IFT1176 - Session Automne, Final - Solution

Mohamed Lokbani

IFT1176 - FINAL - Solution

Nom: _____| Prénom(s): _____|

Signature: _____| Code perm: _____|

Section: _____|

Date:15 décembre 2001

Durée: 3 heures (de 10h30 à 13h30) Local: P-310 du Pavillon Principal (UdM) et L-604 à Longueuil.

Directives:

- Toute documentation permise.
- Calculatrice (simple) permise.

- Répondre directement sur le questionnaire.

1._____/10	GUI - Question Théorique (Q1.1 | Q1.2 | Q1.3)
2._____/15	GUI - Question Pratique (Q)
3._____/20	GUI – Programmation (Q)
4._____/12	Bdd - Requêtes SQL (Q4.1 | Q4.2 | Q4.3 | Q4.4)
5._____/18	JDBC - Sortie SQL + Programmation (Q5.1 | Q5.2 | Q5.3)
6._____/25	Servlet - Programmation

Total:_____/100

Question 1 (10 points: 3/3/4)

Si `frame` est un objet faisant référence au composant `JFrame` (une fenêtre) et `layout` est un objet faisant référence au gestionnaire de mise en forme (`layout manager`):

-Q1.1- Quelle est le nom de la méthode qu'il faudra utiliser pour récupérer le conteneur de la fenêtre `frame` pour y ajouter par exemple un bouton?

getContentPane()

-Q1.2- Écrire l'instruction (en java) qui permet de récupérer le conteneur `C` de la fenêtre `frame`:

Container c = frame.getContentPane();

-Q1.3- Écrire l'instruction (en java) qui permet d'y ajouter le bouton `JButton` au conteneur `C`.

Si bouton est un JButton, l'instruction est:

c.add(bouton);

Question 2 (15 points)

Java met à la disposition des programmeurs des méthodes standards leur permettant d'afficher aux utilisateurs un message donné. Ce message reste affiché à l'écran tant que les utilisateurs n'ont pas appuyé sur un bouton OK. La classe JOptionPane dispose de l'une de ces méthodes, showMessageDialog. Une de ses variantes à l'entête suivante:

```
public static void showMessageDialog(Component parentComponent,
                                      Object message,
                                      String title,
                                      int messageType)
```

où:

parentComponent: détermine la frame sur laquelle la fenêtre de dialogue est affichée. Si null, c'est la frame par défaut qui est prise en compte.
message: le message à afficher.
title: le titre de la fenêtre de dialogue.
messageType: le type d'icône parmi un nombre donné dont:

JOptionPane.INFORMATION_MESSAGE 

JOptionPane.QUESTION_MESSAGE 

La constante YES_NO_OPTION définit la nature des boutons figurant dans la boite de dialogue, ici YES, et NO. Elle est associée à un icône de questionnement.

Question: Expliquez le fonctionnement du programme suivant en dessinant les différents affichages successifs obtenus à l'écran pour n=4 et n=5.

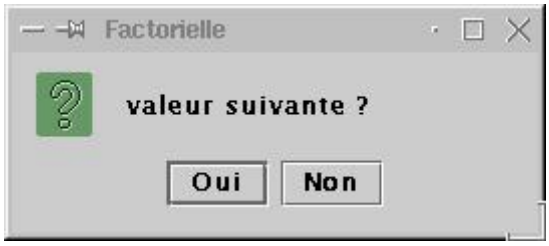
```
import javax.swing.*;

public class Factorielle
{
    public static void main(String[] args)
    {
      int n=4;
      int rep;
      do {
          JOptionPane.showMessageDialog(null,
                              n + " a pour factorielle " + fact(n),
                              "Factorielle",
                              JOptionPane.INFORMATION_MESSAGE);
          n++;
          rep = JOptionPane.showConfirmDialog(null,"valeur suivante ?",
                              "Factorielle",
                              JOptionPane.YES_NO_OPTION);
      }
      while (rep == JOptionPane.YES_OPTION);
      System.exit(0);
    }
    public static int fact(int n) {
      if (n<=0) return 1;
      else {
          return (n*fact(n-1));
      }
    }
}
```

Le programme calcule la factorielle d'un nombre entier. Pour la question demandée, nous allons nous contenter de dessiner les différentes figures obtenus dans le cas de n=5 et n=6.

