

Université Paris 7
Master 1 d'Informatique, Interfaces graphiques.
19 juin 2009 Durée : 2h

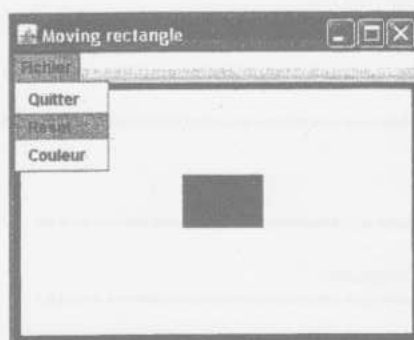
Documents manuscrits, notes de cours, notes de TD/TP autorisés. Livres interdits.

Le sujet comporte 4 pages.

Votre code doit être écrit de façon lisible, avec des indentations et des accolades appropriées permettant de voir la fin de blocs de code (fin de boucles, etc.).

Rappel : L'interface `MouseListener` possède les méthodes suivantes : `mouseClicked`, `mouseEntered`, `mouseExited`, `mousePressed`, `mouseReleased`. L'interface `MouseMotionListener` possède les méthodes : `mouseDragged`, `mouseMoved`. Toutes ces méthodes ont `MouseEvent` pour l'argument. Les méthodes `int getX()` et `int getY()` de l'objet `MouseEvent` permettent de trouver la position du curseur.

Exercice 1 L'application ci-dessous



permet de déplacer un rectangle par un geste de l'utilisateur.

Quand on appuie sur le bouton de la souris à l'intérieur du rectangle on peut traîner (drag) le rectangle dans le `JComponent` qui constitue la composante principale de l'application. Quand on déplace le rectangle la position relative du curseur à l'intérieur du rectangle reste inchangée.

Le item "Quitter" du menu termine l'exécution de l'application.

Le item "Reset" du menu met le rectangle sur la position initiale (on suppose que c'est la position (0,0) dans le `JComponent` `MovingRectangle` (la composante principale de l'application).

Le item "Couleur" lance `JColorChooser` permettant de changer la couleur du rectangle.

Le programme ci-dessus implémente le squelette de cette application. Il faut remplir les trous indiqués pour obtenir le comportement recherché.

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

public class MovingRectangle extends JComponent{
    private int larg=60,haut=40;
    private Rectangle r=new Rectangle(0,0,larg,haut);

    // (dx, dy) le vecteur entre la position de la souris et
    // le sommet gauche/haut du rectangle au moment
    // ou on appuie sur le bouton de la souris
    // a l'interieur du rectangle
    private int dx, dy;

    //true si on appuie sur le bouton de la souris a l'interier
    //du rectangle
    private boolean pressedInside = false;

    private Color color=Color.RED;

    public static void main(final String[] args) {
        SwingUtilities.invokeLater( new Runnable(){
            public void run(){
                new MovingRectangle(300,200);
            }
        })
    }
}
```

```

    } });
}

private MovingRectangle(int w, int h){
    setMinimumSize(new Dimension(w,h));
    setPreferredSize(getMinimumSize());

    JFrame frame = new JFrame("Moving rectangle");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JPanel panel = new JPanel();
    panel.setLayout(new BorderLayout());

    panel.setBorder( BorderFactory.createLineBorder(Color.BLACK,5) );
    frame.setContentPane(panel);

    panel.add(this, BorderLayout.CENTER);
    Dimension d = new Dimension(w+10,h+10);
    frame.setSize(w+10,h+50);

    /*****
     * trou 1 a remplir
     *****/
} //fin constructeur

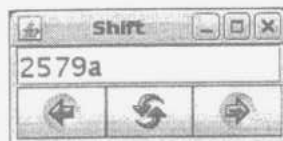
/*****
 * trou 2 a remplir
 * D'autres methodes comme paintComponent
 *****/
}

```

Indications : La classe `java.awt.Rectangle` possède les champs publiques de type `int` : `x`, `y` pour la position du sommet gauche/haut du rectangle et `int height`, `width` pour son largeur et hauteur. Cette classe possède les méthodes `void setLocation(int,int)` qui permet de changer la position du rectangle et `boolean contains(int,int)` qui retourne `true` si le point dont les coordonnées on passe en paramètre est à l'intérieur du rectangle.

L'objet `Graphics` possède les méthodes `void setColor(Color)` qui permet de changer la couleur utilisée pour dessiner et la méthode `fillRect(x,y,largeur,hauteur)` qui dessine le rectangle avec le sommet gauche/haut dans `(x,y)` en utilisant la couleur spécifiée auparavant avec `setColor`.

Exercice 2 L'interface du programme ci-dessous



est constituée d'une zone de texte (`JTextField`) et de trois boutons (`JButton`). Les boutons gauche et droite permutent circulairement le texte d'un caractère vers la gauche ou la droite (la permutation gauche consiste à déplacer le premier caractère à la fin et la permutation droite est l'inverse). Le bouton du milieu inverse l'ordre de caractère de la chaîne, par exemple la chaîne "1234" est transformée en "4321".

Chacun des trois boutons est contrôlé par un écouteur (listener) d'une classe *anonyme*. Écrire le programme pour obtenir le résultat recherché, en complétant le code ci-dessous.

