Université Nationale du Vietnam, Hanoï, (UNV)
Institut Francophone pour l'Informatique, IFI

Rapport De TP

GÉNIE LOGICIEL AVANCÉ

Sujet : Réalisation d'un Petit Gestionnaire de taches

Réalisé par *:*
SYLLA ABOUBACAR
Promo 21, IFI, Aout 2017

Table des matières

1 - INTRODUCTION

Dans le cadre de la réalisation du premier projet de mise a niveau du cours de **Génie Logiciel Avancé**, ce présent rapport explique et clarifie le projet qui consiste à concevoir une application dénommée « **Un petit** G***estionnaire des Taches »***. Grâce à ce travail, nous comprendrons de façon générale les notions liées à la modélisation avec UML, à la programmation orientée-objet et au langage de programmation « java ».
En plus de ces concepts, ce TP nous permet également de savoir et d'apprendre l' environnement de développement intégré (IDE) « libre » (Open Source) ECLIPSE.

Dans ce cas ci, l'outil qui sera développé nous permettra de gérer non seulement les taches et les membres dans une équipe de travail mais également de procéder à l'affectation des taches aux différents membres selon leurs statuts. Cet outil offre plusieurs avantages dans le traitement manuel de la gestion des taches. il lui permet aussi de gagner en temps soit dans la recherche des taches non assignées, soit dans la recherche des membres non affectés aux taches, mais aussi dans la sauvegarde des assignations.

1.1 SPÉCIFICATION DE L'APPLICATION

Pour la réalisation de ce projet ***Petit Gestionnaire de Taches*** nous avons essentiellement deux entités d'entrées a savoir :

- Tache

Pour cette première entité nous aurons les propriétés suivantes :
- ID
- Nom
- Description
- Statut

- Membre

pour cette deuxième entité nous avons les informations qui suivent
- ID
- Nom

A partir de ces deux entités nous avons opté après la modélisation UML pour une architecture de plusieurs a plusieurs.
Cette architecture nous a permet d'avoir une troisième entrée ayant des propriétés dépendantes des deux premières entités. Cette entité permet de renseigner et enregistrer les assignations en utilisant les ID des classes Tache et Personne.

- Assigner
  ID_Tache
  ID_Membre

Cela explique par le faite qu'une tache selon son statut et sa taille peut être affectées a plusieurs personnes.
Et aussi éviter d'assigner des taches aux membres des la création d'une tache.
Une maniéré pour nous de séparer ses deux fonctionnalités.

Ce ***Gestionnaire de Taches*** fournit à l'utilisateur les fonctionnalités suivantes:

- Créer, modifier, supprimer, ajouter une tâche;
- Créer, modifier, supprimer, ajouter un membre ;
- Afficher les taches disponible dans la liste ;
- Afficher les membres enregistres ;
- Assigner une tâche à un membre ;
- Chercher et afficher toutes les tâches assignées à un membre (par son ID) ;
- Chercher et afficher toutes les tâches en fonction de leur statuts (avec le nom du assigné).

2 Exigences Fonctionnelles et Non-Fonctionnelles de l'Application

2. 1 Exigences Fonctionnelles (EF)

Les Exigences Fonctionnelles (EF) expriment la raison d'être d'un système et les idées a être incarnées dans une application en développement. Elles décrivent également 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 :

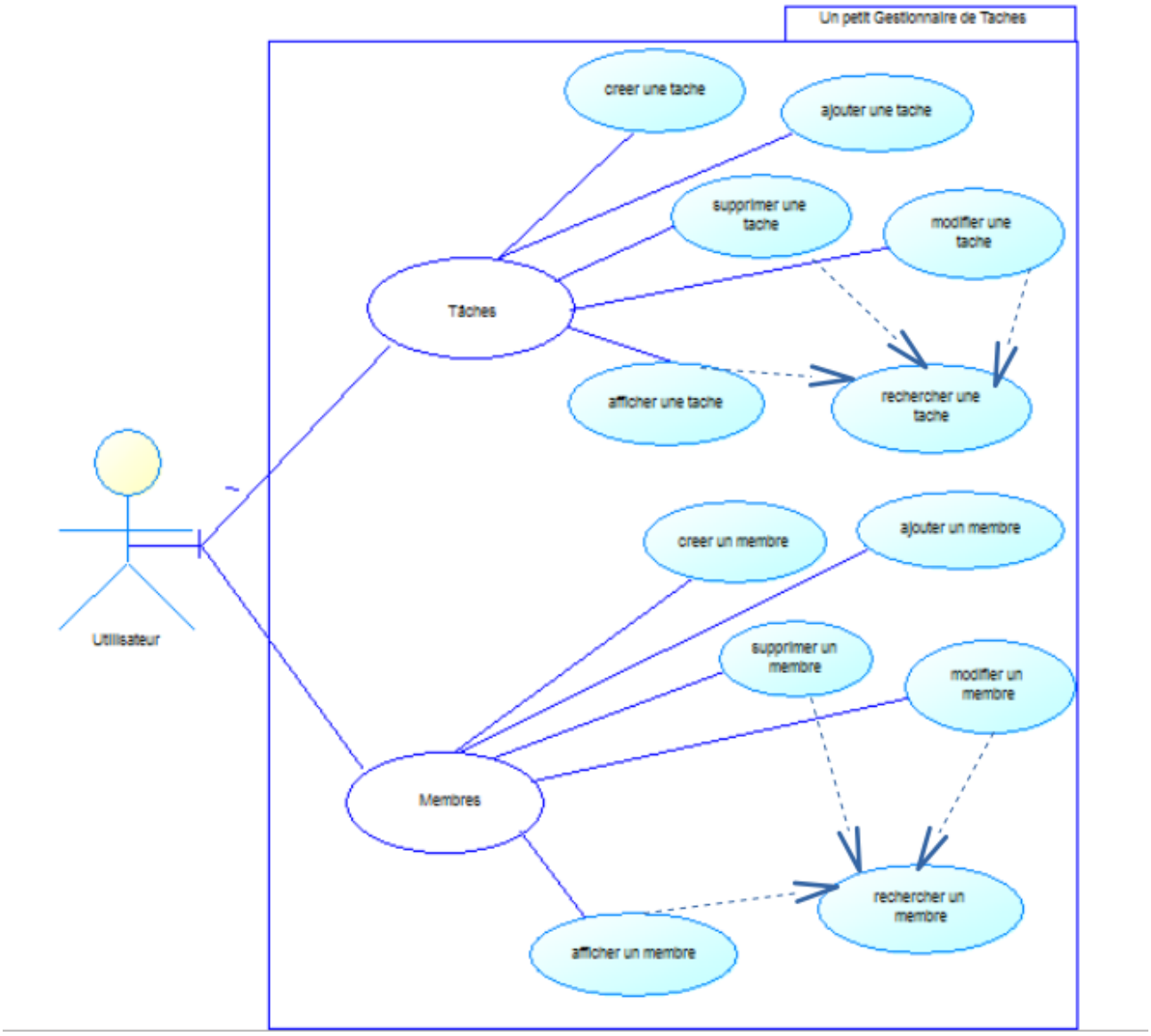
EF I	**Créer une tache**
	Dans la phase de création d'une tache, l'application doit pouvoir: -Demander de saisir les informations concernant la tache -Enregistrer les informations saisies
EF II	**Ajouter une tache**
	Dans la phase de l'ajout d'une tache, l'application doit pouvoir: -Demander si cette tache existe déjà pour confirmation -Enregistrer les informations saisies
EF III	**Suppression d'une tache**
	Cette étape de suppression d'une tache pousse le système a faire les points suivants : -Se rassurer de l'existence de la tache avant la suppression -Autoriser la suppression
EF IV	**Recherche d'une tache**
	Le système doit pouvoir : -Permettre de rechercher les taches par son ID et son nom une fois entré via le clavier.
EF V	**Modification des Taches**
	L'application a ce niveau doit être capable de : - Permettre la modification d'une tache une fois saisi en entrée

	- Enregistrer les informations après la modification
EF VI	**Affichage des Taches**
	Cet outil doit pouvoir : - Permettre l'affichage toutes les personnes avec leurs coordonnées respectives
EF VII	**Créer un membre**
	Dans la phase de création d'une tache, l'application doit pouvoir: -Demander de saisir les informations concernant le membre -Enregistrer les informations saisies
EF VIII	**Ajouter un membre**
	Dans la phase de l'ajout d'un membre l'application doit pouvoir: -Demander si cet membre existe déjà pour confirmation -Enregistrer les informations saisies
EF IX	**Suppression d'une membre**
	Cette étape de suppression d'un membre pousse le système a faire les points suivants : -Se rassurer de l'existence du membre avant la suppression -Autoriser la suppression
EF X	**Recherche d'une membre**
	Le système doit pouvoir : -Permettre de rechercher les membres par son ID et son nom une fois entré via le clavier.
EF XI	**Modification des Membres**
	L'application a ce niveau doit être capable de : - Permettre la modification d'un membre une fois saisi au clavier - Enregistrer les informations après la modification
EF XII	**Affichage des Membres**
	Cet outil doit pouvoir : - Permettre l'affichage de tous les membres enregistres
EF XIII	**Affichage des taches assignées aux membres**
	Cet outil doit pouvoir : - Permettre l'affichage de toutes les assignes et leurs taches respectives

a. Diagramme de Cas d'Utilisation

Les diagrammes de cas d'utilisation représentent généralement toutes les interactions des utilisateurs avec le système.
Un **cas d'utilisation** représente une unité discrète d'interaction entre un utilisateur (humain ou machine) et un système. Il est une unité significative de travail. Elle pour donner une vision globale du comportement fonctionnel d'un système logiciel. Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet,
dans le cadre de ce travail nous présentons cette représentation graphique qui illustre les rôles qu'un utilisateur pourrait être confronté

Figure 1



la figure ci-dessus illustre les différentes fonctionnalités que doit faire notre application lorsqu'un utilisateur lamda veut interagir avec lui.

