



Génie Logiciel Avance

Rapport TP1

* **Sujet du TP**

Gestion de taches en Java

* **Étudiant**

Pierre Rubens MILORME

* **Professeur**

Ho Tuong Vinh

Aout 2017

1 - INTRODUCTION

Ce présent document constitue un rapport accompagnant les codes sources, élaborées dans le cadre du cours de *Génie logiciel* et du travail pratique numéro 1. Il s'agit d'une introduction, d'une mise à niveau et d'un rappel au sujet des concepts tels que la modélisation *UML* ou encore la programmation orientée objet. Il prône et exige l'utilisation d'un IDE (environnement de développement intégré tel que Eclipse ou Netbeans), l'utilisation de tests unitaires par le biais de la bibliothèque *JUnit* ou encore l'utilisation d'un système de contrôle/gestion de versions à l'instar de *Git*. Le programme informatique développé a pour objectif final de permettre à l'utilisateur de gérer des tâches et de les affecter à des membres. L'avantage principal étant de lui fournir un outil efficace et rapide pour la gestion de ses tâches.

1.1 SPÉCIFICATIONS DE L'APPLICATION

Le programme que nous avons construit tourne autour de deux entités que sont les tâches proprement dites et les membres destinés à être assignés à ces tâches. Une tâche possède des caractéristiques telles qu'un identifiant, un nom, une description, et un statut alors qu'un membre ne possède que l'identifiant et le nom.

Entités	Attributs
Tâche	- ID - Nom - Description - Statut
Membre	- ID - Nom

Figure 1 : Tableau décrivant les attributs des tâches et des membres

Au-delà de la structure et de la composition des entités à manipuler dont nous venons tout juste de présenter plus haut, le programme désiré doit proposer un certain nombre de fonctionnalités comme présentées dans la liste présentée ci-dessous :

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 statut (avec le nom du assigné)

2 Exigences Fonctionnelles et Non-Fonctionnelles de l'Application

2. 1 Exigences fonctionnelles

Par définition, les Exigences Fonctionnelles (EF) décrivent ce que le système doit pouvoir faire en terme d'actions et d'attentes. Dans le cadre de notre application, elles sont décrites comme suit :

Exigences Fonctionnelles	Descriptions
EF1	Créer une tâche
EF2	Modifier une tâche
EF3	Supprimer une tâche
EF4	Ajouter une tâche
EF5	Créer un membre
EF6	Modifier un membre
EF7	Supprimer un membre
EF8	Ajouter un membre
EF9	Assigner une tâche a un membre
EF10	Chercher et afficher tous les tâches assignées à un membre (par son ID)
EF11	Chercher et afficher tous les tâches en fonction de leur statut (avec le nom du assigné)

Figure 2 : Fonctionnalites du systeme

a. Diagramme des cas d'utilisation

Les diagrammes de cas d'utilisation représentent généralement toutes les interactions des utilisateurs avec le système.

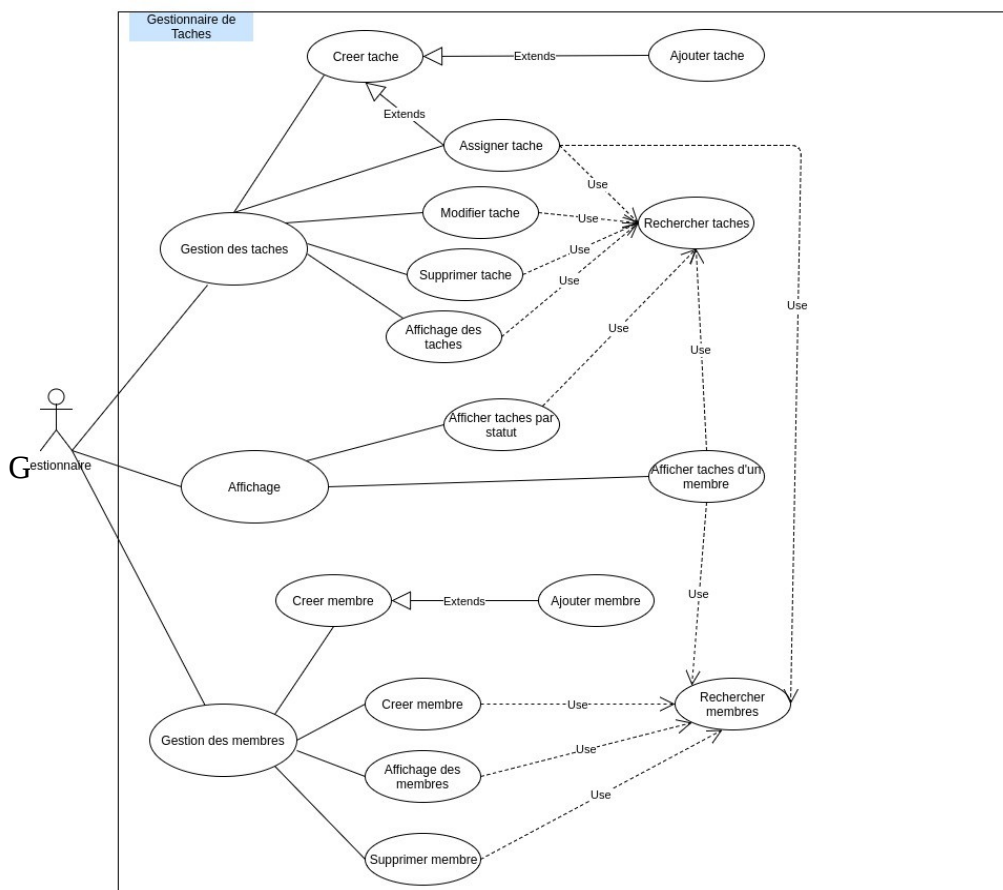


Figure 3 : Diagramme des cas d'utilisation du système

2.2 Exigences non-fonctionnelles (ENF)

Description	Qualité / Attribut
Le temps de réponse du système doit être essentiellement court, maximum 2 secondes pour exécuter la requête de l'utilisateur	Performance
Notre application doit présenter des résultats précis et juste tels que attendus.. Elle doit être capable d'effectuer la bonne opération lorsque demandée.	Exactitude ,Pertinence
En cas d'erreurs venant de l'utilisateur, les informations qui ont été déjà enregistrées ne doivent subir aucun changement.	Disponibilité
Les fonctions de l'application doivent être facilement comprises et interprétés par l'utilisateur.	Compréhension
Le code source de l'application est bien documenté pour la facilité d'entretien et de mise à jour du système à l'avenir.	Faciliter la Maintenance

En plus des exigences non-fonctionnelles sus-mentionnées, nous devons également respecter les contraintes de programmation suivantes :

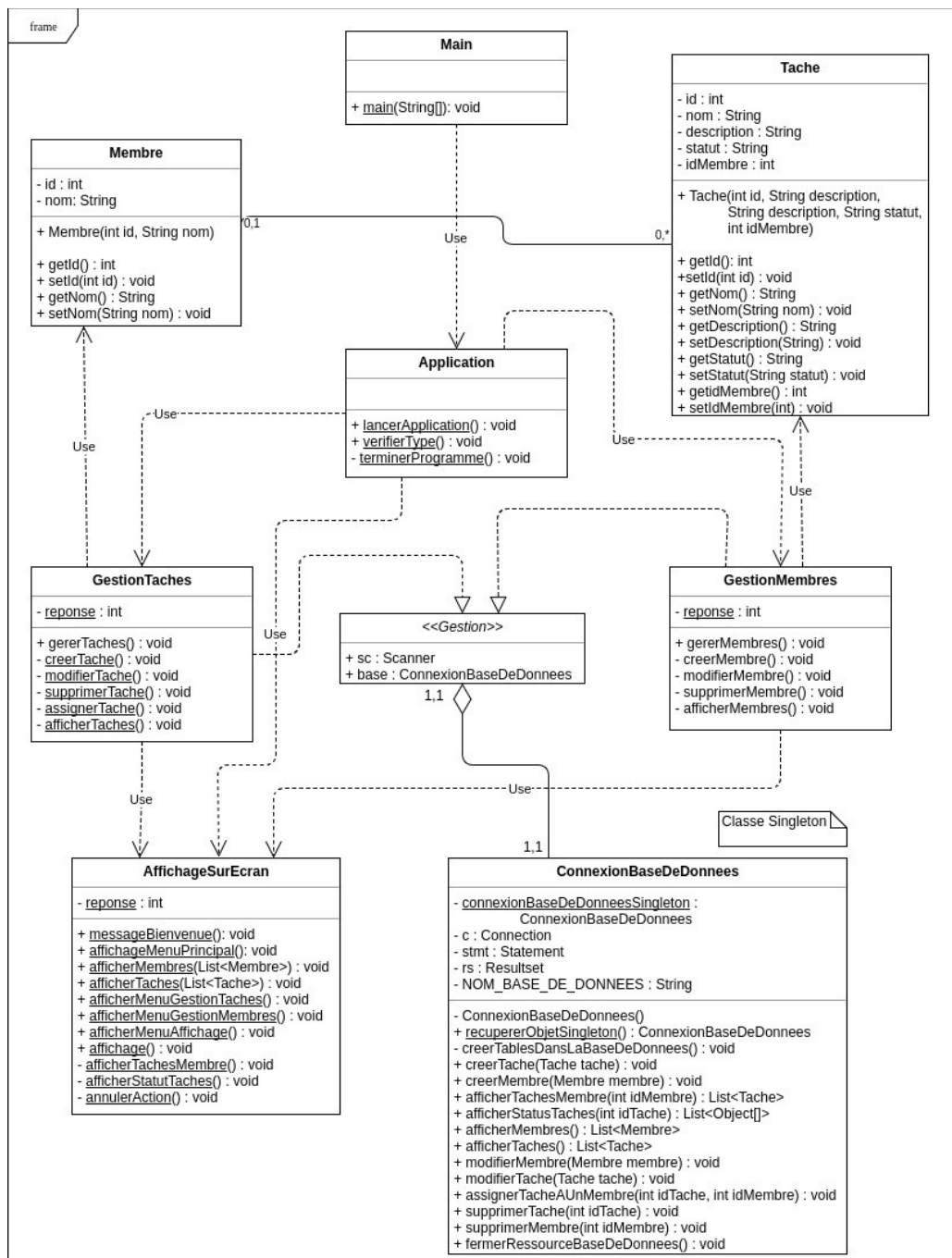
- Programmer de façon orientée objet
- Assurer que la communication entre l'utilisateur et le système est conviviale.
- Respect des consignes de codage

