

Cours
Programmation Orientée Objet 2
Pour
ING 2

Chap 04:
Interfaces Graphiques

MEKAHLIA Fatma Zohra LAKRID
Maître de Conférences Classe B

Laboratoire de Modélisation, Vérification et Evaluation des Performances des systèmes complexes (MOVEP)
Bureau 123

PLAN

- Généralités sur les interfaces graphiques.
- Composants des interfaces graphiques.
- Les packages AWT et Swing.
- Classes de base.
- Création et affichage d'une fenêtre.
- Placer des composants dans une fenêtre .
- Création de Jar exécutable.
- Gestion des événements.
- Le modèle MVC.

GÉNÉRALITÉS SUR LES INTERFACES GRAPHIQUES

- Généralement les programmes informatiques nécessitent:
 - l’affichage de questions posées à l’utilisateur,
 - l’entrée de données par l’utilisateur au moment de l’exécution,
 - l’affichage d’une partie des résultats obtenus par le traitement informatique.

- Cet échange d’informations peut s’effectuer avec une interface utilisateur (**User Interface**) :
 - en mode texte,
 - ou console,
 - ou encore en **mode graphique**.

GÉNÉRALITÉS SUR LES INTERFACES GRAPHIQUES

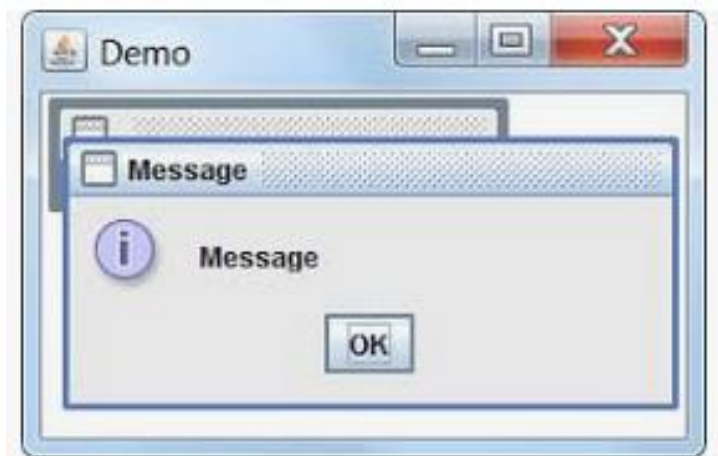
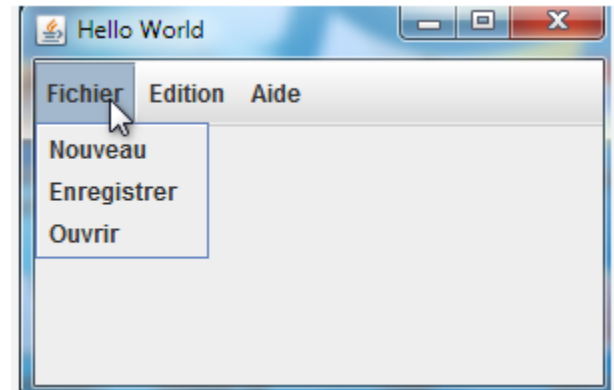
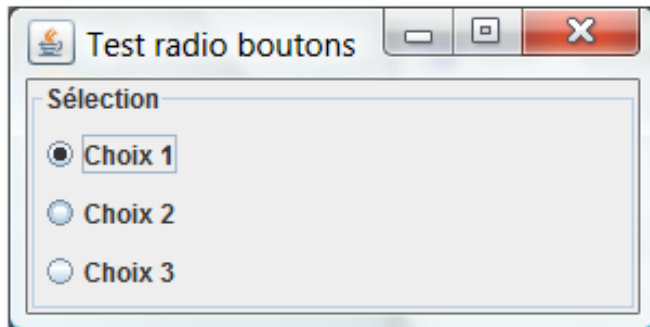
- Une interface graphique est souvent appelé **GUI** (Graphical User Interface), est un ensemble de **composants graphiques permettant à un utilisateur de communiquer avec un logiciel.**



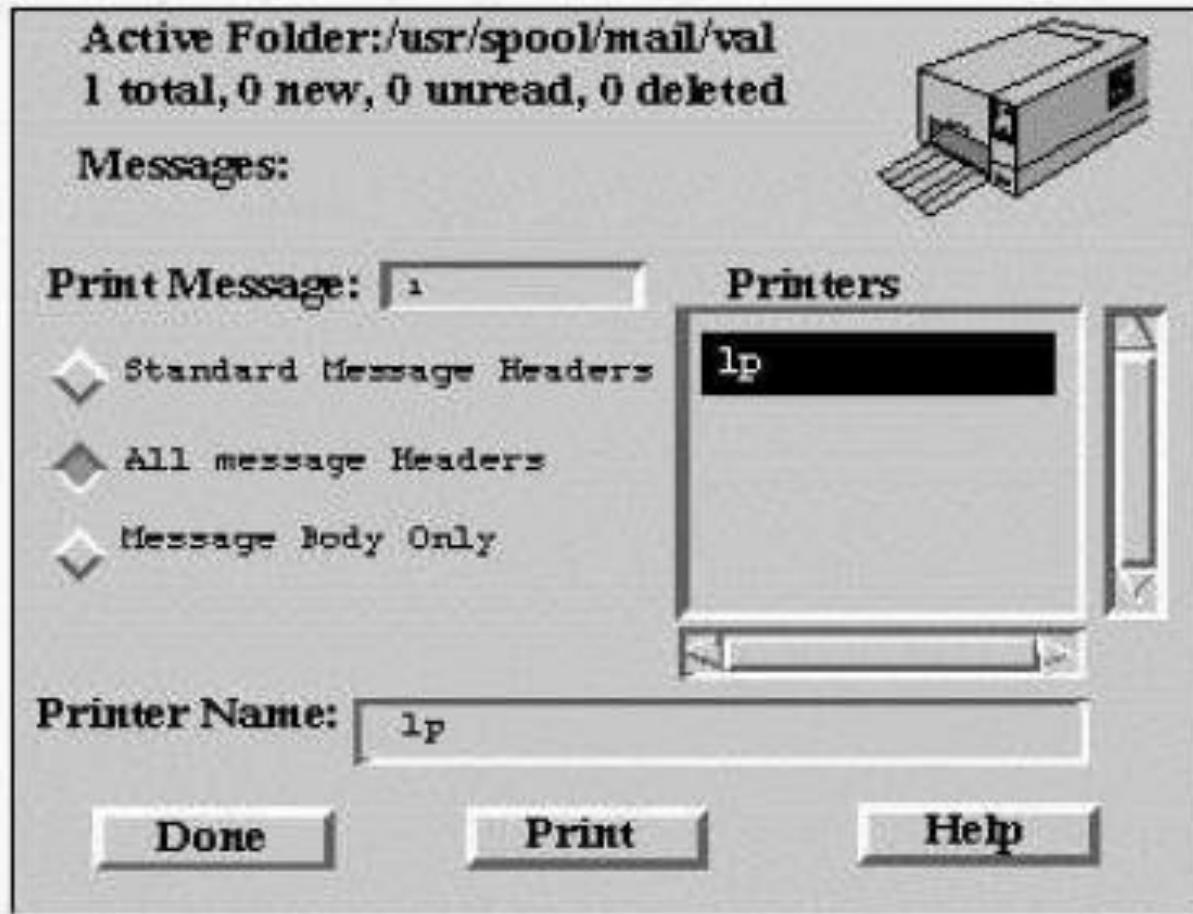
The screenshot shows a window titled "Gestion des salles ENSAJ" with standard Windows window controls (minimize, maximize, close). The interface contains the following elements:

- Two radio buttons for room type: "Salle de cours" (selected) and "Salle de réunion".
- Three text input fields on the left: "Numéro de la salle", "Numéro de l'étage", and "Capacité".
- Two text input fields on the right: "Nom" and "Créneau".
- A "Date" label next to a text input field.
- A "Type" label above two radio buttons: "Amphi" and "Salle".
- A "Valider" button at the bottom right.

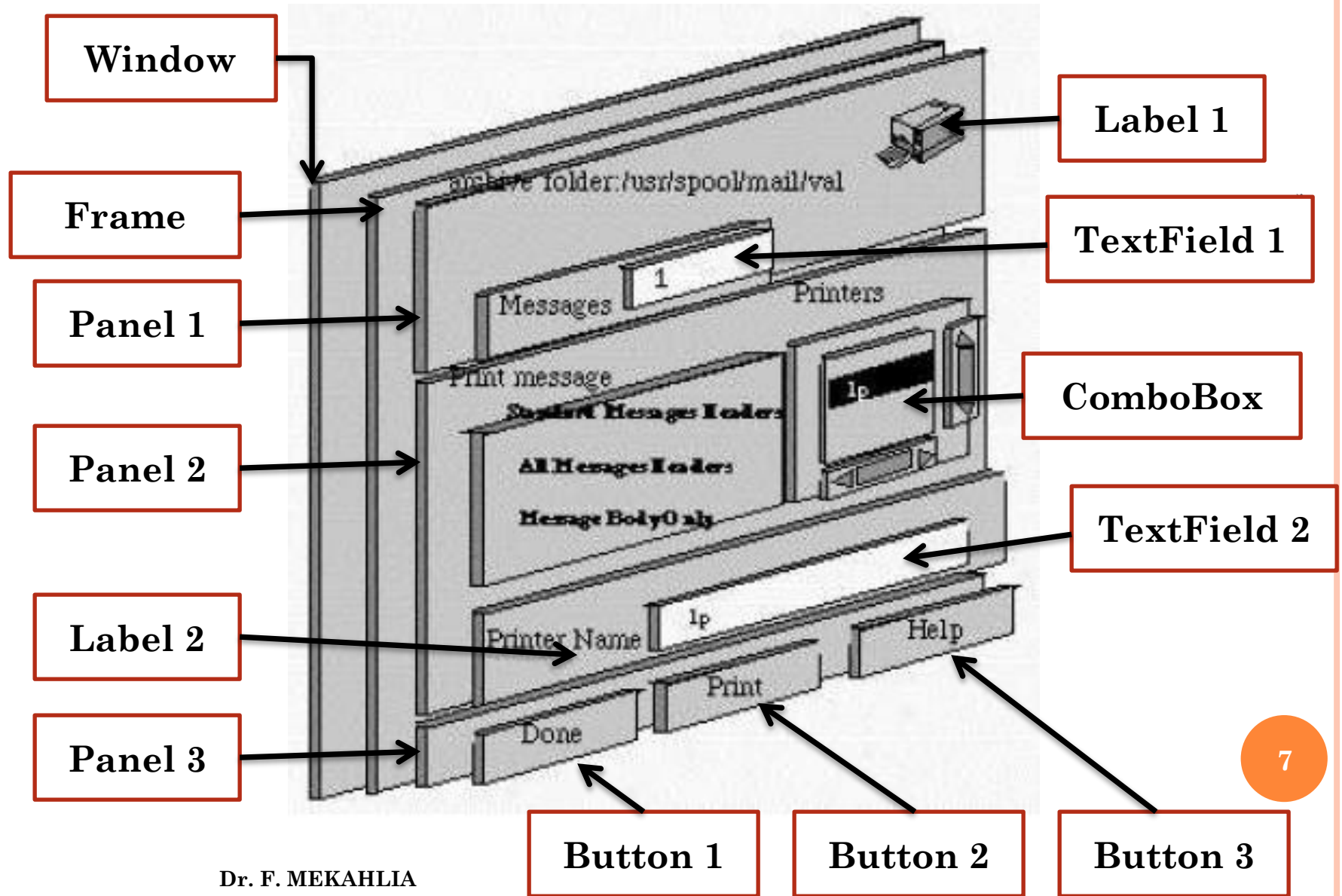
GÉNÉRALITÉS SUR LES INTERFACES GRAPHIQUES



