



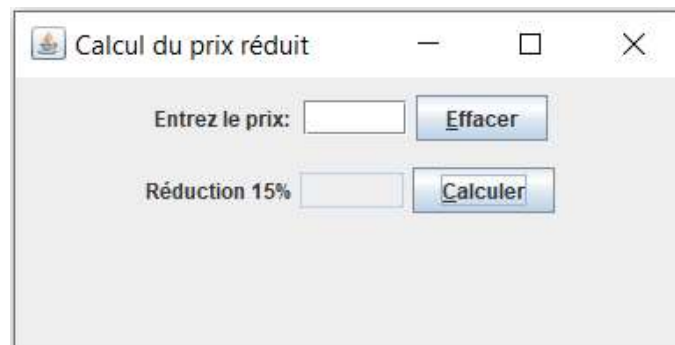
# **LES INTERFACES GRAPHIQUES EN JAVA**

Partiel

# Introduction

- Jusqu'à présent, nous n'avons implémenté que des programmes à **interface console**. Dans cette partie du cours, nous allons développer des programmes simples à interfaces graphiques.
- Dans les programmes à interface console, **c'est le programme qui pilote l'utilisateur** en le sollicitant au moment voulu pour qu'il fournisse des informations **sous forme de texte** dans la console.
- Dans un programme à interfaces graphiques, c'est **l'utilisateur qui a l'impression de piloter le programme**. L'utilisateur exprime des demandes en cliquant sur des boutons, en sélectionnant des options de menu, ou en remplissant des boîtes de dialogue et le programme réagit à cette demande. Ce type de programmation s'appelle « **la programmation événementielle** »,

Exemple : Calcul du prix après une réduction de 15%



# Les bibliothèques

- Pour développer des interfaces graphiques en java, nous nous servons des classes et interfaces du package **java.awt** (Abstract Window Toolkit) et du package **javax.swing**.
- Le package Swing est plus riche que le package AWT, mais **tous les composants de Swing dérivent d'une classe de l'AWT**
- Swing ne remplace pas AWT, mais ne fait que la compléter.

## • AWT

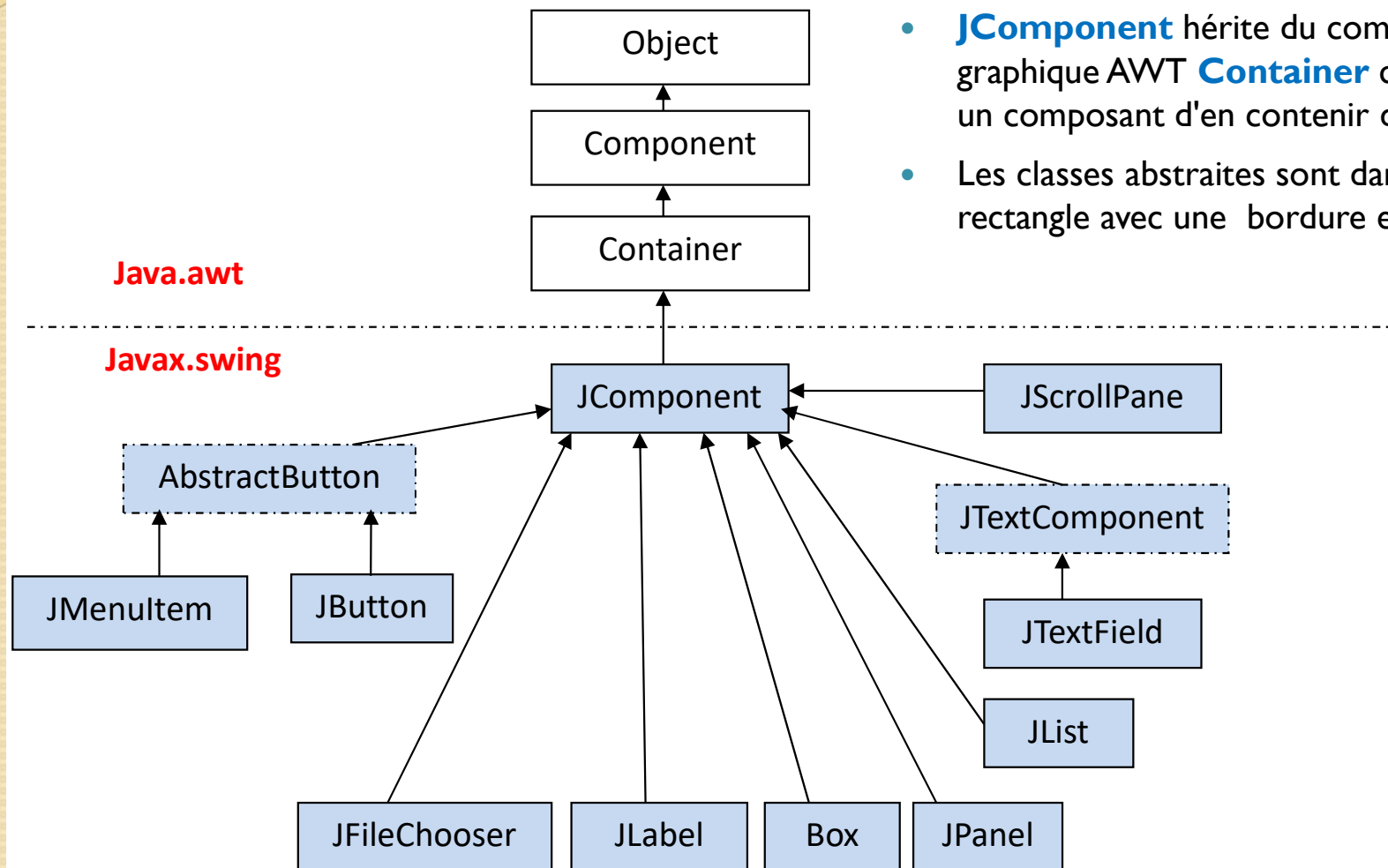
- Elle constitue le package **java.awt** dès le JDK 1.0.
- Les composants sont dessinés et contrôlés par le système d'exploitation. On dit qu'ils sont des **composants lourds**
- Inconvénients:
  - **L'apparence des composants change d'un système à un autre.**
- Avantages:
  - AWT nécessite moins de temps CPU, puisqu'elle utilise directement le système d'exploitation.

