ProgIHM - Cours 5

Les Boites de Dialogue, debuggage d'IHM et autres outils du développeur

V. DESLANDRES

veronique.deslandres@univ-lyon1.fr

IUT Lyon 1

l'excellence technologique

Sommaire de ce cours

Les Boites de dialogue

Les boîtes de dialogue standard

- On utilise la classe `JOptionPane` pour les boîtes de dialogue standard, prêtes à l'emploi
  - Cette classe propose des méthodes **statiques : `showXXXDialog()`**

- Quatre types de boîtes
  - `MessageDialog` pour afficher un message
  - `ConfirmDialog` pour une réponse de l'utilisateur avec `Yes`, `No` et `Cancel`
  - `InputDialog` pour une invite de saisie

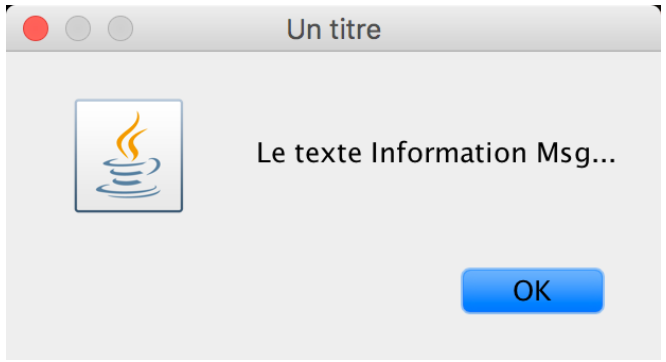
  - `OptionDialog` qui rassemble les caractéristiques des 3 autres boîtes de dialogue

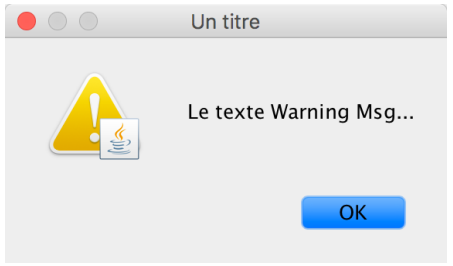
Message Dialog (1/2)

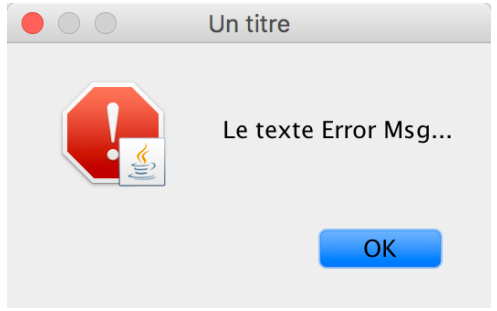
- Un texte, un bouton OK et éventuellement un icône prédéfini
- Ne retourne rien (void)

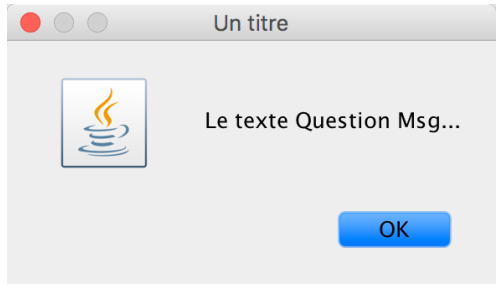
```
JOptionPane.showMessageDialog(fen, "Le texte Information Msg...", "Un titre", JOptionPane.INFORMATION_MESSAGE);
JOptionPane.showMessageDialog(fen, "Le texte Warning Msg...", "Un titre", JOptionPane.WARNING_MESSAGE);
JOptionPane.showMessageDialog(fen, "Le texte Error Msg...", "Un titre", JOptionPane.ERROR_MESSAGE);
JOptionPane.showMessageDialog(fen, "Le texte Question Msg...", "Un titre", JOptionPane.QUESTION_MESSAGE);
JOptionPane.showMessageDialog(fen, "Le texte Plain Msg...", "Un titre", JOptionPane.PLAIN_MESSAGE);
```

Fenêtre parent → le texte → le titre → type d'icône (énumeration)



