CENTRE NATIONAL D'ÉTUDES SPATIALES

# GENIUS V1.9.1

## GENeration of Interface for Users of Scientific S/W

## Formation

## Wiki on http://genius.cnes.fr

CENTRE NATIONAL D'ÉTUDES SPATIALES

# GENIUS

## GENeration of Interface for Users of Scientific S/W

## Background

# Context (1/2)

- **Since the 90's, CNES Flight Dynamics teams has developed specific means to build GUI for their own experts and/or operation tools. These tools were GENESIS/MADONA/XTRACE used for example for the FDS (Flight Dynamic Subsystem) of the ATV-CC (ATV Control Center)**

- **From 2012, following the choice of new *Java* developments for FD tools (thanks to SIRIUS/PATRIUS project), a first mock-up named GENIUS was done internally using some basic GENESIS principles**

- **By the end of 2013, a specific study was done. Its output was:**
  - **A requirement specification thanks to previous GUI feedbacks,**
  - **A recommendation to develop a specific tool as no commercial items answered to our needs,**
  - **Another prototype in order to get an alternative to GENIUS (even if some concepts were common to both of them).**

# Context (2/2)

- **Main differences between GENIUS and the prototype was:**
  - **GENIUS:**
    - Direct interfacing with business data (**PATRIUS** ones for example)
    - 100% Java code approach
  - **Prototype:**
    - Data model independent of the display and the business data (**MVC** model)
    - A code generation approach

- **In January 2014, both products have been presented to a pool of representative users and the choice fell on GENIUS !**

# Why GENIUS (1/4) ?

- In Java world, basic tools, as *swing*, may become relatively complex to use because it stays at a certain low level (on the opposite, it allows to do a lot of things).

- Moreover, *GUI* for flight dynamics tools (or, more generally, scientific tools) need most of the time :
  - To enter input (numerical) data from the screen or the keyboard
  - To read / write these data into files
  - To execute computation thanks to these data,
  - To visualize results

- GENIUS, as previously GENESIS/MADONA/XTRACE, is a higher level layer based on *swing* but allowing to create more easily such *GUI*.

# Why GENIUS (2/4) ?

- Advantages coming from GENESIS and kept with GENIUS
  - Simplified approach, in particular about events management (BEFORE, AFTER)

    => almost identical approach (even simpler …)
  - Performing conditional display

    => identical approach
  - Read / write for files directly integrated

    => almost identical approach
  - Units management

    => almost identical approach

# Why GENIUS (3/4) ?

- **GENESIS drawbacks … versus GENIUS advantages**
  - **Specific language => learning problem, confusion with Fortran and mainly need of a code generation very time consuming**

    **=> fully written in JAVA (absolutely no generation)**
  - **An object approach (mandatory) that might be quite disturbing for people using Fortran**

    **=> fully written in JAVA (then consistency with an object approach )**
  - **Scalability versus optional arguments, so relatively limited**

    **=> use of heritage and possibility of direct *swing* functionalities**
  - **Process management only compatible of *UNIX/LINUX* world**

    **=> portability thanks to JAVA**

# Why GENIUS (4/4) ?

## ❑ Now available outside CNES:

- Open Source (Apache 2.0 licence) via **https://logiciels.cnes.fr**
- Wiki on **http://genius.cnes.fr**
- Contact **genius@cnes.fr**

## ❑ CNES internally:

- Download via Artifactory : **https://tu-dctsb-p02.cst.cnes.fr:8443/artifactory/webapp/browserepo.html**

## ❑ Several applications :

- PSIMU, ELECTRA, OPERA, MIPELEC (also available outside CNES)
- SIRENA, CRASH, DOORS, OSCAR/DRAGON, PAMPERO, CRABIM (CNES only)

**… using also GENOPUS**

CENTRE NATIONAL D'ÉTUDES SPATIALES

# GENIUS

## GENeration of Interface for Users of Scientific S/W

## Basic principles

# GFrame

- We find the same principles as those used by *swing* with classes as:
  - GFrame
  - GPanel
  - …
  - GButton

- About GFrame, nothing particular, except the display() method which allows the display more easily.

*GPanel (cf. following slide)*

```
GFrame frame = new GFrame("Gex1", pan);

frame.display();
```

# GPanel (1/2)

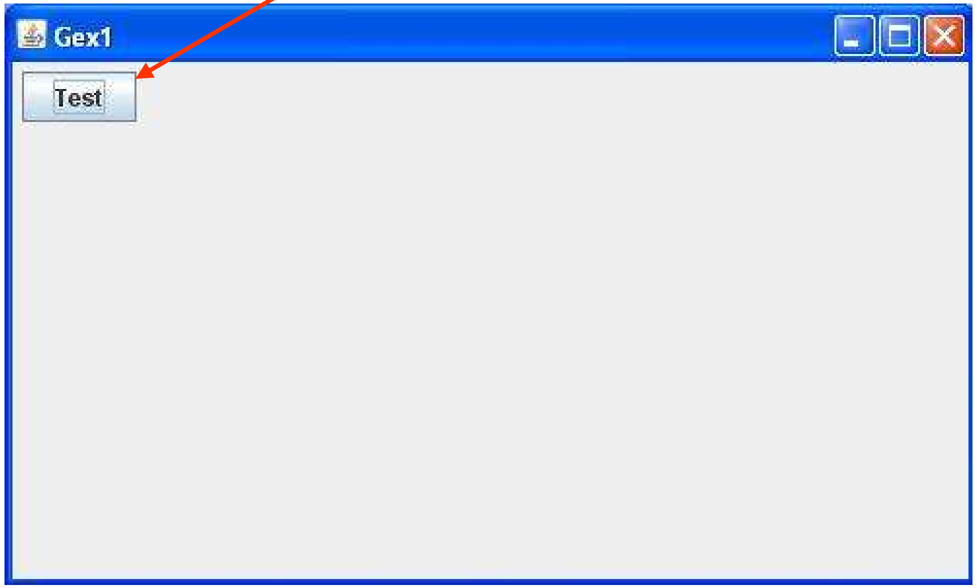
- **GPanel object is a bit more « complex » because, when created, it is necessary to implement both following methods: generic() and display()**
  - **display() method will indicate which graphical objects will be displayed**
    - By these means, it is up to GENIUS to automatically manage refresh ; (no need to call to a « *refresh* » method);
    - To decide what will be displayed, we only need to call in this method, the put method with the object as argument : put(objectName).
  - **generic() method  allows to indicate which graphical objects will be concerned for displaying  … but also for reading or writing into files (see later …)**
  - **Another solution is then to store calls to the put method into generic() and, inside display() method, only calling the generic() method …**

# GPanel (2/2)

```
GPanel pan = new GPanel() {

   GButton but = new GButton("Test");

   public void display() {
       put(but); }

   public void generic() { }

};
```

*… using display ()*

GButton



```
GPanel pan = new GPanel() {

   GButton but = new GButton("Test");

   public void display() {
       generic(); }

   public void generic() {
       put(but); }

};
```

*… using generic ()*

# Basic widgets

- As with GENESIS , we find again basic classes needed to build a scientific tool *GUI* … in particular entering real data with units !!!
  - GButton, GHyperlinkLabel
  - GLabel, GImage, GSeparator
  - GRadioButton, GCheckBox, GChoice, GMultipleChoice
  - GComboBox, GComboBoxWithLabel
  - GList , GPopupList , GPopupListWithLabel , GTree
  - GEntryReal, GEntryInt, GEntryString, GDate
  - GSliderWithLabel, GSliderRealWithLabel
  - GTextArea (text over several lines), GConsole
  - GEntryRealVector, GEntryIntVector, GEntryDateVector
  - GTable1D, GTable2D, GComponentList
  - GMenuBar, GMenu, GMenuItem
  - And many more …

# Links with swing

- **GENIUS classes use *swing* classes but are not directly inherited from them:**
  - **Strictly speaking, it is not recommended to directly use swing classes …**
    - For example JPanel rather than GPanel because, in that case, the « *display* » mechanism will not be effective.
  - **For certain methods, an over layer is proposed by GENIUS …**
    - For example, setEnabled(true/false) method is applicable for a GButton object.
  - **But, in order not to be blocked, one can have direct calls to *swing* methods:**
    - By a call to the *swing* object included in the **GENIUS** one as for example, with the getJButton() method which refer to the swing **J**button object using by **G**Button;
    - By a call to *swing* widgets which do not need **GENIUS** mechanism as JOptionPane or JFileChooser.

CENTRE NATIONAL D'ÉTUDES SPATIALES

# GENIUS

## GENeration of Interface for Users of Scientific S/W

## First exercise

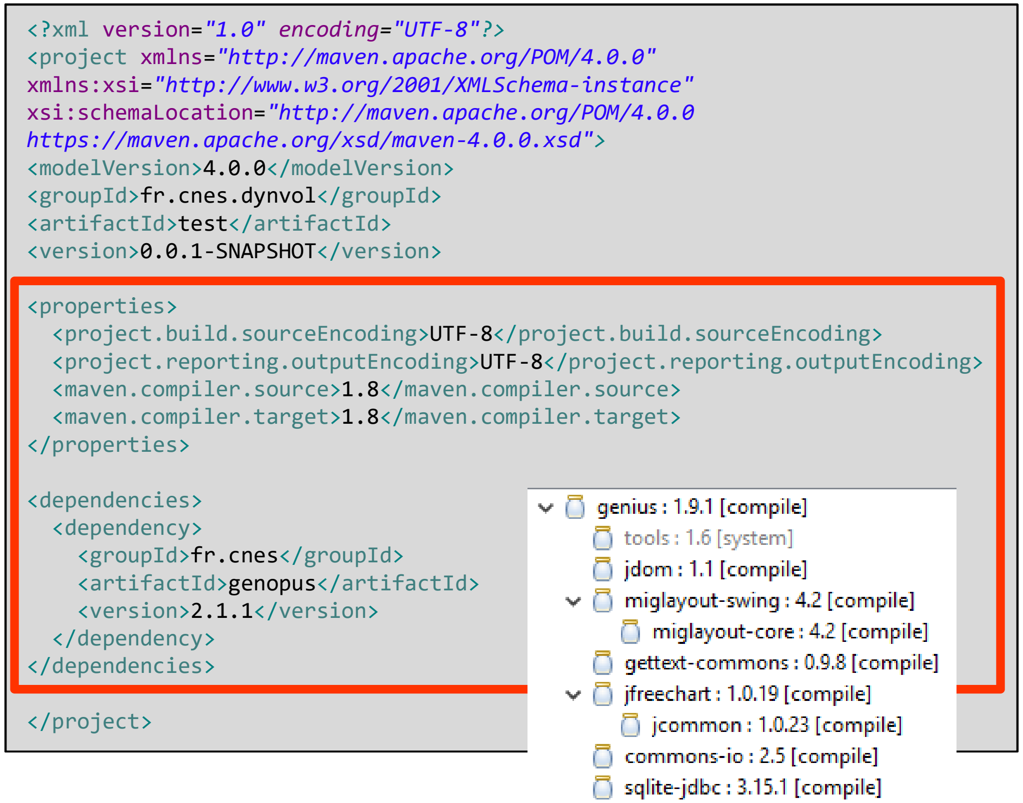
# Exercise 1 (1/2)

- **Launch Eclipse:**
  - Create a specific workspace for example under Documents

- **Create a Simple Maven Project :**
  - File / New / Other … / Maven Project
  - Create a Simple project
  - GroupId: xx.yy.zz…
  - Artifact Id : FormationGenius

- **Be sure to be with a *1.8 Java* version**

- **… and link with GENIUS by adding these lines inside the pom.xml file and save it …**

