Calc.java • Procédure de lancement

```
import javax.swing.*;
import java.awt.*;
import java.io.*;

public class Calc
{
  public static void main(String args[])
  {
    // L'application essaie de prendre le Look And Feel du systeme courant :
    try
    {
      UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName() );
    }
    catch(Exception e) {} // Sinon tant pis !

    // Cree la fenetre principale de l'application :
    JFrame fenetre = new CalcGUI();

    // Lecture de la taille de l'écran :
    java.awt.Dimension tailleEcran = java.awt.Toolkit.getDefaultToolkit().getScreenSize();

    // Lorsque la fenetre sera fermee, l'application quittera :
    fenetre.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // Verification d'integrite des controles de la fenetre :
    fenetre.pack();

    // Maintenant que la fenetre a sa taille finale, la centre sur l'ecran :
    fenetre.setLocation( (tailleEcran.width-fenetre.getWidth())/2,
                         (tailleEcran.height-fenetre.getHeight())/2 );

    // Affiche la fenetre pour notre cher utilisateur :
    fenetre.setVisible(true);
  }
};
```

CalcGUI.java • Fenêtre de la calculatrice

```
import javax.swing.*;
import java.awt.*;
import java.io.*;
import java.awt.event.*;
import javax.swing.tree.*;
import java.util.EventObject;
import javax.swing.JComponent;


public class CalcGUI extends JFrame implements ActionListener
{

  private class CalcGestionSouris extends MouseAdapter
  {
    // Gestion des evenements du type clic sur un des boutons de la calculatrice :
    public void mouseReleased(MouseEvent e)
    {
      gestionEvent(e);
    }
  };


  private class CalcGestionClavier implements ActionListener
  {
    // Recuperation des appuis sur les touches du clavier :
    public void actionPerformed(ActionEvent e)
    {
      gestionEvent(e);
    }
  };


  // Gestionnaire commun des evenements de la calculatrice :
  private void gestionEvent(EventObject e)
  {
    for(int i=0; i<10; i++)
      if(e.getSource()==bTouche[i])
      {
        calcman.addChiffre(i);
        afficheur.setText(String.valueOf(calcman.getValCourante()));
      }

    if(e.getSource()==bEfface)
    {
      calcman.clear();
      afficheur.setText(String.valueOf(calcman.getValCourante()));
    }

    if(e.getSource()==bOperateurDivise)
    {
      calcman.operation(calcman.DIVISION);
      afficheur.setText(String.valueOf(calcman.getValCourante()));
    }
```

```
  if(e.getSource()==bOperateurMultip)
  {
    calcman.operation(calcman.MULTIPLICATION);
    afficheur.setText(String.valueOf(calcman.getValCourante()));
  }

  if(e.getSource()==bOperateurMoins)
  {
    calcman.operation(calcman.SOUSTRACTION);
    afficheur.setText(String.valueOf(calcman.getValCourante()));
  }

  if(e.getSource()==bOperateurPlus)
  {
    calcman.operation(calcman.ADDITION);
    afficheur.setText(String.valueOf(calcman.getValCourante()));
  }

  if(e.getSource()==bPoint)
  {
    calcman.addVirgule();
    afficheur.setText(String.valueOf(calcman.getValCourante()));
  }

  if(e.getSource()==bSigne)
  {
    calcman.invSigne();
    afficheur.setText(String.valueOf(calcman.getValCourante()));
  }

  if(e.getSource()==bEgal)
  {
    calcman.calcul();
    afficheur.setText(String.valueOf(calcman.getValCourante()));
  }

  fenArbre.affiche(calcman.getDerniereValeur());
}


