



ĐẠI HỌC QUỐC GIA HÀ NỘI
VNU
Since 1906



INSTITUT
FRANCOPHONE
INTERNATIONAL

INSTITUT FRANCOPHONE INTERNATIONAL

Matière : Génie Logiciel (GL)

Modélisation et programmation orientée-objet avec Java

TP 1 : Un petit gestionnaire de tâches

Rédigé par:

ANDRÉ Perrault

Professeur : Ho Tuong Vinh

Promotion 21

Introduction

Notre premier TP de génie logiciel (GL) a pour objectif de concevoir et réaliser une application de gestion de tâche afin de nous rappeler les concepts de la modélisation avec UML et la programmation orientée-objet avec Java. Cela nous aidera aussi à nous familiariser avec l'environnement de développement intégré (IDE) libre, (Open Source) ECLIPSE et NetBeans. Ce rapport présente les principes de fonctionnement de notre système, les étapes de conception et de réalisation de notre application.

Nous allons dans un premier temps modéliser notre système. Déterminer les besoins fonctionnels et non fonctionnels du système ensuite définir les acteurs qui interagissent avec le système en utilisant le diagramme des cas d'utilisation. Construire le diagramme de classe et le diagramme de séquence du système. Puis, à la fin, nous allons implémenter notre système en vue d'avoir tous les fonctionnalités souhaité.

Spécification de l'application

Vous devez réaliser un petit gestionnaire de tâches pour une équipe de travail. Ce gestionnaire fournit à l'utilisateur les fonctionnalités suivantes :

1. Créer, modifier, supprimer, ajouter une tâche
2. Créer, modifier, supprimer, ajouter un membre
3. Assigner une tâche à un membre
4. Chercher et afficher tous les tâches assignées à un membre (par son ID)
5. Chercher et afficher tous les tâches en fonction de leur status (avec le nom du assigné)

Une tâche est composée de ces informations suivantes :

- ID
- Nom
- Une description
- Status : nouveau, en-progrès, terminé.

Un membre est composé de ces informations suivantes :

- ID
- Nom.

Exigences fonctionnelles

Ici, nous avons tous les exigences fonctionnelles que nous devons intégrés dans notre système.

- ✓ L'utilisateur doit être capable de créer, modifier, supprimer et ajouter un membre dans le système sans aucune complication.
- ✓ L'utilisateur doit être capable de créer, modifier, supprimer et ajouter une tâche dans le système sans aucune complication.
- ✓ L'utilisateur doit être capable d'assigner une tâche à un membre dans le système.
- ✓ L'utilisateur doit être capable de chercher et afficher tous les tâches assignées à un membre avec l'ID du membre.

- ✓ L'utilisateur doit être capable de chercher et afficher tous les tâches en fonction de leur statut tout en affichant le nom du assigné.

Exigences non-fonctionnelles

Nous avons ici quelques exigences non-fonctionnelles de notre système.

- ✓ **Performance** : le temps maximum de compilation de notre application doit être de deux (2) minutes.
- ✓ **Utilisabilité** : notre application doit être facile à utiliser, doit fournir une interface d'entrée et afficher les résultats en sorties.
- ✓ **Fiabilité** : notre application doit être très précis.
- ✓ **Robustesse** : notre application doit prendre en contre des données invalides saisies par l'utilisateur.
- ✓ **Adaptabilité** : notre application doit s'adapter à d'autres environnements.
- ✓ **Passage à l'échelle** : notre application doit être en mesure d'enregistrer et de manipuler de grandes quantités ou données

Diagramme cas d'utilisation

Dans cette partie, nous allons préciser les fonctionnalités du système dans un diagramme global de cas d'utilisation.

Notre diagramme de cas d'utilisation globale est composé de 3 sous-systèmes :

- ✓ Gestion des membres
- ✓ Gestion des tâches
- ✓ Assignment

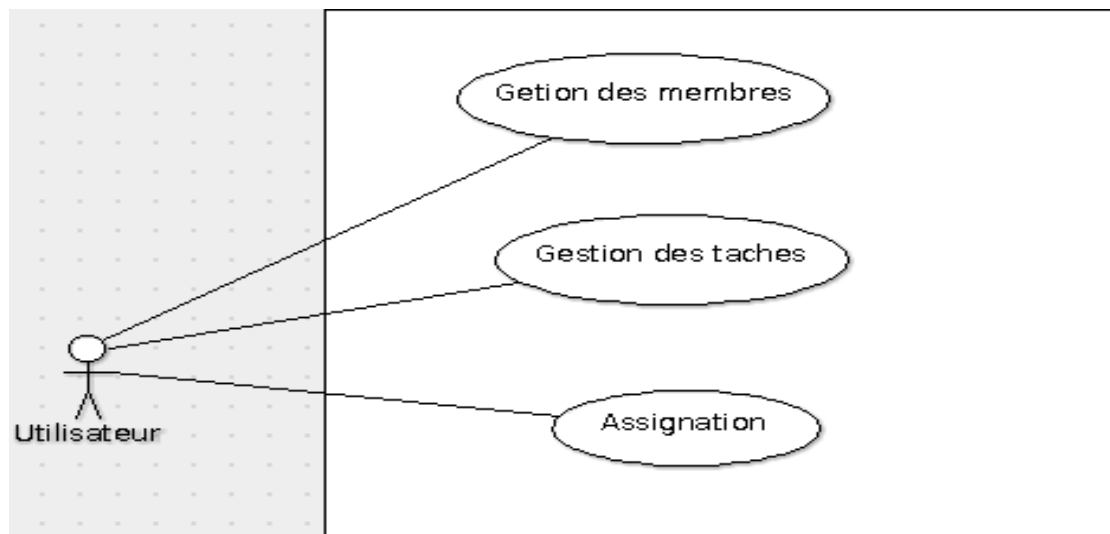


Figure 1_Diagramme de classe globale

Dans cette partie, nous allons préciser les fonctionnalités de chaque sous-système.

1) Représentation de gestion des membres

Dans ce diagramme, nous présentons de façon détaillée les fonctionnalités assurées par l'utilisateur dans le sous-système de gestion des membres.

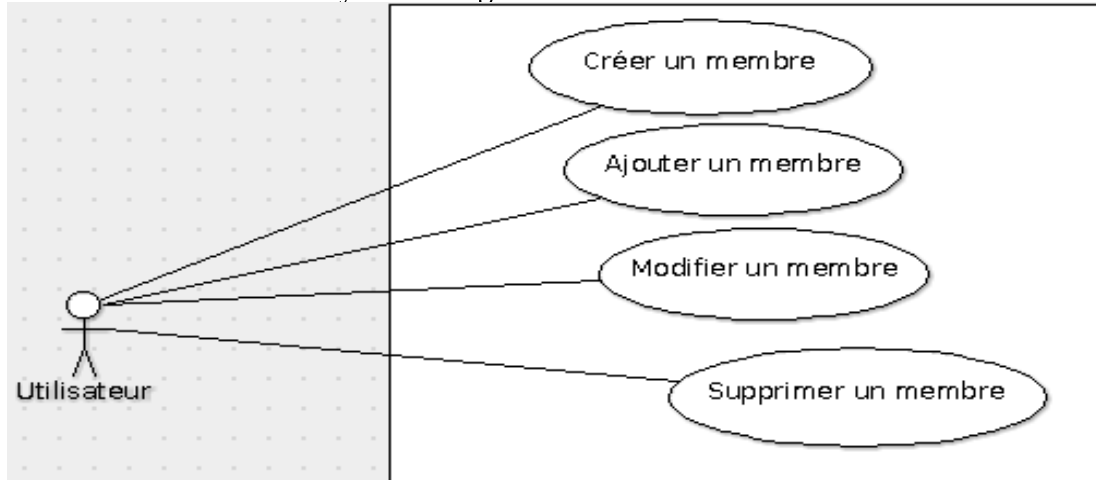


Figure 2_Diagramme de classe de gestion des membres

2) Représentation de gestion des tâches

Dans ce diagramme, nous présentons de façon détaillée les fonctionnalités assurées par l'utilisateur dans le sous-système de gestion des tâches.