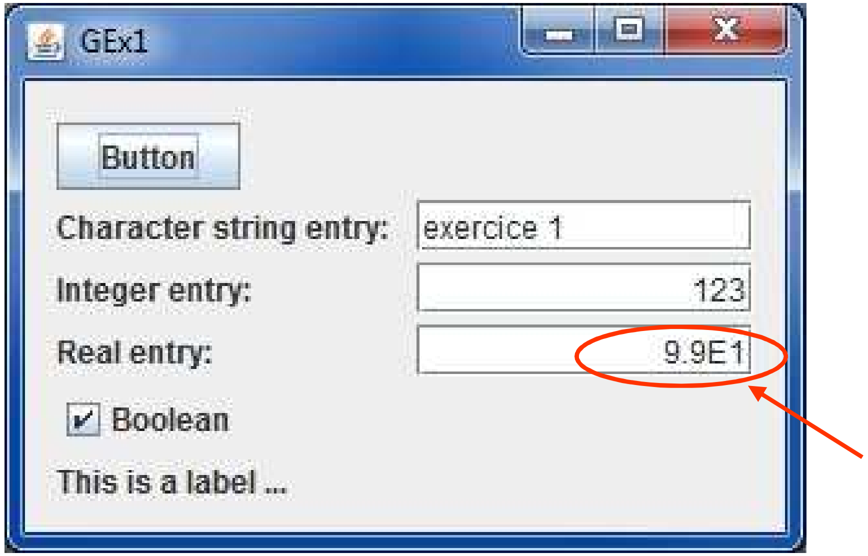


```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>fr.cnes.dynvol</groupId>
<artifactId>test</artifactId>
<version>0.0.1-SNAPSHOT</version>

<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
  <maven.compiler.source>1.8</maven.compiler.source>
  <maven.compiler.target>1.8</maven.compiler.target>
</properties>

<dependencies>
  <dependency>
    <groupId>fr.cnes</groupId>
    <artifactId>genopus</artifactId>
    <version>2.1.1</version>
  </dependency>
</dependencies>

</project>
```

```
genius : 1.9.1 [compile]
  tools : 1.6 [system]
  jdom : 1.1 [compile]
  miglayout-swing : 4.2 [compile]
    miglayout-core : 4.2 [compile]
  gettext-commons : 0.9.8 [compile]
  jfreechart : 1.0.19 [compile]
    jcommon : 1.0.23 [compile]
  commons-io : 2.5 [compile]
  sqlite-jdbc : 3.15.1 [compile]
```

# Exercise 1 (2/2)

- **Create a *GUI* with:**
  - **A button**
  - **An entry area for strings**
  - **An entry area for an integer**
  - **An entry area for a real**
  - **A checkbox**
  - **A label**



Character string entry: exercice 1

Integer entry: 123

Real entry: 9.9E1

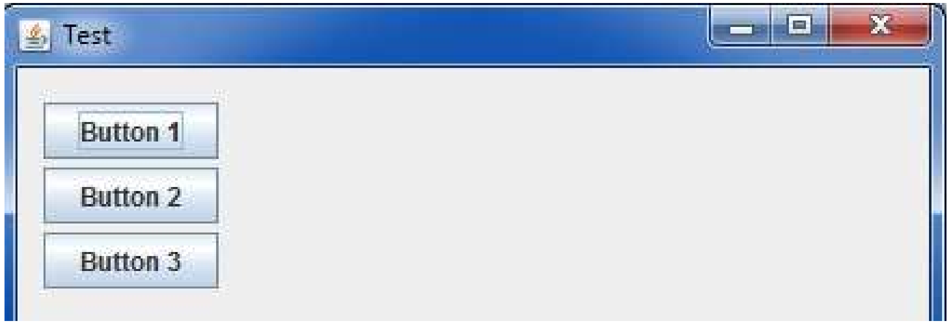
☑ Boolean

This is a label ...

*Possibility to change the format with a right click*

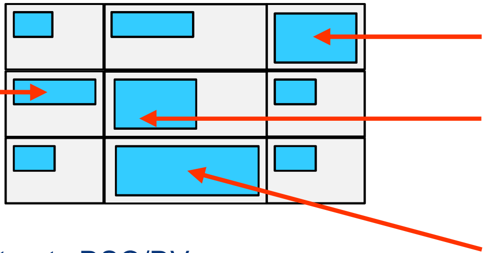
# GLayout (1/5)

- **GENIUS gives a specific Layout (based on MigLayout) well adapted to conditional display**
  - By default, every new graphic widget will be set to the next line,



```
GPanel pan = new GPanel() {

   GButton but1 = new GButton("Bouton 1");
   GButton but2 = new GButton("Bouton 2");
   GButton but3 = new GButton("Bouton 3");

   public void display() throws GException {
       put(but1);
       put(but2);
       put(but3);
 }

   public void generic() {
   }

};
```

  - Based on a grid cell => be careful, the size of a cell may depend on another component situated below …



*Determine column 1 size*

*Determine line 1 and column 3 sizes*

*Determine line 2 size*

*Determine line 3 and column 2 sizes*

# GLayout (2/5)

## ■ Some available « constraints »:

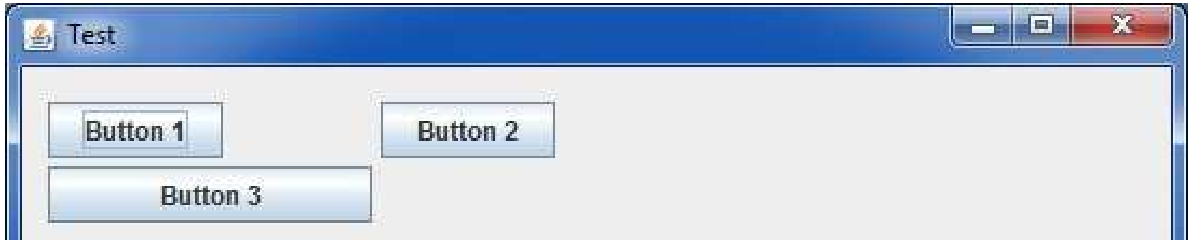
| | |
|---|---|
| **wrap [*gapsize*]** | Go to the next line after the component (gapsize => amount of pixel after it) |
| **newline [*gapsize*]** | Go to the next line before the component (gapsize => amount of pixel before it) |
| **skip [*count*]** | Skip one or several columns (depending of the value of count, 1 by default). |
| **span [*countx* [*county*]]**<br>**spanx [*countx*]**<br>**spany [*county*]** | Allows to the component to spread on several cells (countx for horizontal axis and county for vertical axis) |
| **split [*count*]** | Allows to put several components on a single cell. |
| **flowx, flowy** | Direction when a component is added (flowx by default) |
| **height, width *size*** | Specify the height (resp. width) of the component in pixel (preferred size). |
| **push (pushx, pushy)** | « Push » the next components (visible when the main window is enlarged) |
| **grow (growx, growy)** | Fill the cell with the component. |
| **gap *left* [*right*] [*top*] [*bottom*]**<br>**gaptop, gapbottom, gapleft, gapright [*gap*]** | Specify the gap  (in pixels by default). |
| **align [*alignx, aligny*]**<br>**alignx, aligny [*align*]** | Specifiy alignment: (left, center, right) for alignx and (top, center, bottom) for aligny |

# GLayout (3/5)

- **Two ways to access to the MigLayout constraints :**
  - **The “old fashion” (used in versions prior to V1.2) by calling now the setStringConstraint method that wait for a String as single argument**
    - Not detailed here … and now **obsolete**
  - **The “new fashion” by calling now the setConstraint method, waiting for:**
    - A GConstraint object
    - With arguments given by static methods proposed by GConstraint

*No more “by default” way => on the same line*

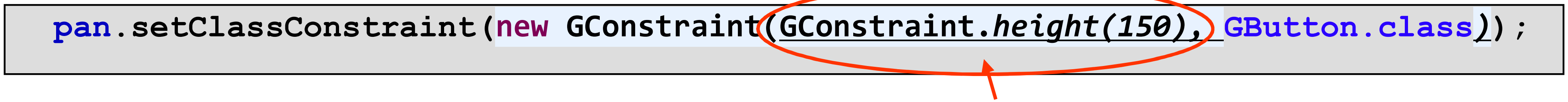
```
  but2.setConstraint(null);
  but3.setConstraint(new GConstraint( GConstraint.newline() , GConstraint.width(150) ));
```



*width 150 : button width fixed to 150 pixels*

# GLayout (4/5)

- We can also take into account all the objects of a given type included in a **GPanel** by calling another **setClassConstraint** method:



```
pan.setClassConstraint(new GConstraint(GConstraint.height(150), GButton.class));
```

*height 50 : height of all the buttons of the panel fixed to 50 pixels*

- Management of some complex widgets as **GEntryReal** may be more confuse than for a simple **GButton** as this widget is composed of several other sub widgets :
  - Sub widget **0**: GLabelWithIndicator
    - Sub widget **0.0**: GLabel
    - Sub widget **0.1**: GIndicator (the "*" when the value is modified)
  - Sub widget **1**: GTextField
  - Sub widget **2**: GUnit

# GLayout (5/5)

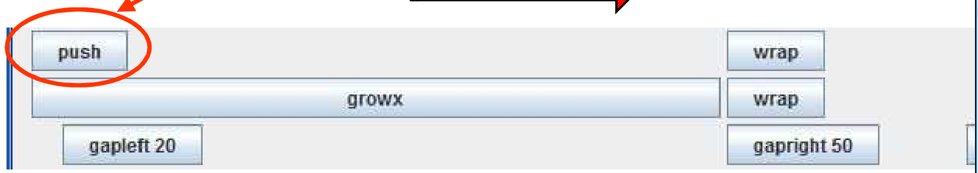
- With the old fashion, it was possible (more or less easily) to access to such sub widgets using “|”, “?”, “+” syntax …
- With the new API, it is easier to explain it with the **setInnerDescendantContraint()** method (and **setInnerDescendantClassContraint()** ),
- For example, if we want:
  - to apply “newline” to the global widget
  - to set 200 pixels for the textfield width
  - to skip a cell to the GUnit field

```
GEntryReal real = new GEntryReal(...);
// Applying "newline" to the GLabel
real.setInnerDescendantConstraint(new GConstraint(GConstraint.newline(), 0, 0);
// Applying "width 200" to the texfield
real.setInnerDescendantConstraint(new GConstraint(GConstraint.width(200)), 1);
// Applying "skip"to the texfield
real.setInnerDescendantConstraint(new GConstraint(GConstraint.skip(1)), 2);
```

# Exercise 2

- Use the **setConstraint** method to build (part of) the following ***GUI***:

If we enlarged the window



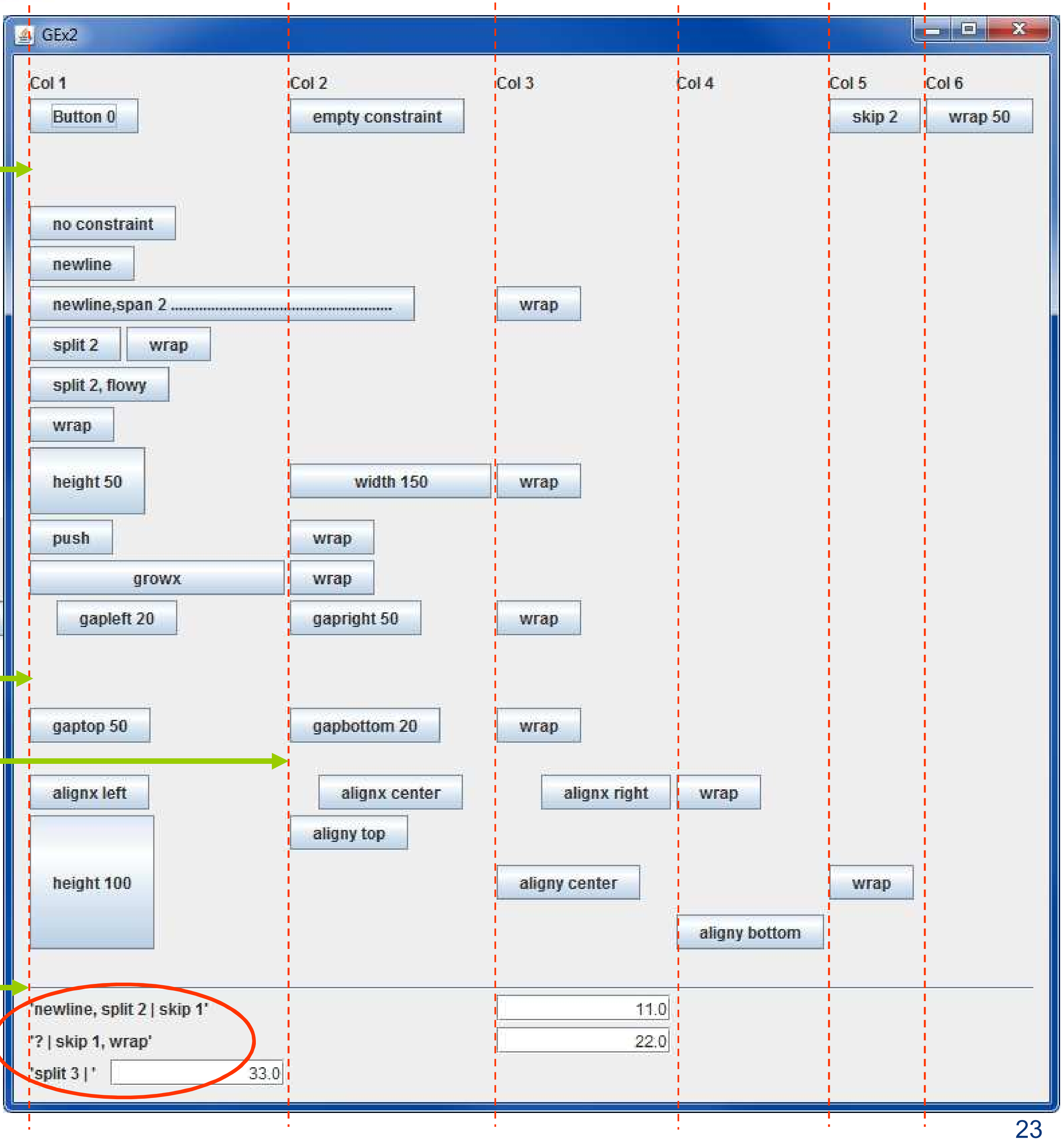
wrap 50

gaptop 50

gapbottom 20

"newline, gaptop 20, growx, spanx 99"

