Informatiser le service Ressources Humaines chez la Faculté des Lettres et Science Humaines

Rchid Anas

Introduction

Ce projet est le résultat d'un stage que j'avais passé chez la Faculté des Lettres et Science Humaines, El Jadida sous le thème Informatiser le service Ressources Humaines. Sous l'encadrement de Mr. A. Madani, et la supervision du chef de service; Mr. Driss Dibaji.

Cahier des Charges

Un service qui est responsable de la gestion des employés et fonctionnaires, leurs diplômes et grades. . .

Cahier des Charges

Un service qui est responsable de la gestion des employés et fonctionnaires, leurs diplômes et grades. . .

- *Implémenter un système de gestion des employés/fonctionnaires*

Cahier des Charges

Un service qui est responsable de la gestion des employés et fonctionnaires, leurs diplômes et grades. . .

- *Implémenter un système de gestion des employés/fonctionnaires*
- *Gérer les diplômes et les grades*

Cahier des Charges

Un service qui est responsable de la gestion des employés et fonctionnaires, leurs diplômes et grades. . .

- *Implémenter un système de gestion des employés/fonctionnaires*
- *Gérer les diplômes et les grades*
- *Suivi des grades*

Cahier des Charges

Un service qui est responsable de la gestion des employés et fonctionnaires, leurs diplômes et grades. . .

- *Implémenter un système de gestion des employés/fonctionnaires*
- *Gérer les diplômes et les grades*
- *Suivi des grades*
- *Suivi d'absence*

Cahier des Charges

Un service qui est responsable de la gestion des employés et fonctionnaires, leurs diplômes et grades. . .

- *Implémenter un système de gestion des employés/fonctionnaires*
- *Gérer les diplômes et les grades*
- *Suivi des grades*
- *Suivi d'absence*
- *Suivi de rémunération du travail les fériés*

Cahier des Charges

Un service qui est responsable de la gestion des employés et fonctionnaires, leurs diplômes et grades. . .

- *Implémenter un système de gestion des employés/fonctionnaires*
- *Gérer les diplômes et les grades*
- *Suivi des grades*
- *Suivi d'absence*
- *Suivi de rémunération du travail les fériés*
- *Générer des attestations de travail*

Cahier des Charges

Un service qui est responsable de la gestion des employés et fonctionnaires, leurs diplômes et grades. . .

- *Implémenter un système de gestion des employés/fonctionnaires*
- *Gérer les diplômes et les grades*
- *Suivi des grades*
- *Suivi d'absence*
- *Suivi de rémunération du travail les fériés*
- *Générer des attestations de travail*
- *Générer des autorisations de congé*

Cahier des Charges

Un service qui est responsable de la gestion des employés et fonctionnaires, leurs diplômes et grades. . .

- *Implémenter un système de gestion des employés/fonctionnaires*
- *Gérer les diplômes et les grades*
- *Suivi des grades*
- *Suivi d'absence*
- *Suivi de rémunération du travail les fériés*
- *Générer des attestations de travail*
- *Générer des autorisations de congé*
- *Générer des fiches de notation annuelle*

Conception

Stockage des données en XML

Conception

Stockage des données en XML

Les données sont stockées dans un fichier `XML`;

Conception

Stockage des données en XML

Les données sont stockées dans un fichier `XML`; puisqu'il est lisible à la fois par la machine et l'humain.

Conception

Stockage des données en XML

Les données sont stockées dans un fichier `XML`; puisqu'il est lisible à la fois par la machine et l'humain.

- *Le root-tag est* `<Employee>` *et qui contient* `0` ou plusieurs *tags de type* `<employee>` *qui représente des employées.*

Conception

Stockage des données en XML

Les données sont stockées dans un fichier `XML`; puisqu'il est lisible à la fois par la machine et l'humain.

- *Le root-tag est* `<Employee>` *et qui contient* 0 ou plusieurs *tags de type* `<employee>` *qui représente des employées.*
- *Chaque tag* `<employee>` *contient* un seul *tag* `<personal>` *et* un seul *tag* `<administrative>` *qui peut contient* 0 ou plusieurs *tags* `<uplift>`.

