Première fenêtre en Java

Utilisation de Window Builder Pro

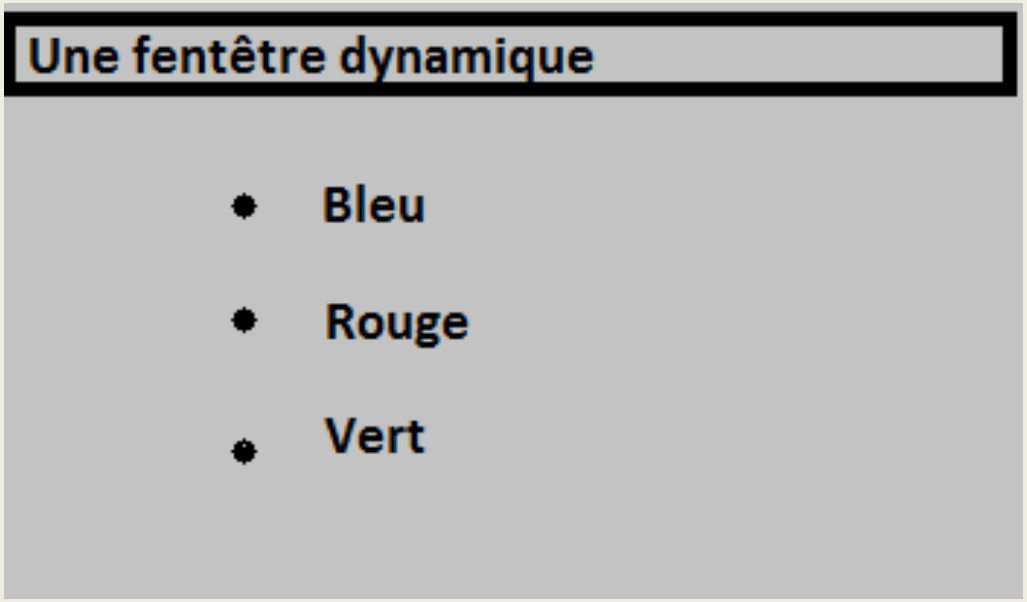
Description

Une application utilisant des fenêtres est appelée Inetrface Homme Machine ou IHM. Ce nom provient du fait que l'IHM permet une communication aisée entre l'utilisateur (Homme) et l'ordinateur (Machine) dans les deux sens.

Nous souhaitons créer une première fenêtre en Java qui permet de choisir la couleur du fond de la fenêtre grâce à des boutons de type *radio*.

Pour créer une IHM, il faut commencer par visualiser sous la forme papier le résultat souhaité, puis passer au codage.

Voici un dessin de l'interface souhaitée :

