Programación Interactiva
Eventos y Swing

Escuela de Ingeniería de Sistemas y Computación
Facultad de Ingeniería
Universidad del Valle

Primer programa en Swing

```
import javax.swing.*;

public class PrimerSwing {
    private static void mostrarVentana() {
        JFrame frame = new JFrame("Nuestra primera ventana");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JLabel label = new JLabel("Este es nuestro primer programa con Swing");
        frame.getContentPane().add(label);
        frame.setVisible(true);
    }

    public static void main(String[] args) {
         mostrarVentana();
    }
}
```

Segundo programa en Swing

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

public class SegundoSwing implements ActionListener {
    private static int numero_clicks = 0;
    private static void mostrarVentana() {
        JFrame frame = new JFrame("Nuestra primera ventana");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        Jbutton boton = new JButton("Comienza a dar click!");
    frame.getContentPane().add(boton);
        frame.setVisible(true);

        boton.addActionListener(this);
    }
    public void actionPerformed(ActionEvent ae) {
        numero_clicks++;
        boton.setText("Numero de clicks: "+ numero_clicks);
    }
    public static void main(String[] args) {
         mostrarVentana();
    }
}
```

Otra forma de hacer lo mismo

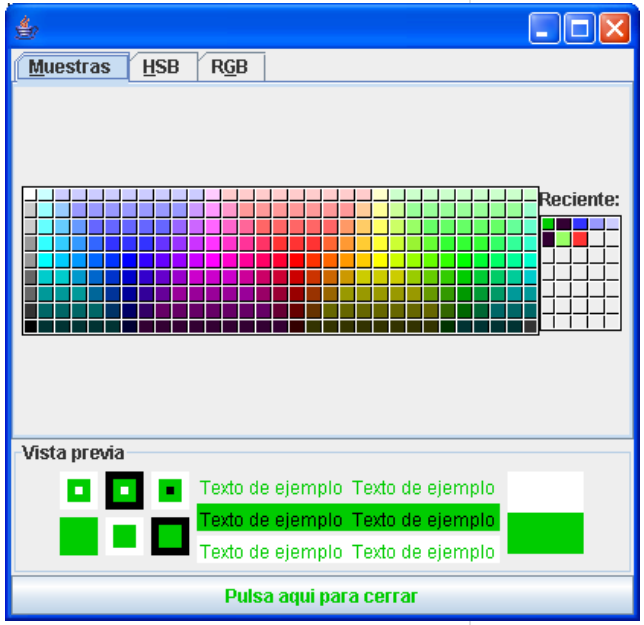
```
import javax.swing.*;
import java.awt.event.*;

public class TercerSwing {
    private static int numero_clicks = 0;
    private static void mostrarVentana() {
        JFrame frame = new JFrame("Nuestra primera ventana");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        Jbutton boton = new JButton("Comienza a dar click!");
    frame.getContentPane().add(boton);
        frame.setVisible(true);

        boton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent ae) {
                numero_clicks++;
                boton.setText("Numero de clicks: "+ numero_clicks);
            }});
    }
    public static void main(String[] args) {
         mostrarVentana();
    }
}
```

Ejemplo con JColorChooser



Ejemplo con JColorChooser

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

public class EjemploJColorChooser extends JFrame implements ChangeListener, ActionListener
{
   JColorChooser jcc;
   JButton jb;
   public EjemploJColorChooser()
   {
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      jcc = new JColorChooser();
      jb = new JButton("Pulsa aqui para cerrar");

      jcc.getSelectionModel().addChangeListener(this);
      jb.addActionListener(this);

      jcc.setMaximumSize(new Dimension(100,100));

      setLayout(new BorderLayout());
      add(jcc, BorderLayout.CENTER);
      add(jb, BorderLayout.SOUTH);

      pack();
   }
```