COMPOSANTS DES INTERFACES GRAPHIQUES



COMPOSANTS DES INTERFACES GRAPHIQUES



COMPOSANTS DES INTERFACES GRAPHIQUES



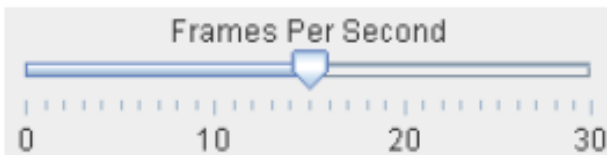
ProgressBar



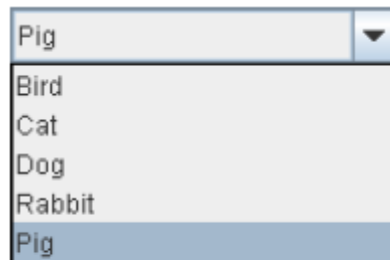
Dialog



ToolTip



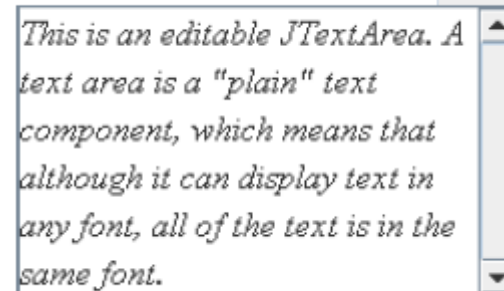
Slider



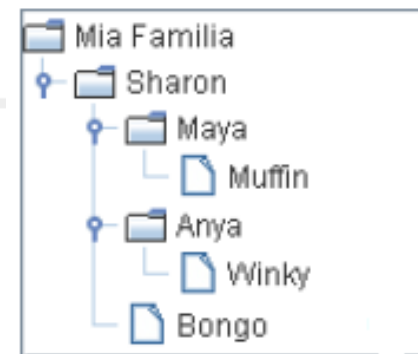
ComboBox



List



TextArea



Tree

More component in:

<https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html>



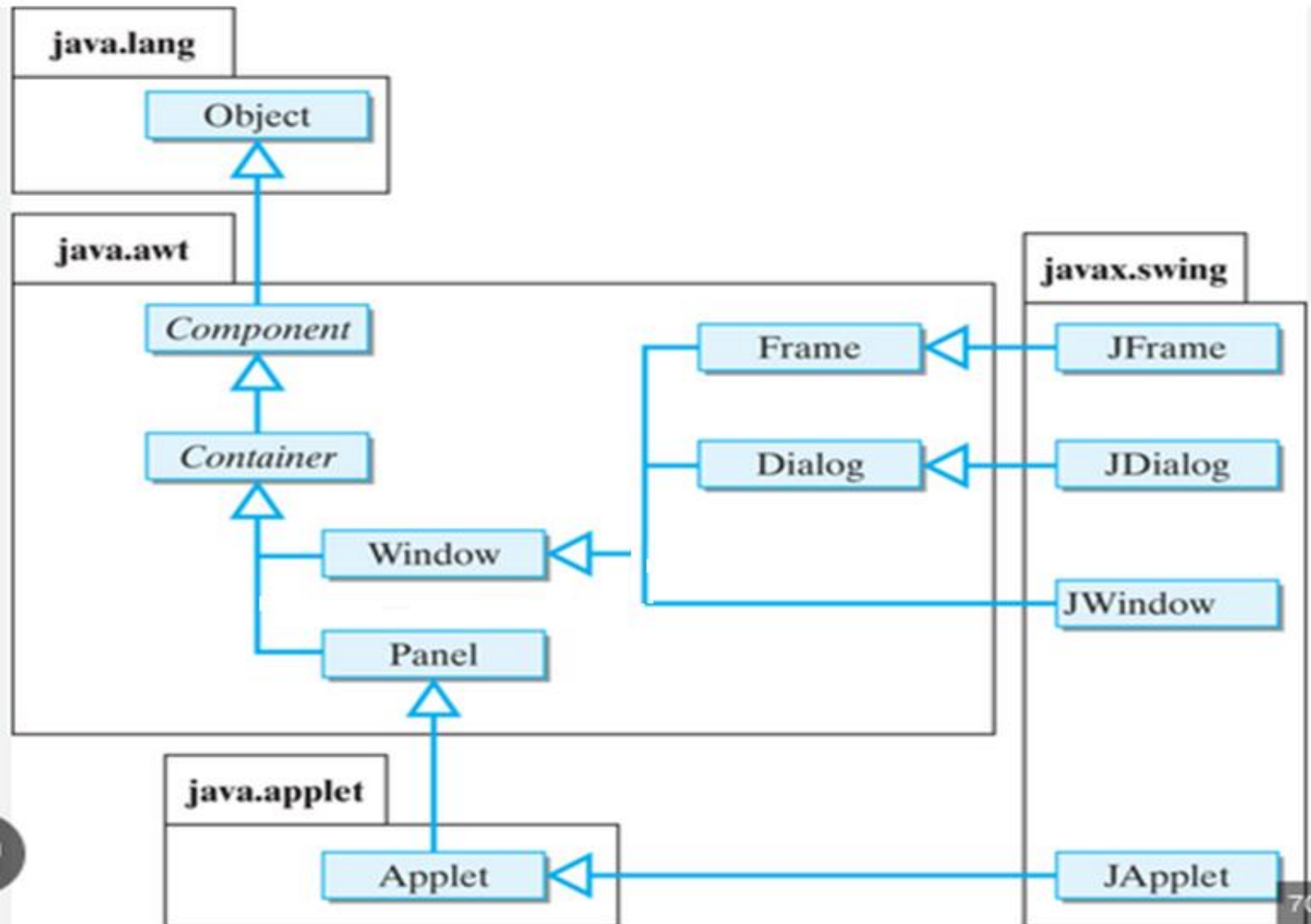
CheckBox

LES PACKAGES AWT ET SWING

- Généralement, les éléments graphiques sont définis dans deux grandes familles de composant graphique:
 - **AWT (abstract window toolkit, *JDK 1.1*)**.
 - **Swing (JDK 1.2)**.
- **Swing** est construit au-dessus de **AWT**:
 - même gestion des événements.
 - les classes de Swing héritent des classes de AWT.

23

LES PACKAGES AWT ET SWING



LES PACKAGES AWT ET SWING

- Swing et AWT font partie de JFC (Java Foundation Classes) qui offre des facilités pour construire des interfaces graphiques.
- Vous pouvez identifier les composants de swing à partir de leur nom qui débute par la lettre **J** (JDialog, JFrame etc.). Ces composants se trouvent dans le paquetage: **javax.swing**
- **x** vient du mot "eXtension« .
- Par contre les composants awt se trouvent dans le package **java.awt**

23

Les packages AWT

LES PACKAGES AWT

- Les classes du toolkit AWT permettent d'écrire des interfaces graphiques indépendantes du système d'exploitation sur lesquelles elles vont fonctionner.
- Cette librairie utilise le système graphique de la plateforme d'exécution (Windows, MacOS ou autre) pour afficher les objets graphiques.
- Le toolkit contient des classes décrivant les composants graphiques, les polices, les couleurs et les images.

