# Ébauche d'un traitement de texte

## Introduction

Ce chapitre ne traite pas d'un nouveau concept en particulier. Il peut être considéré comme un tutorial visant à la création d'un *mini* traitement de texte. Ce traitement de texte pourra être amélioré par la suite grâce aux possibilités de Java, et de Swing pour la partie graphique, jusqu'à intégrer des fonctions telles que celles de JTE2 (téléchargeable dans le section PROGRAMMES du site JGFL { http://perso.wanadoo.fr/guillaume/ } ).
Ce serait pratiquement du suicide d'écrire un programme comme celui-ci avec un vulgaire bloc-notes . Par conséquent, je vous conseille d'utiliser un bon IDE, et de préférence JBuilder 3.5, car c'est avec lui que le projet a été réalisé. Pour le télécharger, en version Foundation, gratuite : http://www.borland.fr

Le code final de chaque classe est surligné en jaune.

## Plan de l'application

On peut créer un petit programme sans réfléchir, au *feeling*, mais un projet même s'il n'est pas très complexe doit posséder une bonne structure. On doit ainsi rassembler un certain nombre de données avant de commencer la création du projet :

- Fonctionnalités (méthodes à implémenter) :
  - Ouverture d'un fichier en texte brut
  - Enregistrement d'un fichier en texte brut
  - Créer un nouveau document
  - Quitter
  - Boîte de dialogue A Propos
- Interface graphique :
  - Composants *Swing*
  - Barre de menu
  - Barre d'icônes
  - Barre de tâches (statusbar)
  - Un composant dans lequel l'utilisateur peut entrer du texte : un *JTextPane*
  - Des barres de défilement assignées à ce JTextPane
  - Boîte de dialogue A Propos sera une instance de JFrame
- Organisation fonctionnelle
  - Partage des classes :
    - Une classe pour lancer le programme : **Application1.java**
    - Une classe pour l'interface graphique : **Cadre1.java**
    - Une classe pour ouvrir des fichiers : **Ouvrir.java**
    - Une classe pour enregistrer des fichiers : **Enregistrer.java**
    - Une classe pour définir la boîte de dialogue A Propos : **Cadre1_AboutBox.java**

## Application1.java

Application1.java est le fichier contenant la fonction *main()* généré par défaut par JBuilder quand un nouveau projet est créé. Voici le code source généré par JBuilder, sans aucune modification, suivi de commentaires :

```
/**
 * Titre : Ebauche de traitement de texte<p>
 * Description : Code source du programme de traitement de texte présenté dans le
cours JGFL {Juin 2000}<p>
 * Copyright : Copyright (c) Guillaume Florimond<p>
 * Société : JGFLsoft<p>
```

```
 * @author Guillaume Florimond
 * @version 1.0
 */
package texte;

import javax.swing.UIManager;
import java.awt.*;

public class Application1 {
  boolean packFrame = false;

  //Construire l'application
  public Application1() {
    Cadre1 frame = new Cadre1();
    //Valider les cadres ayant des tailles prédéfinies
    //Compacter les cadres ayant des infos de taille préférées - ex. depuis leur
disposition
    if (packFrame) {
      frame.pack();
    }
    else {
      frame.validate();
    }
    //Centrer la fenêtre
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = frame.getSize();
    if (frameSize.height > screenSize.height) {
      frameSize.height = screenSize.height;
    }
    if (frameSize.width > screenSize.width) {
      frameSize.width = screenSize.width;
    }
    frame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
    frame.setVisible(true);
  }

  //Méthode principale
  public static void main(String[] args) {
    /*
    try {
      UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch(Exception e) {
      e.printStackTrace();
    }
    */
    new Application1();
  }
}
```

Les commentaires du début sont automatiques. Le package dans lequel seront regroupés toutes les classes du projet est *texte*.
La classe crée une nouvelle instance de *Cadre1.java* nommée *frame*.
Le code suivant cette déclaration permet de centrer la fenêtre à l'écran, comme l'indique le commentaire. C'est une fonction de JBuilder, ce code est écrit automatiquement et il n'est pas nécessaire dans l'immédiat que vous le compreniez.
La méthode *main()* fait appel au constructeur de la classe (*new Application1() ;*) et assigne un Look & Feel. Par défaut JBuilder active le code compris entre les blocs *try* et *catch* de ma méthode *main()*. Ceci a pour effet d'assigner l'apparence Windows 95 au programme Java. Si ces instructions sont mises en remarque (donc ignorées par le compilateur), l'apparence devient automatiquement Metal (celle par défaut de Java).
Si vous utilisez JBuilder, vous n'aurez pratiquement jamais à vous occuper de la classe Appllication1.