3. Conception de l'Application

3.1 Diagramme de classes

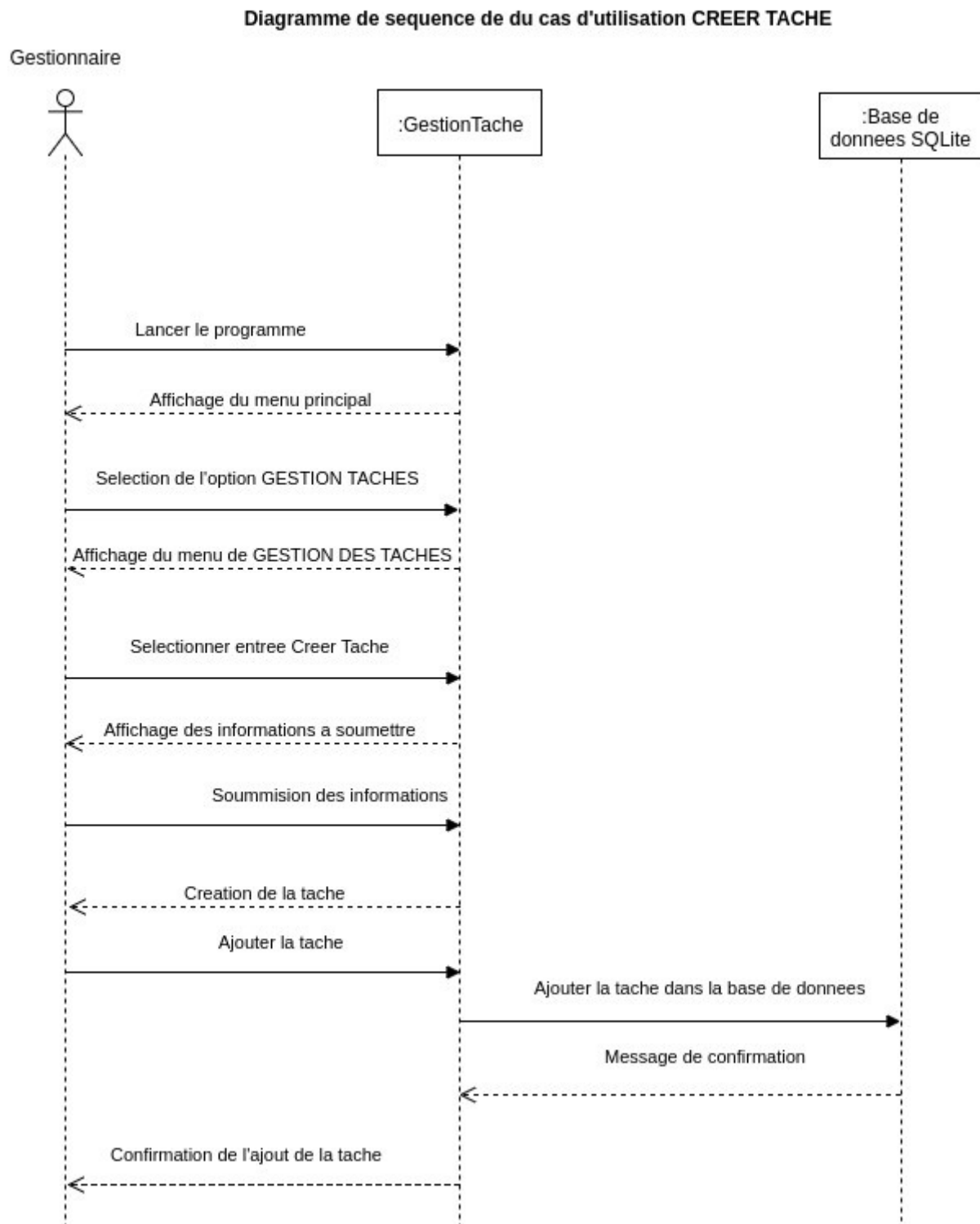
Le diagramme de classe représente les classes constituant le système et les associations entre elles. Elle exprime de manière générale la structure statique du système, en termes de classes et de relations entre ces classes de notre application.

Voici une illustration de notre diagramme de classe ci-dessous



3.2 Diagramme de Séquences

Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique. Ci-dessous une illustration du diagramme de séquences pour le cas d'utilisation <<Créer Tache>>.



4. IMPLÉMENTATION ET TEST

4.0 Environnement Matériel

Pour le développement de notre application de gestion de tâches nous avons fait usage d'un ordinateur de marque DELL modèle INSPIRON-4010 doté d'un processeur Intel ayant une fréquence de calcul 2.3 GHz, une capacité de 4 Go de mémoire vive et une mémoire de masse de 500 Go.

4.1 Environnement Logiciel

Notre application a été développée en utilisant des technologies et plateformes open source dont le système d'exploitation *Linux*, distribution *Ubuntu*, version 16.04 et l'environnement de développement intégré *Eclipse Neon*.

4.2 Langage de programmation

Le langage de programmation *Java* nous a été imposé pour la partie implémentation du projet. Nous avons en ce sens fait usage de la plateforme standard *Java SE* pour réaliser un programme en mode console en utilisant des bibliothèques tierces soit pour interfacer avec notre solution de stockage soit pour la mise en place des tests.

4.3 La sauvegarde des données

La partie persistance des données est assurée par la base de données *SQLite* qui est une solution de base de données très souple simple et surtout portable pour laquelle nous avons utilisé le pilote *jdbc* correspondant.

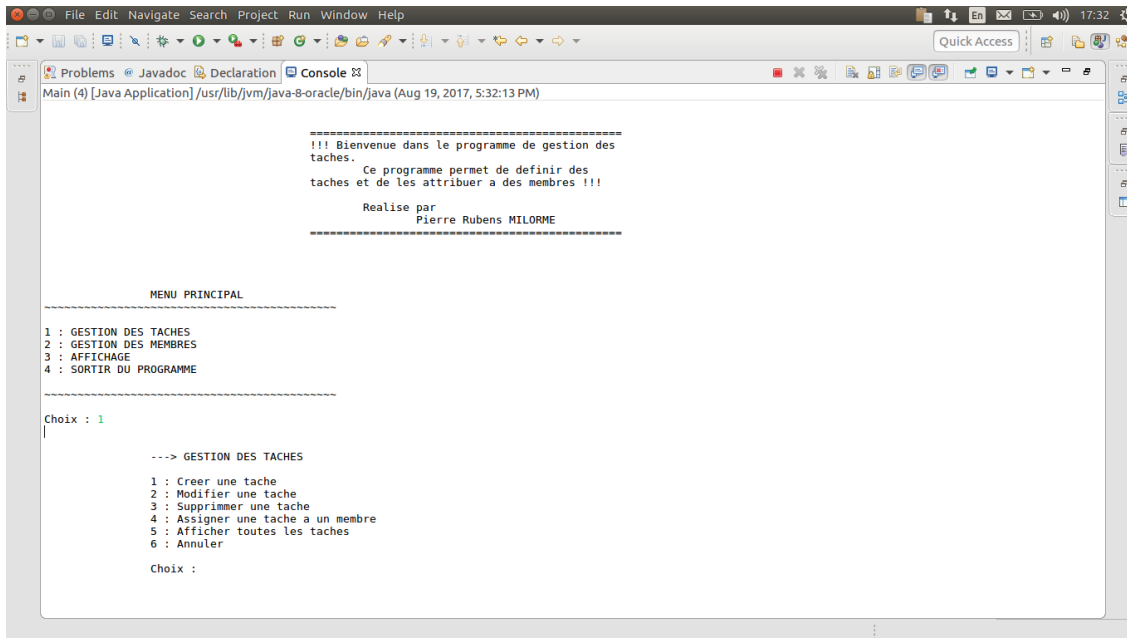
4.4 Méthode de test

Afin de tester au fur et à mesure les composants unitaires de notre application nous avons utilisé la solution *JUnit* pour faire le test unitaire et s'assurer du bon fonctionnement des méthodes utilisées, les variables et les classes utilisées.

5. TEST D'ACCEPTATION

Pour le test d'acceptation, nous avons choisi de présenter les cas suivants : création de taches et affichage des taches d'un membre en utilisant son ID.

