Université de Montpellier
Programmation par objets en Java (HLIN 505)

Premières fenêtres graphiques
(Tutoriel + TP)

Ce tutoriel/TP a pour objectif de vous apprendre à :
• créer des fenêtres graphiques,
• insérer des composants dans d'autres composants,
• gérer la disposition des composants dans un composant.

Vous constaterez que certains composants réagissent déjà à des actions de l'utilisateur (par exemple, une fenêtre peut s'agrandir, se fermer etc…) ; nous verrons plus tard comment définir nos propres traitements d'événements.

La bibliothèque SWING

Nous allons utiliser la bibliothèque de classes graphiques **SWING** (dont les packages commencent par `javax.swing`). Cette bibliothèque remplace partiellement la bibliothèque graphique antérieure **AWT**. Le nom des classes SWING commence par **J** : par exemple `JFrame, JButton, JTextField`, qui remplacent les classes `Frame, Button et TextField` de AWT.

Il est fortement conseillé de **ne jamais mélanger** dans une même application des composants SWING et des composants AWT, **lorsque ceux-ci possèdent une nouvelle version en SWING** ; en effet, les composants des deux bibliothèques ont un fonctionnement différent et cohabitent donc mal. Notez que pour gérer les événements, nous utiliserons des classes des deux bibliothèques, car les classes d'événement SWING complètent celles de AWT (et ne les remplacent donc pas).

I. La classe JFrame pour gérer des fenêtres

La classe javax.swing.JFrame permet de gérer des fenêtres graphiques "indépendantes" (au sens où elles ne dépendent d'aucune autre fenêtre). On les appelle aussi "fenêtres cadres".

1. *Créez votre première fenêtre* sur le modèle ci-dessous :

```
import javax.swing.*;
public class Appli
{
 public static void main(String args[])
 {
  JFrame f = new JFrame();
  f.setTitle("Ma petite application");
  f.setSize(400,400) ; // par défaut la taille est (0,0)
  f.setLocation(100,100) ; // par défaut, coin supérieur gauche écran
  // ou bien setBounds(100,100,400,400);
  f.setVisible(true); // sinon la fenêtre n'est pas visible
 }
}
```

Étudiez le rôle de chacune des méthodes `setTitle, setSize, setLocation` (ou `setBounds` qui effectue setLocation+ setSize)*, en les mettant tour à tour en commentaire.*

Remarquez que vous pouvez déplacer la fenêtre, modifier sa taille, la fermer.

Attention : lorsque vous fermez la fenêtre, vous ne terminez pas l'application pour autant. Vous pouvez le constater avec la fenêtre console (au besoin sous Eclipse, affichez cette fenêtre par le menu Window/Show View/Console ; pour terminer l'application, il vous faut appuyer sur le bouton rouge). Si vous voulez que l'application termine en même temps que la fermeture de la fenêtre, ajoutez : `f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);`

2. On utilisera rarement la classe JFrame directement. On définira plutôt des classes de fenêtre **dérivées** de JFrame, ce qui permettra de leur ajouter différents composants et de spécialiser leur comportement.

Créez une classe dérivée de JFrame, sur le modèle ci-dessous :

```
class MaFrame extends JFrame   // classe publique ou pas : au choix
{
 // constructeur
 public MaFrame()
 {
     super(); // sinon, serait ajouté par le compilateur
     setTitle("Ma petite application");
     setSize(400,400) ;
     setLocation(100,100) ;
 }
}
public class Appli
{
 public static void main(String args[])
 {
  JFrame f = new MaFrame(); // ou : MaFrame f = new MaFrame();
  f.setVisible(true);
 }
}
```

Voici une variante de la classe Appli, où la création de fenêtre se fait dans le constructeur. La méthode main() se contente de créer une instance d'Appli.

```
public class Appli
{
 public Appli()
 {
   JFrame f = new MaFrame();
   f.setVisible(true);
 }
 public static void main(String args[])
 { Appli a = new Appli(); }  //ou simplement : { new Appli(); }
}
```

II. Tout programme graphique a aussi une fenêtre console

Une application graphique, comme toute application Java, possède toujours une fenêtre console. Fenêtres graphiques et fenêtre console peuvent interagir. Dans le petit programme ci-dessous, on crée une fenêtre graphique, et on change son titre par saisie dans la fenêtre console.

