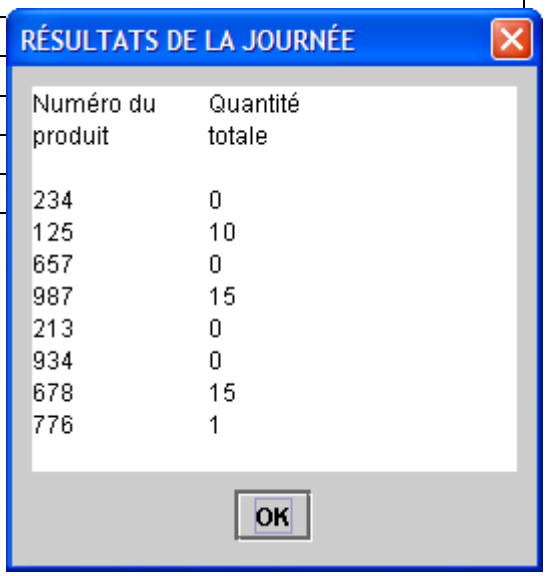
Exercice de révision sur les tableaux : Complétez le programme suivant qui permet de calculer et d'afficher le coût de l'achat d'un client (il y a plusieurs clients à traiter, mais on suppose qu'un client n'effectue qu'un seul achat) et qui va afficher (après avoir traité tous les clients de la journée) la quantité totale vendue pour chaque produit.

```
import javax.swing.*;
import java.text.*;
public class ExerciceTableaux{

   public static void main(String args[]) {
       final int NB_PROD = 8;
       int    tabNoProd[]   = {  234, 125, 657,  987, 213,  934,  678,  776};
       double tabPrixProd[] = {45.99,9.50,5.75,12.35,9.75,87.45,56.99,76.56};


       traiterLesClients(                                                  );
       afficherResultats(                                                  );
       System.exit(0);
   } // fin de la méthode main

   static void traiterLesClients(int tabNoProd[], double tabPrix[],
           int tabQteTotale[], int nbProd) {
       DecimalFormat cash = new DecimalFormat("0.00 $");
       int numero,
           qte,
           posiProd;    // position du numéro dans le tableau tabNoProd
       double cout;
       char reponse;

       do {
           numero = Integer.parseInt(JOptionPane.showInputDialog(
                     "Entrez le numéro du produit à acheter :"));
           qte = Integer.parseInt(JOptionPane.showInputDialog(
                     "Entrez la quantité désirée :"));

           if (posiProd != -1)
           {
               cout =
               JOptionPane.showMessageDialog(null,
                   "Le coût de cet achat est de " + cash.format(cout));

           }
           else
               JOptionPane.showMessageDialog(null, "No de produit erroné");
```

```
            reponse = JOptionPane.showInputDialog(
                    "Avez-vous un autre client à traiter O/N ?").charAt(0);
            reponse = Character.toUpperCase(reponse);
        } while (reponse == 'O');
    } // fin de la méthode traiterLesClients

    static int rechercher(int tab[], int nbEl, int valeurCherchee)
    {
        int posi = -1;
        boolean trouve = false;

        for (int i = 0; i < nbEl && !trouve; i++)
            if (tab[i] == valeurCherchee)
            {
                posi = i;
                trouve = true;
            }
        return posi;
    } // fin de la méthode rechercher

    static void afficherResultats(int tabNoProd[], int                    ,
            int nbProd) {

    } // fin de la méthode afficherResultats
} // fin de la classe
```



Utilisation d'un showOptionDialog

Une fenêtre **showOptionDialog** est une forme de fenêtre avec des boutons personnalisés. On l'utilise souvent pour faire des menus. Le contenu des boutons provient d'un tableau de chaînes de caractères (String). La méthode **showOptionDialog** retourne l'*indice* du bouton sur lequel l'utilisateur a cliqué, le premier bouton portant l'indice 0.

