

# Interfaces Graphiques

διάλογος

Jean-Baptiste.Yunes@u-paris.fr

Université Paris Diderot

©2024

- Boîte de dialogue (*Dialog Box*) ?
  - une fenêtre avec titre et bordure, réclamant une interaction basique (simple) avec l'utilisateur :
  - **modale** (bloquante)
    - oblige à une réponse avant de permettre de continuer
  - **non modale** (non bloquante)
    - informatif

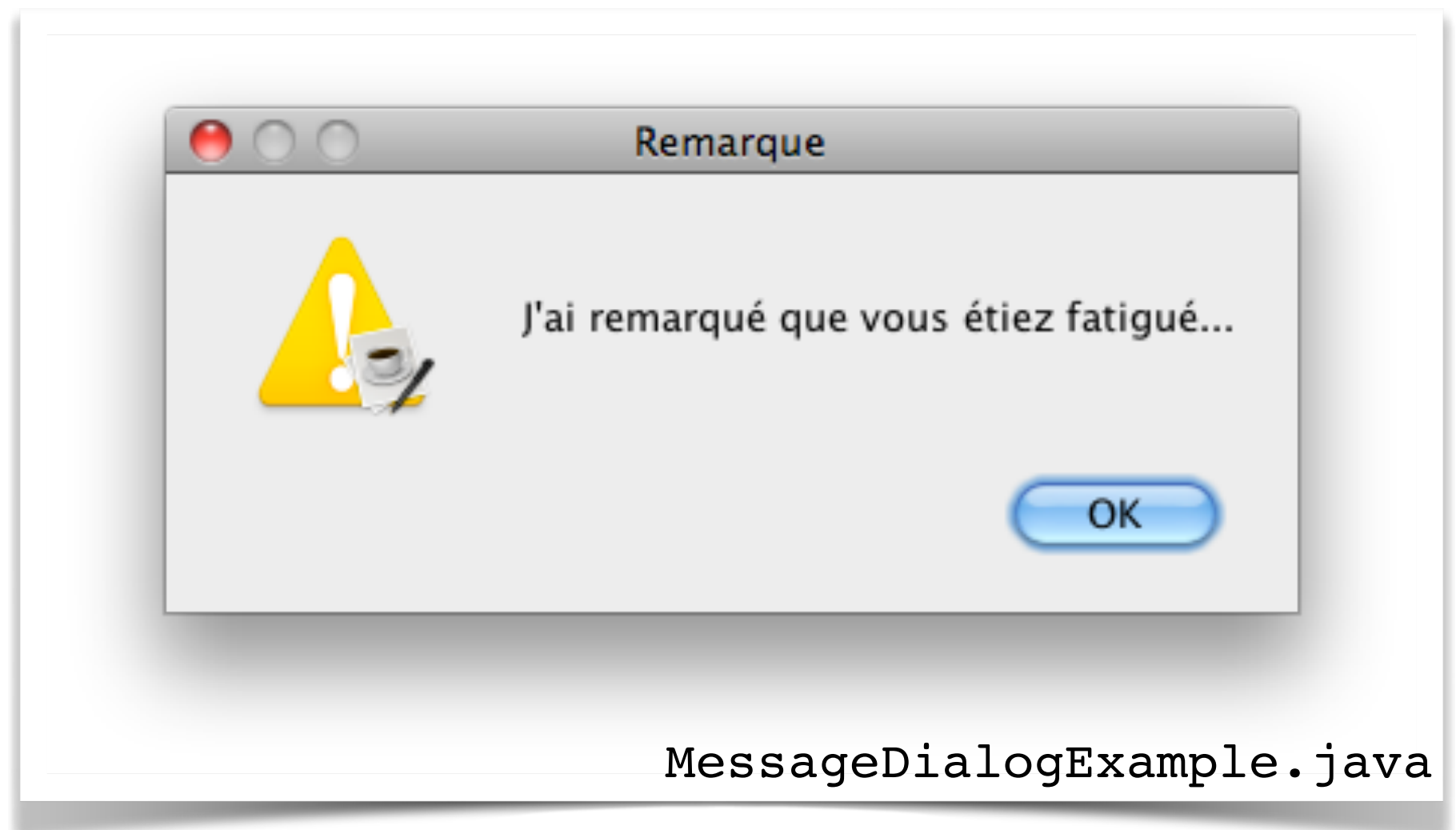
- La classe `JDialog` est la classe mère de tous les dialogues swing
- on l'utilise comme n'importe quelle autre fenêtre
  - on compose en son sein une interface
  - on s'y branche pour détecter des actions
  - pour terminer il suffit de rendre le dialogue invisible : `setVisible(false)`

- Depuis Java SE 6, quatre niveaux de modalité :
  - **sans** (modeless)
  - **document-modal**
    - bloque tous les descendants d'une fenêtre sans parent
  - **application-modal**
    - bloque toutes les fenêtres d'une application
  - **toolkit-modal**
    - n'avait de sens que pour les Applets...

