



Université National du Vietnam

-----  
Institut Francophone International



# Gestionnaire de Tache en Java

GENIE LOGICIEL

## Rapport du TP1

Rédigé par : ABOUBCAR DJIBO Maman Sani

Sous la Supervision de : Dr. HO Tuong Vinh

Hanoi, Août 2017

## Sommaire

Introduction .....	2
I) Les Exigences du TP.....	2
1) Les exigences fonctionnelles et diagramme de cas d'utilisation .....	2
2) Les exigences non fonctionnelles.....	3
II) La Conception.....	4
1) Diagramme de classe .....	4
2) Diagramme de Séquence .....	5
III) Implémentation.....	6
1) Choix de la plate-forme .....	6
2) Choix de la stratégie d'implémentation et spécification de l'application .....	6
IV) Les Tests.....	7
1) Le Test Unitaire avec JUnit.....	7
2) Le Test d'acceptation .....	8
Conclusion .....	17
Annexe .....	18
References .....	42

## **Introduction**

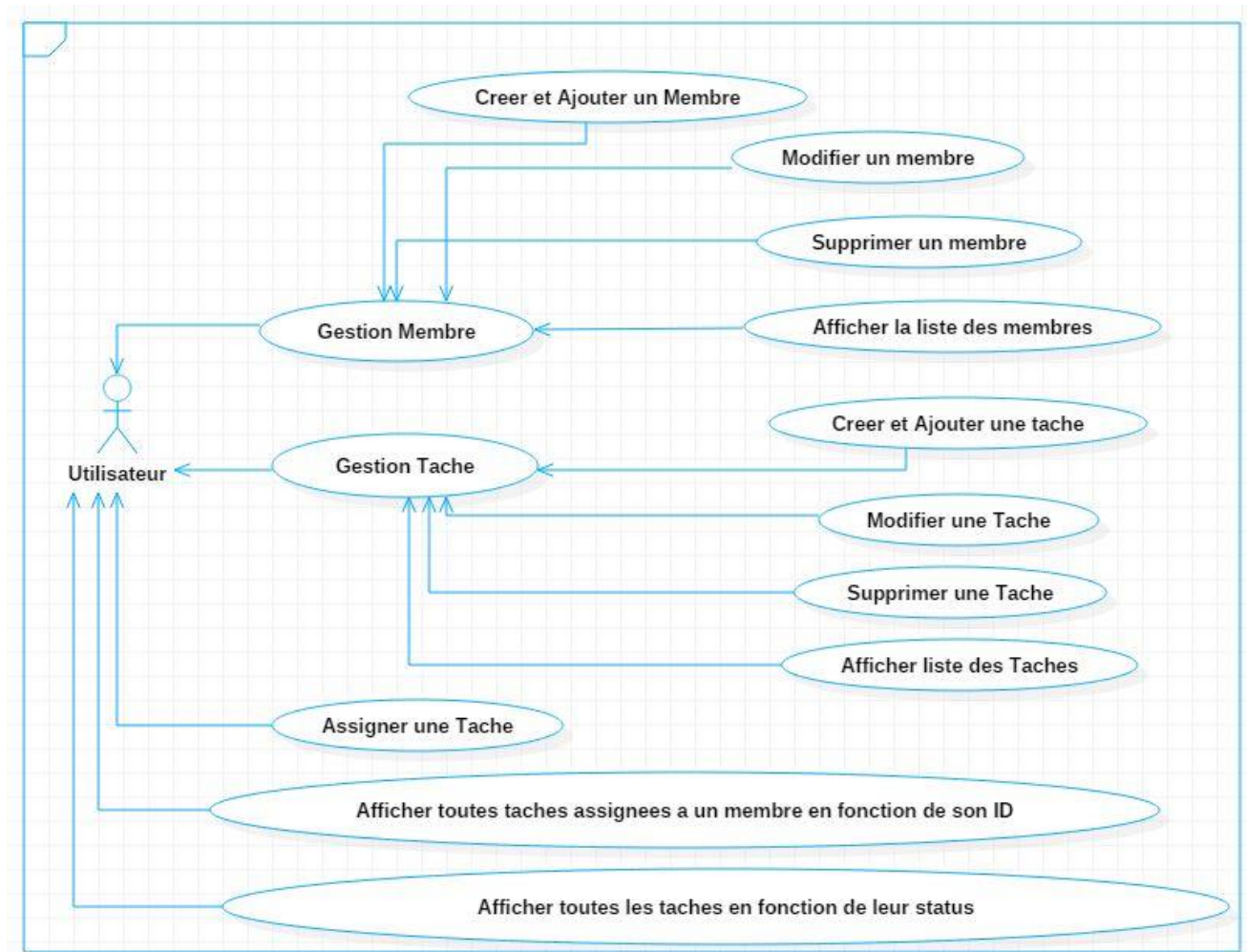
Dans le cadre du premier TP du module Génie-Logiciel Avancée il nous été demande de développer une application de Gestion de Tache pour une équipe de travail. Ce TP a pour but de nous permettre non seulement de nous rappeler les concepts de la modélisation avec UML, de la Programmation Orientée Objet avec Java, mais aussi de nous familiariser à un environnement de développement intégré (IDE) libre comme ECLIPSE et de la plate-forme de gestion de code en ligne Git Hub. Ce présent rapport rend compte du travail que nous avons réalisé. L'application issue de ce travail permet à un utilisateur de créer, ajoute, modifier et supprimer un membre tout comme une tache. Il permet aussi d'assigner une tache a un membre, de lister toutes les taches assignées a un membre en fonction de son identifiant et aussi de lister et afficher toutes les taches en fonction de leurs statu et du nom du membre auquel elle est assignée. Enfin elle permet à l'utilisateur d'enregistrer les opérations qu'il a effectuées.