Conception

Stockage des données en XML

Les données sont stockées dans un fichier `XML`; puisqu'il est lisible à la fois par la machine et l'humain.

- *Le root-tag est* `<Employee>` *et qui contient* 0 ou plusieurs *tags de type* `<employee>` *qui représente des employées.*
- *Chaque tag* `<employee>` *contient* un seul *tag* `<personal>` *et* un seul *tag* `<administrative>` *qui peut contient* 0 ou plusieurs *tags* `<uplift>`.
- *Le tag* `<employee>` *peut aussi avoir* 0 ou plusieurs *tags de type* `<diploma>`, `<medicalcertif>` *et* `<repayment>`.

Les Paquets et leurs Classes

Le code source de l'application est divisé en 4 paquets pricipales:

Les Paquets et leurs Classes

Le code source de l'application est divisé en 4 paquets pricipales:

`model` contient les différentes classe pour mobilisé les donnée en objet

Les Paquets et leurs Classes

Le code source de l'application est divisé en 4 paquets pricipales:

`model` contient les différentes classe pour mobilisé les donnée en objet

`app` contient les différentes énumération utilisées dans l'application.

Les Paquets et leurs Classes

Le code source de l'application est divisé en 4 paquets pricipales:

`model` contient les différentes classe pour mobilisé les donnée en objet

`app` contient les différentes énumération utilisées dans l'application. Ce paquet contient aussi `app.utils`, qui contient des utilitaires utiles pour le développement, notamment la gestion du fichier XML

Les Paquets et leurs Classes

Le code source de l'application est divisé en 4 paquets pricipales:

`model` contient les différentes classe pour mobilisé les donnée en objet

`app` contient les différentes énumération utilisées dans l'application. Ce paquet contient aussi `app.utils`, qui contient des utilitaires utiles pour le développement, notamment la gestion du fichier XML

`wins` contient des interfaces graphiques, y compris celles qui sont responsables des opérations CRUD normales qui existent dans `wins.crud`

Les Paquets et leurs Classes

Le code source de l'application est divisé en 4 paquets pricipales:

`model` contient les différentes classe pour mobilisé les donnée en objet

`app` contient les différentes énumération utilisées dans l'application. Ce paquet contient aussi `app.utils`, qui contient des utilitaires utiles pour le développement, notamment la gestion du fichier XML

`wins` contient des interfaces graphiques, y compris celles qui sont responsables des opérations CRUD normales qui existent dans `wins.crud`