Format général :

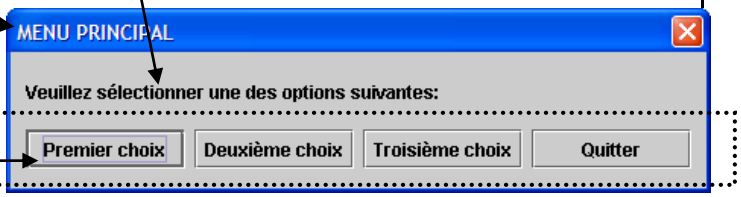
```
JOptionPane.showOptionDialog(null,
   message,              // texte à afficher
   titre,                // titre de la fenêtre
   0,
   type d'icône,         // JOptionPane.PLAIN_MESSAGE ou autre...
   null,
   nomDuTableau,         // tableau de String pour texte des boutons
   nomDuTableau[n])      // où n est l'indice du bouton présélectionné
                         // (celui qui sera choisi si on fait Entrée)
```

Exemple 1 :

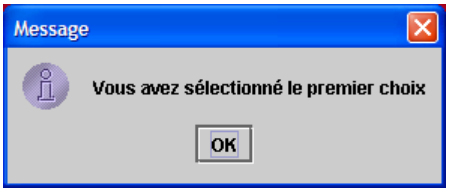

```
    // Exemple de menus pour des jeux
    public static void main(String[] args) {

        String menu[] = {"Jeu de dés", "Devinette d'un nombre",
                         "Jeu du pendu", "Terminer"};

        int choix;

        do
        {
            choix = JOptionPane.showOptionDialog(null,
                   "Faites votre choix :", "MENU DES JEUX",
                   0, JOptionPane.QUESTION_MESSAGE, null,
                   menu, menu[2]);
            switch (choix)
            {
                case 0: jouerAuxDes();
                        break;
                case 1: devinerNombre();
                        break;
                case 2: jouerAuPendu();
                        break;
            }
        } while (choix != 3);

        System.exit(0);
    } // fin de la méthode main
```

Exemple 2 :

```
JOptionPane.showOptionDialog(null,
      "Veuillez sélectionner une des options suivantes:",
      "MENU PRINCIPAL",
      0,
      JOptionPane.PLAIN_MESSAGE,
      null,
      options,
      options[0])
```




```
/*
 * TestShowOptionDialog.java
 * Ce programme démontre l'utilisation du showOptionDialog
 * pour faire un menu
 */
import javax.swing.*;
public class TestShowOptionDialog
{
    public static void main(String[] args)
    {
        final int PREMIER = 0,
                  DEUXIEME = 1,
                  TROISIEME = 2,
                  QUITTER = 3;

        int choixMenu;

        String options[] = {"Premier choix", "Deuxième choix",
                            "Troisième choix", "Quitter"};

        do
        {
            choixMenu = JOptionPane.showOptionDialog(null,
                   "Veuillez sélectionner une des options suivantes:",
                   "MENU PRINCIPAL", 0, JOptionPane.PLAIN_MESSAGE,
                   null, options, options[0]);

            switch(choixMenu)
            {
                case PREMIER:
                    JOptionPane.showMessageDialog(null,
                          "Vous avez sélectionné le premier choix");
                    break;
                case DEUXIEME:
                    JOptionPane.showMessageDialog(null,
                          "Vous avez sélectionné le deuxième choix");
                    break;
                case TROISIEME:
                    JOptionPane.showMessageDialog(null,
                          "Vous avez sélectionné le troisième choix");
            }
        } while (choixMenu != QUITTER);

        System.exit(0);
    }
}
```



Les fichiers texte

Comme son nom l'indique, un fichier texte contient des lignes de texte. Chaque ligne est un objet String, exactement comme ce qu'on obtient à partir d'un **`JOptionPane.showInputDialog`**.

1. Définition d'un fichier texte

Un fichier est un ensemble de données stockées, en général, sur un support externe (disque dur, disquette, clé USB, etc.). Un fichier structuré contient une suite d'enregistrements homogènes, qui regroupent, le plus souvent, plusieurs composantes (champs).

Un fichier texte est constitué de lignes de texte. Même les valeurs numériques ont été enregistrées sous forme de texte. Chaque ligne du fichier constitue un enregistrement qui contient les informations relatives à un dossier (ou un produit, un employé, etc.). Une fois la ligne lue, il faut extraire les informations de la ligne pour retrouver les divers champs qui la composent.