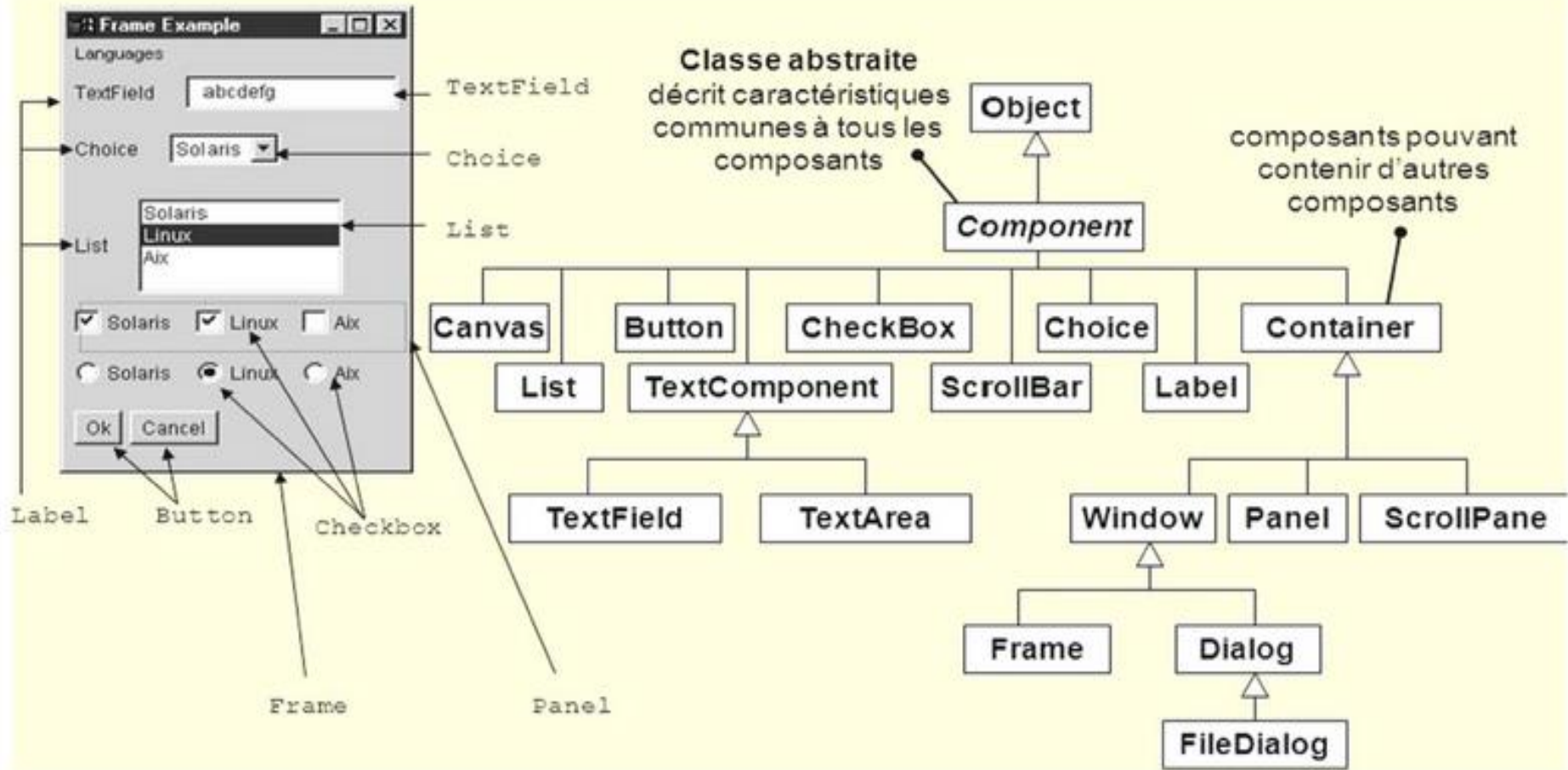
23

LES PACKAGES AWT

- L'apparence des fenêtres et boutons diffère d'un système d'exploitation à l'autre, car chaque composant AWT est dessiné et contrôlé par un composant **tiers natif** spécifique au système d'exploitation.

22

LES PACKAGES AWT



Les packages SWING

LES PACKAGES SWING

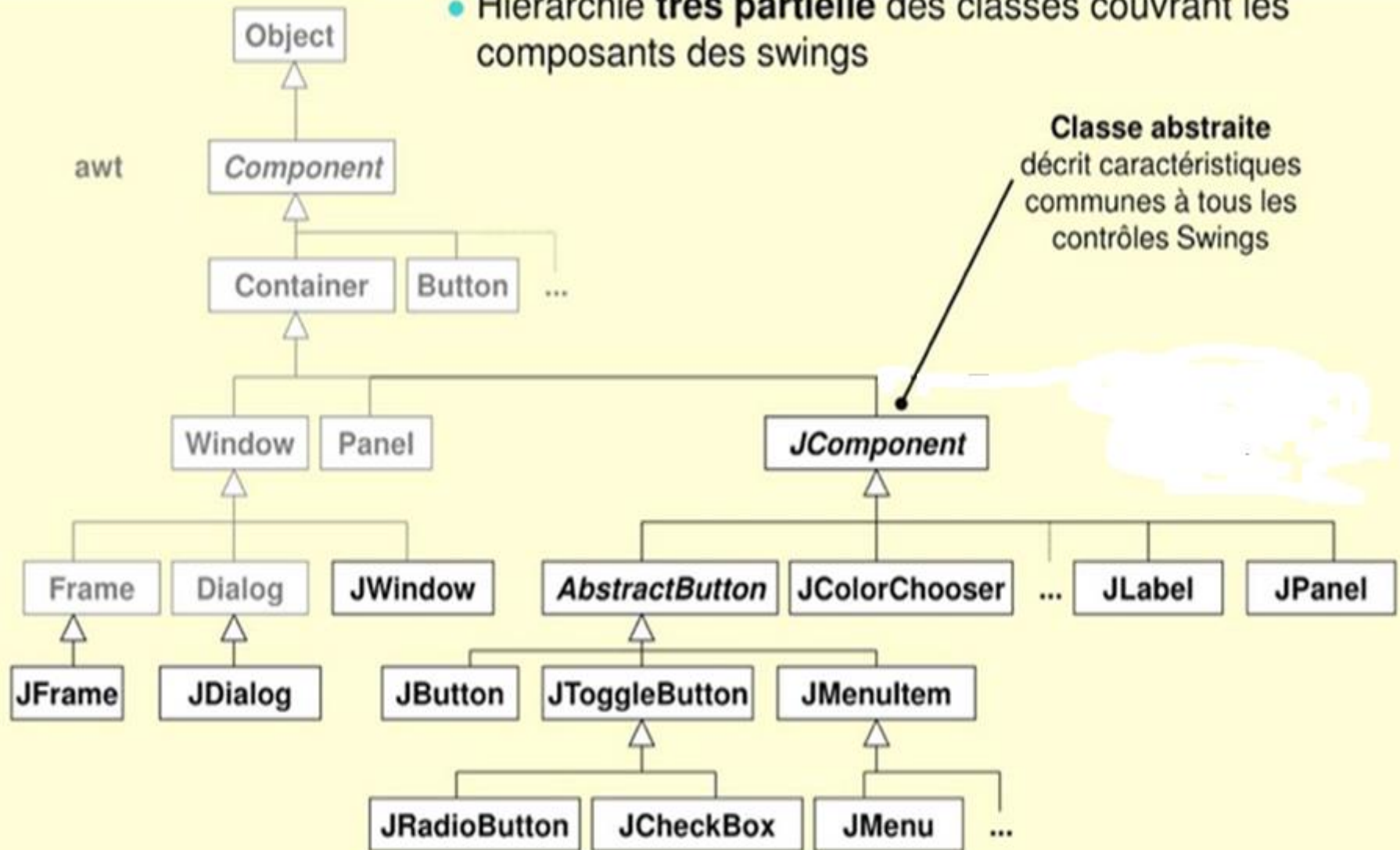
- Il existe un autre package d'interfaces graphiques et qui hérite de AWT appelée **Swing** qui est entièrement autonome, qui ne dépend pas du système d'exploitation.
- SWING ne requièrent pas d'allocation de ressources natives de la part du système d'exploitation, mais utilise les ressources de leurs ancêtres.

LES PACKAGES SWING

- Swing est une API dont le but est similaire à celle de l'AWT mais dont le mode de fonctionnement et d'utilisation est totalement différent.
- Swing a été intégrée au JDK depuis sa version 1.2. Cette bibliothèque existe séparément pour le JDK 1.1.
- La plupart des programmes utilisant Swing nécessite aussi l'importation de deux importants paquets de AWT :
 1. **`import java.awt.*;`**
 2. **`import java.awt.event.*;`**

LES PACKAGES SWING

- Hiérarchie **très partielle** des classes couvrant les composants des swings



LES PACKAGES SWING

- Les composants Swing forment une nouvelle hiérarchie parallèle à celle de l'AWT.
- Presque tous ces composants sont écrits en pure Java : ils ne possèdent **aucune partie native** **sauf** ceux qui assurent l'interface avec le système d'exploitation : JApplet, JDialog, JFrame, et JWindow.

23

AWT vs SWING

AWT vs SWING

- AWT utilise des composants natifs du système alors que Swing utilise des composants dessinés par Java.
- AWT est Lourd. Alors que les composants SWING sont généralement légers car ils n'ont pas besoin d'objets OS natifs pour implémenter leurs fonctionnalités.

22

AWT vs SWING

- Les composants AWT sont moins nombreux que les composants Swing. Alors que , les composants Java Swing sont bien plus nombreux.
- Les composants AWT ne suivent pas l'architecture MVC (Model View Controller). Tandis que, les composants Swing suivent le modèle MVC (Model View Controller).
- AWT signifie Abstract Windows Toolkit. Alors que, les composants Swing en Java sont également appelés JFC(Java Foundation Classes).

AWT vs SWING

- Les composants AWT ont besoin du package **java.awt**. Alors que, les composants Swing ont besoin du package **javax.swing**.
- Les composants AWT nécessitent et occupent un espace mémoire plus important. Tandis que, Les composants Swing n'occupent pas autant d'espace mémoire que les composants AWT.

23

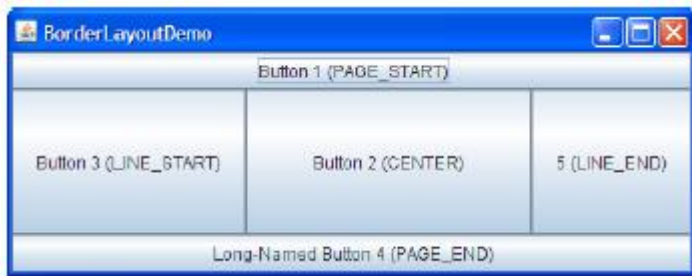
POUR INFORMATION ...

- il existe deux autres librairies de mise en oeuvre d'IHM:
- 1. Avec l'apparition de Java 8 **JavaFX** est désormais l'API d'interface graphique principale du Java. Dans sa conception, elle est plus moderne que Swing et permet de produire des interfaces graphiques pouvant facilement être utilisées sur différents types d'écrans (écran standard de PC, smartphone & tablettes et applications Web).

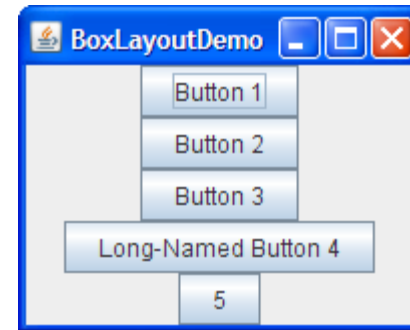
Architecture d'une application Swing

ARCHITECTURE D'UNE APPLICATION SWING

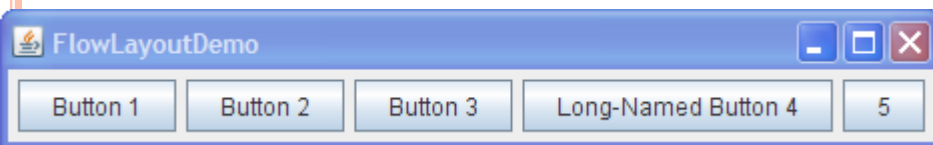
- Une application est un ensemble de fenêtres qui contient des items bien placées.
- Un conteneur (container) en top-level c'est le composant racine par exemple **JFrame**.
- JFrame contient d'autres composants qui peuvent être :
 - Composants atomiques (simples), par ex: un **bouton**.
 - Des composants intermédiaires qui permettent de diviser la fenêtre comme le **JPanel** (des panneaux).
- Le placement des composants dans un conteneur correspond à une stratégie de placement, par exemple soit délégué à un *LayoutManager* (qui est interface).