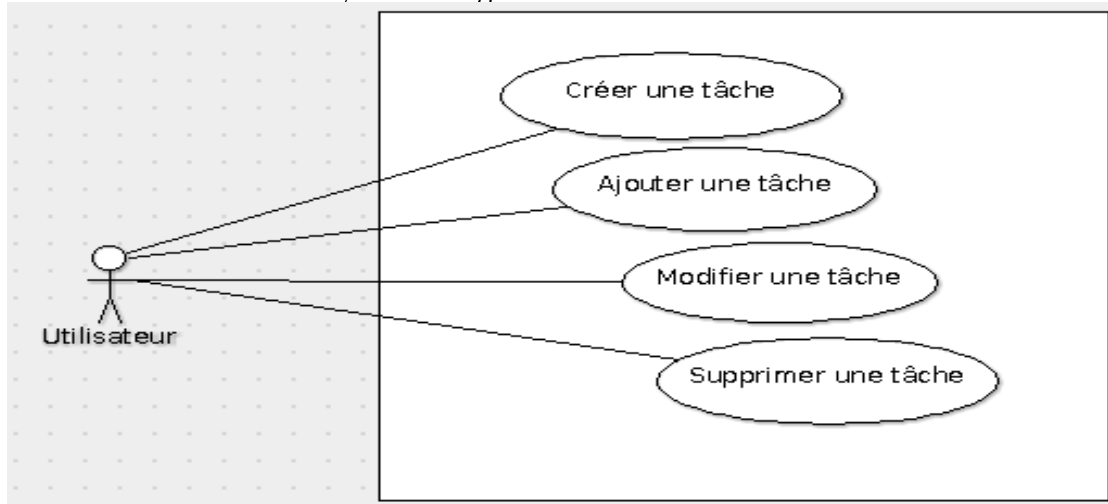


Figure 3_Diagramme de classe de gestion des tâches

3) Représentation de l'assignation

Dans ce diagramme, nous présentons de façon détaillée les fonctionnalités assurées par l'utilisateur dans le sous-système de l'assignation.

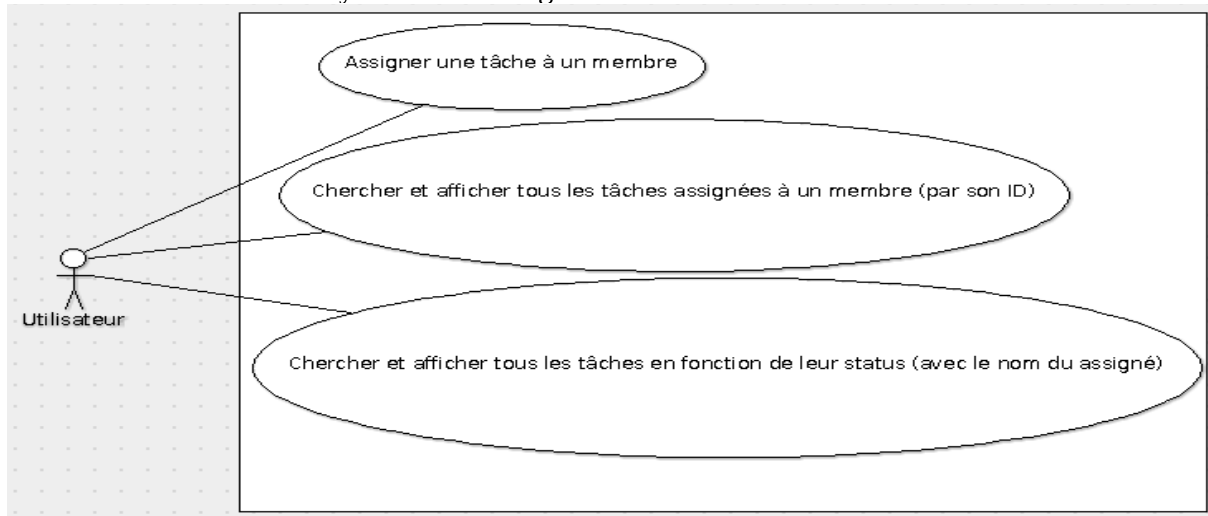


Figure 4_Diagramme d'assignation

Diagramme de classe

Le diagramme de classes exprime de manière générale la structure statique d'un système en termes de classes et de relations entre les classes. Une classe permet de décrire un ensemble d'objets (attributs et comportement), tandis qu'une relation ou association permet de faire apparaître des liens entre ces objets. Un diagramme de classes fait abstraction des aspects dynamiques et temporels : il permet de modéliser les vues statiques du système. Le diagramme de classes est le point central dans un développement orienté objet.

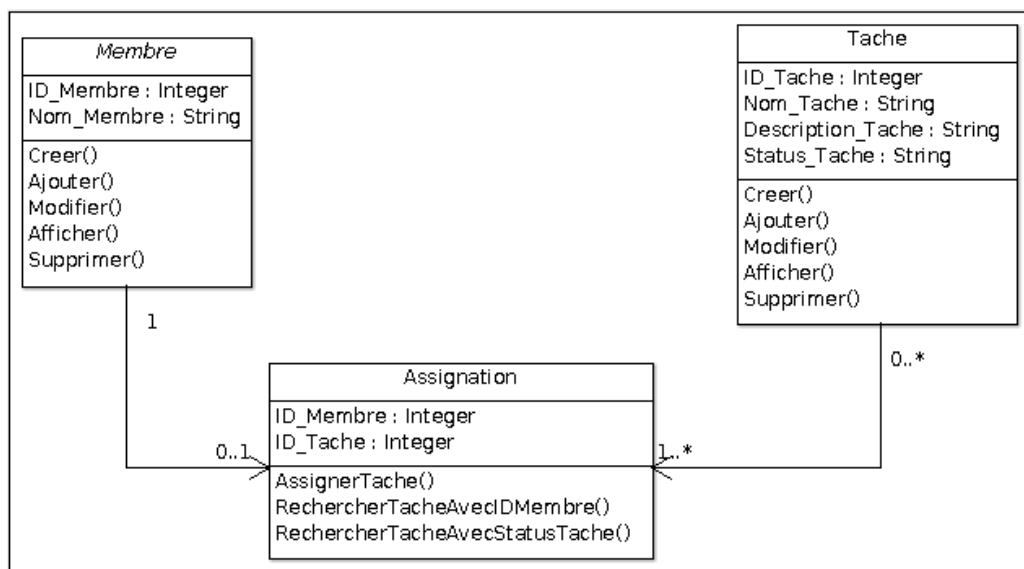


Figure 5_Diagramme des classes

Diagramme de séquence

Les diagrammes de séquence montrent des interactions entre objets selon un point de vue temporel. Le contexte des objets n'est pas représenté de manière explicite comme dans les diagrammes de collaboration. La représentation se concentre sur l'expression des interactions.

