Université Paris 7
Master 1 d'Informatique, Interfaces graphiques.
16 mai 2009 Durée : 1h45

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 dessiner un rectangle :
L'action de dessiner se déroule en trois étapes :

(1) Le dessin du rectangle commence quand on appuie sur un bouton de la souris. Les coordonnées de la position de la souris en ce moment, appelons les `(xa,ya)`, donnent le premier sommet du rectangle. Ces coordonnées ne changent plus jusqu'à ce que le dessin soit terminé.

(2) Le coordonnées du deuxième sommet du rectangle, appelons les `(xb,yb)`, sont données par la position courante de la souris. Ce sommet change quand on déplace la souris avec le bouton enfoncé (drag). Le rectangle doit être redessiné à nouveau chaque fois quand la souris est déplacée et tant que le dessin n'est pas terminé, c'est-à-dire le rectangle dessiné doit être modifié en temps réel suivant le mouvement de la souris.

(3) Le dessin est fini au moment quand on relâche le bouton de la souris, la position `(xb,yb)` du deuxième sommet devient fixe.

Si on commence le dessin d'un nouveau rectangle l'ancien rectangle peut disparaître (on ne demande pas de garder tous les rectangles, il suffit que seulement le dernier rectangle soit affiché à un moment donné).
Le menu "Fichier" contient deux items : "Quitter" pour terminer l'exécution de l'application et "Effacer" qui efface le rectangle.
Le programme ci-dessus implémente le squelette de cette application. Il faut remplir les trous indiqués pour obtenir le comportement recherché.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class Rect extends JComponent  {
    private int xa, ya;
    private int xb, yb;
    private boolean inside;//true si mousePressed
    //a l'interieur de Rect
    private boolean dessiner=true;//dessiner ou effacer

    public Rect(int l, int h)  {
      setPreferredSize(new Dimension(l,h));
```

```
        /*****************************************
         *  TROU 1: installer les ecouteurs
         *****************************************/

        JFrame view = new JFrame();
        view.setTitle("Rectangle");
        view.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        view.setContentPane(this);


        /*****************************************
         * TROU 2: installer le menu
         *****************************************/

        view.pack();
        view.setVisible(true);
    }//fin constructeur

    /*****************************************
     * TROU 3: d'autres methodes necessaires,
     * sans doute au moins painComponent.
     *****************************************/

    public static void main(String [] args)
    {
      SwingUtilities.invokeLater( new Runnable(){
                public void run(){
                Rect r = new Rect(200,200);
                }
            });
    }
}
```

Indication : Pour dessiner un rectangle on utilise la méthode

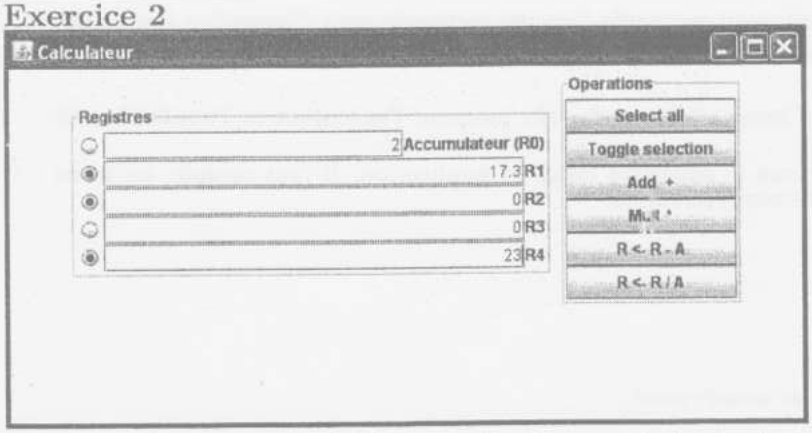
```
public void drawRect(int x, int y, int largeur, int hauteur)
```

de la classe `Graphics`, où `(x,y)` les coordonnées du sommet gauche/haut du rectangle. Dans notre cas, si nous avons deux sommets avec les coordonnées `(xa,ya)` et `(xb,yb)` respectivement alors `x = minimum(xa,xb)`, `y = minimum(ya,yb)`, `largeur` c'est la valeur absolue de `(xa-xb)`, `hauteur` c'est la valeur absolue de `(ya-yb)`.
Pour effacer on peut utiliser

