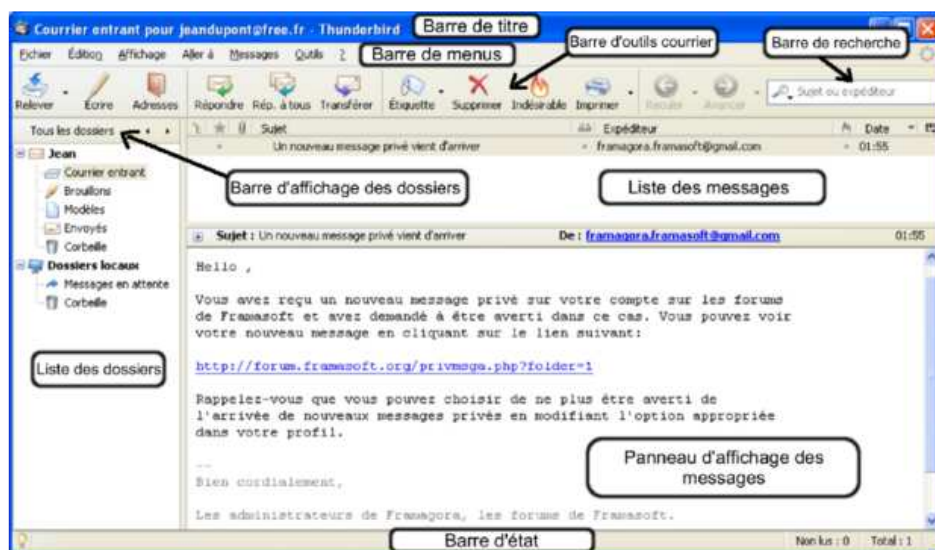


Swing

1

1

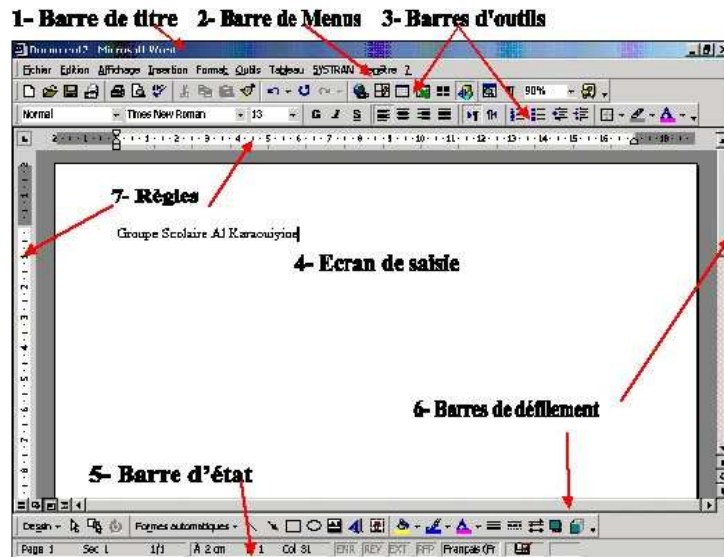
Rappel du vocabulaire des zones d'une application avec IHM (UI) sous Windows
La barre d'état tout en bas



2



Rappel du vocabulaire des zones d'une application avec IHM (UI) sous Windows



3



Documentation pour Swing

Où trouver de la documentation sur
Swing en anglais? :
La javadoc et le tutoriel officiel
?en français?

XH

4

4



Module java.desktop

- contenant les packages pour Swing

Module java.desktop

Defines the AWT and Swing user interface toolkits, plus APIs for accessibility, audio, imaging, printing, and JavaBeans.