# Cadre1.java

La classe Cadre1.java représente le gros de l'application. Comme son nom l'indique, il étend de JFrame. On peut le visionner et ajouter des composants par l'intermédiaire du Concepteur de JBuilder.
Voici le code de base de ce fichier (code généré par JBuilder par défaut) :

```
package texte;

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

public class Cadre1 extends JFrame {

  public Cadre1() {
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }

  private void jbInit() throws Exception {
  }
}
```

Le constructeur du cadre (méthode exécutée automatiquement lors de la création d'un nouveau objet Cadre1 par Application1) appelle la méthode jbInit() en *catchant* les exceptions de type *printStackTrace* dont nous n'avons pas besoin d'en savoir plus. La méthode jbInit() n'est pas une méthode typique de Java (comme main() pour les applications, init() pour les applets…) ; elle est systématiquement générée par JBuilder et vous vous y habituerez rapidement. C'est dans cette méthode que sont ajoutés au cadre tous les composants graphiques. Ainsi, si l'on utilise l'assistant de JBuilder pour créer Cadre1.java et sélectionnant les options générer : barre de menus, de statut, d'icônes…, le code automatiquement écrit est le suivant :

```
/**
 * Titre : Ebauche de traitement de texte<p>
 * Description : Code source du programme de traitement de texte présenté dans le
cours JGFL {Juin 2000}<p>
 * Copyright : Copyright (c) Guillaume Florimond<p>
 * Société : JGFLsoft<p>
 * @author Guillaume Florimond
 * @version 1.0
 */
package texte;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Cadre1 extends JFrame {
  JPanel contentPane;
  JMenuBar menuBar1 = new JMenuBar();
  JMenu menuFile = new JMenu();
  JMenuItem menuFileExit = new JMenuItem();
  JMenu menuHelp = new JMenu();
  JMenuItem menuHelpAbout = new JMenuItem();
  JToolBar toolBar = new JToolBar();
  JButton jButton1 = new JButton();
  JButton jButton2 = new JButton();
  JButton jButton3 = new JButton();
  ImageIcon image1;
  ImageIcon image2;
  ImageIcon image3;
  JLabel statusBar = new JLabel();
  BorderLayout borderLayout1 = new BorderLayout();

  //Construire le cadre
  public Cadre1() {
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
```

```
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }

  //Initialiser le composant
  private void jbInit() throws Exception  {
    image1 = new ImageIcon(Cadre1.class.getResource("openFile.gif"));
    image2 = new ImageIcon(Cadre1.class.getResource("closeFile.gif"));
    image3 = new ImageIcon(Cadre1.class.getResource("help.gif"));
    contentPane = (JPanel) this.getContentPane();
    contentPane.setLayout(borderLayout1);
    this.setSize(new Dimension(400, 300));
    this.setTitle("Traitement de texte - Cours JGFL - ");
    statusBar.setText(" ");
    menuFile.setText("Fichier");
    menuFileExit.setText("Quitter");
    menuFileExit.addActionListener(new Cadre1_menuFileExit_ActionAdapter(this));
    menuHelp.setText("Aide");
    menuHelpAbout.setText("A propos");
    menuHelpAbout.addActionListener(new Cadre1_menuHelpAbout_ActionAdapter(this));
    jButton1.setIcon(image1);
    jButton1.setToolTipText("Ouvrir un fichier");
    jButton2.setIcon(image2);
    jButton2.setToolTipText("Fermer le fichier");
    jButton3.setIcon(image3);
    jButton3.setToolTipText("Aide");
    toolBar.add(jButton1);
    toolBar.add(jButton2);
    toolBar.add(jButton3);
    menuFile.add(menuFileExit);
    menuHelp.add(menuHelpAbout);
    menuBar1.add(menuFile);
    menuBar1.add(menuHelp);
    this.setJMenuBar(menuBar1);
    contentPane.add(toolBar, BorderLayout.NORTH);
    contentPane.add(statusBar, BorderLayout.SOUTH);
  }

  //Opération Fichier | Quitter effectuée
  public void fileExit_actionPerformed(ActionEvent e) {
    System.exit(0);
  }

  //Opération Aide | A propos effectuée
  public void helpAbout_actionPerformed(ActionEvent e) {
    Cadre1_AboutBox dlg = new Cadre1_AboutBox(this);
    Dimension dlgSize = dlg.getPreferredSize();
    Dimension frmSize = getSize();
    Point loc = getLocation();
    dlg.setLocation((frmSize.width - dlgSize.width) / 2 + loc.x, (frmSize.height -
dlgSize.height) / 2 + loc.y);
    dlg.setModal(true);
    dlg.show();
  }

  //Remplacé, ainsi nous pouvons sortir quand la fenêtre est fermée
  protected void processWindowEvent(WindowEvent e) {
    super.processWindowEvent(e);
    if (e.getID() == WindowEvent.WINDOW_CLOSING) {
      fileExit_actionPerformed(null);
    }
  }
}

class Cadre1_menuFileExit_ActionAdapter implements ActionListener {
  Cadre1 adaptee;

  Cadre1_menuFileExit_ActionAdapter(Cadre1 adaptee) {
    this.adaptee = adaptee;
```

```
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.fileExit_actionPerformed(e);
  }
}

class Cadre1_menuHelpAbout_ActionAdapter implements ActionListener {
  Cadre1 adaptee;

  Cadre1_menuHelpAbout_ActionAdapter(Cadre1 adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.helpAbout_actionPerformed(e);
  }
}
```