```
public void clearRect(int x, int y, int largeur, int hauteur)
```

de la classe `Graphics`, où `(x,y)` les coordonnées du sommet gauche/haut du rectangle. Cette méthode met la couleur du fond sur tout le rectangle désigné.

Exercice 2



L'interface du programme ci-dessus représente une calculatrice dont le nombre de registres peut être spécifié au lancement du programme.
Le registre R0, appelé accumulateur, joue un rôle spécial, tous les autres registres sont des registres généraux. Les registres sont affiché comme des `JTextFields`. A gauche de chaque registre il y

a un `JRadioButton`, si ce bouton est sélectionné le registre participe à l'opération effectuée par l'utilisateur (une opération peut impliquer plusieurs registres, description détaillée suit ci-dessous). L'utilisateur choisit l'opération à exécuter avec les boutons `JButton` situés dans le panneau droit. Les opérations :

- `Select all` met tous les `JRadioButtons` dans l'état "sélectionné".
- `Toggle selection` : bascule l'état de chaque `JRadioButton` à l'état inverse (sélectionné $\leftrightarrow$ de-sélectionné).
- `Add *` fait la somme de valeurs de tous les registres sélectionnés et met le résultat dans l'accumulateur.
- `Mult *` calcule le produit de valeurs de tous les registres sélectionnés et met le résultat dans l'accumulateur.
- `R <- R - A` – pour chaque registre $R[i]$, $i = 1, 2, ...$, sélectionné on calcule la différence $R[i] - A$ où A c'est la valeur dans l'accumulateur et on remet le résultat dans $R[i]$. Cela ne concerne que les registres différents de l'accumulateur qui ne changera pas sa valeur même s'il est sélectionné. Autrement, on fait $R[i] \leftarrow R[i] - A$ pour les registres sélectionné (sauf accumulateur).
- `R <- R / A` – pour chaque registre $R[i]$, $i = 1, 2, ...$, sélectionné on calcule le quotient $R[i]/A$ où A c'est la valeur dans l'accumulateur et on remet le résultat dans $R[i]$. Cela ne concerne que les registres différents de l'accumulateur qui ne changera pas sa valeur même s'il est sélectionné. Autrement, on fait $R[i] \leftarrow R[i]/A$ pour les registres sélectionnés (sauf accumulateur).

Compléter le code de cette application donné ci-dessous pour obtenir le comportement indiqué et l'aspect visuel comme sur l'image ci-dessus :

```
import javax.swing.*;
import javax.swing.text.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.border.*;

public class Operation extends JFrame {
    private static int nbRegistres; //le nombre de registres (sans accumulateur)
    private JLabel[] labels; // noms de registres
    private JTextField[] registres; // les registres y inclu l'accumulateur
    private JRadioButton[] radio; //les boutons de radio
    private JButton[] operations; //les boutons pour les operations
    private Box[] boxes; // Un tableau de Box horizontales.
                         // Dans chaque Box on peut mettre un JRadioButton
                         // un JTextField (registre) et un JLabel (etiquette
                         // avec le nom du registre.
    /* d'autres variables si necessaire */

    public static void main(final String[] args) {
      Operation.nbRegistres=4;
      if( args.length > 0 )
          Operation.nbRegistres = Integer.parseInt(args[0]);
      SwingUtilities.invokeLater( new Runnable(){
            public void run(){
                JFrame frame = new Operation();
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setVisible(true);
            }
          });
    }//fin main

    private Operation(){
      super("Calculateur");
      labels = new JLabel[Operation.nbRegistres+1];
      registres = new JTextField[Operation.nbRegistres+1];
      radio = new JRadioButton[Operation.nbRegistres+1];
      boxes = new Box[Operation.nbRegistres+1];

      labels[0]=new JLabel("Accumulateur (R0)");
      registres[0]=new JTextField(20);
      registres[0].setText("0");
      registres[0].setHorizontalAlignment(JTextField.RIGHT);
      radio[0]=new JRadioButton();
      boxes[0]=Box.createHorizontalBox();
      boxes[0].add(radio[0]);
      boxes[0].add(registres[0]);
```

```
        boxes[0].add(labels[0]);

        Box all = Box.createVerticalBox();
        all.add(boxes[0]);

        for(int i=1; i< registres.length; i++ ){
            labels[i]=new JLabel("R"+i);
            registres[i]=new JTextField(20);
            registres[i].setText("0");
            registres[i].setHorizontalAlignment(JTextField.RIGHT);

            radio[i]=new JRadioButton();
            boxes[i]=Box.createHorizontalBox();
            boxes[i].add(radio[i]);
            boxes[i].add(registres[i]);
            boxes[i].add(labels[i]);
            all.add(boxes[i]);
        }

        JPanel h = new JPanel();
        h.setLayout(new BorderLayout());
        h.setBorder(BorderFactory.createTitledBorder("Registres"));
        h.add(all,BorderLayout.CENTER);


        /*******************************************************
         * TROU A REMPLIR: construire le tableau operation de
         * JButton's. Associer les ecouteurs, mettre tous ces
         * JButtons dans un JPanel avec un encadrement approprie,
         * etc, etc
         *******************************************************/



        setMinimumSize(new Dimension(600,300));
    }//fin de constructeur

    // getValue() recupere la valeur dans un JTextField et le transforme
    // en double
    private static double getValueDouble(JTextField text){
      System.err.println("get "+text.getText());
      return Double.parseDouble( text.getText( ) );
    }
}
```