Module Graph:

```
graph TD
    java.desktop --> java.xml
    java.desktop --> java.datatransfer
    java.xml --> java.base
    java.datatransfer --> java.base
```

Packages

Exports

Package

XH

5



De nombreux packages

Overview | **Module** | **Package** | **Class** | **Use** | **Tree** | **Deprecated** | **Index** | **Help**


ALL CLASSES | **MODULE: DESCRIPTION** | **MODULES** | **PACKAGES** | **SERVICES**

Search:

Package	Description
javax.sound.midi.spi	Supplies interfaces for service providers to implement when offering new MIDI devices, MIDI file readers and writers, or sound bank readers.
javax.sound.sampled	Provides interfaces and classes for capture, processing, and playback of sampled audio data.
javax.sound.sampled.spi	Supplies abstract classes for service providers to subclass when offering new audio devices, sound file readers and writers, or audio format converters.
javax.swing	Provides a set of "lightweight" (all-Java language) components that, to the maximum degree possible, work the same on all platforms.
javax.swing.border	Provides classes and interface for drawing specialized borders around a Swing component.
javax.swing.colorchooser	Contains classes and interfaces used by the JColorChooser component.
javax.swing.event	Provides for events fired by Swing components.
javax.swing.filechooser	Contains classes and interfaces used by the JFileChooser component.
javax.swing.plaf	Provides one interface and many abstract classes that Swing uses to provide its pluggable look-and-feel capabilities.
javax.swing.plaf.basic	Provides user interface objects built according to the Basic look and feel.

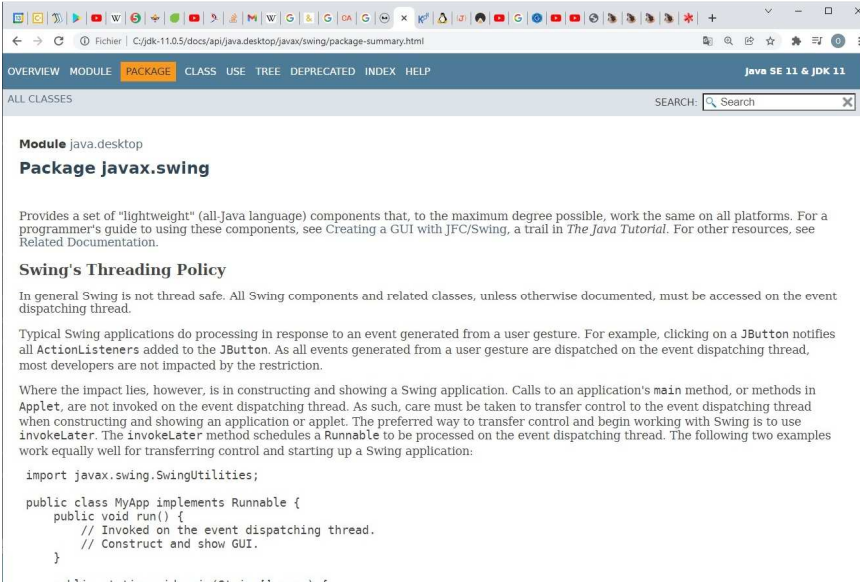
XH

6



Package javax.swing

- Idée de code pour une application graphique



Module java.desktop

Package javax.swing

Provides a set of "lightweight" (all-Java language) components that, to the maximum degree possible, work the same on all platforms. For a programmer's guide to using these components, see *Creating a GUI with JFC/Swing*, a trail in *The Java Tutorial*. For other resources, see *Related Documentation*.

Swing's Threading Policy

In general Swing is not thread safe. All Swing components and related classes, unless otherwise documented, must be accessed on the event dispatching thread.

Typical Swing applications do processing in response to an event generated from a user gesture. For example, clicking on a JButton notifies all ActionListener added to the JButton. As all events generated from a user gesture are dispatched on the event dispatching thread, most developers are not impacted by the restriction.