## • SWING

- Constitue le package **javax.swing** (à partir du JDK 1.2)
- La plupart des composants Swing sont complètement pris en charge par la machine virtuelle (JVM). On dit qu'ils sont des **composants légers**.
- Avantages:
  - L'exécution est indépendante de la plateforme = portabilité totale pour la plupart des composantes.
- Inconvénients:
  - la bibliothèque Swing nécessite plus de temps CPU, car la machine virtuelle doit dessiner les moindres détails des composants.

# Hiérarchie partielle des composants Swing

- les noms de composants Swing commencent généralement par **J**
- Plusieurs composants héritent du composant graphique **JComponent**
- **JComponent** hérite du composant graphique AWT **Container** qui permet à un composant d'en contenir d'autres
- Les classes abstraites sont dans un rectangle avec une bordure en pointillés.



# Les d'objets graphiques

- Swing dispose de 3 types d'objets graphiques
- Les objets de haut niveau qui permettent de définir une fenêtre.
  - **JFrame**: fenêtre normale
  - **JWindow**: fenêtre non décorée
  - **JDialog**: boîte de dialogue
  - **JApplet**: destiné à être téléchargé et exécuté par un navigateur du poste client

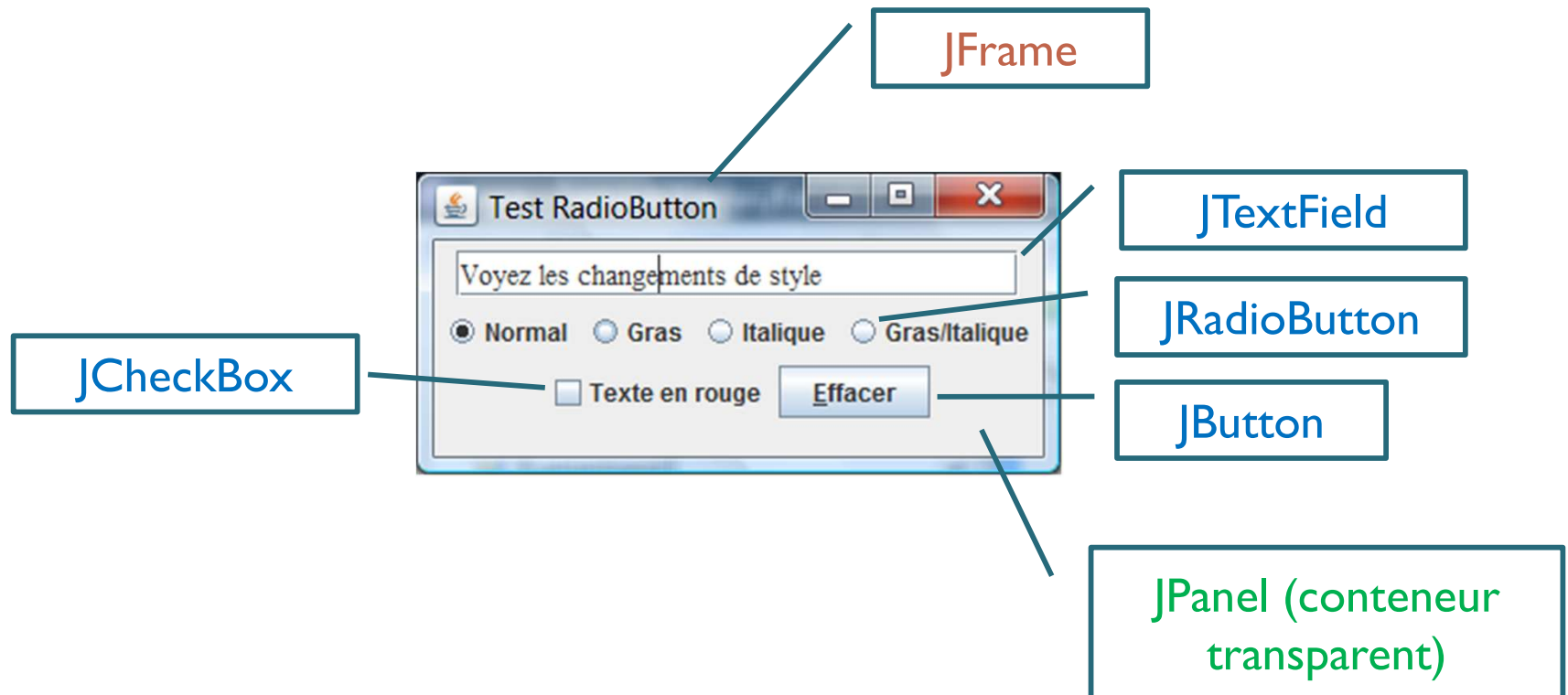
Les fenêtres Swing sont les seuls composants lourds, puisqu'elles sont essentiellement liées au système d'exploitation.

