TP1

Ce TP vous permettra d'apprendre à utiliser l'objet `JFrame`, présent dans le package `javax.swing`. Vous serez alors à même de créer une fenêtre, de définir sa taille, etc.

L'objet JFrame

Pour utiliser une fenêtre de type `JFrame`, vous devez l'instancier, comme ceci :

```
Import javax.swing.JFrame;
public class Test {
public static void main(String[] args){
JFrame fenetre = new JFrame();
  }
}
```

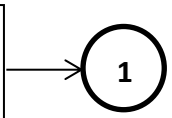
Lorsque vous exécutez ce code, vous n'obtenez rien, car par défaut, votre `JFrame` n'est pas visible. Vous devez donc lui dire « sois visible » on doit ajouter : **`fenetre.setVisible(true);`**

Pour obtenir une fenêtre plus conséquente, il faudrait donc : qu'elle soit plus grande ; qu'elle comporte un titre; qu'elle figure au centre de l'écran; que notre programme s'arrête réellement lorsqu'on clique sur la croix rouge, car, pour ceux qui ne l'auraient pas remarqué, le processus Eclipse tourne encore même après la fermeture de la fenêtre.

```
Import javax.swing.JFrame;

public class Test {
public static void main(String[] args){

JFrame fenetre = new JFrame();

    //Définit un titre pour notre fenêtre
fenetre.setTitle("Ma première fenêtre Java");
    //Définit sa taille : 400 pixels de large et 100 pixels de haut
fenetre.setSize(400, 100);
    //Nous demandons maintenant à notre objet de se positionner au centre
fenetre.setLocationRelativeTo(null);
    //Termine le processus lorsqu'on clique sur la croix rouge
fenetre.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
//Et enfin, la rendre visible
fenetre.setVisible(true);
  }
}
```

Afin de ne pas avoir à redéfinir les attributs à chaque fois, je pense qu'il serait utile que nous possédions notre propre objet. Comme ça, nous aurons notre propre classe !

```
Import javax.swing.JFrame;

public class Fenetre extends JFrame {
      public Fenetre(){
            this.setTitle("Ma première fenêtre Java");
            this.setSize(400, 500);
            this.setLocationRelativeTo(null);
            this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            this.setVisible(true);
  }
}
```



Dans votre `main`, la ligne de code suivante doit y figurer :**`Fenetre fen = new Fenetre();`**

L'objet JPanel

Maintenant, nous allons utiliser un `JPanel`, composant de type conteneur dont la vocation est d'accueillir d'autres objets de même type ou des objets de type composant (boutons, cases à cocher…).

1. Importer la classe `javax.swing.JPanel` dans notre classe héritée de `JFrame`.
2. Instancier un `JPanel` puis lui spécifier une couleur de fond pour mieux le distinguer.
3. Avertir notre `JFrame` que ce sera notre `JPanel` qui constituera son content pane.

```
importjava.awt.Color;
importjavax.swing.JFrame;
importjavax.swing.JPanel;

public class Fenetre extends JFrame {
public Fenetre(){
      1
//Instanciation d'un objet JPanel
JPanel pan = new JPanel();
    //Définition de sa couleur de fond
pan.setBackground(Color.ORANGE);
    //On prévient notre JFrame que notre JPanel sera son content pane
this.setContentPane(pan);
this.setVisible(true);
  }
}
```

Utiliser la classe JButton

Comme indiqué dans le titre, nous allons utiliser la classe `JButton` issue du package `javax.swing`.

Dans la classe `Fenetre`, nous allons créer une variable d'instance de type `JPanel` et une autre de type`JButton`. Faisons de `JPanel` le content pane de notre `Fenetre`, puis définissons le libellé (on parle aussi d'étiquette) de notre bouton et mettons-le sur ce qui nous sert de content pane (en l'occurrence, `JPanel`). Il ne nous reste plus qu'à ajouter ce bouton sur notre content pane grâce à la méthode `add()` de l'objet`JPanel`. Voici donc notre code :

```
import javax.swing.JButton;
import javax.swing.JFrame;
importjavax.swing.JPanel;

public class Fenetre extends JFrame{
  private JPanel pan = new JPanel();
private JButton bouton = new JButton("Mon bouton");
  public Fenetre(){
1
    //Ajout du bouton à notre content pane
pan.add(bouton);
    //this.setContentPane(pan);
this.setVisible(true);
  }
}
```

Maintenant, pour utiliser le content pane d'une `JFrame`, il faut appeler remplacer l'instruction `this.setContentPane(pan);`par : **`this.getContentPane().add(bouton);`**et supprimer tout ce qui concerne notre`JPanel`.