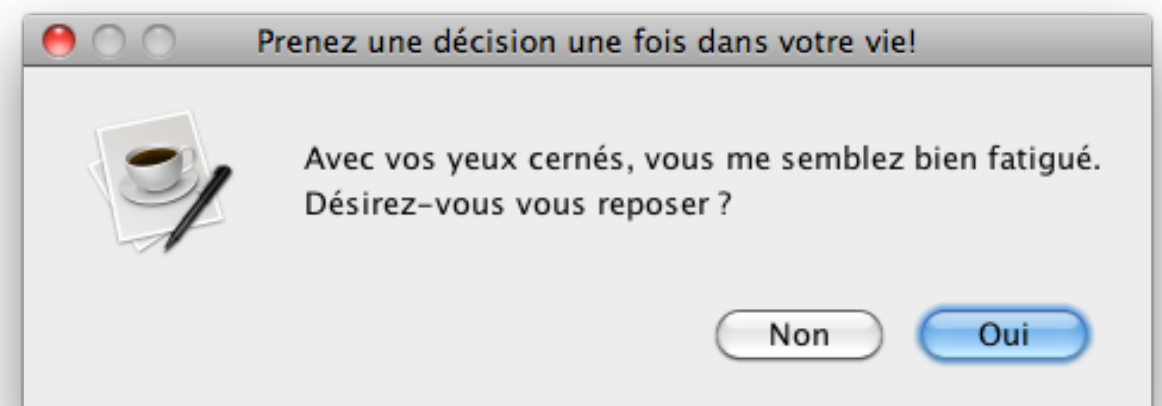
- JDialog sans propriétaire : non modal
  - ex : ModelessExample.java
- JDialog avec propriétaire : statut de l'icônification de la fenêtre transmis aux dialogues attachés
  - ex : ModelessExample.java (modifié)
- ex : DocModalExample.java
- ex : AppModalExample.java

- la classe `JOptionPane` est très utile
- elle permet d'obtenir de nombreux dialogues standardisés pour :
  - message informatif
  - confirmation (oui/non/cancel)
  - options (choix)
  - entrée

- `showMessageDialog`
- un titre, une phrase, ok
- un type : `ERROR_MESSAGE`, `QUESTION_MESSAGE`, `INFORMATION_MESSAGE`, `WARNING_MESSAGE`, `PLAIN_MESSAGE`



- `showConfirmDialog`
  - un titre, une question, oui - non - annuler
  - un type : `ERROR_MESSAGE`, `QUESTION_MESSAGE`, `INFORMATION_MESSAGE`, `WARNING_MESSAGE`, `PLAIN_MESSAGE`
- valeurs renvoyées : `YES_OPTION` (ou `OK_OPTION`), `NO_OPTION`, `OK_OPTION`, `CANCEL_OPTION`, `CLOSED_OPTION`





- `showOptionDialog`
  - un titre, une question, des options, une option par défaut
  - renvoie l'indice de l'option dans la collection, ou `CLOSED_OPTION`