Un diagramme de séquence représente une interaction entre objets en insistant sur la chronologie des envois de messages. La notation est dérivée des « Object Message Sequence du Siemens Pattern Group ».

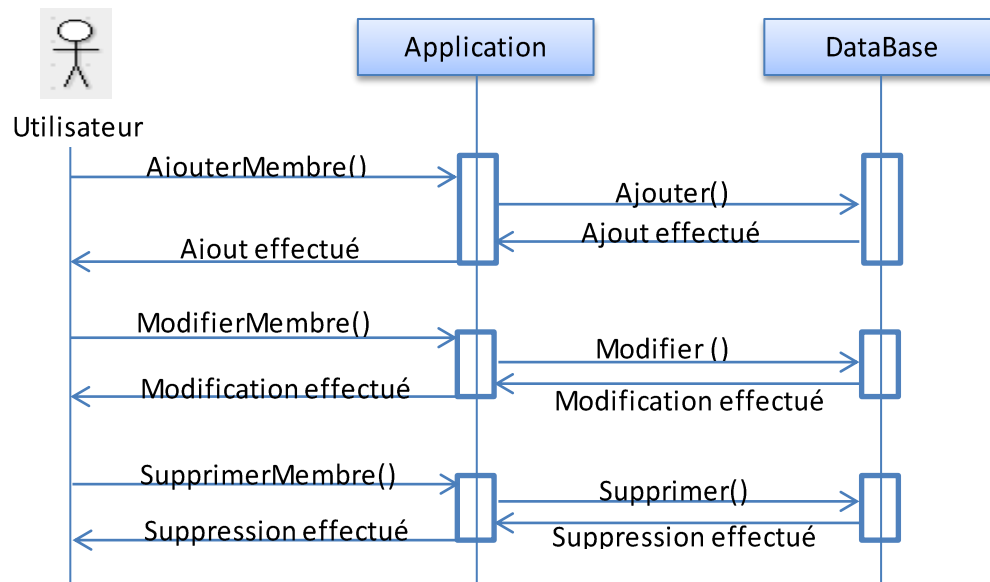


Figure 6_Diagramme de séquence de gestion de membre

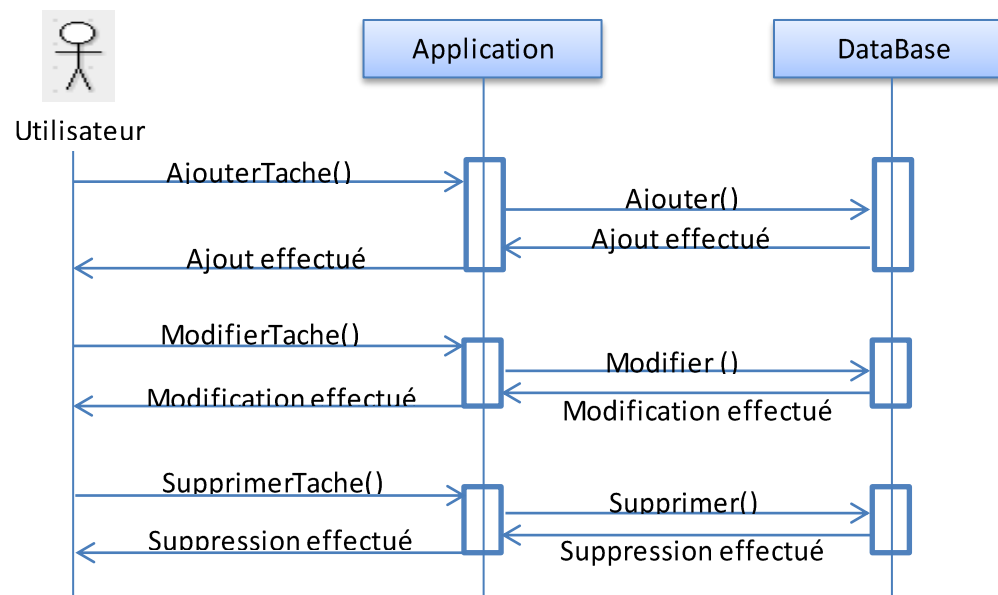
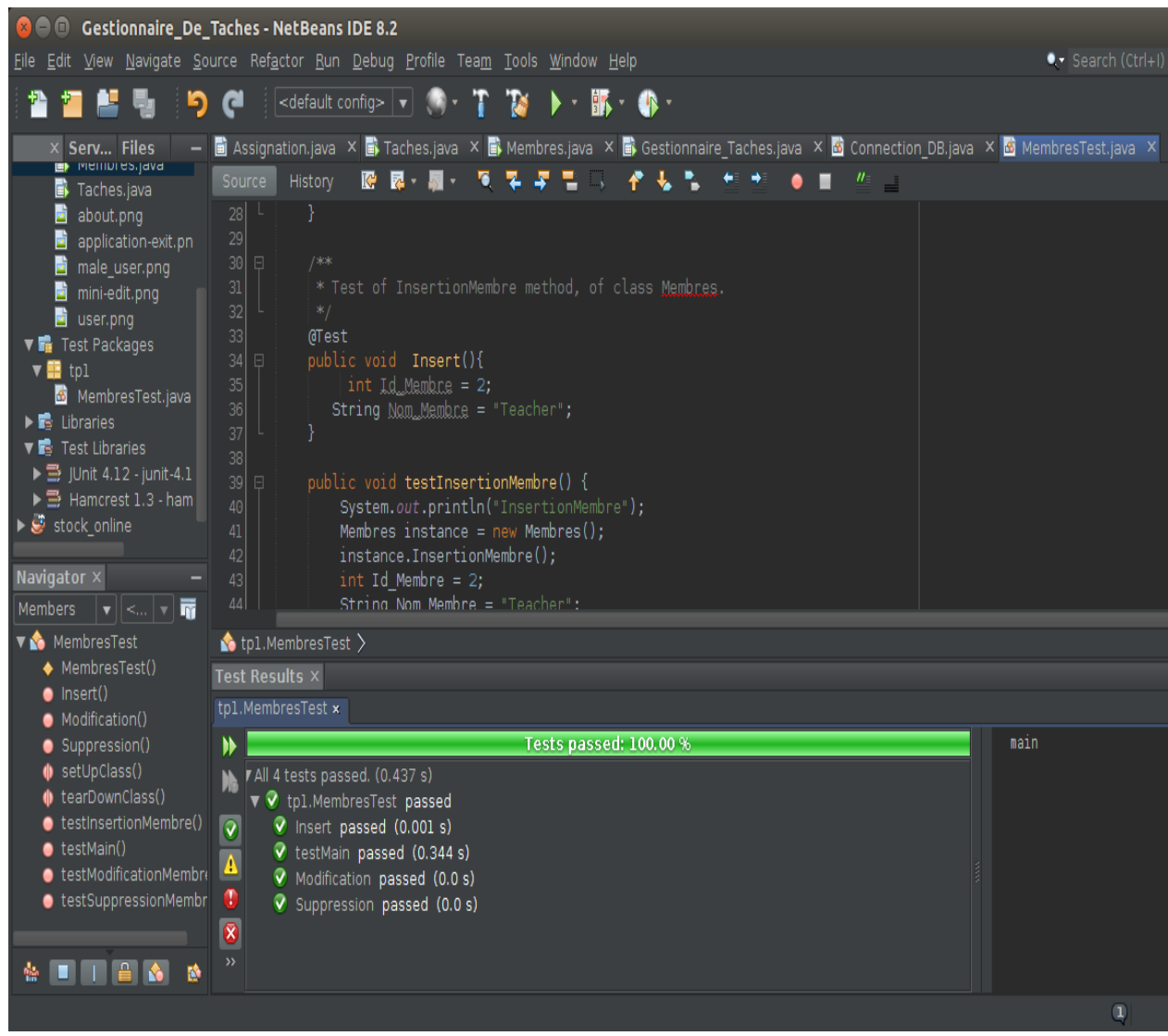


Figure 7_Diagramme de sequence de gestion de tache

Test unitaire sur la classe « Membre »

Dans cette section nous allons faire le test unitaire sur la classe Membre.