BorderLayout : présentation avec bordures.



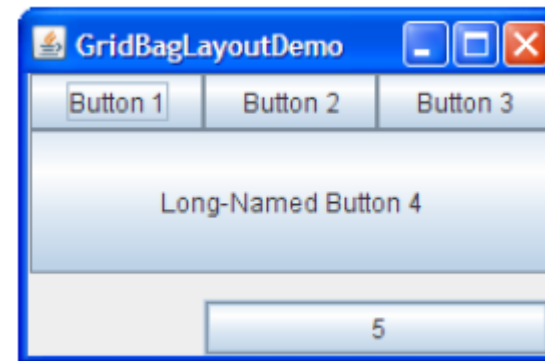
BoxLayout : en ligne ou en colonne



FlowLayout : présentation en file



GridLayout : en grille

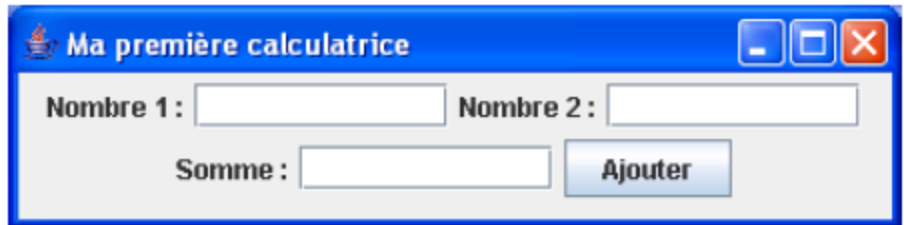


GridBagLayout : en grille composite qui est plus sophistiqué.

Cas d'études

CAS D'ÉTUDES

- Nous voulons créer une calculatrice simple capable d'ajouter deux nombres et d'afficher le résultat.



JFame: Ma première calculatrice

JPanel: contenuFenêtre

JLabel:
Nombre 1

JLabel:
Nombre 2

JLabel:
Somme

JButton:
lancer

JTextFie
Id :
entrée1

JTextFie
Id :
entrée2

JTextFie
Id :
résultat

- **import javax.swing.*;**
- **import java.awt.FlowLayout;**
- **public class CalculatriceSimple {**
- **public static void main(String[] args) {**
- // Crée la fenetre**
- **JFrame fenetre= new JFrame("Ma première calculatrice");**
- // Créer un panneau**
- **JPanel contenuFenêtre = new JPanel();**
- // Affecter un gestionnaire de disposition à ce panneau**
- **FlowLayout disposition = new FlowLayout();**
- **contenuFenêtre.setLayout(disposition);**
- // Créer les contrôles en mémoire**
- **JLabel label1 = new JLabel("Nombre 1 :");**
- **JTextField entrée1 = new JTextField(10);**
- **JLabel label2 = new JLabel("Nombre 2 :");**
- **JTextField entrée2 = new JTextField(10);**
- **JLabel label3 = new JLabel("Somme :");**
- **JTextField résultat = new JTextField(10);**
- **JButton lancer = new JButton("Ajouter");**

23

// Ajoute les contrôles au panneau

- contenuFenêtre.add(label1);
- contenuFenêtre.add(entrée1);
- contenuFenêtre.add(label2);
- contenuFenêtre.add(entrée2);
- contenuFenêtre.add(label3);
- contenuFenêtre.add(résultat);
- contenuFenêtre.add(lancer);

// ajouter le panneau dans la fenetre

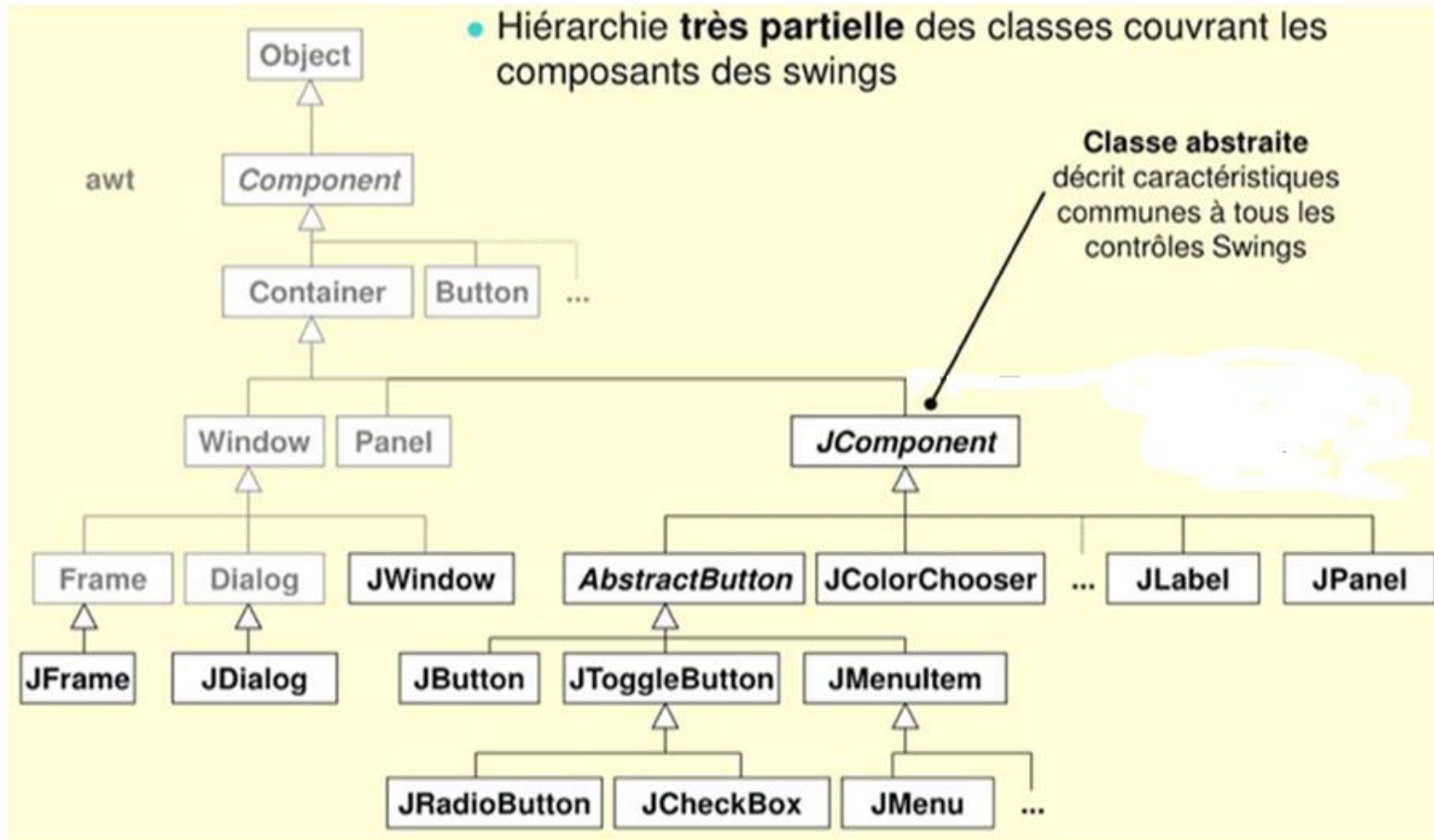
- fenetre.setContentPane(contenuFenêtre);

// Positionner les dimensions et rend la fenêtre visible

- fenetre.setSize(400,100);
- fenetre.setVisible(**true**);} }

Classes de base

CLASSES DE BASE

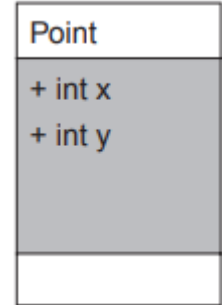


CLASSES DE BASE

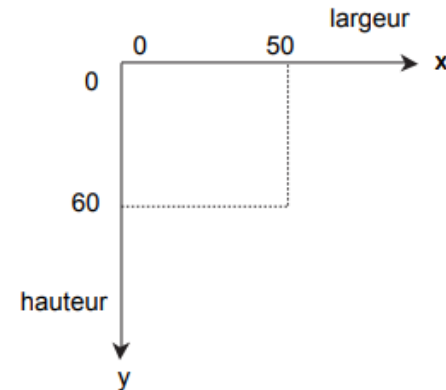
- Pour pouvoir comprendre les fonctionnalités des classes de base, nous allons commencer par la présentation des classes Point, Dimension, Rectangle, Color et Graphics qui sont souvent utilisées dans la création des interfaces graphiques.

2

LA CLASSE POINT



- **Point:**
- La classe Point (**java.awt.Point**) définit un point repéré par ses attributs entiers x et y sur un plan ou à partir d'un autre Point.



- **Elle implémente l'interface:** Le système des coordonnées graphiques : l'axe des y est vers le bas.

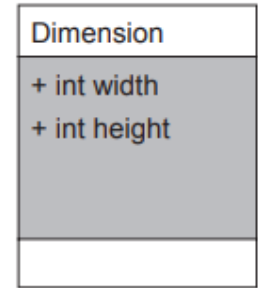
java.io.Serializable

- **Elle hérite les packages:**

java.lang.Object

java.awt.geom.Point2D

LA CLASSE DIMENSION



Dimension:

- La classe `Dimension` (**java.awt.Dimension**) définit deux attributs entiers représentant les dimensions d'un objet en largeur et en hauteur (width et height).
- Un objet `Dimension` peut aussi être construit à partir d'un autre objet `Dimension`, ou à partir de deux entiers.

- Elle implémente l'interface:

java.io.Serializable

- **Elle hérite les packages:**

java.lang.Object

java.awt.geom.Point2D

LA CLASSE RECTANGLE

Rectangle

+ int x
+ int y
+ int width
+ int height

Rectangle:

- La classe Rectangle définit un rectangle sur le plan caractérisé par le point **p** repérant son coin supérieur gauche (CSG),
- et par sa dimension **d** (largeur et hauteur).
- Rectangle a 4 attributs public : x et y, coordonnées du point, et width et height, valeurs de d.
- **Elle implémente les interfaces:**

java.awt.Shape

java.io.Serializable

- **Elle hérite les packages :**

java.lang.Object

java.awt.geom.RectangularShape

java.awt.geom.Rectangle2D

LA CLASSE COLOR

Color

+ static final Color black
+ static final Color blue
+ etc.

