TP n°2 : création de fenêtres SWING, gestion d'événements, utilisation des *interfaces*

1. Une nouvelle classe « Fenetre »

- Créez une classe **Principal** dans laquelle vous écrivez une méthode **main** représentant le point d'entrée de votre programme.
- Créez une classe **Fenetre** et faites en sorte qu'elle soit une sous-classe de la classe java « JFrame » : ceci s'effectue en ajoutant **extends JFrame** lors de la déclaration de classe. La classe JFrame est un cadre graphique où l'on peut placer des fenêtres graphiques, des boutons, et prendre en compte des événements de la souris. Pour que cette classe soit trouvée par le compilateur, ajoutez en tête du fichier **import javax.swing.*;** Ceci permet d'inclure toutes les classes du paquetage (package en anglais) « javax.swing » qui est une bibliothèque de classes graphiques.
- Créez ensuite un **constructeur** à la classe Fenetre.

Pour créer la fenêtre graphique associée à l'objet et la présenter à l'écran, nous devons faire appel à trois méthodes attachées à la classe JFrame à l'intérieur de notre constructeur Fenetre :

- **super();** pour construire toute la partie JFrame de notre fenêtre (appel du constructeur de la classe parente, classe parente « JFrame » désignée par *super*)
- **this.setSize(250,200);** pour définir la taille de la zone JFrame (vous pouvez choisir d'autres valeurs que 250 et 200, mais ne dépassez pas la taille de l'écran !). *setSize* est une méthode de la classe JFrame.
- **this.setVisible(true);** pour faire apparaître la fenêtre graphique en premier plan à l'écran. *setVisible* est une méthode de la classe JFrame.

Remarque : le mot clé « **this** » fait référence à la classe dans laquelle on travaille. Ainsi, on applique la méthode setSize à this qui est un objet de type Fenetre. Puisque notre objet hérite de la classe JFrame, on peut lui appliquer la méthode setSize. L'utilisation de this est ici optionnelle, mais il est fortement recommandé de l'utiliser pour la lisibilité de votre code.

Compilez la nouvelle classe Fenetre, ainsi que la classe Principal permettant de créer un objet fenêtre. Exécutez le tout.

2. Modifions un peu la présentation...

En vous aidant du site regroupant les API java (la « javadoc »), trouvez dans la classe JFrame un moyen de mettre un titre à la fenêtre graphique créée. Trouvez aussi un moyen de faire en sorte que

la fenêtre se ferme lors du clic sur la croix en haut à droite de la fenêtre. Ajoutez toutes ces méthodes dans le constructeur de votre fenêtre.

3. Choisir la taille de la fenêtre en passant des paramètres au programme

Dans la classe « Principal », la fonction main peut recevoir des arguments, éventuellement les communiquer ensuite au constructeur de classe « Fenetre ».

a) affichage de tous les arguments reçus lors de l'exécution

Le paramètre args figurant dans main est un tableau de chaînes de caractères (les arguments). La variable length attachée à args donne le nombre d'arguments présents, et l'on peut afficher tous les arguments fournis à la fonction principale *main* de la façon suivante :

```
for (int i=0;i < args.length; i++)
      System.out.println(args[i]);
```

Essayez ainsi d'afficher tous les arguments qui suivent le lancement de votre programme.

b) Récupération de la hauteur et largeur de la fenêtre en arguments

Vérifier qu'il y a bien deux et seulement deux arguments fournis. Utilisez la méthode **java.lang.Integer.parseInt** avec en paramètre chacune des chaînes de caractères correspondantes aux arguments du programme principal. Modifiez le constructeur Fenetre pour lui passer en paramètre la largeur et la hauteur de la fenêtre à créer, fournies en arguments à l'exécution de Principal.

La méthode **parseInt** ci-dessus peut générer une exception (problème) si la chaîne de caractères n'est pas convertible en un entier. Essayez l'exécution avec un argument différent d'un entier et regardez le résultat...

Essayons d'attraper l'exception, et de la traiter localement : => englobez les deux instructions contenant parseInt (pouvant poser des difficultés) dans un bloc ***try*** : **try {** instructions java .... **}**

=> rattrapez les problèmes dus à cet essai avec ***catch*** :

```
try { ....
} catch ( NumberFormatException e) {
// instructions en cas de problème de conversion en entier
// par exemple : afficher « Eh ?! Donnez moi deux entiers !!! » et
fixer la largeur et la hauteur aux valeurs maximales
}
```

Essayez à nouveau votre programme.