Dans jbInit() est écrit le code nécessaire pour afficher les composants dans le cadre. 3 images sont assignées aux 3 boutons de la barre d'icônes, ces images sont fournies par défaut avec JBuilder.

Le code correspondant à l'appel de la boîte de dialogue Cadre1_AboutBox et celui correspondant à Fichier > Quitter sont écrits automatiquement. Les 2 *inner*-classes crées en fin de programme correspondent aux gestionnaires d'événement associée à Fichier > Quitter et à l'appel de Cadre1_AboutBox.
Ce code est dit à *Adapteur Standard*, par opposition du code à *Adapteur Anonyme*. Le résultat est de toute façon le même, seule change la manière d'écrire le code. Pour enclencher le mode *Adapteur Standard*, cocher la case appropriée, dans JBuilder en passant par **Projet > Propriétés du projet >** (onglet) **Style de Code**.
Vous n'aurez le plus souvent pas à vous soucier de ce code (c'est pourquoi il n'est pas expliqué ici) car il est automatique pour JBuilder et le programmeur n'a pas à le modifier de quelque manière que se soit.
Pour plus d'informations sur les gestionnaires d'évènements, référez-vous au chapitre sur Swing du cours JGFL.
Il faut remarquer l'instruction qui permet de fermer la fenêtre lors du clic sur la croix (X) sous Windows ou UNIX : **System.exit(0) ;**

Le pointeur *this* fait ici référence à Cadre1, donc à une fenêtre dans laquelle on peut ajouter des composants, à laquelle on peut assigner un titre, une taille… (setTitle(), setSize()…). On remarquera que si l'on ajoute des composants par l'intermédiaire de **this.add(Component) ;** une erreur est déclenchée quand le programme est lancé (le compilateur ne voit rien) et un message nous dit d'utiliser à la place **this.getContentPane().add() ;** ce qui revient à **contentPane().add() ;** dans notre cas car *contentPane* est le nom du premier Panel (conteneur de composants, cf. chapitre sur Swing) de Cadre1.

On remarquera aussi que le code qui permet de centrer la fenêtre à l'écran est réutilisé pour Cadre1_AboutBox.

Nous devons ensuite ajouter des éléments à la barre de menu et à la barre d'icônes précédemment créées et leur assigner des gestionnaires d'événement.
Mais avant nous devons nous référer à la structure de notre programme :

- L'utilisateur clique sur un bouton (Enregistrer ou Ouvrir)
- Le gestionnaire d'événement associé à ce bouton est actionné
- Ce gestionnaire d'événement contient un appel à une méthode de la classe Cadre1 :
  - Méthode ouvrir() pour Fichier > Ouvrir
  - Méthode enregistrer() pour Fichier > Enregistrer
- Ces méthodes créent un JFileChooser (dialogue de choix de fichier) puis appellent les méthodes save() de le classe Enregistrer() ou open() de la classe Ouvrir en fournissant comme argument le nom du fichier renvoyé par le JFileChooser.

