```java
//déclaration du package
package cms_tp20;

//les imports
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

//l'en-tête de la classe publique Fenestra qui correspond à un container de premier
//niveau et qui écoute les 4 boutons placés dans son contentPane
public class Fenestra extends JFrame implements ActionListener {

    //déclaration d'un champ privé de type tableau de boutons swing, nommé boutons et
    //initialisé avec l'adresse d'un nouveau tableau de quatre éléments créé pour
    //l'occasion
    private JButton[] boutons = new JButton[4];

    //déclaration d'un champ privé de type tableau de couleurs, nommé couleurs et
    //initialisé avec l'adresse d'un nouveau tableau de quatre éléments créé pour
    //l'occasion et qui a lui-même comme éléments initiaux les couleurs rose, jaune,
    //cyan et verte
    private Color[] couleurs = {Color.PINK, Color.YELLOW, Color.CYAN, Color.GREEN};

    //déclaration d'un champ privé de type texte, nommé textes et initialisé avec
    //l'adresse d'un nouveau tableau de quatre éléments créé pour l'occasion et qui a
    //lui-même comme éléments initiaux les textes ROSE, JAUNE, CYAN et VERTE
    private String[] textes = {"ROSE", "JAUNE", "CYAN", "VERTE"};

    //l'en-tête du constructeur public sans argument
    public Fenestra() {

        //instruction qui impose l'arrêt de la machine virtuelle suite à la fermeture du
        // container de premier niveau (appelé par la suite tout simplement fenêtre)
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
//instruction qui donne le titre Couleurs à la fenêtre
setTitle("Couleurs");

//instruction qui fixe la taille de la fenêtre à 225 pixels x 400 pixels
setSize(225, 400);

//instruction qui rend la fenêtre non redimensionnable
setResizable(false);

//instruction qui associe à la fenêtre un nouveau gestionnaire de mise en forme
//créé pour l'occasion et correspondant à une grille avec quatre lignes et
//une seule colonne
setLayout(new GridLayout(4,1));

//l'en-tête d'une boucle appropriée
for(int i=0; i<4; i++) {

     //création d'un bouton swing dont l'adresse est stockée dans le
     //"bon" élément du tableau boutons
     boutons[i] = new JButton();

     //instruction qui associe à chaque bouton la couleur correspondant
     //à l'élément de même indice du tableau couleurs
     boutons[i].setBackground(couleurs[i]);

     //instruction qui associe  à chaque bouton le texte correspondant
     //à l'élément de même indice du tableau textes
     boutons[i].setText(textes[i]);

     //instruction qui met chaque bouton sous l'écoute de la fenêtre
     boutons[i].addActionListener(this);

     //instruction qui ajoute chaque bouton à la fenêtre
     add(boutons[i]);
}//fin de la boucle
```

```
        //instruction qui rend la fenêtre visible
        setVisible(true);
    }//fin du constructeur

    //la méthode getIndice doit permettre de savoir si l'adresse d'un certain
    //bouton indiqué comme argument de la méthode se trouve parmi les éléments
    //du tableau boutons et, si c'est le cas, quel est l'indice qui lui correspond

    //l'en-tête de la méthode privée getIndice qui a un seul argument de type
    //bouton swing nommé jbt et retourne un résultat de type numérique entier
    private int getIndice(JButton jbt) {

        //l'en-tête d'une boucle appropriée
        for(int i=0; i<boutons.length; i++) {

            //test si l'adresse jbt correspond à un des éléments du tableau boutons
            if(jbt == boutons[i]) {

                // instruction qui retourne l'indice de cet élément
                return i;
            }//fin du test
        }

        //instruction qui retourne la valeur "spéciale" -1
        return -1;
    }//fin de la méthode getPosition


    @Override
    //l'en-tête du gestionnaire d'événements approprié qui doit rendre les quatre
    //boutons sensibles aux clics de la souris
    public void actionPerformed(ActionEvent ae)
    {
```

```
        //test si la source de l'événement traité par le gestionnaire est un objet dont
        //le type n'est pas celui d'un bouton swing (ou compatible par polymorphisme)
        if(! (ae.getSource() instanceof JButton)) {

             //instruction pour quitter immédiatement le gestionnaire
             return;
        }//fin du test

        //instructions qui permettent de quitter immédiatement le gestionnaire
        //si l'adresse de la source de l'événement traité par le gestionnaire
        //ne se retrouve pas parmi les éléments du tableau boutons
        JButton btSource = (JButton)ae.getSource();
        int pos = getIndice(btSource);

        if(pos == -1) {
             return;
        }

        //instructions qui permettent au bouton source de l'événement traité d'échanger
        //la couleur et le texte associé avec le "bon" bouton parmi les trois autres
        //boutons
        JButton btEnBas = boutons[(pos+1) % 4];

        Color coulSource = btSource.getBackground();
        String texteSource = btSource.getText();
        Color coulEnBas = btEnBas.getBackground();
        String texteEnBas = btEnBas.getText();

        btSource.setBackground(coulEnBas);
        btSource.setText(texteEnBas);
        btEnBas.setBackground(coulSource);
        btEnBas.setText(texteSource);
     }//fin du gestionnaire d'événements
}//fin de la classe graphique
```