5.1 Créer Tache



```
=====
!!! Bienvenue dans le programme de gestion des
taches.
Ce programme permet de definir des
taches et de les attribuer a des membres !!!

Realise par
Pierre Rubens MILORME
=====

MENU PRINCIPAL
-----
1 : GESTION DES TACHES
2 : GESTION DES MEMBRES
3 : AFFICHAGE
4 : SORTIR DU PROGRAMME
-----

Choix : 1

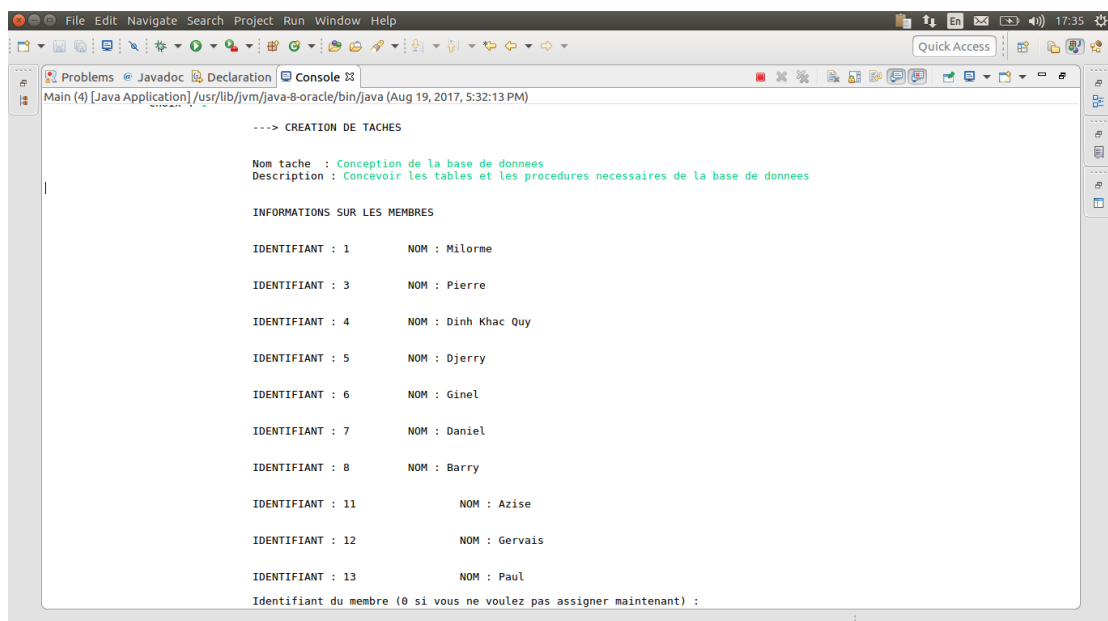
---> GESTION DES TACHES

1 : Creer une tache
2 : Modifier une tache
3 : Supprimer une tache
4 : Assigner une tache a un membre
5 : Afficher toutes les taches
6 : Annuler

Choix :
```

La capture ci-dessus présente le menu de gestion des taches après avoir fait le choix dans le menu principal.

Les deux captures ci-dessous présente les écrans où l'utilisateur rentre les informations nécessaires pour la création de la tache et l'ajout de cette tache.



```
---> CREATION DE TACHES

Nom tache : Conception de la base de donnees
Description : Concevoir les tables et les procedures necessaires de la base de donnees

INFORMATIONS SUR LES MEMBRES

IDENTIFIANT : 1      NOM : Milorme
IDENTIFIANT : 3      NOM : Pierre
IDENTIFIANT : 4      NOM : Dinh Khac Quy
IDENTIFIANT : 5      NOM : Djerry
IDENTIFIANT : 6      NOM : Ginel
IDENTIFIANT : 7      NOM : Daniel
IDENTIFIANT : 8      NOM : Barry
IDENTIFIANT : 11     NOM : Azise
IDENTIFIANT : 12     NOM : Gervais
IDENTIFIANT : 13     NOM : Paul
Identifiant du membre (0 si vous ne voulez pas assigner maintenant) :
```



```
File Edit Navigate Search Project Run Window Help
Main (4) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Aug 19, 2017, 5:32:13 PM)

IDENTIFIANT : 1      NOM : Milorme
IDENTIFIANT : 3      NOM : Pierre
IDENTIFIANT : 4      NOM : Dinh Khac Quy
IDENTIFIANT : 5      NOM : Djerry
IDENTIFIANT : 6      NOM : Ginel
IDENTIFIANT : 7      NOM : Daniel
IDENTIFIANT : 8      NOM : Barry

IDENTIFIANT : 11     NOM : Azise
IDENTIFIANT : 12     NOM : Gervais
IDENTIFIANT : 13     NOM : Paul

Identifiant du membre (0 si vous ne voulez pas assigner maintenant) : 1
Ajouter la tache creee ?
1 : OUI 2 : NON
Reponse : 1

Tache enregistree

Voulez-vous creer une autre tache ?
1 : OUI 2 : NON
Choix :
```

Affichage de la tache nouvellement créée :

```
File Edit Navigate Search Project Run Window Help
Main (4) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Aug 19, 2017, 5:32:13 PM)

ID : 17
NOM : Test d integration
DESCRIPTION : fsdgfd
STATUT : nouveau
MEMBRE ASSOCIE : 0

ID : 18
NOM : Analyse
DESCRIPTION : sdfds
STATUT : nouveau
MEMBRE ASSOCIE : 0

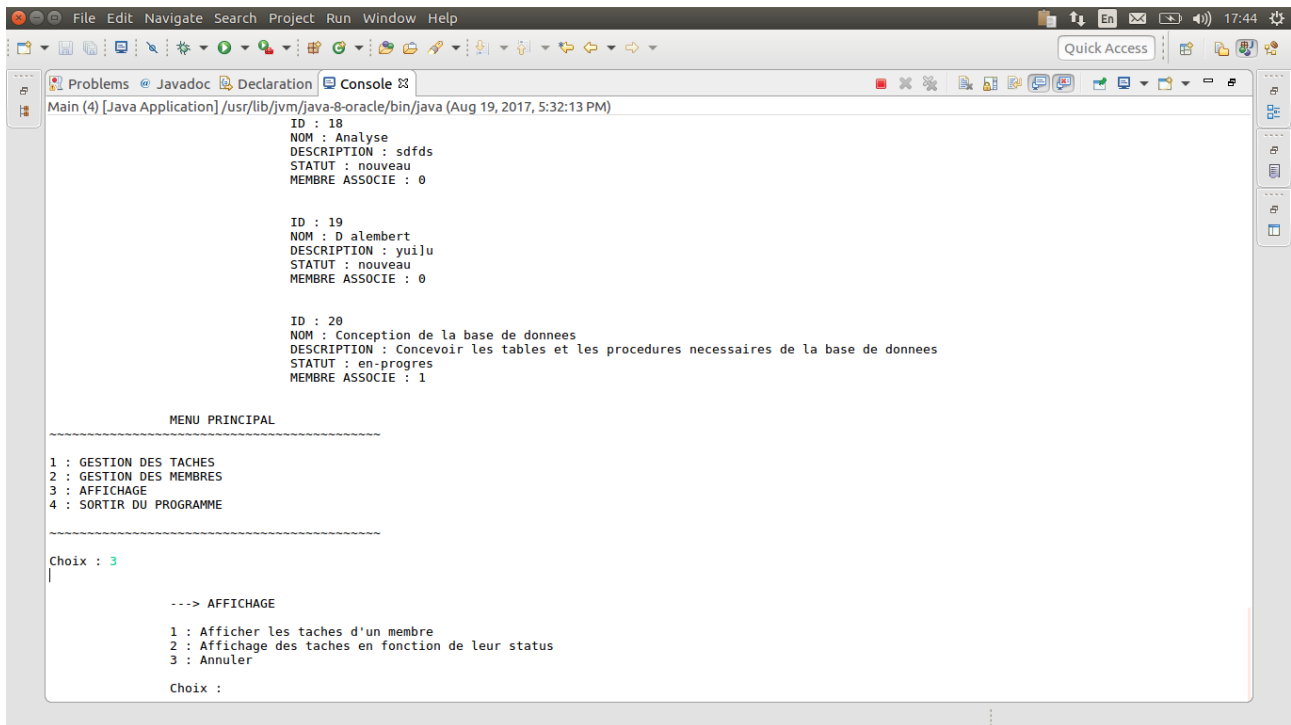
ID : 19
NOM : D alembert
DESCRIPTION : yuillu
STATUT : nouveau
MEMBRE ASSOCIE : 0

ID : 20
NOM : Conception de la base de donnees
DESCRIPTION : Concevoir les tables et les procedures necessaires de la base de donnees
STATUT : en-progres
MEMBRE ASSOCIE : 1

-----
MENU PRINCIPAL
-----
1 : GESTION DES TACHES
2 : GESTION DES MEMBRES
3 : AFFICHAGE
4 : SORTIR DU PROGRAMME
-----
Choix :
```