Voici l'implémentation de ces méthodes ouvrir() et enregistrer() :

```
public void ouvrir() {
  // création d'une boîte de dialogue de choix de fichier
  JFileChooser filechoose = new JFileChooser();
  // répertoire courant (à l'ouverture) est celui d'exec. du programme
  filechoose.setCurrentDirectory(new File("."));
```

```
  // label du bouton de validation
  String approve = new String("OUVRIR");
  // si ce bouton est actionné:
  int resultatEnregistrer = filechoose.showDialog(filechoose, approve);
  if (resultatEnregistrer == JFileChooser.APPROVE_OPTION) {
    Ouvrir.open(filechoose.getSelectedFile());
  }
  }

  public void enregistrer() {
  // création d'une boîte de dialogue de choix de fichier
  JFileChooser filechoose = new JFileChooser();
  // répertoire courant (à l'ouverture) est celui d'exec. du programme
  filechoose.setCurrentDirectory(new File("."));
  // label du bouton de validation
  String approve = new String("ENREGISTRER");
  // si ce bouton est actionné:
  int resultatEnregistrer = filechoose.showDialog(filechoose, approve);
  if (resultatEnregistrer == JFileChooser.APPROVE_OPTION) {
    Enregistrer.save(filechoose.getSelectedFile());
  }
  }
```

Ces méthodes ouvrir() et enregistrer() de la classe Cadre1 sont aussi appelées quand l'utilisateur clique sur les boutons Ouvrir ou Enregistrer de la barre d'icônes.