Modifiez votre programme précédent pour faire de même.

```
import java.util.Scanner;
import javax.swing.*;

public class GraphConsole extends JFrame
 {
 public GraphConsole()
 {
  setSize(200, 200);
  setTitle("Première fenêtre");
 }

 public static void main(String args[])
 {
  GraphConsole f = new GraphConsole();
  f.setVisible(true);
  while(true)
  {
   System.out.println("nouveau titre?");
```

```
   Scanner sc = new Scanner(System.in);
   String rep = sc.nextLine();
   f.setTitle(rep);
  }
 }
}
```

III. Ajout de composants graphiques

Certains composants graphiques peuvent contenir d'autres composants graphiques. Ces composants "**conteneurs**" dérivent de la classe `java.awt.Container`, qui introduit la méthode **`add`**`(Component c)` : cette méthode permet d'ajouter le composant passé en paramètre comme élément de l'objet receveur de la méthode.

Le programme ci-dessous définit une classe de fenêtre composée d'un bouton.

```
import java.awt.*;
import javax.swing.*;

public class FenBouton extends JFrame
{
 public FenBouton()
 {
  setSize(400, 400);
  setTitle("Fenêtre à un bouton");

  JButton bouton = new JButton("Hello");
  this.add(bouton); // équivalent à add(bouton, "Center"), voir plus loin
 }

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

Dans le programme précédent, remarquez que la référence "bouton" est une variable *locale* au constructeur. A la fin de l'exécution du constructeur, le bouton aura été ajouté comme *élément* de la fenêtre, par contre la référence "bouton" n'existera plus. En général, on a besoin d'accéder aux éléments graphiques d'une fenêtre au cours de la vie de la fenêtre : on ajoute donc des *attributs* à l'objet fenêtre qui vont permettre de désigner ses éléments graphiques. Créons donc un attribut correspondant au bouton :

```
public class FenBouton extends JFrame
{
 private JButton bouton;

 public FenBouton()
 {
  setSize(400, 400);
  setTitle("Fenêtre à un bouton");

  bouton = new JButton("Hello");
  this.add(bouton); // équivalent à add(bouton, "Center"), voir plus loin
 }

 public static void main(String args[]) {…}
}
```

Ajoutez de la même façon un bouton à votre fenêtre. Constatez que le bouton réagit lorsque vous l'actionnez (même s'il ne fait rien de plus pour l'instant).

Le bouton occupe tout l'espace disponible dans la fenêtre, ce qui n'est pas forcément ce qu'on souhaite. Qui en a décidé ainsi ? Le "gestionnaire de disposition" ("**layout manager**") de la fenêtre. Chaque composant "conteneur" possède en effet un layout manager qui décide de la disposition et de la taille de ces éléments. On ne décide donc pas de la position ni de la taille d'un composant graphique que l'on ajoute dans un autre, on se contente de l'ajouter (par la méthode add). On peut cependant changer de type de layout manager, chacun ayant sa propre façon de procéder.

Cette façon de faire permet une gestion dynamique de la disposition des composants : lorsque la fenêtre change de taille, le layout manager adapte la disposition de ses éléments.

Faites l'essai suivant : ajoutez dans le constructeur *de FenBouton la ligne ci-dessous qui change le type de layout manager de la fenêtre (on choisit un "FlowLayout") et constatez la différence d'apparence du bouton :*

```
setLayout(new FlowLayout());
```

Essayer d'ajouter deux boutons et voyez le résultat.

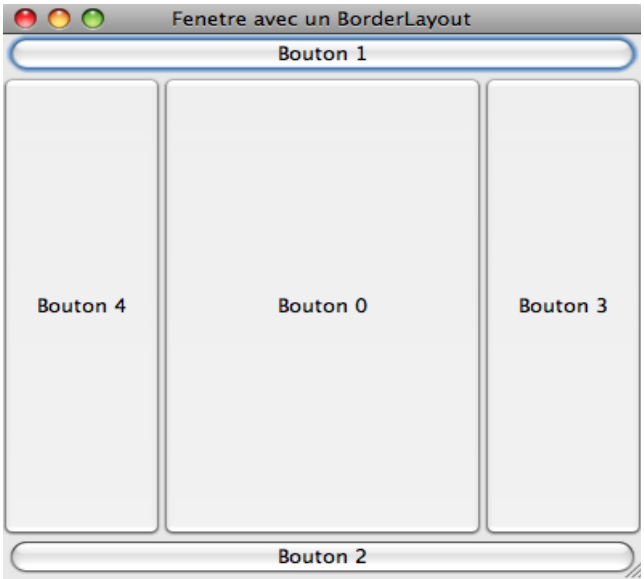
IV. Les gestionnaires de disposition (layout managers)

Une JFrame est par défaut gérée par un layout manager de type BorderLayout.

BorderLayout

Ce gestionnaire divise l'espace disponible en 5 zones : centre, nord, sud, est et ouest. Le centre occupe une place prépondérante. Par défaut, l'ajout se fait au centre. Si plusieurs éléments sont ajoutés dans la même zone, ils se superposent. Le programme ci-dessous vous en donne un exemple :