Where the impact lies, however, is in constructing and showing a Swing application. Calls to an application's main method, or methods in Applet, are not invoked on the event dispatching thread. As such, care must be taken to transfer control to the event dispatching thread when constructing and showing an application or applet. The preferred way to transfer control and begin working with Swing is to use invokeLater. The invokeLater method schedules a Runnable to be processed on the event dispatching thread. The following two examples work equally well for transferring control and starting up a Swing application:

```
import javax.swing.SwingUtilities;

public class MyApp implements Runnable {
    public void run() {
        // Invoked on the event dispatching thread.
        // Construct and show GUI.
    }
}

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

7



Tutoriel officiel (in english)

- <http://ads.gigatux.nl/tutorial7/ui/TOC.html>
- Nouvel emplacement



The Java™ Tutorials

« Previous Home Page

Trail: Graphical User Interfaces: Table of Contents

[A Brief Introduction to the Swing Package](#)

[What is Swing?](#)

[A Swing Demo](#)

[Swing Features](#)

[A Visual Guide to Swing Components \(Java Look and Feel\)](#)

[A Visual Guide to Swing Components \(Windows Look and Feel\)](#)

[Pluggable Look and Feel](#)

[Data Transfer](#)

[Internationalization and Localization](#)

[Accessibility](#)

[Integrating with the Desktop](#)

[System Tray Icon Support](#)

[Questions and Exercises: Graphical User Interfaces](#)

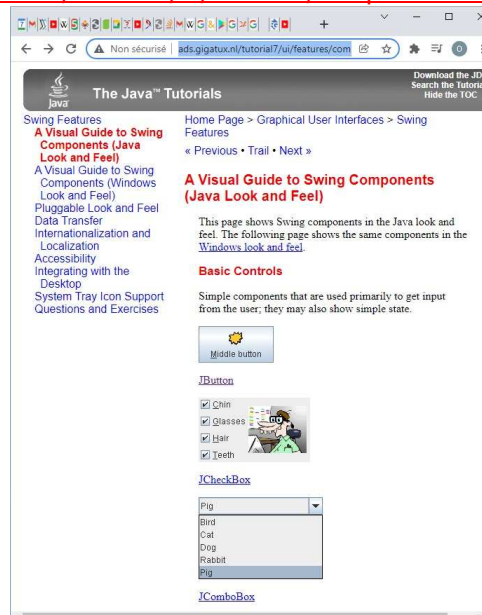
« Previous

8



Liste illustrée des composants

<http://ads.gigatux.nl/tutorial7/ui/features/components.html>



XH

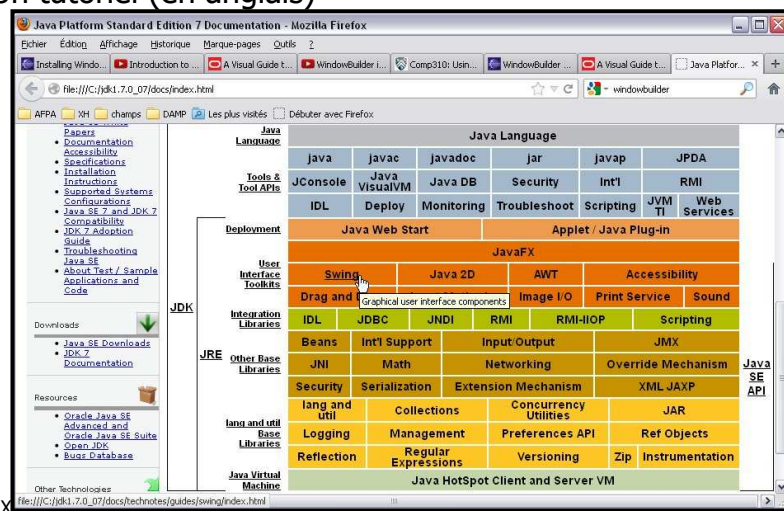
9

9



Ancien emplacement du tutoriel officiel de Swing


- On passait par la documentation officielle du JDK 8
- Un tutoriel (en anglais)



10



11



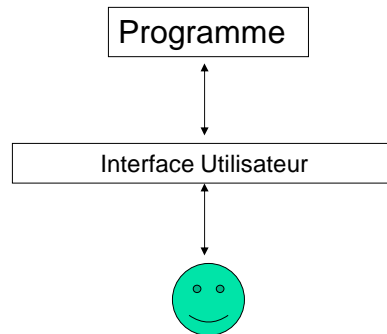
Conception d'une fenêtre graphique

- Construire une **interface** graphique
 - Objets graphiques
 - Composition, affichage
- Programmation évènementielle
 - par événement
 - Principe MVC – Modèle-Vue-Contrôle
- API Swing

12



Généralité



Rôles d'une interface utilisateur:

- montrer le résultat de l'exécution
- permettre à l'utilisateur d'interagir

13



Exemple simple