Nous avons ici notre test unitaire que nous avons effectué sur la classe « Membre » tout en tenant compte des trois (3) méthodes qui composent cette classe : InsertionMembre(), ModificationMembre() et SuppressionMembre(). La capture d'écran ci-dessus nous permet de visualiser le résultat du test unitaire avec un pourcentage de 100%.

Donc, nous pouvons dire que le test unitaire avec Junit a été parfaitement réussi.

Implémentation et tests :

Notre application a été développée avec le langage Java, sous Linux distribution Ubuntu (17.01), avec le langage de programmation JAVA et à l'aide de l'IDE Netbeans.

Nous avons créé 5 classes dans l'implémentation de notre application de gestion de tâches :

1. La classe « Membre » qui définit les attributs et méthodes d'un membre.
2. La classe « Tache » qui définit les attributs et méthodes d'une tâche.
3. La classe « Assignment » qui définit les attributs et méthodes de l'assignation des tâches à un membre.
4. La classe « Gestionnaire_Taches » qui joue le rôle de classe principale servant à l'exécution de notre application.
5. La classe « Connection_DB » qui sert à connecter notre programme à notre base de données.

Test d'acceptation

Ici nous avons l'interface principale de notre application. Pour créer et ajouter un membre il faut que l'utilisateur clique sur le bouton « Membres ».



Une fois que c'est fait une autre interface apparaît. C'est à ce moment que l'utilisateur peut ajouter, modifier ou supprimer un membre.



Maintenant nous allons ajouter un membre, avec l'ID = 3 et Nom = Perrault. L'utilisateur doit cliquer sur le bouton « Ajouter » après la saisie des informations du membre.

The screenshot shows a window titled "Gestion des membres". On the left, there are two input fields: "ID" with the value "3" and "Nom" with the value "Perrault". To the right of these fields is a large empty rectangular box. At the bottom, there are four buttons: "Retour", "Ajouter", "Modifier", and "Supprimer". The "Ajouter" button is highlighted with a blue border, indicating it has been clicked.

Cette figure nous montre que le membre a été enregistré avec succès.

The screenshot shows the same "Gestion des membres" window. The "ID" and "Nom" input fields are now empty. The large rectangular box now contains a table with the following data:

ID Membre	Nom
3	Perrault

The "Ajouter" button remains highlighted with a blue border.

Maintenant essayons de modifier le nom du membre à l'aide de son ID, l'utilisateur doit cliquer sur le bouton « Modifier » après la saisie des informations du membre.

The screenshot shows the "Gestion des membres" window. The "ID" input field contains the value "3", and the "Nom" input field contains the value "Andre". The table in the center still shows the member "Perrault" with ID 3. The "Modifier" button at the bottom is highlighted with a blue border, indicating it has been clicked.

La figure suivante montre le résultat de la modification du membre.

The screenshot shows a window titled "Gestion des membres". On the left, there are two input fields labeled "ID" and "Nom". The "ID" field contains the value "3". To the right of these fields is a table with two columns: "ID Membre" and "Nom". The table contains one row with the values "3" and "Andre". At the bottom of the window, there are four buttons: "Retour", "Ajouter", "Modifier" (which is highlighted with a blue border), and "Supprimer".

ID Membre	Nom
3	Andre

Pour mieux tester la suppression, nous allons ajouter d'autres membres dans la base de données.

The screenshot shows the same "Gestion des membres" window. The "ID" and "Nom" input fields are empty. The table now contains four rows of data:

ID Membre	Nom
1	JB
3	Andre
6	Peret
7	Marc

The "Ajouter" button is now highlighted with a blue border.

Maintenant l'utilisateur doit cliquer sur le bouton « Supprimer » et entrer l'ID du membre qu'il aimerait supprimer.

The screenshot shows a small dialog box titled "Input". It has a question mark icon on the left. The text inside says "Entrer l'ID du membre à supprimer". Below this text is an input field containing the value "3". At the bottom of the dialog, there are two buttons: "Cancel" and "OK".

La figure suivante montre le résultat de la suppression du membre.

The screenshot shows a web application window titled "Gestion des membres". On the left, there are input fields for "ID" and "Nom". In the center, a table displays the remaining members:

ID Membre	Nom
1	JB
6	Peret
7	Marc

At the bottom, there are four buttons: "Retour", "Ajouter", "Modifier", and "Supprimer". The "Supprimer" button is highlighted with a blue border.

Une fois terminée, pour retourner au menu principal l'utilisateur doit cliquer sur le bouton retour.

De retour au menu principal, l'utilisateur doit cliquer sur le bouton «Tâche» pour accéder à l'interface qui permet de faire la gestion de tâche. Une fois que c'est fait, l'interface apparaît. L'utilisateur peut maintenant faire des ajouts, modifier ou supprimer une tâche.

Essayons d'enregistrer une tâche dans la base de données.

The screenshot shows a web application window titled "Gestion des taches". On the left, there are input fields for "ID" (containing "2"), "Nom" (containing "DevOps"), "Description" (containing "Developper une ap"), and a "Status" dropdown menu (set to "Nouveau"). On the right, there is a large empty rectangular box. At the bottom, there are four buttons: "Retour", "Ajouter", "Modifier", and "Supprimer".

La figure ci-dessous montre que le résultat a été enregistré.

The screenshot shows the same "Gestion des taches" window, but now the table on the right contains one row of data:

ID Tache	Nom	Description	Status
2	DevOps	Developper une apk de vote	Nouveau

The input fields on the left remain the same, and the buttons at the bottom are also present.

Maintenant essayons de modifier le nom de la tâche à l'aide de son ID, l'utilisateur doit cliquer sur le bouton « Modifier » après le saisi des informations de la tâche.

Saisissons les informations suivants : ID = 2, Nom=Testeur, Description=Tester l'application de vote, Status=En-progrès.

ID Tache	Nom	Description	Status
2	Testeur	Tester l'application de vote	En-progrès

Pour mieux tester la suppression, nous allons ajouter d'autres tâches dans la base de données. Puis l'utilisateur doit cliquer sur le bouton « Supprimer » et entrer l'ID du membre qu'il aimerait supprimer.

ID Tache	Nom	Description	Status
2	Testeur	Tester l'applica...	En-progrès
3	Designer	Design the inte...	Terminé
4	DevOps	Developper une...	Nouveau
7	Srum Master	Gere l'equipe d...	Terminé

La figure suivante montre le résultat de la suppression de la tâche.

