

```
    JButton button;
```

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

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

```
    public void actionPerformed(ActionEvent e) {  
        button.setText("ados! botoia sakatu duzu!");  
    }  
}
```



Gertaeren kudeaketa

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.

Gartaera iturri bezala:

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

Aplikazioa

Listener-ak gartaera jasoko du

Ey! eta nik zer?

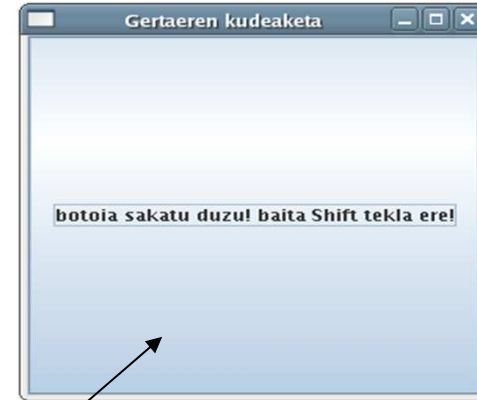
gartaera

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



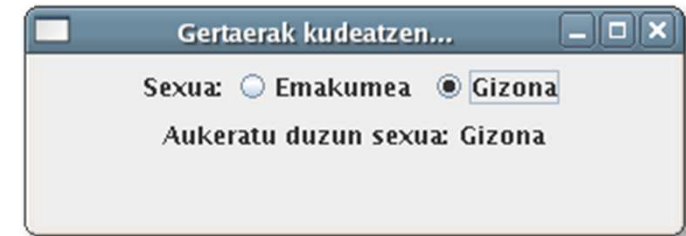
Gartaera iturria,
gartaera bidaliko du

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);
}
```

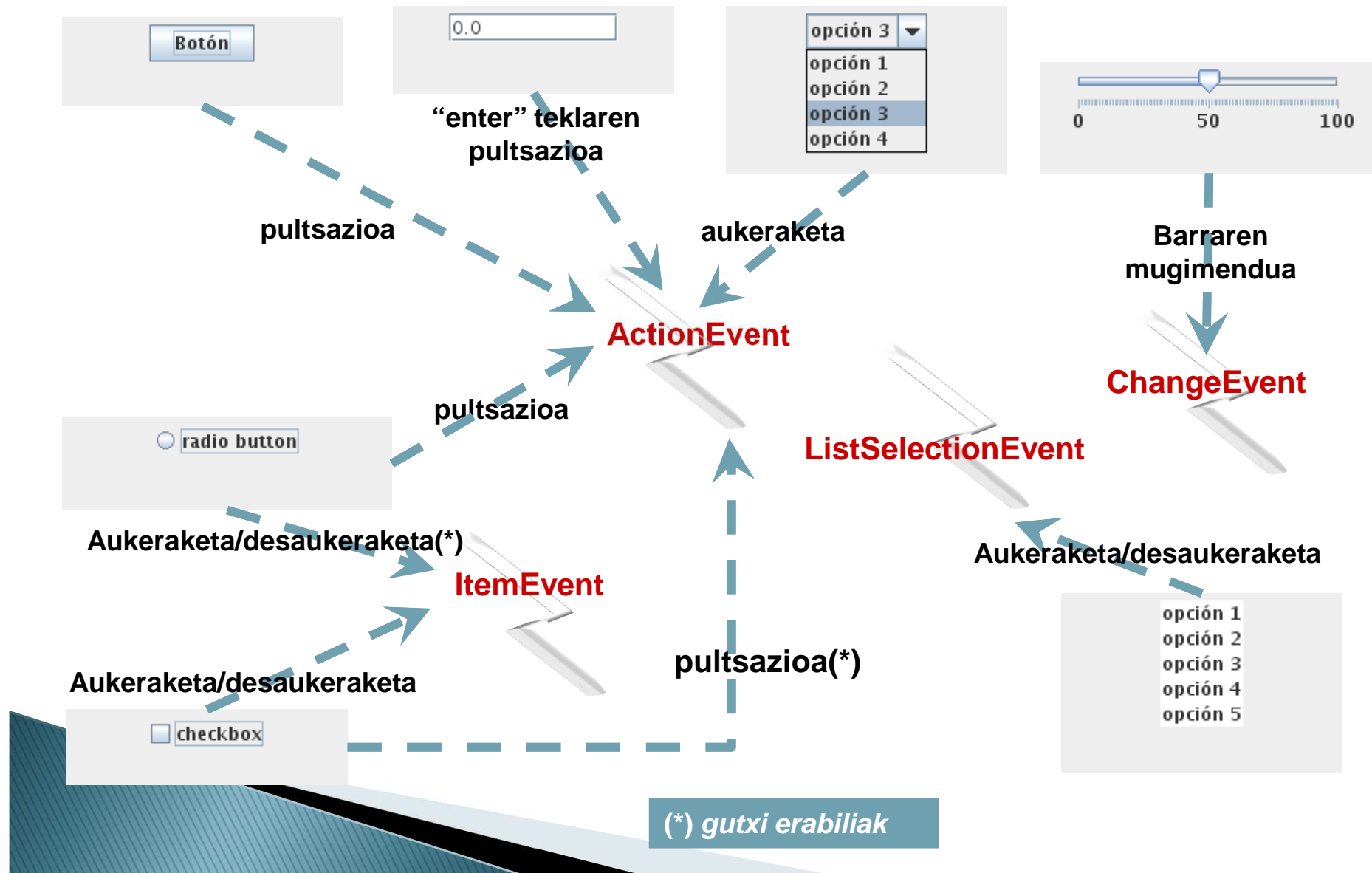
```
public class GertaeraKudeatzaile implements
ActionListener {
```