5.2 Affichage des taches d'un membre en utilisant son ID

La capture ci-après présente le menu principal et le menu de l'option Affichage permettant d'afficher les taches pour un membre.



```
File Edit Navigate Search Project Run Window Help
Main (4) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Aug 19, 2017, 5:32:13 PM)
ID : 18
NOM : Analyse
DESCRIPTION : sdfds
STATUT : nouveau
MEMBRE ASSOCIE : 0

ID : 19
NOM : D alembert
DESCRIPTION : yuifu
STATUT : nouveau
MEMBRE ASSOCIE : 0

ID : 20
NOM : Conception de la base de donnees
DESCRIPTION : Concevoir les tables et les procedures necessaires de la base de donnees
STATUT : en-progres
MEMBRE ASSOCIE : 1

MENU PRINCIPAL
=====
1 : GESTION DES TACHES
2 : GESTION DES MEMBRES
3 : AFFICHAGE
4 : SORTIR DU PROGRAMME
=====

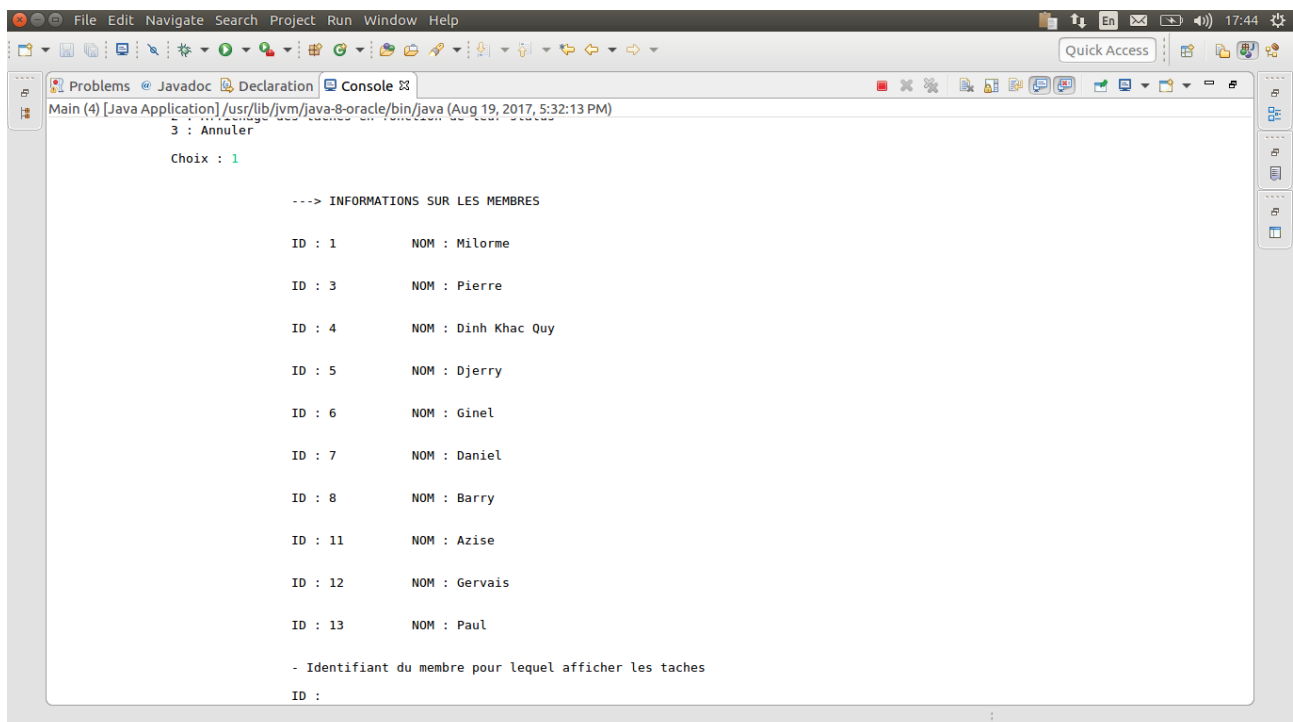
Choix : 3
|

----> AFFICHAGE

1 : Afficher les taches d'un membre
2 : Affichage des taches en fonction de leur status
3 : Annuler

Choix :
```

L'utilisateur doit faire un choix parmi tous les membres de la base de données qui lui sont présentés.



```
File Edit Navigate Search Project Run Window Help
Main (4) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Aug 19, 2017, 5:32:13 PM)
3 : Annuler

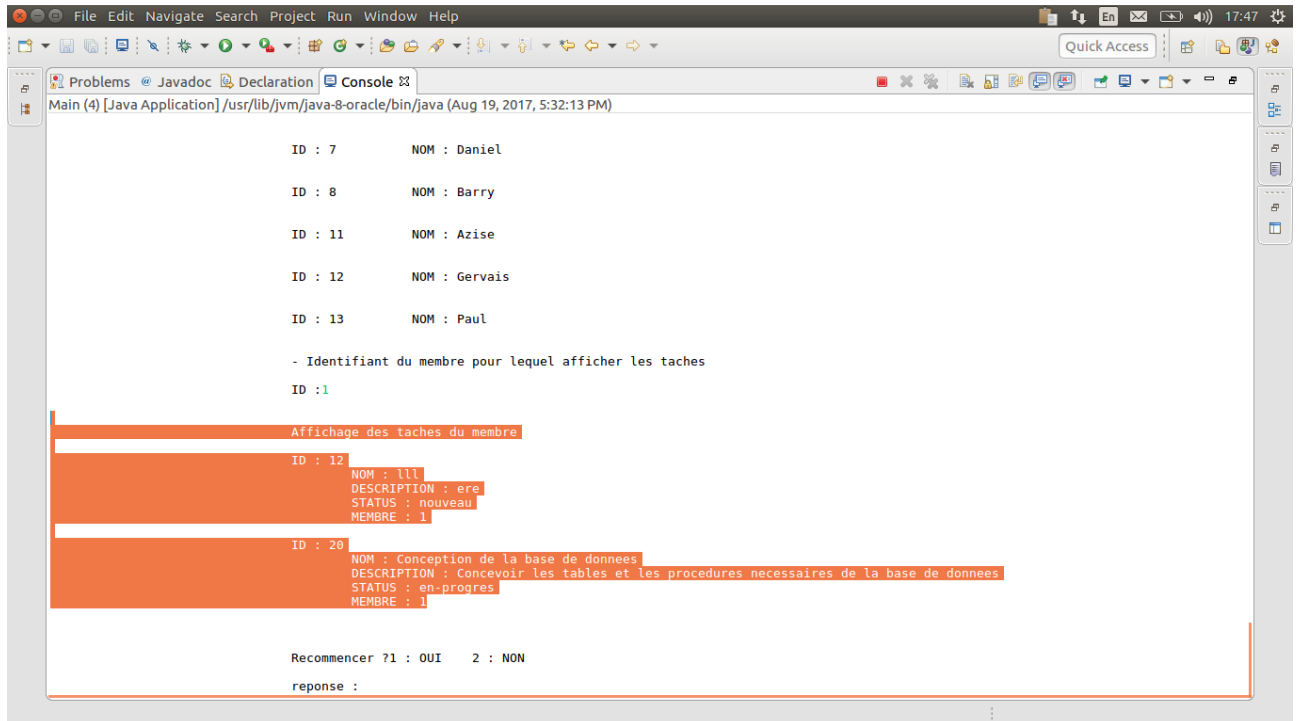
Choix : 1

----> INFORMATIONS SUR LES MEMBRES

ID : 1      NOM : Milorme
ID : 3      NOM : Pierre
ID : 4      NOM : Dinh Khac Quy
ID : 5      NOM : Djerry
ID : 6      NOM : Ginel
ID : 7      NOM : Daniel
ID : 8      NOM : Barry
ID : 11     NOM : Azise
ID : 12     NOM : Gervais
ID : 13     NOM : Paul

- Identifiant du membre pour lequel afficher les taches
ID :
```

Affichage des taches pour le membre dont l'ID est 1, c'est-a-dire dont le nom est Milorme.



The screenshot shows a Java IDE window with a console output. The console displays a list of members with their IDs and names. Below this, it prompts for a member ID to display tasks. The user enters '1'. The program then displays tasks for member 1, which are highlighted in orange. The tasks are:

- ID : 12, NOM : lll, DESCRIPTION : ere, STATUS : nouveau, MEMBRE : 1
- ID : 20, NOM : Conception de la base de donnees, DESCRIPTION : Concevoir les tables et les procedures necessaires de la base de donnees, STATUS : en-progres, MEMBRE : 1

At the bottom, there is a prompt 'Recommencer ? 1 : OUI 2 : NON' and a 'reponse :' label.

```
File Edit Navigate Search Project Run Window Help
Main (4) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Aug 19, 2017, 5:32:13 PM)

ID : 7      NOM : Daniel
ID : 8      NOM : Barry
ID : 11     NOM : Azise
ID : 12     NOM : Gervais
ID : 13     NOM : Paul

- Identifiant du membre pour lequel afficher les taches
ID : 1

Affichage des taches du membre
ID : 12     NOM : lll
            DESCRIPTION : ere
            STATUS : nouveau
            MEMBRE : 1

ID : 20     NOM : Conception de la base de donnees
            DESCRIPTION : Concevoir les tables et les procedures necessaires de la base de donnees
            STATUS : en-progres
            MEMBRE : 1

Recommencer ? 1 : OUI 2 : NON
reponse :
```

6. CONCLUSION

Dans ce rapport nous avons présenté les travaux que nous avons eu à faire afin de réaliser le programme de gestion de tâches. Nous avons décrit le programme en termes de fonctionnalités et des exigences non-fonctionnelles, illustré les interactions entre l'utilisateur et le système en utilisant le diagramme des cas d'utilisation et présenté le déroulement de séquentiel et chronologique des actions pour un cas d'utilisation choisi. La réalisation d'un tel TP nous a servi de rappel sur les concepts de la modélisation avec UML et programmation orientée-objet avec Java. Certes, il serait important d'effectuer d'autres itérations sur le projet en adoptant par exemple une approche agile ou itérative et incrémentale afin de détecter les éventuelles bugs et erreurs de conceptions ou pour améliorer et ajouter des paramètres et d'autres fonctionnalités potentiellement utiles.

Toutefois, l'apport de ce travail a été d'une importance considérable : en effet, il nous a permis de suivre une méthodologie de travail bien étudiée, d'approfondir des connaissances sur les méthodes de développement des applications.

7- CODES SOURCES

Pour consulter les codes du TP, référez a la partie Annexe de ce document ,ou rendez vous sur le lien suivant : <https://github.com/senburbens/gestionTache.git>

Exécution du programme

- 1) Récupérer le projet depuis le repository correspondant dont le lien est <https://github.com/senburbens/gestionTache.git>.
- 2) Ouvrir le projet avec **Eclipse IDE**.
- 3) Cliquer droit sur la classe principale du projet **Main.java** se trouver dans le package **com.gestiontache.milorme.app**, se rendre dans l'entrée **Run as**, sélectionner **Java Application**.
- 4) Dans le cas ou l'application renvoie des erreurs causées par un mauvais référencement des bibliothèques jar pour interfacier avec la base de données ou pour l'utilisation de **JUnit**, rajouter celles-ci dans le **CLASSPATH** du projet en cliquant sur celui-ci et sélectionner **Properties**, puis **Build Path**, sélectionner **Librairies**, enlever les jar mal références, cliquer sur **Add External jars**, chercher les dans le répertoire racine du projet, valider.
- 5) Refaire les manipulations 1, 2 et 3.