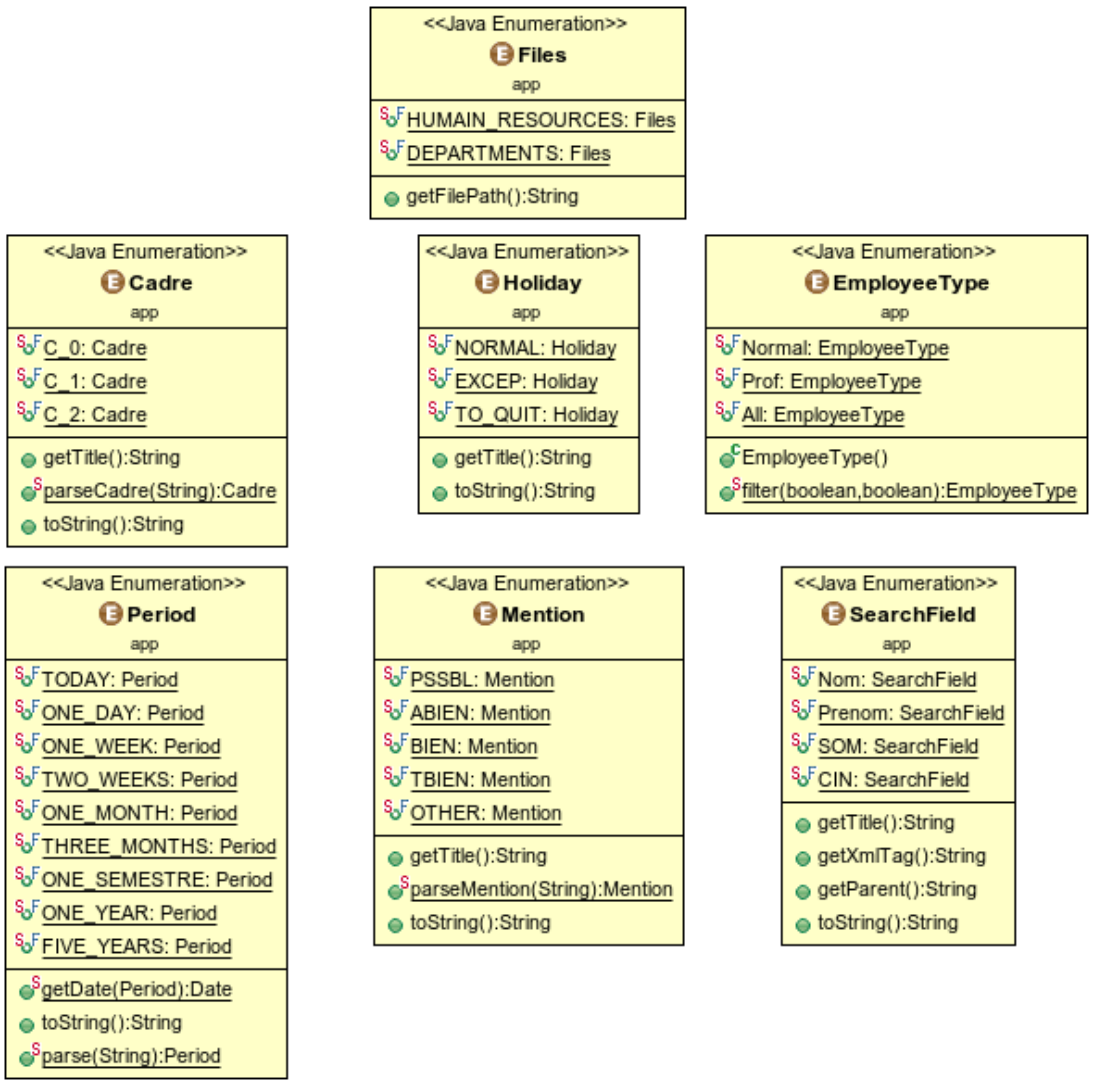
`views` contient des pages générées pour l'impression.