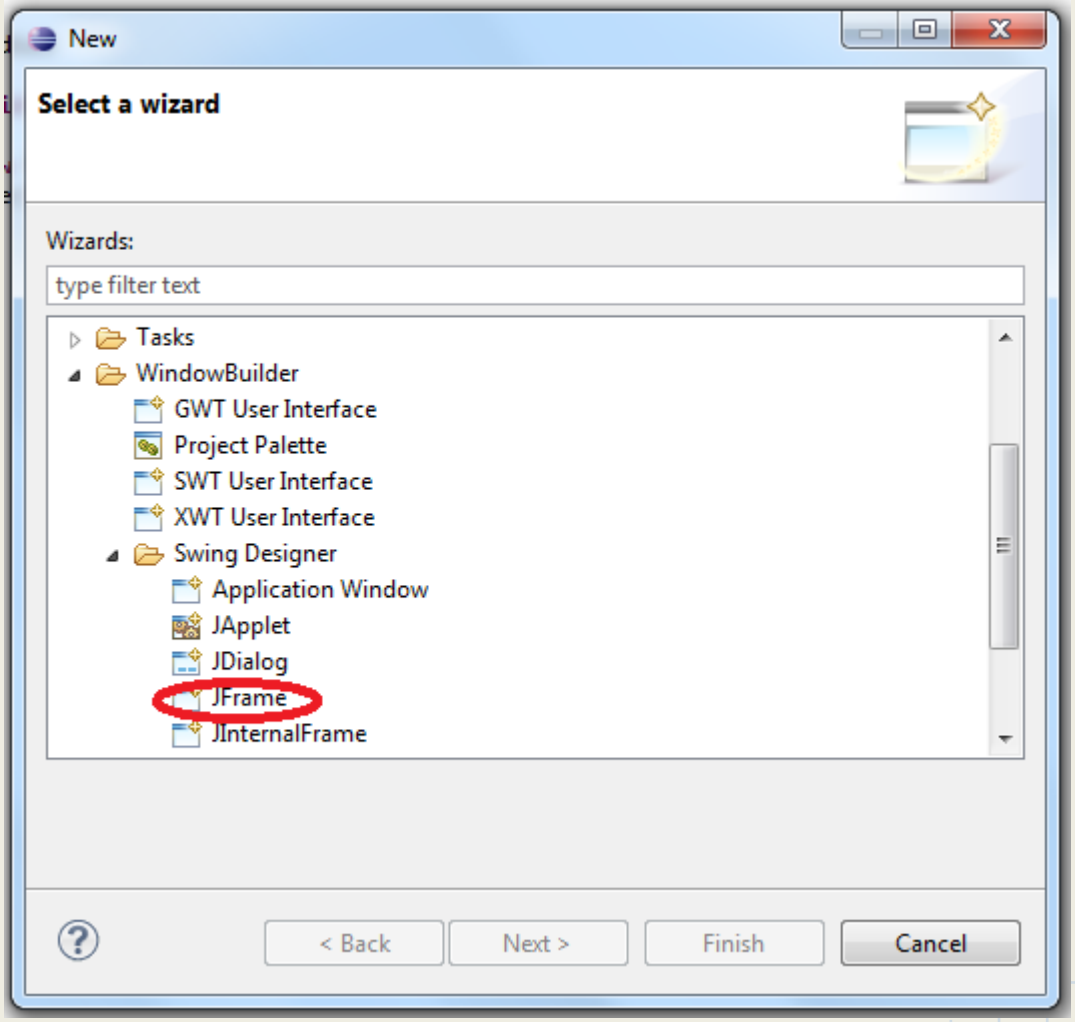


Mise en place de la fenêtre avec Eclipse

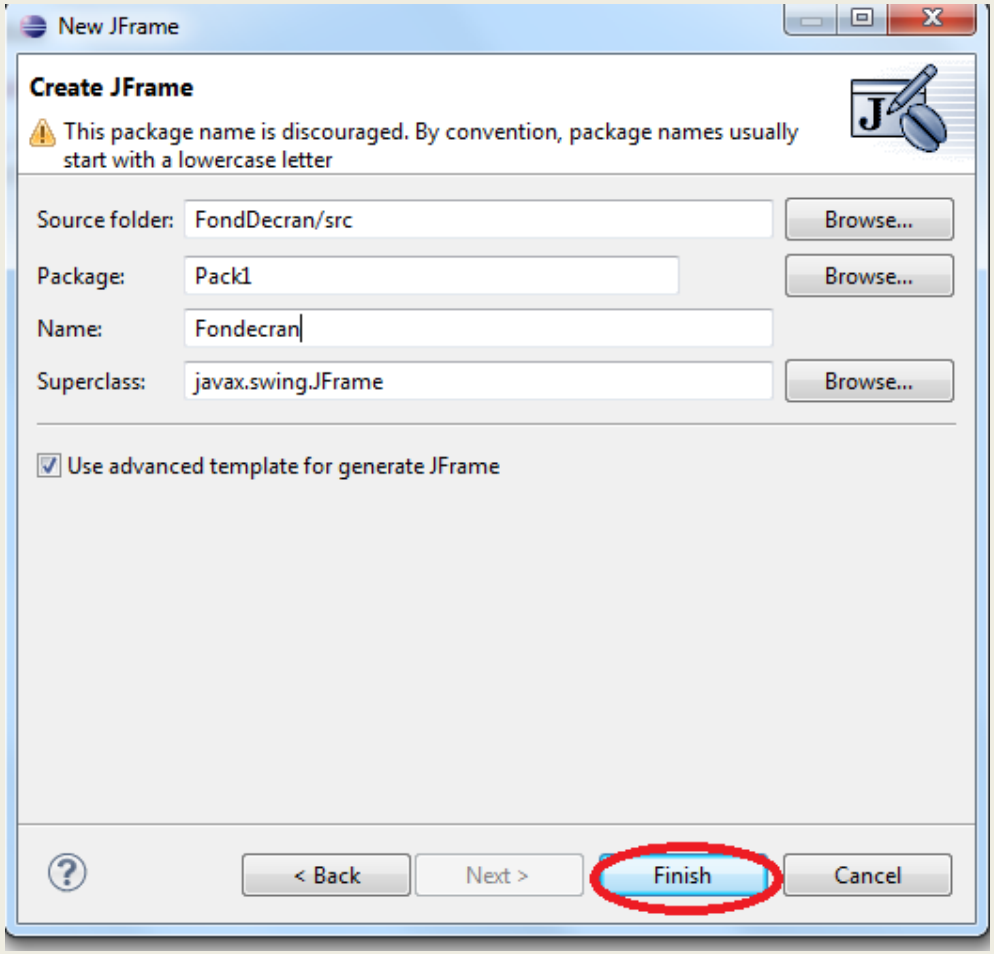
La version *juno d'Eclipse* contient l'utilitaire *Window Builder Pro* qui est un outil développé pour créer de manière simple des IHM.

