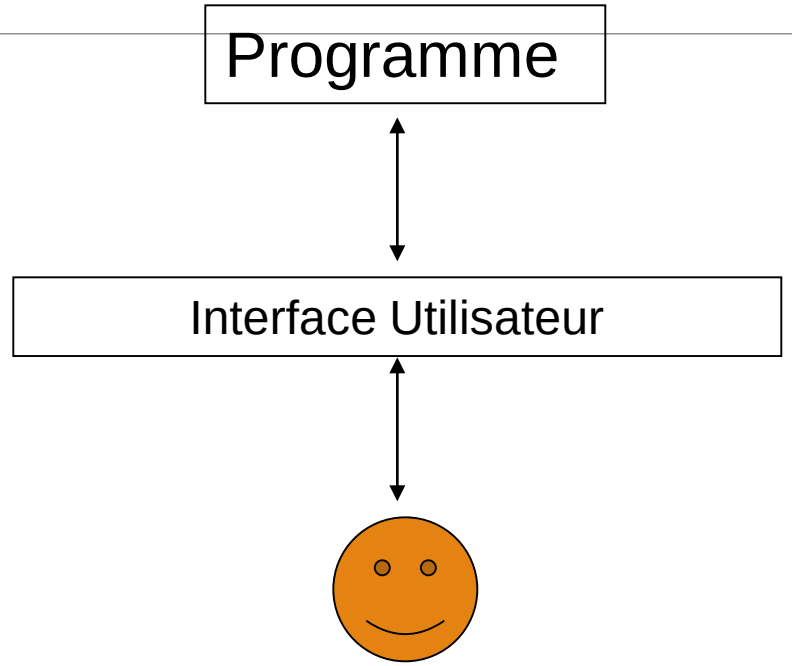
IHM

Concepts

- Construire une fenêtre graphique
  - Objets graphiques
  - Composition, affichage
- Programmation par événement
  - Comment faire pour que le programme réagisse?
  - Principe MVC – Modèle-Vue-Contrôle
- Package Swing :

Généralité



Rôles d'une interface utilisateur:

- montrer le résultat de l'exécution
- permettre à l'utilisateur d'interagir
- (1) Montrer – (2) Réagir

Généralité

- Le langage JAVA comparé aux autres langages tels que C ou pascal, permet une grande abstraction moyennant des types appelés classes.
- Ces classes permettent de modéliser les concepts rencontrés dans la vie quotidienne (Personne, Vehicule, …).
- La JDK contient des classes dédiées à la création d'interfaces graphiques. Ces dernières sont organisée dans des packages spéciaux et se conforment à des motifs de conception évolués.

Les APIs graphiques de JAVA

- Deux apis pour la programmation d'interfaces graphiques: AWT (Abstract Windowing Toolkit) et Swing.
- L'API AWT a été introduit en java dès la version 1.0.
- Swing a été introduit avec la version 1.1 de la jdk comme faisant partie de la Java Foundation Classes (JFC).

Packages Java pour l'IHM

- java.awt : abstract window toolkit (java 1.1) composants lourds (primitifs).
  - ✓ Obsolète.
  - ✓ L'apparence de l'application change selon le Système d'exploiatation.
- Support d'Oracle.
- javax.swing : composants légers écrits en java (java 1.2).
  - ✓ L'apparence de l'application est identique quelque soit Système d'exploiatation.
- SWT(The Standard Widget Toolkit) libre, initié par IBM/Eclipse.

Api Swing

- Pas de code natif → rendu visuel indépendant de la plateforme. Look and feel adhoc.
- Gestion des événement par AWT.
- Les noms des composants sont préfixés par J*.

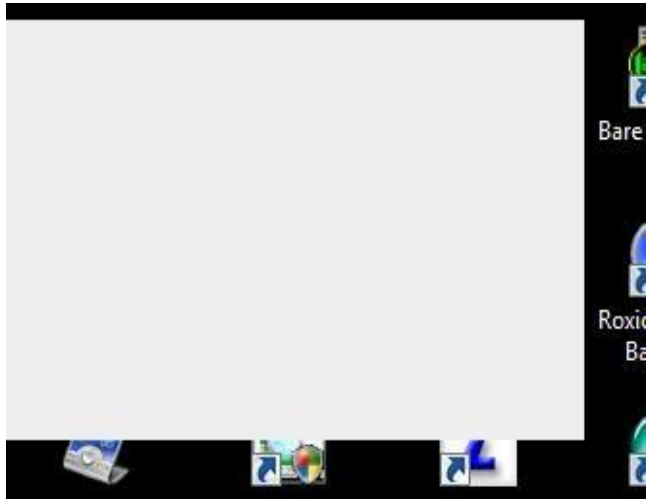
Création d'une application Swing

1. Définir les composants graphiques.
2. Placer les composants dans un conteneur.
3. Spécifier le gestionnaire de positionnement.
4. Définir les actions associées aux évènement (Listener).
5. Associer Ces derniers aux composants.