CODES SOURCES

Classe Main.java

```
/
*****
*****
* Module : Genie Logiciel Avance
* TP1 : Programme de Gestion de Taches
* Auteur : Pierre Rubens MILORME
* Professeur : Ho Thuong Vinh
* Fait en aout 2017
*
*****
*****/

package com.gestiontache.milorme.app;

public class Main{
    public static void main(String[] args){
        Application.lancerApplication();
    }
}
```

Classe Application.java

```
package com.gestiontache.milorme.app;

import java.io.IOException;
import java.sql.SQLException;
import java.util.Hashtable;
import java.util.List;
import java.util.Scanner;
import com.gestiontache.milorme.entites.Tache;
import com.gestiontache.milorme.manager.GestionMembres;
import com.gestiontache.milorme.manager.GestionTaches;
import com.gestiontache.milorme.affichage.AffichageSurEcran;
import com.gestiontache.milorme.manager.Gestion;
import com.gestiontache.milorme.basededonnees.ConnexionBaseDeDonnees;
import com.gestiontache.milorme.entites.Membre;

public class Application implements Gestion{

    //Methode principale de l'application
    public static void lancerApplication() {
        int reponse=0;

        AffichageSurEcran.messageBienvenue();//Appel de la methode
        messagBienvenue

        while(true){

            AffichageSurEcran.affichageMenuPrincipal(); //Methode
            affichage du menu principale
            reponse=0;
        }
    }
}
```

```

        while(reponse<1 || reponse>4){
            verifierType();
            reponse = sc.nextInt();
            if(reponse<1 || reponse>4)
                System.out.println("Entrez 1, 2, 3, ou
4\nChoix :");
        }

        switch(reponse){
            //GESTION DES TACHES
            case 1 :
                GestionTaches.gererTaches();
                break;
            //GESTION DES MEMBRES
            case 2 :
                GestionMembres.gererMembres();
                break;
            //AFFICHAGE
            case 3 :
                AffichageSurEcran.affichage();
                break;
            //SORTIR DU PROGRAMME
            case 4 :
                terminerProgramme();
                break;
        }
    }

    //Fonction de verification de type
    public static void verifierType(){
        if(!sc.hasNextInt()){
            System.out.print("Veuillez entrer un entier !!!\nChoix :");
            sc.nextLine();
            verifierType();
        }
    }

    //Mettre fin au programme
    private static void terminerProgramme(){
        System.out.println("\n\nAU REVOIR ET A BIENTOT !!!");
        base.fermerRessourceBaseDeDonnees();
        System.exit(0);
    }
}

```

Classe AffichageSurEcran.java

```

package com.gestiontache.milorme.affichage;

import java.io.IOException;
import java.sql.SQLException;
import java.util.List;
import com.gestiontache.milorme.entites.Tache;

```

[illegible]


```

        System.out.print("\n\t\t\t\tMEMBRE ASSOCIE :
"+t.getIdMembre());
        System.out.println();
    }
}

public static void afficherMenuGestionTaches(){
    System.out.println("\n\n\t\t---> GESTION DES TACHES\n");
    System.out.print("\t\t1 : Creer une tache\n");
    System.out.print("\t\t2 : Modifier une tache\n");
    System.out.print("\t\t3 : Supprimer une tache\n");
    System.out.print("\t\t4 : Assigner une tache a un membre\n");
    System.out.print("\t\t5 : Afficher toutes les taches\n");
    System.out.print("\t\t6 : Annuler\n");
    System.out.print("\n\t\tChoix : ");
}

public static void afficherMenuGestionMembres(){
    System.out.println("\n\n\t\t---> GESTION DES MEMBRES\n");
    System.out.print("\t\t1 : Creer un membre\n");
    System.out.print("\t\t2 : Modifier un membre\n");
    System.out.print("\t\t3 : Supprimer un membre\n");
    System.out.print("\t\t4 : Afficher tous les membres\n");
    System.out.print("\t\t5 : Annuler\n");
    System.out.print("\n\t\tChoix : ");
}

public static void afficherMenuAffichage(){
    System.out.println("\n\n\t\t---> AFFICHAGE\n");
    System.out.print("\t\t1 : Afficher les taches d'un membre\n");
    System.out.print("\t\t2 : Affichage des taches en fonction de leur
status\n");
    System.out.print("\t\t3 : Annuler\n");
    System.out.print("\n\t\tChoix : ");
}

//Affichage de toutes les taches d'un membre
//Affichage des taches en fonctions de leurs statuts
public static void affichage(){

    AffichageSurEcran.afficherMenuAffichage();
    reponse=0;

    while(reponse<1 || reponse>3){
        Application.verifierType();
        reponse = sc.nextInt();
        if(reponse<1 || reponse>3)
            System.out.println("Entrez 1, 2 ou 3\n");
    }
    switch(reponse){
        case 1 :
            afficherTachesMembre();
            break;
        case 2 :
            afficherStatutTaches();
            break;
        case 3 :
            annulerAction();
            break;
    }
}

```

```

private static void afficherTachesMembre(){

    //Affichage des informations sur les membres
    reponse=1;
    while(reponse==1){
        System.out.println("\n\n\t\t\t\t\t--> INFORMATIONS SUR LES
MEMBRES\n");
        try {
            List<Membre> listeMembre=base.afficherMembres();
            for(Membre m : listeMembre){
                System.out.println("\n\t\t\t\t\tID : " + m.getId()+
"\t\t\tNOM : " + m.getNom());
                System.out.println();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        System.out.print("\n\t\t\t\t\t- Identifiant du membre pour
lequel afficher les taches\n\n\t\t\t\t\tID :");
        Application.verifierType();
        reponse=sc.nextInt();
        //Affichage des taches du membre
        System.out.println("\n\n\t\t\t\t\tAffichage des taches du
membre\n");
        try {
            List<Tache>
listeTaches=base.afficherTachesMembre(reponse);
            //System.out.print("\t\t\t\t\tID\t\t\tNOM\t\t\tDESCRIPTION\t\t\t
STATUS\t\t\tMEMBRE\n\n");
            for(Tache m : listeTaches){
                System.out.println("\t\t\t\t\tID : " + m.getId());
                System.out.println("\t\t\t\t\tNOM : " +
m.getNom());
                System.out.println("\t\t\t\t\tDESCRIPTION : " +
m.getDescription());
                System.out.println("\t\t\t\t\tSTATUS : " +
m.getStatus());
                System.out.println("\t\t\t\t\tMEMBRE : " +
m.getIdMembre());
                System.out.println();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        System.out.print("\n\n\t\t\t\t\tRecommencer ?1 : OUI\t2 :
NON\n\n\t\t\t\t\treponse : ");
        reponse=0;
        while(reponse<1 || reponse >2){
            Application.verifierType();
            reponse=sc.nextInt();
        }
    }
}

private static void afficherStatutTaches(){
    reponse=1;
    while(reponse==1){
        System.out.print("\t\t\t\t\tVeuillez entrer le status des taches
a afficher\n\t\t\t\t\t1 : nouveau\t2 : en_progres\t3 : termine\n\t\t\t\t\tChoix :
");
    }
}

```

```

        reponse=0;
        while(reponse<1 || reponse>3){
            Application.verifierType();
            reponse = sc.nextInt();
            if(reponse<1 || reponse>3)
                System.out.println("\t\t\t\t\tEntrez 1, 2 ou 3\n");
        }
        try {
            List<Object[]>
statusTaches=base.afficherStatusTaches(reponse);
            Tache t;
            Membre m;

            System.out.println("\n\t\t\t\t\tListe des taches\n");
            for(Object[] obj : statusTaches){
                t= (Tache) obj[0];
                m= (Membre) obj[1];

                System.out.println("\t\t\t\t\tID: " + t.getId());
                System.out.println("\t\t\t\t\tNOM TACHE: " +
t.getNom());
                System.out.println("\t\t\t\t\tDESCRIPTION TACHE: "
+ t.getDescription());
                System.out.println("\t\t\t\t\tSTATUS TACHE: " +
t.getStatus());
                System.out.println("\t\t\t\t\tID MEMBRE: " +
m.getId());
                System.out.println("\t\t\t\t\tNOM MEMBRE: " +
m.getNom());
                System.out.println();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        System.out.print("\n\n\t\t\t\t\tRecommencer ?1 : OUI\t2 :
NON\n\n\t\t\t\t\treponse : ");
        reponse=0;
        while(reponse<1 || reponse >2){
            Application.verifierType();
            reponse=sc.nextInt();
        }
    }

    private static void annulerAction(){
        try {
            Runtime.getRuntime().exec("clear");
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.println("\n\t\t\t\t\tAction Annulee\n");
    }
}

```

Classe Gestion.java

```

package com.gestiontache.milorme.manager;

import java.util.Scanner;

```

```

import com.gestiontache.milorme.basededonnees.ConnexionBaseDeDonnees;

public interface Gestion {
    public Scanner sc = new Scanner(System.in);
    public ConnexionBaseDeDonnees base =
ConnexionBaseDeDonnees.recupererObjetSingleton();
}

```

Classe GestionTaches.java

```

package com.gestiontache.milorme.manager;