2. Création d'un fichier texte

On crée généralement un fichier texte dans **Bloc-notes** mais on peut aussi le créer directement dans Éclipse. À partir d'Eclipse, une fois le projet sélectionné, on fait ***Fichier / Nouveau / Fichier***, puis on entre le nom du fichier, par exemple **données.txt**. Il suffit d'entrer le contenu du fichier dans la fenêtre d'édition. Il ne faut jamais terminer par **Entrée**, car il y aurait alors une ligne vide à la fin de votre fichier. On peut aussi créer un fichier texte à partir d'un programme.

3. Lecture d'un fichier texte

À partir d'un programme java, on peut lire un fichier texte. On suppose que le fichier texte à lire est situé dans le même dossier (projet) que le programme.

```
// inclure l'importation suivante
import java.io.*;

// dans main, préciser qu'il peut y avoir des erreurs d'entrées/sorties
public static void main(String args[]) throws IOException {

// il faut déclarer le fichier et son tampon de lecture
FileReader fich = new FileReader("fich.txt");
BufferedReader fichier = new BufferedReader(fich);
        ou
BufferedReader fichier = new BufferedReader(new FileReader("fich.txt"));

// pour lire une ligne du fichier
String ligneLue = fichier.readLine();

// boucle tant que la ligne lue n'est pas null car alors le fichier est terminé
// il ne faut pas oublier de relire à la fin de la boucle while
while (ligneLue != null)

// lorsque la lecture est terminée, fermer le fichier
fichier.close();
```

Exemple :

Numéros des colonnes

```
0    0    1    1    2    2    3
0....5....0....5....0....5....0..
123 Côté Denis           M12.50
222 St-Pierre Nathalie   F14.75
333 Lavoie Pierre-Luc    M  8
444 Robert Antoine       M115.20
555 Tremblay Julie       F 9.99
```

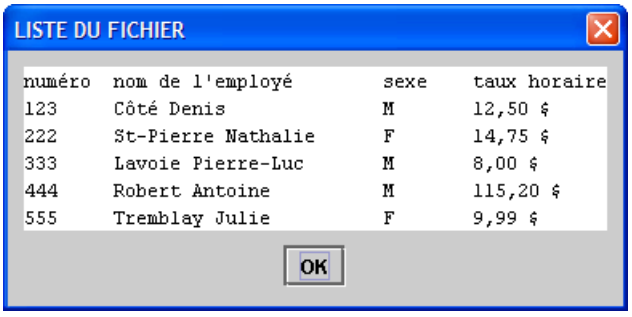
Contenu du fichier lu

```
/*
 * ExempleFichier.java
 * Lecture d'un fichier texte et
 * extraction des informations
 * auteur : Claudette Chapleau
 * date   : octobre 2006
 */
import java.io.*;       // nécessaire pour les fichiers
import java.text.*;
import javax.swing.*;
import java.awt.*;
public class ExempleFichier
{
  public static void main(String args[]) throws IOException
  {
    String  uneLigne,
            nom;
    char    sexe;
    int     noDossier;
    double  tauxHoraire;

    DecimalFormat monnaie = new DecimalFormat("0.00 $");
    JTextArea sortie = new JTextArea();

    BufferedReader fichier = new BufferedReader(
                             new FileReader("personnel.txt"));

    sortie.append("numéro\tnom de l'employé\tsexe\ttaux horaire");

    uneLigne = fichier.readLine();

    while (uneLigne != null) {
      // extraction des informations de la ligne lue
      noDossier   = Integer.parseInt(uneLigne.substring(0, 3));
      nom         = uneLigne.substring(4, 23);
      sexe        = uneLigne.charAt(24);
      tauxHoraire = Double.parseDouble(uneLigne.substring(25));

      sortie.append("\n" + noDossier + "\t" + nom + "\t" + sexe + "\t" +
                    monnaie.format(tauxHoraire));

      uneLigne = fichier.readLine();
    } // fin du while

    fichier.close();
    sortie.setFont(new Font("Courier", Font.PLAIN, 12));
    JOptionPane.showMessageDialog(null, sortie, "LISTE DU FICHIER",
                                  JOptionPane.PLAIN_MESSAGE);
    System.exit(0);

  } // fin de la méthode main

} // fin de la classe ExempleFichier
```

