Tableur.java • Fenêtre principale de l'application et fonction *main*

```
import javax.swing.*;
import java.awt.*;
import java.io.*;
import java.awt.event.*;
import javax.swing.tree.*;
import javax.swing.table.*;
import java.util.Vector;
import javax.swing.event.TableModelEvent;


import calculatrice.*;
import calculatrice.event.*;


public class Tableur extends JFrame implements CalcListener
{
  private class CalcGestionSouris extends MouseAdapter
  {
    // Gestion des evenements du type clic sur un des boutons de la calculatrice :
    public void mouseReleased(MouseEvent e)
    {
      if(e.getSource()==bNouveau)
      {
        actionNouveau();
      }

      if(e.getSource()==bOuvrir)
      {
        actionOuvrir();
      }

      if(e.getSource()==bEnregistrer)
      {
        actionEnregistrerSous();
      }

      if(e.getSource()==bQuitter)
      {
        System.exit(0);
      }
    }
  };


  public class GrilleFileFilter extends javax.swing.filechooser.FileFilter
  {
    public boolean accept(File f)
    {
      return f.getName().toLowerCase().endsWith(".grd")
          || f.getName().toLowerCase().endsWith(".tab")
          || f.getName().toLowerCase().endsWith(".txt")
          || f.isDirectory();
    }

    public String getDescription()
    {
      return "Fichier grille";
    }
  };


  // Barre d'outils :
  private JButton bNouveau;
  private JButton bOuvrir;
  private JButton bEnregistrer;
  private JButton bInserer;
  private JButton bQuitter;
  private JToolBar barre;

  // Titre de l'application :
  private final String titreApplication= "Mini-Tableur";
  private String nomFichierGrille;

  // Calculatrice :
  private CalcBean calto;

  // Grille du tableur :
  private int nColonnes,nLignes;
  private DefaultTableModel modele;
  private Vector entetesColonnes;
  private JTable grille;
  private JScrollPane defilement;
  private int dernierRow=-1;
  private int dernierColumn=-1;

  // Gestionnaire des evenements de la souris :
  private CalcGestionSouris gsouris;


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

    int i,j;
```

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

  // Creation de la JToolBar :
  barre=new JToolBar();
  bNouveau=new JButton( "Nouveau", new ImageIcon("img/tnew.gif", "Nouveau"));
  bOuvrir=new JButton( "Ouvrir", new ImageIcon("img/topen.gif", "Ouvrir"));
  bEnregistrer=new JButton( "Enregistrer", new ImageIcon("img/tsave.gif", "Enregistrer"));
  bQuitter=new JButton( "Quitter", new ImageIcon("img/tquit.gif", "Quitter"));

  bNouveau.setToolTipText("Commencer une nouvelle grille");
  bOuvrir.setToolTipText("Ouvrir un fichier");
  bEnregistrer.setToolTipText("Enregistrer la grille");
  bQuitter.setToolTipText("Quitter le tableur");

  barre.add(bNouveau);
  barre.add(bOuvrir);
  barre.add(bEnregistrer);
  barre.add(bQuitter);

  // Creation et configuration de la grille :
  nColonnes=12;
  nLignes=100;
  entetesColonnes=new Vector();
  for(i=0; i<nColonnes; i++) entetesColonnes.add(String.valueOf(i+1));

  // Cree le modele de table vide :
  Vector lignevide,vide=new Vector();
  for(i=0; i<50; i++)
  {
    lignevide=new Vector();
    for(j=0; j<=nColonnes; j++) lignevide.add("");
    vide.add(lignevide);
  }

  modele=new DefaultTableModel(entetesColonnes, nLignes);
  grille=new JTable(modele);

  // Cree la magnifique calculatrice :
  calto=new CalcBean();

  // Place les elements sur la fenetre :
  defilement=new JScrollPane(grille);
  getContentPane().add(barre, BorderLayout.NORTH);
  getContentPane().add(defilement, BorderLayout.CENTER);
  getContentPane().add(calto, BorderLayout.EAST);