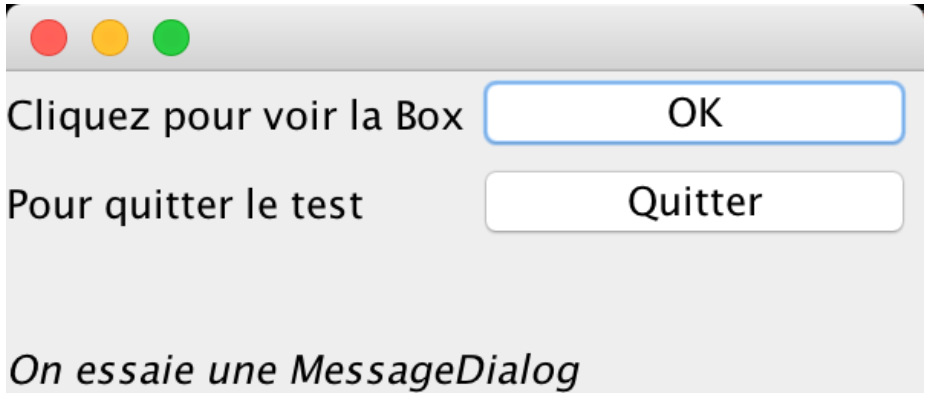


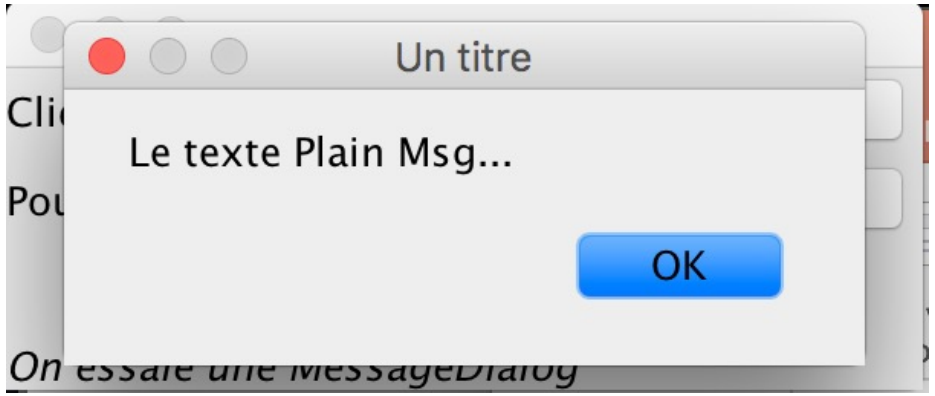




Message Dialog (2/2)

- Le 1er argument est une `JFrame`
- Si on met **null**, ça marche aussi et la fenêtre sera **centrée sur l'écran**
- Si on met une fenêtre parent, la fenêtre Dialog sera centrée sur cette dernière :





- Ceci est valable pour **toutes** les fenêtres de dialogue

Confirm Dialog

- Une question, de 1 à 3 boutons (Oui/OK, Non, Annuler) et un icône
- Retourne un *int* correspondant à l'énumération sélectionnée

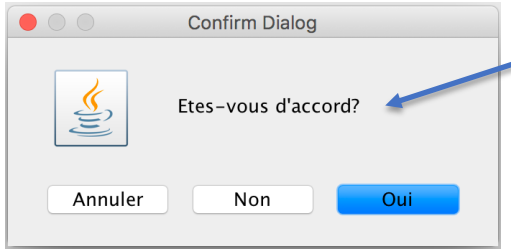
```
JOptionPane.showConfirmDialog(fen,"Etes-vous d'accord?","Confirm Dialog",JOptionPane.YES_NO_CANCEL_OPTION);
JOptionPane.showConfirmDialog(fen,"Etes-vous d'accord?","Confirm Dialog",JOptionPane.YES_NO_OPTION);
JOptionPane.showConfirmDialog(fen,"Etes-vous d'accord?","Confirm Dialog",JOptionPane.OK_CANCEL_OPTION);
JOptionPane.showConfirmDialog(fen,"Etes-vous d'accord?","Confirm Dialog",JOptionPane.DEFAULT_OPTION);
```