// Membres donnees des menus :
private JMenuBar menuBar;
private JMenu menuFichier;
private JMenuItem itemQuitter;
private JMenu menuAffichage;
private JMenuItem itemArbreOn;
private JMenuItem itemArbreOff;

// Boutons :
private JButton bTouche[]; // 0 a 9
private JButton bOperateurPlus;
private JButton bOperateurMoins;
private JButton bOperateurMultip;
private JButton bOperateurDivise;
private JButton bEfface;
private JButton bPoint;
private JButton bSigne;
private JButton bEgal;

// Zone de texte afficheur :
private JTextField afficheur;

// Gestionnaire de la geometrie de Patrice :
private GridBagContainLayout layout;

// Titre de l'application :
private final String titreApplication= "Calculatrice Java";

// Objet effectuant les calculs : "Processeur" de notre calculatrice :
private CalcSys calcman;

// Gestionnaire des evenements de la souris :
private CalcGestionSouris gSouris;
private CalcGestionClavier gClavier;

// Fenetre de l'arbre :
CalcTree fenArbre;



// Constructeur : initialisation de la fenetre :
public CalcGUI()
{
  super();

  int i;

  // Specifie le titre de la fenetre :
  setTitle(titreApplication);

  // Creation des boutons :
  bTouche=new JButton[10];
  for(i=0; i<10; i++) bTouche[i]=new JButton(String.valueOf(i));
  bOperateurPlus=new JButton("+");
  bOperateurMoins=new JButton("-");
  bOperateurMultip=new JButton("x");
  bOperateurDivise=new JButton("/");
  bEfface=new JButton("Efface");
  bPoint=new JButton(",");
  bSigne=new JButton("+/-");
  bEgal=new JButton("=");


  // Creation et configuration de l'afficheur :
  afficheur=new JTextField(10);
  afficheur.setText("[init]");
  afficheur.setEditable(false);
  afficheur.setHorizontalAlignment(JTextField.TRAILING);
  Font engras=new Font("Sans Serif", Font.BOLD, 18);
  afficheur.setFont(engras);
```

```
  // Placement des boutons :
  layout=new GridBagContainLayout();
  getContentPane().setLayout(layout);
  layout.constrain(getContentPane(), afficheur, 0, 0, 4, 1, 2);
  layout.constrain(getContentPane(), bEfface, 0, 1, 2, 1, 2);
  layout.constrain(getContentPane(), bOperateurDivise, 2, 1, 1, 1, 2);
  layout.constrain(getContentPane(), bOperateurMultip, 3, 1, 1, 1, 2);
  layout.constrain(getContentPane(), bTouche[7], 0, 2, 1, 1, 2);
  layout.constrain(getContentPane(), bTouche[8], 1, 2, 1, 1, 2);
  layout.constrain(getContentPane(), bTouche[9], 2, 2, 1, 1, 2);
  layout.constrain(getContentPane(), bOperateurMoins, 3, 2, 1, 1, 2);
  layout.constrain(getContentPane(), bTouche[4], 0, 3, 1, 1, 2);
  layout.constrain(getContentPane(), bTouche[5], 1, 3, 1, 1, 2);
  layout.constrain(getContentPane(), bTouche[6], 2, 3, 1, 1, 2);
  layout.constrain(getContentPane(), bOperateurPlus, 3, 3, 1, 1, 2);
  layout.constrain(getContentPane(), bTouche[1], 0, 4, 1, 1, 2);
  layout.constrain(getContentPane(), bTouche[2], 1, 4, 1, 1, 2);
  layout.constrain(getContentPane(), bTouche[3], 2, 4, 1, 1, 2);
  layout.constrain(getContentPane(), bTouche[0], 0, 5, 1, 1, 2);
  layout.constrain(getContentPane(), bPoint, 1, 5, 1, 1, 2);
  layout.constrain(getContentPane(), bSigne, 2, 5, 1, 1, 2);
  layout.constrain(getContentPane(), bEgal, 3, 4, 1, 2, 2);


  // Autorise le traitement des clics sur les boutons :
  gSouris=new CalcGestionSouris();
  bEfface.addMouseListener(gSouris);
  bOperateurDivise.addMouseListener(gSouris);