```

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

class Shift extends JFrame {
    public Shift() {
        setTitle("Shift");
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        Container contentPane = getContentPane();

```

```

// Icons
ImageIcon leftIcon = new ImageIcon("Images/" + "left.png");
ImageIcon rightIcon = new ImageIcon("Images/" + "right.png");
ImageIcon revIcon = new ImageIcon("Images/" + "reverse.png");

// Zone de texte
final JTextField textField = new JTextField(8);

// Changement de fonte de la zone de texte
Font font = textField.getFont().deriveFont(18.0f);
textField.setFont(font);

/* A COMPLETER */
}

/* méthode pour effectuer le decalage gauche */
public static String decalageGauche(String s){
    StringBuffer buffer = new StringBuffer(s);
    if (buffer.length() > 0) {
        char c = buffer.charAt(0);
        buffer.delete(0, 1);
        buffer.append(c);
        return buffer.toString();
    }
    return s;
}

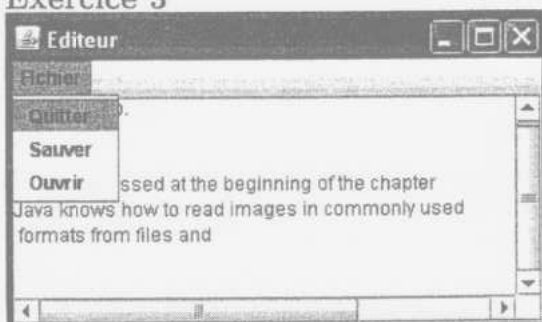
/* méthode pour effectuer le decalage droite */
public static String decalageDroite(String s){
    int i;
    StringBuffer buffer = new StringBuffer(s);
    if ( (i = buffer.length()) > 0) {
        char c = buffer.charAt(i-1);
        buffer.delete(i-1, i);
        buffer.insert(0, c);
        return buffer.toString();
    }
    return s;
}

/* méthode pour inverser un String */
public static String inverser(String s){
    StringBuffer buffer = new StringBuffer(s);
    buffer.reverse();
    return buffer.toString();
}

public static void main(String [] args)
{
    SwingUtilities.invokeLater( new Runnable(){
        public void run(){
            Shift view = new Shift();
            view.pack();
            view.setVisible(true);
        }
    });
}
}

```

Exercice 3



L'interface du programme ci-dessus est constitué d'un d'un JTextArea où l'utilisateur peut taper un texte. JTextArea se trouve à l'intérieur d'un JScrollPane. Le menu est composé d'un seul élément JMenu intitulé "Fichier". Ce menu contient trois JMenuItem :

- "Quitter" termine l'exécution de l'application. Le choix de ce item doit lancer un dialogue demandant la confirmation de la part de l'utilisateur s'il veut effectivement terminer l'exécution.
- "Sauver" - le choix de ce JMenuItem provoque le lancement d'un JFileChooser qui permet à l'utilisateur de choisir le fichier dans lequel on sauvegardera le texte affiché dans JTextArea.
- "Ouvrir" - le choix de ce JMenuItem lance un JFileChooser qui permet à l'utilisateur de choisir un fichier. Le contenu de ce fichier sera lu et mis dans JTextArea et affiché.

Complétez le code ci-dessous pour obtenir le comportement décrit. Pour faciliter votre travail notez que le code fourni contient déjà deux méthodes auxiliaires :

- la méthode écrire permet d'écrire le contenu de JTextArea dans un fichier, si on connaît déjà l'objet File approprié,
- la méthode lire qui lit un fichier texte et retourne le contenu de fichier comme un objet String.

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

class Editeur extends JFrame {
    final JTextArea area = new JTextArea(8,40);

    private Editeur() {
        setTitle("Editeur");
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        setPreferredSize(new Dimension(300,200));

        /*****
         * le code a completer
         *****/

    }

    /*****
     * la methode écrire écrit le texte écrit dans JTextArea
     * dans le fichier passe par parametre
     *****/
    public void écrire(File file) throws IOException{
        PrintWriter writer = new PrintWriter(file);
        String s = area.getText();
        writer.print(s);
        writer.close();
    }

    /*****
     * la methode lire lit le texte a partir d'un fichier
     * et le retourne comme un String
     *****/
    public String lire(File file) throws IOException{
        BufferedReader reader = new BufferedReader(new FileReader(file));
        String result="";
        for(;;){
            String s = reader.readLine();
            if( s == null )
                break;
            result += s + "\n";
        }
        reader.close();
        return result;
    }

    public static void main(String [] args)
    {
        SwingUtilities.invokeLater( new Runnable(){
            public void run(){
                Editeur view = new Editeur();
                view.pack();
                view.setVisible(true);
            }
        });
    }
}
```