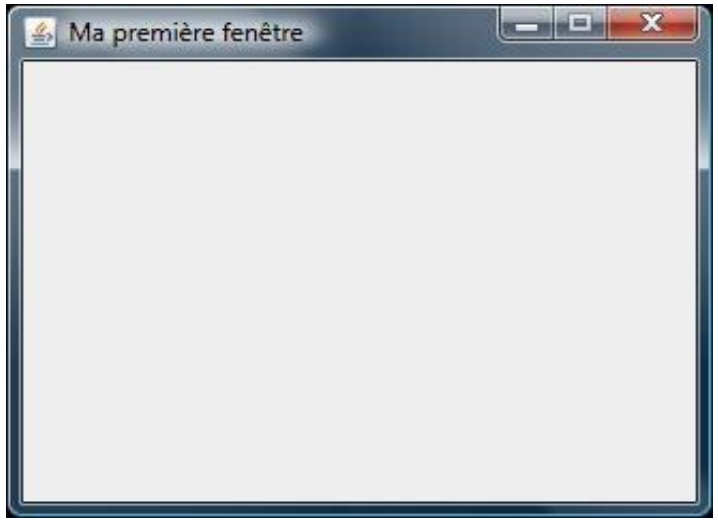
Création d'une application Swing (Suite)

- Une IHM Swing est un arbre ayant pour racine un conteneur choisit parmi :
  - ✓ JFrame : fenêtre système décoré.
  - ✓ JWindow : fenêtre système non décorée.
  - ✓ JDialog : boite de dialogue.
  - ✓ JApplet : zone d'affichage au sein d'un navigateur web.