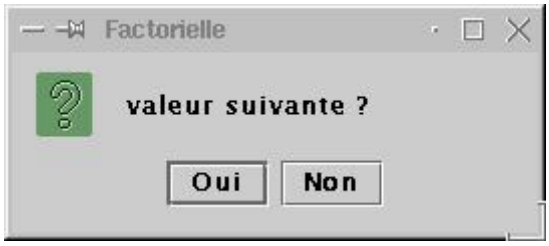
Après avoir lancé le programme avec la commande ***java Factorielle***, nous obtenons les figures suivantes:
Au départ n=4, et la factorielle de 4, calculée par la méthode ***fact***, vaut 24.

On clique sur OK, pour poursuivre le dialogue avec l'application.



Avant de faire ce calcul, le programme, nous propose de quitter le programme en appuyant sur le bouton ***Non***, ou bien de poursuivre le calcul de la prochaine factorielle (ici, du nombre 5) en appuyant sur ***Oui***. Dans notre cas, vu qu'il a été demandé dérouler l'exemple pour n=4 et n=5, nous devons appuyez sur **Oui**.

On clique la aussi sur OK.



Puisque nous avons affiché le calcul de la factorielle pour les nombres 4 et 5, nous devons donc quitter l'application en appuyant sur le bouton ***Non***.

Question 3 (20 points)

Écrire un programme comportant une zone de texte éditable d'une seule ligne ainsi que trois boutons « transformer », « effacer » et « quitter ». Le bouton « transformer » rajoute un caractère « * » au début et à la fin du texte entré par l'utilisateur. Le bouton « effacer » efface le contenu de la zone de texte. On utilisera un gestionnaire de disposition en grille.


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

public class Exo3 extends JFrame {

    private JButton quitter;
    private JButton transformer;
    private JButton effacer;
    private JTextField texte;

    // On définit des classes internes a la classe principale
    // pour gérer les actions des différents boutons

    private class GestionQuitter implements ActionListener {
      public void actionPerformed(ActionEvent e) {
          System.exit(0);
      }

    }
    private class GestionTransformer implements ActionListener {
      public void actionPerformed(ActionEvent e) {
          texte.setText("*" + texte.getText() + "*");
      }

    }
    private class GestionEffacer implements ActionListener {
      public void actionPerformed(ActionEvent e) {
          texte.setText("");
      }

    }

    // Cette méthode associe les actions aux événements
    private void actions() {
      this.quitter.addActionListener(new GestionQuitter());
      this.transformer.addActionListener(new GestionTransformer());
      this.effacer.addActionListener(new GestionEffacer());
    }
```

```
    // Cette méthode crée et met en place les objets graphiques
    private void disposer() {

      // On crée les objets
      setTitle("Exercice 3");
      // Largeur initiale de la zone de texte : 20 colonnes
      this.texte = new JTextField(20);
      this.quitter = new JButton("quitter");
      this.transformer = new JButton("transformer");
      this.effacer = new JButton("effacer");
      // On récupère la zone d'affichage de la fenêtre principale
      Container affichage = getContentPane();
      // On crée le gestionnaire de disposition
      // 2 lignes et 1 colonne
      GridLayout disposition = new GridLayout(2, 1);
      // On attribue le gestionnaire de disposition à la zone
      // d'affichage
      affichage.setLayout(disposition);
      // On crée une zone d'affichage pour les boutons
      JPanel boutons = new JPanel();
      // On attribue le gestionnaire de disposition des boutons
      // 1 ligne et 3 colonnes
      boutons.setLayout(new GridLayout(1,3));
      // On y insère les boutons
      boutons.add(transformer);
      boutons.add(effacer);
      boutons.add(quitter);
      // On ajoute les objets graphiques à la zone d'affichage
      affichage.add(texte);
      affichage.add(boutons);
      // On adapte la taille de la fenêtre afin qu'elle puisse afficher
      // tous ses composants graphiques
      this.pack();
      // On rend la fenêtre visible
      this.setVisible(true);
      // On définit le bouton de femeture de la fenêtre
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    // Constructeur, fait appel aux méthodes permettant de disposer
    // les objets graphiques et de traiter les actions sur ces objets.
    public Exo3() {
      disposer();
      actions();
    }