Paquet app

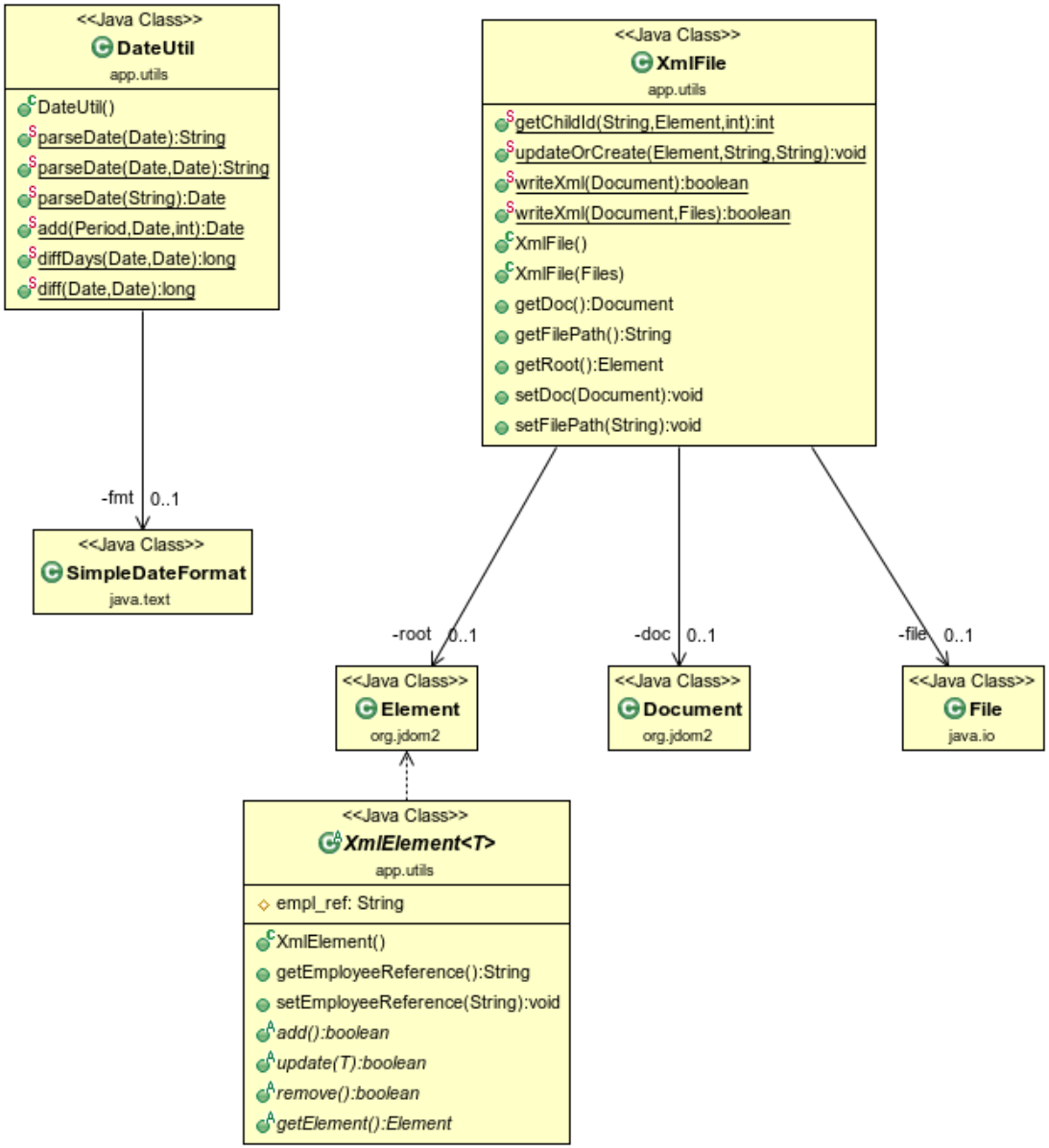
Ce paquet contient que les énumérations

Paquet app

Ce paquet contient que les énumérations



Paquet app.utils



La classe `XmlFile`

`XmlFile`, la couche DAO de l'application. Elle est responsable a tout interaction avec le fichier `XML`. Elle est basée sur un paquet `Java` appelée `JDOM`.

Les composant de JDOM

`Element`	la représentation des tags XML en objet
`Document`	la représentation du document XML en objet
`SAXBuilder`	pour initialiser les instances `Document`
`XMLOutputter`	pour l'écriture de `Document` et fichier réal

Les méthodes de la classe XmlFile

En peut sauvegarder les changements dans le fichier XML avec la méthode `static write Xml()`.

Définition de la méthode

```
1  public static boolean writeXml(Document doc, Files f) {
2      try {
3          XMLOutputter out = new XMLOutputter( );
4          out.setFormat(Format.getPrettyFormat( ));
5          out.output(
6              doc, new FileWriter(f.getFilePath( )));
7          return true;
8      } catch (IOException e) {
9          return false;
10     }
11 }
```

Les méthodes de la classe XmlFile

Avec l'aide de `static updateOrCreate()` on peut faire une mise à jour a une valeur d'un tag dans le fichier `XML`.