Déclaration du fichier et de son tampon de lecture, on suppose que le fichier est dans le même dossier que le programme

Lecture d'une ligne du fichier. Le fichier est terminé si la ligne lue est **null**

Fermeture du fichier



4. Écriture d'un fichier texte

À partir d'un programme java, on peut écrire un fichier texte. On suppose que le fichier texte à écrire est situé dans le même dossier (projet) que le programme.

```
// inclure l'importation suivante
import java.io.*;

// préciser qu'il peut y avoir des erreurs d'entrées/sorties
public static void main(String args[]) throws IOException {

// il faut déclarer le fichier et son tampon d'écriture
FileWriter fich = new FileWriter("fich.txt");
BufferedWriter fichier = new BufferedWriter(fich);
        ou
BufferedWriter fichier = new BufferedWriter(new FileWriter("fich.txt"));

// pour écrire une ligne dans le fichier après avoir
// rempli la ligne avec ce qu'on veut écrire
fichier.write(ligne);
fichier.newLine();

// lorsque l'écriture est terminée, fermer le fichier
fichier.close();
```

dans une boucle pour écrire plusieurs lignes !

On utilise normalement une méthode pour lire ou écrire un fichier. Ceci permet de dégager la logique principale. La méthode main devra quand même avoir **`throws IOException`** puisqu'elle appelle une méthode ayant **`throws IOException`** dans sa signature.

Exemple :

```
// Méthode permettant la lecture et l'écriture d'un fichier texte
static void lireEcrire() throws IOException
{
    char    code;          // A = désaltérante, D = dégustation
    String  ligneEntree,   // ligne lue
            ligneSortie,   // ligne à écrire
            nom,           // nom de la bière
            pays;          // pays d'origine
    double  alcool,        // % d'alcool
            prix;          // prix au détail

    // description des fichiers d'entrée et de sortie
    BufferedReader ficEntree = new BufferedReader(
                               new FileReader("fichIn.txt"));
    BufferedWriter ficSortie = new BufferedWriter(
                               new FileWriter("fichOut.txt"));

    // redéfinition du symbole pour le séparateur des décimales
    // car certains systèmes utilisent la virgule comme séparateur
    DecimalFormatSymbols pointDecimal = new DecimalFormatSymbols();
    pointDecimal.setDecimalSeparator('.');

    // définition des formats des nombres réels pour l'écriture
    DecimalFormat formatAlcool = new DecimalFormat("0.0", pointDecimal);
    DecimalFormat formatPrix   = new DecimalFormat("0.00", pointDecimal);

    // lecture de la première ligne du fichier d'entrée
    ligneEntree = ficEntree.readLine();

    while (ligneEntree != null)   // tant que pas la fin du fichier
    {
        code   = ligneEntree.charAt(0);
        nom    = ligneEntree.substring(1, 30);
        pays   = ligneEntree.substring(31, 42);
        alcool = Double.parseDouble(ligneEntree.substring(43, 46));
        prix   = Double.parseDouble(ligneEntree.substring(47));

        // écriture des données dans le fichier de sortie
        ligneSortie = code + nom + " " + pays + " " +
                      formatAlcool.format(alcool) + " " +
                      formatPrix.format(prix);
        ficSortie.write(ligneSortie);
        ficSortie.newLine();

        // lecture de la ligne suivante, dans le fichier d'entrée
        ligneEntree = ficEntree.readLine();
    }

    ficEntree.close();
    ficSortie.close();
}
```