Nous présenterons, dans la suite de ce rapport, les exigences fonctionnelles et non fonctionnelles de notre application, la conception(les différents diagrammes), l'implémentation, le test d'acceptation, le test unitaire (avec JUnit) et en annexe le code source de notre application.

### **I) Les Exigences du TP**

#### **1) Les exigences fonctionnelles et diagramme de cas d'utilisation**

Dans cette section nous allons utiliser le diagramme de cas d'utilisation (Use Case) pour donner une vision globale du comportement fonctionnel de notre application. Ces sont les diagrammes utilises pour exprimer les besoins des utilisateurs d'un système informatique. Pour le cas de notre application nous en avons un seul qui est représenté ci-dessous :



**Figure 1** : Use Case Gestionnaire de tache

## 2) Les exigences non fonctionnelles

Nous présentons dans le tableau ci-dessous les exigences non-fonctionnelles de notre application

Nom de l'exigence	Description	Exigences fonctionnelles requises
<b>Robustesse</b>	Le système doit être disponible durant toutes les opérations et même dans les cas les plus extrêmes.	Toutes les exigences fonctionnelles
<b>Intégrité</b>	En cas de défaillance du système, les données ne doivent pas subir un changement	Toutes les exigences fonctionnelles
<b>Facilité d'emploi</b>	Le système doit présenter un environnement très convivial pour l'interaction avec l'utilisateur	Toutes les exigences fonctionnelles