Ejemplo con JColorChooser

```
    public void stateChanged(ChangeEvent e)
    {
        jb.setForeground(jcc.getColor());
    }

    public void actionPerformed(ActionEvent e)
    {
        JOptionPane.showMessageDialog(this, "Chao!");
        System.exit(0);
    }

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

Introducción

- Cada vez que el usuario escribe un carácter, oprime un botón del mouse, hace un movimiento con el cursor del mouse, presiona una combinación de teclas, ocurre un evento.

- El objeto que recibe el evento (un botón, un área de texto, un panel, una lista, entre otros), es notificado en tiempo de ejecución de que recibió el evento.

- Todo lo que se debe hacer es implementar la interfaz apropiada (event handler) y registrarla como un escucha (event listener) en el componente GUI (event source u objeto que va a recibir el evento) apropiado.

Tipos de Escuchadores

- Los eventos están agrupados de acuerdo a su naturaleza en los siguientes grupos:

  - ***ActionListener:*** acciones sobre componentes.
  - ***WindowListener:*** cierre o manipulación una ventana (Frame/Dialog).
  - ***MouseListener:*** presión de un botón del mouse mientras el cursor está sobre el componente.
  - ***MouseMotionListener:*** movimiento del cursor sobre un componente.
  - ***ComponentListener:*** visibilidad del componentes.
  - ***FocusListener:*** obtención del foco del teclado.
  - ***ListSelectionListener:*** selección de ítems dentro de una lista.

Tipos de Escuchadores

- Cuando un usuario hace click sobre un botón o presiona la tecla Return mientras digitaba en un textfield o escoje una opción de un menú, se genera un evento, que tiene los siguientes elementos:
  - La ***fuente del evento*** (***event source):*** Es el componente que origina el evento.
  - El ***escuchador:*** (***event listener)*** es el encargado de atrapar o escuchar el evento.
  - El ***manejador del evento*** (***event handler)***, es el método que permite implementar la interfaz, es decir el escuchador!!. Este método:
    - Recibe un objeto evento (*ActionEvent*) el cuál tiene información sobre el evento que sucedió,
    - Descifra el evento, con dicho objeto, y
    - Procesa lo solicitado por el usuario.