    public static void main(String[] args) {
      Exo3 appli = new Exo3();
    }

}
```

Question 4 (12 points: 3/3/3/3)

Soit les deux tables suivantes:

ENO	ENOM	PROF	DATEEMB	SAL	COMM	DNO
10	Joe	Ingénieur	1.10.93	4000	3000	3
20	Jack	Technicien	1.5.88	3000	2000	2
30	Jim	Vendeur	1.3.80	5000	5000	1
40	Lucy	Ingénieur	1.3.80	5000	5000	3

Table EMP

où:

ENO : numéro d'employé, (une clé)
ENOM : nom de l'employé
PROF : profession
DATEEMB : date d'embauche
SAL : salaire
COMM : commission
DNO : numéro de département, (une clé)

DNO	DNOM	DIR	VILLE
1	Commercial	30	New York
2	Production	20	Houston
3	Développement	40	Boston

Table DEPT

où

DNO : numéro de département, (une clé)
DNOM : nom du département
DIR : directeur du département
VILLE : lieu du département (ville)

Question: Exprimer en **SQL** les requêtes suivantes:

-Q4.1- Donner tous les n-uplets de DEPT.

```
SELECT * FROM DEPT;
```

-Q4.2- Donner les noms et les salaires des employés.

```
SELECT ENOM, SAL FROM EMP;
```

-Q4.3- Donner les professions des employés (après élimination des duplicats).

```
SELECT DISTINCT PROF FROM EMP;
```

-Q4.4- Donner les noms, emplois et salaires des employés par emploi croissant (ASC) et, pour chaque emploi, par salaire décroissant (DESC).

```
SELECT ENOM, PROF, SAL FROM EMP
        ORDER BY PROF ASC, SAL DESC;
```

Question 5 (18 points: 4/6/8)

Toutes les questions de cet exercice reprennent les tables de la question -4- de la page 7.

Soit la requête SQL:

```
SELECT E1.ENOM, E2.ENOM
       FROM EMP E1, EMP E2, DEPT D
       WHERE E1.DNO=D.DNO AND E2.ENO = D.DIR;
```

-Q5.1- Exprimer cette requête en français.

Donnez les noms des employés et les noms de leurs directeurs.

-Q5.2- Donner le résultat de l'exécution de cette requête.

Joe	Lucy
Jack	Jack
Jim	Jim
Lucy	Lucy

-Q5.3- Écrire la méthode qui permet de traiter le résultat obtenu en Q5.2 et l'afficher sur la sortie standard. Cette méthode doit-être codée en Java (fait partie d'un programme java-jdbc), elle accepte un seul argument du type ResultSet (le résultat de la requête). L'appel de cette méthode doit se faire dans la méthode main.

