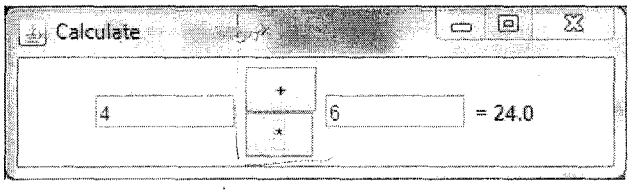
Université Paris 7
Master 1 d'Informatique, Interfaces graphiques.
21 juin 2010 Durée : 2h30

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.).

Exercice 1



L'application ci-dessus est constituée d'un champ de texte (`JTextField`), deux boutons (`JButton`), d'un deuxième `JTextField` et d'un `JLabel`. L'utilisateur remplit les deux champs `JTextField` avec les nombres `double` et clique sur un de deux bouton. Le clique sur le bouton + calcule la somme et affiche dans `JLabel` (après le signe =), le clique sur le bouton * affiche le résultat de multiplication de deux nombres.

Compléter le code pour obtenir ce comportement.

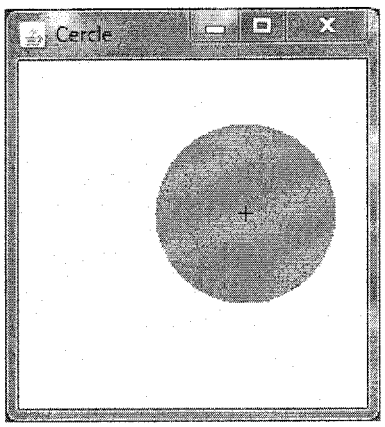
Remarque. `setText(String s)` permet de changer le texte dans `JLabel`

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import javax.swing.*;
4  class Calc extends JComponent  {
5      private JTextField a,b;
6      private JLabel r;
7      private JButton plus, mult;
8
9      public Calc()  {
10
11         a = new JTextField(7);
12         b = new JTextField(7);
13         r =new JLabel("=              ");
14         plus=new JButton("+");
15         mult=new JButton("*");
16         setLayout(new FlowLayout(FlowLayout.CENTER));
17         Box box = Box.createVerticalBox();
18         box.add(plus);
19         box.add(mult);
20         add(a);
21         add(box);
22         add(b);
23         add(r);
24
25         /*******************************
26          * installer les ecouteurs
27          *******************************/
28
29     }
30
31     /**********************************************
32      *  methode getD(JTextField f)
33      *  retourne la valeur double qui se trouve dans f
34      **********************************************/
35     public static double getD(JTextField f){
36         String text = f.getText();
37         double d;
38         try{
39             d=Double.parseDouble(text);
40         }catch(NumberFormatException e){
```

```
41              return Double.NaN;
42          }
43          return d;
44      }
45
46      public static void main(String [] args)
47      {
48          SwingUtilities.invokeLater( new Runnable(){
49                  public void run(){
50                      JFrame view = new JFrame();
51                      view.setTitle("Calculate");
52                      view.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
53                      view.setContentPane(new Calc());
54                      view.pack();
55                      view.setVisible(true);
56                  }
57              });
58      }
59  }
```

Exercice 2

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.
L'application ci-dessous



permet de dessiner un cercle plein avec centre.
L'affichage du cercle commence quand le curseur se trouve à l'intérieur de la fenêtre et l'utilisateur enfonce un bouton de la souris (sans le relâcher, ce n'est pas un clique de la souris). La position du curseur en ce moment donne le centre `(x_centre, y_centre)` du cercle.
Tant que l'utilisateur maintient le bouton de la souris enfoncé le cercle est redessiné à la volée. La position courante du curseur donne un point sur le cercle avec le coordonnées `(x_autre,y_autre)`.
Le cercle sera redessiné tant que l'utilisateur déplace la souris avec le bouton enfoncé. Une fois l'utilisateur relâche le bouton le dessin est terminé.
Le programme ci-dessus implémente le squelette de cette application, il faut remplir les trous indiqués pour obtenir le comportement recherché.
Si on commence le dessin d'un nouveau cercle l'ancien cercle peut être effacé, il suffit que seulement le dernier cercle soit affiché à un moment donné.
Noter que le centre du cercle doit être visible comme un croisement de deux segments (de longueur de 6 pixels par exemple).
Le cercle plein est dessiné en rouge par contre les deux segments qui désignent le centre du cercle sont en noir.

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import javax.swing.*;
4  class Cercle extends JComponent  {
```

```
5       /* (x_centre, y_centre) - les coordonnees du centre du cercle */
6       private int x_centre ;
7       private int y_centre ;
8       /* (x_autre, y_autre) les coordonnees d'un point sur le cercle */
9       private int x_autre;
10      private int y_autre;
11
12      private boolean inside;
13
14      public Cercle(int l, int h)  {
15          setPreferredSize(new Dimension(l,h));
16
17          /*****************************************
18           * premier trou,
19           * installer les ecouteurs
20           *****************************************/
21      }/* fin constructeur Cercle */
22
23
24      /*****************************************************************
25       * deuxieme trou
26       * la metode qui fait peindre le cercle avec le centre
27       *****************************************************************/
28
29
30
31      /*****************************************************************
32       * la methode distance calcule la distance entre le point (xa,ya) et
33       * le point (xb.yb)
34       *****************************************************************/
35      private static int distance( int xa, int ya, int xb, int yb ){
36          return (int) Math.sqrt( (xa-xb)*(xa-xb)+(ya-yb)*(ya-yb) );
37      }
38
39      public static void main(String [] args)
40      {
41          SwingUtilities.invokeLater( new Runnable(){
42                  public void run(){
43                      JFrame view = new JFrame();
44                      view.setTitle("Cercle");
45                      view.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
46                      view.setContentPane(new Cercle(200,200));
47                      view.pack();
48                      view.setVisible(true);
49                  }
50              });
51      }
52  }
```