Définition de la méthode

```
1  public static void updateOrCreate(Element el,
2                                    String node,
3                                    String value) {
4      if (el.getChild(node) == null) {
5          el.addContent(
6              new Element(node).addContent(value));
7          writeXml(el.getDocument( ));
8      } else {
9          el.getChild(node).setText(value);
10     }
11 }
```

La classe XmlElement

```
1  public abstract class XmlElement<T> {
2      public abstract boolean add();
3      public abstract boolean update(T updated);
4      public abstract boolean remove();
5      public abstract Element getElement();
6
7      /* référence du employé */
8      protected String empl_ref;
9      public String getEmployeeReference( ) {
10         return empl_ref;
11     }
12
13     public void setEmployeeReference(String ref) {
14         this.empl_ref = ref;
15     }
16 }
```

La classe DateUtil

La classe `DateUtil` est utilisé pour la manipulation des dates, et la conversion des dates de/vers `String` avec l'aide du classe system `SimpleDateFormat`. Pour les dates, j'ai choisi un format standard, `YYYY-MM-DD`, pour touts les dates dans le projet.

Extrait de la classe

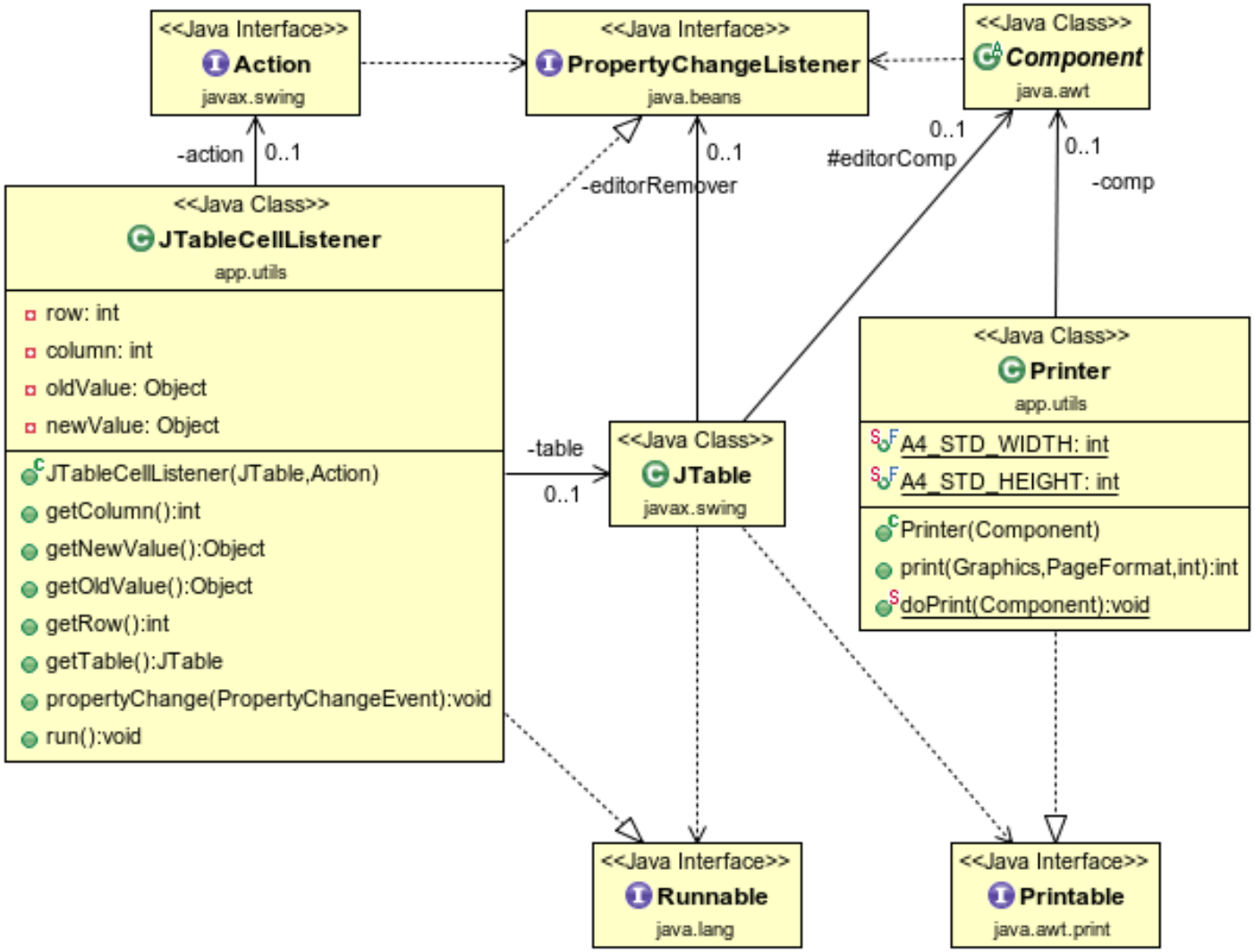
```
1 public class DateUtil {
2     private SimpleDateFormat fmt;
3
4     public DateUtil() {
5         fmt = new SimpleDateFormat("yyyy-MM-dd");
6     }
7 }
```