2.2 Les Exigences Non-Fonctionnelles (ENF)

Les Exigences Non-Fonctionnelles, ENF sont des exigences qui caractérisent une propriété ou une performance de qualité souhaitée par un système telle que son utilité, sa performance, sa robustesse, sa convivialité, sa maintenabilité, etc.

Dans le cadre de notre application, les ENF sont décrites comme suit :

Exigences Non-Fonctionnelles	Les Détails	Qualité / Attribut
ENF I	Le feedback doit s'exécuter dans un temps records maximum 1 seconde	Performance
ENF II	La cohérence dans l'affichage des résultats par rapport a ce que l'on attend de lui Elle doit être capable d'effectuer un bon algorithme ci nécessaire	Exactitude , Pertinence
ENF III	Avoir la capacité de sauvegarder les données en cas d'erreurs. Rendre accessible les données.	Disponibilité
ENF IV	Une application doit être interactive et facile a utiliser.	Compréhension
ENF V	La utilisabilité des codes sources d'une application doit se caractériser par la facilité a un administrateur de faire des entretiens et les mises a jour de l'application à l'avenir.	Faciliter la Maintenance

3. Conception et l'Architecture de l'Application

3.1 Diagramme de Classe

Le **diagramme de classes** est un schéma utilisé en génie logiciel pour présenter les classes et les interface des systèmes ainsi que les différentes relations entre celles-ci. il fait partie de la partie statique d'UML car il fait abstraction des aspects temporels et dynamiques.
Dans le cadre de la conception de notre application nous proposons un concept de many to many.

Voici une illustration de notre diagramme de classe ci-dessous

Figure 2 *Le diagramme de classe.*

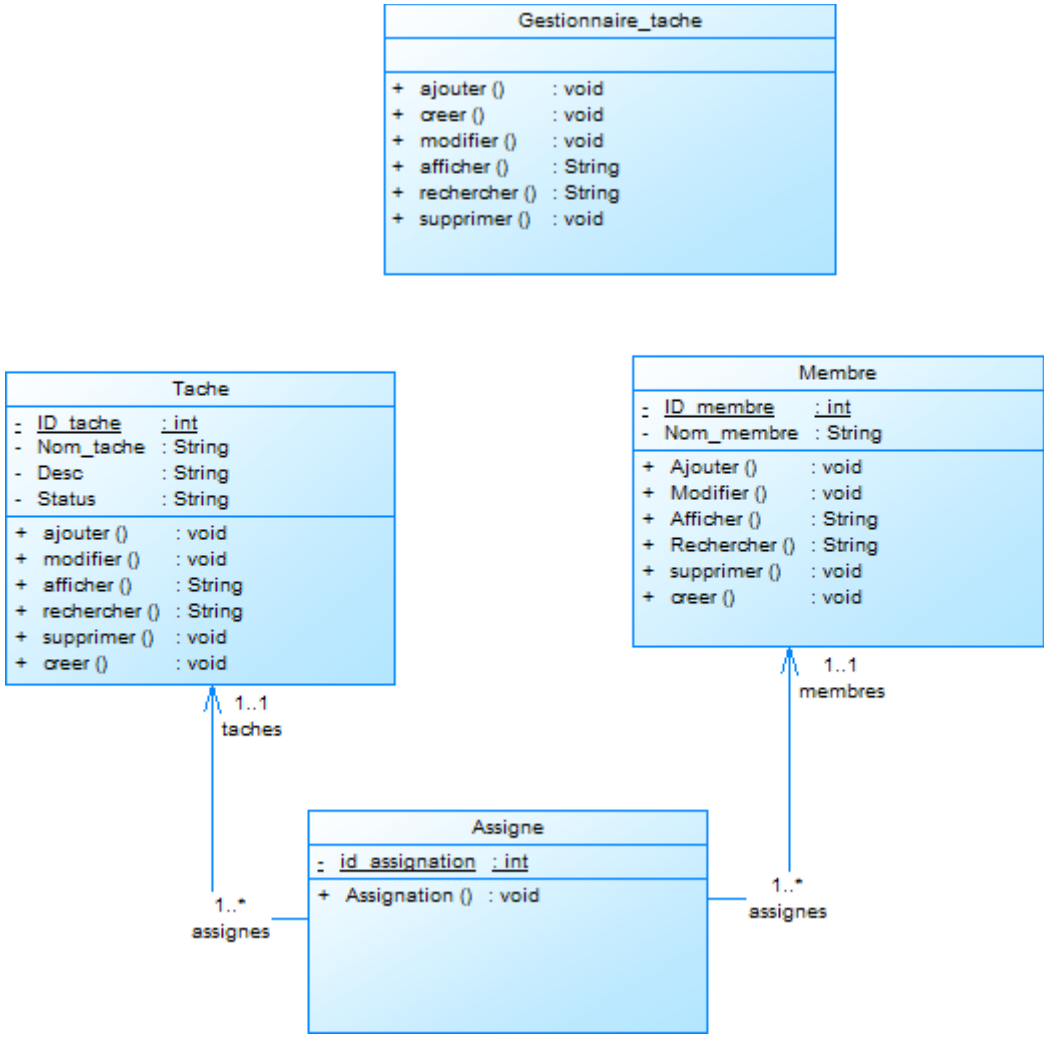
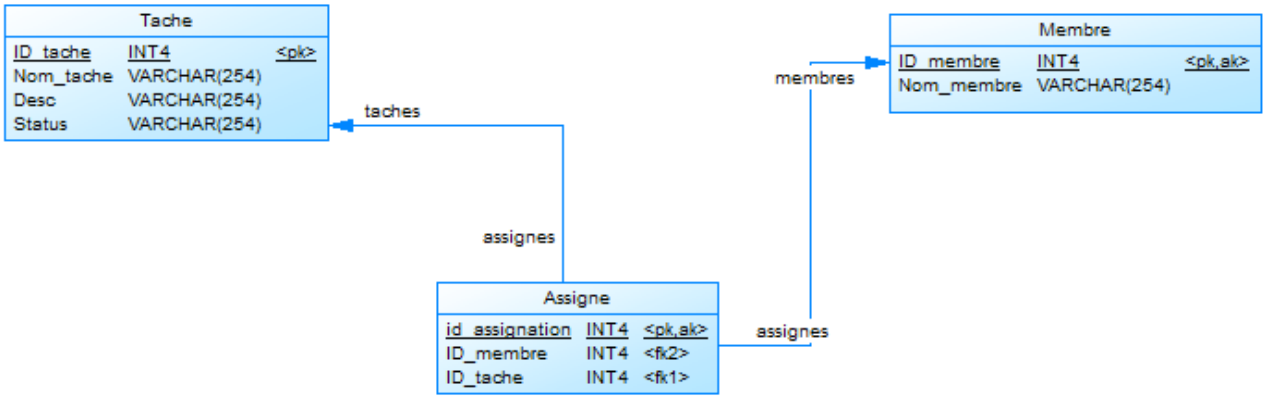