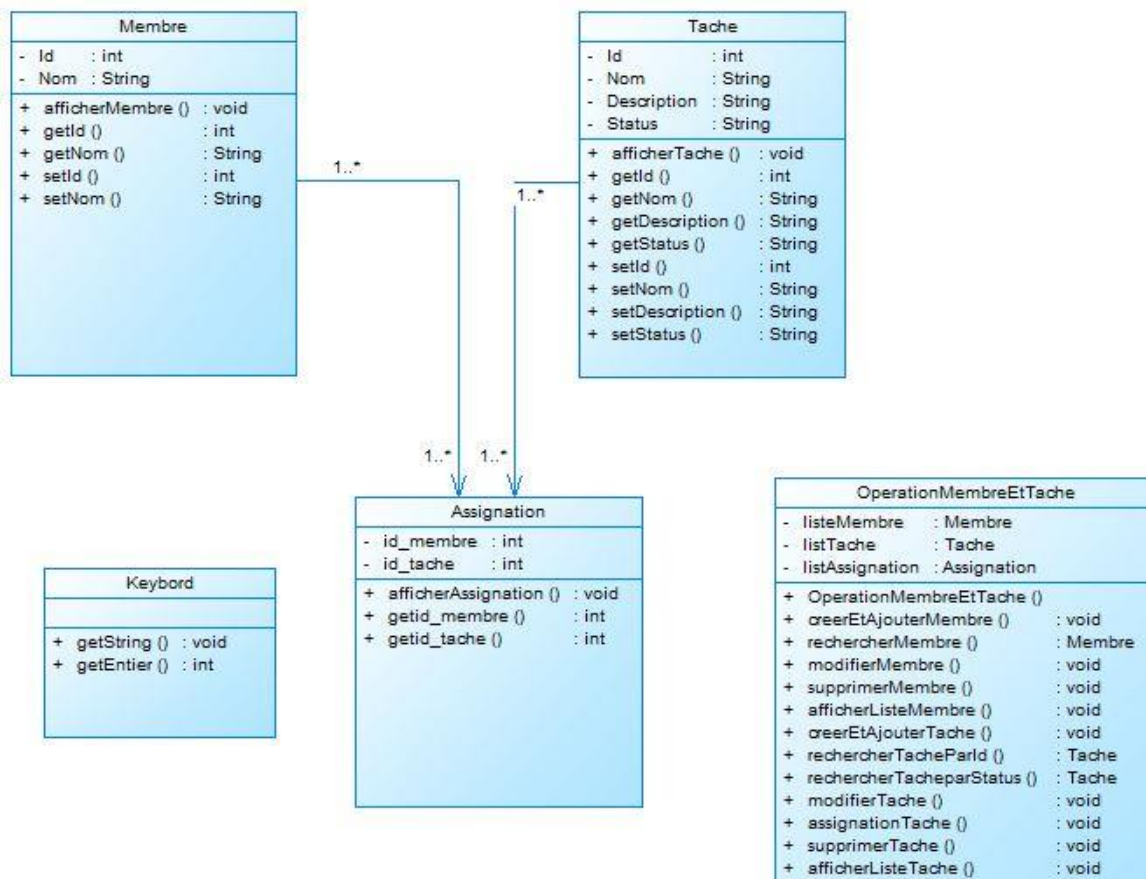
<b>Temps de réponse</b>	Le système ne doit pas mettre un long temps avant de donner une réponse à l'utilisateur	Toutes les exigences fonctionnelles
-------------------------	---	-------------------------------------

Tableau des exigences non-fonctionnelles

## II) La Conception

### 1) Diagramme de classe

Dans cette section nous allons présenter l'aspect conceptuel de notre application à travers un diagramme structurel qu'est celui de classe. Il est l'un des plus importants diagrammes dans le développement Orienté Objet. Ce diagramme représente l'architecture conceptuelle de notre application.



**Figure 2** : Diagramme de classe : Gestionnaire de Tache

Pour notre système nous avons déterminé cinq (05) classes à savoir :

- ❖ La classe Membre ;
- ❖ La classe Tache ;

- ❖ La classe Assignment ;
- ❖ La classe OperationMembreEtTache ;
- ❖ La classe Keyboard ;

Pour une instance de Membre nous avons une ou plusieurs tâches qui peuvent lui être assignées.

