**ItemEvent and ItemListener**

* An event of type **ItemEvent** is generated when a source such as a checkbox is clicked to check/uncheck it or when a list item is clicked.
* A class to listen and respond to an event of type, **ItemEvent**, must implement an interface, **ItemListener**.

**Method of ItemListener**

|  |  |
| --- | --- |
|  |  |

**public void itemStateChanged(ItemEvent e) :** Invoked when an item has been selected or deselected by the user

## Some methods of ItemEvent class

|  |  |
| --- | --- |
| **Method** | **Description** |
| **public Object getItem()** | Returns the item which was clicked and triggered the **ItemEvent.** |
| **public ItemSelectable getItemSelectable()** | Returns the item which was clicked. |
| **public int getStateChange()** | Returns the current state of item which was clicked. |

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class ItemEx1 implements ItemListener

{

JFrame jf;

Checkbox chk1, chk2;

Label label1;

ItemEx1()

{

jf= new JFrame("Checkbox");

chk1 = new Checkbox("Happy");

chk2 = new Checkbox("Sad");

label1 = new Label();

jf.add(chk1);

jf.add(chk2);

chk1.addItemListener(this);

chk2.addItemListener(this);

jf.setLayout(new FlowLayout());

jf.setSize(220,150);

jf.setVisible(true);

jf.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

public void itemStateChanged(ItemEvent ie)

{

Checkbox ch =(Checkbox)ie.getItemSelectable();

if(ch.getState()==true)

{

label1.setText(ch.getLabel()+ " is checked");

jf.add(label1);

jf.setVisible(true);

}

else

{

label1.setText(ch.getLabel()+ " is unchecked");

jf.add(label1);

jf.setVisible(true);

}

}

public static void main(String... ar)

{

new ItemEx1();

}

}

**WindowEvent and WindowListener**

An event of type **WindowEvent** is generated in such situations -

* When a window is activated for the first time.
* When a window is minimized.
* When a window is brought up back from minimized state.
* When the close button **(x)** of window is clicked to close it.
* **WindowListener** interface methods -

|  |  |
| --- | --- |
| **Method** | **Description** |
| **public void windowOpened(WindowEvent e)** | This method is called when a window is opened for the first time. |
| **public void windowActivated(WindowEvent e)** | This method is called when a window shows up on screen. |
| **public void windowDeactivated(WindowEvent e)** | This method is called is no longer the window in use or active. |
| **public void windowIconified(WindowEvent e)** | This method is called when a window is changed from a normal to a minimized state. |
| **public void windowDeiconified(WindowEvent e)** | This method is called when a window is brought up on the screen from a minimized state. |
| **public void windowClosing(WindowEvent ke)** | This method is called a user clicks on the (x) icon to close the window. |
| **public void windowClosed(WindowEvent e)** | This method is called when a window has been closed. |

**What is Key Listener**

The basic purpose of the Key Listener interface is for when an operation is performed on the keyboard, when a key is pressed or released the Key Listener interface is invoked.

**Methods of Key Listener**

* **KeyTyped(KeyEvent)**

This method is invoked just after the user types a Unicode character.

* **KeyPressed(KeyEvent)**

This method is called just after the user presses the key.

* **KeyReleased(KeyEvent)**

This method is called just after the user releases the key.

## some methods of KeyEvent class

|  |  |
| --- | --- |
| **Method** | **Description** |
| **public char getKeyChar()** | Returns the character associated with the key pressed on the keyboard, which triggered the **KeyEvent.** |
| **public int getKeyCode()** | Returns an int key code associated with the key pressed on the keyboard. |
| **public boolean isActionKey()** | Returns true if key pressed was an "action" key, i.e. keys that don't generate a character, such as **Cut, Copy, Paste, Page Up, Caps Lock, the arrow and function keys.** |

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class KeyEx1 implements KeyListener

{

Label label1, label2;

TextField field1;

JFrame jf;

String str;

KeyEx1()

{

jf = new JFrame("Handling KeyEvent");

label1= new Label("Press any key on keyboad, to see the events it generates -", Label.CENTER);

label2= new Label();

field1 = new TextField(20); //calling TextField(String)

jf.setLayout(new FlowLayout());

jf.add(label1);

jf.add(field1);

jf.add(label2);

field1.addKeyListener(this); //As soon as button is clicked, data from all the textfields is read

jf.setSize(360,200);

jf.setVisible(true);

}

public void keyPressed(KeyEvent ke)

{

str= "KeyCode : " + ke.getKeyCode() + ", -Key Pressed- ";

label2.setText(str);

jf.setVisible(true);

jf.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

public void keyReleased(KeyEvent ke)

{

str+=" -Key Released- ";

label2.setText(str);

jf.setVisible(true);

str="";

}

public void keyTyped(KeyEvent ke)

{

str+=" -Key Typed- ";

label2.setText(str);

jf.setVisible(true);

}

public static void main(String... ar)

{

new KeyEx1();

}

}