Figure 3. *Le Modèle Physique de données.*

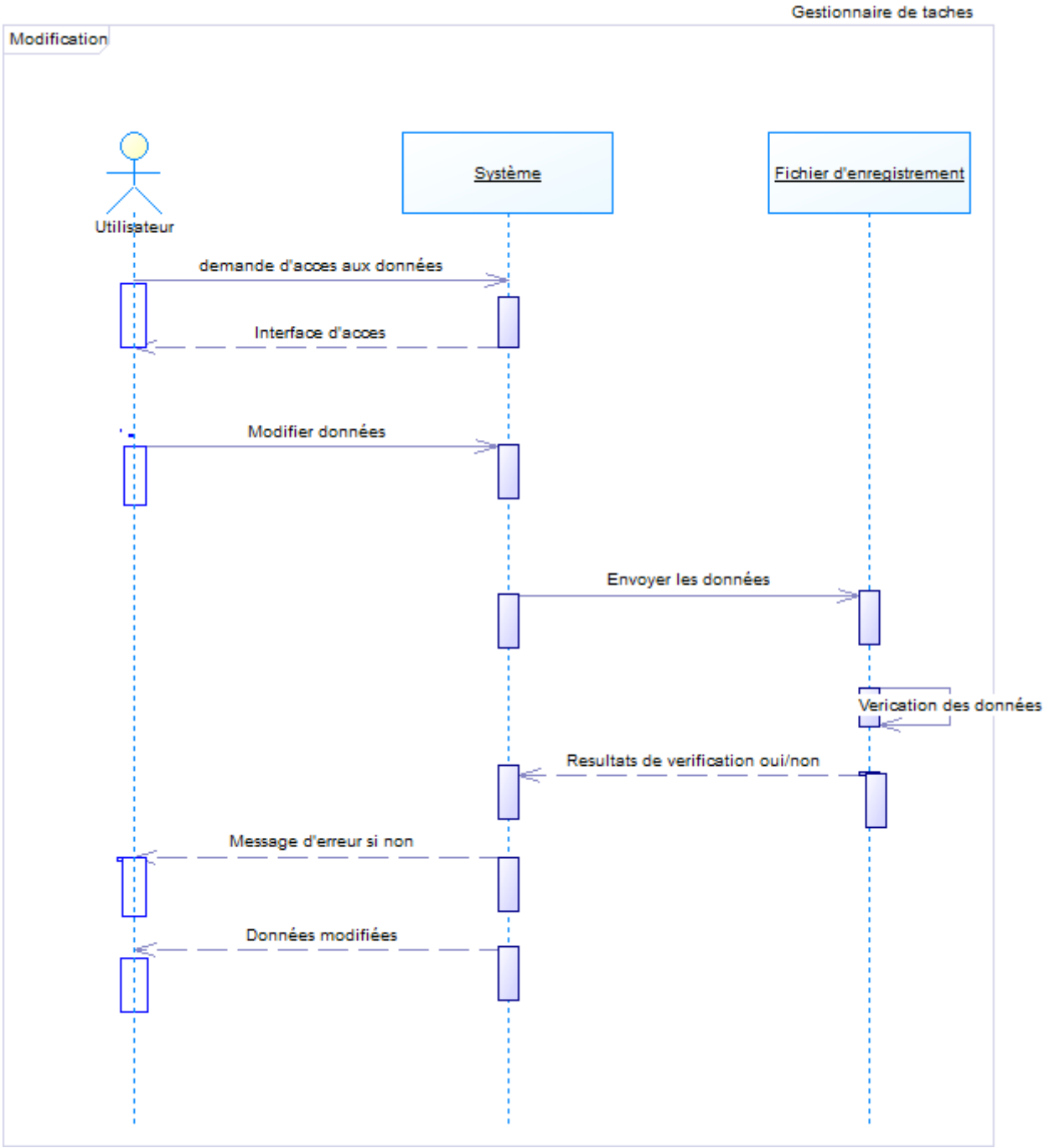


Les deux figures représentent le modèle de conception dont notre application sera conçue et implémenté. Dans notre Gestionnaire de taches nous avons trois classes a implémenté pour le fonctionnement de notre application en tenant compte des consignent du TP

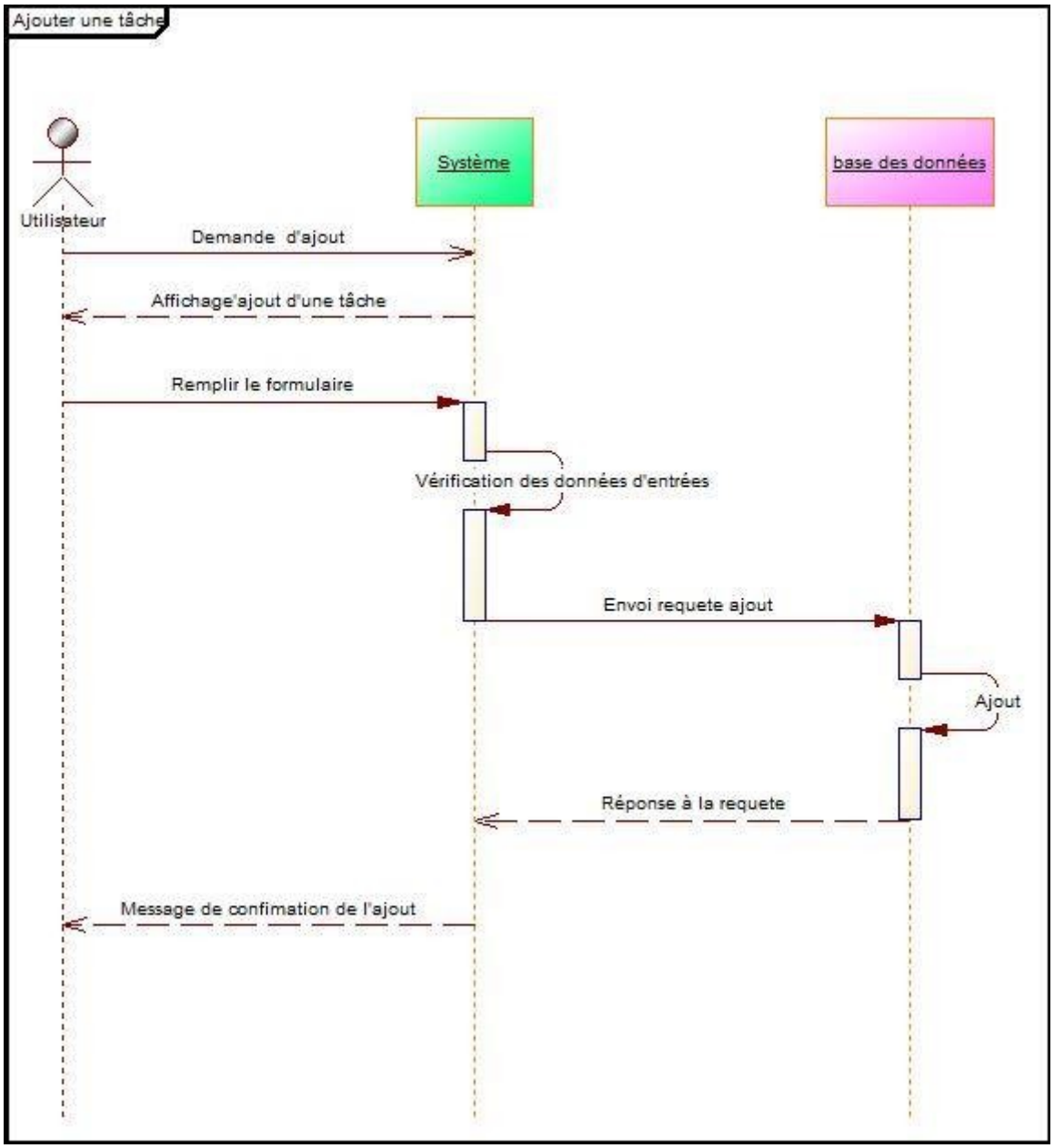
3.2 Diagramme de Séquence

Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique. En principe un diagramme de séquence en nombre selon les fonctionnalités d'une application. Mais nous nous choisirons l'exemple deux fonctions

A- Modifier d'une tache ou d'un membre :(Diagramme de Séquence)



B-Créer une tache ou un membre : Diagramme de Séquence



4. IMPLÉMENTATION ET TEST

4.0 Environnement Matériel

La conception de notre application a été réalisée sur une machine ASUS IN SEARCH OF INCREDIBLE
Un processeur Intel Dual core E1, 2.16 GHz.
Une mémoire vive de 2Go.
Un disque dur 500 Go.

4. 1 Environnement Logiciel

L'environnement de développement dans lequel cette application est réalisée est le système libre Ubuntu avec une version 16.0. L'IDE utilise est l'éclipse version néon 1.2.

4.2 Langage de programmation

Dans le cadre du respect strict des contraintes exigées dans le TP nous avons procéder a la programmation orientée objet. Ainsi, le langage utilisé pour l'implémentation de ce petit gestionnaire de taches est celui du java qui nous a permis de bien organiser notre projet.

4.3 Le plan du modèle de conception de l'application :

Pour rendre possible ce travail et en tenant compte de toutes les exigences, quatre classes nous ont servi pour concrétiser ce TP a savoir :

- La classe Membre qui définit les propriétés d'un membre;
- La classe Tache qui élabore les attributs et méthodes associes a une tache ;
- La classe Assigne qui permet de faire une jointure et les assignations ;
- La classe Gestion_tache qui permet de gérer toutes les fonctionnalités constituantes et qui joue le rôle de classe principale servant à l'exécution de notre application de notre application.

Notre application contient un total de 967 lignes de code écrites.

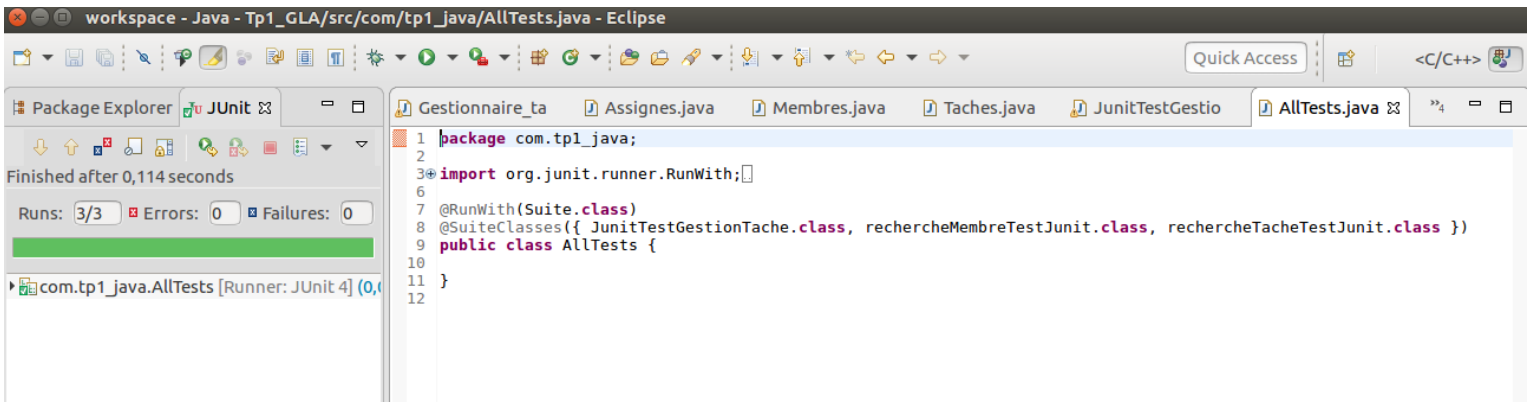
4.4 La sauvegarde des données

Vous constaterez dans la gestion de nos enregistrements nous avons stocké nos données dans des tableaux de listes de telles sortes que ces seront stockées dans des fichiers pour la sauvegarde. Nous enregistrons nos données principalement dans trois fichiers a savoir :
- le fichier pour la classe Membre
- le fichier pour la classe Tache
- le fichier pour l'assignation.