  bOperateurMultip.addMouseListener(gSouris);
  bOperateurMoins.addMouseListener(gSouris);
  bOperateurPlus.addMouseListener(gSouris);
  bPoint.addMouseListener(gSouris);
  bSigne.addMouseListener(gSouris);
  bEgal.addMouseListener(gSouris);
  for(i=0; i<10; i++) bTouche[i].addMouseListener(gSouris);


  // Raccourcis clavier pour taper directement les commandes au clavier :
  gClavier=new CalcGestionClavier();
  for(i=0; i<10; i++)
     bTouche[i].registerKeyboardAction(gClavier, "Presse", KeyStroke.getKeyStroke((char)('0'+i)), JComponent.WHEN_IN_FOCUSED_WINDOW);
  bOperateurDivise.registerKeyboardAction(gClavier, "Presse", KeyStroke.getKeyStroke('/'), JComponent.WHEN_IN_FOCUSED_WINDOW);
  bOperateurMultip.registerKeyboardAction(gClavier, "Presse", KeyStroke.getKeyStroke('*'), JComponent.WHEN_IN_FOCUSED_WINDOW);
  bOperateurPlus.registerKeyboardAction(gClavier, "Presse", KeyStroke.getKeyStroke('+'), JComponent.WHEN_IN_FOCUSED_WINDOW);
  bOperateurMoins.registerKeyboardAction(gClavier, "Presse", KeyStroke.getKeyStroke('-'), JComponent.WHEN_IN_FOCUSED_WINDOW);
  bEfface.registerKeyboardAction(gClavier, "Press1", KeyStroke.getKeyStroke(KeyEvent.VK_DELETE, 0), JComponent.WHEN_IN_FOCUSED_WINDOW);
  bEfface.registerKeyboardAction(gClavier, "Press2", KeyStroke.getKeyStroke(KeyEvent.VK_BACK_SPACE, 0), JComponent.WHEN_IN_FOCUSED_WINDOW);
  bEgal.registerKeyboardAction(gClavier, "Presse", KeyStroke.getKeyStroke(KeyEvent.VK_ENTER, 0), JComponent.WHEN_IN_FOCUSED_WINDOW);
  bEgal.registerKeyboardAction(gClavier, "Presse", KeyStroke.getKeyStroke('='), JComponent.WHEN_IN_FOCUSED_WINDOW);
  bPoint.registerKeyboardAction(gClavier, "Press1", KeyStroke.getKeyStroke('.'), JComponent.WHEN_IN_FOCUSED_WINDOW);
  bPoint.registerKeyboardAction(gClavier, "Press2", KeyStroke.getKeyStroke(','), JComponent.WHEN_IN_FOCUSED_WINDOW);


  // Definition de la barre de menu et de l'unique menu Fichier :
  menuBar=new JMenuBar();
  menuFichier=new JMenu("Fichier");
  menuAffichage=new JMenu("Affichage");


  // Definition et gestion des clics sur les items de menu :
  itemArbreOn=new JMenuItem("Afficher l'arbre");
  itemArbreOn.addActionListener(this);

  itemArbreOff=new JMenuItem("Effacer l'arbre");
  itemArbreOff.setEnabled(false);
  itemArbreOff.addActionListener(this);

  itemQuitter=new JMenuItem("Quitter");
  itemQuitter.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_Q, ActionEvent.CTRL_MASK));
  itemQuitter.addActionListener(this);


  // Ajout des items aux menus :
  menuAffichage.add(itemArbreOn);
  menuAffichage.add(itemArbreOff);
  menuFichier.add(itemQuitter);


  // Ajout des menus a la fenetre :
  menuBar.add(menuFichier);
  menuBar.add(menuAffichage);
  setJMenuBar(menuBar);


  // Creation de l'objet calculatrice virtuelle :
  calcman=new CalcSys();
  afficheur.setText(String.valueOf(calcman.getValCourante()));

  // Cree et configure la fenetre de l'arbre (sans l'afficher) :
  fenArbre=new CalcTree(this);
  fenArbre.pack();
}


// Gestion des evenements du type clic sur un item de menu :
public void actionPerformed(ActionEvent e)
{
  if(e.getSource()==itemArbreOn)
  {
    // Affichage de l'arbre...
    fenArbre.setLocation( (int)(this.getX()+this.getWidth()*1.1), this.getY() );
    fenArbre.setSize(getSize());
    setArbreVisible(true);
  }

  if(e.getSource()==itemArbreOff)
  {
    // Effacement de l'arbre...