Fenêtre parent

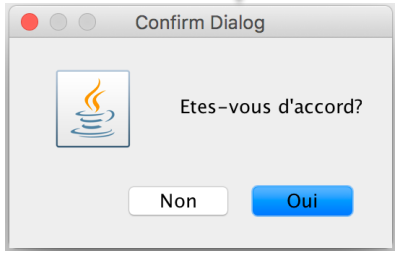
message

titre

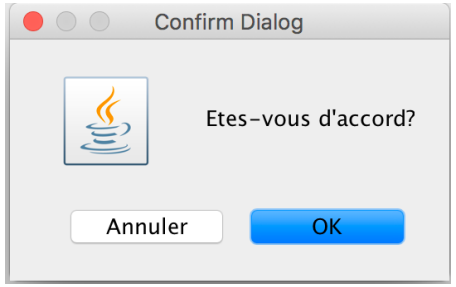
Boutons à afficher (énumération)



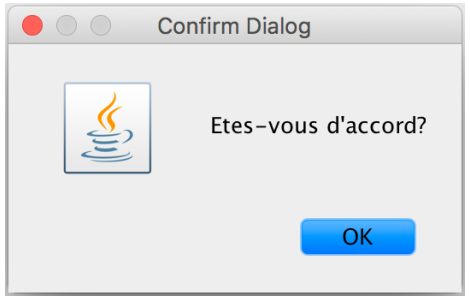
YES_NO_CANCEL_OPTION



YES_NO_OPTION



OK_CANCEL_OPTION



DEFAULT_OPTION

Récupérer la réponse :

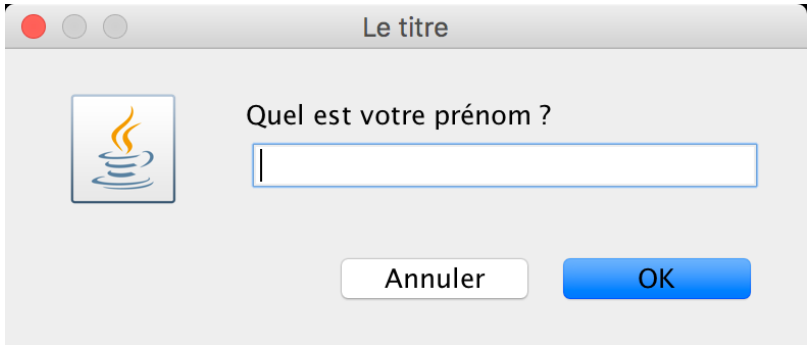
int **reply** = JOptionPane.showConfirmDialog(null, message, title, JOptionPane.YES_NO_OPTION);
if (**reply** == JOptionPane.YES_OPTION) …

Input Dialog : boîte de saisie

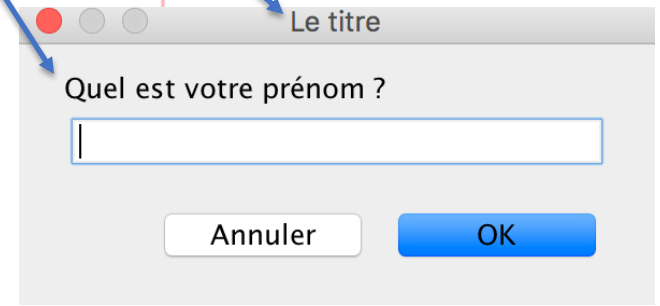
- Une question, une invite de saisie, 2 boutons (OK et annuler) et un icône
- Retourne une **chaîne**, correspondant au champ saisi

```
String message = "Quel est votre prénom ?";

String reponse = JOptionPane.showInputDialog(this, message, "Le titre", JOptionPane.PLAIN_MESSAGE);
// ou WARNING_MESSAGE, INFORMATION_MESSAGE
// ou ERROR_MESSAGE, PLAIN_MESSAGE);
```

