Menus, toolbar et modèle

Exercice 1 - Les Menus

Dans une nouvelle fenêtre `JFrame`, nous allons ajouter des menus.

1 Créer une barre de menu (`JMenubar`).
2 Ajouter un menu File (`JMenu`).
3 Ajouter deux items Open et Save (des `JMenuItem`) au menu File (à l'aide de la méthode `add()`).
4 Puis ajouter le menu File à la barre de menu.
5 Enfin, ajouter la barre de menu à la fenêtre (`setJMenuBar()`).

Dans un second temps, ajouter un autre menu contenant des boutons à cocher (`JCheckBoxMenuItem`). Puis grouper-les (souvenez-vous des `ButtonGroup`).

Exercice 2 - La barre d'outils

Dans une nouvelle fenêtre, nous allons ajouter une barre d'outil. Pour cela, nous allons auparavant créer les boutons contenant des images pour mettre ceux-ci à l'intérieur de la barre d'outils.

- Créer une barre de d'outil (`JToolBar`), créer trois boutons Cut, Copy et Paste et ajouter les trois boutons à la barre d'outils, elle-même placée au nord (avec un `add(toolbar, BorderLayout.NORTH)`).
- Associer (`setIcon()`) une image (`ImageIcon`) à chaque bouton.
  -  Couper
  -  Copier
  -  Coller

  (`http://developer.java.sun.com/developer/techDocs/hi/repository/`)

Dans cet exercice, les boutons n'effectueront aucune action.

Exercice 3 - Modèle, vue, contrôleur

Réutiliser le programme précédent et ajouter au centre une liste contenant les fichiers du répertoire courant.
Dans un premier temps, afficher la liste des fichiers dans une `JList`. Pour cela, utiliser le constructeur `JList(Object[])`
(La méthode `listFiles()` permet d'obtenir l'ensemble des fichiers d'un répertoire et `new File(".")` correspond au répertoire courant).
Il existe une autre façon de faire en utilisant un modèle. Nous allons pour cela, implémenter le modèle de liste (`ListModel`) pour que celui-ci renvoie les fichiers contenus dans un répertoire.

- Écrire une classe `MyListModel` héritant de `AbstractListModel`. Cette classe devra prendre en paramètre un repertoire de type `File` et implémenter les méthodes `getElementAt()` et `getSize()`.

```
class MyListModel  extends AbstractListModel {
  public MyListModel(File file) {
    ... // a implémenter
  }
  public Object getElementAt(int index) {
    ... // a implémenter
  }
  public int getSize() {
    ... // a implémenter
  }
}
```

- Utiliser le modèle ainsi défini pour construire une `JList` (regarder les constructeurs).
- Ajouter un bouton ("Reload") qui recharge la liste pour voir si des fichiers ont changé.
- Comment faire pour afficher dans une autre `JList` uniquement les 10 premiers fichiers du répertoire courant.

Exercice 4 - À la maison ...

On cherche à afficher la liste des arguments de la ligne de commande dans une `JList`.
De plus, on cherche à ajouter une case à cocher permettant de trier l'intérieur de la `JList` par ordre croissant ou décroissant (on utilisera pour cela les méthodes `Arrays.sort()` et `Collections.reverseOrder()` du paquetage `java.util`).
Créer pour cela une classe implémentant le modèle de liste (`ListModel`) en héritant de `AbstractListModel`, en redéfinissant les méthodes `getElementAt()` et `getSize()` puis en rajoutant une méthode `sort(boolean order)`. Utiliser le constructeur `JList(ListModel)` pour créer une liste ayant pour modèle la classe créée ci-dessus.
Enfin faire en sorte qu'un appel à la méthode `sort()` demande à la vue (la `JList`) de se redessiner.