Indication : Pour dessiner un cercle plein on utilise la méthode

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

de la classe `Graphics`, où `(x,y)` sont les coordonnées du sommet gauche/haut du rectangle dont lequel s'inscrit le cercle (donc `x = x_centre - rayon, y = y_centre - rayon` où `rayon` c'est le rayon du cercle).
Pour dessiner un segment on utilise la méthode

`void drawLine(int x1, int y1, int x2, int y2)`

de la classe `Graphics` qui dessine un segment entre les points `(x1,y1)` et `(x2,y2)`. Dans notre cas le segment verticale aura par exemple `x1=x2=x_centre` et `y1=y_centre-3, y2=y_centre + 3`. Le segment horizontal est obtenu de la même façon en remplaçant x par y et vice versa.
Noter que pour obtenir le dessin il faut commencer par dessiner le cercle et seulement ensuite les segments (sinon le cercle couvrira les segments).
Noter aussi que la couleur utilisée pour dessiner doit être chaque fois renseignée par la méthode `void setColor(Color color)` de la classe `Graphics` et ce n'est pas la même pour le cercle que pour les segments.

Exercice 3

L'application ci-dessus est composée d'un champs texte `GtkEntry`, d'un bouton `GtkButton` et d'une étiquette `GtkLabel`. Lorsqu'on clique sur le button, la valeur de l'entier qui se trouve dans la zone du texte est affichée dans l'étiquette en hexadécimal.
Le squelette du programme ci-dessous ne contient pas de gestion de signaux GTK+. Il faut compléter les trous laissés dans le code.

```
1  #include <string.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <gtk/gtk.h>
5
6  static GtkWidget *text;
7  static GtkWidget *label;
8
9  /* transforme un entier vers sa representation hexadecimale */
10 static char *int_to_hexa(int d){
11 #define TAILLE 20
12   static char tampon[TAILLE];
13   sprintf(tampon,"%19x",d);
14   return tampon;
15 }
16
17 /* A COMPLETER  */
18
19 int
20 main( int    argc, char *argv[] )
21 {
22     GtkWidget *window;
23     GtkWidget *hbox;
24     GtkWidget *bouton;
25     gtk_init (&argc, &argv);
26
27     /* Creation de la fenetre principale */
28     window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
29     gtk_window_set_title (GTK_WINDOW (window), "VersHexa");
30     g_signal_connect (G_OBJECT (window), "destroy",
31                       G_CALLBACK (gtk_main_quit), NULL);
32     g_signal_connect (G_OBJECT (window), "delete_event",
33                       G_CALLBACK (gtk_main_quit), NULL);
34
35     gtk_container_set_border_width (GTK_CONTAINER (window), 10);
36
37     text = gtk_entry_new();
38
39     label = gtk_label_new("");
40
41     bouton = gtk_button_new_with_label("vaut_en_hexa");
42
43
44     hbox = gtk_hbox_new(TRUE,3);
45
46     gtk_box_pack_start(GTK_BOX(hbox),text,FALSE,FALSE,1);
47     gtk_box_pack_start(GTK_BOX(hbox),bouton,FALSE,FALSE,1);
48     gtk_box_pack_start(GTK_BOX(hbox),label,FALSE,FALSE,1);
49
50     /* A COMPLETER */
51
52     gtk_container_add (GTK_CONTAINER(window), hbox);
53
54     gtk_widget_show_all (window);
55
56     gtk_main ();
57
58     return 0;
59 }
```

Remarque : Pour changer la valeur d'une étiquette la fonction

```
void gtk_label_set_text( GtkLabel *label, const char *str )
```