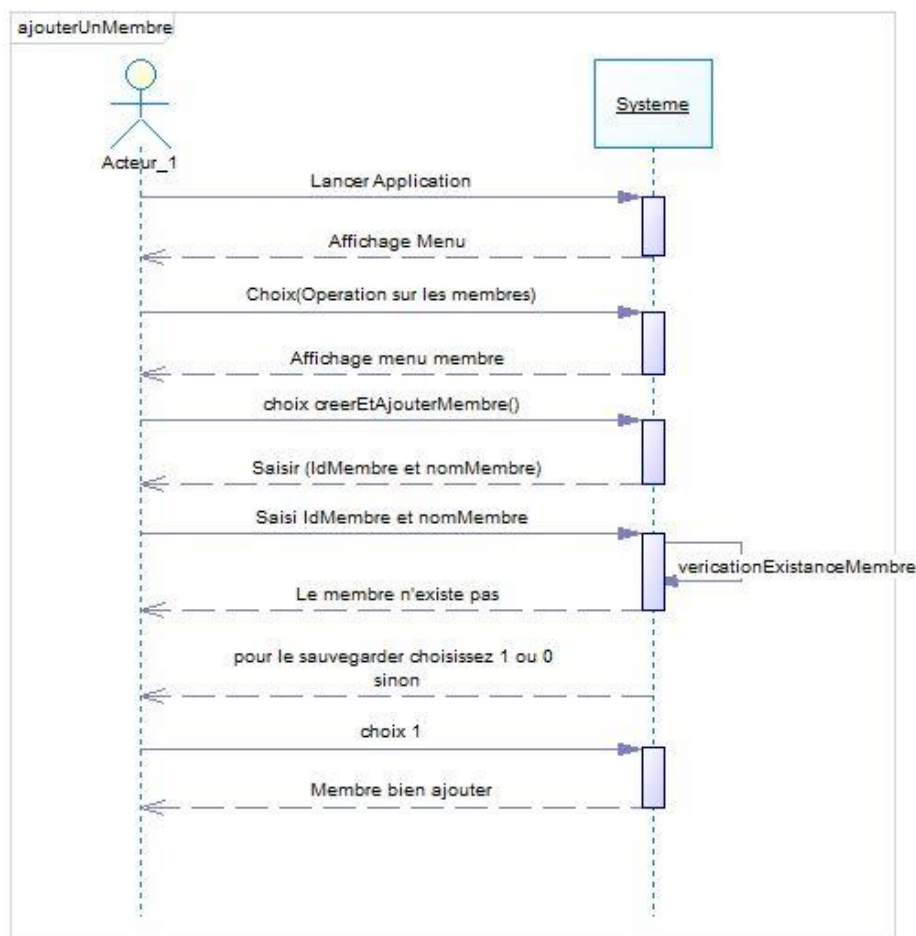
Pour une instance de Tache nous avons une ou plusieurs tâches qui peuvent être assignées à un membre.

La classe OperationMembreEtTache nous permet de gérer toutes les opérations de tâche et membre. La classe Keyboard quant à elle nous permet de gérer les entrées en claviers des utilisateurs.

## 2) Diagramme de Séquence

Le diagramme de séquence représente la succession chronologique des opérations réalisées par un acteur, ce qui fait de lui un des diagrammes d'interaction.

- ❖ Création et ajout d'un membre



### III) Implémentation

#### 1) Choix de la plate-forme

Pour le choix de la plateforme nous nous sommes orientés vers l'IDE **Eclipse** qui est un environnement de développement intégré permettant de développer dans une multitude de langages de programmation à savoir Java, PHP, C, C++, etc.

Ce choix se justifie par le fait qu'il nous a été recommandé, mais aussi par le fait qu'il intègre le Framework de test unitaire **JUnit** et aussi par le fait qu'il est facile d'utilisation et dispose d'une documentation importante pour l'aide.

Pour le langage de programmation nous avons utilisé **Java** qui est un langage multi plateforme, portable, modulable, structuré, orienté objet et dont la maintenance est beaucoup plus facile.