Ces différents fichiers sont chargés par instruction du système dans la liste du menu.

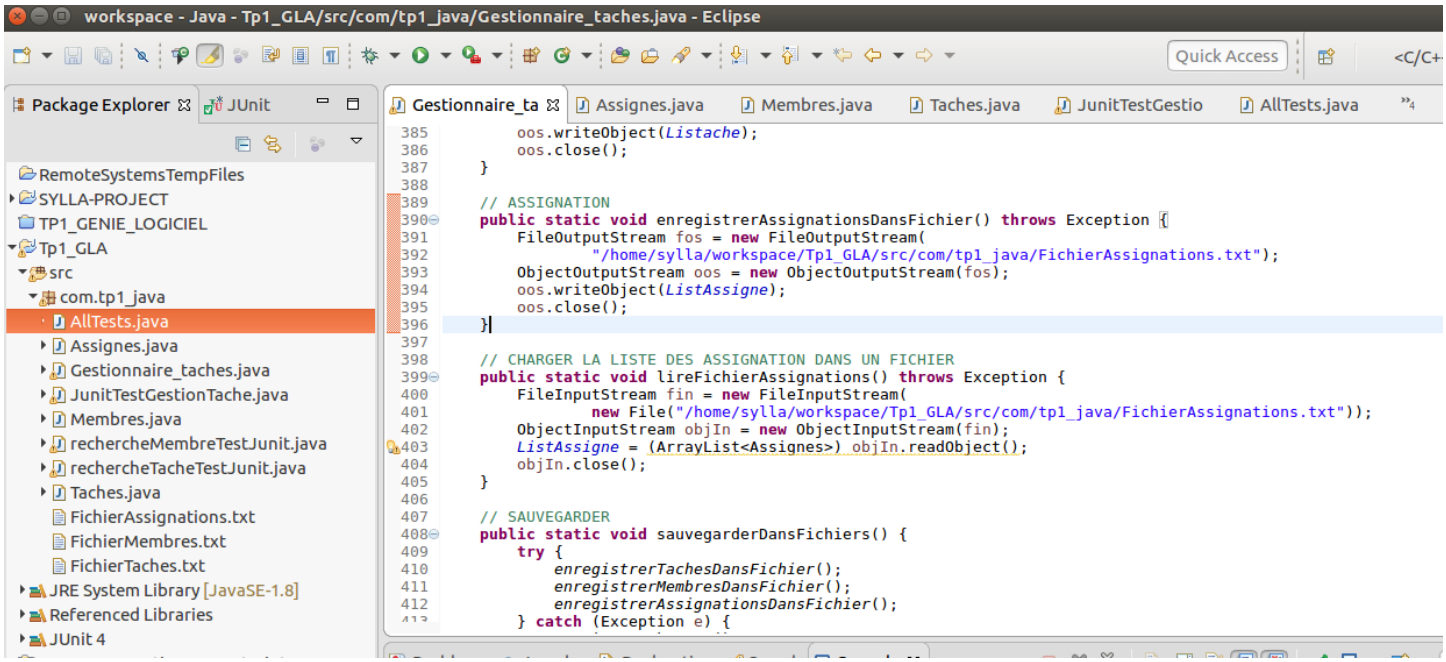
4.5 Méthode de test

Nous avons utilisé JUnit, pour faire le test unitaire et s'assurer du bon fonctionnement des méthodes utilisés, pour notre cas ce test a été applique sur deux fonctions entre autre la fonction : rechercheTache() et la fonction rechercheMembre().
Comme le montre la figure ci-dessous :



5. TEST D'ACCEPTATION

Avant de rentrer dans le vif du sujet nous vous présentons un aperçu de l'interface de compilation de notre application qui permettra une idée large de notre environnement de travail et de sa structuration



Dès après le lancement de notre petit gestionnaire de taches nous accédons au menu principal qui se présente comme suit :

```
|*****************************************************************************************************|
|**                                  MENU DE LA GESTION DES MEMBRES                                   **|
|*****************************************************************************************************|
|**                                                                                                   **|
|**                CHOISISSEZ UNE OPERATION  OU 0 POUR QUITTER ET ENREGISTRER                         **|
|**                                                                                                   **|
|*****************************************************************************************************|
|**                                    |                                                              **|
|**        1 - CREER UNE MEMBRE        |                                                              **|
|**                                    |                                                              **|
|**        2 - MODIFIER UN MEMBRE      |                                                              **|
|**                                    |                                                              **|
|**        3 - AFFICHER LES MEMBRES    |   9 - CREER UNE ASSIGNATION                                  **|
|**                                    |                                                              **|
|**        4 - SUPPRIMER UN MEMBRE     |   10 - AFFICHER LES ASSIGNATIONS                             **|
|**                                    |                                                              **|
|**************************************   11 - RECHERCHE DES TACHES ASSIGNEES PAR ID DU MEMBRE       **|
|**        5 - CREER UNE TACHE         |                                                              **|
|**                                    |   12 - RECHERCHE DES TACHES PAR LEURS STATUS ET NOM DE L'ASSIGNE **|
|**        6 - MODIFIER UNE TACHE      |                                                              **|
|**                                    |   0 - QUITTER ET SAUVEGARDER LES LISTES                      **|
|**        7 - AFFICHER LES TACHES     |                                                              **|
|**                                    |                                                              **|
|**        8 - SUPPRIMER UNE TACHE     |                                                              **|
|**                                    |                                                              **|
|*****************************************************************************************************|

DANS LE MENU DES MEMBRES FAITES VOTRE CHOIX :
```

A l'issu de ce menu nous procéderons à l'exécution de l'ensemble de toutes fonctionnalités parmi les quelles nous citerons quelques unes :

5.1 *Créer une tache*

L' interface ci-dessous permet de saisir les coordonnées pour créer une tache.

```
--------------------------------------
-------CREATION D'UNE TACHE-----------
Saisissez l'Identifiant :       21
Saisissez le Nom de la tache :  ENSEIGNER
Saisissez le Description de la tache :  IFI-VNU
Saisissez le Status de la tache :       NOUVEAU
------------------------------------------------------------
```

Cette interface nous permet saisir les informations concernant la tache et les enregistrer dans les fichiers et ensuite un message de continuation viendra pour une nouvelle ajout d'une tache.

5.2 Modifier d'une tache

```
-------------------------------------
-------MODIFICATION D'UNE TACHE-----------
Saisissez l'ancien de la tache :        DOCTEUR
Saisissez le nouveau nom de la tache :  COMPTABLE
Saisissez la nouvelle Description :     SEA-BANK
Saisissez le nouveau Status :   TERMINER
Voulez vous modifier une autre tache si oui 1 si non retourner sur le menu principal en tapant 0 :
```

Pour cette opération cette interface nous permet de modifier une selon les informations d'entrée comme l'indique la figure ci.

5.3 Supprimer une tache

Une fois choisi la commande une boite de dialogue invite à saisir les informations de la tache à supprimer. Comme vous constaterez ci dessous.

```
--------------------------------------
-------SUPPRESSION D'UNE TACHE-----------
Saisissez le nom du tache devant etre supprime :        DEMARCHEURS
------------------------------------------------------------
```

Pour supprimer , il faut saisir le nom de la tache. Une fois les informations saisies, le système lance la recherche pour trouver la tache et les informations relatives. Après avoir trouvé les informations il les supprime.

5.4 Affichage des assignations

Dans cette options nous afficherons la liste des assignes existants dans les listes et nous permettra de procéder a la visualisation des taches.
Vous remarquerez que chaque assignation est séparée par une ligne en pointée et est constituée de trois lignes a savoir : la ligne de l'assignation et les deux autres sont la tache et le membre associés.
La figure ci dessous en fait foi de cette explication.

```
          DANS LE MENU DES MEMBRES FAITES VOTRE CHOIX :     10
1       4       1

Tache : 4       INFORMATICIEN   IFI     NOUVEAU
Membre :        1       SYLLA
-------------------------------------------------
2       6       2

Tache : 6       MECANICIEN      GARAGE  PROGRES
Membre :        2       SANI
-------------------------------------------------
3       8       5

Tache : 8       COMPTABLE       VNU     COURS
Membre :        5       MALLE
-------------------------------------------------
```

Les différents tests d'acceptation sont aussi valables pour l'ensemble de toutes les autres fonctionnalités qui s'en suivent. Notamment pour la classe membre.

5.5 Affichage de la recherche des assignations par statuts

Après avoir choisi l'option 12 du menu principal, le système affiche ce résultat suivant qui donne pour chaque assignation le nom de l'associé

```
        -------------------------------------
        -------RECHERCHE ASSIGNATION PAR STATUS DES TACHES-----------
        Saisissez le status de la tache :       TERMINER
        -----------------------------------------------------

                Les Taches assignees aux membre ayant le statusTERMINER sont :
1       4       1

**************************************************
        SYLLA
**************************************************
2       6       2

**************************************************
        SANI
**************************************************
3       8       5

**************************************************
        MALLE
**************************************************
```

NB : VEUILLEZ ENREGISTRER LES OPÉRATIONS AVANT LA SORTIE DE L'EXÉCUTION DU PROGRAMME POUR DES MESURES DE SAUVEGARDE

CETTE COMMANDE FAIT EN TAPANT COMME CHOIX DU MENU PRINCIPAL LE NOMBRE 0

6. CONCLUSION

- Ainsi, c'est la description de l'application créée, un petit gestionnaire de taches d'où l'objet du TP .