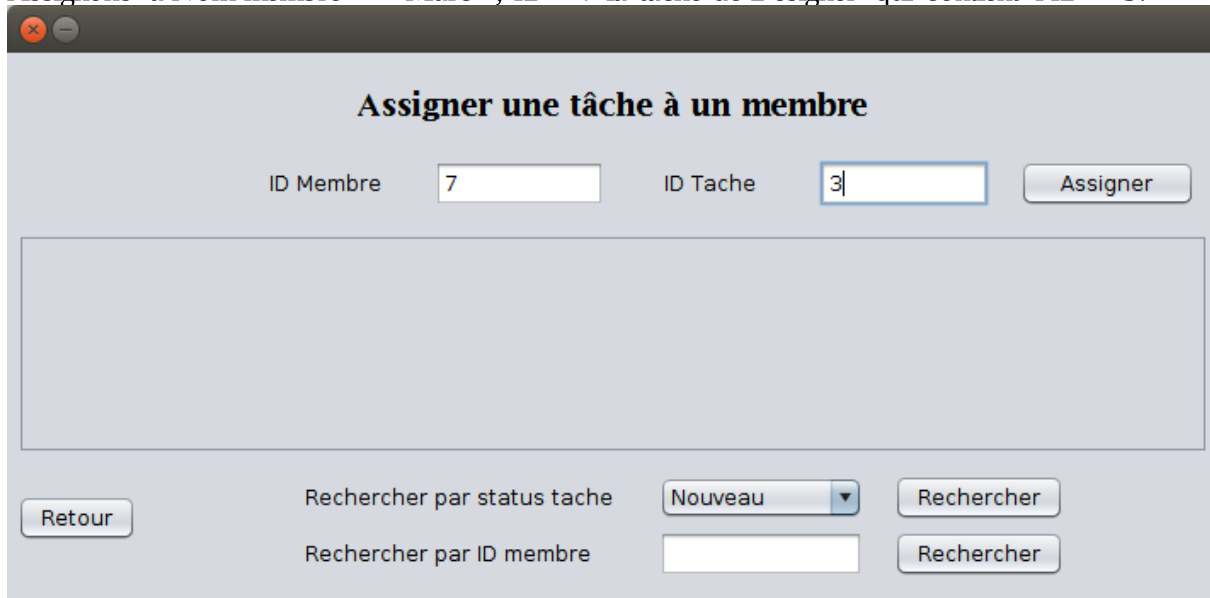
ID Tache	Nom	Description	Status
3	Designer	Design the inte...	Terminé
4	DevOps	Developper une...	Nouveau
7	Srum Master	Gere l'equipe d...	Terminé

Une fois terminée, pour retourner au menu principal l'utilisateur doit cliquer sur le bouton retour.

De retour à nouveau menu principal, l'utilisateur doit cliquer sur le bouton « Gestion Taches » pour accéder à l'interface qui permet de faire l'assignation. Une fois que c'est fait, l'interface apparait. L'utilisateur peut maintenant assigner des tâches à un membre.

À ce niveau, essayons d'assigner des tâches à un membre.

Assignons à Nom membre = « Marc », ID = 7 la tâche de Designer qui contient l'ID = 3.



The screenshot shows a web application window titled "Assigner une tâche à un membre". It features two input fields: "ID Membre" with the value "7" and "ID Tache" with the value "3". An "Assigner" button is to the right of the "ID Tache" field. Below these fields is a large, empty rectangular box. At the bottom, there are two search options: "Rechercher par status tache" with a dropdown menu set to "Nouveau" and a "Rechercher" button, and "Rechercher par ID membre" with an empty input field and a "Rechercher" button. A "Retour" button is located on the left side of the bottom section.

Puis, assignons à ce même membre la tâche de DevOps qui contient l'ID = 4.



This screenshot is identical to the previous one, showing the "Assigner une tâche à un membre" interface. The "ID Membre" field still contains "7", but the "ID Tache" field now contains "4". The "Assigner" button remains to the right. The rest of the interface, including the empty box and the search options at the bottom, is unchanged.

L'utilisateur peut maintenant rechercher et afficher tous les tâches assignées à un membre (par son ID) ou tous les tâches en fonction de leur statut (avec le nom du assigné).

Recherchons et affichons premièrement tous les tâches en fonction de leur statut (avec le nom du assigné).

L'utilisateur doit faire déroulé la liste qui se trouve à droite de « Rechercher par status tache », sélectionner l'un des éléments de la liste, puis cliquer sur le bouton Rechercher.

1) Avec statut tâche : Nouveau

The screenshot shows a window titled "Assigner une tâche à un membre". At the top, there are input fields for "ID Membre" and "ID Tache", followed by an "Assigner" button. Below this is a table with the following data:

ID Tache	Nom Tache	Description Tache	Status Tache	Nom Mem...
4	DevOps	Developper une application qui permet de voter en ligne	Nouveau	Marc

At the bottom, there are two search sections. The first, "Rechercher par status tache", has a dropdown menu currently set to "Nouveau" and a "Rechercher" button. The second, "Rechercher par ID membre", has an empty input field and a "Rechercher" button. A "Retour" button is located on the left side.

2) Avec statut tâche : Terminé

The screenshot shows the same window as before, but the dropdown menu for "Rechercher par status tache" is now set to "Terminé". The table below shows a different task:

ID Tache	Nom Tache	Description Tache	Status Tache	Nom Membre
3	Designer	Design the interface of the application	Terminé	Marc

The rest of the interface, including the "Assigner" button, the other search fields, and the "Retour" button, remains the same.

Maintenant nous allons rechercher et afficher tous les tâches assignées à un membre (par son ID).

L'utilisateur doit saisir l'ID du membre à rechercher dans la barre de recherche à droite de « Rechercher par ID membre », puis cliquer sur le bouton Rechercher.

The screenshot shows a window titled "Assigner une tâche à un membre". At the top, there are two input fields: "ID Membre" and "ID Tache", followed by an "Assigner" button. Below these is a large empty rectangular box. At the bottom, there are two search options: "Rechercher par status tache" with a dropdown menu set to "Nouveau" and a "Rechercher" button, and "Rechercher par ID membre" with an input field containing the number "7" and another "Rechercher" button. A "Retour" button is located on the left side of the bottom section.

Résultat renvoyé par l'application.

This screenshot shows the same window as the previous one, but now it displays a table of search results. The table has five columns: "ID Tache", "Nom Tache", "Description Tache", "Status Tache", and "Nom Membre". There are two rows of data. Below the table is the same search interface as before, with the "Rechercher par ID membre" button highlighted by a blue border.

ID Tache	Nom Tache	Description Tache	Status Tache	Nom Membre
3	Designer	Design the interface of the application	Terminé	Marc
4	DevOps	Developper une application qui permet de voter e...	Nouveau	Marc

Après avoir effectué tous ces tests sur l'application, nous concluons que notre application marche correctement et respecte tous les normes et spécifications demandées. Donc le test d'acceptation est validé.

Conclusion

À la fin de ce TP, nous pourrions dire que tous les objectifs fixés ont été atteints, ça nous a permis d'améliorer notre compréhension de la programmation orientée objet avec java et nous familiarisé mieux avec le IDE Netbeans. Après pas mal de test, nous pourrions aussi dire que les spécifications demandées ont toutes été respectées et notre application marche correctement. Cependant, il reste quelques efforts à fournir dans la mise en application des bonnes pratiques de programmation, la maîtrise du langage java. Pour finir, ce TP a été un excellent exercice pour l'assimilation des concepts de base du cours de génie logiciel.

