Corrigé du TD 5 : étude d'un applet simple, classe Vector

1 Étude syntaxique d'un applet

```
// Permet d'acceder à la classe JApplet du package javax.swing
import javax.swing.JApplet;
// Permet d'acceder à la classe Graphics du package java.awt
import java.awt.Graphics;
// Permet d'acceder à la classe Color du package java.awt
import java.awt.Color;
// Permet d'acceder à toutes les classes du package java.awt.event
import java.awt.event.*;

// Déclaration d'une nouvelle classe appelé "ACommenter"
// Cette hérite de "JApplet"
// Cette classe implémente les éléments décrits par une
// interface appelé "MouseListener", en d'autres termes
// elle devra implémenter toutes les méthodes dont les prototypes sont
// donnés par MouseListener
public class ACommenter extends Applet implements MouseListener
{
    // Déclaration de deux membres de type entier, qui seront
    // automatiquement initialises a 0
    int lx,ly;

    // Declaration d'une methode (c'est a dire, une fonction membre)
    // appelé "paint". Cette methode (public) est accesible a partir
    // de n'importe quelle autre classe.
    // Cette methode prend un argument de type "Graphics".
    // Cet argument s'appele "g", il s'agit  d'une reference vers un
    // objet de type "Graphics"
    // Il se trouve que paint est deja defini dans la classe "JApplet" donc
    // en le redefinissant ici on remplace la methode "paint" de "JApplet"
    public void paint(Graphics g)
    {
        // Appel d'une methode statique appelé "random", d'une
        // classe appelé "Math"
        // Declaration d'un nombre flotant double precision qui
        // sera initialisé par la valeur retourné par Math.random()
        double r=Math.random();


        // Appel d'une methode non statique appelé "setColor" de la
        // classe "Graphics". Cet appel se fait sur l'objet
        // referencé par "g"
```

```
    // En argument: membre statique "green" d'une classe
    // appelé "Color"
    g.setColor(Color.green);

    // Meme chose...
    // En argument:
    // -Creation d'un nouveau objet de type Color
    // -Appel de son constructeur avec 3 valeurs float
    g.setColor(new Color(.1f,.5f,.9f));

    // Appel d'une methode non statique appelé "drawLine" de la
    // classe "Graphics"
    // Cet appel se fait sur l'objet referencé par "g"
    g.drawLine(0,0,lx,ly);
}

// Meme chose que pour "paint", ici pas d'arguments
public void init()
{
    // Appel d'une methode appelé "addMouseListener"
    // on deduit que cette methode doit etre definie quelque part
    // dans "Applet" ou ses parents.
    // En regardant la doc on peut voir que "addMouseListener" fait
    // partie d'une classe "Component" qui est parent de "Container"
    // qui est parent de "Panel" qui est parent de "JApplet"...
    // En argument: this: reference vers l'objet courant
    // (donc de type "ACommenter")
    addMouseListener(this);
}

// Cinq methodes prenant comme argument une reference "e"
// vers un objet de type "MouseEvent"
// En regardant la doc, on voit que ces methodes font partie de
// l'interface "MouseListener".
// On est donc en train d'implementer effectivement
// les elements decrits par l'interface "MouseListener"
// Les quatres premieres ne font rien
public void mouseEntered (MouseEvent e) {;}
public void mouseExited  (MouseEvent e) {;}
public void mousePressed (MouseEvent e) {;}
public void mouseReleased(MouseEvent e) {;}
public void mouseClicked (MouseEvent e)
{

    // appel de deux methodes non-statiques de l'objet
    // reference par "e"
    // affectation du resultat dans lx,ly
    lx=e.getX();
    ly=e.getY();

    // Appel de la methode non-statique "println" du membre
    // statique "out" de la classe "System"
    // En argument:
    // appel de deux methodes non-statiques de l'objet
    // reference par "e"
```

```
        // l'operateur "+" associe a la premiere chaine de caracteres,
        // force la conversion des element suivants en chaines de
        // caracteres et les concatene pour former une seule chaine de
        // caracteres
        System.out.println("PositionA:"+
                           e.getX()   +","+
                           e.getY());

        // Appel d'une methode appelé "repaint"
        // Comme on ne l'a pas definie dans cette classe elle doit etre
        // definie dans Applet ou ses parents. En regardant la doc
        // on voit qu'elle fait partie de la classe "Component"
        repaint();

    }
}
```