```
    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);
}
```

Gertaera motak



Gertaera kudeatzaileak

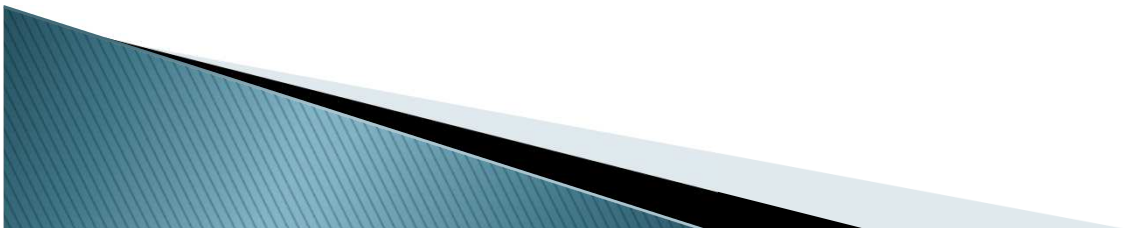
Gertaerak beraiei dagokien interfazea inplementatzen duen **klase kudeatzaile** baten bitartez maneiatzen dira.

Gertaera	Interfaze kudeatzailea	Metodo kudeatzailea
ActionEvent	ActionListener	<code>void actionPerformed (ActionEvent e)</code>
ChangeEvent	ChangeListener	<code>void stateChanged (ChangeEvent e)</code>
ItemEvent	ItemListener	<code>void itemStateChanged (ItemEvent evt)</code>
ListSelectionEvent	ListSelectionListener	<code>void valueChanged (ListSelectionEvent evt)</code>

Gertaerek `java.util.EventObject`-tik gertaera sortu duen osagaia itzultzen duen **Object** `getSource()` metodoa erabiltzen dute.

Gertaerak

Osagai Grafikoa	Gertaera	Entzulea	Metodoak
Jbutton, JTextField...	ActionEvent	ActionListener	actionPerformed(ActionEvent)
Osagaiak	ComponentEvent	ComponentListener	componentHidden(ComponentEvent) componentMoved(ComponentEvent) componentResized(ComponentEvent) componentShown(ComponentEvent)
Osagaiak	FocusEvent	FocusListener	focusGained(FocusEvent) focusLost(FocusEvent)
Osagaiak	KeyEvent	KeyListener	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent)



Gertaerak

Osagai Grafikoa

Osagaiak

Gertaera

MouseEvent

Entzulea

MouseListener

MouseListener

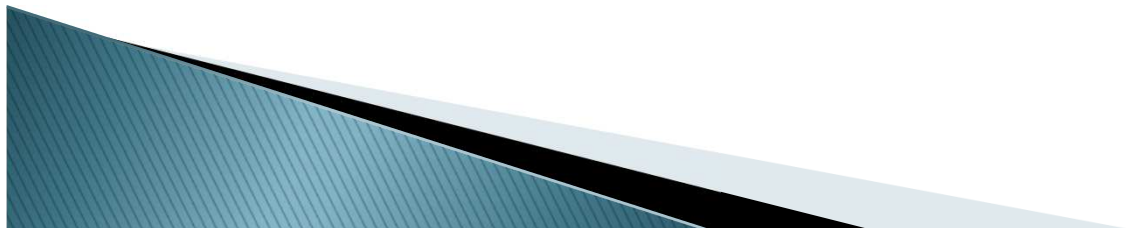
MouseMotionListener

Metodoak

MouseClicked(MouseEvent)
MouseEntered(MouseEvent)
MouseExited(MouseEvent)
MousePressed(MouseEvent)
MouseReleased(MouseEvent)
MouseDragged(MouseEvent)
MouseMoved(MouseEvent)

MouseClicked(MouseEvent)
MouseEntered(MouseEvent)