*Special management for widgets with multiple components*

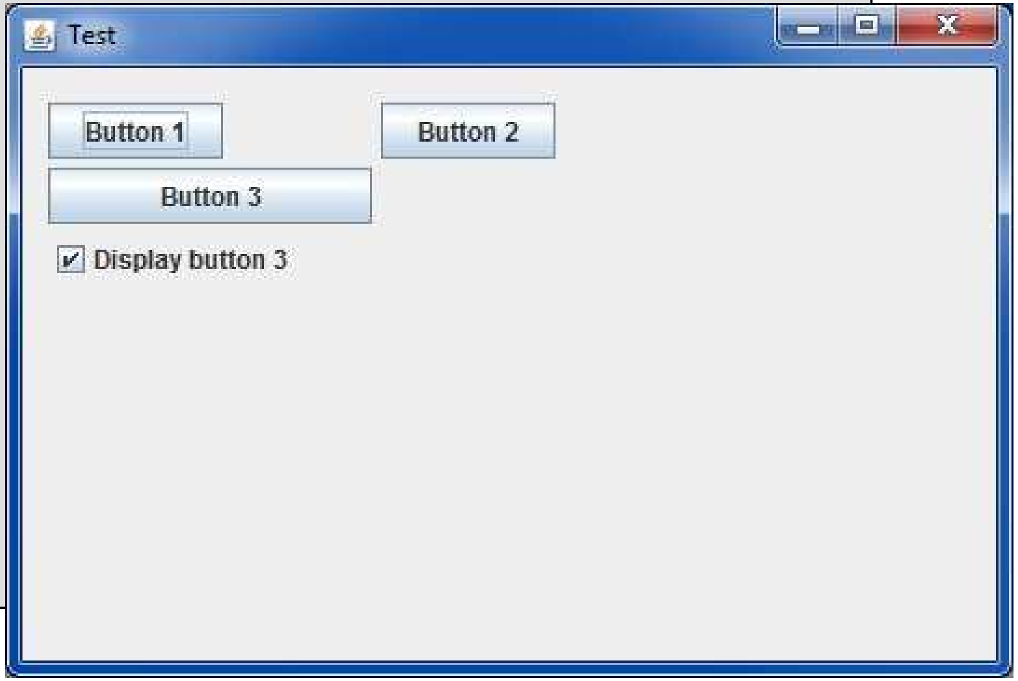


# Conditional display

- Conditional display is simply managed with « if » or « switch » and using the generic() or the display() method:

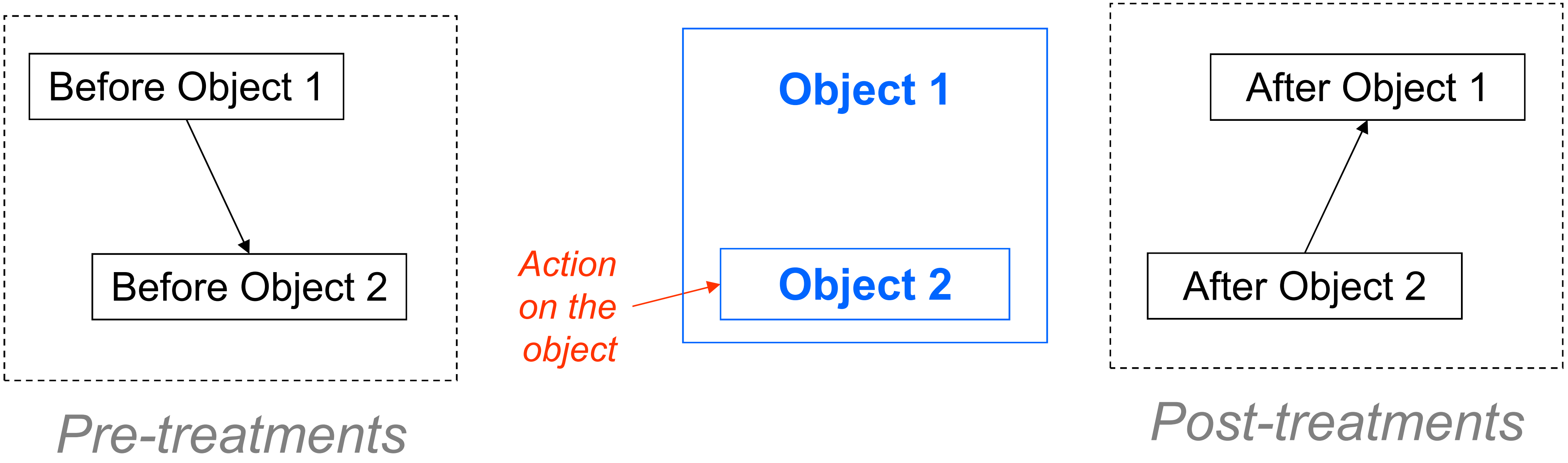
```
public class myPanel extends GPanel {

  GButton but1;
  GButton but2;
  GButton but3;
  GCheckBox cb;

  ...

  public void generic() throws GException {
    put(but1);
    put(but2);
    if ( cb.isSelected() ) { put(but3); }
    put(cb);
  }
```

Simple not ?



# GListener  Interface (before / after)

- To manage actions on widgets, a single interface is avalaible: GListener
  - It allows to manage notion as before / after in a more friendly way than what is proposed by swing (download/upload management of the pile)



# GListener Interface (Before / After) c’td …

```java
public class myPanel extends GPanel implements GListener {

  GButton but1;
  GButton but2;
  GButton but3;

  public myPanel () { ... }

  public void generic() { ... }

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

  public void before(GEvent e) {
  }

  public void after(GEvent e) {
     if ( e.contains(but1) ) { System.out.println("Bouton 1"); }
  }
```

# GListener Interface (sources)

- **GENIUS provides a contains() method associated to a GEvent object : this method needs as input arguments one or several widgets and will return true if one of these widgets have been activated (else false).**

```
if ( e.contains(but1) ) { // Case we push on the but1 button ...
  ... }
if ( e.contains(but1, but2) ) {// Case we push on but1 or but2 buttons ...
  ... }
```

- **If we just want to recover the activated object itself, it can simply done using the getLocalSource() method : it will return the selected widget known “locally”, meaning existing at the current level.**

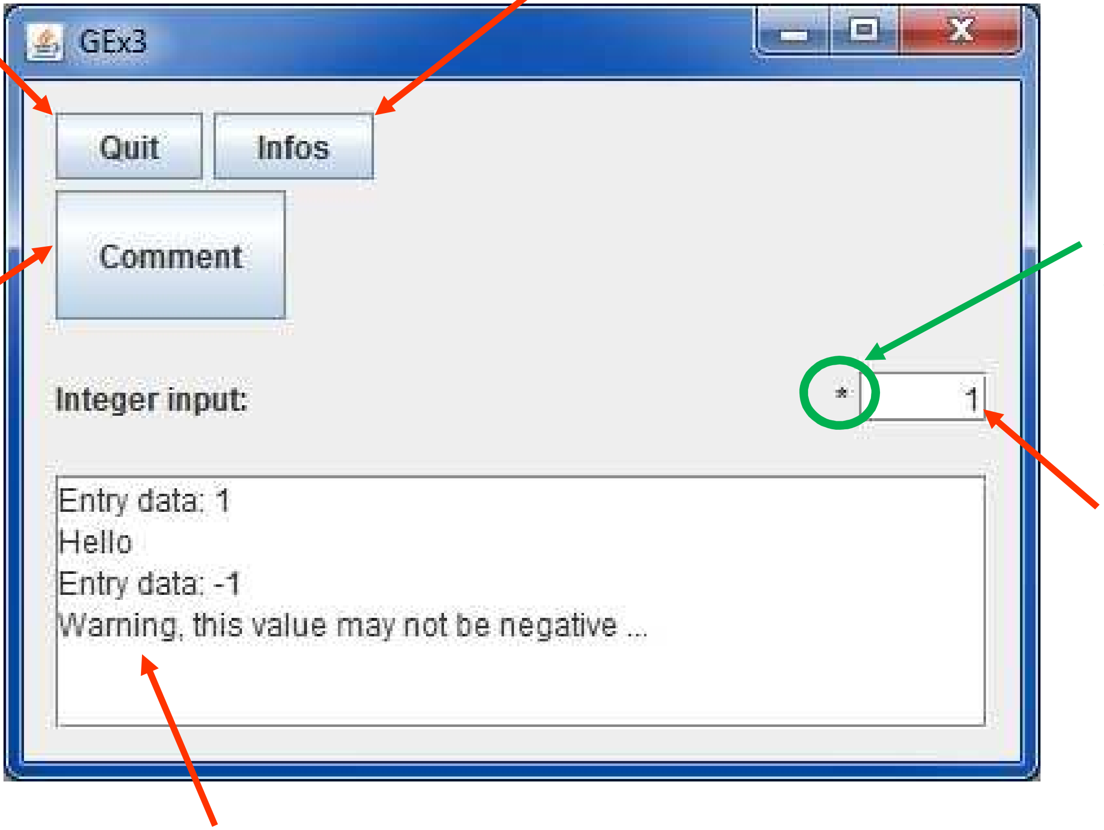
- **If we are inside a GPanel P0, that includes two other GPanel P1 and P2 with P1 including two buttons, B1 and B2 …**
  - **If we push on the B2 button, it is possible to get the object corresponding to B2 by using the getFinalSource() method that will return it.**
  - **So, inside P0:**
    - **getLocalSource() will return P1**
    - **getFinalSource() will return B2**

# Exercise 3

- **Create the following *GUI*:**

Quit the application

Display a modal detached window where it is written:
GENIUS Formation
EXERCICE 3
*(use JOptionPane.showMessageDialog)*



```
Entry data: 1
Hello
Entry data: -1
Warning, this value may not be negative ...
```

*Note that a "*" appears when data has changed*

Display « hello »
in the sub window below;
Disappear if the value of the integer is equal to 0

Send an error message if the input value is < 0 and display the previous value

Use GConsole

CENTRE NATIONAL D'ÉTUDES SPATIALES

# GENIUS

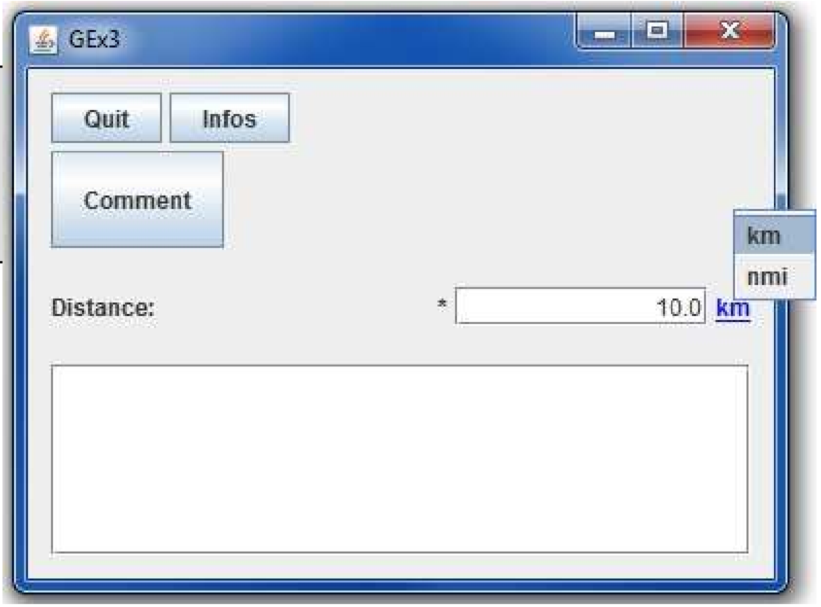
## GENeration of Interface for Users of Scientific S/W

## Continued …

# Units management

- Units are managed with the **GUnit** class or more directly with **GMetricUnit**.
- In case of using **GMetricUnit**, when we define a unit for a real value, it is stored automatically in the computer memory in **SI** (m, kg, rad …)

```
GUnit[] unitDis = { new GMetricUnit ("km") ,
                    new GMetricUnit ("nmi") };

dist  = new GEntryReal("Distance", 10000., unitDis);
```



Distance: * 10.0 km

- Thus, in that case, we may have a difference between what it is displayed (10.0) and what it is actually stored in the memory (10000.).