Code source (an annexe)

I. Class Membre

```
package tpl;

import java.sql.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author perrault
 */
public class Membres extends javax.swing.JFrame {
    Connection_DB conn;
    Statement stm;
    ResultSet Rs;
    DefaultTableModel model = new DefaultTableModel();
    /**
     * Creates new form Membre
     */

    public Membres() {
        initComponents();
    }

    private void jTextField_NomActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void Back_To_MenuActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        this.dispose();
    }

    private void Ajouter_MembreActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        if(("".equals(jTextField_ID.getText()))||("".equals(jTextField_Nom.getText()))){
            JOptionPane.showMessageDialog(null,"Il faut remplir toutes les cases.");
        }else{
            InsertionMembre();
            AfficheInfoMembre();
        }
    }

    private void Supprimer_MembreActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        int Id_membre = Integer.valueOf(JOptionPane.showInputDialog("Entrer l'ID du membre à supprimer"));
        SuppressionMembre(Id_membre);
        AfficheInfoMembre();
    }
}
```



```

private void Modifier_MembreActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(("".equals(jTextField_ID.getText()))||("".equals(jTextField_Nom.getText()))){
        JOptionPane.showMessageDialog(null,"Il faut remplir toutes les cases.");
    }else{
        ModificationMembre();
        AfficheInfoMembre();}
}

public void InsertionMembre(){
    int Id_Membre = Integer.valueOf(jTextField_ID.getText());
    String Nom_membre = jTextField_Nom.getText();

    conn = new Connection_DB();
    conn.InsererMembre(Id_Membre, Nom_membre);
    jTextField_ID.setText("");
    jTextField_Nom.setText("");
}

public void ModificationMembre(){
    int Id_Membre = Integer.valueOf(jTextField_ID.getText());
    String Nom_membre = jTextField_Nom.getText();

    conn = new Connection_DB();
    conn.ModifierMembre(Id_Membre, Nom_membre);
    jTextField_ID.setText("");
    jTextField_Nom.setText("");
}

//effacer les infos dans la base
public void SuppressionMembre(int id) {
    conn = new Connection_DB();
    Rs = conn.Delete_Membre(id);
}

//afficher tous les membres
public void AfficheInfoMembre(){
    try {
        conn = new Connection_DB();
        model.setColumnCount(0);
        model.addColumn("ID Membre");
        model.addColumn("Nom");

        model.setRowCount(0);
        Rs = conn.ListerMembre();
        while(Rs.next()){
            Object[] membre = {Rs.getInt(1),Rs.getString(2)};
            model.addRow(membre);
        }
        if(model.getRowCount()==0)
            JOptionPane.showMessageDialog(null,"Il n'y a pas de membre!");
        Table_Membre.setModel(model);
    } catch (SQLException ex) {
        Logger.getLogger(Membres.class.getName()).log(Level.SEVERE, null, ex);
    }
}
}

```

```

public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)
    //</editor-fold>

    //</editor-fold>
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(() -> {
        new Membres().setVisible(true);
    });
}

// Variables declaration - do not modify
private javax.swing.JButton Ajouter_Membre;
private javax.swing.JButton Back_To_Menu;
private javax.swing.JLabel ID_Membre;
private javax.swing.JButton Modifier_Membre;
private javax.swing.JLabel Nom_Membre;
private javax.swing.JButton Supprimer_Membre;
private javax.swing.JTable Table_Membre;
private javax.swing.JLabel jLabel1;
private javax.swing.JPanel jPanel1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTextField jTextFieldID;
private javax.swing.JTextField jTextFieldNom;
// End of variables declaration
}

```

II. Class Tache

```

package tpi;

import java.sql.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**...4 lines */
public class Taches extends javax.swing.JFrame {

    Connection_DB conn;
    Statement stm;
    ResultSet Rs;
    DefaultTableModel model = new DefaultTableModel();
    /** Creates new form Membre ...3 lines */
    public Taches() {
        initComponents();
    }

    /** This method is called from within the constructor to initialize the form ...5 lines */
    @SuppressWarnings("unchecked")
    Generated Code

    private void Back_to_MenuActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        this.dispose();
    }
}

```

```

private void Ajouter_TacheActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(("".equals(jTextField_ID.getText()))||("".equals(jTextField_Nom.getText()))||("".equals(jTextField_Description.getText()))){
        JOptionPane.showMessageDialog(null,"Il faut remplir toutes les cases.");
    }else{
        InsertionTache();
        AfficheInfoTache();
    }
}

private void Modifier_TacheActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(("".equals(jTextField_ID.getText()))||("".equals(jTextField_Nom.getText()))||("".equals(jTextField_Description.getText()))){
        JOptionPane.showMessageDialog(null,"Il faut remplir toutes les cases.");
    }else{
        ModificationTache();
        AfficheInfoTache();
    }
}

private void Supprimer_TacheActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int Id_Tache = Integer.valueOf(JOptionPane.showInputDialog("Entrer l'ID de la tache à supprimer"));
    SuppressionTache(Id_Tache);
    AfficheInfoTache();
}

public void InsertionTache(){
    int Id_tache = Integer.valueOf(jTextField_ID.getText());
    String Nom_tache = jTextField_Nom.getText();
    String description = jTextField_Description.getText();
    String status = Choix_Status.getSelectedItem().toString();

    conn = new Connection_DB();
    conn.InsererTache(Id_tache, Nom_tache,description,status);
    jTextField_ID.setText("");
    jTextField_Nom.setText("");
    jTextField_Description.setText("");
    Choix_Status.setSelectedItem("");
}

public void ModificationTache(){
    int Id_tache = Integer.valueOf(jTextField_ID.getText());
    String Nom_tache = jTextField_Nom.getText();
    String description = jTextField_Description.getText();
    String status = Choix_Status.getSelectedItem().toString();

    conn = new Connection_DB();
    conn.ModifierTache(Id_tache, Nom_tache, description, status);
    jTextField_ID.setText("");
    jTextField_Nom.setText("");
    jTextField_Description.setText("");
    Choix_Status.setSelectedItem("");
}

//effacer les infos dans la base
public void SuppressionTache(int id) {
    conn = new Connection_DB();
    Rs = conn.Delete_Tache(id);
}

```

```

//afficher tous les tâches
public void AfficheInfoTache(){
try {
    conn = new Connection_DB();
    model.setColumnCount(0);
    model.addColumn("ID Tache");
    model.addColumn("Nom");
    model.addColumn("Description");
    model.addColumn("Status");

    model.setRowCount(0);
    Rs = conn.ListerTache();
    while(Rs.next()){
        Object[] task = {Rs.getInt(1),Rs.getString(2),Rs.getString(3),Rs.getString(4)};
        model.addRow(task);
    }
    if(model.getRowCount()==0)
        JOptionPane.showMessageDialog(null,"Il n'y a pas de tache!");
    Table_Taches.setModel(model);
} catch (SQLException ex) {
    Logger.getLogger(Membres.class.getName()).log(Level.SEVERE, null, ex);
}
}

```

```

public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)
    //</editor-fold>

    //</editor-fold>
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(() -> {
        new Taches().setVisible(true);
    });
}

