Université Paris 7

Master 1 d'Informatique, Interfaces graphiques.

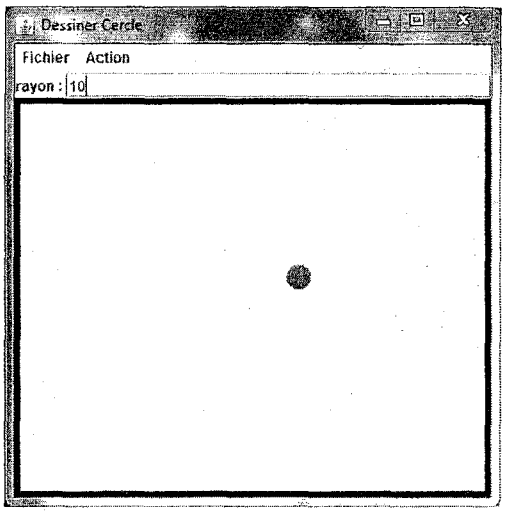
26 mai 2010 Durée : 2h30

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

Le sujet comporte 5 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 (15 points) L'application ci-dessous



est composée d'un menu, d'un objet `JLabel` contenant le mot `rayon`, d'un `JTextField` et d'un `JPanel` entouré d'un bord noir.

Initialement `JTextField` contient la valeur 10 et le `JPanel` au-dessous est vide. Quand l'utilisateur clique sur le `JPanel` on dessinera dans ce `JPanel` un cercle dont le rayon est donné par la valeur dans `JTextField` et dont le centre se trouve sur le point où se trouve le curseur quand on clique.

Le menu est composé de deux élément. Le menu `Fichier` contient un seul `JMenuItem` : `quitter` qui terminera l'exécution de l'application (le racourci clavier `q`).

Le menu `Action` contient deux `JMenuItems` :

- `effacer` qui efface tout dessin de `JPanel` (efface le(s) cercle(s)), le raccourci clavier `e`,
- `couleur` – l'activation de cet item lance un `JColorChooser`. L'utilisateur pourra choisir la couleur qui sera utilisée pour dessiner le cercles à partir de ce moment. (Initialement c'est le rouge qui est utilisé.) Le racourci clavier `c`.

Le code suivant présente le squelette de cette application

```
1  import javax.swing.*;
2  import javax.swing.event.*;
3  import java.awt.*;
4  import java.awt.event.*;
5  import java.awt.geom.*;
6
7
8  /*****************************************************************
9   *  DessinerCercle --- le JPanel c'est la toile
10  *  sur laquelle on dessine le(s) cercle(s)
11  *****************************************************************/
12 public class DessinerCercle extends JPanel{
13     private int rayon=10; //le rayon du cercle
14     private  int cx, cy; //les coordonnees du centre du cercle
15     private Color color=Color.RED; //la couleur utilisee pour dessiner le cercle
16     private JTextField text = new JTextField(10);
17     private boolean clear = true; //clear est true signifie
```

```
18      //que la toile doit etre vide
19
20      public static void main(final String[] args) {
21          SwingUtilities.invokeLater( new Runnable(){
22                  public void run(){
23                      new DessinerCercle(400,400);
24                  }
25              } );
26      }
27
28      private DessinerCercle(int w, int h){
29          JFrame frame = new JFrame("Dessiner Cercle");
30          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
31          frame.setSize(w,h);
32
33          /* creer le bord autour de la toile */
34          setBorder( BorderFactory.createLineBorder(Color.BLACK,5) );
35
36          /*********************************************************
37           *  TROU 1:
38           *  creation de JLabel "rayon"
39           *  arranger tous les elements : JLabel, text et
40           *  la toile elle meme (this)
41           *  comme c'est indique dans le texte de l'exercice
42           *  pour le mettre dans le JFrame frame
43           *********************************************************/
44
45          /*********************************************************
46           *  TROU 2
47           *  associer un ecouteur a JTextField (text)
48           *********************************************************/
49
50          /*********************************************************
51           * TROU 3
52           * ajouter un ecouteur a la toile (this)
53           * C'est l'ecouteur souris pour recuperer le clic sur la toile
54           * et provoquer l'action de dessiner le cercle
55           *********************************************************/
56
57          /*********************************************************
58           *  TROU 4
59           *  mettre en oevre le menu avec les ecouteurs appropries
60           *********************************************************/
61          frame.setVisible(true);
62      }// fin constructeur DessinerCercle
63
64
65      /*********************************************************
66       * TROU 5
67       * ici la methode qui effectivemnt dessine le cercle
68       * Autrement c'est ici qu'il faudra mettre votre version de
69       * la methode ... (a vous deviner laquelle, c'est la seule qui
70       * a l'objet Graphics en parametre et qui permet donc faire le dessin.
71       *********************************************************/
72  }
```