Pour commencer nous allons créer une nouveau *projet* Java : fondecran, puis un *package* : pack1.

Ensuite il faut effectuer un clic droit sur le package et choisir : *New - Other* puis choisir dans le répertoire *WindowBuilder - Swing Designer* le type *JFrame* comme ci-dessous :


New
Select a wizard
Wizards:
type filter text
Tasks
WindowBuilder
GWT User Interface
Project Palette
SWT User Interface
XWT User Interface
Swing Designer
Application Window
JApplet
JDialog
JFrame
JInternalFrame
< Back
Next >
Finish
Cancel

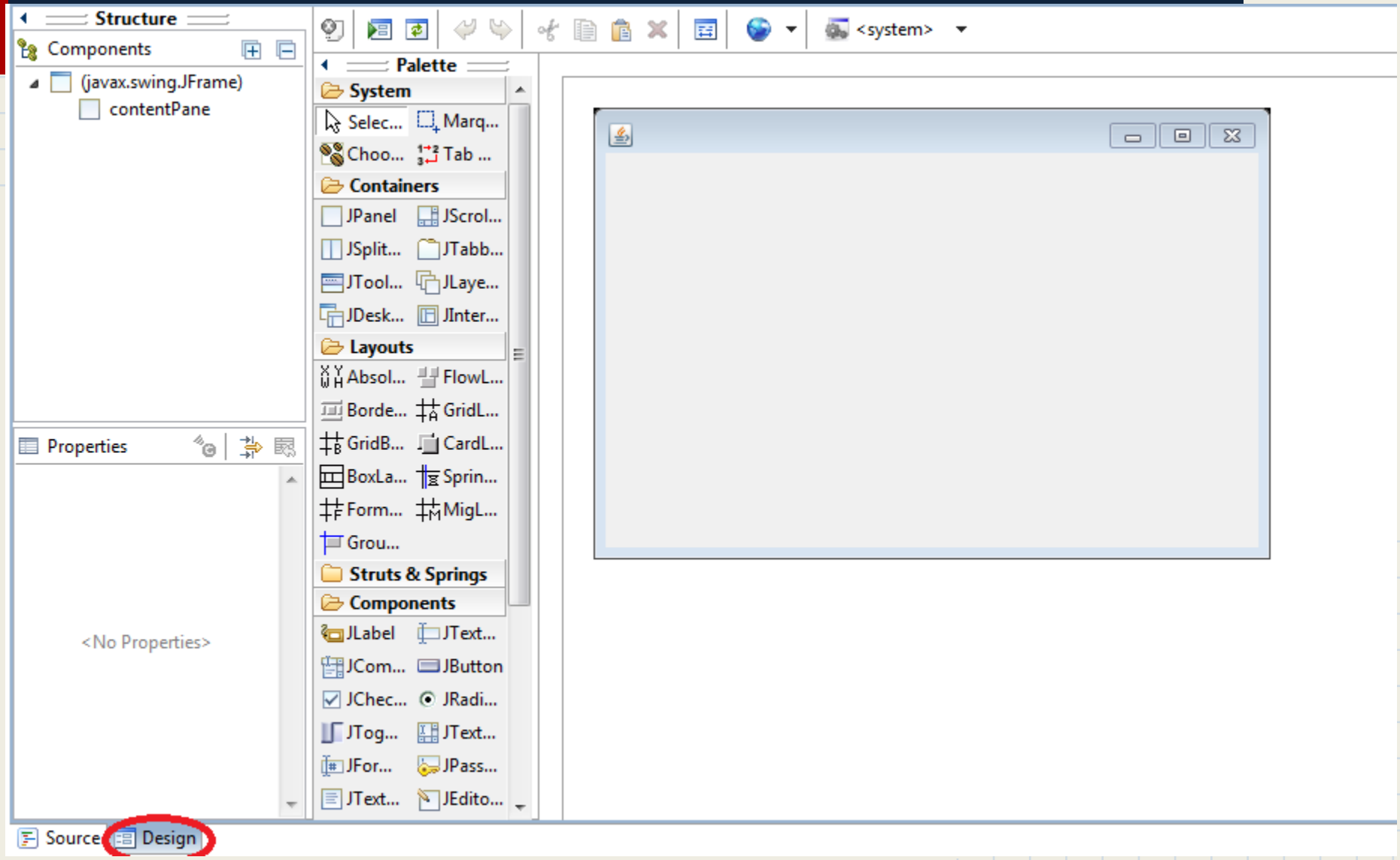
On peut alors l'appeler *Fondecran* par exemple :