Fenêtre parent

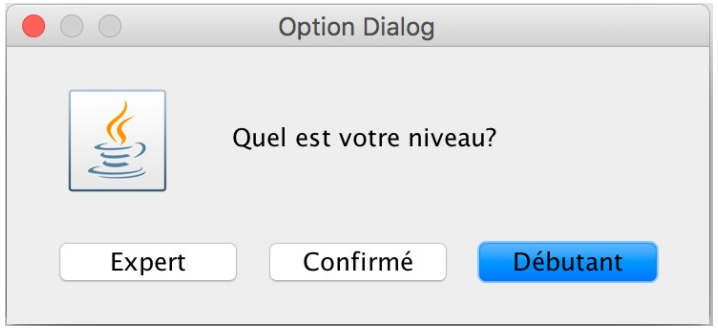


(icône : *INFORMATION_MESSAGE* ou *QUESTION_MESSAGE*)



(aucun icône : JOptionPane.*PLAIN_MESSAGE)*

Option Dialog (1/2)



Elle permet de définir ses propres options de réponses.

Caractéristiques :
- Une question avec des boutons représentant des choix différents et un icône

```
int showOptionDialog(Component parentComponent,
                     Object message,
                     String title,
                     int optionType,
                     int messageType,
                     Icon icon,
                     Object[] options,
                     Object initialValue)
```

Soit *optionType* est à 0 : on définit des `options`

Soit un *optionType* est fourni (ex. YES_NO_OPTION), et donc `options` est **null**

Ouvre une fenêtre de dialogue avec **l'icône proposé**, un texte avec le **message** proposé, les **options** de réponses possibles, et le choix initial prédéfini par le paramètre initialValue.

La méthode renvoie l'indice de l'option choisie (dans le tableau options) par l'utilisateur.

Option Dialog (2/2)

```
String[] choix={"Débutant", "Confirmé", "Expert"};
JOptionPane.showOptionDialog(fen,"Quel est votre niveau?","Option Dialog",0,JOptionPane.INFORMATION_MESSAGE, null, choix, choix[0]);
JOptionPane.showOptionDialog(fen,"Quel est votre niveau?","Option Dialog",0,JOptionPane.ERROR_MESSAGE, null, choix, choix[0]);
JOptionPane.showOptionDialog(fen,"Quel est votre niveau?","Option Dialog",0,JOptionPane.WARNING_MESSAGE, null, choix, choix[0]);
JOptionPane.showOptionDialog(fen,"Quel est votre niveau?","Option Dialog",0,JOptionPane.QUESTION_MESSAGE, null, choix, choix[0]);
JOptionPane.showOptionDialog(fen,"Quel est votre niveau?","Option Dialog",0,JOptionPane.PLAIN_MESSAGE, null, choix, choix[1]);
```

(on propose des boutons)

Fenêtre parent

Texte du msg

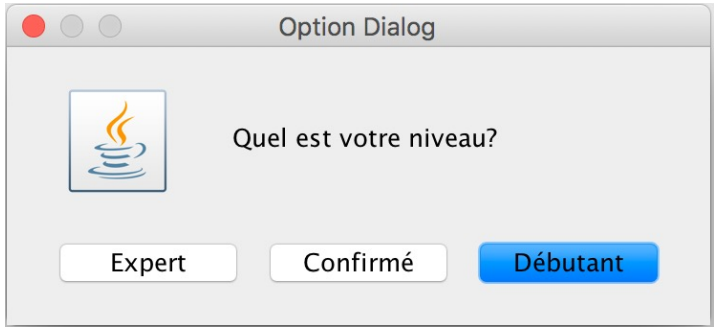
Titre

type icône

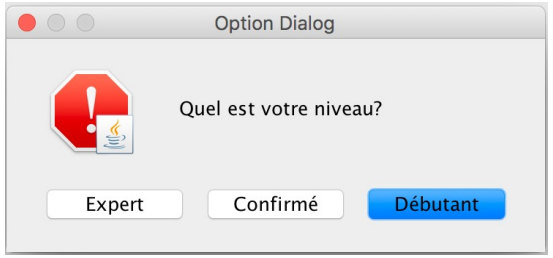
(icône éventuel)