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.

Ce travail n'est pas à sa plus haute perfection. Celle ci est à sa toute première version et nous comptons y améliorer pour une gestion d'autres taches dans un travail d'équipe.

Cependant, l'apport de ce travail a été d'une importance très considérable, en effet, il permet : de suivre une méthodologie de travail bien étudié, d'approfondir des connaissances sur les méthodes de développement des applications.

CODES SOURCES EN ANNEXE

```
package com.tp1_java;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import java.util.Scanner;
import javax.swing.plaf.basic.BasicBorders.MenuBarBorder;
import java.io.File;
import java.io.*;

public class Gestionnaire_taches {
	// LA DECLARATION DES LISTES OU TABLEAU DE LISTES CHAINEE

	static ArrayList<Assignes> ListAssigne = new ArrayList<Assignes>();
	static ArrayList<Membres> ListMembre = new ArrayList<Membres>();
	static ArrayList<Taches> Listache = new ArrayList<Taches>();
	static Scanner sc = new Scanner(System.in);

	// LA FONCTION MENU PRINCIPAL

	public static void menuPrincipal() {
	System.out.println("\t"+"\t"+"|
**************************************************************************
************************|");
		System.out.println("\t"+"\t"+"|**
MENU DE LA GESTION DES MEMBRES                                    **|");
		System.out.println("\t"+"\t"+"|
**************************************************************************
************************|");
		System.out.println("\t"+"\t"+"|**
**|");
		System.out.println("\t"+"\t"+"|**                CHOISISSEZ UNE
OPERATION  OU 0 POUR QUITTER ET ENREGISTRER                       **|");
```

```
            System.out.println("\t"+"\t"+"|**
**|");
            System.out.println("\t"+"\t"+"|
**************************************************************************
************************|");
            System.out.println("\t"+"\t"+"|**                                    |
**|");
            System.out.println("\t"+"\t"+"|**        1 - CREER UNE MEMBRE        |
**|");
            System.out.println("\t"+"\t"+"|**                                    |
**|");
            System.out.println("\t"+"\t"+"|**        2 - MODIFIER UN MEMBRE      |
**|");
            System.out.println("\t"+"\t"+"|**                                    |
**|");
            System.out.println("\t"+"\t"+"|**        3 - AFFICHER LES MEMBRES    |
9 - CREER UNE ASSIGNATION                                **|");
            System.out.println("\t"+"\t"+"|**                                    |
**|");
            System.out.println("\t"+"\t"+"|**        4 - SUPPRIMER UN MEMBRE     |
10 - AFFICHER LES ASSIGNATIONS                           **|");
            System.out.println("\t"+"\t"+"|**                                    |
**|");
            System.out.println("\t"+"\t"+"|**********************************
11 - RECHERCHE DES TACHES ASSIGNEES PAR ID DU MEMBRE     **|");
            System.out.println("\t"+"\t"+"|**        5 - CREER UNE TACHE         |
**|");
            System.out.println("\t"+"\t"+"|**                                    |
12 - RECHERCHE DES TACHES PAR LEURS STATUS ET NOM DE L'ASSIGNE **|");
            System.out.println("\t"+"\t"+"|**        6 - MODIFIER UNE TACHE      |
**|");
            System.out.println("\t"+"\t"+"|**                                    |
0 - QUITTER ET SAUVEGARDER LES LISTES                    **|");
            System.out.println("\t"+"\t"+"|**        7 - AFFICHER LES TACHES     |
**|");
            System.out.println("\t"+"\t"+"|**                                    |
**|");
            System.out.println("\t"+"\t"+"|**        8 - SUPPRIMER UNE TACHE     |
**|");
            System.out.println("\t"+"\t"+"|**                                    |
**|");
            System.out.println("\t"+"\t"+"|
**************************************************************************
************************|");
            System.out.println("  ");
            System.out.print("\t"+" DANS LE MENU DES MEMBRES FAITES VOTRE
CHOIX :     ");

      }

      // LA PARTIE DES FONCTIONS DE LA CLASSE MEMBRE

      // LA FONCTION DE RECHERCHE DES MEMBRES

      public static Membres recherche(String Nom) {
            Membres resultats = null;
            for (Membres mbre : ListMembre) {
                  if (mbre.getNom_membre().equalsIgnoreCase(Nom)) {
                        resultats = mbre;
                        break;
                  }
            }
```

```
		return resultats;
	}


	// RECHERCHE ID MEMBRES

	public static Membres rechercheIDMembre(int ID) {
		Membres resultats = null;
		for (Membres mbre : ListMembre) {
			if (mbre.getID_membre()== ID) {
				resultats = mbre;
				break;
			}
		}
		return resultats;
	}

	// RECHERCHE ID TACHES

	public static Taches rechercheIDTache(int ID) {
		Taches resultats = null;
		for (Taches mbre : Listache) {
			if (mbre.getID_tache()== ID) {
				resultats = mbre;
				break;
			}
		}
		return resultats;
	}

	// LA FONCTION D'AJOUT ET DE CREATION DES MEMBRES

	public static void addMembres(Membres m) {

		Membres membreNom = recherche(m.getNom_membre());
		Membres membreID = rechercheIDMembre(m.getID_membre());
		if (membreNom != null || membreID!=null) {
			System.out.println("\t"+ "\t" +"Desole ID ou Nom saisi existe
deja"+ "\t");
		} else {

			ListMembre.add(m);
		}
	}

	// LA FONCTION DE SUPPRESSION DES MEMBRES DANS LA LISTE

	public static void delMembres(String Nom) {
		Membres membre2 = recherche(Nom);

		if (membre2 != null) {
			int index = ListMembre.indexOf(membre2);
			ListMembre.remove(index);

		} else {
			System.out.println("\t" + " Desole l'element saisie n'existe
pas dans la liste");
		}
	}

	// LA FONCTION DE MODIFICATION DES MEMBRES
```

```
	public static void editMembre(String Anom, String Nnom) {
		Membres membre3 = recherche(Anom);

		if (membre3 != null) {
			int index = ListMembre.indexOf(membre3);

			ListMembre.remove(index);
			Membres creeMembre = new Membres(index, Nnom);
			ListMembre.add(creeMembre);

		} else {
			System.out.println("\t" + "Desole Ce nom N'existe pas");

		}

	}

	// LA FONCTION AFFICHAGE DES MEMBRES

	public static void affichageMembres() {
		if (!ListMembre.isEmpty()) {
			for (Membres tache : ListMembre) {
				System.out.println("\t"+"|"+tache.getID_membre() + "\t"
+ "|" +tache.getNom_membre());
				System.out.println("\n");
			}
		} else {
			System.out.println("\t" +"Desole La liste des taches est
vide");
		}

	}

	// LA PARTIE DES FONCTIONS DE LA CLASSE TACHE

	// LA FONCTION DE RECHERCHE DES TACHES
	public static Taches rechercheTache(String Nom) {
		Taches resultatache = null;
		for (Taches tach : Listache) {
			if (tach.getNom_tache().equalsIgnoreCase(Nom)) {
				resultatache = tach;
				break;
			}
		}
		return resultatache;
	}

	// LA FONCTION D'AJOUT ET DE CREATION DES TACHES
	 public static void addTaches(Taches t) {

		Taches tache1 = rechercheTache(t.getNom_tache());
		Taches tacheID = rechercheIDTache(t.getID_tache());
		if (tache1 != null || tacheID!=null) {
			System.out.println("\t"+ "\t" +"Desole ID ou Nom saisi existe
deja"+ "\t");

		} else {

			Listache.add(t);
		}
	}
```

```
	// LA FONCTION DE SUPPRESSION DES TACHES DANS LA LISTE
	public static void delTaches(String Nom) {
		Taches tache2 = rechercheTache(Nom);

		if (tache2 != null) {
			int index = Listache.indexOf(tache2);
			Listache.remove(index);

		} else {
			System.out.println("\t" +"Desolé l'element saisie n'existe pas
dans la liste");
		}
	}

	// LA FONCTION DE MODIFICATION DES TACHES
	public static void editTache(String AnomTache,String NnomTache, String
NDesc, String NStatus) {
		Taches tache3 = rechercheTache(AnomTache);
		if (tache3 != null) {
			int index1 = Listache.indexOf(tache3);
			Listache.remove(index1);
			Taches creeTache = new Taches(index1,
NnomTache,NDesc,NStatus);
			Listache.add(creeTache);

		} else {
			System.out.println("\t" +"Desolé Ce nom existe deja");
		}
	}

	// LA FONCTION AFFICHAGE DES TACHES
	public static void affichageTaches() {

		if (!Listache.isEmpty()) {
			for (Taches tache : Listache) {
				System.out.println("\t"+"|"+tache.getID_tache() +
"\t"+"|" + tache.getNom_tache() + "\t"+"|" + tache.getDesc() + "\t"+"|"
					+ tache.getStatus());
				System.out.println("\n");
			}
		} else {
			System.out.println("Desolé La liste des taches est vide");
		}

	}
// LA FONCTION DE RECHERCHE DES ASSIGNATIONS
	public static Assignes recherchAssignation(int idmembres, int idtaches) {
		Assignes resultats = null;
		for (Assignes ass : ListAssigne) {
			if (ass.getID_Membre() == idmembres && ass.getID_Tache() ==
idtaches) {
				resultats = ass;
				break;
			}
		}
		return resultats;
	}

	// LA FONCTION DE CREATION D'UNE ASSIGNATION
	public static void addAssignes(Assignes ass1) {
```

```
            Assignes assigne1 = recherchAssignation(ass1.getID_Membre(),
ass1.getID_Tache());

            if (assigne1 != null) {
                  System.out.println("\t" +" Desolé Cette Assignation existe
déja dans la liste Essayer une autre !!!!");

            } else {

                  ListAssigne.add(ass1);
            }
      }

      // LA SUPPRESSION SUMULTANNEE DES TACHES ET MEMBRES DANS ASSIGNATION

      // SUPPRESION DE L'ID_TACHE APRES SA SUPPRESSION DANS LA LISTE TACHE
      public static void supprimerIDtacheDansAssignation(int ID_tache) {
            for (int i = 0; i < ListAssigne.size(); i++) {
                  Assignes assignation = ListAssigne.get(i);
                  if (assignation.getID_Tache() == ID_tache) {
                        ListAssigne.remove(assignation);
                  }
            }
      }

      // SUPPRESION DE L'ID_MEMBRE APRES SA SUPPRESSION DANS LA LISTE MEMBRE
      public static void supprimerIDMembreDansAssignation(int ID_membre) {

            for (int i = 0; i < ListAssigne.size(); i++) {
                  Assignes assignation = ListAssigne.get(i);
                  if (assignation.getID_Membre() == ID_membre) {
                        ListAssigne.remove(assignation);
                  }
            }

      }

      // AFFICHAGE DE TOUTES LES TACHES ASSIGNEES
      public static void affichageDesTachesAssignees() {

            for (Assignes a : ListAssigne) {
                  System.out.println(a.toString() + "\n");

                  System.out.print("Tache :\t");

                  for (Taches tache : Listache) {
                        if (a.getID_Tache() == tache.getID_tache()) {
                              System.out.println(tache.getID_tache() + "\t" +
tache.getNom_tache() + "\t" + tache.getDesc() + "\t"
                                          + tache.getStatus());
                              break;
                        }
                  }
                  System.out.print("Membre :\t");
                  // Recherche de l'element membre dans la liste des membres
                  for (Membres membre : ListMembre) {
                        if (a.getID_Membre() == membre.getID_membre()) {
                              System.out.println(membre.getID_membre() + "\t" +
membre.getNom_membre());
                              break;