```
import javax.swing.*;
public class Application {
    public static void main (String[] args) {
        JFrame fenetre = new JFrame("Application de dessin");

        fenetre.setSize(300,200);
        fenetre.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fenetre.setVisible(true); System.exit(0);
    }
}
```

Annotations for the code example:

- Importer le package**: points to the `import javax.swing.*;` line.
- Créer un objet**: points to the `new JFrame("Application de dessin");` line.
- Définir la taille**: points to the `fenetre.setSize(300,200);` line.
- afficher**: points to the `fenetre.setVisible(true);` line.



14



Afficher votre application graphique

- Importer le package (les classes)
 - Les classes sont regroupées en package
 - Importer un package = importer toutes les classes du package
 - `import javax.swing.*;`
- Créer une fenêtre graphique (JFrame, ...)
- Mettre les paramètres (taille, ...)
- Afficher
- Ajouts par la suite:
 - `import java.awt.*;` les classes dans awt
 - `import java.awt.event.*;` les classes dans event

15



Les objets graphiques

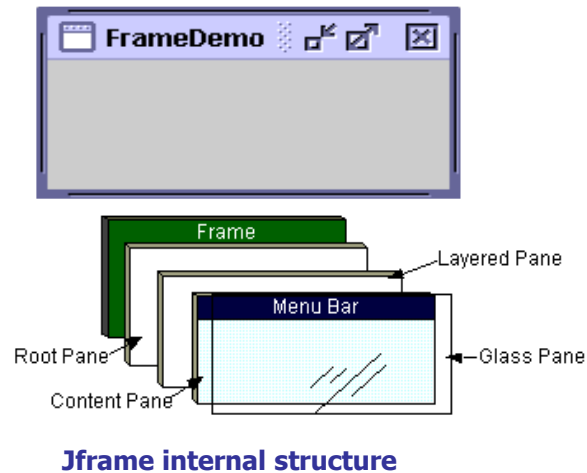
- 3 niveaux
 - Haut niveau
 - Définir une fenêtre
 - *JApplet, JDialog, JFrame, JWindow*
 - Niveau intermédiaire
 - Pour composer la fenêtre
 - *JPanel, JScrollPane, JSplitPane, ...*
 - Niveau inférieur
 - Les éléments de base
 - *JButton, JCheckBox, JPasswordField, JTextArea, ...*

16



Plaqer des éléments dans la fenêtre

- Composition d'une fenêtre JFrame
- La contentPane est déjà présente



17



Ajouter des composants dans une fenêtre

```
import javax.swing.*;

public class Application {
    public static void main (String[] args) {
        JFrame f = new JFrame("FrameDemo");
        JLabel label = new JLabel("Hello World");
        JPanel p = (JPanel)f.getContentPane();
        p.add(label);
        f.setSize(300,200); //alternative: f.pack();
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setVisible(true);
    }
}
```



18



Composer une fenêtre « à la main »

- Créer un ou des composants intermédiaires (2) (Panneau)
 - Pour *JFrame*, un *JPanel* est associé implicitement (ContentPane)
- Créer des composants de base (3)
- Insérer (3) dans (2)
- Créer une fenêtre (1)
- Insérer (2) dans (1)
- Afficher la fenêtre (1)

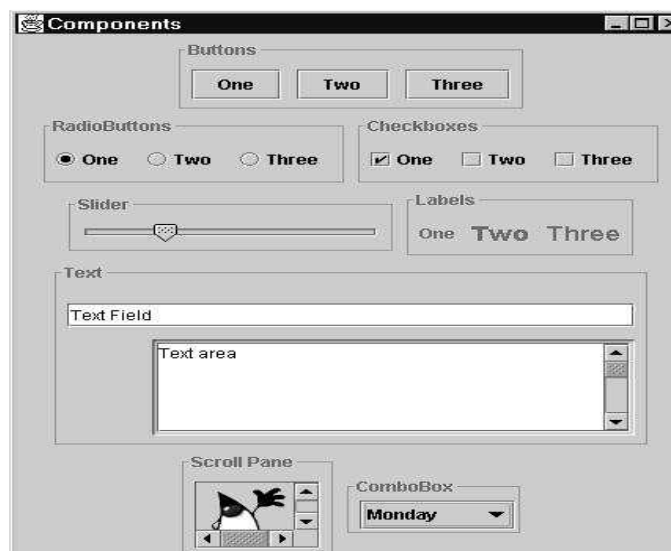


19



Swing components: Illustration

Voir la démo SwingSet2.jar



20



Composants de base pour afficher l'information

- *JLabel* contains text string, an image, or both.
- *JProgressBar* communicates progress of some work.
- *JToolTip* describes purpose of another component.
- *Jtree* a component that displays hierarchical data in outline form.
- *Jtable* a component user to edit and display data in a two-dimensional grid.
- *JTextArea*, *JTextPane*, *JEditorPane*
 - define multi-line areas for displaying, entering, and editing text.

21



Composant de base (pour obtenir des données); contrôles

- *JButton*
- *JCheckBox* a toggled on/off button displaying state to user.
- *JRadioButton* a toggled on/off button displaying its state to user.
- *JComboBox* a drop-down list with optional editable text field. The user can key in a value or select a value from drop-down list.
- *JList* allows a user to select one or more items from a list.
- *JMenu* popup list of items from which the user can select.
- *JSlider* lets user select a value by sliding a knob.
- *TextField* area for entering a single line of input.

22



Composants intermédiaires

- Utilisés pour organiser ou positionner d'autres composants (de base)
 - *JPanel* utilisé pour regrouper d'autres composants
 - *JScrollPane* fournir une vue avec scroll bars
 - *JSplitPane* divise en 2 composants
 - ...

23



Exemple d'organisation des composants

ñ Ajouter 2 boutons dans un Panel