liste des boutons

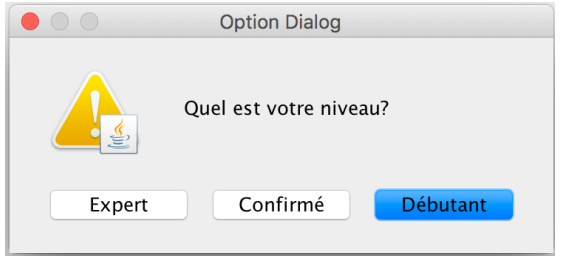
Choix présélectionné



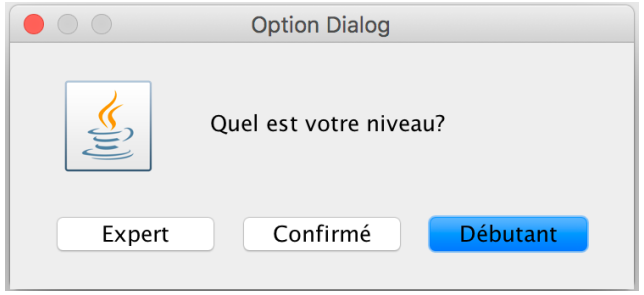
INFORMATION_MESSAGE



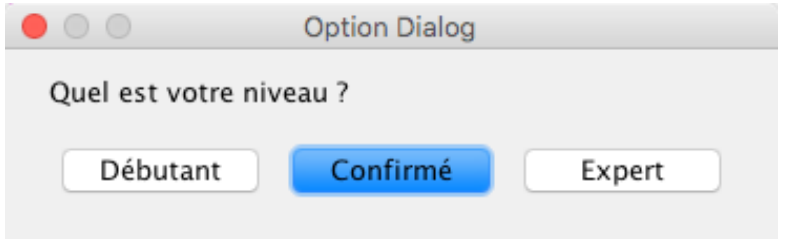
ERROR_MESSAGE



WARNING_MESSAGE



QUESTION_MESSAGE



PLAIN_MESSAGE

Exemples d'utilisation

Cf aussi :
FenetreDeDialogueDemo.zip

Confirm Dialog

```
int response=JOptionPane.showConfirmDialog(fen,"Etes-vous d'accord?","Confirm Dialog",
        JOptionPane.YES_NO_CANCEL_OPTION);
if(response==JOptionPane.YES_OPTION)
    System.out.println("Yes Option !");
if(response==JOptionPane.NO_OPTION)
    System.out.println("No Option !");
if(response==JOptionPane.CANCEL_OPTION)
    System.out.println("Cancel Option !");
if(response==JOptionPane.CLOSED_OPTION)
    System.out.println("Closed Option !");
```

Input Dialog

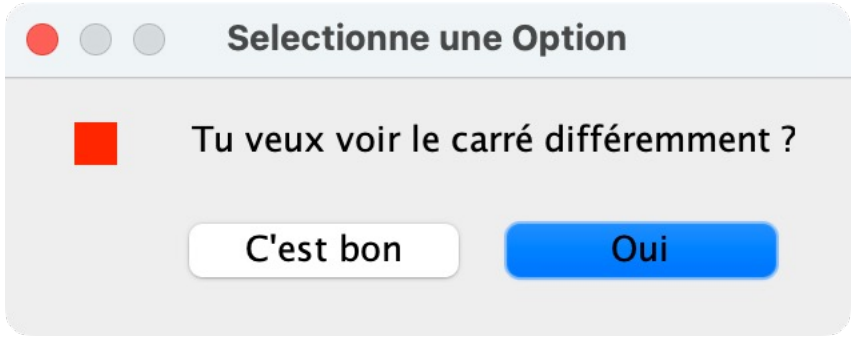
```
String reponse = JOptionPane.showInputDialog(fen, "Quel est votre prénom ?", "Le titre",
        JOptionPane.INFORMATION_MESSAGE);
if (reponse.length() != 0) {
    System.out.println("Bonjour " + reponse);
```