#### 2) Choix de la stratégie d'implémentation et spécification de l'application

Pour l'implémentation de notre nous n'avons pas utilisé les bases de données, nous avons utilisé les vecteurs pour stocker les membres et les tâches qui sont une variante des tableaux dynamiques en Java.

Pour la création des membres et tâches nous avons utilisé les ID (identificateurs) pour différencier les différents membres et les différentes tâches créées. Ceci étant dit nous nous sommes basés sur la logique qu'il peut y avoir deux membres avec le même nom dans une équipe mais des identifiants différents ce qui nous permettra de les différencier. Par contre il ne peut pas y avoir deux (02) tâches de même spécification (i.e. même nom, description, statuts) même si elles ont des IDs différents.

Pour la suppression et la modification nous avons fait en sorte que le programme affiche d'abord la liste des membres ou tâches pour permettre à l'utilisateur de voir les membres ou tâches déjà créés et de choisir l'ID du membre ou tâche à modifier ou à supprimer.

Pour l'assignation d'une tâche nous avons géré le cas de l'affectation de deux (02) mêmes tâches à un même membre.

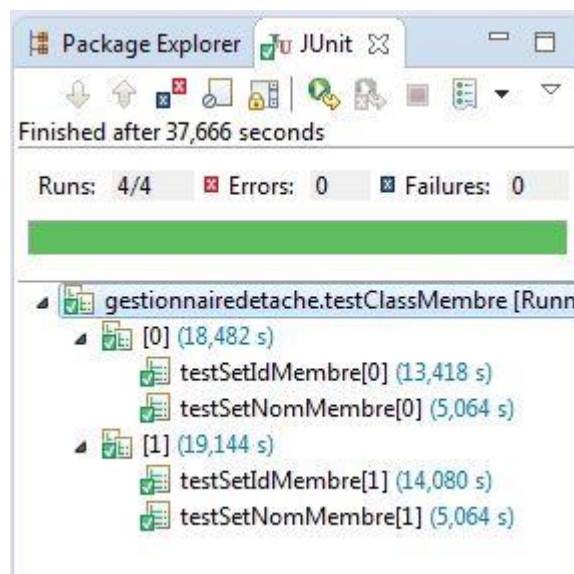
Pour les entrées claviers nous avons géré les erreurs de saisie des utilisateurs.

## IV) Les Tests

### 1) Le Test Unitaire avec JUnit

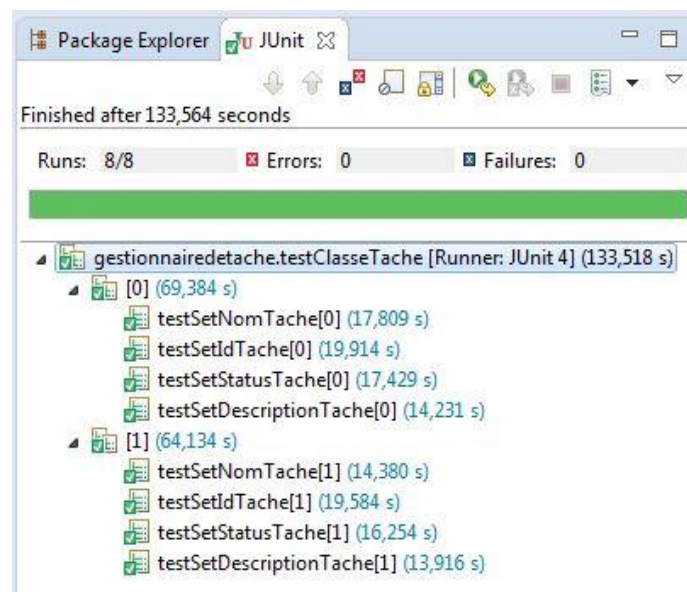
Dans cette section nous allons à travers des captures d'écran illustré le résultat du test unitaire effectué sur la classe Membre et Tache.