Nous disposons maintenant d'une implémentation complète de Cadre1. C'est la version finale de cette classe :

```
/**
 * Titre : Ebauche de traitement de texte<p>
 * Description : Code source du programme de traitement de texte présenté dans le
cours JGFL {Juin 2000}<p>
 * Copyright : Copyright (c) Guillaume Florimond<p>
 * Société : JGFLsoft<p>
 * @author Guillaume Florimond
 * @version 1.0
 */


package texte;

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

public class Cadre1 extends JFrame {
  // créé automatiquement par JBuilder
  JPanel contentPane;
  // barre de menu
  JMenuBar menuBar1 = new JMenuBar();
  // un menu dans cette barre: FICHIER
  JMenu menuFile = new JMenu();
  JMenuItem menuFileExit = new JMenuItem();
  JMenu menuHelp = new JMenu();
  JMenuItem menuHelpAbout = new JMenuItem();
  // barre d'icônes...
  JToolBar toolBar = new JToolBar();
  JButton jButton1 = new JButton();
  JButton jButton2 = new JButton();
  JButton jButton3 = new JButton();
  // icônes de la barre d'icônes
  ImageIcon image1;
  ImageIcon image2;
  ImageIcon image3;
  // la barre d'état doit être statique: on y fait référence
  // dans les classes Ouvrir et Enregistrer
```

```
  public static JLabel statusBar = new JLabel();
  BorderLayout borderLayout1 = new BorderLayout();
  // panneau de défilement associé à notre JTextPane
  JScrollPane jScrollPane1 = new JScrollPane();
  public static JTextPane jTextPane1 = new JTextPane();
  JMenuItem jMenuItem1 = new JMenuItem();
  JMenuItem jMenuItem2 = new JMenuItem();
  JMenuItem jMenuItem3 = new JMenuItem();

  //Construire le cadre
  public Cadre1() {
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
    try {
      // exécutre la méthode jbInit()
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }


  public void ouvrir() {
  // création d'une boîte de dialogue de choix de fichier
  JFileChooser filechoose = new JFileChooser();
  // répertoire courant (à l'ouverture) est celui d'exec. du programme
  filechoose.setCurrentDirectory(new File("."));
  // label du bouton de validation
  String approve = new String("OUVRIR");
  // si ce bouton est actionné:
  int resultatEnregistrer = filechoose.showDialog(filechoose, approve);
  if (resultatEnregistrer == JFileChooser.APPROVE_OPTION) {
    Ouvrir.open(filechoose.getSelectedFile());
  }
  }

  public void enregistrer() {
  // création d'une boîte de dialogue de choix de fichier
  JFileChooser filechoose = new JFileChooser();
  // répertoire courant (à l'ouverture) est celui d'exec. du programme
  filechoose.setCurrentDirectory(new File("."));
  // label du bouton de validation
  String approve = new String("ENREGISTRER");
  // si ce bouton est actionné:
  int resultatEnregistrer = filechoose.showDialog(filechoose, approve);
  if (resultatEnregistrer == JFileChooser.APPROVE_OPTION) {
    Enregistrer.save(filechoose.getSelectedFile());
  }
  }


  //Initialiser le composant
  private void jbInit() throws Exception  {
    // quels fichiers représentent ces icônes ?
    image1 = new ImageIcon(Cadre1.class.getResource("openFile.gif"));
    image2 = new ImageIcon(Cadre1.class.getResource("closeFile.gif"));
    image3 = new ImageIcon(Cadre1.class.getResource("help.gif"));
    // voir remarque dans le cours
    contentPane = (JPanel) this.getContentPane();
    // ajout d'un layout
    contentPane.setLayout(borderLayout1);
    // taille de la fenêtre
    this.setSize(new Dimension(400, 300));
    // titre de la fenêtre
    this.setTitle("Traitement de texte - Cours JGFL - ");
    // inscrit dans la barre d'état
    statusBar.setText(" ");
    // "caption" des menus
    menuFile.setText("Fichier");
    menuFileExit.setText("Quitter");
    // assigner un gestionnaire d'evènement...
    menuFileExit.addActionListener(new Cadre1_menuFileExit_ActionAdapter(this));
    menuHelp.setText("Aide");
```

```
    menuHelpAbout.setText("A propos");
    menuHelpAbout.addActionListener(new Cadre1_menuHelpAbout_ActionAdapter(this));
    jButton1.setIcon(image1);
    jButton1.addActionListener(new Cadre1_jButton1_actionAdapter(this));
    jButton1.setToolTipText("Ouvrir un fichier");
    jButton2.setIcon(image2);
    jButton2.addActionListener(new Cadre1_jButton2_actionAdapter(this));
    jButton2.setToolTipText("Fermer le fichier");
    jButton3.setIcon(image3);
    jButton3.addActionListener(new Cadre1_jButton3_actionAdapter(this));
    jButton3.setToolTipText("Aide");
    jMenuItem1.setText("Nouveau");
    jMenuItem1.addActionListener(new Cadre1_jMenuItem1_actionAdapter(this));
    jMenuItem3.setText("Ouvrir");
    jMenuItem3.addActionListener(new Cadre1_jMenuItem3_actionAdapter(this));
    jMenuItem2.setText("Enregistrer");
    jMenuItem2.addActionListener(new Cadre1_jMenuItem2_actionAdapter(this));
    // ajoutes ces composants à la barre d'icônes
    toolBar.add(jButton1);
    toolBar.add(jButton2);
    toolBar.add(jButton3);
    menuFile.add(jMenuItem1);
    menuFile.add(jMenuItem3);
    menuFile.add(jMenuItem2);
    menuFile.add(menuFileExit);
    menuHelp.add(menuHelpAbout);
    menuBar1.add(menuFile);
    menuBar1.add(menuHelp);
    this.setJMenuBar(menuBar1);
    contentPane.add(toolBar, BorderLayout.NORTH);
    contentPane.add(statusBar, BorderLayout.SOUTH);
    contentPane.add(jScrollPane1, BorderLayout.CENTER);
    // mettre notre JTextPane dans notre JScrollPane
    jScrollPane1.getViewport().add(jTextPane1, null);
  }

  //Opération Fichier | Quitter effectuée
  public void fileExit_actionPerformed(ActionEvent e) {
    System.exit(0);
  }

  //Opération Aide | A propos effectuée
  public void helpAbout_actionPerformed(ActionEvent e) {
    Cadre1_AboutBox dlg = new Cadre1_AboutBox(this);
    // pour centrer la fenêtre à l'écran
    Dimension dlgSize = dlg.getPreferredSize();
    Dimension frmSize = getSize();
    Point loc = getLocation();
    dlg.setLocation((frmSize.width - dlgSize.width) / 2 + loc.x, (frmSize.height -
dlgSize.height) / 2 + loc.y);
    dlg.setModal(true);
    dlg.show();
  }

  //Remplacé, ainsi nous pouvons sortir quand la fenêtre est fermée
  protected void processWindowEvent(WindowEvent e) {
    super.processWindowEvent(e);
    if (e.getID() == WindowEvent.WINDOW_CLOSING) {
      // en cas de clic sur la croix (X) en haut à droite de la fenêtre (sous Win95)
      fileExit_actionPerformed(null);
    }
  }

  void jMenuItem1_actionPerformed(ActionEvent e) {
       // Cela ne correspond pas vraiment à un Fichier>Nouveau classique mais
       // plutôt à un effacement complet de l'écran...
       jTextPane1.setText("");
  }

  void jMenuItem3_actionPerformed(ActionEvent e) {
       // appel de la méthode ouvrir() quand ce gestionnaire d'évènement est appelé
       ouvrir();
  }
```

```
  void jMenuItem2_actionPerformed(ActionEvent e) {
        enregistrer();
  }

  void jButton1_actionPerformed(ActionEvent e) {
        ouvrir();
  }

  void jButton2_actionPerformed(ActionEvent e) {
        enregistrer();
  }

  void jButton3_actionPerformed(ActionEvent e) {
    Cadre1_AboutBox dlg = new Cadre1_AboutBox(this);
    Dimension dlgSize = dlg.getPreferredSize();
    Dimension frmSize = getSize();
    Point loc = getLocation();
    dlg.setLocation((frmSize.width - dlgSize.width) / 2 + loc.x, (frmSize.height -
dlgSize.height) / 2 + loc.y);
    dlg.setModal(true);
    dlg.show();
  }
}


// à partir de là, vous n'aurez pas à étudier le code dans les détails
class Cadre1_menuFileExit_ActionAdapter implements ActionListener {
  Cadre1 adaptee;

  Cadre1_menuFileExit_ActionAdapter(Cadre1 adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.fileExit_actionPerformed(e);
  }
}

class Cadre1_menuHelpAbout_ActionAdapter implements ActionListener {
  Cadre1 adaptee;

  Cadre1_menuHelpAbout_ActionAdapter(Cadre1 adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.helpAbout_actionPerformed(e);
  }
}

class Cadre1_jMenuItem1_actionAdapter implements java.awt.event.ActionListener {
  Cadre1 adaptee;

  Cadre1_jMenuItem1_actionAdapter(Cadre1 adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItem1_actionPerformed(e);
  }
}

class Cadre1_jMenuItem3_actionAdapter implements java.awt.event.ActionListener {
  Cadre1 adaptee;

  Cadre1_jMenuItem3_actionAdapter(Cadre1 adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItem3_actionPerformed(e);
  }
```

```
}

class Cadre1_jMenuItem2_actionAdapter implements java.awt.event.ActionListener {
  Cadre1 adaptee;

  Cadre1_jMenuItem2_actionAdapter(Cadre1 adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItem2_actionPerformed(e);
  }
}

class Cadre1_jButton1_actionAdapter implements java.awt.event.ActionListener {
  Cadre1 adaptee;

  Cadre1_jButton1_actionAdapter(Cadre1 adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButton1_actionPerformed(e);
  }
}

class Cadre1_jButton2_actionAdapter implements java.awt.event.ActionListener {
  Cadre1 adaptee;

  Cadre1_jButton2_actionAdapter(Cadre1 adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButton2_actionPerformed(e);
  }
}

class Cadre1_jButton3_actionAdapter implements java.awt.event.ActionListener {
  Cadre1 adaptee;

  Cadre1_jButton3_actionAdapter(Cadre1 adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButton3_actionPerformed(e);
  }
}
```