```

```

import java.sql.SQLException;
import java.util.List;
import com.gestiontache.milorme.entites.Tache;
import com.gestiontache.milorme.affichage.AffichageSurEcran;
import com.gestiontache.milorme.entites.Membre;
import com.gestiontache.milorme.app.Application;

public class GestionTaches implements Gestion{

    private static int reponse;

    public static void gererTaches(){

        AffichageSurEcran.afficherMenuGestionTaches();
        reponse=0;

        while(reponse<1 || reponse>6){
            Application.verifierType();
            reponse = sc.nextInt();
            if(reponse<1 || reponse>6)
                System.out.println("Entrez un entier entre 1 et 5
inclus\n");
        }

        switch(reponse){
            case 1 :
                creerTache();
                break;
            case 2 :
                try {
                    modifierTache();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
                break;
            case 3 :
                supprimerTache();
                break;
            case 4 :
                assignerTache();
                break;
            case 5 :
                afficherTaches();
                break;

```

```

        case 6 :
            System.out.println("\n0operation annulee\n");
            break;
    }
}

private static void creerTache(){
    reponse=1;
    System.out.println("\n\t\t\t\t\t---> CREATION DE TACHES\n");
    while(reponse==1){
        System.out.print("\n\t\t\t\t\tNom tache : ");
        sc.nextLine();
        String nomTache=sc.nextLine();
        System.out.print("\t\t\t\t\tDescription : ");
        String descriptionTache=sc.nextLine();
        System.out.println("\n\n\t\t\t\t\tINFORMATIONS SUR LES
MEMBRES\n");
        try {
            List<Membre> listeMembre=base.afficherMembres();
            for(Membre m : listeMembre){
                System.out.println("\n\t\t\t\t\tIDENTIFIANT : " +
m.getId()+ "\t\tNOM : " + m.getNom());
                System.out.println();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        System.out.print("\t\t\t\t\tIdentifiant du membre (0 si vous ne
voulez pas assigner maintenant) : ");
        Application.verifierType();
        int identifiant=sc.nextInt();
        System.out.print("\t\t\t\t\tAjouter la tache
creee ?\n\t\t\t\t\tt1 : OUI\t2 : NON\n\t\t\t\t\tReponse : ");
        reponse=0;
        while(reponse<1 || reponse>2){
            Application.verifierType();
            reponse=sc.nextInt();
            if(reponse<1 || reponse>2){
                System.out.println("\t\t\t\t\tEntrez 1 ou
2\n\t\t\t\t\tChoix : ");
            }
        }
        if(reponse==1){
            base.creerTache(new
Tache(0,nomTache,descriptionTache,"",identifiant));
        }

        System.out.print("\t\t\t\t\tVoulez-vous creer une autre
tache ?\n\t\t\t\t\tt1 : OUI\t2 : NON\n\t\t\t\t\tChoix : ");
        reponse=0;
        while(reponse<1 || reponse >2){
            Application.verifierType();
            reponse=sc.nextInt();
            if(reponse<1 || reponse>2){
                System.out.print("\t\t\t\t\tEntrez 1 ou
2\n\t\t\t\t\tChoix : ");
            }
        }
    }
}
}

```

```

private static void modifierTache() throws SQLException{
    reponse=1;
    List<Tache> listeTache;
    while(reponse==1){
        System.out.println("\n\n\t\t\t\tLISTE DES TACHES\n");
        try {
            listeTache=base.afficherTaches();
            for(Tache t : listeTache){
                System.out.println("\n\t\t\t\t\tIDENTIFIANT : " +
t.getId()+ "\n\t\t\t\t\tNOM : " + t.getNom() + "\n\t\t\t\t\tDESCRIPTION : "+
t.getDescription() + "\n\t\t\t\t\tSTATUS : " +
t.getStatus() + "\n\t\t\t\t\tIDENTIFIANT
MEMBRE : "+ t.getIdMembre());
                System.out.println();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        System.out.print("\n\t\t\t\t\tIdentifiant de la tache a
modifier\n\t\t\t\t\tID :");
        Application.verifierType();
        int identifiantTacheAModifier=sc.nextInt();

        System.out.println("\n\t\t\t\t\t---> Modification de la
tache\n");

        System.out.print("\t\t\t\t\tNom tache : ");
        sc.nextLine();
        String nomTacheAModifier=sc.nextLine();
        System.out.print("\t\t\t\t\tDescription : ");
        String descriptionTacheAModifier=sc.nextLine();
        //Verifier que l'utilisateur a entrer un entier
        System.out.print("\t\t\t\t\tIdentifiant du membre auquel
assigner la tache : ");
        AffichageSurEcran.afficherMembres(base.afficherMembres());
        System.out.print("\t\t\t\t\tChoix : ");
        Application.verifierType();
        int identifiantDuMembre=sc.nextInt();

        String statusAModifier="";
        sc.nextLine();
        while(!statusAModifier.equalsIgnoreCase("nouveau") && !
statusAModifier.equalsIgnoreCase("en-progres") && !
statusAModifier.equalsIgnoreCase("termine")){
            System.out.print("\t\t\t\t\tModification du statut :
(nouveau, en-progres ou termine) :\n\t\t\t\t\t");
            statusAModifier=sc.nextLine();
            if(!statusAModifier.equalsIgnoreCase("nouveau") && !
statusAModifier.equalsIgnoreCase("en-progres") && !
statusAModifier.equalsIgnoreCase("termine")){
                System.out.println("\t\t\t\t\tMauvaise saisie.
Entrez : nouveau ou en-progres ou termine");
            }
        }

        System.out.print("\t\t\t\t\tVoulez-vous proceder a la
modification ?\n\t\t\t\t\tt1 : OUI\t2 : NON\n\t\t\t\t\tChoix : ");
        reponse=0;
        while(reponse<1 || reponse>2){
            Application.verifierType();
            reponse=sc.nextInt();
            System.out.print("\t\t\t\t\tEntrez 1 ou 2\n\t\t\t\t\tChoix :
");

```

```

    }
    if(reponse==1){
        base.modifierTache(new
Tache(identifiantTacheAModifier,nomTacheAModifier,descriptionTacheAModifier,stat
usAModifier,identifiantDuMembre));
    }
    System.out.print("\t\t\t\tVoulez-vous modifier un autre membre
?\n\t\t\t\tt1 : OUI\t2 : NON\n\t\t\t\ttChoix : ");
    reponse=0;
    while(reponse<1 || reponse >2){
        Application.verifierType();
        reponse=sc.nextInt();
        System.out.print("\t\t\t\tEntrez 1 ou 2\nChoix : ");
    }
}

private static void supprimerTache(){
    reponse=1;
    List<Tache> listeTache;
    while(reponse==1){
        System.out.println("\n\n\t\t\t\t---> LISTE DES TACHES\n");
        try {
            listeTache=base.afficherTaches();
            for(Tache t : listeTache){
                System.out.println("\n\t\t\t\tIDENTIFIANT : " +
t.getId()+ "\n\t\t\t\tNOM : " + t.getNom() + "\n\t\t\t\tDESCRIPTION"+
t.getDescription() + "\n\t\t\t\tSTATUS" +
t.getStatus() + "\n\t\t\t\tIDENTIFIANT
MEMBRE"+ t.getIdMembre());
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        System.out.print("\n\t\t\t\t---> ID : ");
        Application.verifierType();
        int identifiantTacheASupprimer=sc.nextInt();
        System.out.print("\t\t\t\tVoulez-vous proceder a la
suppression ?\n\t\t\t\tt1 : OUI\t2 : NON\n\t\t\t\ttChoix : ");
        reponse=0;
        while(reponse<1 || reponse>2){
            Application.verifierType();
            reponse=sc.nextInt();
            if(reponse<1 || reponse>2){
                System.out.print("\t\t\t\tEntrez 1 ou
2\n\t\t\t\ttChoix : ");
            }
        }
        if(reponse==1){
            base.supprimerTache(identifiantTacheASupprimer);
        }
        System.out.print("\t\t\t\tVoulez-vous supprimer une autre
tache ?\n\t\t\t\tt1 : OUI\t2 : NON\n\t\t\t\ttChoix : ");
        reponse=0;
        while(reponse<1 || reponse >2){
            Application.verifierType();
            reponse=sc.nextInt();
            if(reponse<1 || reponse>2){
                System.out.print("\t\t\t\tEntrez 1 ou
2\n\t\t\t\ttChoix : ");
            }
        }
    }
}

```

```

    }
}

private static void assignerTache(){
    //Assigner tache a un membre code here
    reponse=1;
    List<Tache> listeTache;
    while(reponse==1){
        System.out.println("\n\n\t\t\t\t\t--> LISTE DES TACHES\n");
        try {
            //Affichage des taches
            listeTache=base.afficherTaches();
            for(Tache t : listeTache){
                System.out.println("\n\t\t\t\t\tIDENTIFIANT : " +
t.getId()+ "\n\t\t\t\t\tNOM : " + t.getNom() + "\n\t\t\t\t\tDESCRIPTION"+
t.getDescription() + "\n\t\t\t\t\tSTATUS" +
t.getStatus() + "\n\t\t\t\t\tIDENTIFIANT
MEMBRE"+ t.getIdMembre());
                System.out.println();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        System.out.print("\n\t\t\t\t\tID Tache : ");
        Application.verifierType();
        int identifiantTacheAAssigner=sc.nextInt();

        //Affichage des membres
        System.out.println("\n\n\t\t\t\t\t--> LISTE DES MEMBRES");
        List<Membre> listeMembre;
        try {
            listeMembre = base.afficherMembres();
            for(Membre t : listeMembre){
                System.out.println("\n\t\t\t\t\tIDENTIFIANT : " +
t.getId()+ "\n\t\t\t\t\tNOM : " + t.getNom());
                System.out.println();
            }
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        System.out.print("\n\t\t\t\t\tID membre : ");
        Application.verifierType();
        int identifiantMembreAAssigner=sc.nextInt();
        System.out.print("\t\t\t\t\tVoulez-vous proceder a l'assignation
?\n\t\t\t\t\t1 : OUI\t2 : NON\nChoix : ");
        reponse=0;
        while(reponse<1 || reponse>2){
            Application.verifierType();
            reponse=sc.nextInt();
            if(reponse<1 || reponse>2){
                System.out.print("\t\t\t\t\tEntrez 1 ou
2\n\t\t\t\t\tChoix : ");
            }
        }
        if(reponse==1){
            base.assignerTacheAUnMembre(identifiantTacheAAssigner,
identifiantMembreAAssigner);
        }
    }
}