- To get the value stored in memory (same for an integer):

```
double val = dist.getValue(); // Always in SI
```

# GContainer (1/2)

- When we want to merge several basic widgets (for example several GEntryReal), we can encapsulate them inside a GPanel :
  - Advantage: directly displayed
  - Drawback: when created, we don’t know sometimes exactly how to display it

- Another solution is to put these objects inside a GContainer
  - Drawback: it is not possible to display it directly
  - Advantages:
    - Display management will be done by the final user
    - We may use this GContainer several times inside a same GPanel (for example several orbit parameters or several maneuvers laws)

# GContainer (2/2)

*Use this interface for display*

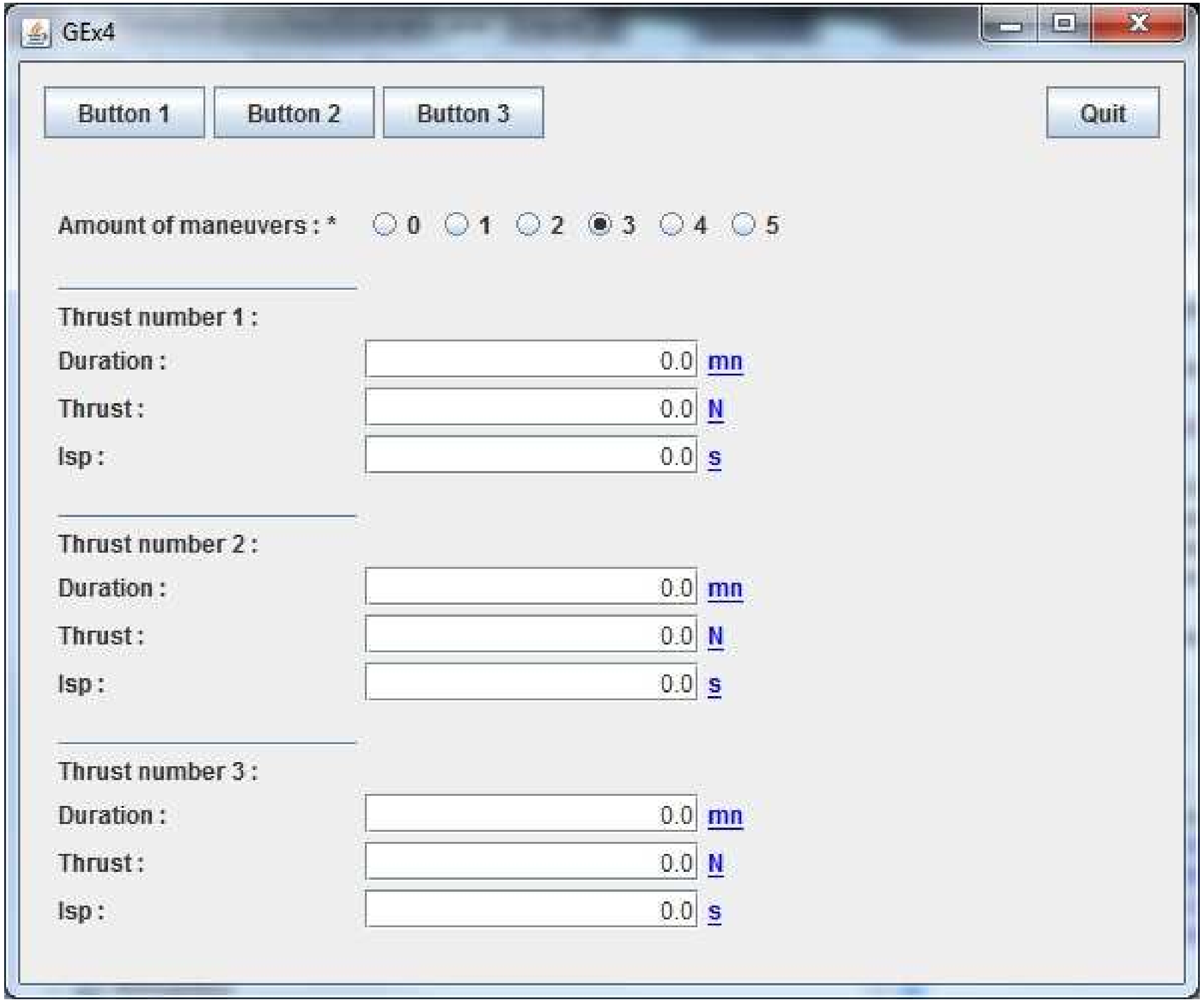
```
public class MyContainer extends GContainer implements GDisplay {

  GButton but1;
  GButton but2;
  GButton but3;

  public MyContainer () {
     but1 = new GButton(“Button1");
     but2 = new GButton(“Button 2");
     but3 = new GButton(“Button 3");
  }

  public void generic() throws GExcepti
     put(but1);
     put(but2);
     put(but3);
  }

  public void display() throws GExcepti
     generic();
  }

}
```

```
GPanel pan = new GPanel() {

  MyContainer cont = new MyContainer();

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

  public void generic() throws GException {
     put(cont);
  }

};
```

# Exercise 4 (1/2)

- **Create the following *GUI* using notions of:**
  - **GContainer**
  - **GUnit/GMetricUnit**
  - **setConstraint()**

***Note : we could also use the GPanTest class to test unitarily the GManoeuvre class***



Amount of maneuvers : * ○ 0 ○ 1 ○ 2 ● 3 ○ 4 ○ 5

Thrust number 1 :

Duration : 0.0 mn

Thrust : 0.0 N

Isp : 0.0 s

Thrust number 2 :

Duration : 0.0 mn

Thrust : 0.0 N

Isp : 0.0 s

Thrust number 3 :

Duration : 0.0 mn

Thrust : 0.0 N

Isp : 0.0 s

# Exercice 4 (2/2)

To do it, try to respect the following plan :

1. Create a Maneuver class including duration, thrust and isp attributes and corresponding « *getters* »

2. Create a GManeuver class extending GContainer, implementing GDisplay and corresponding to the Maneuver class
   - Create two constructors : one with no arguments (initial values will be 0.), the second one with a Maneuver object as input.
   - Create a getter method returning a Maneuver object

3. Create a GScenario class extending GPanel including :
   - a GChoice widget (for the amount of maneuvers)
   - a loop on GManeuver widgets.
   - A getter method returning an ArrayList of Maneuver objects.

4. Create a main class including GButton widgets and the GScenario widget.

# GReadWrite interface

- As for GENESIS, GENIUS proposes a way to read and write into files, consistent with the display:
  - By calling GReadWrite interface
  - By definition of the read() and write() methods calling the put() method

… and if we have the same logic as for display, we put all inside the generic() method!

```
public class MyContainer extends GContainer
                 implements GDisplay, GReadWrite {

  GEntryReal   valR;
  GEntryInt    valI;
  GEntryString valS;

  public MyContainer () {
     valR = new GEntryReal("Real value" , 0.);
     valI = new GEntryInt("Integer value", 0);
     valS = new GEntryString("Chain", "");
  }

  public void generic() throws GException {
     put(valR);
     put(valI);
     put(valS);
  }

  public void display() throws GException {
         generic(); }
  public void read() throws GException    {
         generic(); }
  public void write() throws Gexception   {
         generic(); }

}
```

# Data files management

- To read (or write) using GENIUS tools, we only need to open a file and store inside the GENIUS corresponding object the data contained in this file. To do it:
  - We use static methods from class GFileManipulation
  - The file will be in a specific XML (~ MADONA) format

*Remark : the GENIUS object may contain itself other GENIUS objects etc…*

*Must implement the GReadWrite interface*

```
  MyGeniusObject obj = new MyGeniusObject (…);

  GFileManipulation.readConfig  (fileName, XMLRootName, obj, false);
  GFileManipulation.writeConfig (fileName, XMLRootName, obj, true);
```

# File example

- Possibility to differentiate the label displayed on the screen and the XML variable name using method setNameInConfigFile

```
valR.setNameInConfigFile("nomXML");
```

- Possibility to have data structures:

```
public void generic() {
       beginOfElement(structTypeFromEnum, "structureName");
          put( … );
       endOfElement(); }

public void read() { generic(); }
public void write() { generic(); }
```

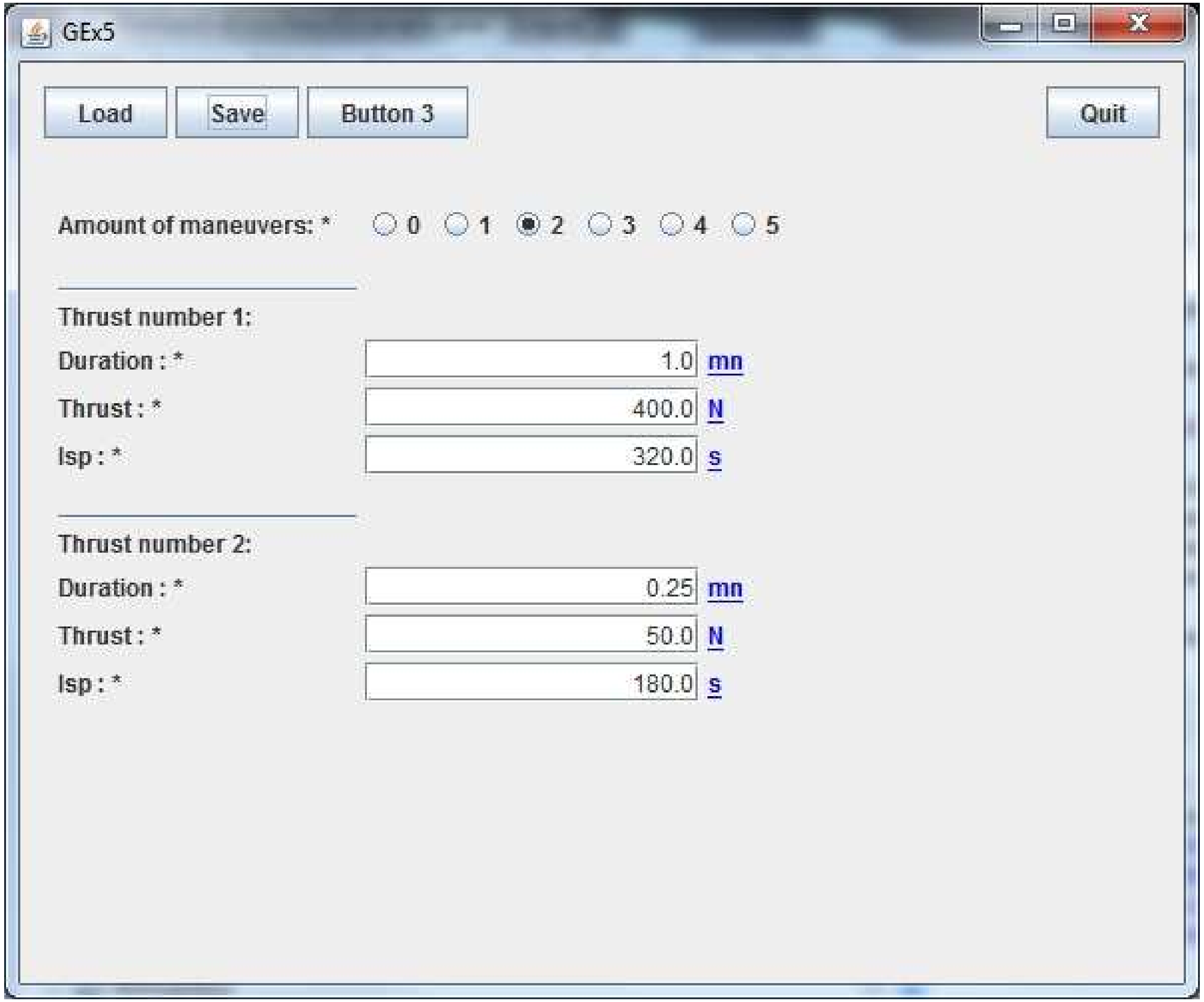
```
<Potential name="earthPotential">
     <Real name="mu" unit="km^3/s^2">398600.64</Real>
     <Real name="g0" unit="m/s^2">9.805</Real>
     <Real name="rt" unit="km">6378.139</Real>
     <Real name="ze" unit="km">120.0</Real>
     <Real name="wt" unit="deg/s">0.004178071267451</Real>
</Potential>
```

# Back on modified data

- **We saw that a "*" character appears when a data is modified by user by comparison to a “saved” value. More precisely :**
  - **The “saved” value corresponds either to a default value (*for data loaded (resp. saved) when reading (resp. writing) a file, it can be customized*)**
  - **When a unit is changed, as the data is not actually changed because the value stored in memory is not changed => no "*" character appears**
  - **If the user enter a “new” value which, in fact, corresponds to the saved value, the "*" character disappears :**
    - Initial value = 0
    - new value = 1 => "*" character appears
    - “new” value = 0 => "*" character disappears
  - **It is possible to manage locally this mechanism using the following methods:**
    - setDisplayIsModifiedIndicator(DisplayIndicatorStatus), the status being « Automatic », « Always » or « Never »
    - setSavedValue(xxx) => if the saved value is the same as the displayed one, "*" character disappears

# Exercise 5

- Add to the previous exercise the possibility to load and store data into files:



Amount of maneuvers: * ○ 0 ○ 1 ● 2 ○ 3 ○ 4 ○ 5

Thrust number 1:

|  |  |  |
|---|---|---|
| Duration : * | 1.0 | mn |
| Thrust : * | 400.0 | N |
| Isp : * | 320.0 | s |

Thrust number 2:

|  |  |  |
|---|---|---|
| Duration : * | 0.25 | mn |
| Thrust : * | 50.0 | N |
| Isp : * | 180.0 | s |

# Execute computation (1/2)

- It is good to have a *GUI* ... but it has to be useful ! And most of the time, it is used to launch a **computation program**.