2 Étude sémantique d'un applet

```
// un petit programme qui va dessiner une ligne chaque
// fois qu'on clique sur un point de l'écran

import javax.swing.JApplet;
import java.awt.Graphics;
import java.awt.Color;
import java.awt.event.*;

// on défini un classe principale qui représente la fenêtre principale
// on va y gérer directement les clicks de souris
public class ACommenter extends Applet implements MouseListener
{
    // les coordonnés du dernier point sur lequel on a cliqué
    int lx,ly;

    public void paint(Graphics g)
    {
        // ceci ne sert a rien, puisque r n'est plus utilisé
        double r=Math.random();
        // ceci ne sert a rien, puisque la couleur sera changé aprés
        g.setColor(Color.green);
        // setColor permet d'indiquer la couleur courante de dessin
        // à partir de maintenant les dessin se feront dans
        // une couleur tendant vers le bleu
        g.setColor(new Color(.1f,.5f,.9f));

        // dessin de la ligne entre l'origine et le point courant
        // determiné par le click de la souris
        g.drawLine(0,0,lx,ly);
    }

    public void init()
    {
        // on dit à JApplet, que c'est l'objet courant (this) qui va
```

```
        // gérer les évènements de la souris
        addMouseListener(this);
    }

    // ces méthodes ne font rien (elles sont nécessaires
    // parce qu'on doit implémenter tout MouseListener)
    public void mouseEntered (MouseEvent e) {;}
    public void mouseExited  (MouseEvent e) {;}
    public void mousePressed (MouseEvent e) {;}
    public void mouseReleased(MouseEvent e) {;}
    // méthode appelé automatiquement par JApplet lorsqu'il y a
    // un evenement click de la souris
    public void mouseClicked (MouseEvent e)
    {
        // on demande à la description de l'évènement
        // de nous fournir les coordonnées du point cliqué
        lx=e.getX();
        ly=e.getY();
        // affichage de ces coordonnées
        System.out.println("PositionA:"+
                           e.getX()   +","+
                           e.getY());
        // on dit a JApplet qu'il faudra rafraîchir l'affichage bientôt
        // JApplet appelera, plus tard, la méthode paint
        repaint();
    }
}
```

3 Classe `Vector`

```
import java.awt.Point;

class Lapin
{
    String nom;
    public String toString(){return nom;}
    Lapin(String _nom){nom=_nom;}
};


class Vector
{
    private int capacite;
    private int taille;
    private Object data[];
    void add(Object o)
    {
        if(taille>=capacite)
        {
            capacite=(capacite!=0 ?  2*capacite : 1);
            Object ndata[]=new Object[capacite];
            for(int i=0;i<taille;i++)
```

```
            {
                ndata[i]=data[i];
            }
            data=ndata;
        }
        data[taille++]=o;
    }
    void add(int pos, Object o)
    {
        add(o);
        for(int i=taille-1;i>pos;i--)
        {
            data[i]=data[i-1];
        }
        data[pos]=o;
    }
    int size(){return taille;}
    Object get(int pos){return data[pos];}
    void remove(int pos)
    {
        for(int i=pos;i<taille-1;i++)
        {
            data[i]=data[i+1];
        }
        data[--taille]=null;
    }

    void clear()
    {
        data=null;
        capacite=0;
        taille=0;
    }
    public String toString()
    {
        String res="[ ";
        for(int i=0;i<taille;i++)
        {
            res+=data[i]+" ";
        }
        res+="]";
        return res;
    }
}

class VectorTestCorrige
{
    public static void main(String args[])
    {
        Vector v=new Vector();
        v.add(new Point(1,2));
        v.add(new Lapin("roger"));
        v.add(1,new Lapin("titi"));
        for(int i=0;i<v.size();i++)
        {
```

```
            // affichage, utilise toString de chaque Object dans v
            System.out.println(v.get(i));
        }
        // enlève l'élément à l'indice 0
        v.remove(0);
        // affichage, utilise toString de Vector
        System.out.println(v);
        v.clear();
    }
}
```