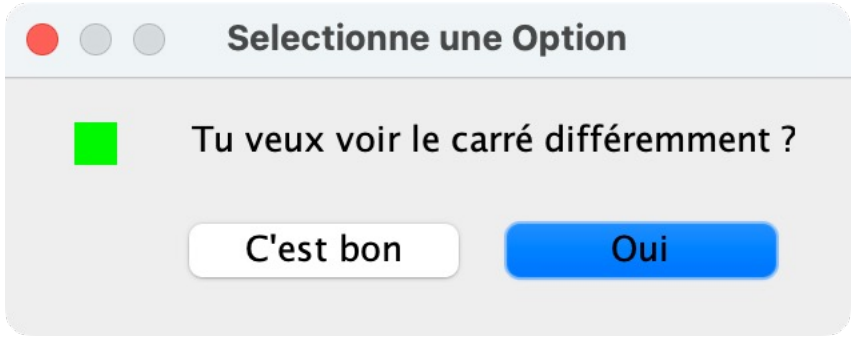
Option Dialog

```
int response=JOptionPane.showOptionDialog(fen,"Quel est votre niveau?","Option Dialog",0,
        JOptionPane.INFORMATION_MESSAGE, null, choix, choix[0]);
if(response==JOptionPane.CLOSED_OPTION)
    System.out.println("Pas de formule choisie !");
else
    System.out.println("Vous avez choisi: "+choix[response]);
```

Un exemple d'OptionDialog avec un icône

- Ici on affiche un carré de couleur qui change quand l'utilisateur clique sur le bouton 'Oui' :





- Pour cela, on définit notre propre icône :

```
Icon blueIcon = new MyIcon(Color.RED);
```

- On l'envoie à JOptionPane :

(parce qu'on exécute directement depuis le main)

```
int reponse = JOptionPane.showOptionDialog(new JDesktopPane(),
"Tu veux voir le carré différemment ?", "Selectionne une Option",
JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, blueIcon,
stringArray, stringArray[0]);
```

La classe interne MyIcon

```
class MyIcon implements Icon {
    Color myColor;
    public MyIcon(Color myColor) {
        this.myColor = myColor;
    }
    @Override
    public int getIconWidth() {
        return 16;
    }
    @Override
    public int getIconHeight() {
        return 16;
    }
```

```
@Override
public void paintIcon(Component c,
Graphics g, int x, int y) {
        g.setColor(myColor);
        g.drawRect(0, 0, 16, 16);
        g.fillRect(0, 0, 16, 16);
    }

public Color getColor() {  // utile
pour brighter() / darker()...
        return myColor;
    }
}
```

Et le *main()*

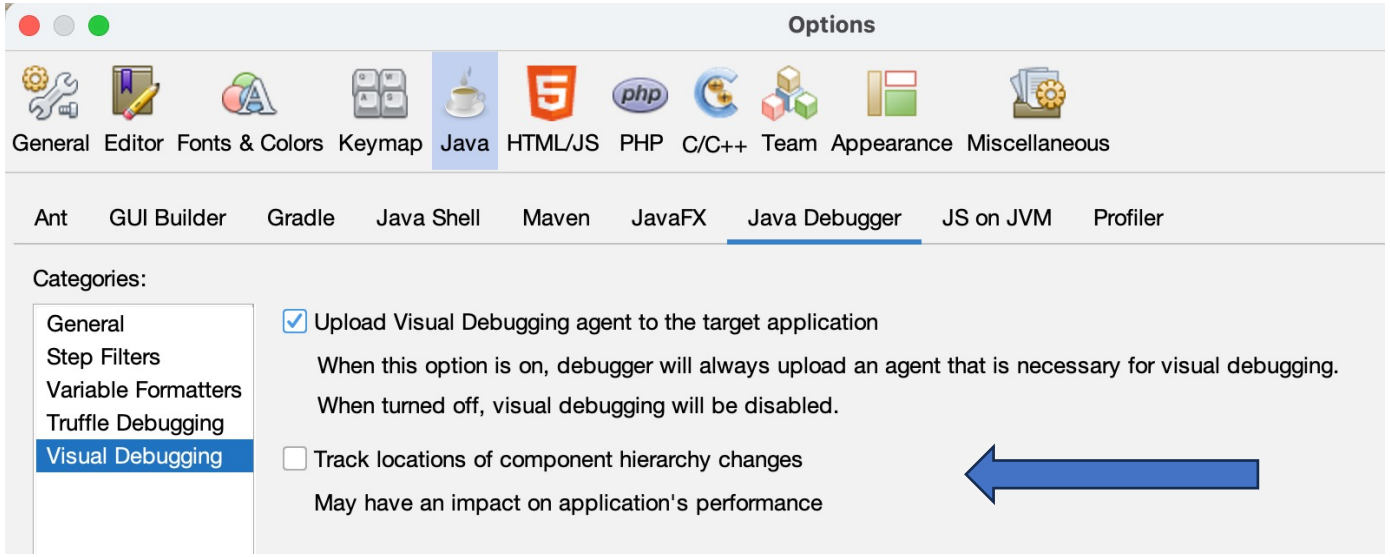
```
public static void main(String[] a) {

    Icon blueIcon = new MyIcon(Color.RED);

    Object stringArray[] = {"Oui", "C'est bon"};

    int reponse, compteur;
    reponse = compteur = 0;
    while (reponse != 1) {
        reponse = JOptionPane.showOptionDialog(new JDesktopPane(), "Tu veux voir le carré différemment ?",
                JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, blueIcon, stringArray,
                stringArray[0]);
        if ((compteur % 2) != 1) { // tous les 2 passages, on change la couleur...
            blueIcon = new MyIcon(Color.GREEN);
            //((MyIcon) blueIcon).getColor().darker(); // pas flagrant !
        } else {
            blueIcon = new MyIcon(Color.RED);
        }

        if (reponse == 1) {
            break;
        }
        compteur++;
    }

}
```