```
public class FenBorderLayout extends JFrame
{
 private JButton [] boutons; // tableau de 5 boutons

 public FenBorderLayout()
 {
  setSize(400, 400);
  setTitle("Fenêtre avec un BorderLayout");

  boutons = new JButton[5];
  for (int i = 0; i < boutons.length; i ++)
   boutons[i]=new JButton("Bouton "+i);
  add(boutons[0]); // au centre
  add(boutons[1],"North");// ou BorderLayout.NORTH
  add(boutons[2],"South");// ou BorderLayout.SOUTH
  add(boutons[3],"East"); // ou BorderLayout.EAST
  add(boutons[4],"West"); // ou BorderLayout.WEST
 }

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

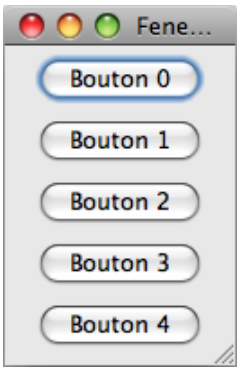


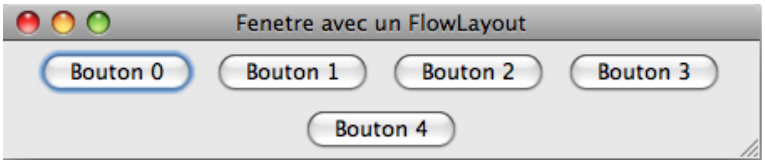
FlowLayout

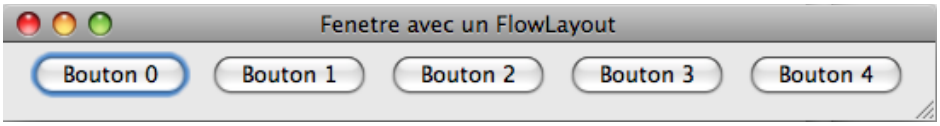
D'autres composants (comme les "panneaux", dont nous allons beaucoup nous servir par la suite) sont gérés par défaut par des FlowLayout. Le FlowLayout ajoute les composants les uns à la suite des autres (de gauche à droite, et de haut en bas), selon leur ordre d'ajout. Chaque élément a la taille minimum permettant de l'afficher correctement (un bouton a ainsi la taille minimum qui permet l'affichage correct de son libellé).

Dans le programme précédent, remplaçons le gestionnaire par un FlowLayout.

```
public class FenFlowLayout extends JFrame
{
 private JButton [] boutons;

 public FenFlowLayout()
 {
  setSize(400, 400);
  setTitle("Fenêtre avec un FlowLayout");

  boutons = new JButton[5];
  for (int i = 0; i < boutons.length; i ++)
    boutons[i]= new JButton("Bouton "+i);
  setLayout(new FlowLayout());
  for (int i = 0; i < boutons.length; i ++)
    add(boutons[i]);
 }

 public static void main(String args[]) {  ... }
}
```







GridLayout

Citons encore le GridLayout, qui dispose ses éléments selon une grille.

```
setLayout(new GridLayout(3,4));
// 3 lignes, 4 colonnes
```

V. Les panneaux (JPanel)

Un panneau est un composant graphique qui représente une surface rectangulaire, sans cadre. Il est très utile pour agencer des composants. **La disposition dans un panneau est gérée par défaut par un FlowLayout**. Prenons l'exemple ci-dessous :

La fenêtre contient **3 panneaux** : le premier de couleur rouge au Nord, le second de couleur bleu au centre et le troisième de couleur jaune à l'est. Le 2ème panneau contient une zone de saisie de texte, et le 3ème contient 3 boutons.

Lisez attentivement le programme ci-dessous, il contient un certain nombre de nouvelles méthodes. Remarquez en particulier qu'on peut indiquer des "dimensions préférées" pour un élément (méthode **setPreferredSize**): cette indication n'est prise en compte par le manager que si elle est compatible avec sa méthode d'agencement. Nous avons utilisé cette méthode pour agrandir les dimensions des panneaux rouge et jaune. *Remarquez aussi que nous n'avons pas donné de dimensions à la fenêtre.* Nous avons utilisé à la place la méthode **pack** qui demande à la fenêtre de prendre des dimensions englobant "tout juste" ses éléments. Cette méthode ne doit être appelée qu'après que tous les éléments ont été ajoutés (sinon certains ne seront pas visibles).