Color:

- La classe Color permet d'accéder à des constantes static indiquant 13 couleurs prédéfinies :
- Color.black Color.blue Color.cyan Color.darkGray
Color.gray Color.green Color.lightGray Color.magenta
Color.orange Color.pink Color.red Color.white
Color.yellow
- Une couleur est créée à partir de ses trois composantes (RVB : rouge, vert, bleu), chaque composante ayant une valeur sur 8 bits (de 0 à 255).
- On peut créer une nouvelle couleur en indiquant ses 3 composantes. Exemple : new Color (100, 100, 100) ; crée un objet référençant une nouvelle couleur.

LA CLASSE COLOR

Color:

- Elle implémente les interfaces:

java.io.Serializable

Java.awt.Paint

- Elle hérite les packages:

java.lang.Object

Color

+ static final Color black
+ static final Color blue
+ etc.

”

LA CLASSE GRAPHICS

Graphics:

- La classe Graphics est une classe abstraite encapsulant (contenant) des informations permettant de dessiner des formes géométriques ou du texte pour un composant.
- Chaque composant a un contexte graphique.
- Les principales méthodes sont :
 - . drawImage (Image, int, int, Color, ImageObserver) : trace une image.
 - drawLine (int, int, int, int) : trace une ligne droite entre deux points.
 - drawRect (int, int, int, int) : trace un rectangle.
 - drawOval (int, int, int, int) : trace un ovale (ellipse) dans le rectangle défini.
 - drawArc (int, int, int, int, int, int) : trace un arc (une partie d'ellipse)

LA CLASSE GRAPHICS

- `drawString (String, int, int)` : trace une chaîne de caractères.
- `fillRec (int, int, int, int)` : remplit un rectangle.
- `getColor ()` : fournit la couleur courante.
- `getFont ()` : fournit la fonte courante.
- `setColor (Color)` : change la couleur courante.
- `setFont (Font)` : change la fonte courante.

NB: La classe **Font** définit les caractéristiques d'une police de caractères

LA CLASSE GRAPHICS

Exemple d'utilisation:

(g est un objet de type Graphics)

`g.setColor (Color.red); // couleur courante : rouge`

`g.drawString ("Bonjour", 5, 50); // "Bonjour" en x = 5
et y = 50`

`g.drawRect (10, 10, 50, 50); // Rectangle de CSG
(10,10); w = 50; h = 50`

LA CLASSE COMPONENT (COMPOSANT GRAPHIQUE)

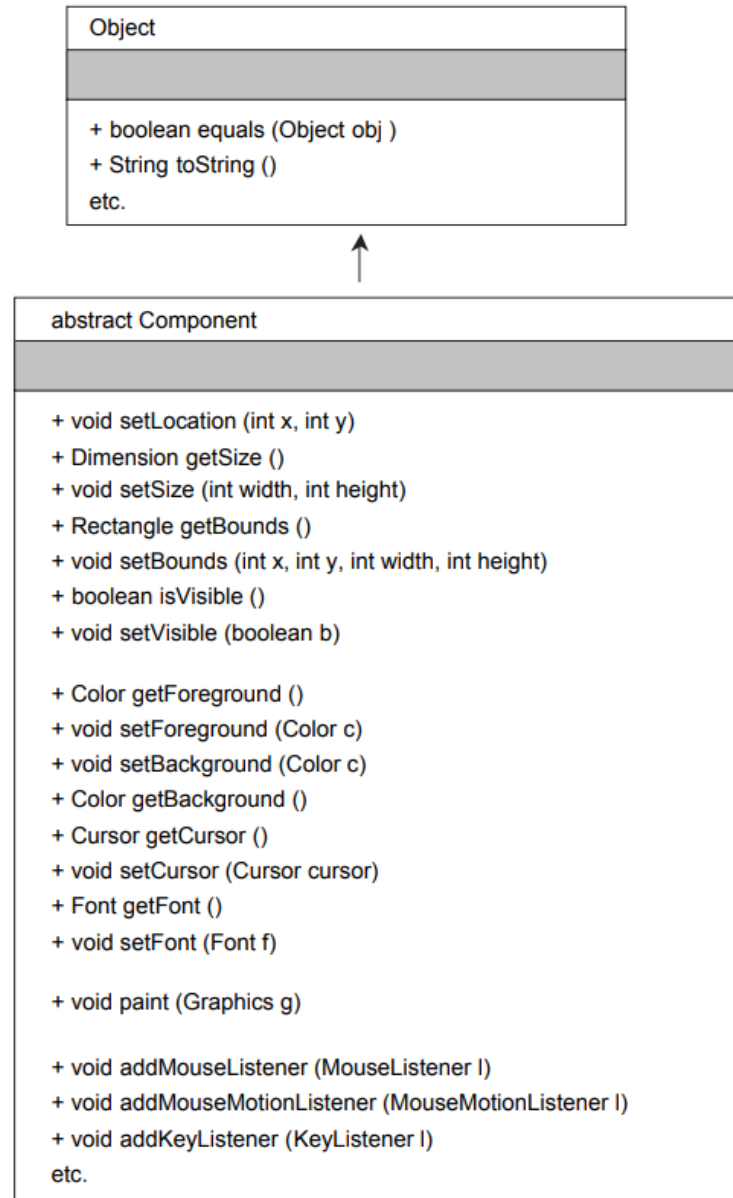
- La classe **abstraite Component** de AWT dérive de la classe Object et contient une bonne centaine de méthodes.
- Elle contient des méthodes permettant d'accéder ou de modifier les caractéristiques communes à tous les composants : dimension du composant, coordonnées du coin supérieur gauche du composant, couleur du texte ou du fond est-il visible ou non ? **etc**

LA CLASSE COMPONENT (COMPOSANT GRAPHIQUE)

- Les composants sont disposés sur la fenêtre initiale. Certains composants peuvent recevoir d'autres composants : on les appelle des conteneurs (container en anglais). L'ensemble des composants d'une application constitue un arbre des composants, la racine de l'arbre étant la fenêtre initiale de l'application.
- La disposition des composants dans la fenêtre peut être imposée par le programme qui indique, pour chaque composant, le coin supérieur gauche du composant (CSG), sa largeur et sa hauteur ou se faire suivant un gestionnaire de mise en page (**LayoutManager**).

LA CLASSE COMPONENT (COMPOSANT GRAPHIQUE)

Il s'agit d'une classe abstraite qui hérite de Object et définit les caractéristiques communes aux composants graphiques.



LA CLASSE COMPONENT (COMPOSANT GRAPHIQUE)

- Les principales méthodes de la classe Component sont:
 - public **Rectangle getBounds()** : fournit un objet Rectangle indiquant la position du composant (CSG : coin supérieur gauche) par rapport au conteneur et ses dimensions (largeur et hauteur) lors de l'appel de cette méthode,
 - public **Dimension getSize ()** : fournit la taille du composant (largeur et hauteur).
 - public **void setBounds (int x, int y, int width, int height)** : modifie le point CSG par rapport au conteneur et la taille du composant.
 - public **void setSize (int width, int height)** : modifie la taille du composant.

LA CLASSE COMPONENT (COMPOSANT GRAPHIQUE)

- **public void setVisible (boolean b)** : le composant est visible si b est vrai, invisible sinon. Par défaut, les composants sont visibles, sauf les fenêtres qui sont a priori invisibles (Frame par exemple).
- **public Color getBackground ()** : fournit la couleur du fond du composant. Si aucune couleur n'est définie pour le composant, c'est celle du conteneur qui est utilisée.
- **public Color getForeground ()** : fournit la couleur du premier plan (texte ou dessin) du composant. Si aucune couleur n'est définie pour le composant, c'est celle du conteneur qui est utilisée.

LA CLASSE COMPONENT (COMPOSANT GRAPHIQUE)

- **public void setBackground (Color c)** : modifie la couleur du fond du composant.
- **public void setForeground (Color c)** : modifie la couleur du premier plan du composant.
- **public Graphics getGraphics ()** : fournit un objet de type Graphics pour ce composant.
- **public void paint (Graphics g)** : dessine le composant en utilisant le contexte graphique g.
- **public void update (Graphics g)** : redessine le fond et appelle la méthode paint (g).
- **public void repaint ()** : appelle update (g), g étant le contexte graphique du composant

LA CLASSE LABEL

- La classe **Label** définit une ligne de texte que le programme peut changer en cours d'exécution. Pour l'utilisateur, ce Label est une chaîne constante qu'il ne peut pas modifier sur son écran.
- Ce composant dispose de toutes les méthodes de Component (héritage). On peut lui définir un emplacement, une taille, une couleur de fond ou de texte, une fonte particulière.
- A cela, s'ajoute quelques spécificités des Label concernant la justification (gauche, centré, droite) dans l'espace attribué.

LA CLASSE LABEL

- Attributs et méthodes de Label :

- Label.CENTER, Label.LEFT, Label.RIGHT : constantes static pour justifier le texte.

- Label (), Label (String eti), Label (String eti, int justification) : constructeurs de Label,

- String getText () : fournit le texte du Label.

- void setText (String text) : modifie le texte du Label,

LA CLASSE BUTTON

- La classe **Button** définit un bouton affichant un texte et déclenchant une action lorsqu'il est activé à l'aide de la souris.
- Exemple : un bouton Quitter.
- Méthodes :
 - Button (String label) : constructeur d'un bouton ayant l'étiquette label.
 - String getLabel () : fournit l'étiquette du bouton.
 - void setLabel (String label) : modifie l'étiquette du bouton.