## Ouvrir.java

La classe Ouvrir possède une méthode, open(File fichier), qui permet d'ouvrir le fichier spécifié en argument. Voici l'implémentation de la classe Ouvrir :

```
/**
 * Titre : Ebauche de traitement de texte<p>
 * Description : Code source du programme de traitement de texte présenté dans le
cours JGFL {Juin 2000}<p>
 * Copyright : Copyright (c) Guillaume Florimond<p>
 * Société : JGFLsoft<p>
 * @author Guillaume Florimond
 * @version 1.0
 */
package texte;
import java.io.*;
```

```
public class Ouvrir {
  public Ouvrir() { // constructeur vide
  }
  // On doit transmettre un argument à la méthode: le nom du fichier à ouvrir
  public static void open(File fichier) {
    // intercepter une éventuelle exception
    try {
      // ouvrir un flux d'entrée sur le fichier donné pour argument à la méthode
      FileInputStream fis = new FileInputStream(fichier);
      int n;
      // tant qu'il reste des caractères dans le fichier...
      while ((n = fis.available()) > 0) {
      // créer une matrice de bytes
      byte[] b = new byte[n];
      // lire le caractère courant dans le flux
      int result = fis.read(b);
      // -1 indique que le dernier caractère est atteint
      if (result == -1) break;
      String s = new String(b);
      // insérer le texte du String s dans le zone de texte de Cadre1
      Cadre1.jTextPane1.setText(s);
      // pour convertir le nom du fichier en String et l'afficher dans le barre
d'état
      Cadre1.statusBar.setText("Fichier ouvert:  \" " + fichier.toString() + " \"");
      }
    } catch (IOException er) {}
  }
}
```