Positionner son composant : les layout managers

Il existe plusieurs sortes de *layout managers*, plus ou moins simples à utiliser, dont le rôle est de gérer la position des éléments sur la fenêtre. Tous ces layout managers se trouvent dans le package `java.awt`.

L'objet `BorderLayout`

Le premier objet que nous aborderons est le `BorderLayout`. Il est très pratique si vous voulez placer vos composants de façon simple.

```
Import java.awt.BorderLayout;
Import javax.swing.JButton;
Import javax.swing.JFrame;

public class Fenetre extends JFrame{
publicFenetre(){
1
    //On définit le layout à utiliser sur le content pane
this.setLayout(new BorderLayout());
    //On ajoute le bouton au content pane de la JFrame
this.getContentPane().add(new JButton("CENTER"), BorderLayout.CENTER);//Au centre
this.getContentPane().add(new JButton("NORTH"), BorderLayout.NORTH);//Au nord
this.getContentPane().add(new JButton("SOUTH"), BorderLayout.SOUTH);//Au sud
this.getContentPane().add(new JButton("WEST"), BorderLayout.WEST);//À l'ouest
this.getContentPane().add(new JButton("EAST"), BorderLayout.EAST);//À l'est
this.setVisible(true);
  }
}
```

L'objet `GridLayout`

Celui-ci permet d'ajouter des composants suivant une grille définie par un nombre de lignes et de colonnes. Les éléments sont disposés à partir de la case située en haut à gauche. Dès qu'une ligne est remplie, on passe à la suivante.

```
importjava.awt.GridLayout;
importjavax.swing.JButton;
importjavax.swing.JFrame;

public class Fenetre extends JFrame{
publicFenetre(){
1
    //On définit le layout à utiliser sur le content pane
    //Trois lignes sur deux colonnes
GridLayout gl = new GridLayout(3, 2);
this.setLayout(gl);
gl.setHgap(5); //Cinq pixels d'espace entre les colonnes (H comme Horizontal)
gl.setVgap(5); //Cinq pixels d'espace entre les lignes (V comme Vertical)
    //Ou en abrégé : GridLayoutgl = new GridLayout(3, 2, 5, 5);

    //On ajoute le bouton au content pane de la JFrame
this.getContentPane().add(new JButton("1"));
this.getContentPane().add(new JButton("2"));
this.getContentPane().add(new JButton("3"));
this.getContentPane().add(new JButton("4"));
this.getContentPane().add(new JButton("5"));
this.setVisible(true);
  }
}
```

L'objet `BoxLayout`

Grâce à lui, vous pourrez ranger vos composants à la suite soit sur une ligne, soit sur une colonne.

```
importjavax.swing.BoxLayout;
importjavax.swing.JButton;
importjavax.swing.JFrame;
importjavax.swing.JPanel;

public class Fenetre extends JFrame{

publicFenetre(){
      1
      JPanelp1 = new JPanel();
      //On définit le layout en lui indiquant qu'il travaillera en ligne
      p1.setLayout(new BoxLayout(p1, BoxLayout.LINE_AXIS));
      p1.add(new JButton("Bouton 1"));

      JPanelp2 = new JPanel();
          //Idem pour cette ligne
      p2.setLayout(new BoxLayout(p2, BoxLayout.LINE_AXIS));
      p2.add(new JButton("Bouton 2"));
      p2.add(new JButton("Bouton 3"));

      JPanelp3 = new JPanel();
          //Idem pour cette ligne
      p3.setLayout(new BoxLayout(p3, BoxLayout.LINE_AXIS));
      p3.add(new JButton("Bouton 4"));
      p3.add(new JButton("Bouton 5"));
      p3.add(new JButton("Bouton 6"));

      JPanelp4 = new JPanel();
          //On positionne maintenant ces trois lignes en colonne
      p4.setLayout(new BoxLayout(p4, BoxLayout.PAGE_AXIS));
      p4.add(p1);
      p4.add(p2);
      p4.add(p3);

      this.getContentPane().add(p4);
      this.setVisible(true);
  }
}
```

Ici nous avons créé trois `JPanel` contenant chacun un certain nombre de `JButton` rangés en ligne grâce à l'attribut `LINE_AXIS`. Ces trois conteneurs créés, nous les rangeons dans un quatrième où, cette fois, nous les agençons dans une colonne grâce à l'attribut `PAGE_AXIS`.