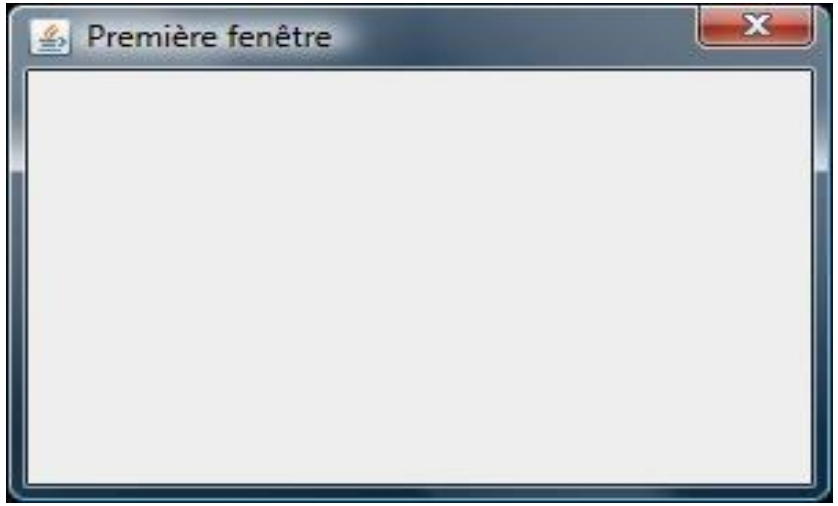
JWindow



JFrame

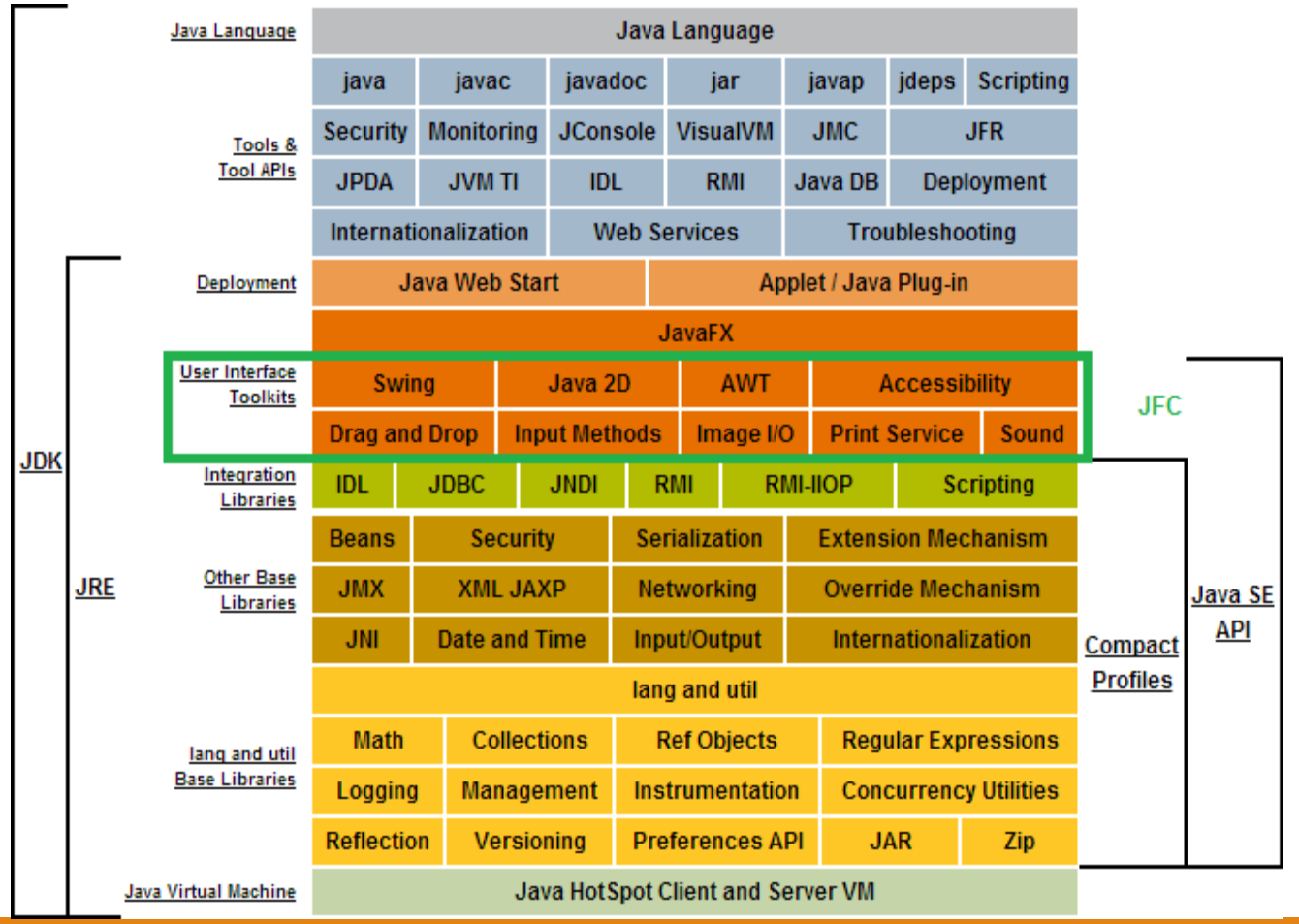


JDialog

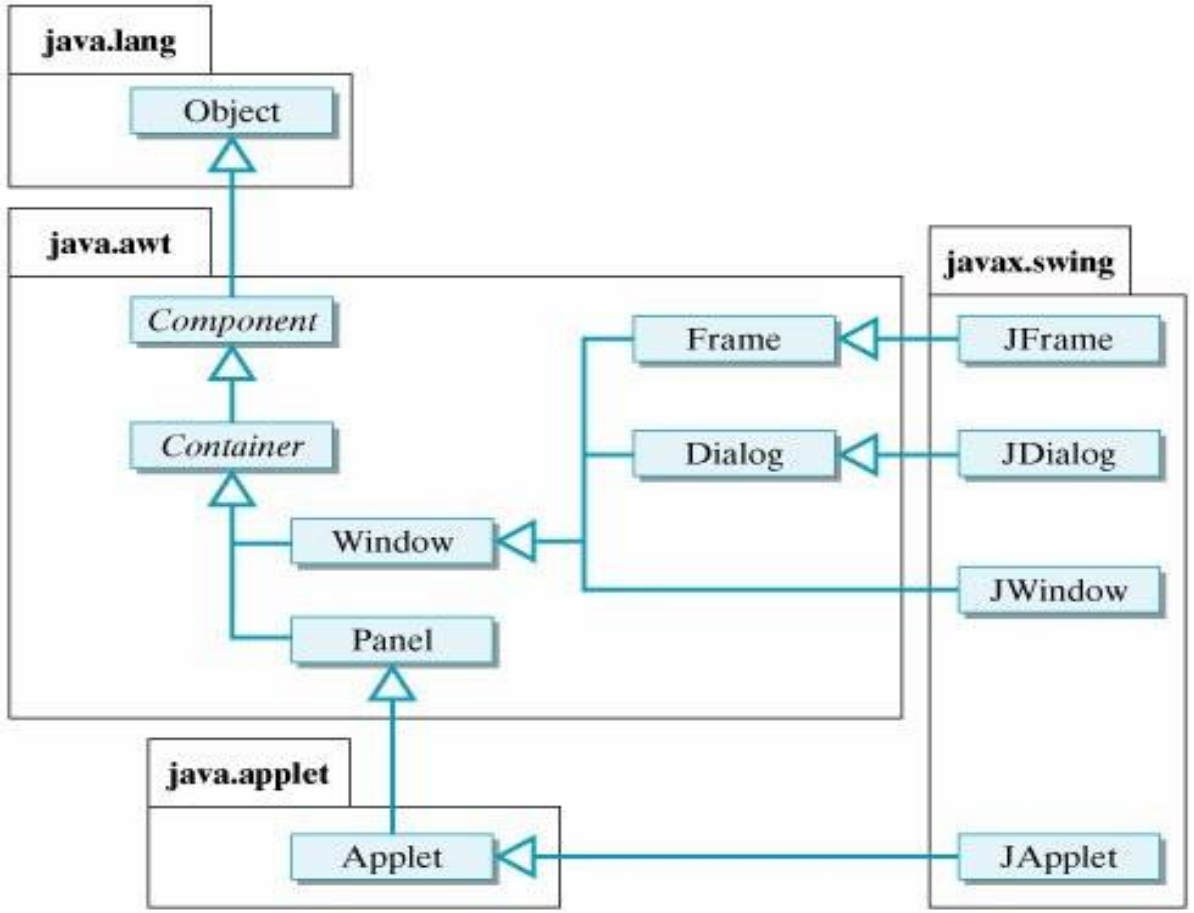


Architecture du framework Swing

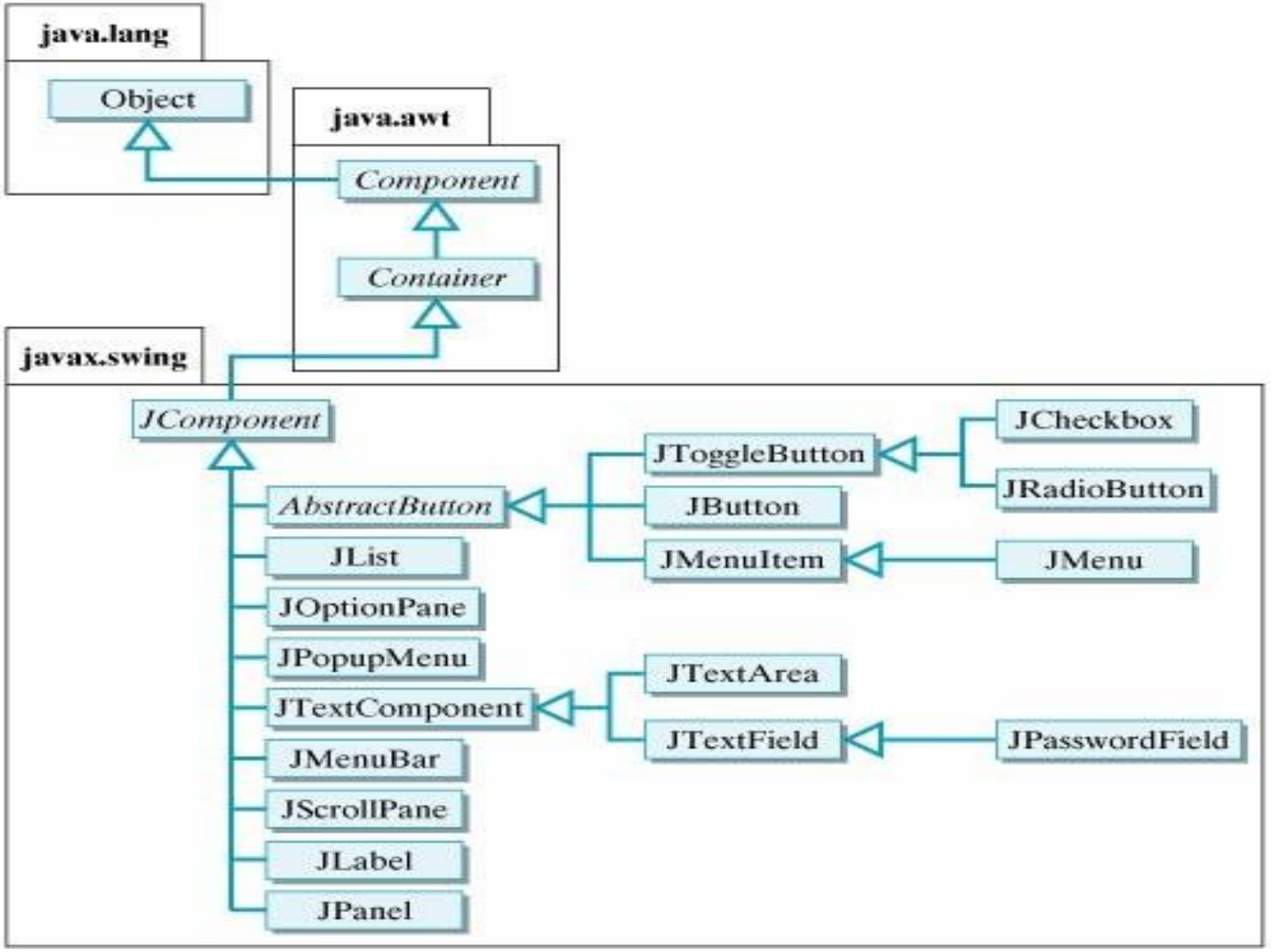
Packages Java pour l'IHM



Arbre AWT


java.lang
Object
java.awt
Component
Container
Window
Panel
Frame
Dialog
javax.swing
JFrame
JDialog
JWindow
java.applet
Applet
JApplet

Arbre Swing


java.lang
Object
java.awt
Component
Container
javax.swing
JComponent
AbstractButton
JToggleButton
JCheckbox
JRadioButton
JButton
JMenuItem
JMenu
JList
JOptionPane
JPopupMenu
JTextComponent
JTextArea
JTextField
JPasswordField
JMenuBar
JScrollPane
JLabel
JPanel

Classification

- Conteneurs Haut-niveau : Composants à la racine de toute arborescence Swing (JFrame, JDialog et JWindow).
- Conteneurs génériques : Conteneur intérmédiares utilisables sous différents circonstances (JPanel, Box).
- Conteneurs à usage spécifique : Conteneur intérmédiares qui jouent un rôle spécifique dans l'IHM (JTabbedPane, JScrollPane, JSplitPane).
- Contrôles basiques : Composants atomiques permettant d'intéragir avec l'utilisateur (Boutons, Cases à cocher, . . .).
- Afficheurs non-éditables : Composants atomiques permettant d'afficher de l'information à l'intention de l'utilisateur.
- Afficheurs éditables avec formattage : Composants atomiques permettant d'afficher de l'information formatté potentiellemnt éditable par l'utilisateur.