```



```

        System.out.println("\t\t\t\tVoulez-vous assigner une autre
tache ?\n\t\t\t\tt1 : OUI\t2 : NON\n\t\t\t\t\tChoix : ");
        reponse=0;
        while(reponse<1 || reponse >2){
            Application.verifierType();
            reponse=sc.nextInt();
            if(reponse<1 || reponse>2){
                System.out.print("\t\t\t\tEntrez 1 ou
2\n\t\t\t\t\tChoix : ");
            }
        }
    }
}

private static void afficherTaches(){
    try {
        AffichageSurEcran.afficherTaches(base.afficherTaches());
    } catch (SQLException e1) {
        e1.printStackTrace();
    }
}
}
}

```

Classe GestionMembres.java

```

package com.gestiontache.milorme.manager;

import java.sql.SQLException;
import java.util.List;
import com.gestiontache.milorme.affichage.AffichageSurEcran;
import com.gestiontache.milorme.entites.Membre;
import com.gestiontache.milorme.app.Application;

public class GestionMembres implements Gestion{

    private static int reponse;

    public static void gererMembres(){

        AffichageSurEcran.afficherMenuGestionMembres();
        reponse=0;

        while(reponse<1 || reponse>5){
            Application.verifierType();
            reponse = sc.nextInt();
            if(reponse<1 || reponse>5)
                System.out.println("Entrez un entier entre 1 et 5
inclus\n");
        }

        switch(reponse){
            case 1 :
                creerMembre();
                break;
            case 2 :
                modifierMembre();
                break;
        }
    }
}

```

```

        case 3 :
            supprimerMembre();
            break;
        case 4 :
            afficherMembres();
            break;
        case 5 :
            System.out.println("\nOperation annulee\n");
            break;
    }
}

private static void creerMembre(){
    reponse=1;
    System.out.println("\n\t\t\t\t\t---> CREATION MEMBRE");
    while(reponse==1){
        System.out.print("\n\t\t\t\t\t\t\tNom membre : ");
        sc.nextLine();
        String nomMembre=sc.nextLine();
        System.out.print("\n\t\t\t\t\t\t\tAjouter le membre
cree ?\n\t\t\t\t\t\t\t1 : OUI\t2 : NON\n\t\t\t\t\t\t\tChoix : ");
        reponse=0;
        while(reponse<1 || reponse>2){
            Application.verifierType();
            reponse=sc.nextInt();
            if(reponse<1 || reponse>2){
                System.out.print("\t\t\t\t\t\t\tEntrez 1 ou
2\n\t\t\t\t\t\t\tChoix : ");
            }
        }
        if(reponse==1){
            base.creerMembre(new Membre(0,nomMembre));
        }
        System.out.print("\n\t\t\t\t\t\t\tVoulez-vous creer un autre
membre ?\n\t\t\t\t\t\t\t1 : OUI\t2 : NON\n\t\t\t\t\t\t\tChoix : ");
        reponse=0;
        while(reponse<1 || reponse >2){
            Application.verifierType();
            reponse=sc.nextInt();
            if(reponse<1 || reponse>2){
                System.out.println("\t\t\t\t\t\t\tEntrez 1 ou
2\n\t\t\t\t\t\t\tChoix : ");
            }
        }
    }
}

private static void modifierMembre(){
    reponse=1;
    List<Membre> listeMembre;

    while(reponse==1){
        try {
            //Affichage des membres
            System.out.println("\n\n\t\t\t\t\t---> LISTE DES
MEMBRES");

            listeMembre=base.afficherMembres();
            for(Membre t : listeMembre){

```

```

        System.out.println("\n\t\t\t\t\tIDENTIFIANT : " +
t.getId()+ "\n\t\t\t\t\tNOM : " + t.getNom());
        System.out.println();
    }
} catch (SQLException e) {
    e.printStackTrace();
}

System.out.print("\n\t\t\t\t\t---> Identifiant du membre a
modifier\n\t\t\t\t\tID : ");
Application.verifierType();
int identifiantMembreAModifier=sc.nextInt();

System.out.println("\n\t\t\t\t\t---> Modification du membre\n");
System.out.print("\t\t\t\t\tNom du membre : ");
sc.nextLine();
String nomMembreAModifier=sc.nextLine();
System.out.print("\t\t\t\t\tVoulez-vous proceder a la
modification ?\n\t\t\t\t\tt1 : OUI\t2 : NON\n\t\t\t\t\tChoix : ");
reponse=0;
while(reponse<1 || reponse>2){
    Application.verifierType();
    reponse=sc.nextInt();
    if(reponse<1 || reponse>2){
        System.out.print("\t\t\t\t\tEntrez 1 ou
2\n\t\t\t\t\tChoix : ");
    }
}
if(reponse==1){
    base.modifierMembre(new
Membre(identifiantMembreAModifier,nomMembreAModifier));
}
System.out.println("\t\t\t\t\tVoulez-vous modifier un autre
membre ?\n\t\t\t\t\tt1 : OUI\t2 : NON");
reponse=0;
while(reponse<1 || reponse >2){
    Application.verifierType();
    reponse=sc.nextInt();
    if(reponse<1 || reponse>2){
        System.out.println("\t\t\t\t\tEntrez 1 ou 2\nChoix :
");
    }
}

}

}

private static void supprimerMembre(){
    reponse=1;
    List<Membre> listeMembre;

    while(reponse==1){
        System.out.println("\n\n\t\t\t\t\t---> LISTE DES MEMBRES");
        try {
            listeMembre=base.afficherMembres();
            for(Membre t : listeMembre){
                System.out.println("\n\t\t\t\t\tID : " + t.getId()+
"\n\t\t\t\t\tNOM : " + t.getNom());
                System.out.println();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

```

```

    }
    System.out.print("\n\t\t\t\t\t---> Identifiant du membre a
supprimer : ");
    Application.verifierType();
    int identifiantMembre=sc.nextInt();
    System.out.println("\t\t\t\t\tVoulez-vous proceder a la
suppression ?\n\t\t\t\t\tt1 : OUI\t2 : NON");
    reponse=0;
    while(reponse<1 || reponse>2){
        Application.verifierType();
        reponse=sc.nextInt();
        if(reponse<1 || reponse>2){
            System.out.println("\t\t\t\t\tEntrez 1 ou
2\n\t\t\t\t\t\tChoix : ");
        }
    }
    if(reponse==1){
        base.supprimerMembre(identifiantMembre);
    }
    System.out.print("\t\t\t\t\tVoulez-vous supprimer un autre
membre ?\n\t\t\t\t\tt1 : OUI\t2 : NON\n\t\t\t\t\t\tChoix : ");
    reponse=0;
    while(reponse<1 || reponse >2){
        Application.verifierType();
        reponse=sc.nextInt();
        if(reponse<1 || reponse>2){
            System.out.println("\t\t\t\t\tEntrez 1 ou
2\n\t\t\t\t\t\tChoix : ");
        }
    }
}

private static void afficherMembres(){
    try {
        AffichageSurEcran.afficherMembres(base.afficherMembres());
    } catch (SQLException e1) {
        e1.printStackTrace();
    }
}
}

```

Classe Membre.java

```

package com.gestiontache.milorme.entites;

public class Membre {
    //Constructeur
    public Membre(int id, String nom) {
        this.id = id;
        this.nom = nom;
    }

    //Getters et Setters
    public int getId() {
        return id;
    }
}

```

```

    public void setId(int id) {
        this.id = id;
    }

    public String getNom() {
        return nom;
    }

    public void setNom(String nom) {
        this.nom = nom;
    }

    //Variables d'instances
    private int id;
    private String nom;
}

```

Classe Tache.java

package com.gestiontache.milorme.entites;

```

public class Tache {

    //Constructeur
    public Tache(int id, String nom, String description, String status, int
idMembre) {
        super();
        this.id = id;
        this.nom = nom;
        this.description = description;
        this.status = status;
        this.idMembre = idMembre;
    }

    //Getters et Setters
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNom() {
        return nom;
    }

    public void setNom(String nom) {
        this.nom = nom;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }
}

```

```

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public int getIdMembre() {
        return idMembre;
    }

    public void setIdMembre(int idMembre) {
        this.idMembre = idMembre;
    }

    //Variables d'instances
    private int id;
    private String nom;
    private String description;
    private String status;
    private int idMembre; //Identifiant du membre associe a la tache
}

```

Classe ConnexionBaseDeDonnees.java

```

package com.gestiontache.milorme.basededonnees;

```

```

import java.io.File;
import java.sql.*;
import java.util.ArrayList;
import java.util.Hashtable;
import java.util.List;
import com.gestiontache.milorme.entites.Tache;
import com.gestiontache.milorme.entites.Membre;

```

```

public class ConnexionBaseDeDonnees {
    //Variables representant les ressources manipulees
    private static ConnexionBaseDeDonnees
connexionBaseDeDonneesSingleton=null;
    Connection c = null;
    Statement stmt = null;
    ResultSet rs=null;
    final String NOM_BASE_DE_DONNEES ="gestiontache.db";

    //private final String STATUS_NOUVEAU="nouveau";
    //private final String STATUS_EN_PROGRES="en-progres";
    //private final String STATUS_TERMINE="termine";

    //Methode permettant de recuperer l'objet singleton de la base de donnees
    public static ConnexionBaseDeDonnees recupererObjetSingleton(){
        if(connexionBaseDeDonneesSingleton==null){
            connexionBaseDeDonneesSingleton = new
ConnexionBaseDeDonnees();
        }
        return connexionBaseDeDonneesSingleton;
    }
}