Contenu du fichier
fichIn.txt

Le fichier écrit
fichOut.txt
a exactement le même contenu

→

```
AMort Subite Kriek Lambic      Belgique    4.0 4.65
DPentagel Superior English Ale Angleterre  4.5 3.15
ALa Trappe                     Hollande    8.0 2.90
AGuinness Pub Draught          Irlande     4.8 2.40
ABishop's Finger Ale           Angleterre  5.4 3.85
DBlanche de Bruxelles          Belgique    4.5 2.55
DDouglass Scotch Ale           Ecosse      8.6 3.30
ASeigneuriale Réserve          Québec      7.5 4.30
```

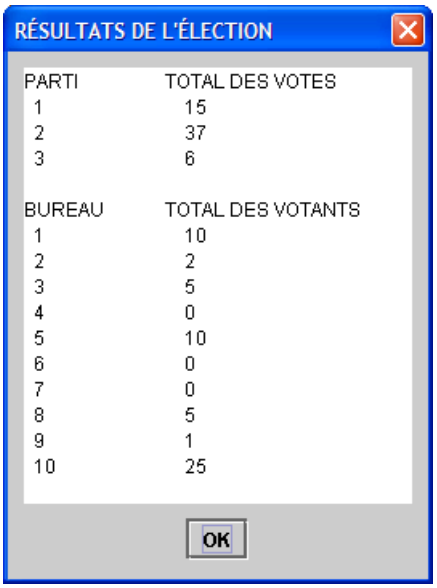
Labo1 partie 1 (2 exercices à faire)

Exercice 1 : Reprenez l'exercice **Exercice de révision sur les tableaux** en supposant que la liste des produits vendus par la compagnie est dans un fichier appelé **produits.txt**. On suppose que la compagnie offre à sa clientèle un maximum de 20 produits différents.

```
import javax.swing.*;
import java.text.*;

public class Exercice1 {

   public static void main(String args[])                                    {
       final int NB_MAX_PROD = 20;
       int    tabNoProd[]   =
       double tabPrixProd[] =
       int tabQte[]         =
       int nbProd;    // le nombre de produits contenus dans le fichier

       nbProd =
       traiterLesClients(tabNoProd, tabPrixProd, tabQte, nbProd);
       afficherResultats(tabNoProd, tabQte, nbProd);
       System.exit(0);
   } // fin de la méthode main

   static int batirTableaux(                                                  )
                                  {

   }

   // autres méthodes comme dans l'exercice 10

} // fin de la classe
```

Exercice 2 : Trois partis politiques (dont les numéros sont **1**, **2** et **3**) se présentent à une élection municipale. Les citoyens doivent aller voter à l'un des dix bureaux de scrutin (dont les numéros vont de **01** à **10**). Les résultats en provenance de ces bureaux furent conservés dans le fichier central **votes.txt**, selon leur ordre d'arrivée. Écrivez le programme qui permet de :

- lire le fichier de données dans lequel chaque enregistrement contient :
  - le numéro du bureau de scrutin (2 chiffres)
  - un espace
  - le numéro du parti (un chiffre)
  - un espace
  - le nombre de votes pour ce parti
- afficher le rapport suivant :




```
import javax.swing.*;
import java.io.*:
public class Exercice2 {

   public static void main(String args[])                    {
       final int NB_PARTIS = 3,
                 NB_BUREAUX = 10;

       int tabTotParti[]  =
       int tabTotBureau[] =

       int noParti,
           noBureau,
           nbVotes;
       String ligne;
       BuffereReader fichier = new BufferedReader(
                               new FileReader("votes.txt"));
       JTextArea sortie = new JTextArea();

       ligne =
       while (                              )
       {
          noBureau =
          noParti  =
          nbVotes  =


       }
```

```
        fichier.close();
        sortie.append("PARTI\tTOTAL DES VOTES\n");
        sortie.append("\nBUREAU\tTOTAL DES VOTANTS\n");
        JOptionPane.showMessageDialog(null, sortie,
            "RÉSULTATS DE L'ÉLECTION", JOptionPane.PLAIN_MESSAGE);
        System.exit(0);
    }
} // fin de la classe
```

Déposer ces exercices dans LEA.