Composant Swing : Points commun (Tailles)

- Trois tailles pour un composant : minimum, maximum, preferred
- Méthodes disponibles :
  - public Dimension getMinimumSize();
  - public Dimension getMaximumSize();
  - public Dimension getPreferredSize() ;
  - public void setMinimumSize(Dimension minimumSize);
  - public void setMaximumSize(Dimension maximumSize);
  - public void setPreferredSize(Dimension preferredSize) ;

Composant Swing : Points commun (ToolTip)

- A tout composant on peut associer une info bulle.
  - ✓ public void setToolTipText (String txt) ;
- Raccourcis clavier
  - ✓ public void registerKeyboardAction(ActionListener anAction, String aCommand, KeyStroke aKeyStroke, int aCondition) ;

Composant Swing : Points commun (Bordures)

- Bordure visible du composant
- Intérêt : structuration visuelle de l'IHM (regrouper les widgets qui vont ensemble)
- Usage : création d'une bordure (class Border) et affection de cette bordure (setBorder) à un composant
- Bordures prédéfinies (EmptyBorder blanc, LineBorder : ligne coloré, BevelBorder : Bordure 3D)

Swing et le MVC

Modèle Vue Contrôleur (MVC) est un design pattern :

- Le Modèle qui gère et stocke les données (abstraites).
- Des Vues qui implantent un rendu visuel à partir du modèle.
- Le Contrôleur gère les interactions avec l'utilisateur et modifie le modèle via vues.

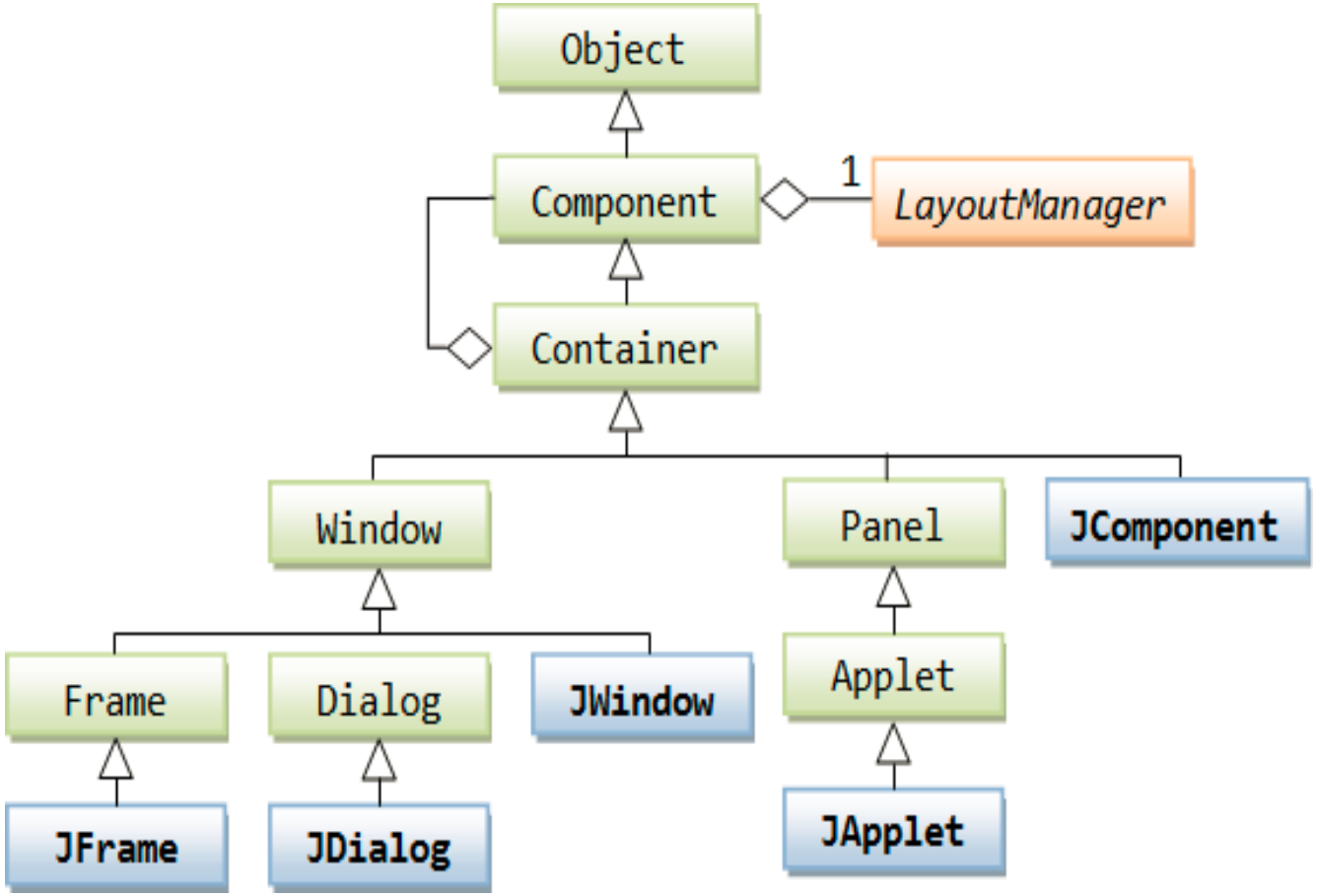
Les composants Swing respectent une version compacte du modèle MVC.