- Les objets de niveau intermédiaire: pour composer la fenêtre
  - **JPanel**: conteneur transparent
  - **JScrollPane**: barre de défilement
  - **JSplitPane**: pour diviser deux Components.
  - ...
- Les objets de niveau inférieur: les composants de base
  - **JButton**,
  - **JCheckBox**,
  - **JLabel**,
  - **JTextField**,
  - **JTextArea**,



# Les d'objets graphiques

- Objet de haut niveau
- Objet de niveau intermédiaire
- Objet de niveau inférieur

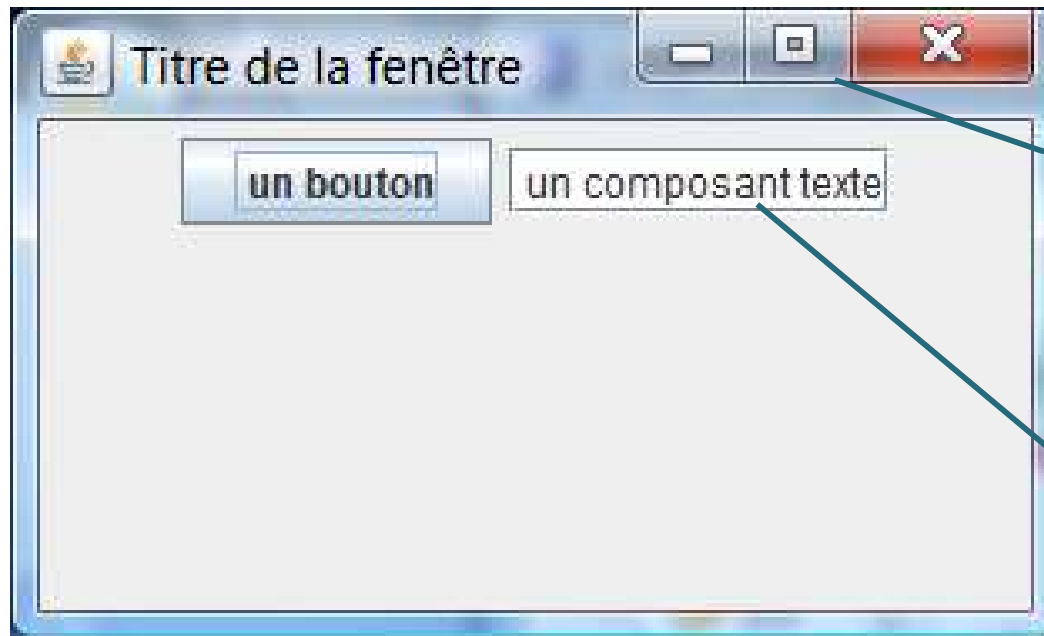






# La fenêtre de type JFrame

# Le composant JFrame



L'affichage de la fenêtre dépend du système d'exploitation: **composant lourd**.

L'affichage du bouton et de la zone de texte est du style de Swing : identique sur tous les systèmes d'exploitation. Ce sont des **composants légers**.

Le composant **JFrame** possède un **titre**, une **taille** modifiable, une **icône**, des **boutons de redimensionnement** et éventuellement une **barre de menu**.