                        }
                  }
```

```
		System.out.println("-----------------------------------------------");
			}
		}

		// AFFICHAGE DE TOUTES LES TACHES ASSIGNEES EN FONCTOIN DE ID_MEMBRE
		public static void afficherTachesAssigneesParIdMembre(int ID) {
			System.out.println("\t" + "\t" + " Les Taches assignees a ce membre
ayant " +ID+ "sont : ");


System.out.println("*********************************************************");
			for (Assignes ass : ListAssigne) {
				System.out.println(ass.toString() + "\n");
				if (ass.getID_Membre() == ID) {
					for (Taches tach : Listache) {
						if (ass.getID_Tache() == tach.getID_tache()) {
							System.out.println(tach.getID_tache() + "\t"
+ tach.getNom_tache() + "\t" + tach.getDesc()
									+ "\t" + tach.getStatus());
							break;
						}
					}
				}


System.out.println("*********************************************************");
			}
		}

		// AFFICHAGE DE TOUTES LES TACHES ASSIGNEES AUX MEMBRES EN FONCTOIN DU
		// STATUT DE LA TACHE

		public static void afficherTachesAssigneesAuxMembresParStatutTache(String
Statut) {

			System.out.println("\t" + "\t" + " Les Taches assignees aux membre
ayant le status"+Statut+" sont : ");
			for (Assignes ass : ListAssigne) {

				System.out.println(ass.toString() + "\n");



System.out.println("*********************************************************");
				for (Taches tache : Listache) {
					if (ass.getID_Tache() == tache.getID_tache() &&
tache.getStatus() == Statut) {
						System.out.print(tache.getID_tache() + "\t" +
tache.getNom_tache() + "\t" + tache.getDesc());
						break;
					}
				}
				System.out.print("\t");
				for (Membres membre : ListMembre) {
					if (ass.getID_Membre() == membre.getID_membre()) {
						System.out.println(membre.getNom_membre());
						break;
					}
				}
```

```
System.out.println("*****************************************************");
            }
      }


      // LA GESTION DES CHEMINS ABSOLUS VERS LES FICHIERS

      public static String absolutePathMembre() {
              File fill = new File("FichierMembres.txt");
              String absolutePathMembre = fill.getAbsolutePath();
              return absolutePathMembre;
         }

      public static String absolutePathTache() {
        File fille = new File("FichierTaches.txt");
        String absolutePathTache = fille.getAbsolutePath();
        return absolutePathTache;

        }
      public static String absolutePathTacheAssignation() {
        File fie = new File("FichierAssignations.txt");
        String absolutePathTacheAssignation = fie.getAbsolutePath();
        return absolutePathTacheAssignation;

        }

      // STOCTAGE DES LISTES DANS DES FICHIERS

      // CHARGER LA LISTE DES MEMBRES DANS FICHIER


      public static void lireFichierMembre() throws Exception {

            FileInputStream fin = new FileInputStream("absolutePathMembre");
            ObjectInputStream objIn = new ObjectInputStream(fin);
            ListMembre = (ArrayList<Membres>) objIn.readObject();
            objIn.close();
      }

      // CHARGER LE FICHIER DES MEMBRES DANS LA LISTE
      public static void enregistrerMembresDansFichier() throws Exception {
            FileOutputStream fos = new FileOutputStream("absolutePathMembre");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(ListMembre);
            oos.close();
      }

      // CHARGER LA LISTE DES TACHES DANS FICHIER ET RECIPROQUEMENT
      public static void lireFichierTaches() throws Exception {
            FileInputStream fin = new FileInputStream("absolutePathTache");
            ObjectInputStream objIn = new ObjectInputStream(fin);
            Listache = (ArrayList<Taches>) objIn.readObject();
            objIn.close();
      }

      // CHARGER LE FICHIER DES TACHES DANS LA LISTE
      public static void enregistrerTachesDansFichier() throws Exception {
            FileOutputStream fos = new FileOutputStream("absolutePathTache");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(Listache);
```

```
            oos.close();
      }

      // ASSIGNATION
      public static void enregistrerAssignationsDansFichier() throws Exception {
            FileOutputStream fos = new
FileOutputStream("absolutePathTacheAssignation");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(ListAssigne);
            oos.close();
      }

      // CHARGER LA LISTE DES ASSIGNATION DANS UN FICHIER
      public static void lireFichierAssignations() throws Exception {
            FileInputStream fin = new
FileInputStream("absolutePathTacheAssignation");
            ObjectInputStream objIn = new ObjectInputStream(fin);
            ListAssigne = (ArrayList<Assignes>) objIn.readObject();
            objIn.close();
      }

      // SAUVEGARDER
      public static void sauvegarderDansFichiers() {
            try {
                  enregistrerTachesDansFichier();
                  enregistrerMembresDansFichier();
                  enregistrerAssignationsDansFichier();
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }

      //CHARGEMENT DES LISTES
      public static void chargement() {

            try {
                  lireFichierTaches();
                  lireFichierMembre();
                  lireFichierAssignations();
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }

      // Fonction de verification du type de la saisie de l'utilisateur

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

      public static void verifierPlage(int choix, int borneInferieure, int
borneSuperieure) {
            while (choix < borneInferieure || choix > borneSuperieure) {
                  verifierType();
                  System.out.println(
                              "\n\nEntrez un entier entre " + borneInferieure +
" et " + borneSuperieure + " compris\nChoix : ");
                  choix = sc.nextInt();
```

```
            }
      }

      public static void clearScreen() {
          System.out.print("\033[H\033[2J");
          System.out.flush();
         }

/
**********************************************************************************
*************************************************/
/
**********************************************************************************
*************************************************/

      // LA FONCTION PRINCIPALE
public static void main(String[] args) {
      absolutePathMembre();
      absolutePathTache();
      absolutePathTacheAssignation();
      int choix=0;
      int continu;
      chargement();//Initialisation des listes a partir des fichiers
      int sortieMenu = 1;
      while (sortieMenu == 1) {
      menuPrincipal();
      verifierType();
      verifierPlage(choix, 0, 12);
      choix = sc.nextInt();
      //verifierPlage(choix, 0, 12);
                  switch (choix) {
                        case 1:
                              int sortieMembreCreer = 1;
                              while (sortieMembreCreer == 1) {
                                    int idMembre;
                                    String nomMembre;
                                    System.out.println("\n");
                                    System.out.println("\t" +
"---------------------------------------");
                                    System.out.println("\t" + "-------CREATION
D'UN MEMBRE-----------");
                                    System.out.print("\t" + "Saisissez
l'Identifiant : " + "\t");
                                    Scanner sc1 = new Scanner(System.in);
                                    idMembre = sc1.nextInt();
                                    System.out.print("\t" + "Saisissez le Nom :
" + "\t");
                                    Scanner scc = new Scanner(System.in);
                                    nomMembre = scc.next();


System.out.println("\t"+"----------------------------------------------------------
---"+"\n");
                                    Membres creeMembre = new Membres(idMembre,
nomMembre);
                                    addMembres(creeMembre);
                                    affichageMembres();
                                    System.out.print("Voulez vous creer un autre
membre si oui 1 si non retourner sur le menu principal en tapant 0 : " + "\t");
                                    continu = sc1.nextInt();
                                    if (continu == 1) {
                                          sortieMembreCreer = 1;
```

```
                                    } else
                                          sortieMembreCreer=0;
                                          clearScreen();
                                    }
                              break;

                              case 2:
                                    int sortieMembreModifier = 1;
                                    Scanner scn = new Scanner(System.in);
                                    while (sortieMembreModifier == 1) {
                                          int idMembre;
                                          String NouveauNom;
                                          String AncienNom;
                                          System.out.println("\n");
                                          System.out.println("\t" +
"--------------------------------------");
                                          System.out.println("\t" + "-------
MODIFICATION D'UN MEMBRE-----------");
                                          System.out.print("\t" + "Saisissez
l'ancien : " + "\t");
                                          Scanner sccc1 = new Scanner(System.in);
                                          AncienNom = sc.next();
                                          System.out.print("\t" + "Saisissez le
nouveau nom : " + "\t");
                                          Scanner scc1 = new Scanner(System.in);
                                          NouveauNom = sc.next();
                                          editMembre(AncienNom, NouveauNom);
                                          System.out.print("Voulez vous modifier un
autre membre si oui 1 si non retourner sur le menu principal en tapant 0 : " +
"\t");
                                          continu = sc.nextInt();
                                          if (continu == 1) {
                                                sortieMembreModifier = 1;
                                          } else
                                                sortieMembreModifier=0;
                                                clearScreen();
                                    }
                              break;

                              case 3:
                                    int sortieAffichageMembnres = 1;
                                    while (sortieAffichageMembnres == 1) {

      System.out.println("\t"+"-----------------------------------------");
                                    affichageMembres();

      System.out.println("\t"+"-----------------------------------------");

                                    System.out.print("Voulez vous reafficher a nouveau
si oui 1 si non retourner sur le menu principal en tapant 0 : " + "\t");