- Several solutions are available:
  - Launch a Java **thread** … but it could not be stopped asynchronously (*stop* method is deprecated) except by stopping the GUI !!!
  - Launch an **executable independent** of the GUI

- **GENIUS** makes available classes **G[Java]CommandLauncher**, **GExecButton** and **GExecMenuItem**. They will launch:
  - Either a **Java class**, if it owns a « main » static method
  - Either an **executable** (for example issued from a Fortran compilation)

- A consequence is that entry data will only be passed by files.

# Execute computation (2/2)

```
String path = System.getProperty("java.class.path");
cmd = new GJavaCommandLauncher ( new String[] {"myClass", "args …"}, path,
                                 "Start computation", "Stop computation", null);
cmd.setCopyOutputToStdout(true);
GExecButton butExec = cmd.getGExecButton();
```

Class (or jar) owning a « main » static method

Specific button

```
public void generic() throws GException {
   ...
   put(cmd);
}
```

!!! Do not forget !!!

```
public void before(GEvent e) throws GFileManipulatorException {
    if ( e.contains(butExec) ) {
        if ( ! cmd.isRunning() ) { // Program initialization if it is not yet running
            if ( valeursOK ) { // Test if GUI values are OK
                GFileManipulation.writeConfig("data.xml", "MAN", objetIhm);
            }
            else {cmd.setInhibited(true); }
        }
    }
}
```

We write the context file …

… or, finally, we do not launch it !

```
public void after(GEvent e) {
    if ( e.getFinalSource() == cmd ) {
        // We launched the application
        if ( cmd.getProcessStatus() == ProcessStatus.FINISHED_NORMALY ) {
            System.out.println("Computation nominally finished ...");
        }
        else if ( cmd.getProcessStatus() == ProcessStatus.FINISHED_BY_USER ) {
            System.out.println("Computation stopped by user ...");
        }
}
```

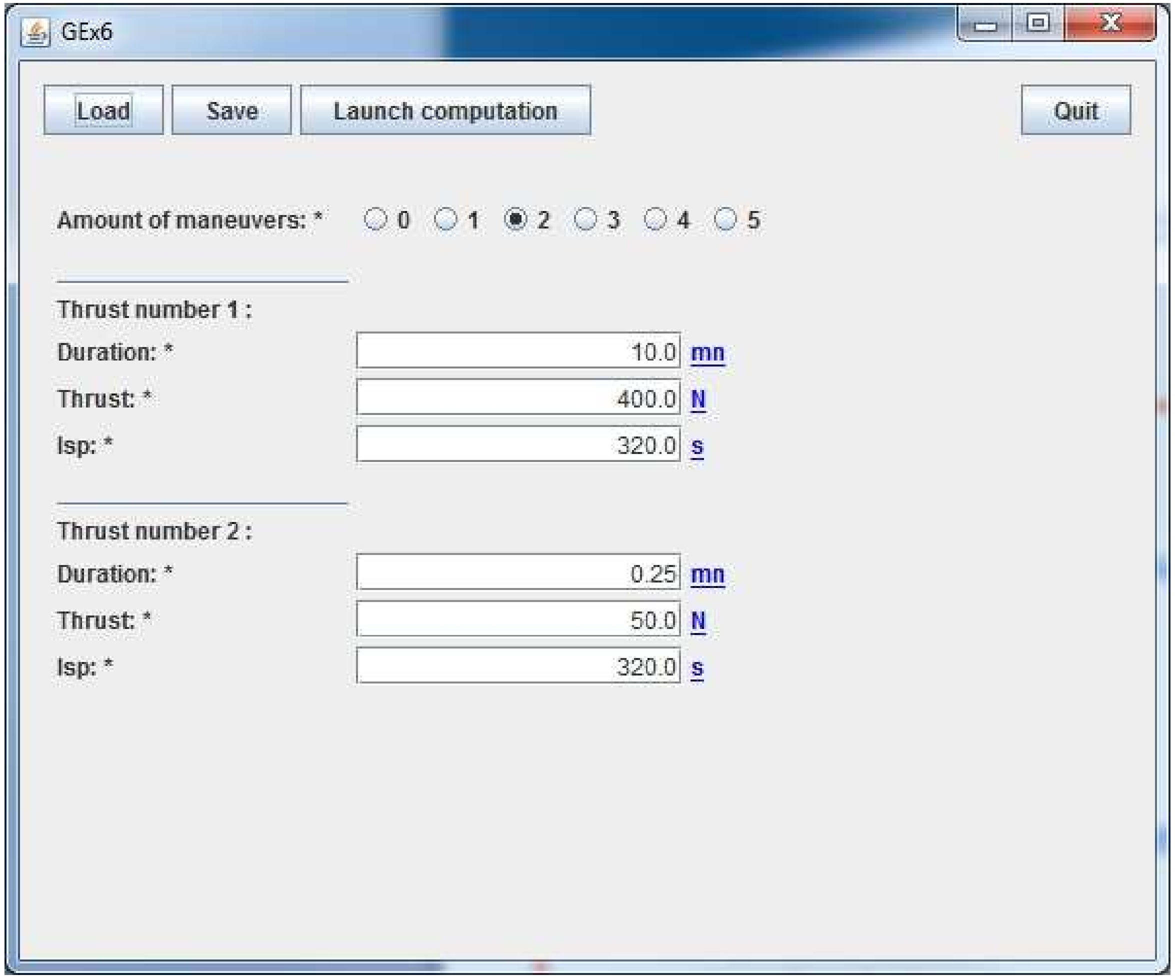
We can catch output status

# Exercise 6

- **Add to the following exercise the possibility to launch a computation:**
  - **Create**
    - a class to execute some computation (for example sum of the thrusts duration) : “*subroutine mode*”

    ⇨ separation between computation and GUI

    - another one reading the *XML* file then extracting maneuvers data thanks to the previous created getters and calling the computation class : “*batch mode*”



Amount of maneuvers: * ○ 0 ○ 1 ● 2 ○ 3 ○ 4 ○ 5

Thrust number 1 :

| | | |
|---|---|---|
| Duration: * | 10.0 | mn |
| Thrust: * | 400.0 | N |
| Isp: * | 320.0 | s |

Thrust number 2 :

| | | |
|---|---|---|
| Duration: * | 0.25 | mn |
| Thrust: * | 50.0 | N |
| Isp: * | 320.0 | s |

CENTRE NATIONAL D'ÉTUDES SPATIALES

# GENIUS

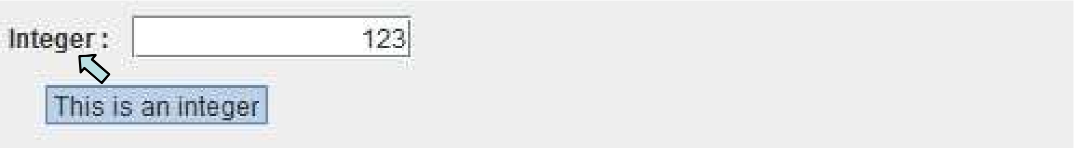
## GENeration of Interface for Users of Scientific S/W

## Still more …

# Tooltips

- GENIUS proposes very simply the possibility to add tooltips using the setToolTipText method

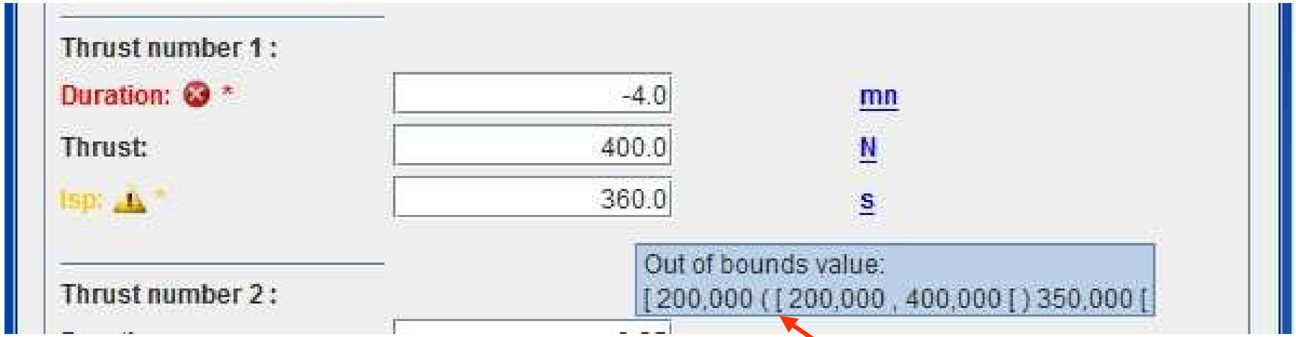
```
GEntryInt  valI = new GEntryInt("Integer :", 123);

valI.setToolTipText("This is an integer");
```



Integer : 123

# Validity controls (1/2)

- **GENIUS gives the possibility to manage validity intervals (consistent with SIRIUS requirements):**
  - **Only for GEntryReal, GEntryInt, GEntryRealVector et GEntryIntVector widgets**
  - **Possibility to get an error and/or warning information**
  - **For real values, these validity controls of course take into account units management**



Thrust number 1 :

| | | |
|---|---|---|
| Duration: ⊗ * | -4.0 | mn |
| Thrust: | 400.0 | N |
| Isp: ⚠ * | 360.0 | s |

Out of bounds value:
[ 200,000 ( [ 200,000 , 400,000 [ ) 350,000 [

Thrust number 2 :

*Tool tip when mouse passes over the input area*

# Validity controls (2/2)

```
GUnit[] unitDuration = {new GMetricUnit("mn"), new GMetricUnit("s")};
GUnit[] unitThrust = {new GMetricUnit("N")};
GUnit[] unitIsp = {new GMetricUnit("s")};

// Error control validity
durationIhm =  new GEntryReal("Duration:", val1, unitDuration);
durationIhm.addGInterval( new GInterval(0., Double.POSITIVE_INFINITY) );

// No validity control
thrustIhm =  new GEntryReal("Thrust:", val2, unitThrust);

// Error and warning control validity
// Error if ]-Inf,200[ or[400,+Inf[
// Warning if [200,250[ or[350,400[
// OK if[250,350[
ispIhm     =  new GEntryReal("Isp:", val3, unitIsp);
ispIhm.addGInterval(
   new GInterval(250., 350., GInterval.Rule.INCLUSIVE, GInterval.Rule.EXCLUSIVE,
                 200., 400., GInterval.Rule.INCLUSIVE, GInterval.Rule.EXCLUSIVE) );
```