# Création d'une fenêtre

- Pour créer une fenêtre, nous pouvons simplement créer un objet de type **JFrame**. Puis, nous utiliserons les fonctionnalités présentes dans cette classe.

```
import javax.swing.JFrame;

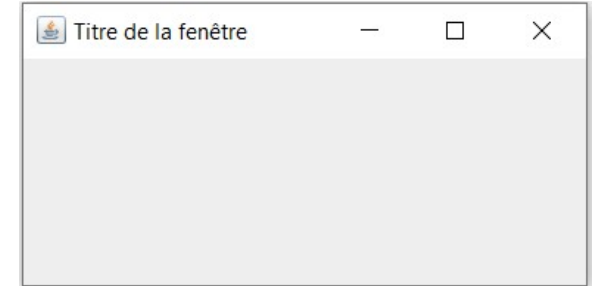
public class Fenetre0 {

    public static void main(String arg[]) {

        //construire la fenêtre en définissant son titre
        JFrame fenetre = new JFrame("Titre de la fen\u00Eatre");
        // préciser sa taille: largeur et hauteur. Par défaut 0,0
        fenetre.setSize(400, 200);
        // préciser le comportement de la fenêtre lors de la fermeture
        fenetre.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        // rendre la fenêtre visible. Par défaut, elle est invisible
        fenetre.setVisible(true);
        // Ne pas modifier la fenêtre après qu'elle soit visible

    }

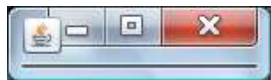
}
```



# Création d'une fenêtre

- Quelques constructeurs de JFrame

Constructeur	Exemple
<code>JFrame()</code>	<code>JFrame fen = new JFrame();</code>
<code>JFrame(String)</code>	<code>JFrame fen = new JFrame("Un titre");</code>

- Pour afficher la fenêtre à l'écran, on doit **spécifier sa taille** avec la méthode ***setSize(int largeur, int hauteur)***.
  - Par défaut largeur = hauteur = 0. 
  - Exemple: pour une fenêtre de 400 pixels de largeur et 200 pixels de hauteur, on écrit:  
`fenetre.setSize(400, 200);`
- Il faut **demande l'affichage** de la fenêtre en appelant la méthode ***setVisible(boolean b)***.
  - Par défaut une fenêtre est invisible.
  - Exemple: `fenetre.setVisible(true);`

# Création d'une fenêtre

- On doit fermer la fenêtre. On utilise la méthode:  
*`setDefaultCloseOperation(int operation)`*.
  - Par défaut, la fenêtre est simplement cachée quand elle est fermée (si on clique sur le bouton X).

- Exemples:

*//fermer l'application*

```
fenetre.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

*//rend le bouton X inactif*

```
fenetre.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
```

# Création d'une classe fenêtre personnalisée

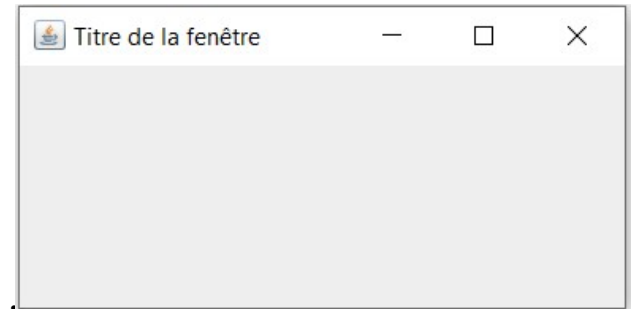
- Pour construire une fenêtre, on crée une classe dérivée de **JFrame**, puis on crée un objet de ce nouveau type.

```
import javax.swing.JFrame;

public class Fenetre1 extends JFrame {

    // Constructeur
    public Fenetre1() {
        super("Titre de la fen\u00EAtre ");
        setSize(400, 100);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

public class Application1 {
    public static void main (String arg[]){
        Fenetre1 fenetre = new Fenetre1();
        fenetre.setVisible(true);
    }
}
```

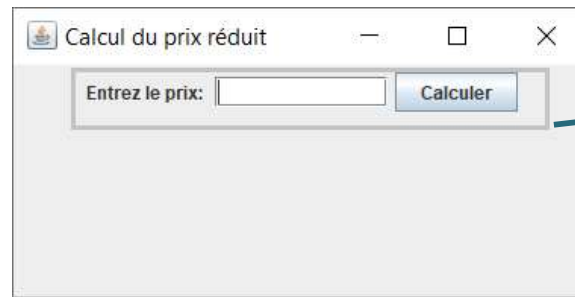


- On peut écrire autrement le code de Fenetre1. On peut utiliser le constructeur sans paramètre et la méthode **setTitle()** pour attribuer le titre.