                                    continu = sc.nextInt();
                                    if (continu == 1) {
                                          sortieAffichageMembnres = 1;
                                    } else
                                          sortieAffichageMembnres=0;
                                          clearScreen();
                                    }
                                    break;

                              case 4:
                                    int sortieSuppMembre = 1;
                                    while (sortieSuppMembre == 1) {
```

```
                                    Membres m=null;
                                    String nomMembre;
                                    System.out.println("\n");
                                    System.out.println("\t" +
"-------------------------------------");
                                    System.out.println("\t" + "-------
SUPPRESSION D'UN MEMBRE-----------");
                                    System.out.print("\t" + "Saisissez le nom du
membre devant etre supprime : " + "\t");
                                    Scanner sc3 = new Scanner(System.in);
                                    nomMembre = sc3.next();


System.out.println("\t"+"-----------------------------------------------------
---"+"\n");
                                  //Membres suppMembre = new
Membres(m.getID_membre(), nomMembre);
                                  delMembres(nomMembre);

//supprimerIDMembreDansAssignation(m.getID_membre());
                                  affichageMembres();
                                    System.out.print("Voulez vous supprimer un
autre membre si oui 1 si non retourner sur le menu principal en tapant 0 : " +
"\t");
                                    continu = sc.nextInt();
                                    if (continu == 1) {
                                          sortieSuppMembre = 1;
                                    } else
                                          sortieSuppMembre=0;
                                          clearScreen();
                              }
                        break;

                        case 5:
                              int sortieTachesCreer = 1;
                              while (sortieTachesCreer == 1) {

                                    int idTache;
                                    String nomTache;
                                    String Description;
                                    String Status;
                                    System.out.println("\n");
                                    System.out.println("\t" +
"-------------------------------------");
                                    System.out.println("\t" + "-------CREATION
D'UNE TACHE-----------");
                                    System.out.print("\t" + "Saisissez
l'Identifiant : " + "\t");
                                    Scanner sc5 = new Scanner(System.in);
                                    idTache = sc5.nextInt();
                                    System.out.print("\t" + "Saisissez le Nom de
la tache : " + "\t");
                                    Scanner sc6 = new Scanner(System.in);
                                    nomTache = sc6.next();
                                    System.out.print("\t" + "Saisissez le
Description de la tache : " + "\t");
                                    Scanner sc7 = new Scanner(System.in);
                                    Description = sc7.next();
                                    System.out.print("\t" + "Saisissez le Status
de la tache : " + "\t");
                                    Scanner sc8 = new Scanner(System.in);
                                    Status = sc8.next();
```

```
                                    Taches creeTache = new Taches(idTache,
nomTache,Description, Status);


System.out.println("\t"+"-----------------------------------------------------
---"+"\n");
                                    addTaches(creeTache);
                                    affichageTaches();
                                    System.out.print("Voulez vous creer une
nouvelle tache si oui 1 si non retourner sur le menu principal en tapant 0 : " +
"\t");
                                    continu = sc.nextInt();
                                    if (continu == 1) {
                                          sortieTachesCreer = 1;
                                    } else
                                          sortieTachesCreer=0;
                                          clearScreen();
                              }
                              break;

                        case 6:
                              int sortieTacheModifier = 1;
                              while (sortieTacheModifier == 1) {
                                    int idMembre;
                                    String AncienNom;
                                    String NouveauNom;
                                    String NDesc;
                                    String NStatus;
                                    System.out.println("\n");
                                    System.out.println("\t" +
"--------------------------------------");
                                    System.out.println("\t" + "-------
MODIFICATION D'UNE TACHE-----------");
                                    System.out.print("\t" + "Saisissez l'ancien
de la tache : " + "\t");
                                    Scanner sccc1 = new Scanner(System.in);
                                    AncienNom = sc.next();
                                    System.out.print("\t" + "Saisissez le
nouveau nom de la tache : " + "\t");
                                    Scanner scc1 = new Scanner(System.in);
                                    NouveauNom = sc.next();
                                    System.out.print("\t" + "Saisissez la
nouvelle Description : " + "\t");
                                    Scanner sc1 = new Scanner(System.in);
                                    NDesc = sc.next();
                                    System.out.print("\t" + "Saisissez le
nouveau Status : " + "\t");
                                    Scanner s1 = new Scanner(System.in);
                                    NStatus = sc.next();
                                    editTache(AncienNom, NouveauNom, NDesc,
NStatus);
                                    System.out.print("Voulez vous modifier une
autre tache si oui 1 si non retourner sur le menu principal en tapant 0 : " +
"\t");
                                    continu = sc.nextInt();
                                    if (continu == 1) {
                                          sortieTacheModifier = 1;
                                    } else
                                          sortieTacheModifier=0;
                                          clearScreen();
                              }
                        break;
```

```
                        case 7:
                              int sortieAffichageTaches = 1;
                              while (sortieAffichageTaches == 1) {

      System.out.println("\t"+"-----------------------------------------");
                              affichageTaches();

      System.out.println("\t"+"-----------------------------------------");
                              System.out.print("Voulez vous reafficher a nouveau
les taches si oui 1 si non retourner sur le menu principal en tapant 0 : " +
"\t");
                              continu = sc.nextInt();
                              if (continu == 1) {
                                    sortieAffichageTaches = 1;
                              } else
                                    sortieAffichageTaches=0;
                                    clearScreen();
                           }
                        break;

                        case 8:
                              int sortieSuppTache = 1;
                              while (sortieSuppTache == 1) {
                                    Taches m=null;
                                    String nomTache;
                                    System.out.println("\n");
                                    System.out.println("\t" +
"-------------------------------------");
                                    System.out.println("\t" + "-------
SUPPRESSION D'UNE TACHE-----------");
                                    System.out.print("\t" + "Saisissez le nom du
tache devant etre supprime : " + "\t");
                                    Scanner sc3 = new Scanner(System.in);
                                    nomTache = sc3.next();


System.out.println("\t"+"-----------------------------------------------------
---"+"\n");
                                   //Taches suppTache = new
Taches(m.getID_tache(),m.getNom_tache(),m.getDesc(),m.getStatus());
                                    delTaches(nomTache);
                                   //
supprimerIDMembreDansAssignation(suppTache.getID_tache());
                                    affichageTaches();
                                      System.out.print("Voulez vous supprimer une
autre tache si oui 1 si non retourner sur le menu principal en tapant 0 : " +
"\t");
                                    continu = sc.nextInt();
                                    if (continu == 1) {
                                          sortieSuppTache = 1;
                                    } else
                                          sortieSuppTache=0;
                                          clearScreen();
                              }


                        break;


                        case 9:
```

```
                                    System.out.println("\t" + "Voici la liste des
Membres");
                                    affichageMembres();
                                    System.out.println("\t" + "Voici la liste des
Taches");
                                    affichageTaches();
                                    int sortieAssignationCreer = 1;
                                    while (sortieAssignationCreer == 1) {
                                          int idAssignes, idMembre,idTache;
                                          System.out.println("\n");
                                          System.out.println();
                                          System.out.println("\t" + "-------CREATION
DES ASSIGNATIONS-----------");
                                          System.out.print("\t" + "Saisissez
l'Identifiant de l'assignation : " + "\t");
                                          Scanner sc10 = new Scanner(System.in);
                                          idAssignes = sc10.nextInt();
                                          System.out.print("\t" + "Saisissez l'ID du
membre devant etre affecte a une tache : " + "\t");
                                          Scanner sc11 = new Scanner(System.in);
                                          idMembre = sc11.nextInt();
                                          System.out.print("\t" + "Saisissez l'ID de