MouseExited(MouseEvent)
MousePressed(MouseEvent)
MouseReleased(MouseEvent)

MouseDragged(MouseEvent)
MouseMoved(MouseEvent)



Gertaerak

Osagai Grafikoa

Gertaera

Entzulea

Metodoak

Edukiontzia

ContainerEvent

ContainerListener

ComponentAdded(ContainerEvent)
ComponentRemoved(ContainerEvent)

Leihoa

WindowEvent

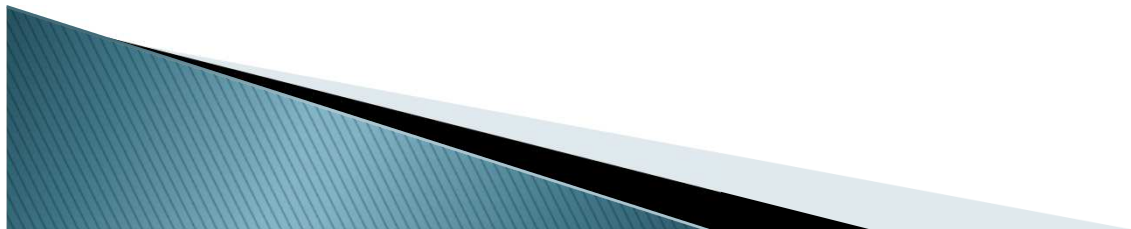
WindowListener

windowActivated(WindowEvent)
windowClosed(WindowEvent)
windowClosing(WindowEvent)
windowDeactivated(WindowEvent)
windowDeiconified(WindowEvent)
windowIconified(WindowEvent)
windowOpened(WindowEvent)



Adapterrak

- ▶ Kasu gehienetan ez ditugu osagai baten gertaera posible guztiak kudeatu behar.
- ▶ Gertaera kudeatzaileen inplementazioa sinplifikatzeko, “Adapter”-ak erabili daitezke.
- ▶ Adapter klase batek Listener interfaza inplementatzen du eta metodo guztiak hutsik daude.



Adapterrak

```
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import javax.swing.JFrame;
public class Sketcher {
    JFrame window = new JFrame("Sketcher");
    public Sketcher() {
        window.setBounds(30, 30, 300, 300);
        window.addWindowListener(new WindowHandler());
        window.setVisible(true);
    }
    class WindowHandler extends WindowAdapter {
        public void windowClosing(WindowEvent e) {
            System.out.println("closing");
            window.dispose(); // Release the window resources
            System.exit(0); // End the application
        }
    }
    public static void main(String[] args) {
        new Sketcher();
    }
}
```

WindowAdapter luzatuz,
WindowListener luzatu
beharrean, behar diren
metodoak bakarrik
implementatu behar dira