```

```

// Variables declaration - do not modify
private javax.swing.JButton Ajouter_Tache;
private javax.swing.JButton Back_to_Menu;
private javax.swing.JComboBox<String> Choix_Status;
private javax.swing.JLabel Description_Tache;
private javax.swing.JLabel ID_Tache;
private javax.swing.JButton Modifier_Tache;
private javax.swing.JLabel Nom_Tache;
private javax.swing.JLabel Status_Tache;
private javax.swing.JButton Supprimer_Tache;
private javax.swing.JTable Table_Taches;
private javax.swing.JLabel jLabel1;
private javax.swing.JPanel jPanel1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTextField jTextField_Description;
private javax.swing.JTextField jTextField_ID;
private javax.swing.JTextField jTextField_Nom;
// End of variables declaration
}

```

III. Class Assignment

```
package tpl;

import java.awt.HeadlessException;
import java.sql.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**...4 lines */
public class Assingation extends javax.swing.JFrame {

    Connection_DB conn;
    Statement stm;
    ResultSet Rs;
    DefaultTableModel model = new DefaultTableModel();
    /** Creates new form Membre ...3 lines */
    public Assingation() {
        initComponents();
    }

    /** This method is called from within the constructor to initialize the form ...5 lines */
    @SuppressWarnings("unchecked")
    Generated Code

    private void ID_TacheActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void ID_MembreActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jTextField_Rechercher_Status_TacheActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void Rechercher_Par_ID_MembreActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        if(!"".equals(jTextField_Rechercher_Par_ID_Membre.getText())){
            JOptionPane.showMessageDialog(null,"Il faut remplir la case.");
        }else{
            model = (DefaultTableModel) Table_Assossiation.getModel();
            model.setColumnCount(0);
            int status = Integer.valueOf(jTextField_Rechercher_Par_ID_Membre.getText());
            RechercherParIDMembre(status);}
    }

    private void Back_To_MenuActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        this.dispose();
    }

    private void Assigner_tacheActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        if(!"".equals(ID_Membre.getText())||!"".equals(ID_Tache.getText())){
            JOptionPane.showMessageDialog(null,"Il faut remplir toutes les cases.");
        }else{
            InsertionAssossiation();
            //AffichageAssingation();
        }
    }
}
```

```

private void Rechercher_Par_Status_TacheActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    model = (DefaultTableModel) Table_Assossiation.getModel();
    model.setColumnCount(0);
    String status = jTextField_Rechercher_Status_Tache.getSelectedItem().toString();
    RechercherParStatusTache(status);
}

public void InsertionAssossiation(){
    int Id_membre = Integer.valueOf(ID_Membre.getText());
    int Id_tache = Integer.valueOf(ID_Tache.getText());

    conn = new Connection_DB();
    conn.InsererAssigation(Id_membre, Id_tache);
    ID_Membre.setText("");
    ID_Tache.setText("");
}

//rechercher et afficher tous les tâches en fonction de leur status
public void RechercherParStatusTache(String stat) {
    conn = new Connection_DB();

    model.addColumn("ID Tache");
    model.addColumn("Nom Tache");
    model.addColumn("Description Tache");
    model.addColumn("Status Tache");
    model.addColumn("Nom Membre");
    try{
        model.setRowCount(0);
        Rs = conn.Search_Status_Task(stat);
        while(Rs.next()){
            Object[] client = {Rs.getInt(1),Rs.getString(2),Rs.getString(3),Rs.getString(4),
                               Rs.getString(5)};
            model.addRow(client);
        }
        if(model.getRowCount()==0)
            JOptionPane.showMessageDialog(null,"Il n'y a pas de membre!");
    }catch(SQLException | HeadlessException e){
        System.err.println(e);
        JOptionPane.showMessageDialog(null,e.getMessage());
    }
    Table_Assossiation.setModel(model);
}

//rechercher et afficher tous les tâches avec l'id membre
public void RechercherParIDMembre(int idmembre) {
    conn = new Connection_DB();

    model.addColumn("ID Tache");
    model.addColumn("Nom Tache");
    model.addColumn("Description Tache");
    model.addColumn("Status Tache");
    model.addColumn("Nom Membre");
    try{
        model.setRowCount(0);
        Rs = conn.Search_ID_Membre(idmembre);
        while(Rs.next()){
            Object[] client = {Rs.getInt(1),Rs.getString(2),Rs.getString(3),Rs.getString(4),
                               Rs.getString(5)};
            model.addRow(client);
        }
        if(model.getRowCount()==0)
            JOptionPane.showMessageDialog(null,"Il n'y a pas de membre!");
    }catch(SQLException | HeadlessException e){
        System.err.println(e);
        JOptionPane.showMessageDialog(null,e.getMessage());
    }
    Table_Assossiation.setModel(model);
}
}

```

```

//afficher tous les membres
public void AfficheInfoMembre(){
try {
    conn = new Connection_DB();
    model.addColumn("ID Tache");
    model.addColumn("Nom Tache");
    model.addColumn("Description Tache");
    model.addColumn("Status Tache");
    model.addColumn("ID Membre");
    model.addColumn("Nom Membre");

    model.setRowCount(0);
    Rs = conn.ListerMembre();
    while(Rs.next()){
        Object[] membre = {Rs.getInt(1) ,Rs.getString(2),Rs.getString(3),Rs.getString(4),
                            Rs.getInt(5), Rs.getString(6)};
        model.addRow(membre);
    }
    if(model.getRowCount()==0)
        JOptionPane.showMessageDialog(null,"Aucune tache est assigner a un/des membre(s)!");
    Table_Assossiation.setModel(model);
} catch (SQLException ex) {
    Logger.getLogger(Membres.class.getName()).log(Level.SEVERE, null, ex);
}
}

public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
    * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException | InstantiationException | IllegalAccessException | javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(Assignation.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(() -> {
        new Assignation().setVisible(true);
    });
}

// Variables declaration - do not modify
private javax.swing.JButton Assigner_tache;
private javax.swing.JButton Back_To_Menu;
private javax.swing.JTextField ID_Membre;
private javax.swing.JTextField ID_Tache;
private javax.swing.JLabel Membre;
private javax.swing.JLabel Rechercher_ID_Membre;
private javax.swing.JButton Rechercher_Par_ID_Membre;
private javax.swing.JButton Rechercher_Par_Status_Tache;
private javax.swing.JLabel Rechercher_Status_Tache;
private javax.swing.JTable Table_Assossiation;
private javax.swing.JLabel Tache;
private javax.swing.JLabel jLabel1;
private javax.swing.JPanel jPanel1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTextField jTextFieldRechercher_Par_ID_Membre;
private javax.swing.JComboBox<String> jTextFieldRechercher_Status_Tache;
// End of variables declaration
}