Eclipse crée alors automatiquement une nouvelle classe *Fondecran* qui est une extension de la classe *JFrame*. Cela signifie que mon *Fondecran* aura toutes les caractéristiques que possèdent les fenêtres du type *JFrame*. Il possédera par exemple une taille, une couleur de fond, ...

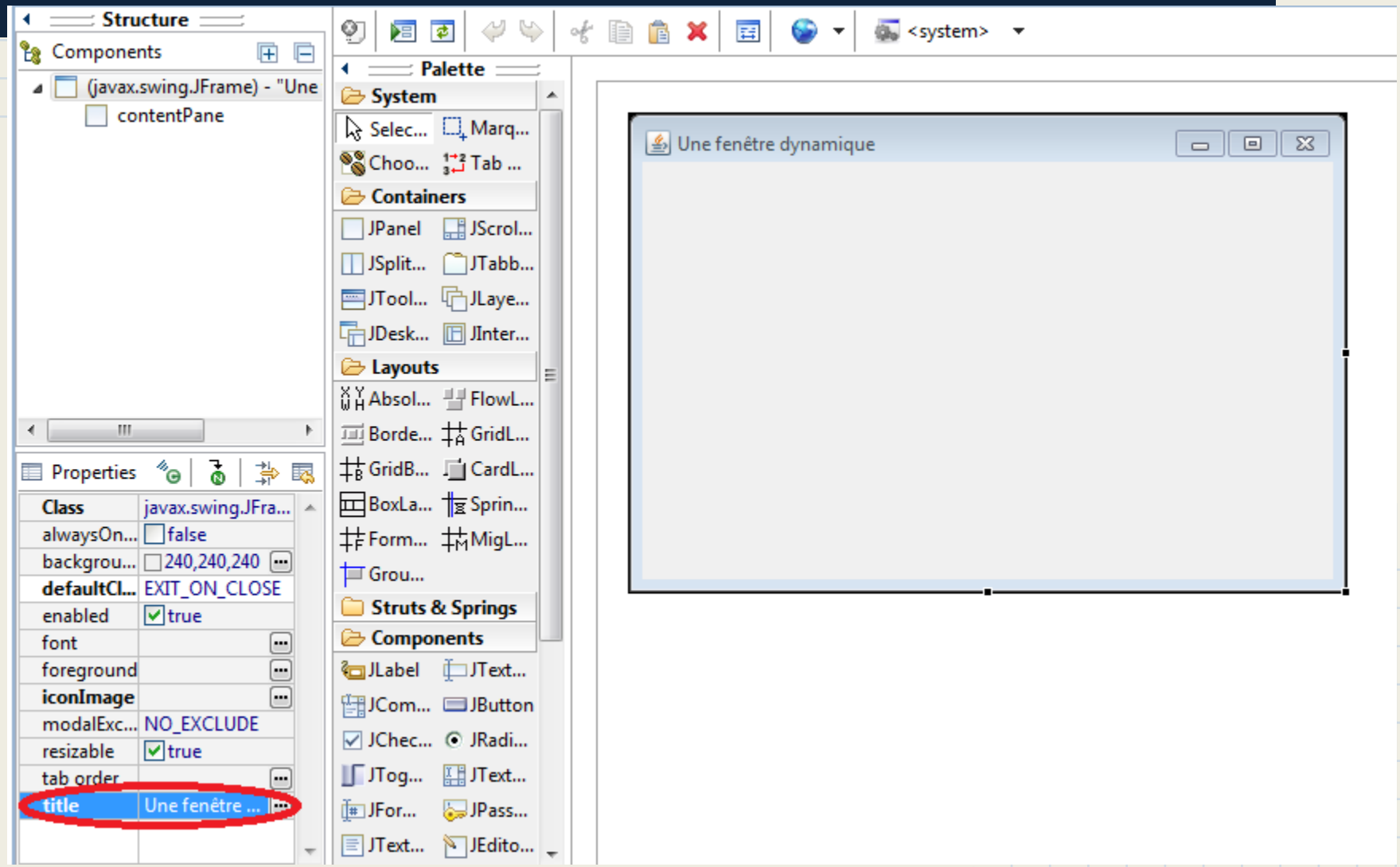
Un code est généré automatiquement et ce sera le cas pour chaque changement que nous allons faire avec la souris sur la fenêtre.