Comportamiento de los Componentes

- Un componente dispara o activa los manejadores (*event handler)* dependiendo del tipo de evento que ha ocurrido.

- Un componente puede tener registrado más de un manejador que maneja el mismo tipo de evento.

- Un evento puede ser observado o escuchado por más de un manejador de eventos.

Implementación

El manejar un evento requiere que:

3. En la declaración de la clase que maneja el evento (*event handler*), se debe indicar que **implementa la interfaz correspondiente al evento** (*ABCListener*, donde *ABC* es el tipo de evento a observar o escuchar). La clase puede implementar mas de un *ABC*Listener. Si la clase no implenenta la interfaz puede ser que extienda a una clase que sí la implementa.

public class miClase implements ActionListener{…}

Implementación

1. En los componentes que son la fuente del evento (event source) se registra una instancia del manejador del evento (event handler), como un observador o escucha del (de los) tipo (s) de eventos que maneja (ABCListener). Es decir, se le dice al componente que va a escuchar los eventos del tipo del manejador.

componente.addActionListener(
instancia_de_miClaseListener);

Implementación

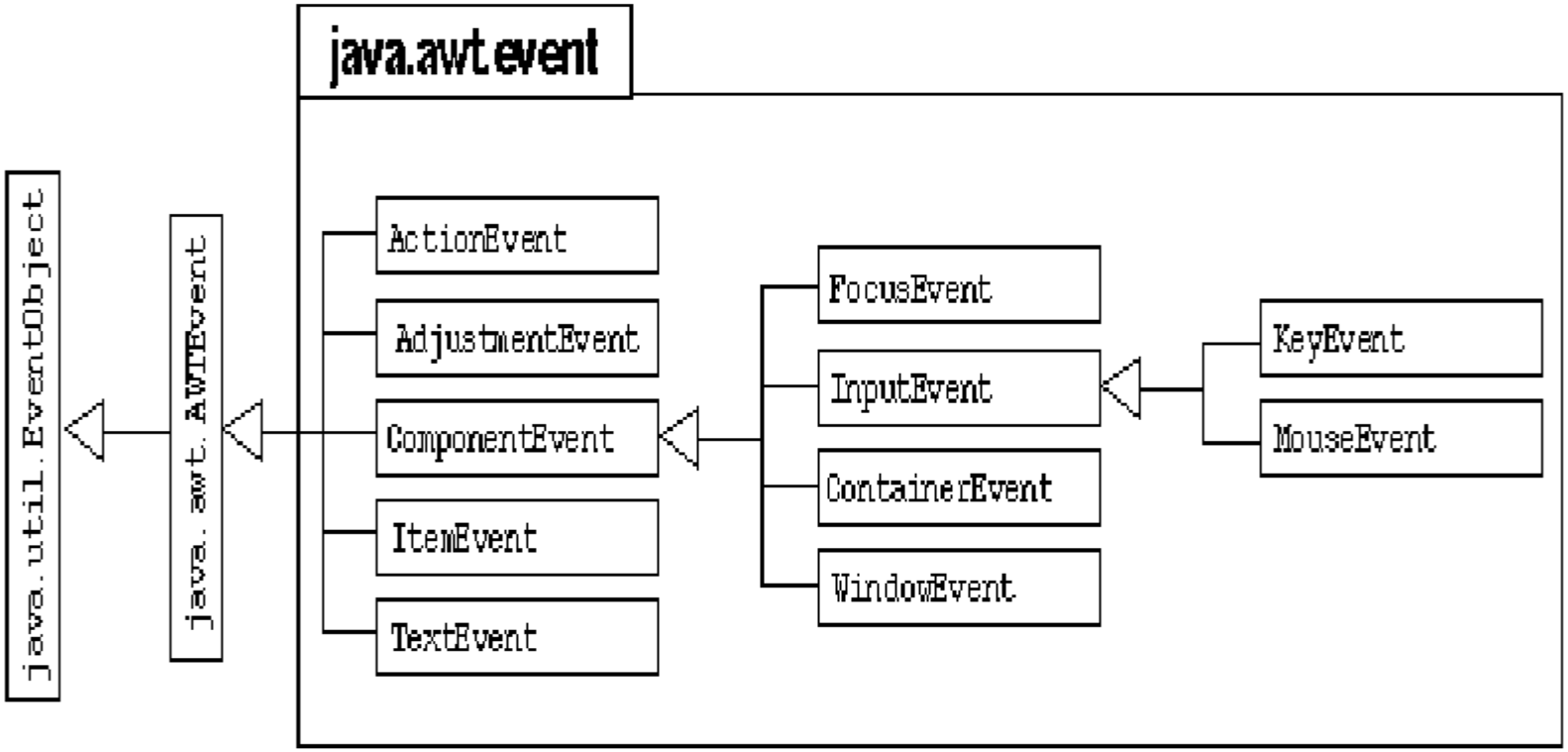
1. En la clase que maneja el evento (*event handler*) se deben implementar los métodos de la interface *ABC*Listener que descifren el evento (*ABC*Event) y lo procesen.

public void actionPerformed(ActionEvent e) {

...//Código que reaccione a la acción

}

Implementación



ActionListener

- Cuando un usuario hace click sobre un botón o presiona la tecla Return mientras digita en un textfield o escoje una opción de un menú, se genera un evento **ActionEvent** y un mensaje **actionPerformed** se envía a todos observadores o escuchas (osea los componentes), que implementan la interfaz **ActionListener,** registrados en el componente.

- La interfase ActionListener presenta el método:

  void actionPerformed(ActionEvent eve)

  que tiene como parámetro el evento generado, que es un instancia de la clase ActionEvent, la que se genera en tiempo de ejecución cuando el usuario inicia el evento.

ActionListener

- La clase ActionEvent presenta además 2 métodos útiles:

```
String getActionCommand()
int getModifiers()
```

- Que permiten saber qué teclas se estaban presionando cuando se produjoi el evento (Shift, Ctrl, Ald, etc)

Ejemplo 1

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

public class Pitar extends Applet implements ActionListener
{
        Button boton;
        public void init()
        {
            boton = new Button(“Presioneme");
            add(boton);
            boton.addActionListener(this);
        }
        public void actionPerformed(ActionEvent e)
        {
            Toolkit.getDefaultToolkit().beep();
        }
}
```

Ejemplo 2

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

public class AppletActionevent extends Applet implements
     ActionListener {
     Button b1;
     Label lb1;
     boolean modificado = false;
     public void init ()
     {
        b1 = new Button("Presioneme");
        b1.setActionCommand("boton Presioneme");
        add(b1);
        b1.addActionListener(this);
        lb1 = new Label(" ");
        add(lb1);
     }
```

Ejemplo 2

```
    public void actionPerformed(ActionEvent eve) {
       if (!modificado) {
              b1.setLabel("cancelar");
              modificado = true;
              lb1.setText(b1.getActionCommand());
       } else {
              b1.setLabel("oprimame");
              modificado = false;
              Button origen = (Button)eve.getSource();
              lb1.setText(origen.getLabel());
       }
}
```

ComponentListener

- Uno o más eventos se disparan o activan después que un componente es escondido, se hace visible, es movido ó se cambia su tamaño.

- La interfaz ComponentListener y su correspondiente clase **Adapter** ComponentAdapter, presentan los siguientes métodos:

ComponentListener

- *void componentHidden(ComponentEvent evento)*
- *void componentMoved(ComponentEvent evento)*
- *void componentResized(ComponentEvent evento)*
- *void componentShown(ComponentEvent evento)*

- La clase ComponentEvent presenta el método:

Component c = getComponent()

Ejemplo 3

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

public class AppletComponentEvent extends Applet {
      Button b1,b2;
      Label lb1;
      boolean mostrado = true;
      public void init () {
          b1 = new Button("esconder");
          add(b1);
          lb1 = new Label(" ");
          add(lb1);
          b2 = new Button("boton");
         add(b2);
         b1.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent eve) {
                           if (!mostrado) {
                                    b2.setVisible(true);
```

Ejemplo 3

```
                    mostrado = true;
                    b1.setLabel("esconder");
                } else {
                    b2.setVisible(false);
                    mostrado = false;
                    b1.setLabel("mostrar");
                }
            }
        });
        b2.addComponentListener(new ComponentAdapter() {
            public void componentHidden(ComponentEvent eve) {
                lb1.setText("escondido");}
            public void componentMoved( ComponentEvent eve) {
                lb1.setText("movido");}
            public void componentShown (ComponentEvent eve) {
                lb1.setText("mostrado")}
            });
        }
}
```

ItemListener

- Los eventos ItemEvent son disparados o activados por componentes que implementan la interfaz ItemSelectable. Estos componentes mantienen un estado on/off para uno o más items u opciones.

- La interface ItemListener tiene un solo método
  void itemStateChanged(ItemEvent eve)

- El método se ejecuta justo después de que un componente cambia de estado.

- La clase ItemEvent tiene los siguientes métodos:
  Object getItem()
  ItemSelectable getItemSelectable()
  int getStateChange()

MouseListener

- Los eventos MouseEvent le informan cuando el usuario utiliza el mouse para interactuar con un componente.
- Ocurren cuando el cursor entra o sale del area del componente en la ventana y cuando el usuario oprime o libera el botón del mouse.
- La interfaz MouseListener y su correspondiente clase Adapter MouseAdapter, contienen los siguientes métodos:

void *mouseClicked (**MouseEvent** eve)*
void *mousePressed (**MouseEvent** eve)*
void *mouseReleased (**MouseEvent** eve)*
void *mouseEntered (**MouseEvent** eve)*
void *mouseExited (**MouseEvent** eve)*

MouseMotionListener

- Los eventos *MouseMotion* informan cuando el usuario mueve el cursor encima del componente.

- La interface MouseMotionListener y su correspondiente clase “adapter” MouseMotionAdapter, contienen los siguientes métodos:

  void mouseMoved(MouseEvent eve)
  void mouseDragged(MouseEvent eve)

- Si su programa necesita usar eventos del Mouse y eventos de *MouseMotion*, usted puede usar la Clase de Swing, *MouseInputAdapter*, que implementa *MouseListener* y *MouseMotionListener*.

KeyListener

- Los eventos del teclado nos dicen cuando un usuario presiona o libera una tecla.

- Existen 2 tipos de Eventos:
  - Digitar un carácter Unicode (keytiped)
  - Presionar/liberar una tecla (key-pressed)

- Para que se active un de los eventos el componente debe tener el foco:

void *keyTyped (**KeyEvent** eve)*
void *keyPressed (**KeyEvent** eve)*
void *keyReleased (**KeyEvent** eve)*

KeyListener

- La Clase **KeyEvent** define los siguientes métodos útiles:

  int *getKeyChar()*

  void *setKeyChar()*

  int *getKeyCode()*

  void *setKeyCode(**int**)*

- La Clase **KeyEvent**, define muchas constantes así:

  KeyEventVK_A especifica la tecla **A**.

  KeyEventVK_ESCAPE especifica la tecla **ESCAPE**.

  void *setModifiers(**int**)*

  String *getKeyText()*

  String *getKeyModifiersText()*

Ejemplo

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

public class ClaseKeyEvent extends Applet {
        Label lb1;
        TextField t1;
        public void init () {
                t1 = new TextField(20);
                add(t1);
               lb1 = new Label(" ");
               add(lb1);
               t1.addKeyListener(new KeyListener() {
                                char caracter;
                                String cadena;
                                public void keyTyped(KeyEvent eve) {
                                                carácter = eve.getKeyChar();
                                                cadena = String.valueOf(caracter);
                                                lb1.setText(cadena);
                                }
```

Ejemplo