```

IV. Class Gestionnaire Taches

```
package tp1;

import javax.swing.JOptionPane;

/**...4 lines */
public class Gestionnaire_Taches extends javax.swing.JFrame {

    /** Creates new form Gestionnaire_Taches ...3 lines */
    public Gestionnaire_Taches() {
        initComponents();
    }

    /** This method is called from within the constructor to initialize the form ...5 lines */
    @SuppressWarnings("unchecked")
    Generated Code

    private void MemberActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //this.dispose();
        new Membres().show();
    }

    private void TasksActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //this.dispose();
        new Taches().show();
    }

    private void Gestion_TachesActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //this.dispose();
        new Assigination().show();
    }

    private void Menu_MembreActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        new Membres().show();
    }

    private void Menu_Gestion_TacheActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        new Assigination().show();
    }

    private void Menu_TacheActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        new Taches().show();
    }

    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        int Ex = JOptionPane.showConfirmDialog(null, "Exit?");
        if(Ex == JOptionPane.YES_OPTION){
            System.exit(0);
        }
    }
}
```



```

public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)

    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(() -> {
        new Gestionnaire_Taches().setVisible(true);
    });
}

// Variables declaration - do not modify
private javax.swing.JButton Gestion_Taches;
private javax.swing.JButton Member;
private javax.swing.JMenu Menu_Exit;
private javax.swing.JMenuItem Menu_Gestion_Tache;
private javax.swing.JMenuItem Menu_Membre;
private javax.swing.JMenuItem Menu_Tache;
private javax.swing.JButton Tasks;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JMenu jMenu2;
private javax.swing.JMenu jMenu3;
private javax.swing.JMenuBar jMenuBar1;
private javax.swing.JMenuItem jMenuItem1;
private javax.swing.JMenuItem jMenuItem2;
private javax.swing.JPanel jPanel1;
private javax.swing.JPopupMenu.Separator jSeparator1;
private javax.swing.JPopupMenu.Separator jSeparator2;
// End of variables declaration
}

```

V. Class Connection DB

```
package tpl;
import java.sql.*;
import javax.swing.JOptionPane;

public class Connection_DB {
    Connection con;
    Statement stm;

    public Connection_DB(){
        try{
            Class.forName("com.mysql.jdbc.Driver");
        }catch(ClassNotFoundException e){
            JOptionPane.showMessageDialog(null, e); //pour afficher l'erreur
        }
        try{
            con=(Connection) DriverManager.getConnection("jdbc:mysql://127.0.0.1/TP1","root","qwertyuiop");
        }catch(SQLException e){
            JOptionPane.showMessageDialog(null, e);
        }
    }

    public ResultSet Search_ID_Membre(int ID_Membre){
        ResultSet rs = null;
        try{
            if(stm != null)
                stm.close();
            stm = con.createStatement();
            String requete = "call Task_Search_With_IDMembre('"+ID_Membre+"')";
            rs = stm.executeQuery(requete);
        }catch(SQLException e){
            JOptionPane.showMessageDialog(null," Erreur acces à la table Membre \nVerifier si les informations sont correcte!");
        }
        return rs;
    }

    public ResultSet Search_Status_Task(String StatusT){
        ResultSet rs = null;
        try{
            if(stm != null)
                stm.close();
            stm = con.createStatement();
            String requete = "call Task_Search('"+StatusT+"')";
            rs = stm.executeQuery(requete);
        }
        catch(SQLException e){
            JOptionPane.showMessageDialog(null," Erreur acces à la table Tache \nVerifier si les informations sont correcte!");
        }
        return rs;
    }

    public void InsererMembre(int ID_Membre,String Nom){
        try{
            if(stm != null)
                stm.close();
            stm = con.createStatement();
            String requete = "call Member_Insert('"+ID_Membre+"','"+Nom+"')";
            stm.executeUpdate(requete);
        }
        catch(SQLException e){
            JOptionPane.showMessageDialog(null," Erreur insertion données dans Membre "
                + "\nVerifier si les informations sont correcte ou si l'ID existe!");
        }
    }
}
```

```

public void InsérerTache(int ID_Tache,String Nom,String Description,String StatusT){
    try{
        if(stm != null)
            stm.close();
        stm = con.createStatement();
        String requete = "call Task_Insert('"+ID_Tache+"','"+Nom+"','"+Description+"','"+StatusT+"')";
        stm.executeUpdate(requete);
    }catch(SQLException e){
        JOptionPane.showMessageDialog(null," Erreur insertion données dans Tache "
            + "\nVérifier si les informations sont correcte ou si l'ID existe!");
    }
}

public void InsérerAssignment(int ID_Membre,int ID_Tache){
    try{
        if(stm != null)
            stm.close();
        stm = con.createStatement();
        String requete = "call Assignment_Insert('"+ID_Membre+"','"+ID_Tache+"')";
        stm.executeUpdate(requete);
    }catch(SQLException e){
        JOptionPane.showMessageDialog(null," Erreur d'assigner une tache à un membre "
            + "\nVérifier si l'ID du membre et l'ID de la tache existe!");
    }
}

public ResultSet Delete_Membre(int ID_Membre){
    ResultSet rs = null;
    try{
        if(stm != null)
            stm.close();
        stm = con.createStatement();
        String requete = "call Member_Delete ('"+ID_Membre+"')";
        rs = stm.executeQuery(requete);
    }catch(SQLException e){
        JOptionPane.showMessageDialog(null," Erreur suppression de Membre dans la base de données \nVérifier si l'ID du membre existe!");
    }
    return rs;
}

public ResultSet Delete_Tache(int ID_Tache){
    ResultSet rs = null;
    try{
        if(stm != null)
            stm.close();
        stm = con.createStatement();
        String requete = "call Task_Delete ('"+ID_Tache+"')";
        rs = stm.executeQuery(requete);
    }catch(SQLException e){
        JOptionPane.showMessageDialog(null," Erreur suppression de Tache dans la base de données \nVérifier si l'ID de la tache existe!");
    }
    return rs;
}

public void ModifierMembre(int ID_Membre,String Nom){
    try{
        if(stm != null)
            stm.close();
        stm = con.createStatement();
        String requete = "call Member_Modify('"+ID_Membre+"','"+Nom+"')";
        stm.executeUpdate(requete);
    }catch(SQLException e){
        JOptionPane.showMessageDialog(null," Erreur modification données dans Membre "
            + "\nVérifier si les informations sont correcte ou si l'ID existe!");
    }
}

public void ModifierTache(int ID_Tache,String Nom,String Description,String StatusT){
    try{
        if(stm != null)
            stm.close();
        stm = con.createStatement();
        String requete = "call Task_Modify('"+ID_Tache+"','"+Nom+"','"+Description+"','"+StatusT+"')";
        stm.executeUpdate(requete);
    }catch(SQLException e){
        JOptionPane.showMessageDialog(null," Erreur modification données dans Tache "
            + "\nVérifier si les informations sont correcte ou si l'ID existe!");
    }
}
}

```

```

public ResultSet ListerMembre(){
    ResultSet rs = null;
    try{
        if(stm != null)
            stm.close();
        stm = con.createStatement();
        String requete = "call Member_List ()";
        rs = stm.executeQuery(requete);
    }catch(SQLException e){
        JOptionPane.showMessageDialog(null," Erreur affiche des membres!");
    }
    return rs;
}

public ResultSet ListerTache(){
    ResultSet rs = null;
    try{
        if(stm != null)
            stm.close();
        stm = con.createStatement();
        String requete = "call Task_List ()";
        rs = stm.executeQuery(requete);
    }catch(SQLException e){
        JOptionPane.showMessageDialog(null," Erreur affiche des taches!");
    }
    return rs;
}

public ResultSet Lister(){
    ResultSet rs = null;
    try{
        if(stm != null)
            stm.close();
        stm = con.createStatement();
        String requete = "call Task_List ()"; //fait appelle au procedure Task_List() dans la base de donnee
        rs = stm.executeQuery(requete);
    }catch(SQLException e){
        JOptionPane.showMessageDialog(null," Erreur affiche des taches!");
    }
    return rs;
}

Connection obtenirconnexion(){
    return con;
}
}

```