la tache a assignee au membre : " + "\t");
                                          Scanner sc12 = new Scanner(System.in);
                                          idTache = sc12.nextInt();


System.out.println("\t"+"-----------------------------------------------------------
---"+"\n");
                                          Assignes creeAssignes = new
Assignes(idAssignes,idMembre, idTache);
                                          addAssignes(creeAssignes);
                                          affichageDesTachesAssignees();
                                          System.out.print("Voulez vous creer une
nouvelle assignation si oui 1 si non retourner sur le menu principal en tapant 0
: " + "\t");
                                          continu = sc.nextInt();
                                          if (continu == 1) {
                                                sortieAssignationCreer = 1;
                                          } else
                                                sortieAssignationCreer=0;

                                                clearScreen();
                                    }
                              break;

                              case 10:
                                    int sortieAffichageTachesAssignees = 1;
                                    while (sortieAffichageTachesAssignees == 1) {
                                    affichageDesTachesAssignees();
                                    System.out.print("Voulez vous reafficher la liste
des assignations si oui 1 si non retourner sur le menu principal en tapant 0 : "
+ "\t");
                                    continu = sc.nextInt();
                                    if (continu == 1) {
                                          sortieAffichageTachesAssignees = 1;
                                    } else
                                          sortieAffichageTachesAssignees=0;

                                          clearScreen();
                                 }
                              break;
```

```
                        case 11:
                              int sortieRechercheAssignesID = 1;
                              while (sortieRechercheAssignesID == 1) {
                                    int ID;
                                    System.out.println("\n");
                                    System.out.println("\t" +
"-------------------------------------");
                                    System.out.println("\t" + "-------RECHERCHE
ASSIGNATION PAR ID DES MEMBRES-----------");
                                    System.out.print("\t" + "Saisissez
l'Identifiant : " + "\t");
                                    Scanner scc1 = new Scanner(System.in);
                                    ID = scc1.nextInt();


System.out.println("\t"+"-----------------------------------------------------------
---"+"\n");
                                    afficherTachesAssigneesParIdMembre(ID);


System.out.println("\t"+"-----------------------------------------------------------
---"+"\n");
                                    System.out.print("Voulez vous effectuer une
autre recherher si oui 1 si non retourner sur le menu principal en tapant 0 : "
+ "\t");
                                    continu = sc.nextInt();
                                    if (continu == 1) {
                                          sortieRechercheAssignesID = 1;
                                    } else
                                          sortieRechercheAssignesID=0;

                                          clearScreen();
                              }
                        break;

                        case 12:
                              int sortieRechercheAssignesStatus = 1;
                              while (sortieRechercheAssignesStatus == 1) {
                                    String STATUS;
                                    System.out.println("\n");
                                    System.out.println("\t" +
"-------------------------------------");
                                    System.out.println("\t" + "-------RECHERCHE
ASSIGNATION PAR STATUS DES TACHES-----------");
                                    System.out.print("\t" + "Saisissez le status
de la tache : " + "\t");
                                    Scanner sccc1 = new Scanner(System.in);
                                    STATUS = sccc1.next();


System.out.println("\t"+"-----------------------------------------------------------
---"+"\n");

      afficherTachesAssigneesAuxMembresParStatutTache(STATUS);


System.out.println("\t"+"-----------------------------------------------------------
---"+"\n");
                                    System.out.print("Voulez vous effectuer une
autre recherher si oui 1 si non retourner sur le menu principal en tapant 0 : "
+ "\t");
```

```
                                    continu = sc.nextInt();
                                    if (continu == 1) {
                                          sortieRechercheAssignesStatus = 1;
                                    } else
                                          sortieRechercheAssignesStatus=0;

                                          clearScreen();
                                    }

                              break;

                              case 0:
                                    sauvegarderDansFichiers();
                                    System.exit(0);
                              break;

                                    default:
                                    break;
                              }
                        }
            choix = sc.nextInt();
            if (choix == 1) {
                  sortieMenu = 1;
            } else
                  sortieMenu=0;
            }
}


//LA CLASSE DES ASSIGNATIONS


package com.tp1_java;

import java.io.Serializable;

public class Assignes implements Serializable {
      /**
       *
       */
      private static final long serialVersionUID = 1L;
      int id_assigne;
      int ID_Membre;
      int ID_Tache;

      // LES CONSTRUCTEURS

      public Assignes() {
            super();
      }

      public Assignes(int id_assigne, int ID_Membre, int ID_Tache) {
            super();
            this.id_assigne = id_assigne;
            this.ID_Membre = ID_Membre;
            this.ID_Tache = ID_Tache;
      }

      public String toString(){
            return this.getId_assigne() + "\t" + this.getID_Tache()+
                        "\t"+this.getID_Membre();
```

```
	}

	public int getId_assigne() {
		return id_assigne;
	}

	public void setId_assigne(int id_assigne) {
		this.id_assigne = id_assigne;
	}

	public int getID_Membre() {
		return ID_Membre;
	}

	public void setID_Membre(int ID_Membre) {
		this.ID_Membre = ID_Membre;
	}

	public int getID_Tache() {
		return ID_Tache;
	}

	public void setID_Tache(int ID_Tache) {
		this.ID_Tache = ID_Tache;
	}

}

//LA CLASSE DES MEMBRES

package com.tp1_java;

import java.io.Serializable;

//LES ATTRIBUTS

public class Membres implements Serializable {
	/**
	 *
	 */
	private static final long serialVersionUID = 1L;
	private int ID_membre;
	private String Nom_membre;

	// LES CONSTRUCTEURS

	public Membres() {

	}

	public String toString() {
		return this.getID_membre() + "\n" + this.getNom_membre();
	}

	public Membres(int ID_membre, String Nom_membre) {

		this.ID_membre = ID_membre;
		this.Nom_membre = Nom_membre;
	}

	// LES GETHERS ET LES SETHERS
```

```java
	public int getID_membre() {
		return ID_membre;
	}

	public void setID_membre(int ID_membre) {
		this.ID_membre = ID_membre;
	}

	public String getNom_membre() {
		return Nom_membre;
	}

	public void setNom_membre(String Nom_membre) {
		this.Nom_membre = Nom_membre;
	}

}

//LA CLASSE DES TACHES

package com.tp1_java;

import java.io.Serializable;

////LES ATTRIBUTS

public class Taches implements Serializable{
/**
	 *
	 */
private static final long serialVersionUID = 1L;
private int ID_tache;
private String Nom_tache;
private String Desc;
private String Status;

// LES CONSTRUCTEURS

public Taches() {

}

public Taches(int ID_tache, String Nom_tache, String Desc, String Status) {

	this.ID_tache = ID_tache;
	this.Nom_tache = Nom_tache;
	this.Desc = Desc;
	this.Status = Status;
}

public String toString(){
	return this.getID_tache() +"\t"+this.getNom_tache()+"\t"+this.getDesc()
+"\t"+this.getStatus();
}

// LES GETHERS ET LES SETHERS

public int getID_tache() {
	return ID_tache;
}

public void setID_tache(int ID_tache) {
```

```
	this.ID_tache = ID_tache;
}

public String getNom_tache() {
	return Nom_tache;
}

public void setNom_tache(String Nom_tache) {
	this.Nom_tache = Nom_tache;
}

public String getDesc() {
	return Desc;
}

public void setDesc(String Desc) {
	this.Desc = Desc;
}

public String getStatus() {
	return Status;
}

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

}

//LA CLASSE DES TACHES

package com.tp1_java;

import java.io.Serializable;

////LES ATTRIBUTS

public class Taches implements Serializable{
/**
	 *
	 */
private static final long serialVersionUID = 1L;
private int ID_tache;
private String Nom_tache;
private String Desc;
private String Status;

// LES CONSTRUCTEURS

public Taches() {

}

public Taches(int ID_tache, String Nom_tache, String Desc, String Status) {

	this.ID_tache = ID_tache;
	this.Nom_tache = Nom_tache;
	this.Desc = Desc;
	this.Status = Status;
}

public String toString(){
```

```java
	return this.getID_tache() +"\t"+this.getNom_tache()+"\t"+this.getDesc()
+"\t"+this.getStatus();
}

// LES GETHERS ET LES SETHERS

public int getID_tache() {
	return ID_tache;
}

public void setID_tache(int ID_tache) {
	this.ID_tache = ID_tache;
}

public String getNom_tache() {
	return Nom_tache;
}

public void setNom_tache(String Nom_tache) {
	this.Nom_tache = Nom_tache;
}

public String getDesc() {
	return Desc;
}

public void setDesc(String Desc) {
	this.Desc = Desc;
}

public String getStatus() {
	return Status;
}

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

}

//LES TESTS JUNIT

package com.tp1_java;

import static org.junit.Assert.*;

import org.junit.Test;

public class rechercheMembreTestJunit {

	@Test
	public void test() {
		Gestionnaire_taches Test = new Gestionnaire_taches();
		Membres output = Test.recherche("SYLLA");
		assertEquals(1,"SYLLA",output);
	}
	private void assertEquals(int i, String string, Membres output) {
		// TODO Auto-generated method stub

	}
}
```

```
package com.tp1_java;

import static org.junit.Assert.*;

import org.junit.Test;

public class rechercheTacheTestJunit {

	@Test
	public void test() {
Gestionnaire_taches Test = new Gestionnaire_taches();

		Taches output1 = Test.rechercheTache("VENDEUR");

		assertEquals(13,"NOBBY",output1);
	}

	private void assertEquals(int i, String string, Taches output1) {
		// TODO Auto-generated method stub

	}



}

// TEST POUR TOUS LES TESTS
```

```
package com.tp1_java;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)
@SuiteClasses({ JunitTestGestionTache.class, rechercheMembreTestJunit.class,
rechercheTacheTestJunit.class })
public class AllTests {

}
```