

GListener interface

De Wiki

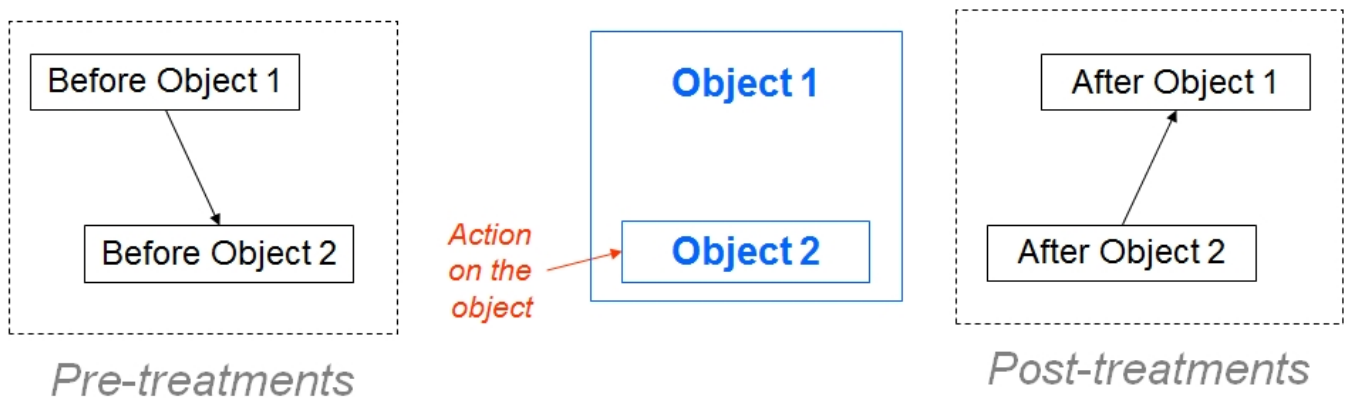
Aller à : [navigation](#), [rechercher](#)
[GListener interface](#)

before() and after() methods

To manage actions on widgets, a single interface is available: [GListener](#). Then, automatically, you must implement these both methods :

- [before\(\)](#)
- [after\(\)](#)

Indeed, the [GListener](#) interface allows to manage notion as **after** (something like a call-back) but also **before**, in a more friendly way than what is proposed by swing with an automatic download/upload management of the pile as described in the diagram just below.



We will go inside the [after\(\)](#) method after any action with the left button of the mouse or the "Enter" key of the keyboard.

The [before\(\)](#) action may be a bit more difficult to understand. Of course, there is now `//before()` action for a button widget (even if the [GListener](#) interface implies to get it). It is more clear for a [GEntryInt](#) widget as described with the example below: we will enter in the [before\(\)](#) method just before the value of the widget will be modified (as [after\(\)](#) will be activated after the value will be changed).

```
public void before(GEvent e) throws GException {  
    System.out.println("Old value was "+real.getValue());  
}  
public void before(GEvent e) throws GException {  
    System.out.println("New value is "+real.getValue());  
}
```

Imagine that we initialized the real value to "1.0", then enter "2.0" at the keyboard. After pushing on the "Enter" key, we will obtain on the console ...

Old value was 1.0

New value is 2.0

How to know which widget has been selected ?

Moreover, we see on the previous example that, these both methods will have as an input argument a [GEvent](#) objet that will correspond to the event occured. Thus, this objet will contain all the information linked to this event.

In order to implement correctly the [before\(\)/after\(\)](#) methods, we must also know how to discriminate which widget has been activated. Indeed, in case of a [GPanel](#) , containing several widgets, it is necessary to know it, as the action to do will surely depend on the selected widget. So, to do it, **GENIUS** gives us the [contains\(\)](#) method associated to the [GEvent](#) object given by the [before\(\)/after\(\)](#) methods. This method needs as input arguments one or several widgets and will return true if one of these widgets have been activated (else false).

```
private GButton but1;
private GButton but2;

...

public void after(GEvent e) {
    if ( e.contains(but1) ) { // Case of we push on the but1 button ...
        ...
    }
    if ( e.contains(but1, but2) ) { // Case of we push on the but1 button or
the but2 button ...
        ...
    }
}
```

But sometimes, we just want to recover the activated object itself without testing all the possibilities ! It can simply done using the [getLocalSource\(\)](#) method : it will return the selected widget known "locally", meaning existing at the current [GPanel](#) level.

But, imagine that we are inside a [\[1\]](#) level **P0**, that includes two other [GPanel](#) **P1** and **P2** with **P1** including two buttons, **B1** and **B2** ...

```
P0 => P1 => B1
      => B2
      => P2
```

If we push on the **B2** button, it is possible to get the object corresponding to **B2** by using the [getFinalSource\(\)](#) method that will return it.

So, [getLocalSource\(\)](#) will return **P1** as [getFinalSource\(\)](#) will return **B2**.