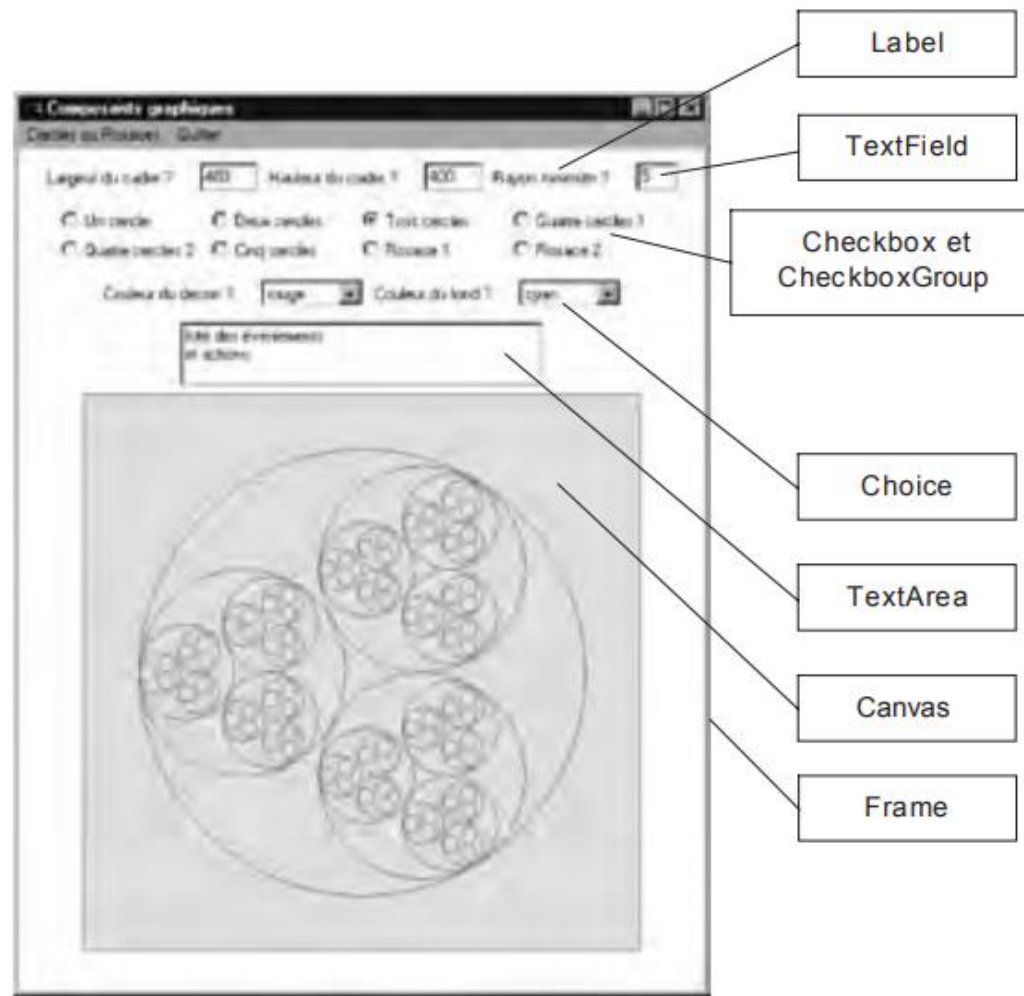
LA CLASSE BUTTON

- `void addActionListener (ActionListener l)` : enregistre un objet de type `ActionListener` qui indique ce qu'il faut faire quand on appuie ou relâche le bouton. Cet enregistrement est retiré à l'aide de la méthode `removeActionListener()`. Le bouton devient alors inopérant.
- `void setActionCommand (String command)` : définit la chaîne de caractères qui est délivrée lorsque le bouton est activé.
- `public String getActionCommand ()` : fournit la chaîne définie par `setActionCommand()` ou l'étiquette du bouton par défaut.

LA CLASSE CANEVA

- La classe **Canvas** définit une zone rectangulaire de dessin.
- Méthodes sont :
 - Canvas () : constructeur de la zone de dessin.
 - void paint (Graphics g) : repeint le composant avec la couleur du fond.
 - Les applications doivent dériver la classe Canvas et redéfinir la méthode paint() en fonction de l'application.

LA CLASSE CANEVA



LA CLASSE CHECKBOX

- La classe `Checkbox` définit une boîte à cocher. Celle-ci a une étiquette et un état booléen coché ou non.
- Les méthodes sont:
 - `public Checkbox (String label, boolean etat)` : crée une boîte à cocher d'étiquette `label` avec l'état coché ou non suivant le booléen `etat`,
 - `public Checkbox (String label, boolean etat, CheckboxGroup group)` : constructeur qui insère la boîte dans un groupe de boîtes.
 - `String getLabel ()` : fournit l'étiquette de la boîte à cocher.
 - `boolean getState ()` : fournit l'état de la boîte à cocher.

LA CLASSE CHECKBOX

- **void addItemListener (ItemListener l) :** l'objet l de type ItemListener indique ce qu'il faut faire si la boîte à cocher est sélectionnée.

- **removeItemListener (ItemListener l):** annule l'opération précédente ; l'écouteur l devient inopérant.

LA CLASSE CHOICE, LIST ET SCROLLBAR

- La **classe Choice** définit un menu contenant une liste d'options dont une seule peut être activée (choisie).
- La **classe List** définit une liste déroulante d'éléments. Le nombre d'éléments affichés peut être modifié. On peut également sélectionner un ou plusieurs éléments de la List.
- La **classe Scrollbar** définit une barre de défilement vertical ou horizontal Cette barre représente les valeurs entières comprises entre les deux entiers minimum et maximum. Le curseur représente la valeur courante de la barre. La valeur courante peut être modifiée (avec un incrément paramétrable) suite à des clics de la souris sur cette barre.

LA CLASSE TEXTFIELD ET TEXTAREA

- La **classe TextField** est une zone de saisie d'une seule ligne de texte. Les caractères entrés peuvent être remplacés par un caractère choisi dit écho (pour les mots de passe par exemple),
- La **classe TextArea** est une zone de saisie ou d'affichage de texte sur plusieurs lignes. Cette zone peut avoir ou non des barres de défilement lorsque la fenêtre est trop petite pour afficher tout le texte horizontalement ou verticalement.

LA CLASSE CONTAINER

<i>abstract Container</i>
+ Component add (Component <i>comp</i>)
+ Component getComponent (int <i>n</i>)
+ void remove (int <i>n</i>)
+ void remove (Component <i>comp</i>)
+ int getComponentCount ()
+ void setLayout (LayoutManager <i>mgr</i>)
+ paint (Graphics <i>g</i>)
+ etc.

- Un objet de type **Container** est un composant graphique qui peut contenir d'autres composants graphiques tels que ceux définis précédemment : Label, Button, Checkbox, Choice, Scrollbar, etc. ou un autre container.
- Les composants ajoutés à un conteneur sont enregistrés dans une liste. On peut retirer un composant de la liste. À un conteneur, on peut ajouter

LA CLASSE PANEL

- La **classe Panel** définit un conteneur fournissant de l'espace dans lequel on peut ajouter d'autres composants y compris d'autres Panel. La mise en page est par défaut de type FlowLayout. La classe définit deux constructeurs, le deuxième précisant le type de gestionnaire de mise en page choisi. Les autres caractéristiques sont définies à l'aide des méthodes de la super-classe Component (espace occupé, couleur de fond, etc.)

LA CLASSE WINDOW ET FRAME

- La **classe Window** définit une fenêtre sans bordure et sans barre de menu. Par défaut, la mise en page est de type BorderLayout et la fenêtre est invisible. La fenêtre est sensible aux clics de la souris qui la font passer par défaut au premier plan. Un objet Window a obligatoirement pour père un objet de type Frame, Window ou Dialog qui doit être fourni au constructeur,
- La **classe Frame** définit une fenêtre principale d'application ayant un titre et une bordure. Par défaut, la mise en page est de type BorderLayout et la fenêtre est invisible.

LA CLASSE DIALOG

- La **classe Dialog** définit une fenêtre de dialogue (saisie ou affichage de messages). On ne peut rien faire d'autre tant que la fenêtre n'est pas fermée.
- **Remarque** : un composant de type Window (et donc Frame et Dialog) ne peut être ajouté à un conteneur. C'est un composant de niveau 1 (top-level window).

LA CLASSE JComponent

- **JComponent** est la classe de base de tous les composants Swing (JButton, JPanel, etc.) à l'exception des conteneurs de niveau supérieur (JFrame et JDialog). hérite la classe Container qui hérite elle-même Component.

LA CLASSE JFrame

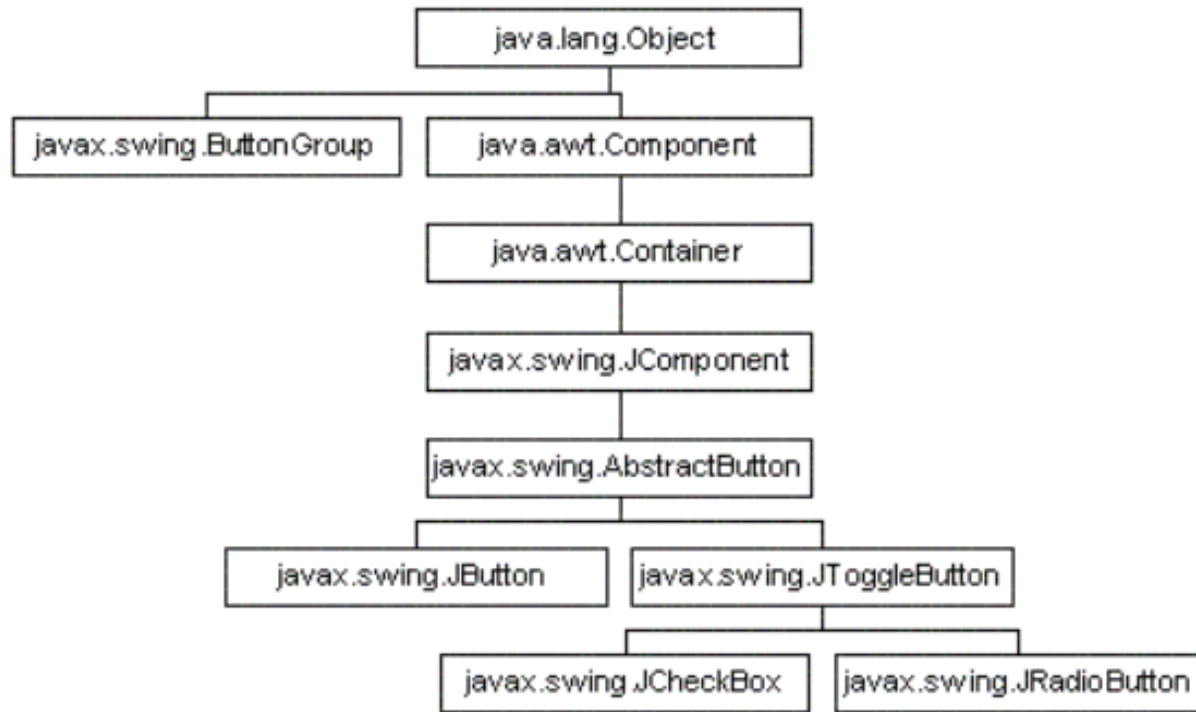
- Contre Frame, la JFrame utilise un Panneau de contenu pour insérer des composants (ils ne sont plus insérés dans le JFrame mais dans l'objet contentPane qui lui est associé). Elle représente une fenêtre principale qui possède un titre, une taille modifiable et éventuellement un menu.
- Par défaut, une JFrame est affichée dans le coin supérieur gauche de l'écran. Pour la centrer dans l'écran, il faut procéder comme pour une Frame : déterminer la position de la Frame en fonction de sa dimension et de celle de l'écran et utiliser la méthode setLocation() pour affecter cette position.
- La méthode setIconImage() permet de modifier l'icône de la JFrame,

LA CLASSE JLABEL ET JPANEL

- JLabel propose les mêmes fonctionnalités que les intitulés AWT mais ils peuvent en plus contenir des icônes .
- La classe JPanel est un conteneur utilisé pour regrouper et organiser des composants grâce à un gestionnaire de présentation (layout manager). Le gestionnaire par défaut d'un JPanel est un objet de la classe FlowLayout.