Suite du Paquet `app.utils`

Ce paquet contient des classes important pour l'application.

Suite du Paquet `app.utils`

Ce paquet contient des classes important pour l'application.



La classe `Printer`

La classe `Printer` est responsable de l'impression d'un `Component` (la classe des composants graphiques) avec l'aide de la méthode `static doPrint()` qui prend un `Component` comme paramètre.

La classe Printer

La classe `Printer` est responsable de l'impression d'un `Component` (la classe des composants graphiques) avec l'aide de la méthode `static doPrint()` qui prend un `Component` comme paramètre.

Example de l'appel

```
import app.utils.Printer;

Printer.doPrint(/* le composant graphique */);
```

La classe Printer

La classe `Printer` est responsable de l'impression d'un `Component` (la classe des composants graphiques) avec l'aide de la méthode `static doPrint()` qui prend un `Component` comme paramètre.

Example de l'appel

```
import app.utils.Printer;

Printer.doPrint(/* le composant graphique */);
```

Implémentation de l'interface `java.awt.Printable`

Aussi, `Printer` a une implémentation de la méthode abstrait `print()` de l'interface `Printable`

Extrais de la classe Printer

L'implémentation du print()

```
1  public class Printer implements Printable {
2      private final Component comp;
3      public Printer(Component comp) {
4          this.comp = comp;
5      }
6      @Override
7      public int print(Graphics g, PageFormat format,
8                       int page_index) {
9          if (page_index > 0)
10             return Printable.NO_SUCH_PAGE;
11         /* ... */
12         return Printable.PAGE_EXISTS;
13     }
14 }
```

La classe `JTableCellListener`

Cette classe est responsable à réagir avec une modification qui passe au niveau des cellules d'un `JTable`. Cette classe est à l'écoute des modifications apportées aux données de la table via `TableCellEditor` du paquet `javax.swing.table` avec l'aide du interface `PropertyChangeListener` du paquet `java.beans`.

La classe `JTableCellListener`

Cette classe est responsable à réagir avec une modification qui passe au niveau des cellules d'un `JTable`. Cette classe est à l'écoute des modifications apportées aux données de la table via `TableCellEditor` du paquet `javax.swing.table` avec l'aide du interface `PropertyChangeListener` du paquet `java.beans`.

Principe de listener

Lorsque l'édition est démarrée, la valeur de la cellule est enregistrée. Lorsque l'édition est arrêtée, la nouvelle valeur est enregistrée en tant que `Object`. Lorsque l'ancienne et la nouvelle valeur sont différentes, l'action fournie est invoquée.

Extrais de la classe JTableCellListener

Les attribut de la classe JTableCellListener

```
1 public class JTableCellListener implements
2                                   PropertyChangeListener,
3                                   Runnable {
4     private JTable table;
5     private Action action;
6     private int row, column;
7     private Object oldValue, newValue;
8 }
```

Les constricteurs de la classe JTableCellListener

Le constructeur par défaut utilisé pour créer un Listener, prend la table correspondante et une action.