- ❖ **Test sur la classe Membre** : ce petit test consiste à tester la classe Membre en assignat un ID et un Nom a un membre. Le résultat issu de ce test est le suivant :



Nous pouvons remarquer à travers la capture ci-dessus que notre test est réussi à 100%.

- ❖ **Test sur la classe Tache** : ce petit test consiste à tester la classe Tache en assignat un ID, un Nom, une Description, et un Statut à une tache. Le résultat issu de ce test est le suivant :





Nous pouvons remarquer à travers la capture ci-dessus que notre test est réussi à 100%.

## 2) Le Test d'acceptation

Dans cette section nous allons à travers des captures d'écran illustré le résultat du test unitaire effectué sur la classe Membre et Tache.

- ❖ **Création et ajout d'un membre** : l'utilisateur choisit l'option « Opération membre » puis il choisit l'option « 1 » créer et jouter un membre ensuite il renseigne les informations de du membre i.e. l'ID et le Nom et choisir 1 pour confirmer l'ajout, ci-dessous les images d'illustrations :

```

MENU PRINCIPAL
-----
1. Operation sur les membres
2. Operation sur les taches
3. Assignation de tache
4. Rechercher les taches assigne a un membre
5. Rechercher les taches en fonction de leur status
6. Sauvegarder Operation d'un Membre
7. Sortir
-----

| OPERATIONS SUR LES MEMBRES |
-----
1. Creer et Ajouter un membre
2. Modifier un Membre
3. Supprimer un membre
4. Afficher la liste des membres
5. Sauvegarder Operation Membre
6. Retours au menu principal
-----
Entrez votre choix : 1

Entrez l'ID du membre a creer : 1
Entrez le Nom du membre a creer : Maman Sani

Voulez-vous enregistrez ce Membre ?

Choisissez 1 pour oui et 0 pour non :1

Le membre a bien ete ajouter
  
```

Figure : Ajout d'un Membre

- ✓ **Affichage de la liste des membres créés** : nous allons afficher la liste du membre que nous venons de créer :

```

-----
| OPERATIONS SUR LES MEMBRES |
-----
1. Creer et Ajouter un membre
2. Modifier un Membre
3. Supprimer un membre
4. Afficher la liste des membres
5. Sauvegarder Operation Membre
6. Retours au menu principal
-----
Entrez votre choix : z
Vous n'avez pas entre un entier
Veuillez entrer un entier : -1
Veuillez entrez un entier positif : 4

Le contenu de la liste est :

Id Membre      Nom Membre
1      Maman Sani

```

**Figure** : Affichage de la liste des Membres

Nous pouvons constater que le membre a bien été créé et ajouter. Aussi nous pouvons remarquer la gestion de la saisie de l'utilisateur.

- ❖ **Création d'une tache** : nous allons créer une tache en suivant le même procédé que celui de la création de membre tout en choisissant l'option Opération sur les taches :

```

-----
| OPERATION SUR LES TACHES |
-----
1. Creer et Ajouter une tache
2. Modifier une tache
3. Supprimer une tache
4. Afficher la liste des taches
5. Retours au menu principal
-----
Entrez votre choix : 1

Entrez l'ID de la tache a creer : 1
Nom de la tache : Maintenance
Description de la tache : maintenance preventif
Status de la tache : nouveau

Voulez-vous Ajouter cette Tache ?

Choisissez 1 pour oui et 0 pour non : 1

La tache a bien ete ajouter !