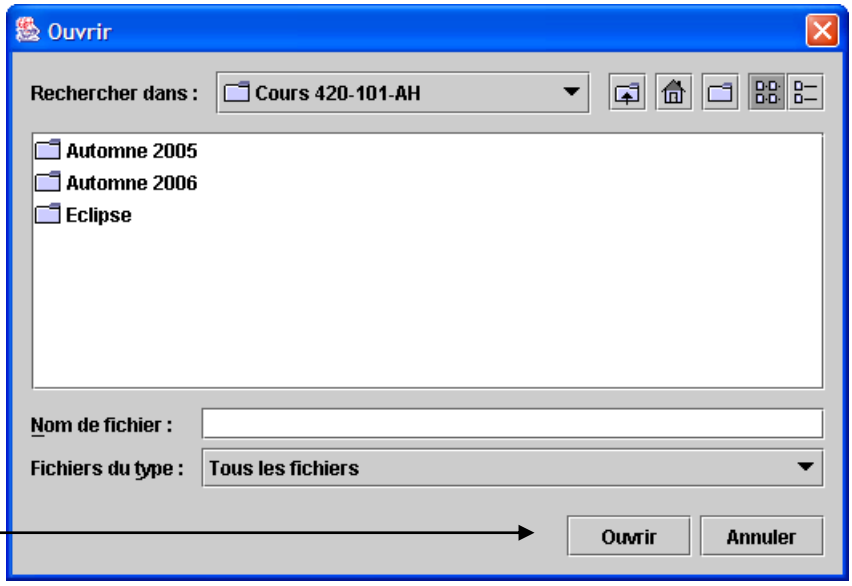
5. Utilisation du JFileChooser

Pour pouvoir lire ou écrire un fichier dont on ne connaît pas le nom, on peut utiliser la classe JFileChooser pour sélectionner le fichier à partir d'une boîte de dialogue.

La méthode **`showOpenDialog`** permet d'afficher une boîte **Ouvrir**

La méthode **`showSaveDialog`** permet d'afficher une boîte **Enregistrer**

Si on clique sur le bouton Ouvrir ou le bouton Enregistrer, la valeur retournée est la constante **`APPROVE_OPTION`** de la classe **`JFileChooser`**.



Exemple :

```
/*
 * ExempleFileChooser.java
 * Même exemple que celui de lecture du fichier
*/

import java.io.*;
import java.text.*;
import javax.swing.*;
import java.awt.*;
public class ExempleFileChooser
{
    public static void main(String args[]) throws IOException
    {
        String  uneLigne,
                nom,
                nomFichier = "";
        char    sexe;
        int     noDossier;
        double  tauxHoraire;

        DecimalFormat monnaie = new DecimalFormat("0.00 $");
        JTextArea sortie = new JTextArea();

        JFileChooser choixFichier = new JFileChooser();
        int valeurRetour = choixFichier.showOpenDialog(choixFichier);

        if (valeurRetour == JFileChooser.APPROVE_OPTION)
             nomFichier = choixFichier.getSelectedFile().getPath();
        else
             System.exit(0);   // si rien choisi, on arrête l'exécution
```

```
        // ouverture du fichier choisi en lecture
        BufferedReader fichier = new BufferedReader(
                                 new FileReader(nomFichier));

        sortie.append("numéro\tnom de l'employé\tsexe\ttaux horaire");

        uneLigne = fichier.readLine();
        while (uneLigne != null)
        {
             // extraction des informations de la ligne lue
             noDossier   = Integer.parseInt(uneLigne.substring(0, 3));
             nom         = uneLigne.substring(4, 23);
             sexe        = uneLigne.charAt(24);
             tauxHoraire = Double.parseDouble(uneLigne.substring(25));

             sortie.append("\n" + noDossier + "\t" + nom + "\t" +
                           sexe + "\t" + monnaie.format(tauxHoraire));

             uneLigne = fichier.readLine();
        } // fin du while

        fichier.close();

        sortie.setFont(new Font("Courier", Font.PLAIN, 12));
        JOptionPane.showMessageDialog(null, sortie, "LISTE DU FICHIER",
                                      JOptionPane.PLAIN_MESSAGE);

        System.exit(0);
    } // fin de la méthode main
} // fin de la classe ExempleFileChooser
```