*Error if the value is out of this interval*

*Opened/Closed interval management*

# GCondensedStatusInterface

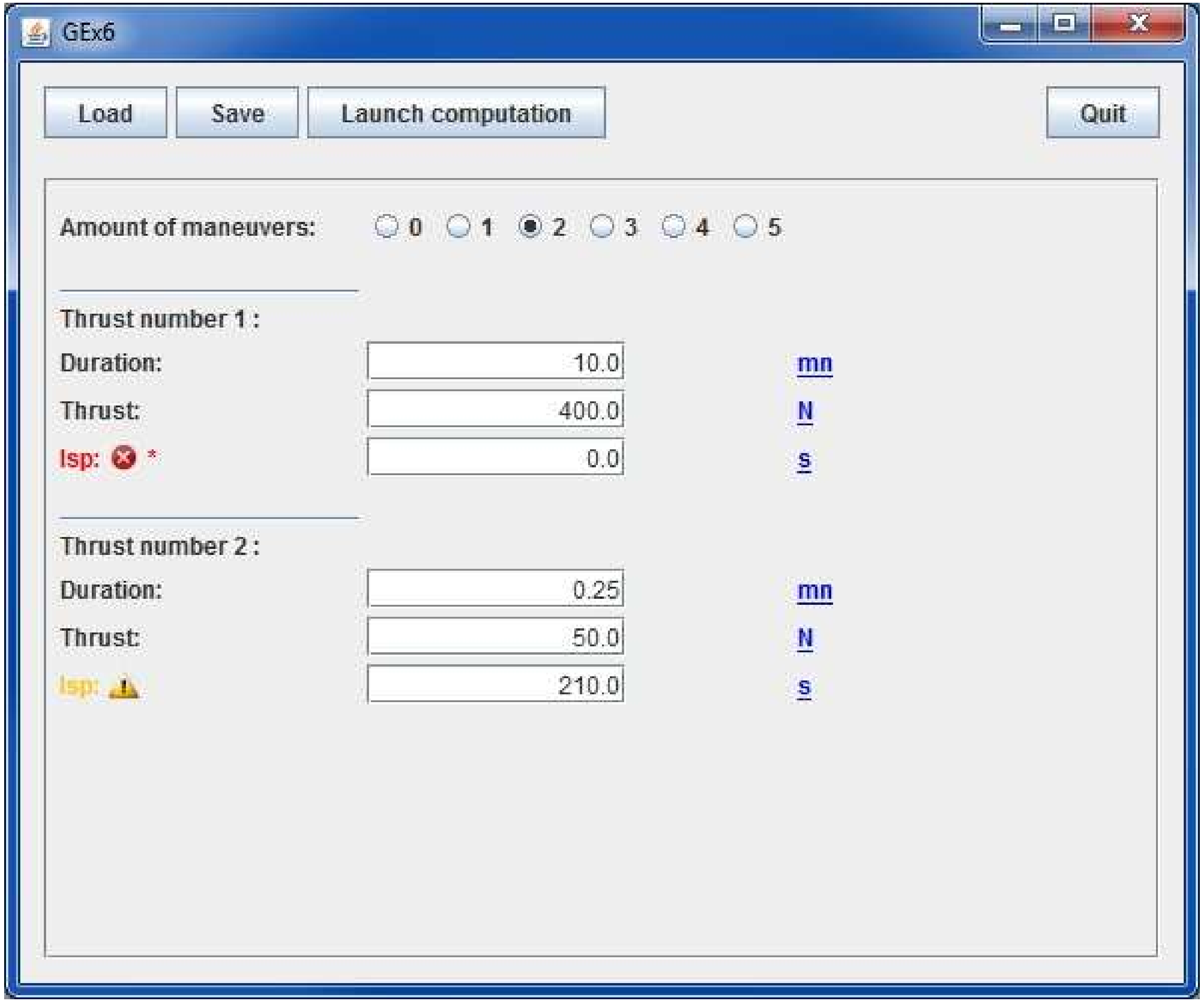
- GENIUS gives also the possibility to manage a “global” status of a set of data via the GCondensedStatusInterface:

```
public class Data extends GPanel implements GCondensedStatusInterface {
  …
  @Override
  public void updateCondensedStatus(GCondensedStatus arg) {
    // durationIhm, thrustIhm and ispIhm are checked
    arg.update(durationIhm , thrustIhm , ispIhm);
  }
}
```

```
GCondensedStatus status = new GCondensedStatus(new Data(…));

// We print the global status ...
System.out.println("Global status: ‟ + status.getStatus());

// We print the list of data with an ERROR status ...
for (int i = 0; i < status.getErrorComponentList().size(); i++) {
  System.out.println(
    "Error on "+status.getErrorComponentList().get(i).getNameInConfigFile() );
}
```

# Exercise 6 (ct'd)

- Redo the exercise 6 adding validity intervals to maneuvers characteristics



Amount of maneuvers: ○ 0 ○ 1 ● 2 ○ 3 ○ 4 ○ 5

Thrust number 1 :

|  |  |  |
|---|---|---|
| Duration: | 10.0 | mn |
| Thrust: | 400.0 | N |
| Isp: ✖ * | 0.0 | s |

Thrust number 2 :

|  |  |  |
|---|---|---|
| Duration: | 0.25 | mn |
| Thrust: | 50.0 | N |
| Isp: ⚠ | 210.0 | s |

# Menu Bar

- As for a “classical” GUI, GENIUS proposes to have a main bar menu with GMenuBar class (on the same principle as Swing JMenuBar)

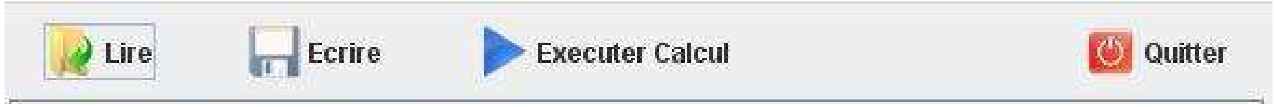
```
public mainPanel() {

  // We create menu items
  itemLoad = new GMenuItem("Load");
  itemSave = new GMenuItem("Save");
  itemQuit = new GMenuItem("Quit");

  // We create the “File” menu
  // containing the previous items
  menuFile = new GMenu("File");
  menuFile.add(itemLoad);
  menuFile.add(itemSave);
  menuFile.add(itemQuit);

  // We add “File” menu to the menu bar
  bar = new GMenuBar(this);
  bar.add(menuFile);

  ...
```

```
public void after(GEvent e) throws Exception {

  if (e.contains(itemLoad) ){
         GFileManipulation.readConfig(...);
  }
  if (e.contains(itemSave) ){
         GFileManipulation.writeConfig(...);
  }
  if (e.contains(itemQuit) ){
         System.exit(0);
  }

}
```

```
  mainPanel pan = new mainPanel();

  // We call the GFrame constructor with a supplementary
  // argument with a GMenuBar onject
  GFrame frame = new GFrame("GEx7", pan, pan.getMenuBar());
```

# Icons

- GENIUS also allows to get icons instead of buttons with label:
  - Always use the GButton class
  - Also applies to GExecButton
  - Standard icons are proposed via GIcon class



*Search for files included into genius.jar*

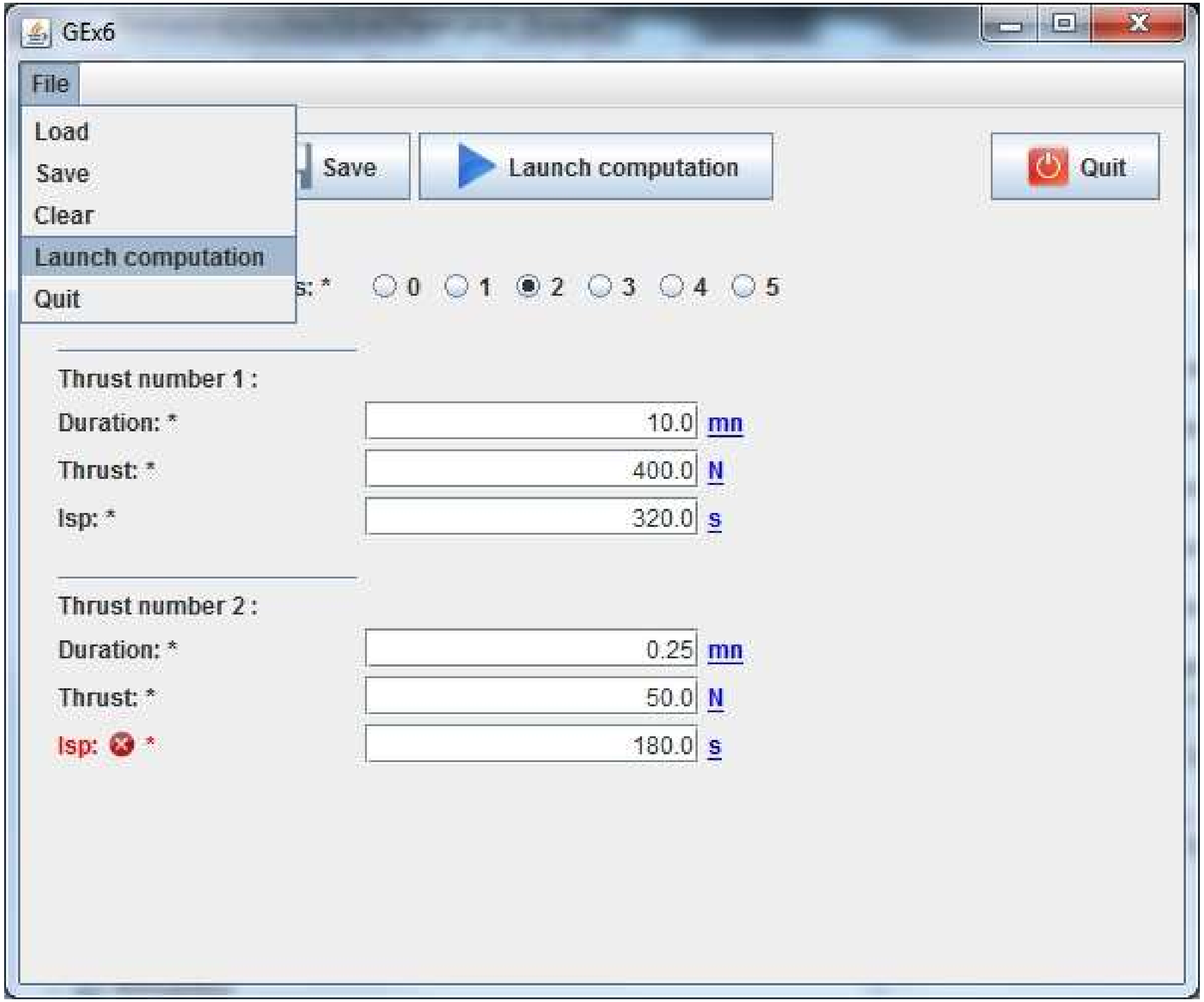
```
butLoad  = new GButton("Load", new GIcon (GIcon.Type.OPEN, 24));
butWrite = new GButton("Write", new GIcon (GIcon. Type.SAVE, 24));

 cmd = new GJavaCommandLauncher( … );
 cmd.setButtonIcons(new GIcon(GIcon.Type.START, 12),
                    new GIcon(GIcon.Type.STOP, 12));

butAppli = new GButton("Appli", String absoluteOrRelativePath);
```

*Icons change automatically when launch/stop*

*Specific icon*

# Exercise 6 (ct'd)

- Continue the exercise 6 by including a menu bar and by using GENIUS by default icons



s: * ○ 0 ○ 1 ● 2 ○ 3 ○ 4 ○ 5

Thrust number 1 :

| | | |
|---|---|---|
| Duration: * | 10.0 | mn |
| Thrust: * | 400.0 | N |
| Isp: * | 320.0 | s |

Thrust number 2 :

| | | |
|---|---|---|
| Duration: * | 0.25 | mn |
| Thrust: * | 50.0 | N |
| Isp: ☒ * | 180.0 | s |

# GClear interface

- As for the GReadWrite interface, GENIUS proposes a GClear interface in order to reinitialize data:
  - The data are then reinitialized to the default value given when the widget has been instantiated
  - There is the possibility to change this default value by using the setDefaultValue method