```
public Fenetre1() {
    super();
    setTitle("Titre de la fen\u00EAtre");
    setSize(400, 100);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
```

# Ajouter des composants à une fenêtre

- Les étapes à suivre pour ajouter des composants à une fenêtre sont:
  1. Créer des composants intermédiaires, exemple un `JPanel`
  2. Créer des composants de bases, exemples  `JButton`,  `JTextField`, ...
  3. Insérer les composants de bases dans les composants intermédiaires
  4. insérer les composants intermédiaires dans la fenêtre.
- Exemple:
  1. Créer un panneau
  2. Créer le bouton(calculer), l'étiquette (entrez le prix) et la zone de texte (vide)
  3. Ajoutez le bouton, l'étiquette (entrez le prix) et la zone de texte (vide) au panneau
  4. Ajoutez le panneau à la fenêtre.



Panneau transparent

# Ajouter des composants à une fenêtre

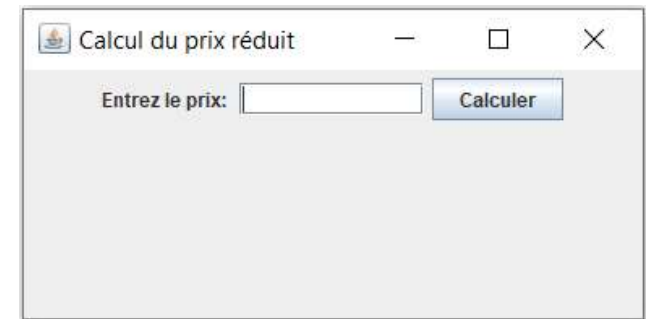
- Exemple: package application I

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class FrmPrixReduit extends JFrame {
    // créer les composants de base
    private JButton btnCalculer = new JButton("Calculer");
    private JTextField txtPrix = new JTextField(10); // largeur 10
    private JLabel lblPrix = new JLabel("Entrez le prix: ");
    // créer le composant intermédiaire: le panneau
    private JPanel panell = new JPanel(); // créer un panneau: conteneur vide transparent

    public FrmPrixReduit () {
        super("Calcul du prix r\u00E9duit");
        setSize(400, 200);
        //ajouter les composants au panneau
        panell.add(lblPrix);
        panell.add(txtPrix);
        panell.add(btnCalculer);
        //ajouter le panneau à la fenêtre
        add(panell);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

public class ApplicationPrixReduit { //la classe principale de l'application
    public static void main (String arg[]){
        FrmPrixReduit fenetre= new FrmPrixReduit ();
        fenetre.setVisible(true);
    }
}
```





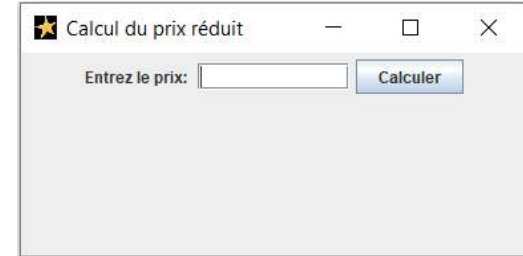
# La personnalisation de l'icône de la fenêtre

- Exemple: package application2
- La méthode `setIconImage(Image image)` permet de modifier l'icône de la JFrame.
- Si l'image est trop grande, elle est redimensionnée.
- Exemple:

```
public class FrmPrixReduit extends JFrame {
    // créer les composants de base
    private JButton btnCalculer = new JButton("Calculer");
    private JTextField txtPrix = new JTextField(10); // largeur 10
    private JLabel lblPrix = new JLabel("Entrez le prix: ");
    // créer le composant intermédiaire: le panneau
    private JPanel panell = new JPanel(); // créer un panneau: conteneur

    public FrmPrixReduit () {
        super("Calcul du prix réduit");
        setSize(400, 200);
        ImageIcon iconeFenetre = new ImageIcon(FrmPrixReduit.class.getResource("/images/etoile.jpg"));
        // chemin relatif de l'image
        setIconImage(iconeFenetre.getImage());

        //ajouter les composants au panneau
        panell.add(lblPrix);
        panell.add(txtPrix);
        panell.add(btnCalculer);
        //ajouter le panneau à la fenêtre
        add(panell);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```





# les composants de base

# La classe JLabel (étiquette): constructeurs

- Un JLabel est un texte **non éditable**. Il peut être accompagné d'une icône.
- Quelques constructeurs

Constructeurs	Rôle
JLabel()	Création d'une instance sans texte ni image.
JLabel(Icon image)	Création d'une instance en précisant l'image.
JLabel(String text)	Création d'une instance en précisant le texte

- Exemples de construction de composants JLabel.