```
        public void keyPressed(KeyEvent eve) {
                if (eve.getKeyCode() == KeyEvent.VK_ALT)
                        cadena = "alt";
                if (eve.getKeyCode() == KeyEvent.VK_F1)
                        cadena = "F1";
                if (eve.getKeyCode() == KeyEvent.VK_F2)
                        cadena = "F2";
                lb1.setText(cadena);
        }
        public void keyReleased(KeyEvent eve) {}
        });
    }
}
```

FocusListener

- Muchos componentes, aún aquellos que inicialmente trabajan con el Mouse (botones), pueden operar con el teclado.
- Para que el presionar una tecla pueda afectar un componente, el componente debe tener el foco del teclado.
- Al menos un componente en la ventana del sistema puede tener el foco del teclado.
- Los eventos de Foco son disparados o activados cuando un componente gana o pierde el Foco
- Típicamente el usuario establece el foco haciendo click sobre una ventana o un componente, ó haciendo “tab” entre los componentes.

ContainerListener

- Los eventos ContainerEvent son disparados o activados despues que un componente es agregado o removido del contenedor.

- Estos Eventos son usados solo para notificar que se presentó la adición o remoción del componente, no es necesario el ContainerListener para que los componentes puedan ser agregados o removidos.

- La interfaz ContainerListener y su correspondiente clase “adapter” ContainerAdapter, presenta 2 métodos:
  void componentAdded(ContainerEvent eve)
  void componentRemoved(ContainerEvent eve)

- La clase ContainerEvent, define 2 métodos útiles:
  Component getChild()
  Component getContainer()

WindowListener

- Los eventos de Ventanas son disparados o activados cuando una ventana es abierta, cerrada, iconificada (iconified), desconificada (deiconified), activada o desactivada.

- Los usos más comunes, son por ejemplo, usar un WindowListener, cuando el usuario cierra una ventana, para preguntar si desea salvar los cambios o si es la última ventana cerrada para salir del programa.

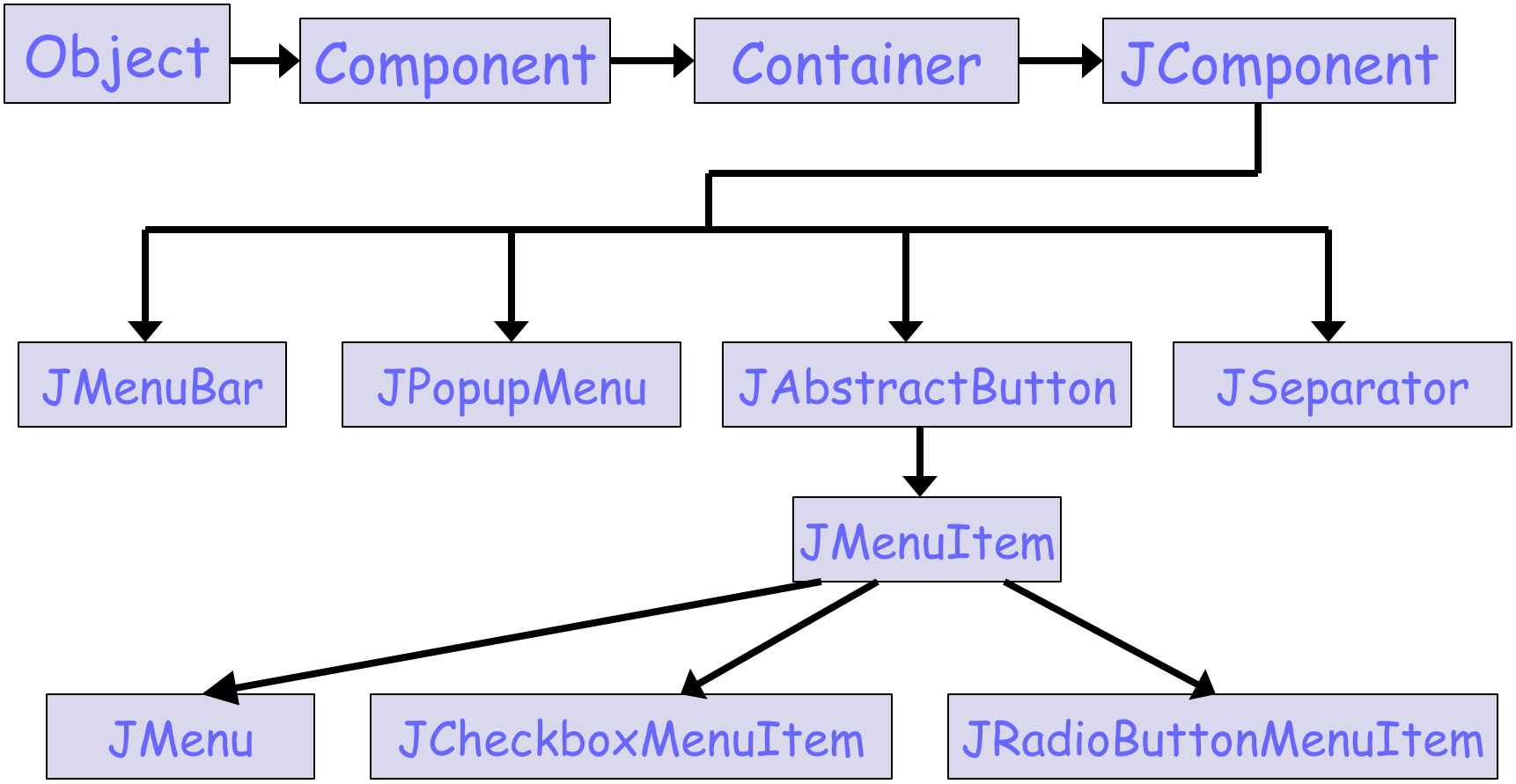
- Por defecto cuando un usuario cierra una ventana esta se vuelve invisible.

GUI con Swing: Menús

- Un menú provee un forma económica en espacio, de escoger una entre varias opciones.

- Un menú aparece en:
  - Una barra de menús (Menú Bar) ó
  - Un Popup Menú

Jerarquía de los Componentes del Menú



Menús

- **Crear Menu Bar:**

  JMenuBar mb = new JMenuBar();

- **Crear Menu:**

  JMenu mn = new JMenu();
  JMenu mn = new JMenu("Archivo");
  mb.add(mn);

- **Agregar Items al Menu:**

  JMenuItem mni = new
  JMenuItem("Salvar");
  mn.add(mni);
  mn.addSeparator();

- **Agregar un Separador:**

  void addSeparator()

Menús

- **Insertar Items en una Posición:**
  JMenuItem insert(JMenuItem, int)
  void insert (String, int)
  void insertSeparator (String, int)

- **Remover Items del Menu:**
  void remove(JMenuItem)
  void remove(int)
  void removeAll()

Ejemplo 10