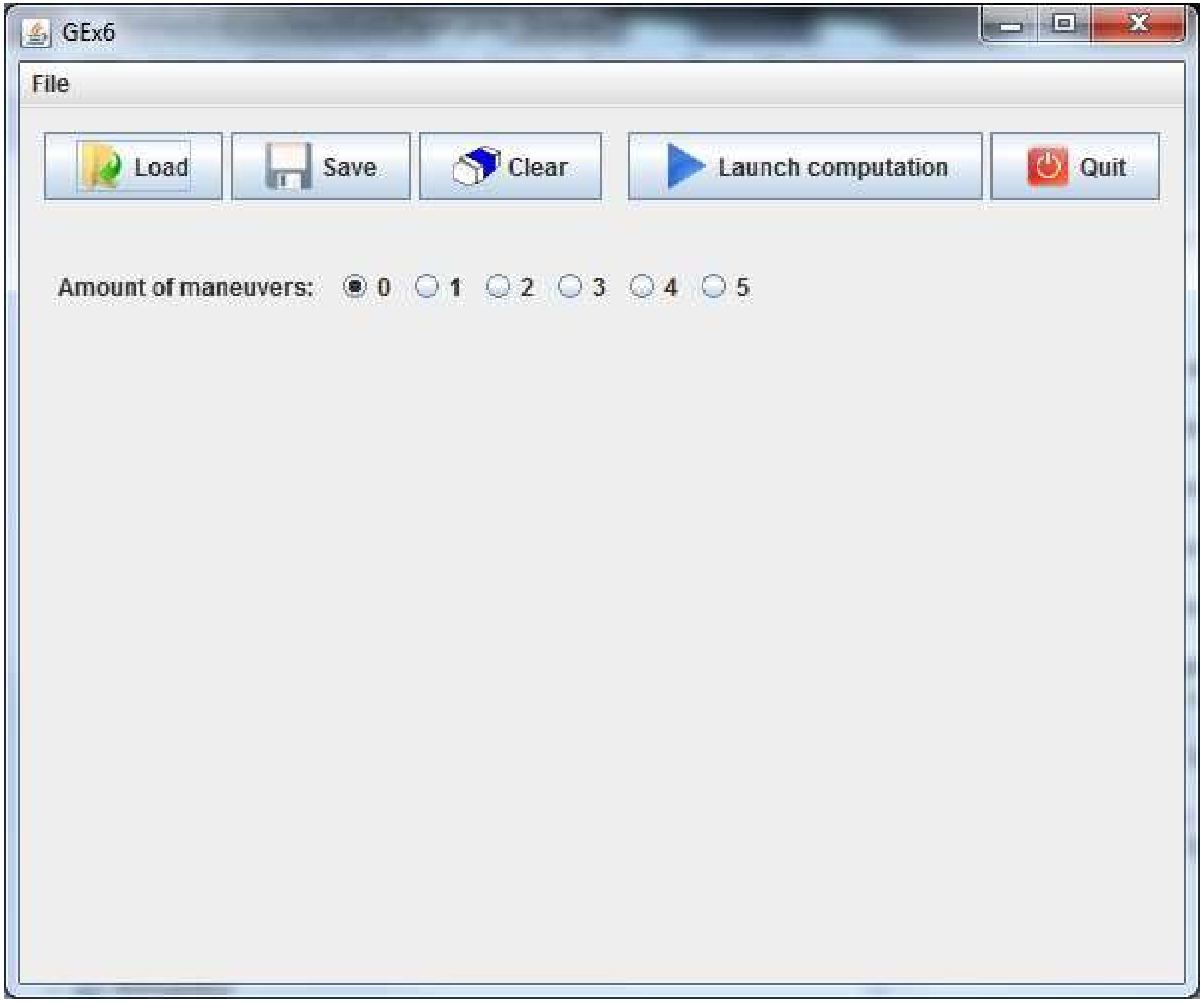
```
public class GManoeuvre extends GContainer implements GDisplay, GReadWrite, GClear {
…
   public void clear() throws GException { generic(); }
…
```

- At last, you will just have to call the mainClear method of the high level object you want to clear (it will correctly call the put methods of the lower level objects, as for display) …

```
   if ( e.contains(butClear) ){ obj.mainClear(); }
```

# Exercise 6 (ct'd)

- Add to the exercise 6 the possibility to clear data …



Amount of maneuvers: ● 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 cnes

CENTRE NATIONAL D'ÉTUDES SPATIALES

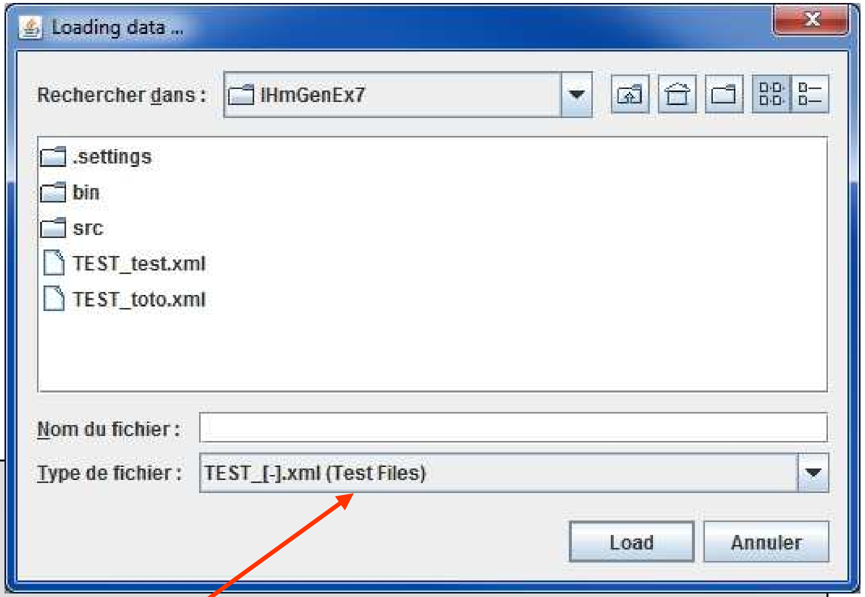
# GENIUS

GENeration of Interface for Users of Scientific S/W

## Some other « high level » widgets …

# GContextFileManagement class

- **GENIUS proposes a class to simplify the search of files into directories:**
  - **GContextFileManagement class**



Loading data ...

Rechercher dans : IHmGenEx7

.settings
bin
src
TEST_test.xml
TEST_toto.xml

Nom du fichier :

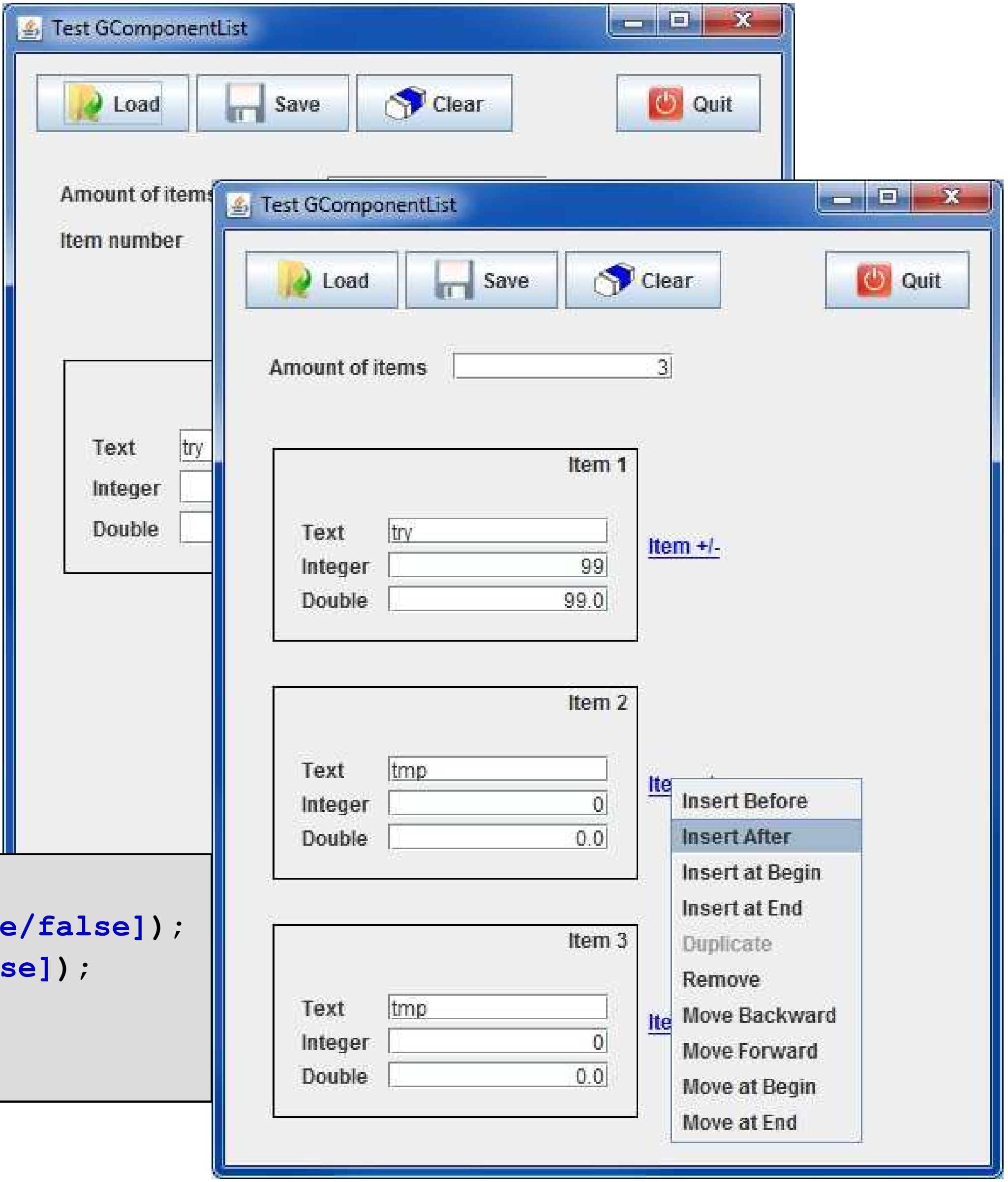
Type de fichier : TEST_[-].xml (Test Files)

```
String prefix  = "TEST_";
String suffix  = ".xml";
String comment = "Test Files";
GFileFilter filter = new GFileFilter(prefix ,suffix ,comment);

String initDir = ".";
String xmlName = "Test";
GContextFileManagement gfm = new GContextFileManagement(initDir, xmlName , filter );

…

public void after(GEvent e) throws GFileManipulatorException {

  if ( e.contains(butLoad) ) { gfm.selectLoadFile(obj, false); }
  if ( e.contains(butSave) ) { gfm.selectSaveFile(obj, true); }
```

*Widget to load or save*

# The GComponentList class

- **Allows to display list of widgets :**
  - These widgets must have a **constructor without arguments**
  - Possibility to duplicate an element only if the **Cloneable** interface (and a clone method) is implemented.
  - « **single** » mode displaying only one widget each time (case of complex widgets)
  - « **multiple** » mode displaying all the widgets placed behind each other



Test GComponentList

Amount of items

Item number

Text: try

Integer

Double

Test GComponentList

Amount of items: 3

Item 1

Text: try

Integer: 99

Double: 99.0

Item 2

Text: tmp

Integer: 0

Double: 0.0

Item 3

Text: tmp

Integer: 0

Double: 0.0

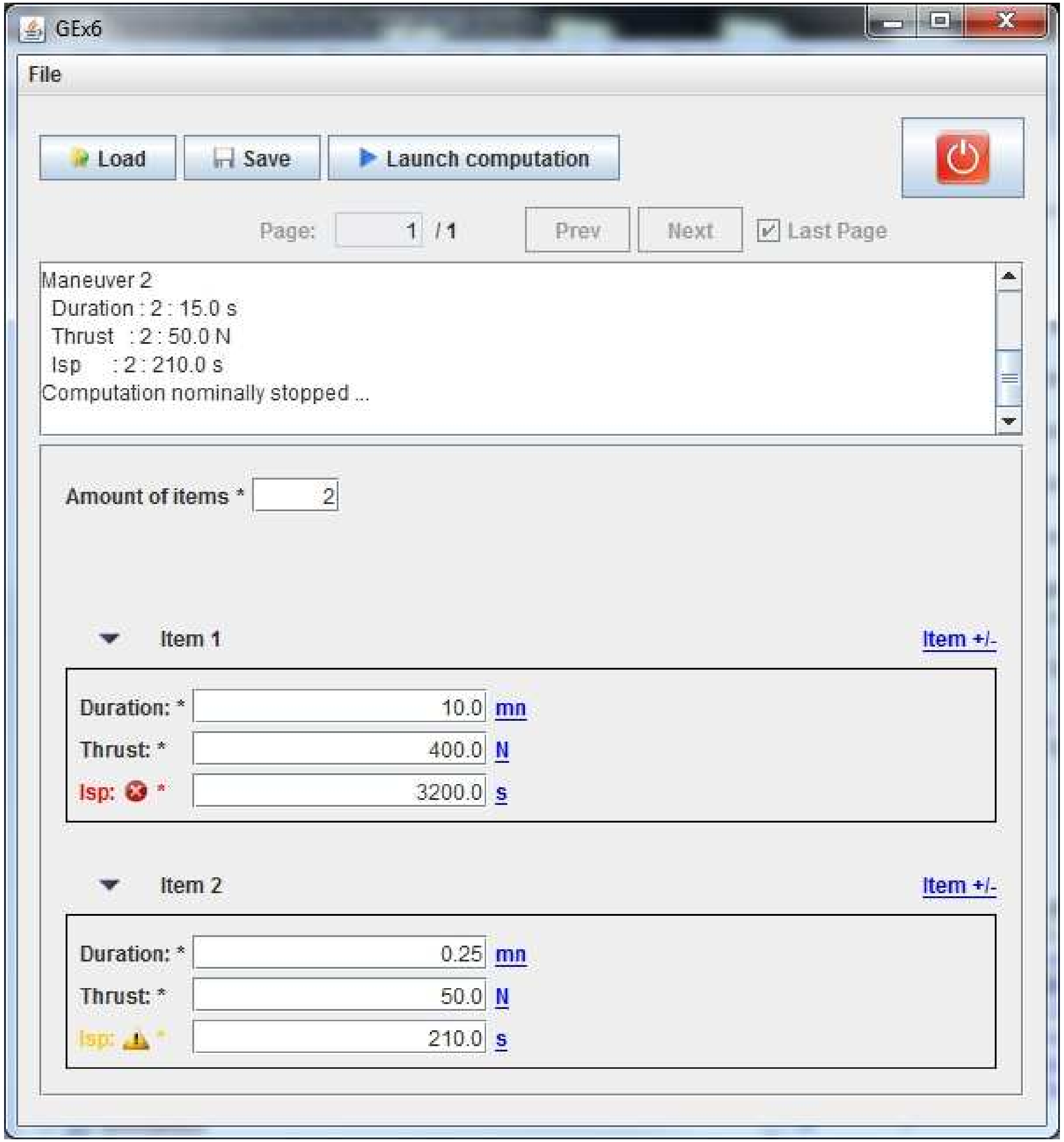
```
 GComponentList test;
test = new GComponentList("name", className.class, mode[true/false]);
test = new GComponentList("name", defaultObj, mode[true/false]);
 …
test.setList (initList);
```