```
JLabel label1 = new JLabel("label1 ");  
ImageIcon icone = new ImageIcon(Etiquettes.class.getResource("images/etoile.jpg"));  
JLabel label2 = new JLabel(icone);
```



# La classe JLabel (étiquette): quelques méthodes

Méthode	Rôle
setText()	Permet d'initialiser ou de modifier le texte affiché
setOpaque(boolean)	Indique si le composant est <b>transparent</b> (paramètre false) ou opaque (true)
setBackground()	Indique la <b>couleur de fond</b> du composant (setOpaque doit être à true)
setFont()	Permet de préciser la <b>police</b> du texte
setForeground()	Permet de préciser la <b>couleur du texte</b>
setHorizontalAlignment()	Permet de modifier l' <b>alignement horizontal</b> du texte et de l'icône
setIcon()	Permet d'assigner une <b>icône</b>

Exemples: modifier les caractéristiques d'un objet JLabel après sa construction.

- Lors de la construction

```
JLabel label1 = new JLabel("Ancienne étiquette");
```

Ancienne étiquette

- Modification du texte et de la couleur de texte

```
label1.setText("Nouvelle étiquette");  
label1.setForeground(Color.RED);  
label1.setOpaque(true);  
label1.setBackground(Color.CYAN);
```

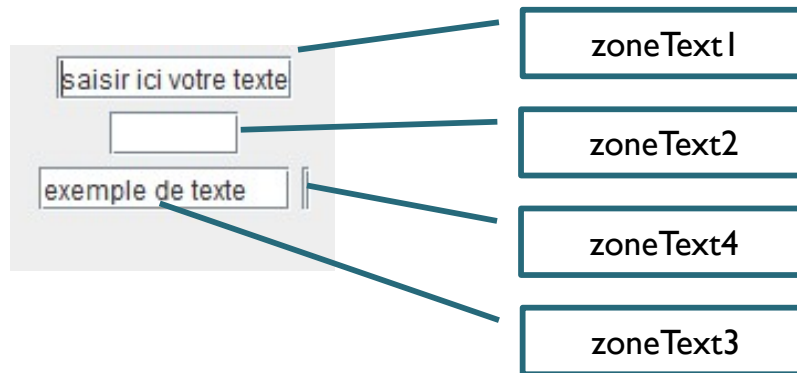
Nouvelle étiquette

# La classe JTextField

- Permet la saisie d'une seule ligne de texte.

- Quelques exemples de construction

```
JTextField zoneText1 = new JTextField("saisir ici votre texte");  
JTextField zoneText2 = new JTextField(5); //longueur 5  
JTextField zoneText3 = new JTextField("exemple de texte",10);  
JTextField zoneText4 = new JTextField(); //longueur 0
```



- Pour modifier le texte

```
setText(String texte);
```

- Exemple:

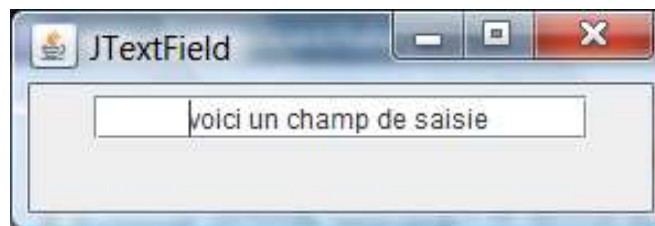
```
zoneText.setText("Nouveau texte");
```

# La classe JTextField

- **Une zone de texte est par défaut éditable** (on peut écrire à l'intérieur). Pour rendre une zone de texte non éditable, on utilise `setEditable(false)`
- La méthode `setHorizontalAlignment(int alignment)` permet de préciser l'alignement du texte dans le composant en utilisant les constantes:
  - `JTextField.LEFT` ,
  - `JTextField.CENTER`
  - `JTextField.RIGHT`.

- Exemple

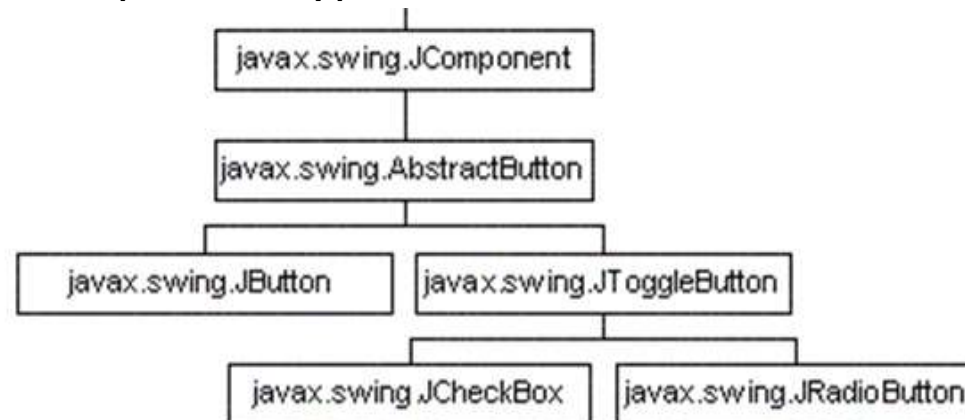
```
JTextField message = new JTextField(20);  
message.setHorizontalAlignment(JTextField.CENTER);  
String phrase = new String("voici un champ de saisie");  
message.setText(phrase);
```





# Les boutons

- Il existe plusieurs types de boutons définis dans Swing



• **JButton**: bouton classique  
• **JToggleButton**: bouton qui reste enfoncé:

- Toutes les classes de boutons héritent de la classe abstraite `AbstractButton`.** Cette classe définit de nombreuses méthodes communes à tous les boutons. Voir la diapositive suivante.
- Quelques méthodes de la classe **`AbstractButton`**

Méthode	Rôle
<code>doClick()</code>	Déclencher un clic par programmation
<code>getText()</code>	Obtenir le texte affiché par le composant
<code>setText()</code>	Mettre à jour le texte du composant
<code>isSelected()</code>	Indiquer si le composant est dans l'état sélectionné
<code>setSelected()</code>	Mettre à jour l'état sélectionné du composant

# La classe JButton

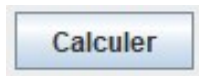
- Un bouton peut contenir un texte et/ou une icône.
- Quelques constructeurs

Constructeur	Rôle
<code>JButton()</code>	Construire un bouton sans texte ni icône
<code>JButton(String text)</code>	Préciser le texte du bouton
<code>JButton(Icon image)</code>	Préciser une icône
<code>JButton(String text, Icon image)</code>	Préciser un texte et une icône

- Toutes les méthodes concernant le contenu du composant `JLabel` sont valables pour le composant `JButton`.

- Exemple:

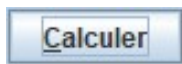
```
JButton btnCalculer = new JButton("Calculer");
```



- Si on veut ajouter un raccourci clavier sur la lettre 'C' de sorte à actionner le bouton avec la combinaison des touches ALT + 'C', on utilise la méthode `setMnemonic(int mnemonic)`.

- Exemple: pour un raccourci avec la lettre 'C'.

