Graphical User Interfaces. Events

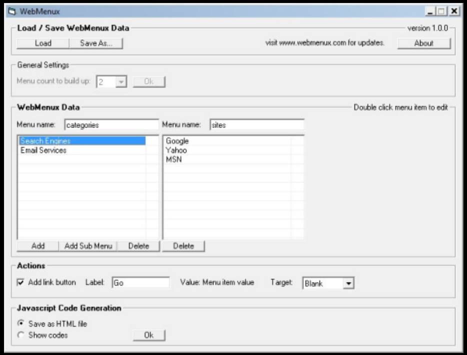
Jose Jesus García Rueda

Session objectives

- Being able to add behaviour to the graphical elements in the interface...
- ...modifying them as a result of the actions on them, also.
- In other words, to cover the whole cycle:
  1. Reciving events that take place on the graphical elements.
  2. Processing them.
  3. Showing feed-back on the screen.

Graphical application architecture

Interface **Processing** **Persistence**



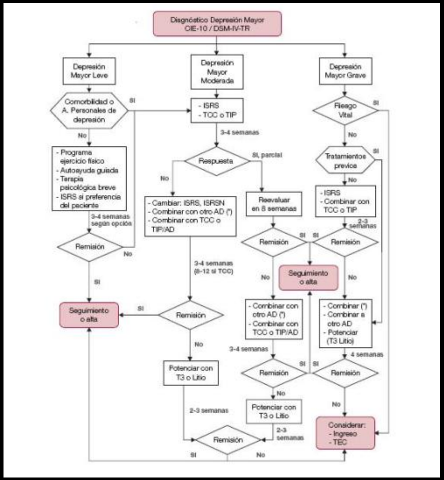
Load / Save WebMenux Data — version 1.0.0
visit www.webmenux.com for updates.

General Settings
Menu count to build up: 2

WebMenux Data — Double click menu item to edit
Menu name: categories
Search Engines
Email Services
Menu name: sites
Google
Yahoo
MSN

Actions
[✓] Add link button Label: Go Value: Menu item value Target: Blank

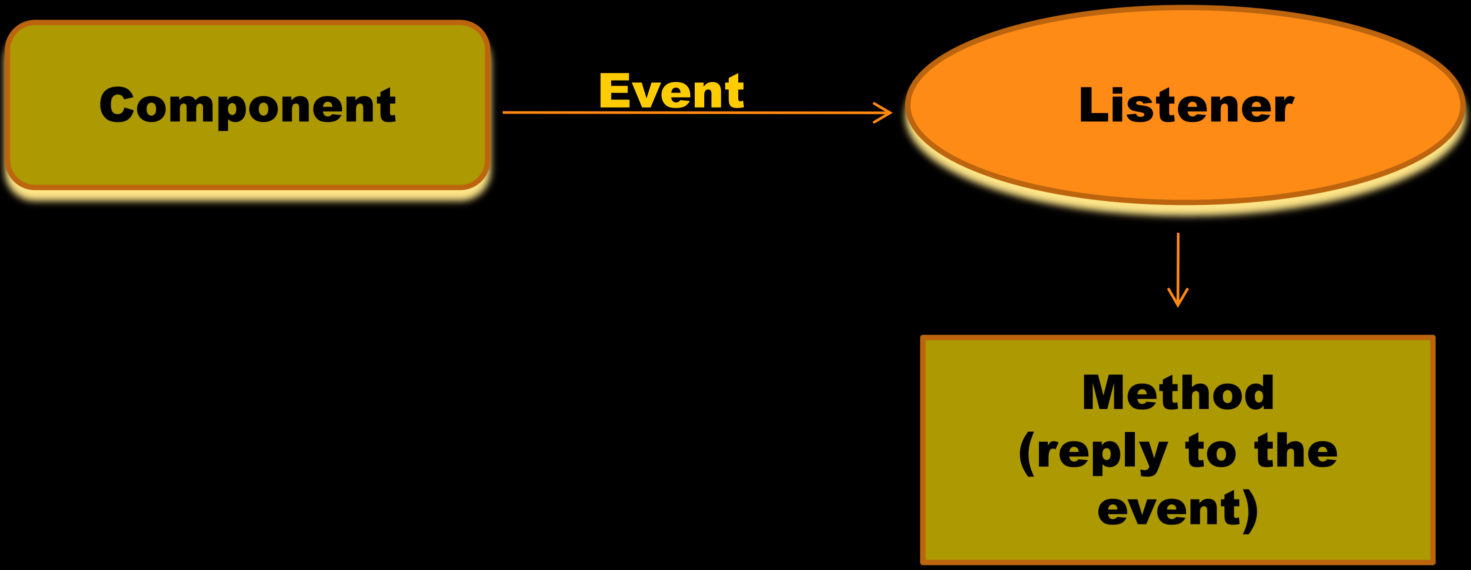
Javascript Code Generation
[✓ Save as HTML file]
Show codes