```
private static void AfficherResultat (ResultSet rs)
        throws SQLException {

        while( rs.next() ){
           System.out.println(rs.getString(1) + " " + rs.getString(2));
        }

}
```

Question 6 (25 points: 5/10/10)

Pour cet exercice, vous allez concevoir une Servlet Liste.

-Q6.1- Sachant que Titre et Description sont deux champs de saisie, et Add un bouton pour valider cette saisie, écrire (juste le nécessaire), en html, le contenu du fichier ServletListe.html décrivant la page web suivante:

figure Q6.1

```
<!--fichier html pour l'exemple de la figure Q6.1-->
<html>
     <body>
            <head>
                  <title>Servlet Liste</title>
            </head>
            <body bgcolor="white">
            <h3>Add</h3>
            <P>
            <form action="../servlet/ServletListe" method=POST>
                  Titre:
                        <input type=text size=20 name="titre">
                  <br>
                  Description:
                        <input type=text size=20 name="description">
                  <br>
                  <input type="submit" name="add" value="Add">
            </form>
      </body>
</html>
```

-Q6.2- En partant de la figure Q6.1 (page ???), écrire une servlet permettant de concevoir la figure suivante (page html) et de traiter son contenu ; sachant ce qui suit:

Si vous insérez les valeurs ("vert", "grenouille, gazon") respectivement dans (titre, description), puis vous validez votre saisie en appuyant sur le bouton "add", la servlet va saisir cette paire et l'afficher dans la zone d'affichage (au dessous du trait) et effacer le contenu des champs (titre, description). La zone d'affichage ne va contenir que la dernière paire de valeurs insérée. Il sera affiché en premier le titre, puis la description.

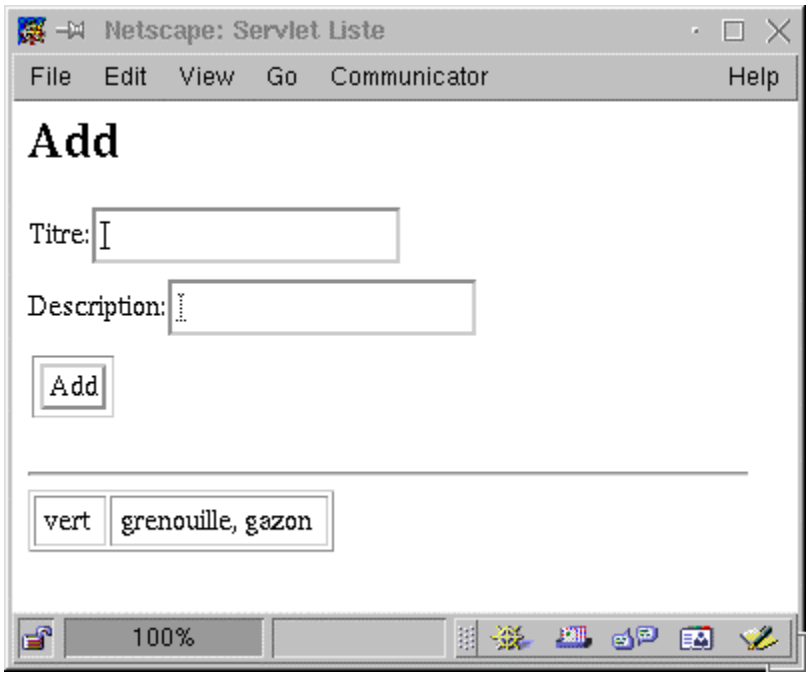


figure -Q6.2-

Attention : Vous devez écrire votre code avec une clarté limpide! Car vous allez devoir le réutiliser dans la question qui va suivre.

```
import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;


// On définit une classe pour sauvegarder un type donné d'élément.

class ItemBean {
    private String title;
    private String description;
    public ItemBean(String aTitle, String aDescription) {
        title = aTitle;
        description = aDescription;
    }
    public String getTitle() {
        return(title);
    }
    public String getDescription() {
        return(description);
    }
}
```

```
public class ServletListe extends HttpServlet {


    public void doPost(HttpServletRequest request,
                       HttpServletResponse response)
        throws IOException, ServletException {


        response.setContentType("text/html");
        PrintWriter out = response.getWriter();


        out.println("<!--fichier html pour l'exemple de la figure Q6.1-->");
        out.println("<html>");
        out.println("<body>");
        out.println("<head>");
        out.println("<title>Servlet Liste</title>");
        out.println("</head>");
        out.println("<body bgcolor=\"white\">");
        out.println("<h3>Add</h3>");
        out.println("<P>");
        out.println("<form action=\"../servlet/ServletListe\" method=POST>");
        out.println("Titre:");
        out.println("<input type=text size=20 name=\"titre\">");
        out.println("<br>");
        out.println("Description:");
        out.println("<input type=text size=20 name=\"description\">");
        out.println("<br>");
        out.println("<input type=\"submit\" name=\"add\" value=\"Add\">");
        out.println("</form>");