Swing et le MVC (Suite)

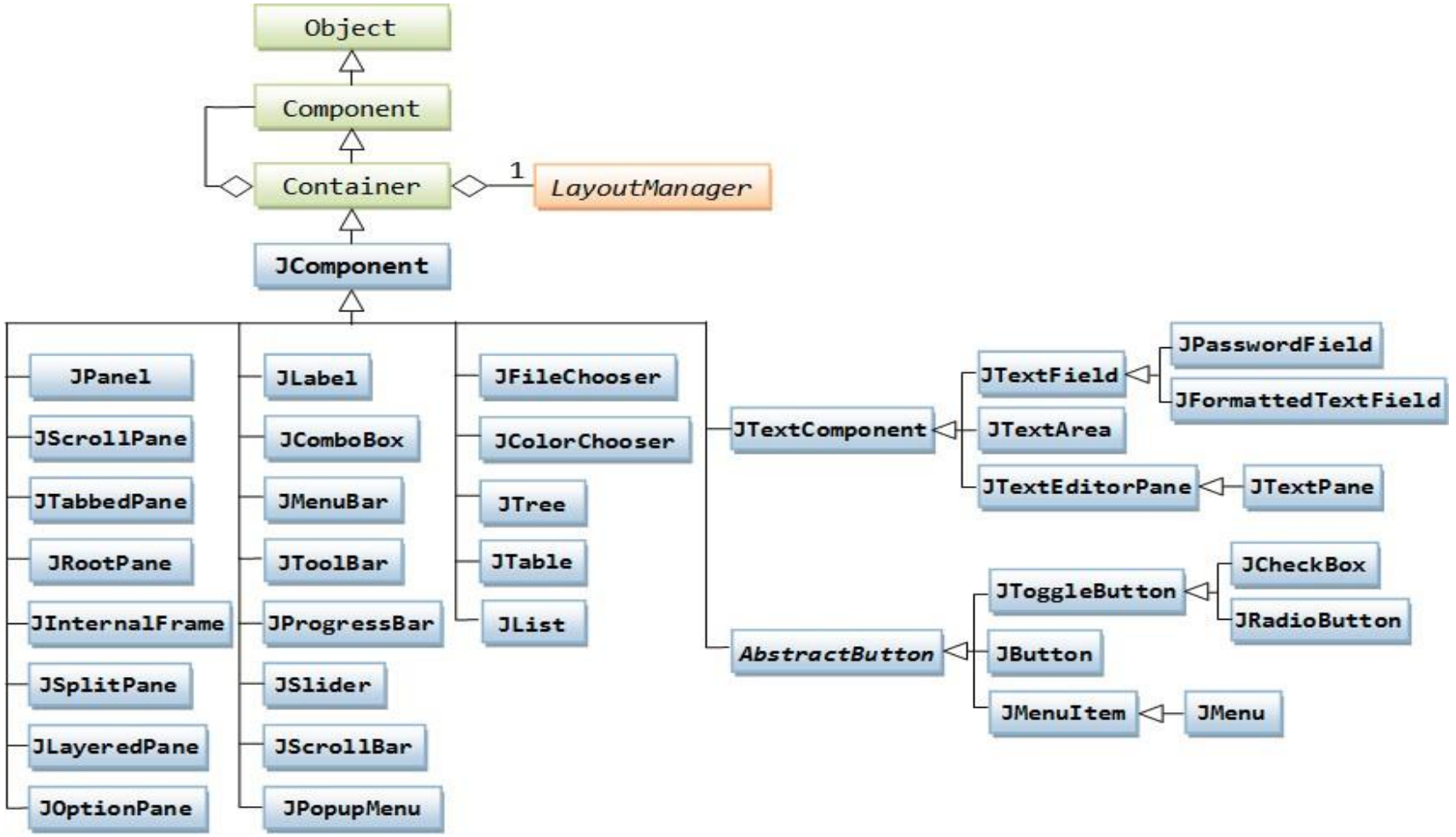
- Le modèle MVC de Swing associe la vue et le contrôleur dans une entité appelé **Délégué UI**.
- Le délégué UI gère l'apparence visuelle du composant et ses interactions avec l'utilisateur.
- Un délégué UI différent produit une apparence différente.
- Le modèle correspond aux types métiers
- Exemple de modèles :
  - ✓ ButtonModel : JButton, JMenu, JMenuItem, . . .
  - ✓ Document : JTextPane, JTextArea, JPasswordField, . . .
- Un modèle peut être associé à plusieurs vues.
- Un composant peut avoir plusieurs types de modèles disponibles.