## Enregistrer.java

La classe Enregistrer possède une méthode, save(File fichier), qui permet d'enregistrer le fichier spécifié en argument. Voici l'implémentation de la classe Enregistrer :

```
/**
 * Titre : Ebauche de traitement de texte<p>
 * Description : Code source du programme de traitement de texte présenté dans le
cours JGFL {Juin 2000}<p>
 * Copyright : Copyright (c) Guillaume Florimond<p>
 * Société : JGFLsoft<p>
 * @author Guillaume Florimond
 * @version 1.0
 */
package texte;

import java.io.*;

public class Enregistrer {

  public Enregistrer() {
  }

  // la méthode a besoin d'un argument: le fichier de sortie
  public static void save(File fichier) {
    // d'éventuelles exceptions...
     try {
     // un String qui contient le texte du JTextPane de Cadre1 est créé
     String h = new String(Cadre1.jTextPane1.getText());
     // le fichier est créé
     FileWriter lu = new FileWriter(fichier);
     // mis en cache pour plus de rapidité et de fiabilité dans le transfert
     BufferedWriter out = new BufferedWriter(lu);
     // et on écrit le String dans le flux de sortie...
     out.write(h);
     // on ferme le flux, c'est terminé !
     out.close();
     // pour convertir le nom du fichier en String et l'afficher dans le barre
d'état
      Cadre1.statusBar.setText("Fichier enregistré:  \" " + fichier.toString() + "
\"");
```

```
    }catch (IOException er) {}
  }

}
```

## Cadre1_AboutBox.java

Ce code est généré intégralement et automatiquement par JBuilder, rien n'a été changé :

```
/**
 * Titre : Ebauche de traitement de texte<p>
 * Description : Code source du programme de traitement de texte présenté dans le
cours JGFL {Juin 2000}<p>
 * Copyright : Copyright (c) Guillaume Florimond<p>
 * Société : JGFLsoft<p>
 * @author Guillaume Florimond
 * @version 1.0
 */
package texte;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

public class Cadre1_AboutBox extends JDialog implements ActionListener {

  JPanel panel1 = new JPanel();
  JPanel panel2 = new JPanel();
  JPanel insetsPanel1 = new JPanel();
  JPanel insetsPanel2 = new JPanel();
  JPanel insetsPanel3 = new JPanel();
  JButton button1 = new JButton();
  JLabel imageControl1 = new JLabel();
  ImageIcon imageIcon;
  JLabel label1 = new JLabel();
  JLabel label2 = new JLabel();
  JLabel label3 = new JLabel();
  JLabel label4 = new JLabel();
  BorderLayout borderLayout1 = new BorderLayout();
  BorderLayout borderLayout2 = new BorderLayout();
  FlowLayout flowLayout1 = new FlowLayout();
  FlowLayout flowLayout2 = new FlowLayout();
  GridLayout gridLayout1 = new GridLayout();
  String product = "Ebauche de traitement de texte";
  String version = "1.0";
  String copyright = "Copyright (c) Guillaume Florimond";
  String comments = "Code source du programme de traitement de texte présenté dans
le cours JGFL {Juin 2000}";
  public Cadre1_AboutBox(Frame parent) {
    super(parent);
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
    //imageControl1.setIcon(imageIcon);
    pack();
  }

  private void jbInit() throws Exception  {
    //imageIcon = new ImageIcon(getClass().getResource("[Votre image]"));
    this.setTitle("A propos");
    setResizable(false);
    panel1.setLayout(borderLayout1);
    panel2.setLayout(borderLayout2);
    insetsPanel1.setLayout(flowLayout1);
```

