Examen (2 heures, avec document)

Préambule : Répondre de manière concise et précise aux questions. Ne pas mettre de commentaires de documentation sauf s'ils sont nécessaires à la compréhension.
Barème indicatif : Exercice 1 : 5 points, exercice 2 : 9 points, exercice 3 : 6 points.

L'objectif est de créer une petite application qui démontre quelques possibilités de l'API Swing de Java pour la construction d'interfaces graphiques. En particulier, on veut voir comment sont agencés des composants graphiques en fonction du gestionnaire de placement utilisé. La figure 1 présente l'interface homme/machine (IHM) de l'application à son lancement (gauche) ou après l'avoir manipulée (droite). Voyons de quoi elle est composée et comment elle fonctionne.

FIGURE 1 – IHM de l'application à son lancement (gauche) et après utilisation (droite)

L'application est composée de trois parties : la *partie centrale* contient un JPanel sur lequel on peut définir un gestionnaire de placement en utilisant l'un des boutons de la *partie inférieure* et ajouter des composants élémentaires en utilisant la *partie supérieure*.

Dans la partie inférieure, chaque bouton correspond à un gestionnaire de placement (BorderLayout, FlowLayout et GridLayout). Quand on clique, sur un de ces boutons, le gestionnaire de placement correspondant est créé et est appliqué au JPanel central.

Dans la partie supérieure, plusieurs boutons (Area, Bouton, Étiquette et Saisie) permettent de créer des composants graphiques élémentaires (JTextArea, JButton, JLabel et JTextField). Quand on clique sur l'un de ces boutons, le composant correspondant est créé et ajouté à la partie centrale de la fenêtre qui contient un JPanel. Pour ce faire, on utilise les deux zones de saisies juste en dessous des boutons. La zone nommée "texte" est utilisée pour construire le composant et deviendra le texte du bouton, de l'étiquette, etc. Celle nommée "position" est utilisée pour ajouter le composant au JPanel du centre de la fenêtre. Donner un tel paramètre sous la forme d'une chaîne de caractères est toujours possible mais le gestionnaire de placement ne saura pas forcément l'exploiter : par exemple FlowLayout et GridLayout ne l'utiliseront pas, BorderLayout ne saura le traiter que s'il correspond à North, South, East, West ou Center.

FIGURE 2 – IHM après avoir cliqué sur Flow (gauche) puis Grid (droite)

Listing 1 – Exemple de fichier texte qui décrit les composants graphiques

```
1 Bouton: javax.swing.JButton
2 Etiquette: javax.swing.JLabel
3 Saisie: javax.swing.JTextField
4 Area: javax.swing.JTextArea
```

Pour arriver à l'état de l'IHM présenté sur la droite de la figure 1, l'utilisateur peut effectuer les actions suivantes :

1. cliquer sur "Border" : le JPanel central est donc équipé d'un BoderLayout,
2. cliquer sur "Étiquette" après avoir renseigné `"une étiquette"` pour texte et `"West"` pour position : un JLabel est ajouté à l'ouest du JPanel central,
3. cliquer sur "Saisie" après avoir renseigné `"une zone de saisie"` pour texte et `"South"` pour position : un JTextField est ajouté au sud du JPanel central.
4. cliquer sur "Aréa" après avoir renseigné `"une zone de saisie\nà plusieurs lignes"` pour texte et `"Center"` pour position : un JTextArea est ajouté au centre du JPanel central,
5. cliquer sur "Bouton" après avoir renseigné `"un bouton"` pour texte et `"North"` pour position : un JButton est ajouté au nord du JPanel central,
6. renseigner le champ texte avec `"Essai"`.

Si on clique ensuite sur le bouton Flow, on obtient l'IHM sur la gauche de la figure 2 et celle de droite si on clique sur Grid.

On veut pouvoir changer les composants graphiques et les gestionnaires de placement proposés par cette application sans avoir à l'éditer et la recompiler. Ainsi, les composants et les gestionnaires de placement sont décrits dans des fichiers textes passés en argument de l'application. Ces deux fichiers ont une structure identique : chaque ligne contient le texte à afficher sur le bouton et le nom de la classe correspondante séparés par le symbole `":"`. Le listing 1 est le fichier texte qui décrit les composants graphiques, le listing 2 celui qui décrit les gestionnaires de placement. Notons que l'ordre dans lequel l'application affiche les boutons n'a pas d'importance. On peut considérer qu'elle les affiche dans l'ordre alphabétique des noms des boutons.