```
JPanel p = new JPanel();  
p.add(new JButton("on"));  
p.add(new JButton("off"));
```

- Ce Panel contient maintenant 2 boutons

24

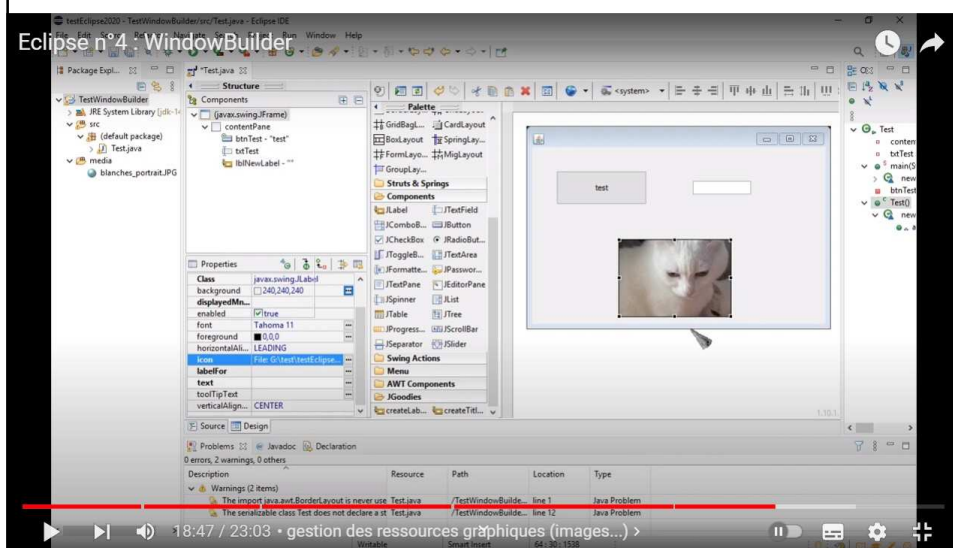
Avec plugin WindowBuilder et Eclipse

VOIR AUTRES SLIDES

25

Tuto-video E mds

Eclipse n°4 : WindowBuilder



XH

26

26



URL tuto E_mds (suite)

- <https://www.youtube.com/watch?v=pQfbr3hpw04>

Intégration de l'outil WindowBuilder dans Eclipse pour pouvoir construire de façon visuelle, une interface graphique.

00:00 : téléchargement et configuration de WindowBuilder

03:00 : création d'une JFrame avec WindowBuilder

05:56 : construction de l'interface graphique

10:43 : gestion des événements

16:15 : gestion des ressources graphiques (images...)

XH

27

27



END



XH

28

28