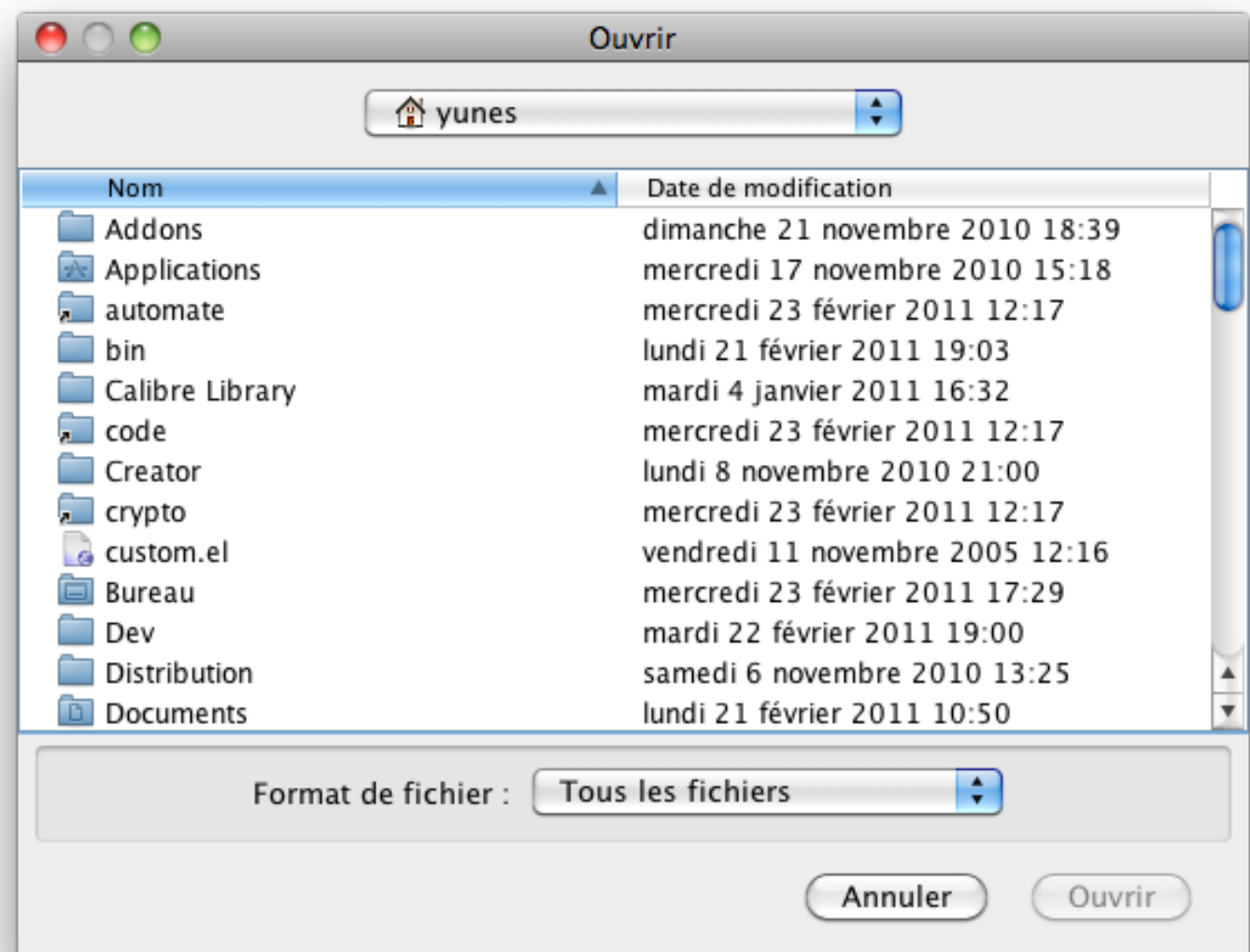


- `showInputDialog`
  - un titre, une question, des options, une option par défaut
  - renvoie l'objet sélectionné ou `null`



- Un autre type de dialogue utile est très courant est celui permettant de choisir un fichier en vue de le lire ou d'y écrire
- la classe `JFileChooser`
  - `showOpenDialog`
  - `showSaveDialog`
    - renvoient `CANCEL_OPTION`, `APPROVE_OPTION` ou `ERROR_OPTION`

- constructeurs
  - `JFileChooser()` dans le répertoire par défaut
  - `JFileChooser(File)` placé dans le répertoire
  - `JFileChooser(String)` placé dans le répertoire