  // Autorise le traitement des clics sur les boutons :
  gsouris=new CalcGestionSouris();
  bNouveau.addMouseListener(gsouris);
  bOuvrir.addMouseListener(gsouris);
  bEnregistrer.addMouseListener(gsouris);
  bQuitter.addMouseListener(gsouris);

  // On ecoute les evenements de la calculatrice :
  calto.addCalcListener(this);

}

// Gestion centralisee des messages d'erreur - dans une boite de dialogue :
public void boiteErreur(String message)
{
  JOptionPane.showMessageDialog(this,message,"Erreur",JOptionPane.ERROR_MESSAGE);
}

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)
  {}

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

  // Lecture de la taille de l'ecran :
  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);
}

public void calcChanged(CalcEvent e)
{
  if(grille.getSelectedRow()>=0) dernierRow=grille.getSelectedRow();
  if(grille.getSelectedColumn()>=0) dernierColumn=grille.getSelectedColumn();

  // Au cas ou aucune cellule n'a ete selectionnee:
  if(dernierRow<0 || dernierColumn<0) return;

  // Si jamais la case est en cours d'edition, arrete l'edition...
  if(grille.getEditingRow()>=0 && grille.getEditingColumn()>=0) grille.getCellEditor().cancelCellEditing();

  // Puis place la valeur calculee par notre calto :
  modele.setValueAt(e.getResultat(), dernierRow, dernierColumn);
  modele.fireTableCellUpdated(dernierRow, dernierColumn);
}
```

```
  // Item Nouveau : reinitialise la grille completement :
  public void actionNouveau()
  {
    modele=new DefaultTableModel(entetesColonnes, nLignes);
    grille.setModel(modele);
    modele.newDataAvailable(new TableModelEvent(modele));
  }

  // Item enregistrer sous :
  public void actionEnregistrerSous()
  {
    // Demande le nom de fichier dans lequel sauver :
    String nomComplet;
    JFileChooser selecFichier=new JFileChooser();
    selecFichier.setDialogTitle("Enregistrer la grille sous...");
    GrilleFileFilter filtre=new GrilleFileFilter();
    selecFichier.setFileFilter(filtre);
    int retour=selecFichier.showSaveDialog(this);

    // Si l'utilisateur a valide le nom, garde le nom en memoire et sauve :
    if(retour==JFileChooser.APPROVE_OPTION)
    {
      // Recupere le nom du fichier choisi par l'utilisateur :
      nomFichierGrille=selecFichier.getSelectedFile().getName();
      nomComplet=selecFichier.getSelectedFile().getAbsolutePath();

      // Ajoute l'extension si celle-ci n'y est pas :
      if(!filtre.accept(selecFichier.getSelectedFile()))
      {
        nomFichierGrille=nomFichierGrille+".grd";
        nomComplet=nomComplet+".grd";
      }
      setTitle(titreApplication + " - " + nomFichierGrille);

      // Enregistrement normal :
      enregistrerGrille(nomComplet);
    }
  }


  // Enregistrement d'une grille du tableur
  private void enregistrerGrille(String cheminComplet)
  {
    try
    {
      FileOutputStream ostream = new FileOutputStream(cheminComplet);
      ObjectOutputStream p=new ObjectOutputStream(ostream);

      // Grace a la magie de la serialisation, cette simple fonction enregistre la
      // totalite de l'objet dans le fichier specifie !
      p.writeObject(modele);
      p.flush();
      ostream.close();
    }
    catch(FileNotFoundException except)
    {
      boiteErreur("Impossible d'ecrire dans le specifie. Verifier le fichier\nn'a pas ete protege "
                  +"et que vous avez les droits suffisants pour ecrire dans ce repertoire.");
    }
    catch(IOException except)
    {
      boiteErreur("Il y a eu un dysfonctionnement lors de l'enregistrement de la grille.");
    }
  }