Il faut remplir les trous indiqués pour obtenir le comportement recherché.

Remarque. Il y a deux versions possibles de ce programme.

La première version, plus facile à réaliser : quand on fait un nouveau cercle on ne demande pas de garder les cercles précédents. Donc chaque fois la toile contient au plus un cercle. Cette version de base est suffisante.

Une version plus compliquée c'est La version ou les cercles consécutifs s'ajoutent sur la toile un après l'autre. Ce programme est plus difficile à réaliser (il faut gérer le reaffichage de plusieurs

cercles) mais rapporte des points supplémentaires. Ceci dit, il faut se concentrer surtout sur la première version et regarder éventuellement la deuxième uniquement si vous avez le temps libre.
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.
Rappel : Pour dessiner le cercle on utilisera la méthode

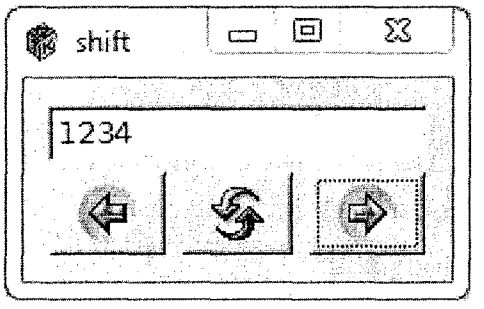
```
void  drawOval(int x, int y, int width, int height)
```

de la classe `Graphics` (ce qui implique que cela est utilisable uniquement là où on peut accéder à l'objet `Graphics` associé à `JPanel` qui nous sert de la toile sur laquelle on dessine). Cette méthode dessine une ellipse. Les paramètres `x,y,width,height` décrivent le rectangle dans laquelle s'inscrit l'ellipse :

- `width, height` — la largeur et la hauteur de ce rectangle (donc on obtient un cercle si la largeur égale à la hauteur égales deux fois le rayon du cercle),
- `x, y` ce sont les coordonnées du sommet gauche haut de ce rectangle (faciles à calculer si on a les coordonnées du centre du cercle et le rayon).

Rappel : L'objet `Graphics` possède la méthode `void setColor(Color)` qui permet de changer la couleur utilisée pour dessiner.