LA CLASSE JBUTTON

- Il existe plusieurs boutons définis par Swing.



LA CLASSE JBUTTON

1. **JButton** est un composant qui représente un bouton : il peut contenir un texte et/ou une icône.
2. **JToggleButton** définit un bouton à deux états : c'est la classe mère des composants JCheckBox et JRadioButton.
 - La méthode **setSelected()** héritée de AbstractButton permet de mettre à jour l'état du bouton. La méthode **isSelected()** permet de connaître cet état.
3. **ButtonGroup** permet de gérer un ensemble de boutons en garantissant qu'un seul bouton du groupe sera sélectionné

LA CLASSE JBUTTON

4. **JRadioButton** représente un groupe de boutons . A un instant donné, un seul des boutons radio associés à un même groupe peut être sélectionné. La classe JRadioButton hérite de la classe AbstractButton.

- pour chacun des états du bouton, en utilisant les méthodes setIcon(), setSelectedIcon() et setPressedIcon().

LA CLASSE JTEXTCOMPONENT

- La classe abstraite **JTextComponent** est la classe mère de tous les composants permettant la saisie de texte.
- **TextField** permet la saisie d'une seule ligne de texte simple.
- **PasswordField** permet la saisie d'un texte dont tous les caractères saisis seront affichés sous la forme d'un caractère particulier ('*' par défaut). Cette classe hérite de la classe `TextField`.
- La méthode `setEchoChar(char)` permet de préciser le caractère qui sera montré lors de la saisie.
- la méthode `getPassword()` est utilisée pour obtenir la valeur du texte saisi.


LA CLASSE JTEXTCOMPONENT

- **JEditorPane** permet la saisie de texte multilignes. Ce type de texte peut contenir des informations de mise en pages et de formatage. En standard, Swing propose le support des formats RTF et HTML.
- **JTextArea** permet la saisie de texte simple en mode multiligne.

Cas d'étude: Création et affichage d'une fenêtre

CRÉATION ET AFFICHAGE D'UNE FENÊTRE

```
Title("Jeu de dame");
```

Siz  Jeu de dame

Loc 

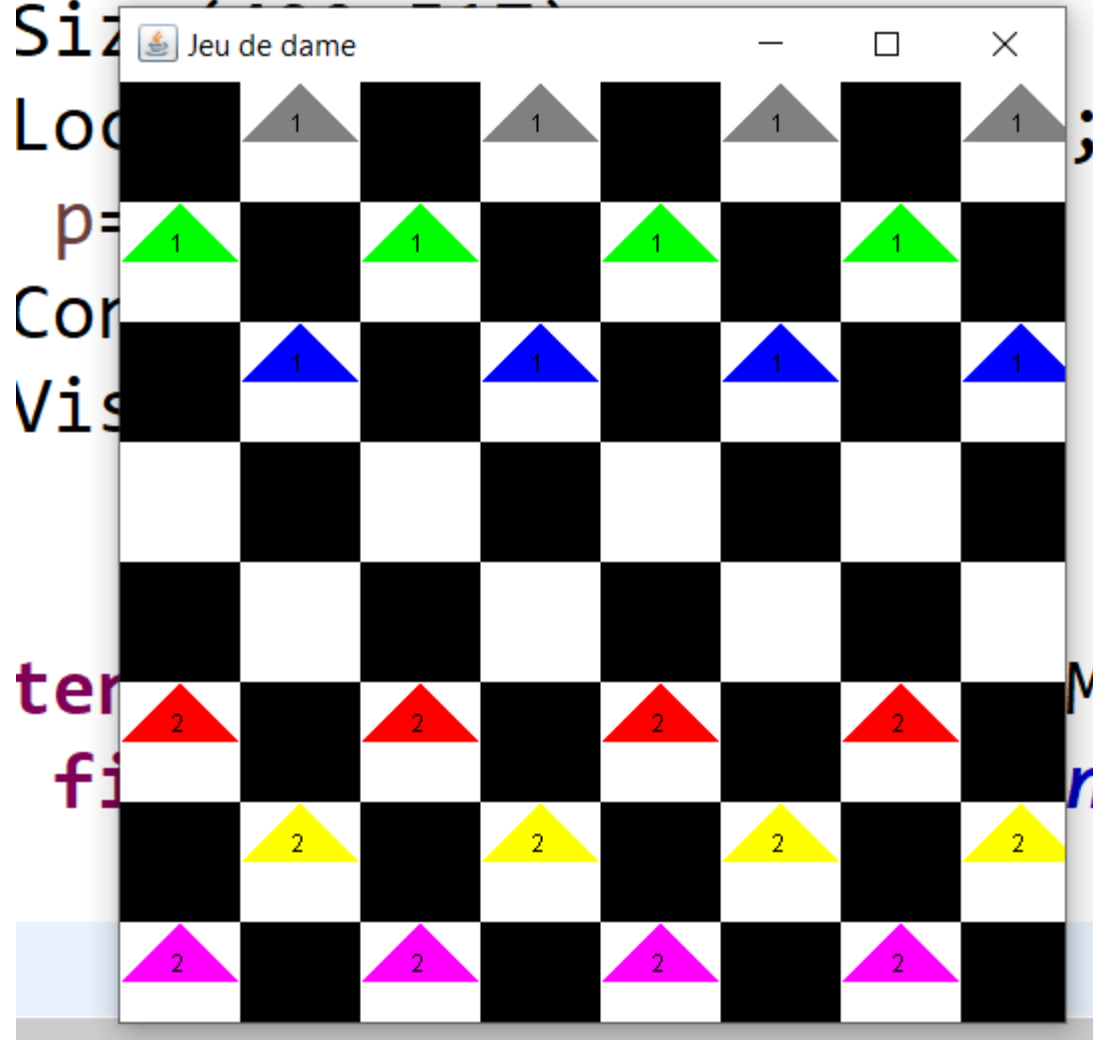
p=

Cor

Vis

ter

fi



CRÉATION ET AFFICHAGE D'UNE FENÊTRE

```
import javax.swing.*;
```

```
public class Table {  
    public static void main(String[] args) {  
        JFrame jf=new JFrame();  
        jf.setTitle("Jeu de dame");  
        jf.setSize(490,517);  
        jf.setLocationRelativeTo(null);  
        Dessin p=new Dessin();  
        jf.getContentPane().add(p);  
        jf.setVisible(true) ;  
    }  
}
```

**Placer des composants dans une
fenêtre**

PLACER DES COMPOSANTS DANS UNE FENÊTRE

```
class Dessin extends JPanel
{
    int [][]m = M();
    public void paint(Graphics g)
    {
        new Case(g,m);
        new Pièce(g,m);
    }
    public static int[][] M() {
        int [][] m=
            { {-1,1,-1,2,-1,3,-1,4},
              {5,-1,6,-1,7,-1,8,-1},
              {-1,9,-1,10,-1,11,-1,12},
              {-3,-1,-3,-1,-3,-1,-3,-1},
              {-1,-3,-1,-3,-1,-3,-1,-3},
              {13,-1,14,-1,15,-1,16,-1},
              {-1,17,-1,18,-1,19,-1,20},
              {21,-1,22,-1,23,-1,24,-1}};
    }
}
```

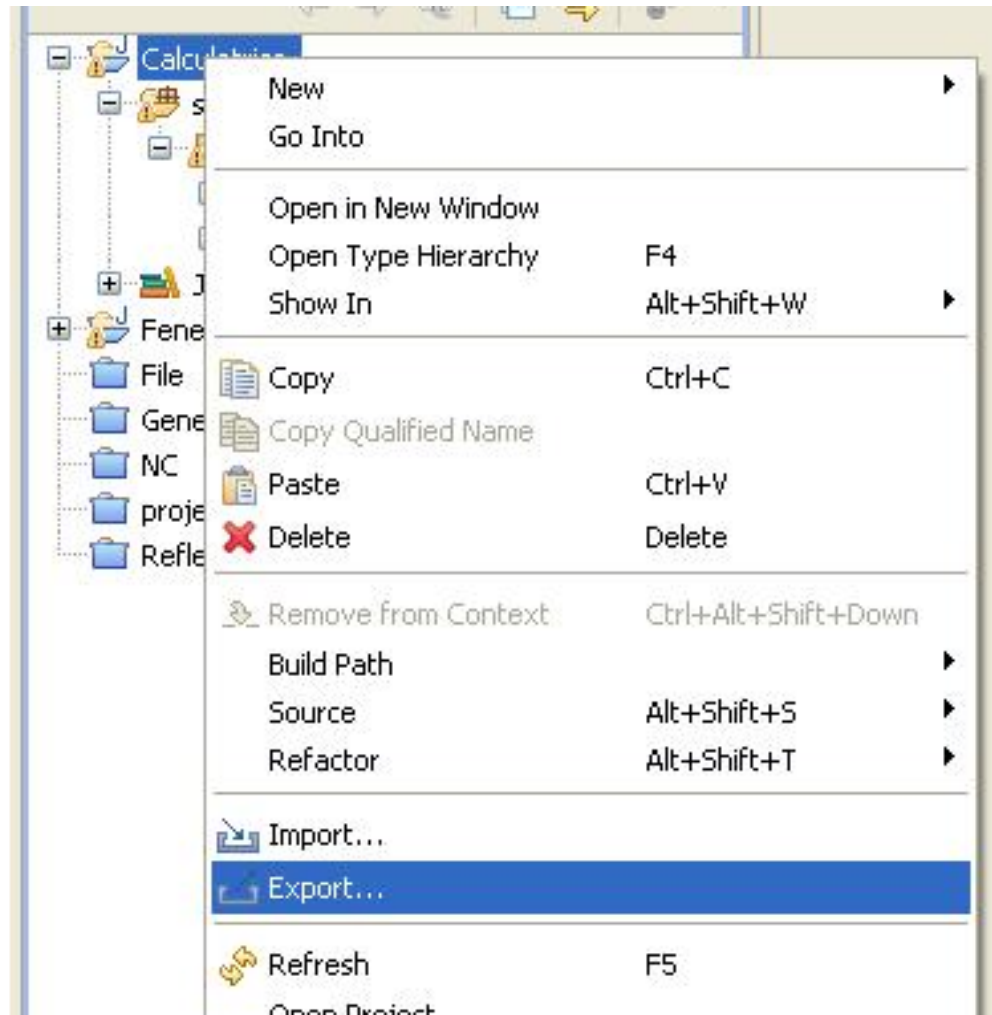
Création de Jar exécutable

CRÉATION DE JAR EXÉCUTABLE

- Un fichier **jar** est une archive Java (**J**ava **A**Rchive). Ce type de fichier contient tout ce dont a besoin la JVM pour lancer l'exécution du programme !
- Une fois votre archive créée, il vous suffira juste de double cliquer sur celle-ci pour lancer l'application.
- Avec la condition d'ajouter les exécutables de JRE (présent dans le répertoire /bin) dans la variable d'environnement PATH !