Les outils de debuggage d'IHM

https://netbeans.apache.org/kb/docs/java/debug-visual.html

Fonctionnalités

- A partir de l'IHM, on peut :
  - Voir la déclaration des composants
  - Voir leur code source (leur définition)
  - On peut voir dans le navigateur d'objets, la position du composant sur son conteneur
  - Si on modifie l'option **Track locations of component hierarchy changes**, on peut voir le code où le composant est ajouté dans son conteneur
    - **Aller dans *Debug configuration* et cocher la case (très lent) :**



Démo !

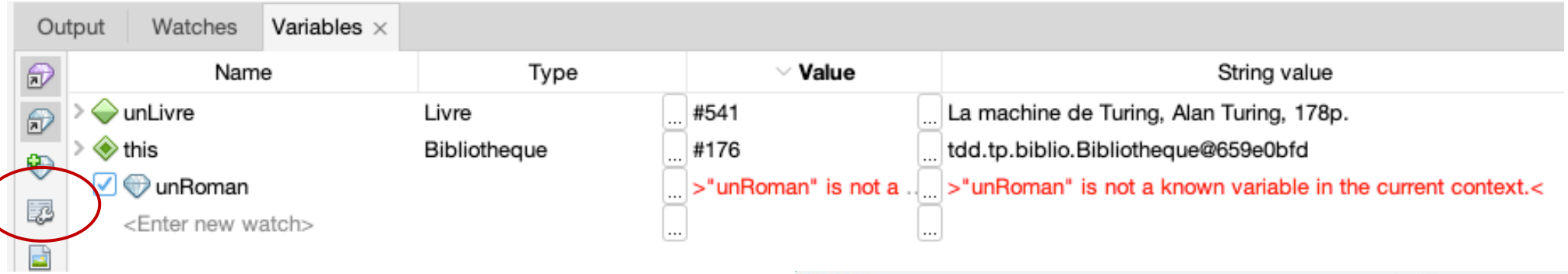
Le debuggage

Debuggage classique

- Avec le debugger, on peut :
  - Démarrer le debuggage à partir d'une instruction donnée (Menu Debug – **Run to cursor**)
  - Tout debugguer ligne par ligne depuis le début : fichier, clic droit : **Debug File**
  - Entrer dans le code des méthodes avec **Step Info**
  - Ou au contraire, passer la méthode et aller à la suite avec **Step Over (F8)**
    - Ou **Step Over Expression** (comportement différent de F8 uniquement en cas d'appels multiples sur la même instruction)
  - Voir le **contenu des variables** du contexte courant avec la fenêtre de Debug, onglet Variables (ajouter des variables : saisie du texte, auto-complétion)
  - Voir certaines variables qu'on souhaite surveiller entre plusieurs sessions de debuggage : on les appelle '**watches**'. On peut les voir dans l'onglet Watch.

Configuration du Debuggueur

- On accède au menu de configuration soit en cours de debuggage :



- Soit depuis les options de Netbeans :