        HttpSession s = request.getSession(true);

        String title = request.getParameter("titre");
        String descr = request.getParameter("description");
        ItemBean UnBean = null;
        if (request.getParameter("add")!=null) {
            UnBean = new ItemBean(title, descr);
        }


        out.println("<hr WIDTH=\"100%\">");

        out.println("<table BORDER CELLPADDING=4>");

        if (UnBean != null) {

            String thisKey = UnBean.getTitle();
            String thisValue = UnBean.getDescription();
            out.println("<TR>");
            out.println("<TD>");
            out.println(thisKey);
            out.println("</TD>");
            out.println("<TD>");
            out.println(thisValue);
            out.println("</TD>");
            out.println("</TR>");
        }
        out.println("</table>");
        out.println("</body>");
        out.println("</html>");
    }


    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        doPost(request, response);
    }

}
```

-Q6.3- En partant du code réalisé à la deuxième question (Q6-2), nous désirons maintenant préserver les différentes saisies dans une liste qui résidera coté serveur, pour toute la durée de la session, indépendamment du nombre de requêtes établies ou bien du nombre de fenêtres (navigateur) ouvertes. Ainsi si un usager a deux navigateurs d'ouvert, il manipulera par l'intermédiaire du premier navigateur ou du second, la même liste et cela uniquement pendant la durée de la session. Vous allez implanter uniquement l'opération d'ajout dans cette liste. Les opérations comme l'édition de la liste ou le retrait d'une paire donnée de la liste ne doivent pas être implantées.

Par exemple, si vous insérez la paire ("vert", "grenouille, gazon"), vous allez obtenir la figure Q6.2 (page 12). Si, par la suite, vous insérez successivement les paires ("blanche", "poudre, barbe, boule"), et ("rouge", "drapeau, pomme, rosette"), vous allez obtenir cette figure :

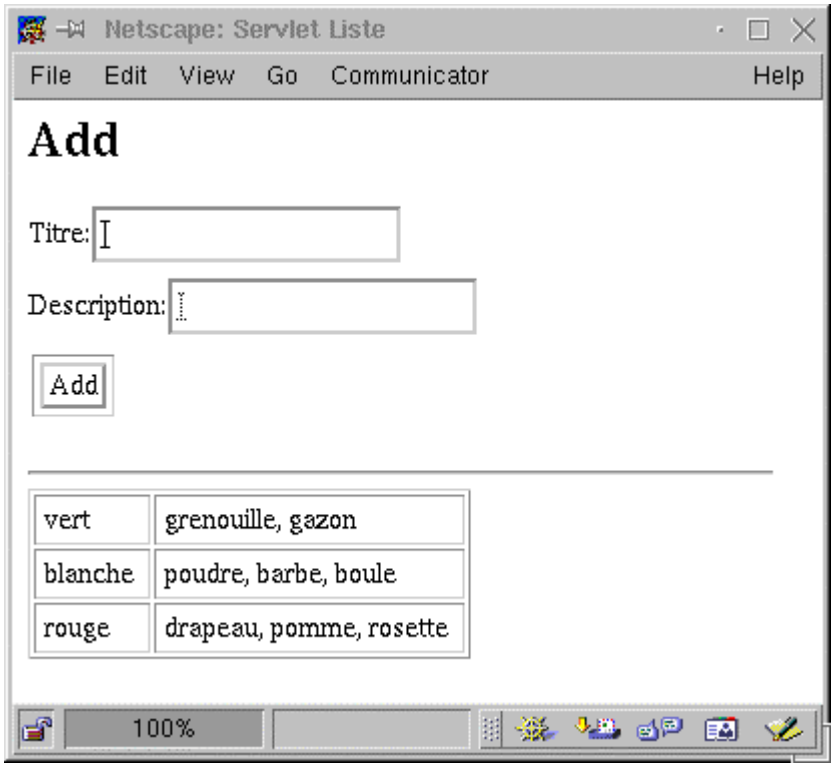


figure -Q6.3-

Attention : Pas besoin de récrire tout le code, écrire uniquement les modifications nécessaires à apporter au précédent programme (Q6-2). **Les modifications doivent être clairement présentées** (pas de jeu de labyrinthe … des flèches dans toutes les directions, et des étoiles par tout!)

La problématique ici est comment sauvegarder les anciennes valeurs insérées, durant une session donnée?
En gras les modifications à introduire dans le précédent programme.