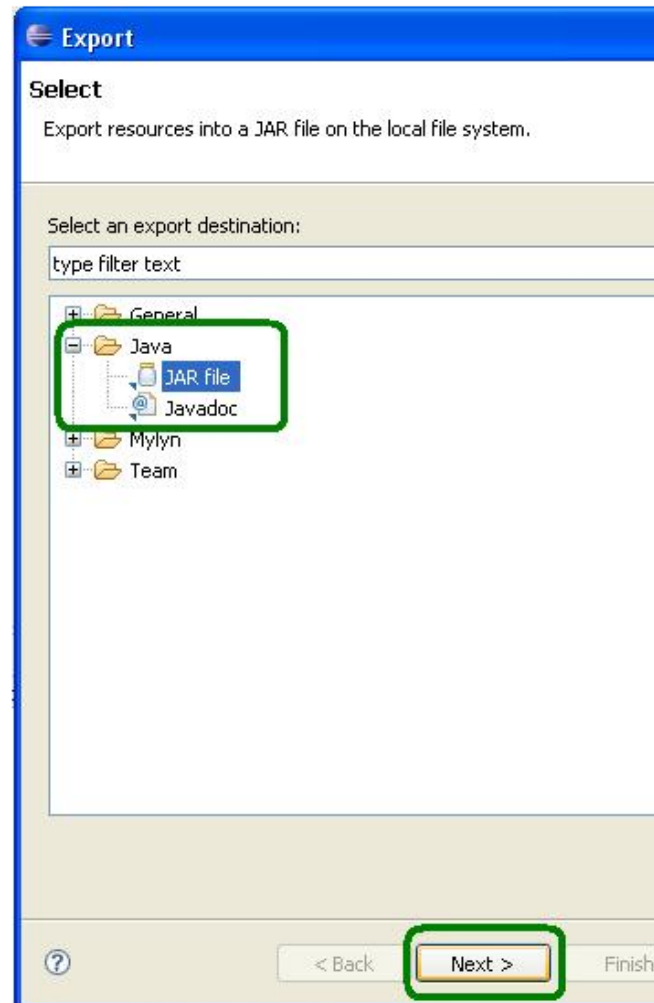
CRÉATION DE JAR EXÉCUTABLE

1. Sous Eclipse, cliquer droit sur le projet et choisissez l'option Export, comme ceci :



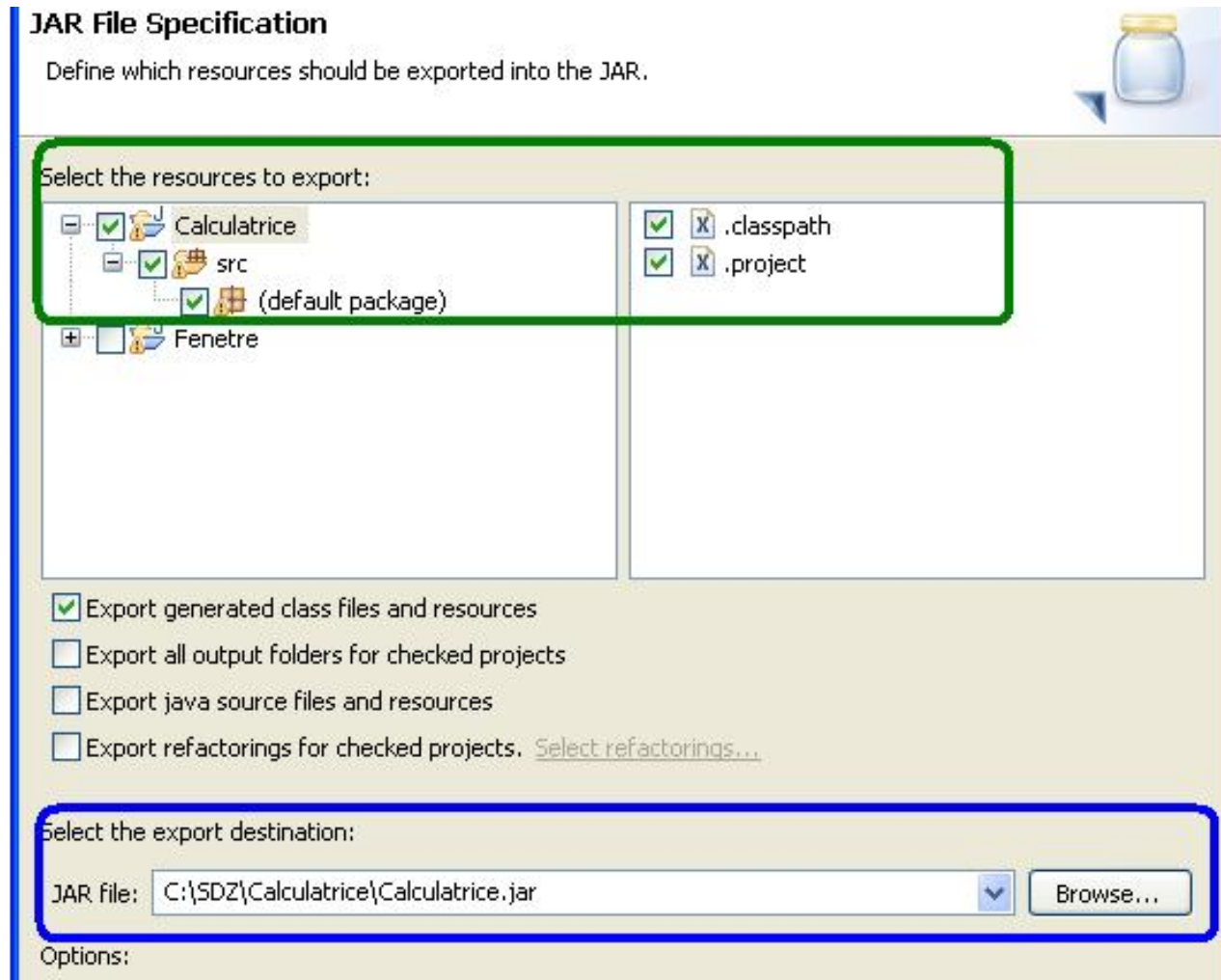
CRÉATION DE JAR EXÉCUTABLE

2. En suite, Eclipse vous demande quel type d'export vous souhaitez faire :
sélectionnez **JAR File** et cliquez sur **Next**.



CRÉATION DE JAR EXÉCUTABLE

- Maintenant, il demande quels fichiers vous voulez mettre dans votre archive.



CRÉATION DE JAR EXÉCUTABLE

Cliquez sur
Next


JAR Export

JAR Packaging Options
Define the options for the JAR export.

Select options for handling problems:

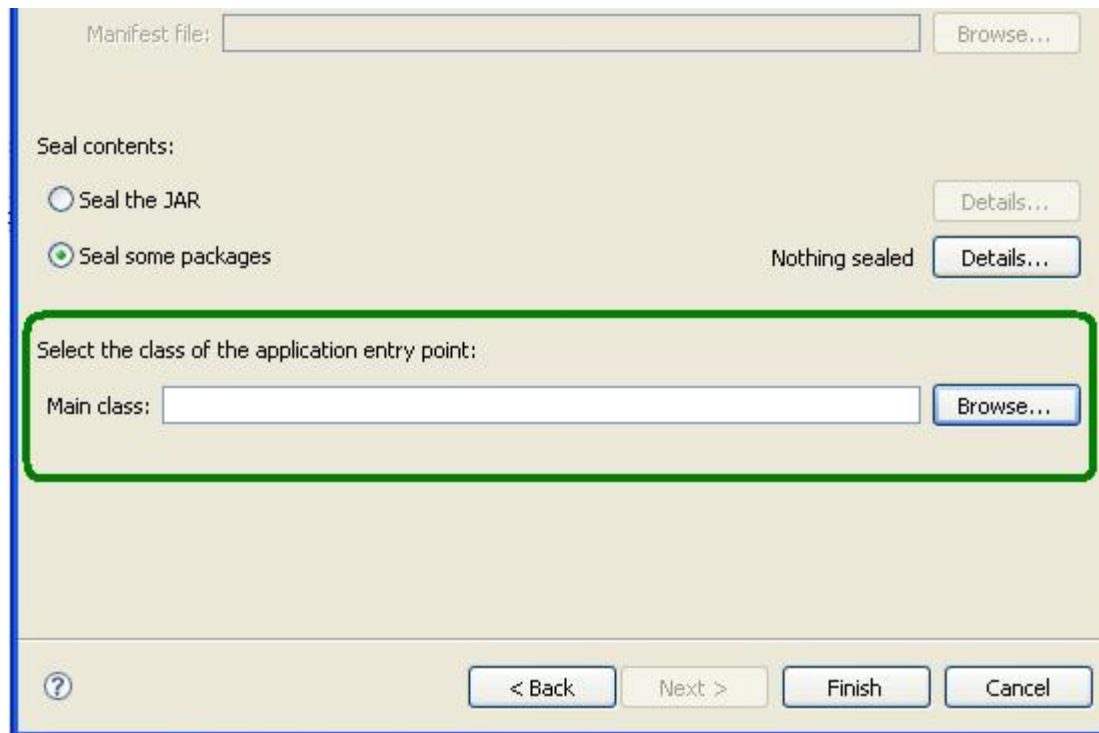
- ☒ Export class files with compile errors
- ☒ Export class files with compile warnings
- ☐ Create source folder structure
- ☒ Build projects if not built automatically
- ☐ Save the description of this JAR in the workspace

Description file:



CRÉATION DE JAR EXÉCUTABLE

- Il faut sélectionner la méthode **main** du programme en cliquant sur **Browse ...** en suite **Finish**:



CRÉATION DE JAR EXÉCUTABLE

- Une fois cette étape validée, vous pouvez voir qu'un fichier **.jar** a bien été généré dans le dossier spécifié ! Double cliquez sur lui pour lancer la calculatrice .