  // Item Ouvrir - charge le contenu d'un objet grille :
  public void actionOuvrir()
  {
    // Selecteur de fichier pour choisir le fichier a lire :
    JFileChooser selecFichier=new JFileChooser();
    selecFichier.setDialogTitle("Ouvrir une grille...");
    selecFichier.setFileFilter(new GrilleFileFilter());
    int retour=selecFichier.showOpenDialog(this);

    if(retour==JFileChooser.APPROVE_OPTION)
    {
      // L'utilisateur a clique sur OK et a choisi un fichier :
      try
      {
        FileInputStream istream=new FileInputStream(selecFichier.getSelectedFile().getAbsolutePath());
        ObjectInputStream p=new ObjectInputStream(istream);

        // Recupere l'objet grace a la magie de la serialisation :
        modele=(DefaultTableModel)p.readObject();
        istream.close();
        grille.setModel(modele);
        // De nouvelles donnees ont ete chargees, il s'agit de rafraichir le controle :
        modele.newDataAvailable(new TableModelEvent(modele));
        // OK, on peut a present enregistrer la grille :
        nomFichierGrille=selecFichier.getSelectedFile().getName();
        setTitle(titreApplication + " - " + nomFichierGrille);
      }
      catch(FileNotFoundException except)
      {
        boiteErreur("Impossible de lire le fichier selectionne. Verifier que le fichier\nn'a pas ete "
                    + " efface et que vous possedez les droits suffisants pour le lire.");
      }
      catch(IOException except)
      {
        boiteErreur("Le fichier specifie n'a pas un format valide.\nAbandon de l'operation de chargement.");
      }
      catch(ClassNotFoundException except)
      {
        boiteErreur("Erreur interne lors du chargement du fichier.\nCes programmeurs ne valent pas un clou !");
      }
    }
  }
};
```

calculatrice/event/CalcEvent.java • Evènement généré par la calculatrice

```
package calculatrice.event;

// Evenement declenche par CalcBean
// Contient le resultat du calcul effectue...
public class CalcEvent extends java.util.EventObject
{
  private String resultat;

  public String toString()
  {
    return resultat;
  }

  public CalcEvent(String r, Object s)
  {
    super(s);
    resultat=r;
  }

  public String getResultat()
  {
    return resultat;
  }
};
```

calculatrice/event/CalcListener.java • Ecouteur pour les évènements de la calculatrice

```
package calculatrice.event;


// Interface des listeners de CalcEvent emis par CalcBean
public interface CalcListener extends java.util.EventListener
{
  void calcChanged(CalcEvent evt);
}
```

calculatrice/CalcBean.java • Contrôle indépendant de la calculatrice

```
package calculatrice;

import javax.swing.*;
import java.awt.*;
import java.io.*;
import java.awt.event.*;
import javax.swing.tree.*;
import java.util.Vector;
import java.util.Iterator;
import java.util.EventObject;

import calculatrice.event.*;


public class CalcBean extends JPanel implements Serializable
{

  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
  {
    public void actionPerformed(ActionEvent e)
    {
      gestionEvent(e);
    }
  };


  private class CalcGestionAfficheur extends ComponentAdapter
  {
    public void componentResized(ComponentEvent e)
    {
      Font nouvpolice=new Font("Sans Serif", Font.BOLD, afficheur.getHeight()/3);
      afficheur.setFont(nouvpolice);
    }
  };



  // 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()));
    // New : envoie un evenement a tous les listeners...
    notifyCalcChanged();
  }
}

// Listeners :
transient private Vector calcListeners=new Vector();


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

// Zone de texte afficheur :
transient private JTextField afficheur;

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

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

// Gestionnaire des evenements de la souris et du clavier :
transient private CalcGestionSouris gSouris;
transient private CalcGestionClavier gClavier;
transient private CalcGestionAfficheur gAfficheur;


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

  int i;

  // 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);
  gAfficheur=new CalcGestionAfficheur();
  afficheur.addComponentListener(gAfficheur);


  // Placement des boutons :
  layout=new GridBagContainLayout();
  setLayout(layout);
  layout.constrain(this, afficheur, 0, 0, 4, 1, 2);
  layout.constrain(this, bEfface, 0, 1, 2, 1, 2);
```