```
btnCalculer.setMnemonic(KeyEvent.VK_C);
```



# La classe JCheckBox



- Quelques constructeurs

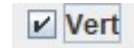
Constructeur	Rôle
<code>JCheckBox()</code>	créer une case à cocher vierge
<code>JCheckBox(String text)</code>	précise l'intitulé
<code>JCheckBox(String text, boolean selected)</code>	précise l'intitulé et l'état (sélectionné ou non)

- Exemples

```
private JCheckBox checkRouge = new JCheckBox("Rouge");
```

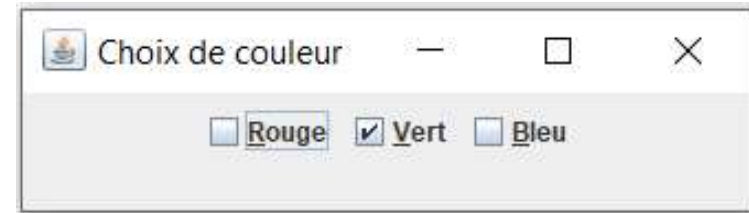


```
private JCheckBox checkVert = new JCheckBox("Vert", true);
```



# La classe JCheckBox

- Exemple: FrmCaseACocher.java



```
public class FrmCaseACocher extends JFrame {
    private JCheckBox checkRouge = new JCheckBox("Rouge");
    private JCheckBox checkVert = new JCheckBox("Vert", true);
    private JCheckBox checkBleu = new JCheckBox("Bleu");

    public FrmCaseACocher() {
        super();
        setTitle("Choix de couleur");
        setSize(350, 100);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPanel panneau = new JPanel();

        // Les mnémonique
        checkRouge.setMnemonic(KeyEvent.VK_R);
        checkVert.setMnemonic(KeyEvent.VK_V);
        checkBleu.setMnemonic(KeyEvent.VK_B);

        panneau.add(checkRouge);
        panneau.add(checkVert);
        panneau.add(checkBleu);
        add(panneau);
    }

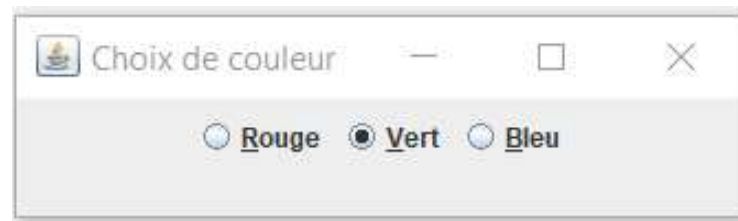
    public static void main(String args[]) {
        FrmCaseACocher fenetre = new FrmCaseACocher();
        fenetre.setVisible(true);
    }
}
```

# La classe JCheckBox

- Quelques méthodes intéressantes héritées de **AbstractButton** :
- `public void setText(String text)`
  - Pour spécifier le texte de la case à cocher.
- `public void setIcon (Icon icon)`
  - Pour spécifier l'image de la case à cocher.
- `public void setSelected (boolean b)`
  - Pour spécifier l'état de la case à cocher.
- `public boolean isSelected ()` :
  - Retourne l'état de la case à cocher.
- `public void setMnemonic(int mnemonic)`
  - Associe un mnémonique (combinaison alt + touche)

# La classe JRadioButton

- Les paramètres des constructeurs sont similaires à ceux de la classe **JCheckBox**.



- Exemple:**