Nous allons maintenant passer en mode *Design*, en sélectionnant l'onglet correspondant :


Structure
Components
(javax.swing.JFrame)
contentPane
Properties
<No Properties>
Palette
System
Selec...
Marq...
Choo...
Tab ...
Containers
JPanel
JScrol...
JSplit...
JTabb...
JTool...
JLaye...
JDesk...
JInter...
Layouts
Absol...
FlowL...
Borde...
GridL...
GridB...
CardL...
BoxLa...
Sprin...
Form...
MigL...
Grou...
Struts & Springs
Components
JLabel
JText...
JCom...
JButton
JChec...
JRadi...
JTog...
JText...
JFor...
JPass...
JText...
JEdito...
<system>
Source
Design

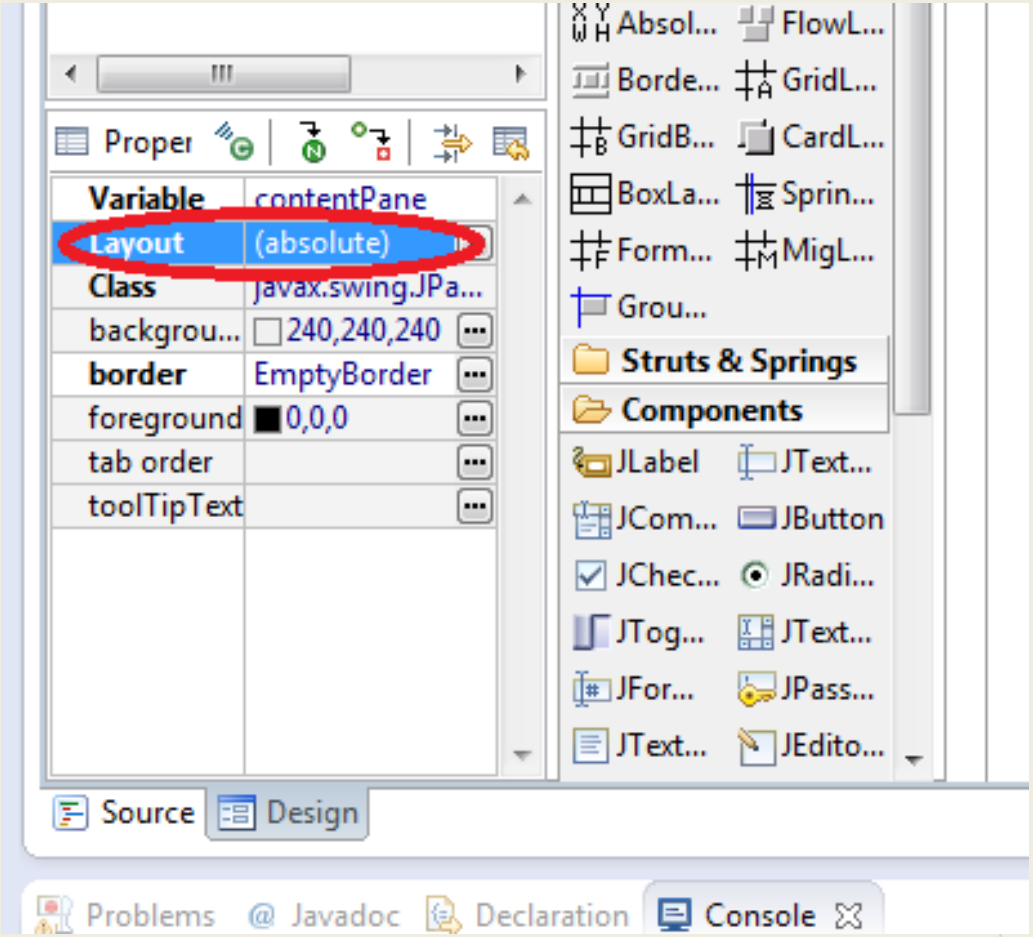
Nous visualisons notre fenêtre, qui pour l'instant de contient rien d'intéressant.
Vous pouvez alors effectuer un test en appuyant sur le bouton *run*.
Vous constatez que votre fenêtre vide s'ouvre. Refermez-la pour pouvoir effectuer des modifications.

Commençons par lui donner un titre : à l'aide de la souris sélectionnez votre fenêtre et sur le partie *Proprietes* aller dans *title* et donner un titre qui convient : Une fenêtre dynamique.


Structure
Components
(javax.swing.JFrame) - "Une
contentPane
Properties
Class
javax.swing.JFra...
alwaysOn...
false
backgrou...
240,240,240
defaultCl...
EXIT_ON_CLOSE
enabled
true
font
foreground
iconImage
modalExc...
NO_EXCLUDE
resizable
true
tab order
title
Une fenêtre ...
Palette
System
Selec...
Marq...
Choo...
Tab ...
Containers
JPanel
JScrol...
JSplit...
JTabb...
JTool...
JLaye...
JDesk...
JInter...
Layouts
Absol...
FlowL...
Borde...
GridL...
GridB...
CardL...
BoxLa...
Sprin...
Form...
MigL...
Grou...
Struts & Springs
Components
JLabel
JText...
JCom...
JButton
JChec...
JRadi...
JTog...
JText...
JFor...
JPass...
JText...
JEdito...
<system>
Une fenêtre dynamique

Pour pouvoir placer des objets sur notre fenêtre (les boutons par exemple) il faut tout d'abord lui donner un endroit ou se placent ces objets : cet endroit s'appelle un *Layout*.

