

```
public class NumberInfo
{
    // valeur numérique
    private double value;

    // nom ou étiquette de la barre
    private String label;

    public NumberInfo(double value, String label)
    {
        this.value = value;
        this.label = label;
    }

    public String getLabel()
    {
        return label;
    }

    public double getValue()
    {
        return value;
    }

    public String toString()
    {
        return label + " : " + value;
    }
}
```

```
public class NumberInfos
{
    // la série des nombres avec leur description
    private ArrayList<NumberInfo> alNumberInfos = new ArrayList<>();

    // le titre de la série
    private String title;

    public NumberInfos(String title)
    {
        this.title = title;
        // pour tester... à commenter/enlever quand c'est OK!
        alNumberInfos.add(new NumberInfo(63, "JAN"));
        alNumberInfos.add(new NumberInfo(65, "FEB"));
        alNumberInfos.add(new NumberInfo(58, "MAR"));
        alNumberInfos.add(new NumberInfo(77, "APR"));
        alNumberInfos.add(new NumberInfo(111, "MAY"));
        alNumberInfos.add(new NumberInfo(124, "JUN"));
        alNumberInfos.add(new NumberInfo(104, "JUL"));
        alNumberInfos.add(new NumberInfo(99, "AUG"));
        alNumberInfos.add(new NumberInfo(80, "SEP"));
        alNumberInfos.add(new NumberInfo(66, "OCT"));
        alNumberInfos.add(new NumberInfo(103, "NOV"));
        alNumberInfos.add(new NumberInfo(80, "DEC"));
    }

    public String getTitle()
    {
        return title;
    }

    public void add(double value, String label)
    {
        alNumberInfos.add(new NumberInfo(value, label));
    }

    public double getTotal()
    {
        double result = 0;
        for (int i = 0; i < alNumberInfos.size(); i++)
            result = result + alNumberInfos.get(i).getValue();
        return result;
    }

    public double getMaximum()
    {
        // On suppose que les valeurs sont >= 0 !!
        double result = 0;
        for (int i = 0; i < alNumberInfos.size(); i++)
            if (result < alNumberInfos.get(i).getValue())
                result = alNumberInfos.get(i).getValue();
        return result;
    }

    public Object[] toArray()
    {
        return alNumberInfos.toArray();
    }

    public int size()
    {
        return alNumberInfos.size();
    }

    public NumberInfo get(int i)
    {
        return alNumberInfos.get(i);
    }

    // .... suite prochaine page ...
}
```

```
public void draw(Graphics g, int pWidth, int pHeight)
{
    // VERSION 1 - sans marges

    // dessiner le titre de l'histogramme (avec un effet)
    g.setColor(Color.DARK_GRAY);
    g.drawString(title, 3, 11);
    g.setColor(Color.BLACK);
    g.drawString(title, 2, 10);

    // ne calculer que s'il y au moins 1 barre
    if (alNumberInfos.size() > 0)
    {
        NumberInfo info;
        double barHeight;
        double unitHeight = (double) (pHeight - 1) / getMaximum();
        double barWidth = (double) (pWidth - 1) / alNumberInfos.size();
        for (int i = 0; i < alNumberInfos.size(); i++)
        {
            info = alNumberInfos.get(i);

            barHeight = info.getValue() * unitHeight;

            int x = (int) (i * barWidth);
            int y = pHeight - (int) barHeight;
            int w = (int) barWidth;
            int h = (int) barHeight;

            // alterner les couleurs pour dessiner la barre (sans contour)
            if (i % 2 == 0)
                g.setColor(Color.YELLOW);
            else
                g.setColor(Color.ORANGE);
            g.fillRect(x, y, w, h);

            // dessiner le contour
            g.setColor(Color.RED);
            g.drawRect(x, y, w, h);

            // dessiner le texte et la valeur en bas de la barre
            g.setColor(Color.BLUE);
            g.drawString(info.getLabel(), x + 5, pHeight - 4);
            g.drawString(String.valueOf(info.getValue()), x + 5, pHeight - 18);
        }
    }
}
```

```
public class DrawPanel extends javax.swing.JPanel
{
    // modèle : vient du MainFrame via le setter
    private NumberInfos numberInfos = null;

    public void setNumberInfos(NumberInfos pNumberInfos)
    {
        numberInfos = pNumberInfos;
    }

    public DrawPanel()
    {
        initComponents();
    }

    public void paintComponent(Graphics g)
    {
        int h = getHeight();
        int w = getWidth();

        // effacer l'arrière fond
        g.setColor(Color.white);
        g.fillRect(0, 0, w, h);

        // dessiner l'histogramme (si présent)
        if (numberInfos != null)
            numberInfos.draw(g, w, h);
    }
    // Skipped: ... initComponents { ... }
    // Skipped: ... graphic attributes
}
```

```
public class MainFrame extends javax.swing.JFrame
{
    // modèle : ne sera construit qu'avec le bouton "New"
    private NumberInfos numberInfos = null;

    public MainFrame()
    {
        initComponents();
    }

    private void updateView()
    {
        // met à jour la liste dans la vue
        // !! à faire à chaque fois que quelque-chose change dans le modèle !!
        numberInfoJList.setListData(numberInfos.toArray());
        repaint();
    }
    // Skipped: ... initComponents { ... }
    private void newButtonActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_newButtonActionPerformed
        // construit un nouveau modèle
        numberInfos = new NumberInfos(titleTextField.getText());

        // transfère vers la "view"
        drawPanel.setNumberInfos(numberInfos);

        // gestion des boutons
        addButton.setEnabled(true);

        updateView();
    } //GEN-LAST:event_newButtonActionPerformed

    private void addButtonActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_addButtonActionPerformed
        numberInfos.add(Double.valueOf(valueTextField.getText()),
            labelTextField.getText());
        updateView();
    } //GEN-LAST:event_addButtonActionPerformed
    // Skipped: ... Look & Feel
    // Skipped: ... graphic attributes
}
```