# Exercice 6 (ct'd)

- Modify the exercise 6 to use GComponentList class

… and if possible :

- ➢ GContextFileManagement
- ➢ GBufferedTextArea



```
Maneuver 2
 Duration : 2 : 15.0 s
 Thrust   : 2 : 50.0 N
 Isp      : 2 : 210.0 s
Computation nominally stopped ...
```

Amount of items * 2

Item 1 — Item +/-

| | | |
|---|---|---|
| Duration: * | 10.0 | mn |
| Thrust: * | 400.0 | N |
| Isp: * | 3200.0 | s |

Item 2 — Item +/-

| | | |
|---|---|---|
| Duration: * | 0.25 | mn |
| Thrust: * | 50.0 | N |
| Isp: * | 210.0 | s |

# Many other interesting items …

## ■ Some other widgets:

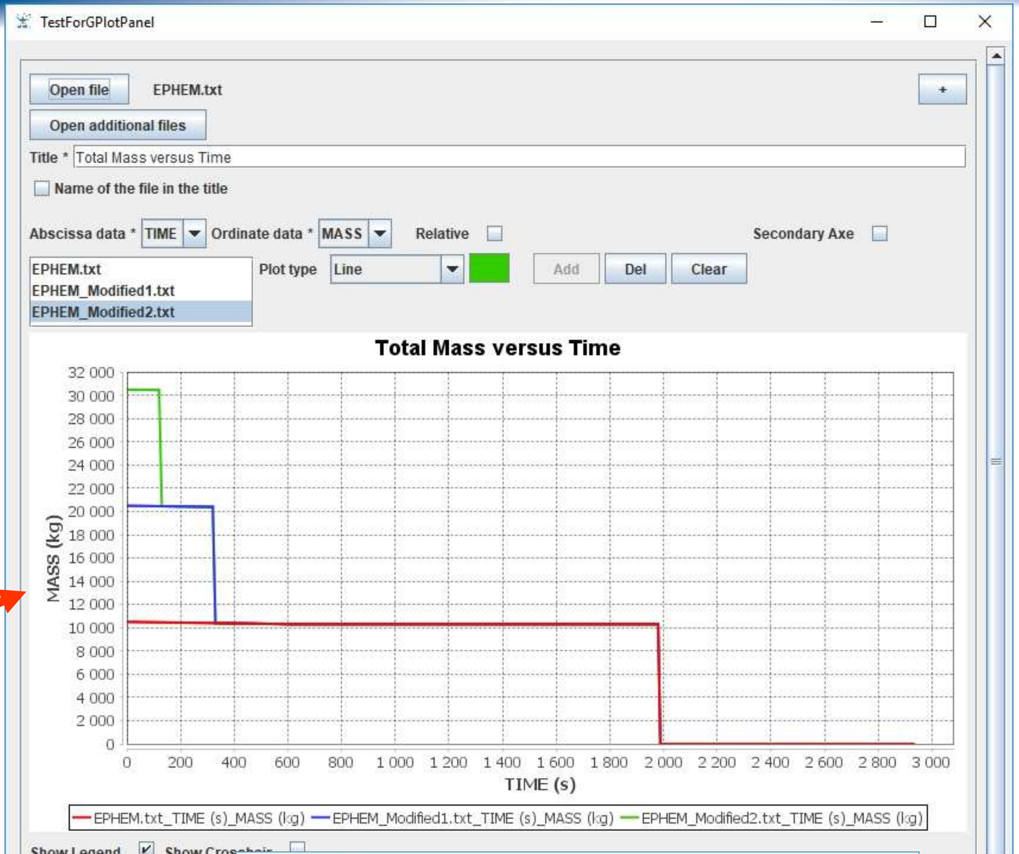
- GTabbedPane
- GTable (1D, 2D)
- GEntryConstant
- GDialog and GDetachedPanel
- GBufferedTextArea

## ■ For plotting:

- GFreeChartXY
- GPlotPanel (gplot-1.9.1.jar)
- GGroundPlotPanel

## ■ Some other functionality:

- Copy & Paste
- How to manage modified data as global status
- Shortcuts
- Internationalization
- How to update same data on different panels
- How to build a Standard Application GUI
- How to create your own widget



TestForGPlotPanel

Open file EPHEM.txt

Open additional files

Title * Total Mass versus Time

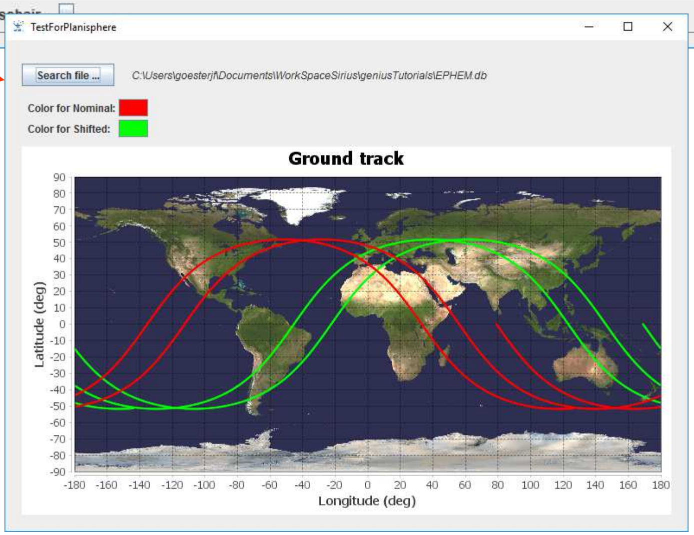
□ Name of the file in the title

Abscissa data * TIME ▾ Ordinate data * MASS ▾ Relative □ Secondary Axe □

EPHEM.txt<br>EPHEM_Modified1.txt<br>EPHEM_Modified2.txt

Plot type Line ▾ 🟩 Add Del Clear

Show Legend ☑ Show Crosshair □



TestForPlanisphere

Search file ... C:\Users\goesterjf\Documents\WorkSpaceSirius\geniusTutorials\EPHEM.db

Color for Nominal: 🟥

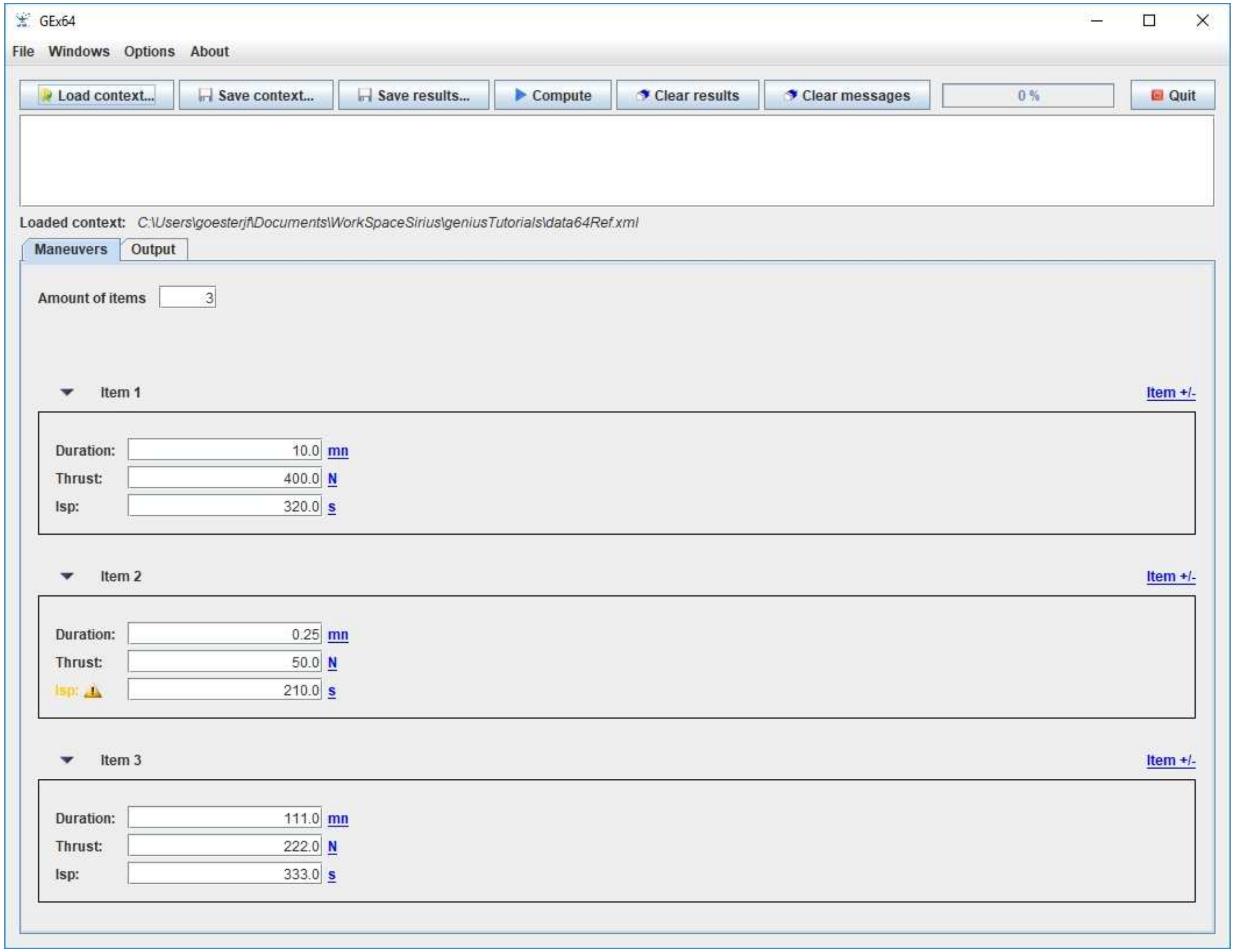
Color for Shifted: 🟩

# Exercice 6 (ct'd)

- **Based on the previous exercise, now use the Standard application frame !**

  **See [http://genius.cnes.fr/index.php/How_to_build_a_standard_application](http://genius.cnes.fr/index.php/How_to_build_a_standard_application)**

*Note : we need an image file for About menu : download from Web, set it to a consistent size and put it on src/main/resources*



Loaded context: C:\Users\goesterjf\Documents\WorkSpaceSirius\geniusTutorials\data64Ref.xml

Amount of items 3

Item 1

| | | |
|---|---|---|
| Duration: | 10.0 | mn |
| Thrust: | 400.0 | N |
| Isp: | 320.0 | s |

Item 2

| | | |
|---|---|---|
| Duration: | 0.25 | mn |
| Thrust: | 50.0 | N |
| Isp: ⚠ | 210.0 | s |

Item 3

| | | |
|---|---|---|
| Duration: | 111.0 | mn |
| Thrust: | 222.0 | N |
| Isp: | 333.0 | s |

CENTRE NATIONAL D'ÉTUDES SPATIALES

# GENIUS

GENeration of Interface for Users of Scientific S/W

## Conclusion

# Conclusion

☺ **Main GENESIS functionalities still exist inside GENIUS product:**

- **Numbers input, conditional display, before/after, read/write into files, units management, plots …**

☺ **Some (big) GENESIS drawbacks have disappeared:**

- **Specific syntax, object approach mixed with Fortran, generation delay …**

☺ **A lot of new widgets or functionalities are available:**

- **"*" character when a data is modified, validity controls, tables of data, list of widgets, …**

😐 **Less concise than GENESIS (due to Java …) but possibility to debug easily !**