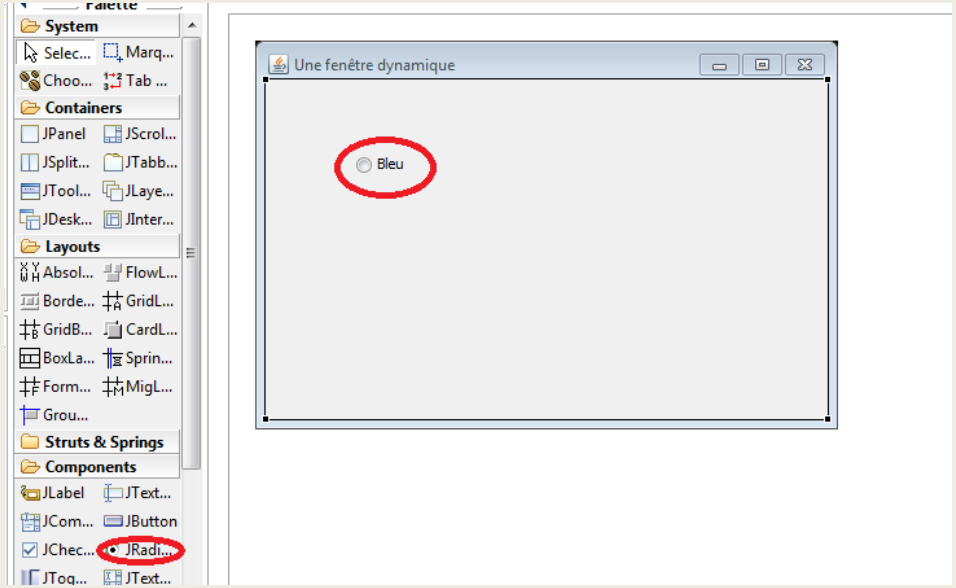
On choisira le *Layout* de type *Absolute Layout* ce qui permettra de placer les objets à tout endroit de la fenêtre.


Proper
Variable
contentPane
Layout
(absolute)
Class
javax.swing.JPa...
backgrou...
240,240,240
border
EmptyBorder
foreground
0,0,0
tab order
toolTipText
Absol...
FlowL...
Borde...
GridL...
GridB...
CardL...
BoxLa...
Sprin...
Form...
MigL...
Grou...
Struts & Springs
Components
JLabel
JText...
JCom...
JButton
JChec...
JRadi...
JTog...
JText...
JFor...
JPass...
JText...
JEdito...
Source
Design
Problems
Javadoc
Declaration
Console

Premier bouton

On va créer un premier bouton, qui va permettre de changer la couleur du fond de la fenêtre.

On sélectionne le *jRadioButton* et on le fait glisser sur notre fenêter. On lui donne un texte : Bleu, par exemple.



Appuyez sur l'onglet *Source* pour visualiser le code java généré par le logiciel.

Effectuer un test avec le bouton *run*.

Pour l'instant, si vous cliquez sur le bouton, rien ne se passe, car on n'a pour l'instant pas fait d'actions sur le bouton.

Mettons en place notre première action, vous allez suivre les instructions suivantes :

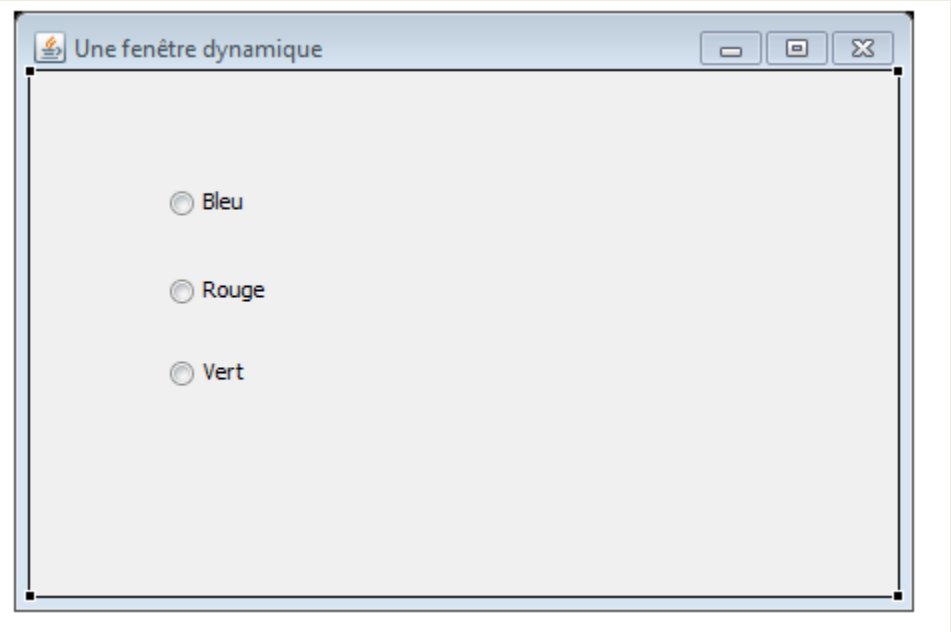
cliquez droit sur le bouton - *add event handler - action - action performed*.