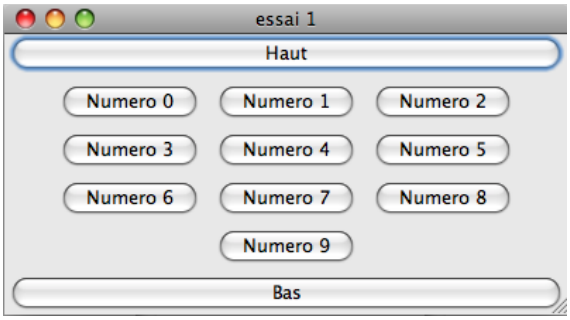
```
public class FenPanneaux extends JFrame
{
 protected JPanel pRouge, pBleu, pJaune;
 protected JButton b1,b2,b3;
 protected JTextField texte;

 public FenPanneaux ()
 {
  setTitle("Fenêtre à panneaux");
  pRouge = new JPanel();
  pRouge.setBackground(Color.red); // change la couleur de fond
  pBleu = new JPanel();
  pBleu.setBackground(Color.blue);
  pJaune = new JPanel();
  pJaune.setBackground(Color.yellow);
  add(pRouge, "North");
  add(pBleu, "Center"); // ou : add(pBleu);
  add(pJaune, "East");
  pRouge.setPreferredSize(new Dimension(400,100));
  pJaune.setPreferredSize(new Dimension(100,200));
  b1 = new JButton("b1"); pJaune.add(b1);
  b2 = new JButton("b2"); pJaune.add(b2);
```

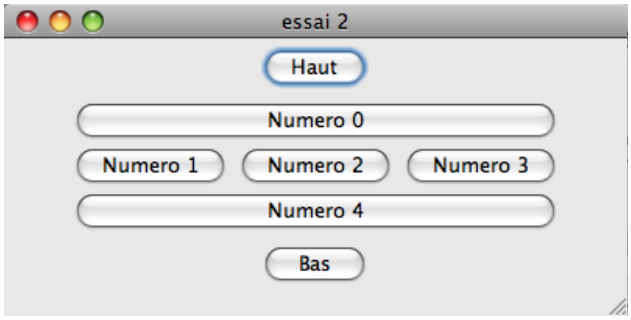
```
  b3 = new JButton("b3"); pJaune.add(b3);
  texte = new JTextField("Hello",20);
  pBleu.add(texte);

  pack();
 }

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

TP – Premières fenêtres graphiques

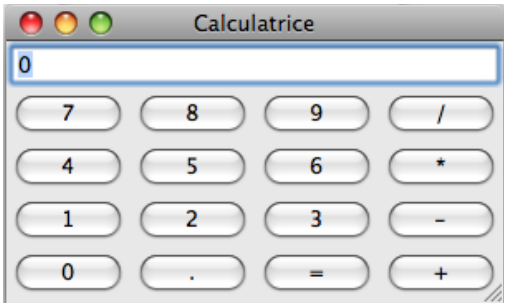
Programmez des classes de fenêtres permettant d'obtenir les visualisations ci-dessous.
Indications :
- pensez à utiliser des **panneaux**, et au besoin changez de Layout Manager.
- **utilisez la documentation de l'API du SDK Java**.

Exercice 1.



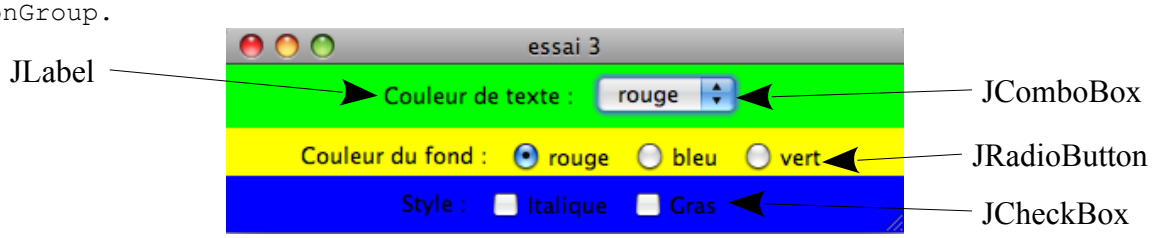


Exercice 2.



Exercice 3.

Vous utiliserez les classes `JLabel` (texte), `JComboBox` (liste déroulante), `JRadioButton` (bouton radio) et `JCheckBox` (cases à cocher). Afin de regrouper plusieurs boutons radio, vous utiliserez une instance de `ButtonGroup` (vous ajouterez les boutons radio au composant qui les contient et vous les ajouterez également à un `ButtonGroup`.



Exercice 4.

Vous utiliserez les classes `JMenuBar` (barre de menu), `JMenu` (menu vertical) `et JMenuItem` (entrée de menu). Un objet `JMenuBar` est ajouté à une `JFrame` grâce à la méthode `setJMenuBar`. Les `JMenu` sont ajoutés à une `JMenuBar` et les `JMenuItem` sont ajoutés à un `JMenu` par la méthode habituelle `"add"`.