```

```

    }

    //Constructeur de la classe permettant d'initialiser, d'instancier et de
    creer une connexion a la base de donnees
    private ConnexionBaseDeDonnees() {
        boolean fichierExiste=false;
        File f1 = new File(NOM_BASE_DE_DONNEES);
        //Test de l'existence de la base
        if(f1.exists()){
            fichierExiste=true;
        }
        try {
            Class.forName("org.sqlite.JDBC");
            c = DriverManager.getConnection("jdbc:sqlite:" +
NOM_BASE_DE_DONNEES);
            c.setAutoCommit(true);

            stmt = c.createStatement();

            if(!fichierExiste){
                creerTablesDansLaBaseDeDonnees();
            }

        } catch (Exception e) {
            //System.err.println(e.getClass().getName() + ": " +
e.getMessage());
            System.out.println("\nErreur au niveau de la base de
donnees\n");
            System.exit(0);
        }
    }

    //Methode de creation des tables dans la base de donnees
    private void creerTablesDansLaBaseDeDonnees(){

        String sql;

        try {
            sql = "CREATE TABLE TACHE " +
                "(ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL ,"
+
                " NOM TEXT NOT NULL," +
                "DESCRIPTION TEXT NOT NULL,"+
                "STATUS TEXT NOT NULL," +
                "ID_MEMBRE INTEGER NULL)";
            stmt.executeUpdate(sql);

            sql="CREATE TABLE MEMBRE " +
                "(ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL ,"
+
                " NOM TEXT NOT NULL)";
            stmt.executeUpdate(sql);

        } catch ( Exception e ) {
            //System.err.println( e.getClass().getName() + ": " +
e.getMessage() );
            System.out.println("\nCreation des tables echouee\n");
            System.exit(0);
        }
        System.out.println("Creation des tables TACHE et MEMBRE reussie\n");
    }

```

```

//Methode permettant de creer une tache
public void creerTache(Tache tache) {
    int idMembre=tache.getIdMembre();
    String status="", sql="";

    tache.setNom(tache.getNom().replace('\\', ' '));
    tache.setDescription((tache.getDescription().replace('\\', ' ')));
    tache.setStatus((tache.getNom().replace('\\', ' ')));

    if( idMembre<=0){
        status="nouveau";
        sql="INSERT INTO TACHE(NOM, DESCRIPTION, STATUS, ID_MEMBRE) "
+ "VALUES('" + tache.getNom()+", '" + tache.getDescription() + "', '" + status +
"' , NULL );";
    }else{
        status="en-progres";
        sql="INSERT INTO TACHE(NOM, DESCRIPTION, STATUS, ID_MEMBRE) "
+ "VALUES('" + tache.getNom()+", '" + tache.getDescription() + "', '" + status +
"' ,'" + idMembre + "');"
    }

    try {
        stmt.executeUpdate(sql);
        sql="UPDATE TACHE SET ID_MEMBRE =NULL WHERE ID_MEMBRE=0;";
        stmt.executeUpdate(sql);
        System.out.println("\n\t\t\t\tTache enregistree\n");
    } catch (SQLException e) {
        e.printStackTrace();
        System.out.println("\n\t\t\t\tErreur !. La tache n'a pas ete
cree.\n");
    }
}

```

```

//Methode permettant de creer un membre
public void creerMembre(Membre membre) {

    membre.setNom(membre.getNom().replace('\\', ' '));

    try {
        stmt.executeUpdate("INSERT INTO MEMBRE (NOM) " +
            "VALUES ('" + membre.getNom()+ "')");
        System.out.println("\n\t\t\t\tMembre enregistre\n");
    } catch (SQLException e) {
        System.out.println("\n\t\t\t\tErreur ! Le membre n'a pas
ete cree\n");
    }
}

```

```

//Methode permettant d'afficher les taches assignees a un membre
public List<Tache> afficherTachesMembre(int identifiant) throws
SQLException {
    rs = stmt.executeQuery("SELECT * FROM TACHE WHERE ID_MEMBRE = " +
    identifiant + ";");
    List<Tache> listeTachesmembre = new ArrayList<Tache>();

    while (rs.next()) {
        int id = rs.getInt("ID");
        String name = rs.getString("NOM");
        String description = rs.getString("DESCRIPTION");
        String status = rs.getString("STATUS");
        int idMembre = rs.getInt("ID_MEMBRE");
    }
}

```



```

        listeTachesmembre.add(new Tache(id, name, description, status,
idMembre));
    }
    return listeTachesmembre;
}

//Methode permettant d'afficher toutes les taches selon leurs status et
les infos du membre associe
public List<Object[]> afficherStatusTaches(int statusARechercher) throws
SQLException {

    List<Object[]> listeTachesStatus = new ArrayList<Object[]>();
    //List temporaire=new ArrayList();
    Object[] temporaire =new Object[2];
    String stringStatus;

    if (statusARechercher == 1) {
        stringStatus = "nouveau";
    } else if (statusARechercher == 2) {
        stringStatus = "en-progres";
    } else {
        stringStatus = "termine";
    }

    rs = stmt.executeQuery("SELECT T.ID AS tacheID, T.NOM AS tacheNOM,
T.DESCRPTION AS tacheDESCRIPTION,"
        + "T.STATUS AS tacheSTATUS, M.ID AS membreID, M.NOM AS
membreNOM FROM TACHE AS T "
        + "LEFT JOIN MEMBRE AS M ON T.ID_MEMBRE = M.ID WHERE
T.STATUS = '"+stringStatus+"'");

    while (rs.next()) {
        //Creation objet tache
        int id = rs.getInt("tacheID");
        String name = rs.getString("tacheNOM");
        String description = rs.getString("tacheDESCRIPTION");
        String status = rs.getString("tacheSTATUS");

        //Creation objet membre
        int idMembre = rs.getInt("membreID");
        String nomMembre=rs.getString("membreNOM");

        temporaire[0]=new Tache(id, name, description, status,
idMembre);
        temporaire[1]=new Membre(idMembre,nomMembre);
        listeTachesStatus.add(temporaire);
    }
    return listeTachesStatus;
}

// Methode permettant de recuperer et d'afficher tous les membres
public List<Membre> afficherMembres() throws SQLException {
    rs = stmt.executeQuery("SELECT * FROM MEMBRE;");
    List<Membre> listeMembres = new ArrayList<Membre>();
    while (rs.next()) {
        int id = rs.getInt("ID");
        String name = rs.getString("NOM");
        listeMembres.add(new Membre(id, name));
    }
    return listeMembres;
}

// Methode permettant de recuperer et d'afficher toutes les taches

```

```

public List<Tache> afficherTaches() throws SQLException {
    rs = stmt.executeQuery("SELECT * FROM TACHE;");
    List<Tache> listeTaches = new ArrayList<Tache>();

    while (rs.next()) {
        int id = rs.getInt("ID");
        String name = rs.getString("NOM");
        String description = rs.getString("DESCRIPTION");
        String status = rs.getString("STATUS");
        int idMembre = rs.getInt("ID_MEMBRE");
        listeTaches.add(new Tache(id, name, description, status,
idMembre));
    }
    return listeTaches;
}

//Modifications
public void modifierTache(Tache tache) {

    tache.setNom(tache.getNom().replace('\\', ' '));
    tache.setDescription((tache.getDescription().replace('\\', ' ')));
    tache.setStatus((tache.getNom().replace('\\', ' ')));

    try {
        stmt.executeUpdate("UPDATE TACHE SET NOM='"+tache.getNom()+
+"',DESCRIPTION='"+tache.getDescription()+
        "' ,STATUS='"+tache.getStatus()+
+"',ID_MEMBRE='"+tache.getIdMembre()+
        "' WHERE ID='"+tache.getId()+"'");

        System.out.println("\n\t\t\t\tTache modifie\n");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

public void modifierMembre(Membre membre) {

    membre.setNom(membre.getNom().replace('\\', ' '));

    try {
        stmt.executeUpdate("UPDATE MEMBRE SET NOM='"+membre.getNom()+
        "' WHERE ID='"+membre.getId()+"'");

        System.out.println("\n\t\t\t\tMembre modifie\n");
    } catch (SQLException e) {
        //e.printStackTrace();
        System.out.println("\t\t\t\t!!! Echec de la modification !!!
Verifiez l'ID\n");
    }
}

//Methode d'assignation de tache
public void assignerTacheAUnMembre(int idTache, int idMembre){
    try {
        stmt.executeUpdate("UPDATE TACHE SET ID_MEMBRE='"+idMembre
        +"'WHERE ID='"+idTache+"'");

        System.out.println("\n\t\t\t\tTache assignee\n");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

```

}

//Suppressions
public void supprimerTache(int idTache) {
    String sql = "DELETE FROM TACHE WHERE ID="+ idTache + ";";
    try {
        stmt.executeUpdate(sql);
    } catch (SQLException e) {
        e.printStackTrace();
    }
    System.out.println("\n\t\t\t\tTache supprimee\n");
}

public void supprimerMembre(int idMembre) {
    String sql = "DELETE FROM MEMBRE WHERE ID="+ idMembre + ";";
    try {
        stmt.executeUpdate(sql);
        sql="UPDATE TACHE SET ID_MEMBRE = NULL WHERE ID_MEMBRE=" +
idMembre + ";";
    } catch (SQLException e) {
        e.printStackTrace();
    }
    System.out.println("\n\t\t\t\tMembre supprime\n");
}

// Methode permettant de fermer et liberer les ressources
public void fermerRessourceBaseDeDonnees() {
    try {
        stmt.close();
        c.close();
    } catch (SQLException e) {
        e.printStackTrace();
        System.out.println("Erreur de fermeture des ressources de la
base de donnees\n\n");
    }
}
}

```