Les composants Swing

Les Frames, Dialogues



Les composants Swing



Les boutons

Instances de JButton avec différentes possibilités de création :

```
1       JButton() //Creates a button with no set text or icon.
2       JButton(Icon icon) //Creates a button with an icon.
3       JButton(String text) //Creates a button with text.
4       JButton(String text, Icon icon)
5       //Creates a button with initial text and an icon
6
7       void setText(String)//Set the text displayed by the button
8       String getText()////Get the text displayed by the button
```

Les boutons

Alignement horizontal du texte du bouton avec les méthodes Héritées de AbstractButton

```
1    public void setHorizontalAlignment(int alignment)
2    public int getHorizontalAlignment()
```

Les constantes d'alignement prédéfinies sont :

```
1    AbstractButton.RIGHT
2    AbstractButton.LEFT
3    AbstractButton.CENTER
4    AbstractButton.LEADING//Texte a gauche
5    AbstractButton.TRAILING//Texte a droite
```

Les boutons

Positionnement vertical du texte du bouton avec les méthodes Héritées de AbstractButton

```
1    public void setVerticalTextPosition(int textPosition)
2    public int getVerticalTextPosition()
```

Les constantes de positionnement prédéfinies sont :

```
1    AbstractButton.CENTER //valeur par defaut
2    AbstractButton.TOP
3    AbstractButton.BOTTOM
```

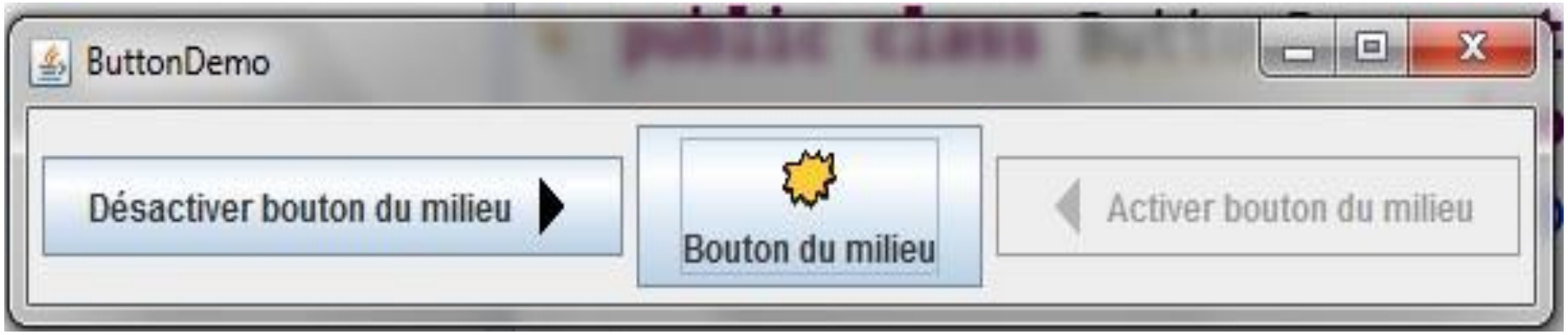
Les boutons : Exemple

```
1     ....
2     protected JButton b1, b2, b3;
3     public Demo() {
4         ImageIcon leftButtonIcon = createImageIcon("images/right.gif");
5         ImageIcon middleButtonIcon = createImageIcon("images/middle.gif");
6         ImageIcon rightButtonIcon = createImageIcon("images/left.gif");
7
8         b1 = new JButton("Desactiver bouton du milieu", leftButtonIcon);
9         b1.setVerticalTextPosition(AbstractButton.CENTER);
10        b1.setHorizontalTextPosition(AbstractButton.LEADING);
11        b1.setActionCommand("disable");
12        b2 = new JButton("Bouton du milieu", middleButtonIcon);
13        b2.setVerticalTextPosition(AbstractButton.BOTTOM);
14        b2.setHorizontalTextPosition(AbstractButton.CENTER);
15        b3 = new JButton("Activer bouton du milieu", rightButtonIcon);
16        b3.setActionCommand("enable");
17
18        b3.setEnabled(false);
19        b1.setToolTipText("Click pour desactiver le bouton du milieu.");
20        b2.setToolTipText("Bouton du milieu ne reponds pas au clicks.");
21        b3.setToolTipText("Click pour activer le bouton du milieu.");
22    }
```

Les boutons (suite) : Exemple

```
1   protected ImageIcon createImageIcon(String path,
2                                       String description) {
3       java.net.URL imgURL = getClass().getResource(path);
4       if (imgURL != null) {
5           return new ImageIcon(imgURL, description);
6       } else {
7           System.err.println("Couldn't find file: " + path);
8           return null;
9       }
10  }
```