```
mb = new JMenuBar();
menuarchivo = new JMenu("Archivo");
menuitemnuevo = new JMenuItem("Nuevo");
menuarchivo.add(menuitemnuevo);
menuitemcargar = new JMenuItem("Cargar");
menuarchivo.add(menuitemcargar);
menuarchivo.addSeparator();
menuitemsalir = new JMenuItem("Salir");
menuarchivo.add(menuitemsalir);
mb.add(menuarchivo);

menuayuda = new JMenu("Ayuda");
menuitemayuda = new MenuItem("Ayuda");
menuayuda.add(menuitemayuda);
mb.add(menuayuda);
setJMenuBar(mb);
```

Repaso

- Cada vez que el usuario escribe, un carácter, u oprime un botón, con el Mouse, ocurre un evento.
- Los Event sources son típicamente componentes, pero otras clases de objetos pueden ser event sources.
- Se puede implementar un action listener para que responda a una indicación del usuario.
- Uno o mas eventos, pueden ser efectuados por un Objeto de tipo componente. Justo después de que el componente se aparece ó se esconde ó se mueve,

Repaso

- *Los* eventos del Mouse nos dicen cuando el usuario usa el mouse *(o un dispositivo de entrada similar) para interactuar con un componente.*

- Los eventos de *Mouse-Motion nos dicen cuando el usuario, mueve el cursor del Mouse sobre la pantalla.*

- *Los eventos del teclado, son lanzados, por el componente, que tenga el foco del teclado. cuando el usuario presiona o libera alguna tecla.*

- *Un menu provee un espacio que le permite al usuario, escoger una de varias opciones.*

Repaso

- *Muchos* componentes, pueden ser operados con el teclado. Para que las teclas lo afecten, el componente, debe tener el foco del teclado.

- *Los* eventos *Container son lanzados, por un contenedor, justo después de que un componente es agregado o removido, del contenedor.*

- Los eventos de Ventanas, son lanzados, justo después de que una ventan es abierta, cerrada, activada, o desactivada.