```
private JRadioButton radioRouge = new JRadioButton("Rouge");
```



```
private JRadioButton radioVert = new JRadioButton("Vert", true);
```





# Les groupes de boutons

- La classe **ButtonGroup** nous permet de créer un groupe de boutons **pour n'avoir qu'un bouton sélectionné à la fois.**
- **ButtonGroup** peut être utilisée avec
  - `JToggleButton` (ne seront pas vus dans ce cours)
  - `JCheckBox`
  - `JRadioButton`
- On ajoute un bouton au groupe avec la méthode **`add()`**
- **ButtonGroup** est un conteneur logique **et ne doit pas être ajouté** à la fenêtre.
- **On ne peut pas** désélectionner un bouton se trouvant dans un **ButtonGroup** en cliquant dessus (voir l'exemple de la diapositive suivante).

# Les groupes de boutons

- Exemple: FrmRadio.java

```
public class FrmRadio extends JFrame {
    private JRadioButton radioRouge = new JRadioButton("Rouge");
    private JRadioButton radioVert = new JRadioButton("Vert");
    private JRadioButton radioBleu = new JRadioButton("Bleu");
    private ButtonGroup groupe = new ButtonGroup();
    private JPanel panneau = new JPanel();

    public FrmRadio() {
        super();
        setTitle("Choix de couleur");
        setSize(300, 100);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Les mnémonique
        radioRouge.setMnemonic(KeyEvent.VK_R);
        radioVert.setMnemonic(KeyEvent.VK_V);
        radioBleu.setMnemonic(KeyEvent.VK_B);

        // Selectionner le bouton rougeRadio
        radioRouge.setSelected(true);
        // ajouter les bouton au groupe
        groupe.add(radioRouge);
        groupe.add(radioVert);
        groupe.add(radioBleu);

        //ajouter les boutons au panneau
        panneau.add(radioRouge);
        panneau.add(radioVert);
        panneau.add(radioBleu);

        // ajouter le panneau à la fenêtre
        add(panneau);
    }
}
```

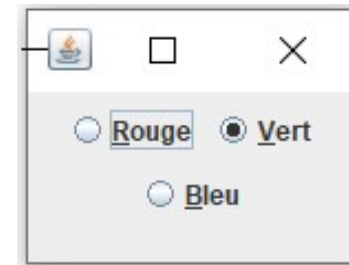




# Les gestionnaires de disposition (layout manager)

# La disposition des composants dans la fenêtre

- Les fenêtres utilisent un "layout manager" (ou gestionnaire de mise en forme) pour positionner leurs composantes.
- Il existe plusieurs layouts:
  - BorderLayout: place aux 4 coins cardinaux
  - FlowLayout: place les composants un à la suite de l'autre comme dans un panneau
  - GridLayout: place dans une grille
  - GridBagLayout: placements complexes
  - BoxLayout: place les composants horizontalement ou verticalement
  - ...
- **Le gestionnaire par défaut d'un composant JPanel est FlowLayout:** les composants sont placés un à la suite de l'autre. S'il n'y a pas d'espace, le composant se place à la ligne suivante.



- **Le gestionnaire utilisé par défaut par un JFrame est le BorderLayout** (voir la diapositive suivante).

# La disposition des composants dans la fenêtre

- Le gestionnaire utilisé par défaut par un `JFrame` est le `BorderLayout`.
- Lorsqu'on fait appel à la méthode `add(composant)` pour ajouter un composant à la fenêtre, celui-ci est **ajouté au centre**.
- On peut utiliser la fonction `setLayout()` pour changer le layout utilisé. Exemple  
`setLayout(new FlowLayout());`

- Exemple: `FrmLayout.java`

```
public class FrmLayout extends JFrame {  
    JButton button1 = new JButton("North");  
    JButton button2 = new JButton("South");  
    JButton button3 = new JButton("East");  
    JButton button4 = new JButton("West");  
    JButton button5 = new JButton("Center");  
  
    public FrmLayout() {  
        super("Layout");  
        add(button1, BorderLayout.NORTH);  
        add(button2, BorderLayout.SOUTH);  
        add(button3, BorderLayout.EAST);  
        add(button4, BorderLayout.WEST);  
        add(button5, BorderLayout.CENTER);  
        setSize(256, 256);  
        setVisible(true);  
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    }  
}
```