Diagnóstico Depresión Mayor CIE-10 / DSM-IV-TR
Depresión Mayor Leve
Depresión Mayor Moderada
Depresión Mayor Grave
Comorbilidad o A. Personales de depresión
SI
No
- ISRS - TOC o TIP
3-4 semanas
Riesgo Vital
SI
No
- Programa ejercicio físico - Autoayuda guiada - Terapia psicológica breve - ISRS si preferencia del paciente
Respuesta
SI, parcial
No
Tratamientos previos
SI
No
- Cambiar: ISRS, ISRSN - Combinar con otro AD (*) - Combinar con TCC o TIP/AD
Reevaluar en 8 semanas
- ISRS - Combinar con TCC o TIP
3-4 semanas según opción
2-3 semanas
Remisión
No
Remisión
SI
SI
Remisión
3-4 semanas (8-12 si TCC)
SI
Seguimiento o alta
No
No
Seguimiento o alta
SI
Remisión
- Combinar con otro AD (*) - Combinar con TCC o TIP/AD
- Combinar (*) - Combinar a otro AD - Potenciar (T3 Litio)
No
3-4 semanas
4 semanas
Potenciar con T3 o Litio
Remisión
SI
SI
Remisión
No
No
Potenciar con T3 o Litio
2-3 semanas
2-3 semanas
Remisión
Considerar: - Ingreso - TEC
SI
No

How is this link created?

Anybody listening?

- When users act on the interface, something should happen.
- For being so, we will have to program events managers (listeners)

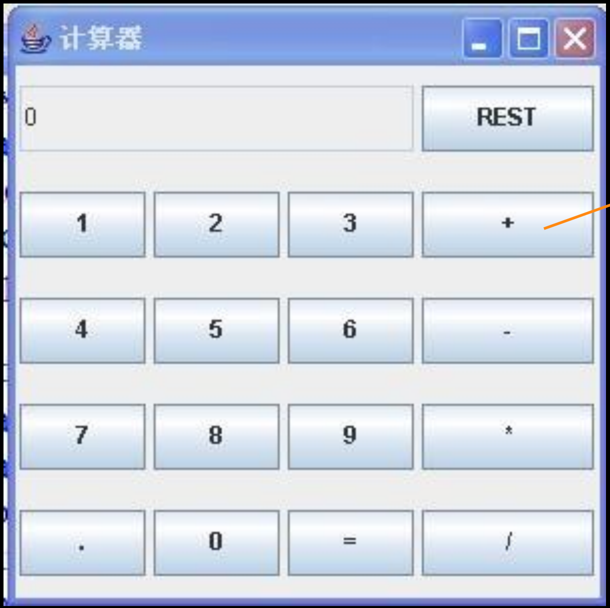


Examples of listeners

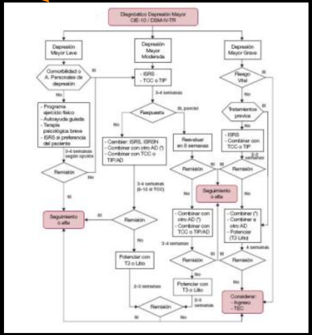
- WindowListener
  - For managing window events.
- ActionListener
  - For managing buttons and other simple components events.
- You’ll have to consult the API constantly!

Active waiting

- Once the GUI is "painted" on the screen…
- … the program stays in a "stand-by" mode, non running any active code!



When something happens on the interface, the associated listener wakes up



And translated into code?

This package includes the listeners

Listeners are interfaces, usually

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

public class ListenerExample implements ActionListener {

   public void actionPerformed (ActionEvent e) {

      System.out.println(“Inside the listener");

   }

}
```

This method is awaken automatically

Who listens whom?

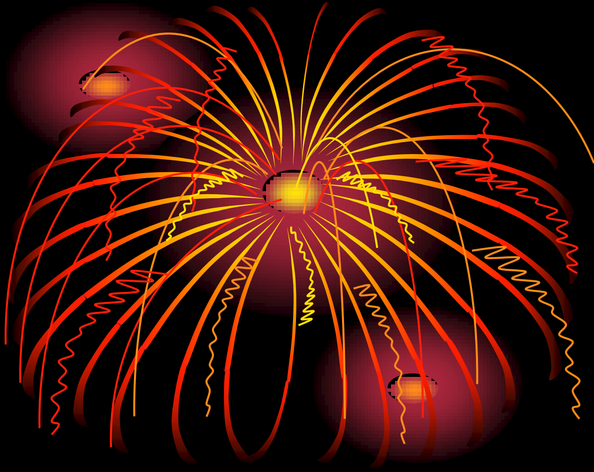
- If we have several graphical components…
- …and we can create as many listeners as we wish…
- Who listens whom?
- We'll have to associate, explicitly, the listeners to the components.
- The possible combinations are multiple:
  - Several listeners associated to the same component.
  - One listener associated to several components.

How to set up the association?