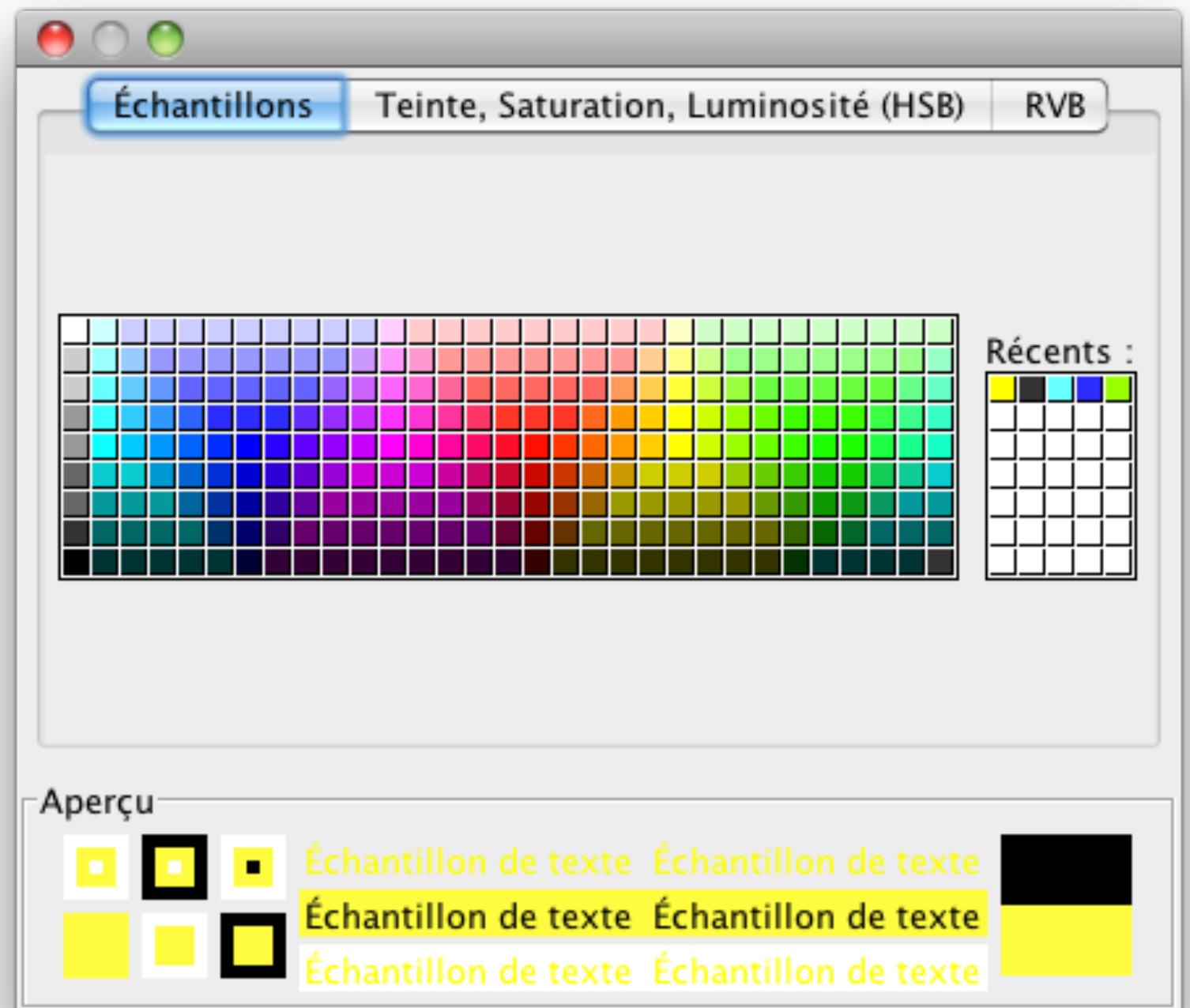


- une fois revenu de `show*Dialog`
  - on peut interroger l'instance pour récupérer la sélection
    - `File getSelectedFile()`
    - etc.
- Rappel : `File` représente un objet quelconque du système de fichier...

- `setMultiSelectionEnabled(boolean)`
- `setFileSelectionMode(int mode)`
  - `FILES_ONLY,`  
`DIRECTORIES_ONLY,`  
`FILES_AND_DIRECTORIES`
- `setFileHiddingEnabled(boolean)`
- `setFileFilter(FileFilter)`
- `addChoosableFileFilter(FileFilter)`

- la classe abstraite `FileFilter` possède deux méthodes
  - `boolean accept(File)`
    - permet de filtrer ou non le fichier en paramètre
  - `String getDescription()`
    - à des fins descriptives
- Il existe une implémentation de base
  - `FileNameExtensionFilter`

- une autre dialogue typique est celui permettant la sélection d'une couleur
- la classe JColorChooser





- trois fonctionnements possibles offerts par défaut
  - dialogue modal renvoyant la couleur sélectionnée
    - `Color showDialog(...)`
  - dialogue avec `Listener`
    - `JDialog createDialog(...)`
- utilisation du `JColorChooser` comme composant et *capture* des modifications de couleur

- Rappel : Swing utilise le MVC pour ses composants
  - JButton  $\rightsquigarrow$  ButtonModel
    - getModel()
  - JColorChooser  $\rightsquigarrow$  ColorSelectionModel
    - getSelectionModel()

- L'interface

`javax.swing.colorchooser.ColorSelectionModel`

- `void addChangeListener(ChangeListener)`
- `void removeChangeListener(ChangeListener)`
- `Color getSelectedColor()`
- `void setSelectedColor(Color)`

- l'interface `javax.swing.event.ChangeListener`
- `void stateChanged(ChangeEvent)`