    setArbreVisible(false);
  }
```

```
    if(e.getSource()==itemQuitter)
      System.exit(0);
  }


  // Affiche l'arbre :
  public void setArbreVisible(boolean flag)
  {
    fenArbre.setVisible(flag);
    itemArbreOn.setEnabled(!flag);
    itemArbreOff.setEnabled(flag);
  }


  // Renvoi l'arborescence des calculs pour affichage par exemple :
  public DefaultMutableTreeNode getArbreCalculs()
  {
    return calcman.getArbreCalculs();
  }

};
```

CalcSys.java • Calculatrice 'virtuelle' : unité arithmétique

```
import java.lang.String;
import javax.swing.tree.DefaultMutableTreeNode;

public class CalcSys
{
  String operande;
  String valeur;

  private boolean virgule;
  private boolean verrouillage;
  private boolean touchegal;
  private boolean arbrevide;
  private boolean erreur;
  private int nchiffres;
  private int ioperation;
  private int voperation;

  private final int MAXCHIFFRES=14;
  public static final int PASDOPERATION=0;
  public static final int MULTIPLICATION=1;
  public static final int DIVISION=2;
  public static final int ADDITION=3;
  public static final int SOUSTRACTION=4;

  // Stockage de l'arborescence des calculs :
  DefaultMutableTreeNode racine;
  DefaultMutableTreeNode dernier;


  public CalcSys()
  {
    valeur=new String("");
    operande=new String("");
    clear();
  }

  // Efface completement l'entree :
  public void clear()
  {
    recommence();
    verrouillage=false;
    touchegal=false;
    zero();
  }

  // Efface uniquement l'entree en cours (valeur) :
  public void zero()
  {
    valeur="0";
    virgule=false;
    nchiffres=0;
  }

  // Recommence un calcul :
  public void recommence()
  {
    erreur=false;
    operande="0";
    ioperation=PASDOPERATION;
    voperation=PASDOPERATION;
    dernier=null;
    arbrevide=true;
    racine=new DefaultMutableTreeNode("(vide)");
  }

  // Enregistre la valeur en cours et l'operation a effectuer :
  public void operation(int typeOperation)
  {
    if(erreur==true) return;
    touchegal=false;
    if(typeOperation!=PASDOPERATION)
    {
      if(voperation!=PASDOPERATION)
      {
        // Il y a d'abord une operation a faire :
        elemcalcul();
        enregistreOperation(voperation);
        verrouillage=true;
        voperation=PASDOPERATION;
      }
      else
      {
        if(verrouillage==false) operande=valeur;
      }
      ioperation=typeOperation;
    }
  }

  // Execute l'operation et affiche le resultat :
  public void calcul()
  {
    if(erreur==true) return;
    touchegal=true;
    if(voperation!=PASDOPERATION)
    {
      // Une operation normale doit etre effectuee :
      enregistreOperation(voperation);
      elemcalcul();
      verrouillage=true;
      voperation=PASDOPERATION;
    }
    else if(ioperation!=PASDOPERATION)
    {
      // Execute l'operation sur la valeur elle-meme :
      enregistreOperation(ioperation);
      elemcalcul();
      verrouillage=true;
    }
  }


  // Enregistre l'operation qui vient d'etre effectuee dans l'arbre :
  private void enregistreOperation(int operation)
  {
    String chaineOperateur;
    switch(operation)
    {
      case MULTIPLICATION: chaineOperateur="*"; break;
      case DIVISION: chaineOperateur="/"; break;
      case ADDITION: chaineOperateur="+"; break;
      case SOUSTRACTION: chaineOperateur="-"; break;
      default: chaineOperateur="!"; break;
    }

    if(arbrevide==true)
    {
      arbrevide=false;
      racine=new DefaultMutableTreeNode(operande);
    }
    DefaultMutableTreeNode noeudOperation=
       new DefaultMutableTreeNode(chaineOperateur);
    DefaultMutableTreeNode feuilleValeur=
       new DefaultMutableTreeNode(valeur);
    noeudOperation.add(racine);
    noeudOperation.add(feuilleValeur);
    if(dernier==null) dernier=feuilleValeur;
    racine=noeudOperation;
  }


  // Ne fait QUE l'operation demandee :
  public void elemcalcul()
  {
    switch(ioperation)
    {
      case MULTIPLICATION:
        operande=String.valueOf( Double.valueOf(operande).doubleValue()
                               * Double.valueOf(valeur).doubleValue());
        break;

      case DIVISION:
      {
        // Empeche la division par zero :
        if(Double.valueOf(valeur).doubleValue()==0.0)
        {
          operande="Erreur";
          erreur=true;
        }