With such a mechanism, it will possible, for example, to discriminate if the action on a [GEntryReal](#) widget was coming from the input area or just from the unit menu (as, in that case, the value has not

been changed).

```
public void after(GEvent e) {
    if ( e.getFinalSource() instanceof GNumberFieldAbstract ) {
        System.out.println("Value changed ...");
    }
    else {
        System.out.println("Only unit change ...");    }
}
```

Example

In the following example, we will show:

1. how to discriminate which widget has been activated (using the [contains\(\)](#) method)
2. how to use the [before\(\)](#) and [after\(\)](#) methods to manage the interdiction to enter negative values for an integer

```
public class myPanel extends GPanel implements GListener {

    private GButton but1;
    private GButton but2;
    private GEntryInt valI;

    private int oldInt; // Integer to keep in memory the old value

    public myPanel() {
        but1 = new GButton("Bouton 1");
        but2 = new GButton("Bouton 2");
        valI = new GEntryInt("Integer input:", 0);
    }

    public void generic() throws GException {
        put(but1);
        put(but2);
        put(valI);
    }

    public void display() throws GException {
        generic();
    }

    public void before(GEvent e) {
        if ( e.contains(valI) ) {
            // We store the value before to be modified ...
            oldInt = valI.getValue();
        }
    }
}
```

```
public void after(GEvent e) {  
  
    if ( e.contains(but1) ) { System.out.println("Click occurred on button  
1"); }  
  
    if ( e.contains(valI) ) {  
        // Test to keep only positive values  
        if ( valI.getValue() < 0 ) {  
            System.out.println("Warning, this value may not be negative ...");  
            // We recover the old value  
            valI.setValue(oldInt);  
        }  
    }  
  
}
```

[Return to the introduction](#) ↑ [Go to the next page](#) →

Récupérée de « http://genius.cnes.fr/index.php?title=GListener_interface&oldid=409 »

Menu de navigation

Outils personnels

- [18.222.67.251](#)
- [Discussion avec cette adresse IP](#)
- [Créer un compte](#)
- [Se connecter](#)

Espaces de noms

- [Page](#)
- [Discussion](#)

Variantes

Affichages

- [Lire](#)
- [Voir le texte source](#)
- [Historique](#)
- [Exporter en PDF](#)

Plus

Rechercher

GENIUS

- [Welcome](#)
- [Quick Start](#)
- [News](#)

Basic principles

- [GFrame and GPanel](#)
- [Main widgets](#)
- [Links with Swing](#)
- [GLayout](#)
- [Conditional Display](#)
- [GListener interface](#)

More deeper in the concept

- [Units management](#)
- [GContainer](#)
- [GReadWrite interface and data files management](#)
- [Modified data](#)
- [Process management](#)

Still more ...

- [Validity controls](#)
- [Menu bar](#)
- [Icons](#)
- [GClear interface](#)

Still more again ...

- [Tooltips](#)
- [Shortcuts](#)
- [Copy & paste](#)
- [Plots](#)
- [Results File Management](#)
- [GPlotPanel](#)
- [GGroundPlotPanel](#)
- [Internationalization](#)

- [Log file](#)
- [Update data](#)

Some other widgets

- [GTabbedPane](#)
- [GTable1D](#)
- [GTable2D](#)
- [GComponentList](#)
- [GDialog and GDetachedPanel](#)
- [GContextFileManagement](#)
- [How to build a standard application](#)
- [GPanTest](#)
- [Create your own widget](#)

Evolutions

- [Main differences between V1.11.4 and V1.12.1](#)
- [Main differences between V1.10.1 and V1.11.4](#)
- [Main differences between V1.10 and V1.10.1](#)
- [Main differences between V1.9.1 and V1.10](#)
- [Main differences between V1.9 and V1.9.1](#)
- [Main differences between V1.8 and V1.9](#)
- [Main differences between V1.7 and V1.8](#)
- [Main differences between V1.6.2 and V1.7](#)
- [Main differences between V1.6.1 and V1.6.2](#)
- [Main differences between V1.6 and V1.6.1](#)
- [Main differences between V1.5 and V1.6](#)
- [Main differences between V1.4.1 and V1.5](#)
- [Main differences between V1.3 and V1.4.1](#)

Training

- [Training slides](#)
- [Tutorials package for V1.12.1](#)
- [Tutorials package for V1.11.4](#)
- [Tutorials package for V1.10.1](#)
- [Tutorials package for V1.9.1](#)
- [Training & tutorials package for V1.8](#)
- [Training & tutorials package for V1.7](#)
- [Training & tutorials package for V1.6](#)

Links

- [CNES freeware server](#)

Outils

- [Pages liées](#)
- [Suivi des pages liées](#)
- [Pages spéciales](#)
- [Adresse de cette version](#)
- [Information sur la page](#)
- [Citer cette page](#)

- Dernière modification de cette page le 10 juillet 2017 à 07:54.
- [Politique de confidentialité](#)
- [À propos de Wiki](#)
- [Avertissements](#)