4. Des boutons, des étiquettes, des menus,... *et des scoubidous*

Appelons *composant* l'un quelconque des éléments **JButton, JCheckbox, JLabel, JList** que l'on peut ajouter à notre fenêtre graphique. Pour utiliser l'un de ces *composants*, voici comment procéder :

a) Déclarez une ou plusieurs variables du composant souhaité en attribut de la classe fenêtre. Une variable est nécessaire pour chaque composant ajouté, car c'est elle qui mémorisera l'état de ce composant (exemple bouton appuyé ou relâché).

```
public class Fenetre extends JFrame {
     private Composant c1,c2,mon_composant;
     ....
```

Exemples :
```
private JButton bouton;
private JLabel texte;
```

b) Dans le constructeur de Fenetre, créez les objets associés à chaque composant en faisant appel au constructeur du *composant* avec les paramètres requis.

```
c2 = new Composant(...);
```

Exemples :
```
bouton = new JButton ("OK");
texte  = new JLabel ("Bonjour !");
```

c) Toujours dans le constructeur de Fenetre, récupérez l'objet (Container) dans lequel s'effectue l'affichage des composants graphiques, puis ajoutez les *composants* en précisant leurs emplacements dans le Container. La classe Container fait partie du package java.awt : n'oubliez donc pas de faire un **import java.awt.*;** avant la déclaration de votre classe.

```
Container c = this.getContentPane();
c.add(bouton, BorderLayout.WEST);
c.add(texte, BorderLayout.CENTER);
```

Voilà c'est tout ! Compilez et exécutez. Essayez de disposer deux JButton et un JLabel dans la fenêtre graphique.

5. Gérer un clic sur des boutons… ou comment être à l'écoute en JAVA…

Nous allons maintenant prendre en compte les événements éventuels lors d'un clic sur l'un des boutons de notre fenêtre graphique.

Le langage Java dispose d'*interfaces*, classes abstraites où toutes les variables et les méthodes sont déclarées mais non implémentées (réalisées). L'avantage d'une *classe interface* ou d'une *classe abstraite* est de fixer les interfaces d'échange entre objets sans contraindre le code exécutable. Lorsqu'on utilise une classe *interface* dans une classe C que l'on souhaite créer, **on doit implémenter toutes les méthodes déclarées dans l'*interface*.** On effectue ainsi une réalisation pour la classe C de

toutes les méthodes préconisées par l'interface; on est libre de mettre le code que l'on souhaite à l'intérieur de chaque méthode, l'essentiel étant de respecter les déclarations imposées par l'interface.

Ajoutez pour la classe Fenetre, à la suite de **extends JFrame implements ActionListener**. ActionListener est une interface (une classe spécifique) que vous pouvez retrouver dans la javadoc. Cette interface permet de récupérer l'événement de clic sur le bouton. Pour pouvoir compiler, rajoutez **import java.awt.event.*;** en tête de fichier. Compilez et lisez le message d'erreur trouvé, complétez votre classe fenêtre avec la méthode nécessaire (avec un corps vide pour l'instant). Enfin, dans le constructeur *Fenetre,* ajoutez l'instruction **bouton.addActionListener(this);** Cette instruction permet de préciser que notre objet s'appelant bouton est à l'écoute des évènement qui vont se produire sur lui-même. Le paramètre this désigne la classe courante et indique que la méthode qui va traiter l'événement est implémentée dans cette classe. Une fois la compilation réussie, réessayez l'exécution.

Ajouter une instruction d'affichage (System.out.println) dans la méthode permettant de générer une action lors du clic. Cela vous permettra de vérifier que tout fonctionne correctement.

Faites les mêmes opérations pour le second bouton de votre Fenetre. Vous remarquerez qu'il n'y a qu'une seule méthode permettant de traiter l'événement clic. Pour différencier le bouton1 du bouton2, il faut procéder de la manière suivante dans votre méthode permettant de traiter l'évènement.

```
if (e.getSource() == bouton1) {
   System.out.println("c'est le bouton 1");
}
if (e.getSource() == bouton2) {
   System.out.println ("c'est le bouton 2");
}
```

Faites en sorte que le clic modifie le texte du JLabel de votre Fenetre. S'il vous reste du temps, trouvez un moyen pour changer la couleur de fond du JLabel (un clic sur le bouton1 affiche le JLabel en rouge et un clic sur le bouton2 l'affiche en vert par exemple). *Indication* : faites attention au fait qu'il faille rendre votre JLabel opaque grâce à la méthode **label.setOpaque(true);** pour pouvoir en changer la couleur de fond (cf Javadoc).