```
import javax.swing.*;

public class Example2 extends JFrame {

   JButton myButton = new JButton ("Click here");
   ListenerExample myListener = new ListenerExample();

   public Example2 () {
         getContentPane().add(myButton);
         myButton.addActionListener(myListener);
   }

   public static void main (String[] arg) {
         Example2 window = new Example2();
         window.setSize(200, 200);
         window.setVisible(true);
     }
}
```

Creating an instance of the corresponding listener

Associating the listener to the component

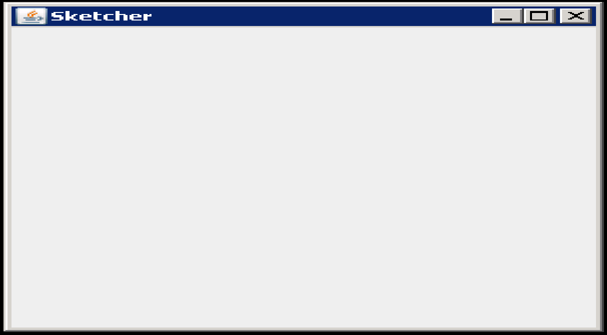


Which part of the listener is awaken?

- **Listeners have different methods to listen to different events.**
- **Java automatically invokes the suitable method, depending on the event.**
- **The body of these methods will be programmed by us. We can invoke other methods from these.**
- **When the method's running is over the program moves on to stand-by again, awaiting for new events.**
- **These methods receive an event object as argument.**

Example: WindowListener

- Among its methods we find:
  - void windowClosing (WindowEvent evt)
  - void windowOpened (WindowEvent evt)
  - void windowClosed (WindowEvent evt)
  - void windowIconified (WindowEvent evt)
  - void windowDeiconified (WindowEvent evt)
  - void windowActivated (WindowEvent evt)
  - void windowDeactivated (WindowEvent evt)



May I get more information about an event?

- The event received as an argument by the listeners’ methods is provided by Java automatically.
- “Asking” to that event object we can find out more things about what really happened.
- Asking, as ever, is done by invoking methods of the event object.

Example

Argument provided by Java automatically

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

public class ListenerExample implements ActionListener {

   public void actionPerformed (ActionEvent e) {

      String source = e.getActionCommand();
      System.out.println("Button: " + source);

   }

}
```

It gives back the label of the component that started the event

Events oriented programming

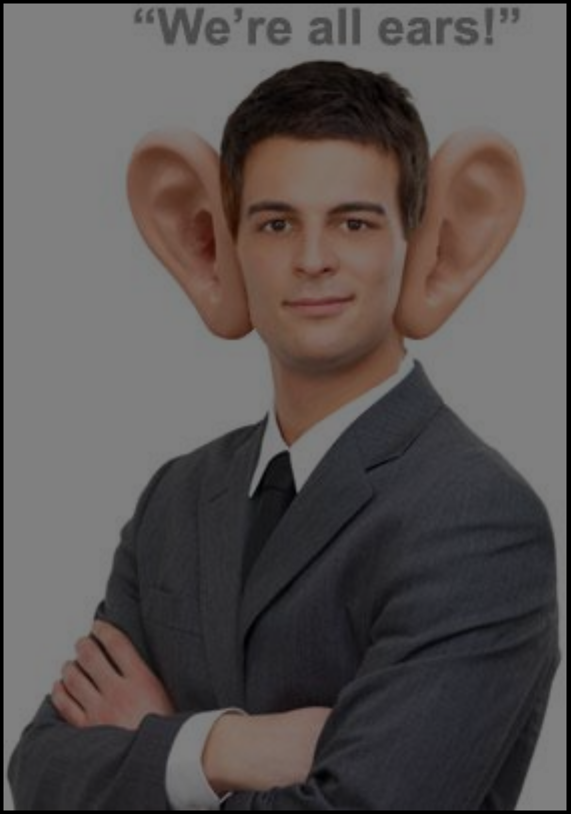
- GUIs in Java is just an example of a more general and very important programming technique: the Events Oriented Programming.
- In a program everything is sequential: the time when each action is going to happen is predictable…
- …How can we take into account those events in the world outside our program that we don't know exactly when will happen?
  - When will that door open?
  - When will this pot of water boil?
  - When will the user push this button?
- Programs have mechanisms to react ("wake up") when specific events take place outside the program.

Code organization

- Everything explained about GUIs is under the principles and rules of the OO programming paradigm...
- ...so everything we know about OO up to now is perfectly valid here.
- We have just added new pieces to the mecano...
  - ...that can be mixed with the rest in the way we consider most suitable.
- Examples:
  - Creating the listeners as independent classes.
  - Creating the listeners as inner classes.
  - Making the graphical components themselves act as listeners.
  - Associating a listener to more than one graphical component.

Adapters

- Some listeners interfaces have lots of methods…
- …and we will have to implement them all (listeners are interfaces).
- Adapters are classes that implement all the methods of a specific listener.
- Being classes, we just have to extend them rewriting the methods we need.
- For every Listener interface, there is an Adapter class:
  - WindowListener → WindowAdapter
  - KeyListener → KeyAdapter
  - MouseListener → MouseAdapter


“We’re all ears!”