```
import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

class ItemBean {
    private String title;
    private String description;
    public ItemBean(String aTitle, String aDescription) {
        title = aTitle;
        description = aDescription;
    }
    public String getTitle() {
        return(title);
    }
    public String getDescription() {
        return(description);
    }
}

public class ServletListe extends HttpServlet {

    Vector data = null;

    public void init() {
        data = new Vector();
    }


    public void doPost(HttpServletRequest request,
                       HttpServletResponse response)
        throws IOException, ServletException {


        response.setContentType("text/html");
        PrintWriter out = response.getWriter();


        out.println("<!--fichier html pour l'exemple de la figure Q6.1-->");
        out.println("<html>");
        out.println("<body>");
        out.println("<head>");
        out.println("<title>Servlet Liste</title>");
        out.println("</head>");
        out.println("<body bgcolor=\"white\">");
        out.println("<h3>Add</h3>");
        out.println("<P>");
        out.println("<form action=\"../servlet/ServletListe\" method=POST>");
        out.println("Titre:");
        out.println("<input type=text size=20 name=\"titre\">");
        out.println("<br>");
        out.println("Description:");
        out.println("<input type=text size=20 name=\"description\">");
        out.println("<br>");
        out.println("<input type=\"submit\" name=\"add\" value=\"Add\">");
        out.println("</form>");


        HttpSession s = request.getSession(true);

        s.setAttribute("data", data);

        data = (Vector) s.getAttribute("data");

        String title = request.getParameter("titre");
        String descr = request.getParameter("description");
```

```
        if (request.getParameter("add")!=null) {
            data.add(new ItemBean(title, descr));
        }

        // Attention ici: J'écrase de facto, l'attribut data pour le remplacer avec une
        // nouvelle donnée.
        // Certains ont proposé d'introduire directement les valeurs lues. Il y aura
        // écrasement des anciennes valeurs dès qu'elles portent le même nom d'attribut,
        // d'où erreur.

        s.setAttribute("data", data);

        out.println("<hr WIDTH=\"100%\">");

        out.println("<table BORDER CELLPADDING=4>");

        for (int i=0; i<data.size(); i++) {
            String thisKey = ((ItemBean) data.elementAt(i)).getTitle();
            String thisValue = ((ItemBean) data.elementAt(i)).getDescription();
            out.println("<TR>");
            out.println("<TD>");
            out.println(thisKey);
            out.println("</TD>");
            out.println("<TD>");
            out.println(thisValue);
            out.println("</TD>");
            out.println("</TR>");
        }

        out.println("</table>");
        out.println("</body>");
        out.println("</html>");
    }



    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        doPost(request, response);
    }

}


/*

Il serait intéressant aussi de voir comment faire le même exercice en utilisant pour
cela, getSession(false). Cette méthode retourne false si la session n'existe pas, et si
c'est le cas elle ne tente pas de la créer. Ce n'est pas le cas de getSession() ou
getSession(true) qui créent la session si cette dernière n'a pas été créée auparavant.

*/
```