Vous vous retrouvez dans le code source à l'endroit souhaité pour qu'une action se produise lorsqu'on sélectionne le bouton. Nous allons ajouter alors la ligne de code qui modifie la couleur du fond :

```
JRadioButton rdbtnBleu = new JRadioButton("Bleu");
rdbtnBleu.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        contentPane.setBackground(Color.BLUE);
    }
});
rdbtnBleu.setBounds(66, 54, 109, 23);
contentPane.add(rdbtnBleu);
```

Après cette modification, testez votre fenêtre.

Créez de même deux autres boutons radios Rouge et Vert.



Nous allons maintenant mettre en relation les trois boutons que nous venons de créer, pour qu'à chaque fois qu'un des boutons est sélectionné, les autres ne le soient plus.

Nous allons donc créer un groupe de boutons appelé *groupeDeBoutons* et ajouter les trois boutons à ce groupe. Voici comment modifier le code java :

```
 */
public Fondecran() {

    ButtonGroup groupeDeBoutons = new ButtonGroup();



    setTitle("Une fen\u00EAtre dynamique");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 450, 300);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(null);

    JRadioButton rdbtnBleu = new JRadioButton("Bleu");
    // Mise en relation du bouton bleu avec les deux autres
    groupeDeBoutons.add(rdbtnBleu);

    rdbtnBleu.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0) {
            contentPane.setBackground(Color.BLUE);
        }
    });
```

De même nous allons créer le bouton *Vert* et le mettre dans le groupe en modifiant le code :

```
JRadioButton rdbtnBleu = new JRadioButton("Bleu");
//mise en relation du bouton bleu avec les deux autres
groupeDeBoutons.add(rdbtnBleu);

rdbtnBleu.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        contentPane.setBackground(Color.BLUE);
    }
});
rdbtnBleu.setBounds(66, 54, 109, 23);
contentPane.add(rdbtnBleu);

JRadioButton rdbtnRouge = new JRadioButton("Rouge");
//mise en relation du bouton rouge avec les deux autres
groupeDeBoutons.add(rdbtnRouge);

rdbtnRouge.setBounds(66, 98, 109, 23);
contentPane.add(rdbtnRouge);

//


JRadioButton rdbtnVert = new JRadioButton("Vert");
//mise en relation du bouton vert avec les deux autres
groupeDeBoutons.add(rdbtnVert);

rdbtnVert.setBounds(66, 139, 109, 23);
contentPane.add(rdbtnVert);
```

De même nous allons faire une action sur chacun des boutons que nous venons de créer, en effectuant un clic droit sur le bouton puis - *add event handler - action - action performed*.

Rajoutons ensuite le code qui change le fond en fonction de la couleur :

```
JRadioButton rdbtnBleu = new JRadioButton("Bleu");
//mise en relation du bouton bleu avec les deux autres
groupeDeBoutons.add(rdbtnBleu);

rdbtnBleu.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        contentPane.setBackground(Color.BLUE);
    }
});
rdbtnBleu.setBounds(66, 54, 109, 23);
contentPane.add(rdbtnBleu);

JRadioButton rdbtnRouge = new JRadioButton("Rouge");
//mise en relation du bouton rouge avec les deux autres
groupeDeBoutons.add(rdbtnRouge);

rdbtnRouge.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        contentPane.setBackground(Color.RED);
    }
});
rdbtnRouge.setBounds(66, 98, 109, 23);
contentPane.add(rdbtnRouge);


JRadioButton rdbtnVert = new JRadioButton("Vert");
//mise en relation du bouton vert avec les deux autres
groupeDeBoutons.add(rdbtnVert);

rdbtnVert.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        contentPane.setBackground(Color.GREEN);
    }
});
rdbtnVert.setBounds(66, 139, 109, 23);
contentPane.add(rdbtnVert);
```

Possibilités d'amélioration

Ceux qui le souhaitent peuvent essayer de colorer tout le fond, texte compris, en modifiant le code java.