Constructeur public

```
1 public JTableCellListener(JTable table,
2                            Action action) {
3     this.table = table;
4     this.action = action;
5     // ajouter cette classe au Listener du table
6     this.table.addPropertyChangeListener(this);
7 }
```

Les constricteurs de la classe JTableCellListener

Ce constricteur est utilise dans la methode `processEditingStopped()` pour intialise une instance de la classe déclenché pour réagir à l'événement.

Constructeur privé

```
1  private JTableCellListener(JTable table,
2                             int row, int column,
3                             Object oldValue,
4                             Object newValue) {
5      this.table = table;
6      this.row = row;
7      this.column = column;
8      this.oldValue = oldValue;
9      this.newValue = newValue;
10 }
```

Implémentation de l'interface `PropertyChangeListener`

La classe `JTableCellListener` doit contient une implementation a la methode `propertyChange()`

Implémentation de l'interface PropertyChangeListener

La classe `JTableCellListener` doit contient une implementation a la methode `propertyChange()`

Extrais de l'implementation

```
1  @Override
2  public void propertyChange(PropertyChangeEvent e) {
3      String prop_name = "tableCellEditor";
4      boolean isediting = this.table.isEditing( );
5
6      if (prop_name.equals(e.getPropertyName( ))) {
7          if (isediting) processEditingStarted( );
8          else processEditingStopped( );
9      }
10 }
```

Suite de l'implementation de PropertyChangeListener

Au démarrage de la processus d'édition

```
1 private void processEditingStarted( ) {
2     SwingUtilities.invokeLater(this);
3 }
```

Suite de l'implementation de PropertyChangeListener

Au démarrage de la processus d'édition

```
1 private void processEditingStarted( ) {
2     SwingUtilities.invokeLater(this);
3 }
```

La relation avec l'interface Runnable

Aussi, on est besoin de réagir avec des événements par exécuté des actions, donc on doit implémenter la méthode `run()` qui est été appelée avec le protocole `SwingUtilities.invokeLater()` dans `processEditingStarted()`.

Implémentation de l'interface Runnable

Le rôle de cet appel est de récupérer la valeur actuelle de la cellule.

Mettre à jour les valeurs de la classe JTableCellListener

```
1  @Override
2  public void run( ) {
3      row = table.convertRowIndexToModel(
4          table.getEditingRow( ));
5      column = table.convertColumnIndexToModel(
6          table.getEditingColumn( ));
7
8      oldValue = table.getModel( ).getValueAt(
9          row, column);
10     newValue = null; /* réinitialisation */
11 }
```

À la fin de la processus d'édition

```
1  private void processEditingStopped( ) {
2      newValue = table.getModel( ).getValueAt(row, column);
3      if (!newValue.equals(oldValue)) {
4          action.actionPerformed(
5              new ActionEvent(
6                  new JTableCellListener(
7                      table, row, column,
8                      oldValue, NewValue
9                  ),
10                 ActionEvent.ACTION_PERFORMED,
11                 "JTableCellEditor"
12             )
13         );
14     }
15 }
```

Paquet `model`

Héritage de la classe `XmlElement`

La méthode `update()` prend un variable de type `T`,

Héritage de la classe `XmlElement`

La méthode `update()` prend un variable de type `T`,ce type est décrit avec un héritage de la classe `XmlElement`

Héritage de la classe `XmlElement`

La méthode `update()` prend un variable de type `T`,ce type est décrit avec un héritage de la classe `XmlElement`

```
1  public class Diploma extends XmlElement<Diploma> {
2      /* les attributs du classe */
3      @Override
4      public boolean update(Diploma updated) {
5          try {
6              /* process la mise à jour */
7              return true;
8          } catch (Exception e) {
9              System.err.println(e.getMessage());
10             return false;
11         }
12     }
13 }
```

L'Interface Graphique

La relation entre les différents Paquets graphiques