        else operande=String.valueOf(
                    Double.valueOf(operande).doubleValue()
                  / Double.valueOf(valeur).doubleValue() );
        break;
      }

      case ADDITION:
        operande=String.valueOf( Double.valueOf(operande).doubleValue()
                               + Double.valueOf(valeur).doubleValue());
        break;

      case SOUSTRACTION:
        operande=String.valueOf( Double.valueOf(operande).doubleValue()
                               - Double.valueOf(valeur).doubleValue());
        break;
    }
    valideOperande();
  }
```

```
  // Saisie d'un chiffre :
  public void addChiffre(int chiffre)
  {
    verificationSaisie();

    // Ne prend pas en compte l'ajout successif de 0 tant
    // qu'il n'y a pas de virgule :
    if(virgule==false && Double.valueOf(valeur).longValue()==0)
      valeur=String.valueOf(chiffre);
    else
    {
      // Ajoute les chiffres docilement jusqu'a la limite interne :
      if(nchiffres<=MAXCHIFFRES)
      {
        valeur=valeur+String.valueOf(chiffre);
        nchiffres++;
      }
    }
  }


  // Saisie de la virgule (ou point) :
  public void addVirgule()
  {
    verificationSaisie();

    // Ajoute la virgule si celle-ci n'y est pas encore ET
    // qu'on peut ajouter des chiffres :
    if(virgule==false && nchiffres<=MAXCHIFFRES)
    {
      virgule=true;
      valeur=valeur+".";
    }
  }


  // Saisie du signe - ou inversement :
  public void invSigne()
  {
    verificationSaisie();

    // Inverse le signe, uniquement si la valeur est NON nulle :
    if(valeur.startsWith("-"))
      valeur=valeur.substring(1);
    else
      if(Double.valueOf(valeur).doubleValue()>0.0) valeur="-" + valeur;
  }


  // Execute a chaque saisie :
  public void verificationSaisie()
  {
    if(verrouillage=true && touchegal==true)
    {
      touchegal=false;
      recommence();
      verrouillage=false;
      zero();
    }
    else if(ioperation!=PASDOPERATION && voperation==PASDOPERATION)
    {
      // L'operation entree est validee :
      zero();
      voperation=ioperation;
      verrouillage=false;
    }
  }

  // Reecrit proprement l'operande :
  private void valideOperande()
  {
    if(operande.endsWith(".0"))
    {
      // L'operande est entiere, supprimons ce .0 pas tres joli :
      operande=String.valueOf(Double.valueOf(operande).intValue());
    }
  }

  // Renvoie la valeur de la calculatrice virtuelle a afficher :
  public String getValCourante()
  {
    if(verrouillage==true) return operande;
    else return valeur;
  }

  // Renvoie l'arborescence des calculs pour affichage par exemple :
  public DefaultMutableTreeNode getArbreCalculs()
  {
    return racine;
  }

  // Renvoi du dernier noeud ajoute pour affichage :
  public DefaultMutableTreeNode getDerniereValeur()
  {
    return dernier;
  }

};
```

CalcTree.java • Arbre des calculs effectués

```
import javax.swing.*;
import java.awt.*;
import java.io.*;
import javax.swing.tree.*;
import java.awt.event.*;


public class CalcTree extends JFrame
{
  // Objets de la fenetre :
  private JTree arbre;
  private JScrollPane scroller;

  // Garde un lien vers la calculatrice mere :
  private CalcGUI calculatrice;


  // Constructeur : initialisation de la fenetre :
  public CalcTree(CalcGUI macalculatrice)
  {
    super("Arbre");

    calculatrice=macalculatrice;

    // Creation du JTree :
    arbre=new JTree(calculatrice.getArbreCalculs());
    arbre.setPreferredSize(new Dimension(200, 200));

    // Cree le scrollpane et insere le JTree dedans :
    scroller=new JScrollPane(arbre);

    // Ajoute le scrollpane a la fenetre :
    getContentPane().add(scroller, BorderLayout.CENTER);

    // Gere l'item de menu dans CalcGUI quand on ferme la fenetre de l'arbre :
    addWindowListener(new WindowAdapter()
    {
      public void windowClosing(WindowEvent e)
      {
        calculatrice.setArbreVisible(false);
      }
    });
  }

  public void affiche(DefaultMutableTreeNode noeud)
  {
    // Regenere l'arborescence a partir de l'arbre stocke :
    DefaultTreeModel modele=new DefaultTreeModel(calculatrice.getArbreCalculs());
    arbre.setModel(modele);

    // Deploit l'arborescence completement :
    if(noeud!=null) arbre.scrollPathToVisible(new TreePath(noeud.getPath()));
  }

};
```