Les boutons : Résultat



Les cases à cocher

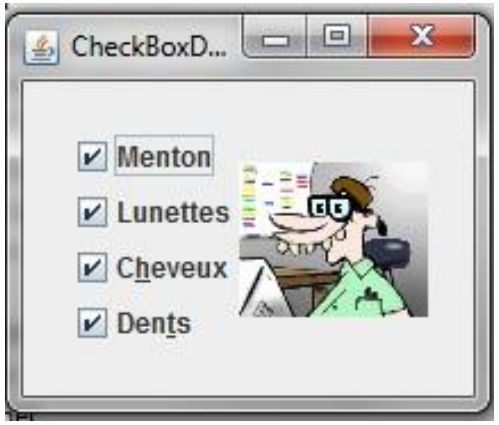
Une case à cocher est une instance de JCheckBox est possède un état.

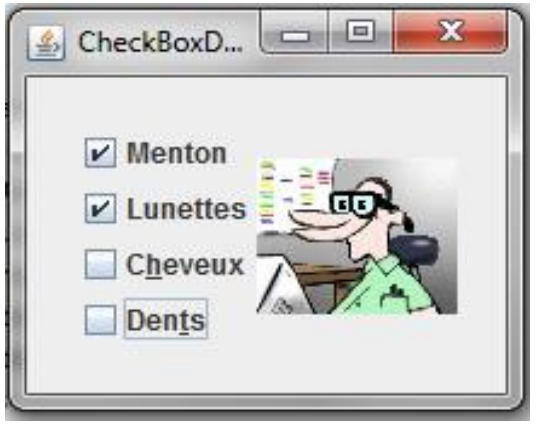
```
1  JCheckBox()
2  //Creates an initially unselected check box button with no text, no icon.
3  JCheckBox(Icon icon)
4  //Creates an initially unselected check box with an icon.
5  JCheckBox(Icon icon, boolean selected)
6  //Creates a check box with an icon and specifies whether or not it is ↙
       ↳ initially selected.
7  JCheckBox(String text)
8  //Creates an initially unselected check box with text.
9  JCheckBox(String text, boolean selected)
10 //Creates a check box with text and specifies whether or not it is ↙
       ↳ initially selected.
11 JCheckBox(String text, Icon icon)
12 //Creates an initially unselected check box with the specified text ↙
       ↳ and icon.
13 JCheckBox(String text, Icon icon, boolean selected)
14 //Creates a check box with text and icon, and specifies whether or not ↙
       ↳ it is initially selected.
```

Les cases à cocher

```
1       public CheckBoxDemo() {
2           chinButton = new JCheckBox("menton");
3           chinButton.setSelected(true);
4           glassesButton = new JCheckBox("Lunettes");
5           glassesButton.setSelected(true);
6           hairButton = new JCheckBox("Cheveux");
7           hairButton.setSelected(true);
8           teethButton = new JCheckBox("Dents");
9           teethButton.setSelected(true);
10      }
```

Les cases à cocher : Exemple





Les boutons radio

Un bouton radio est une instance de JRadioButton regrouper dans un objet de type ButtonGroup.

```
1  JRadioButton()
2  //Creates an initially unselected radio button with no set text.
3  JRadioButton(Icon icon)
4  //Creates an initially unselected radio button with the specified image ↙
       ↳ but no text.
5  JRadioButton(Icon icon, boolean selected)
6  //Creates a radio button with the specified image and selection state, ↙
       ↳ but no text.
7  JRadioButton(String text)
8  //Creates an unselected radio button with the specified text.
9  JRadioButton(String text, boolean selected)
10 //Creates a radio button with the specified text and selection state.
11 JRadioButton(String text, Icon icon)
12 //Creates a radio button that has the specified text and image, and ↙
       ↳ that is initially unselected.
13 JRadioButton(String text, Icon icon, boolean selected)
14 //Creates a radio button that has the specified text, image, and ↙
       ↳ selection state.
```

Les boutons radio

```
1  public RadioButtonDemo() {
2
3          JRadioButton birdButton = new JRadioButton(birdString);
4          birdButton.setActionCommand(birdString);
5          birdButton.setSelected(true);
6          JRadioButton catButton = new JRadioButton(catString);
7          catButton.setActionCommand(catString);
8          JRadioButton dogButton = new JRadioButton(dogString);
9          dogButton.setActionCommand(dogString);
10         JRadioButton rabbitButton = new JRadioButton(rabbitString);
11         rabbitButton.setActionCommand(rabbitString);
12         JRadioButton pigButton = new JRadioButton(pigString);
13         pigButton.setActionCommand(pigString);
14
15         //Grouper les radio buttons.
16         ButtonGroup group = new ButtonGroup();
17         group.add(birdButton);
18         group.add(catButton);
19         group.add(dogButton);
20         group.add(rabbitButton);
21         group.add(pigButton);
22       }
```

Les boutons radio : Exemple