```
    layout.constrain(this, bOperateurDivise, 2, 1, 1, 1, 2);
    layout.constrain(this, bOperateurMultip, 3, 1, 1, 1, 2);
    layout.constrain(this, bTouche[7], 0, 2, 1, 1, 2);
    layout.constrain(this, bTouche[8], 1, 2, 1, 1, 2);
    layout.constrain(this, bTouche[9], 2, 2, 1, 1, 2);
    layout.constrain(this, bOperateurMoins, 3, 2, 1, 1, 2);
    layout.constrain(this, bTouche[4], 0, 3, 1, 1, 2);
    layout.constrain(this, bTouche[5], 1, 3, 1, 1, 2);
    layout.constrain(this, bTouche[6], 2, 3, 1, 1, 2);
    layout.constrain(this, bOperateurPlus, 3, 3, 1, 1, 2);
    layout.constrain(this, bTouche[1], 0, 4, 1, 1, 2);
    layout.constrain(this, bTouche[2], 1, 4, 1, 1, 2);
    layout.constrain(this, bTouche[3], 2, 4, 1, 1, 2);
    layout.constrain(this, bTouche[0], 0, 5, 1, 1, 2);
    layout.constrain(this, bPoint, 1, 5, 1, 1, 2);
    layout.constrain(this, bSigne, 2, 5, 1, 1, 2);
    layout.constrain(this, 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)), WHEN_IN_FOCUSED_WINDOW);
    bOperateurDivise.registerKeyboardAction(gClavier, "Presse", KeyStroke.getKeyStroke('/'), WHEN_IN_FOCUSED_WINDOW);
    bOperateurMultip.registerKeyboardAction(gClavier, "Presse", KeyStroke.getKeyStroke('*'), WHEN_IN_FOCUSED_WINDOW);
    bOperateurPlus.registerKeyboardAction(gClavier, "Presse", KeyStroke.getKeyStroke('+'), WHEN_IN_FOCUSED_WINDOW);
    bOperateurMoins.registerKeyboardAction(gClavier, "Presse", KeyStroke.getKeyStroke('-'), WHEN_IN_FOCUSED_WINDOW);
    bEfface.registerKeyboardAction(gClavier, "Presse1", KeyStroke.getKeyStroke(KeyEvent.VK_DELETE, 0), WHEN_IN_FOCUSED_WINDOW);
    bEfface.registerKeyboardAction(gClavier, "Presse2", KeyStroke.getKeyStroke(KeyEvent.VK_BACK_SPACE, 0), WHEN_IN_FOCUSED_WINDOW);
    bEgal.registerKeyboardAction(gClavier, "Presse1", KeyStroke.getKeyStroke(KeyEvent.VK_ENTER, 0), WHEN_IN_FOCUSED_WINDOW);
    bEgal.registerKeyboardAction(gClavier, "Presse", KeyStroke.getKeyStroke('='), WHEN_IN_FOCUSED_WINDOW);
    bPoint.registerKeyboardAction(gClavier, "Presse1", KeyStroke.getKeyStroke('.'), WHEN_IN_FOCUSED_WINDOW);
    bPoint.registerKeyboardAction(gClavier, "Presse2", KeyStroke.getKeyStroke(','), WHEN_IN_FOCUSED_WINDOW);

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


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


  // Ajout d'un listener :
  public synchronized void addCalcListener(CalcListener l)
  {
    calcListeners.add(l);
  }

  // Suppression d'un listener :
  public synchronized void removeCalcListener(CalcListener l)
  {
    calcListeners.removeElement(l);
  }

  // Emission d'un evenement :
  protected void notifyCalcChanged()
  {
    // Creation de l'evenelent :
    CalcEvent evenement=new CalcEvent(calcman.getValCourante(), this);

    // On copie le vecteur des listeners pour qu'il ne puissent etre modifies
    Vector copieListeners=(Vector)calcListeners.clone();

    // Emission de l'evenement vers les listeners :
    Iterator it=copieListeners.iterator();
    CalcListener element;
    while(it.hasNext())
    {
      element=(CalcListener)it.next();
      element.calcChanged(evenement);
    }
  }
};
```