Exercice 2 (5 points) L'interface du programme ci-dessous



est constituée d'une zone de texte (`GtkEntry`) et de trois boutons (`GtkButton`). 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"`.
Question 1. Quelle est le rôle des instructions qui se trouvent dans les lignes 83,86,96,101 dans le code ci-dessous? (Une phrase suffit comme réponse).
Question 2. Compléter le code ci-dessous pour obtenir le résultat recherché.
Noter que ce qu'il manque ce sont les fonctions callback et le code qui associe les fonction callback aux objets.

```
1  #include <string.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <gtk/gtk.h>
5
6  static GtkWidget *entry;
7
8  static GtkWidget *bouton[3];
9
10 #define TAILLE 20
11 /* le tampon pour stocker le text */
12 static char tampon[TAILLE];
13
14 /*************************************************************
15  * swap fait l'image miroire du texte stocke dans le tampon
```

```
16   ************************************************************/
17  static void swap() {
18      int i,j;
19      char c;
20      i=0;
21      j=strlen(tampon)-1;
22      for ( ; i < j ; i++,j--) {
23          c = tampon[i];
24          tampon[i]=tampon[j];
25          tampon[j]=c;
26      }
27  }
28
29  /************************************************************
30   * right fait le decalage droit du text stocke dans le tampon
31   ************************************************************/
32  static void right() {
33      int i = strlen(tampon);
34      if ( i<=1 )
35          return;
36      char c = tampon[i-1];
37      memmove(tampon+1,tampon,i-1);
38      tampon[0]=c;
39  }
40
41  /************************************************************
42   * left fait le decalage gauche du text stocke dans le tampon
43   ************************************************************/
44  static void left() {
45      int i = strlen(tampon);
46      if ( i <= 1)
47          return;
48      char c = tampon[0];
49      memmove(tampon,tampon+1,i-1);
50      tampon[i-1]=c;
51  }
52
53  /************************************************************
54   *  TROU 1
55   *  Les fonctions calback qu'on associera aux
56   *  trois boutons et a GtkEntry
57   ************************************************************/
58
59  int
60  main( int    argc, char *argv[] ) {
61
62      GtkWidget *window;
63      GtkWidget *image;
64      GtkWidget *hbox, *vbox;
65
66      gtk_init (&argc, &argv);
67
68      /* Creation de la fenetre principale */
69      window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
70
71      gtk_window_set_title (GTK_WINDOW (window), "shift");
72
73      g_signal_connect (G_OBJECT (window), "destroy",
74                        G_CALLBACK (gtk_main_quit), NULL);
75
76      g_signal_connect (G_OBJECT (window), "delete_event",
77                        G_CALLBACK (gtk_main_quit), NULL);
78
```

```
79      gtk_container_set_border_width (GTK_CONTAINER (window), 10);
80
81      tampon[0]='\0';/* initialiser le tampon*/
82
83      entry = gtk_entry_new();
84      gtk_entry_set_max_length( GTK_ENTRY(entry), TAILLE - 1);
85
86      bouton[0] = gtk_button_new();
87      bouton[1] = gtk_button_new();
88      bouton[2] = gtk_button_new();
89      image = gtk_image_new_from_file("Images/left.png");
90      gtk_button_set_image(GTK_BUTTON(bouton[0]),image);
91      image = gtk_image_new_from_file("Images/reverse.png");
92      gtk_button_set_image(GTK_BUTTON(bouton[1]),image);
93      image = gtk_image_new_from_file("Images/right.png");
94      gtk_button_set_image(GTK_BUTTON(bouton[2]),image);
95
96      hbox = gtk_hbox_new(FALSE,5);
97      gtk_box_pack_start( GTK_BOX(hbox), bouton[0], TRUE, TRUE, 1);
98      gtk_box_pack_start( GTK_BOX(hbox), bouton[1], TRUE, TRUE, 1);
99      gtk_box_pack_start( GTK_BOX(hbox), bouton[2], TRUE, TRUE, 1);
100
101     vbox = gtk_vbox_new(FALSE,3);
102     gtk_box_pack_start( GTK_BOX(vbox), entry, TRUE, TRUE, 1);
103     gtk_box_pack_start( GTK_BOX(vbox), hbox, TRUE, TRUE, 1);
104
105     /*************************************************
106      *  TROU 2
107      *  Connecter les boutons aux fonctions callback appropriees.
108      *  Connecter entry a la fonction callback appropropriee.
109      *************************************************/
110
111     gtk_container_add (GTK_CONTAINER(window), vbox);
112     gtk_widget_show_all (window);
113     gtk_main ();
114     return 0;
115 }
```

Rappel. Le signal `clicked` est envoyé quand l'utilisateur appuie sur un bouton `GtkButton`. Le signal `activate` est envoyé quand l'utilisateur tape ENTER après avoir taper un texte dans `GtkEntry`.
Rappel. La fonction

```
gchar*  gtk_entry_get_text(GtkEntry *entry);
```

permet de récupérer le pointeur vers le texte qui se trouve dans `GtkEntry`
La fonction

```
void gtk_entry_set_text(GtkEntry *entry, const gchar *text);
```

permet de mettre un texte (chaîne de caractères) dans `GtkEntry` Le deuxième paramètre de cette fonction est un pointeur vers la chaîne de caractères qui contient le texte à mettre dans `GtkEntry` qu'on passe dans le premier paramètre.