Listing 2 – Exemple de fichier texte qui décrit les gestionnaires de placement

```
Flow: java.awt.FlowLayout
Grid: java.awt.GridLayout
Border: java.awt.BorderLayout
```

Listing 3 – L'interface AnalyseurFichier

```
import java.util.Map;
public interface AnalyseurFichier {
    /** Analyser le contenu d'un fichier.
     * @param nomFichier le nom du fichier à analyser
     * @return les informations du fichier.
     */
    public Map<String, Class> analyseFichier(String nomFichier);
}
```

Exercice 1 : Traiter le fichier texte

Dans ce premier exercice, nous nous intéressons au traitement des fichiers qui contiennent les composants graphiques et les gestionnaires de placement à utiliser. Ces deux types de fichier ont la même structure. On peut donc les traiter de manière uniforme. L'interface du listing 3 spécifie la méthode qui analyse le fichier. Elle prend en paramètre le nom du fichier et retourne un tableau associatif (ou dictionnaire) dont la clé est le texte à afficher dans le bouton et la valeur la classe correspondante (de type Class).

1.1 Écrire une classe AnalyseurFichierTexte qui réalise cette interface et décode le fichier texte. Une ligne contenant une erreur sera ignorée (classe inexistante ou texte déjà utilisé par exemple).

Pour analyser une ligne du fichier, on pourra utiliser `split("\\s*:\\s*")` qui, appliquée à une String, retourne un tableau de chaînes de caractères correspondant au découpage de la chaîne initiale suivant l'expression régulière en paramètre de split.

Par exemple, `"abc: j.l.S".split("\\s*:\\s*")` donnera un tableau de deux chaînes, la première étant `"abc"` et la seconde `"j.l.S"`.

Exercice 2 : IHM en Swing

Le squelette de l'application est donné au listing 5.

2.1 Indiquer les composants à utiliser et comment les agencer (dessiner l'arbre des composants).

2.2 Expliquer de manière succincte comment rendre actif l'un des boutons de l'IHM.

2.3 Compléter le code du listing 5 pour rendre l'application fonctionnelle. On utilisera toujours le constructeur qui prend en paramètre une String pour créer un composant graphique et le constructeur sans paramètre pour créer un gestionnaire de placement.

Exercice 3 : Utilisation d'un fichier XML

En plus des fichiers textes, on veut utiliser des fichiers XML respectant la DTD du listing 4.

3.1 Indiquer la signification de "bien formé" et "valide" pour un fichier/document XML.

3.2 Donner le fichier XML équivalent du fichier du listing 2.

Listing 4 – classes.dtd, la DTD pour décrire les composants ou les gestionnaires de placement

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!ELEMENT classes (classe*)>
<!ELEMENT classe (#PCDATA)>
<!ATTLIST classe texte CDATA #REQUIRED>
```

Listing 5 – Le squelette de l'application

```
import javax.swing.*;
import java.awt.*;
import java.util.*;
public class DemoSwing {
    final JFrame fenetre = new JFrame("démo swing");
    final JPanel demo = new JPanel();       // Le JPanel du centre
    final JTextField texte = new JTextField("Essai");
    final JTextField position = new JTextField("North");

    public DemoSwing(Map<String, Class> composants, Map<String, Class> layouts) {
        Container c = fenetre.getContentPane();

        fenetre.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        fenetre.setSize(500,300);
        fenetre.setVisible(true);
    }

    public static void main(String[] args) {
        assert args.length == 2;
        AnalyseurFichier analyseur = new AnalyseurFichierTexte();
        new DemoSwing(analyseur.analyseFichier(args[0]),
                      analyseur.analyseFichier(args[1]));
    }
}
```

3.3 Écrire une méthode qui retourne un document JDom respectant la DTD du listing 4 à partir d'un tableau associatif obtenu comme résultat de la méthode AnalyseurFichier.analyse. Sa signature est la suivante :

```
public Document toXML(Map<String, Class> elements);
```

Le paramètre est le résultat de l'exercice 1.

3.4 On pourrait écrire une nouvelle réalisation de `AnalyseurFichier` pour lire un fichier XML. Indiquer les possibilités pour traiter en Java un fichier XML. On donnera en quelques phrases leur principe. Aucun code n'est demandé.