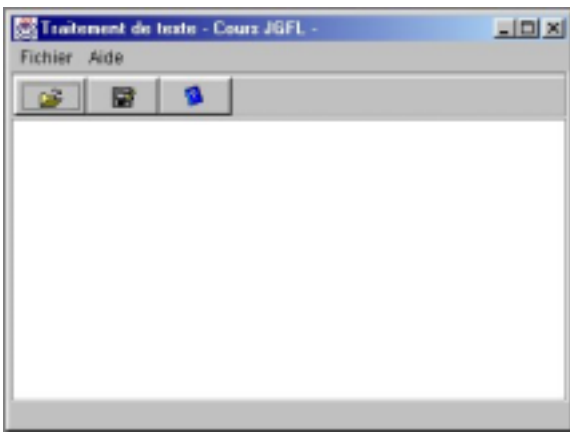
```
    insetsPanel2.setLayout(flowLayout1);
    insetsPanel2.setBorder(new EmptyBorder(10, 10, 10, 10));
    gridLayout1.setRows(4);
    gridLayout1.setColumns(1);
    label1.setText(product);
    label2.setText(version);
    label3.setText(copyright);
    label4.setText(comments);
    insetsPanel3.setLayout(gridLayout1);
    insetsPanel3.setBorder(new EmptyBorder(10, 60, 10, 10));
    button1.setText("Ok");
    button1.addActionListener(this);
    insetsPanel2.add(imageControl1, null);
    panel2.add(insetsPanel2, BorderLayout.WEST);
    this.getContentPane().add(panel1, null);
    insetsPanel3.add(label1, null);
    insetsPanel3.add(label2, null);
    insetsPanel3.add(label3, null);
    insetsPanel3.add(label4, null);
    panel2.add(insetsPanel3, BorderLayout.CENTER);
    insetsPanel1.add(button1, null);
    panel1.add(insetsPanel1, BorderLayout.SOUTH);
    panel1.add(panel2, BorderLayout.NORTH);
  }

  protected void processWindowEvent(WindowEvent e) {
    if (e.getID() == WindowEvent.WINDOW_CLOSING) {
      cancel();
    }
    super.processWindowEvent(e);
  }

  void cancel() {
    dispose();
  }

  public void actionPerformed(ActionEvent e) {
    if (e.getSource() == button1) {
      cancel();
    }
  }
}
```

# Le programme

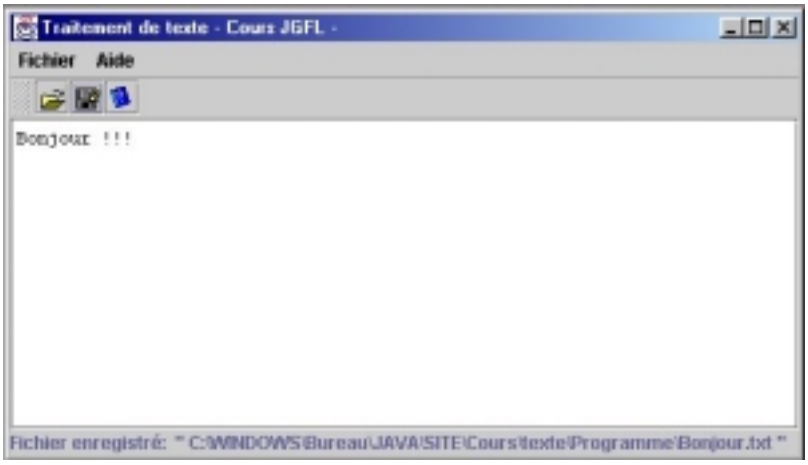
Le code source est disponible à l'adresse suivante :
http://perso.wanadoo.fr/guillaume/Cours/texte/Programme/
Il comprend :

- Un JAR exécutable
- Tous les fichiers .class
- Tous les fichiers .java
- Les fichiers Projet JBuilder
- Les images de la barre d'icônes (dans l'archive JAR)

Je vous conseille vivement de vous servir de JBuilder, en ouvrant le fichier Programme.JPR




Voici des petites captures d'écran, interface Windows, interface Metal :


Traitement de texte - Cours JGFL -
Fichier
Aide
Bonjour !!!
Fichier enregistré: " C:\WINDOWS\Bureau\JAVA\SITE\Cours\texte\Programme\Bonjour.txt "