```

**Figure** : liste des taches

- ✓ **Affichage liste des taches créé** : dans le menu tache l'utilisateur choisit seulement l'option 4.

```

-----
| OPERATION SUR LES TACHES |
-----
1. Creer et Ajouter une tache
2. Modifier une tache
3. Supprimer une tache
4. Afficher la liste des taches
5. Retours au menu principal
-----
Entrez votre choix : 4

```

Id Tache	Nom Tache	Description	Status
1	Maintenance	maintenance preventif	nouveau

**Figure** : liste des taches

- ❖ **Assignation d'une tache** : nous allons assigner au membre que nous créer la tâche que nous venons de créer. Pour ce faire il faut choisir l'option 3 du menu principal et renseigne l'ID du membre et celui de la tache (i.e. IdMembre = 1, idTache = 1) dans cet exemple :

```

MENU PRINCIPAL
-----
1. Operation sur les membres
2. Operation sur les taches
3. Assignation de tache
4. Rechercher les taches assigne a un membre
5. Rechercher les taches en fonction de leur status
6. Sauvegarder Operation d'un Membre
7. Sortir
-----
votre choix : 3

Pour assigner une tache a un membre Veuillez entrez

1'Id du membre auquel la tache sera assigner : 1
1'Id de la tache qui sera assigner : 1

La Tache a bien ete assigner

```

**Figure** : assignation d'une tache

- ❖ **Afficher les taches affectées à un membre en fonction de son ID** : Pour plus explicité cette section nous avons créé un autre membre appelé « **Nom = Aboubacar Djibo, ID = 2** » une 2<sup>em</sup> tache avec le même statut que la 1<sup>er</sup> « **ID= 2, Nom = Conception, Description = Conception application, Statut = nouveau** » et une 3<sup>em</sup> tache avec un statut différent « **ID = 3, Nom = Câblage, Description = câblage réseau, Statut = terminer** » nous allons assigner les trois (03) taches au première membre et les deux (02) premières taches

au deuxième membre créer. Pour ce faire il faut choisir l'option 4 du menu principal.  
Ci-dessous les taches assignées au premier membre :

```

MENU PRINCIPAL
-----
1. Operation sur les membres
2. Operation sur les taches
3. Assignation de tache
4. Rechercher les taches assigne a un membre
5. Rechercher les taches en fonction de leur status
6. Sauvegarder Operation d'un Membre
7. Sortir
-----
votre choix : 4

Veuillez entrez l'ID du membre : 1

la tache assigner a Maman Sani est :
-----
1          Maintenance      maintenance preventif      nouveau
2          Conception        conception application      nouveau
3          Cablage cablage reseau      terminer

```

Les tâches assignées au second membre :

```

MENU PRINCIPAL
-----
1. Operation sur les membres
2. Operation sur les taches
3. Assignation de tache
4. Rechercher les taches assigne a un membre
5. Rechercher les taches en fonction de leur status
6. Sauvegarder Operation d'un Membre
7. Sortir
-----
votre choix : 4

Veuillez entrez l'ID du membre : 2

la tache assigner a Aboubacar Djibo est :
-----
1          Maintenance      maintenance preventif      nouveau
2          Conception        conception application      nouveau

```

**Figure :** Taches assignées à un membre en fonction de son ID

- ❖ **Affichage des taches en fonction de leur statut :** Nous réalisons cet exemple avec les mêmes informations que précédemment. Il faut choisir l'option 5 du menu principal. Ci-dessous les différentes taches en fonction du statut « **nouveau** »



```

MENU PRINCIPAL
-----
1. Operation sur les membres
2. Operation sur les taches
3. Assignation de tache
4. Rechercher les taches assigne a un membre
5. Rechercher les taches en fonction de leur status
6. Sauvegarder Operation d'un Membre
7. Sortir
-----
votre choix : 5

Veuillez entrez le status de la tache a Affiche : nouveau

Les taches ayant comme status [nouveau] sont :
-----

```

Nom Membre :	Maman Sani	Tache Assigner :	1	Maintenance	maintenence preventif	nouveau
Nom Membre :	Aboubacar Djibo	Tache Assigner :	1	Maintenance	maintenence preventif	nouveau
Nom Membre :	Maman Sani	Tache Assigner :	2	Conception	conception application	nouveau
Nom Membre :	Aboubacar Djibo	Tache Assigner :	2	Conception	conception application	nouveau

-----

Ci-dessous les différentes taches en fonctions du statut « terminer »

```

MENU PRINCIPAL
-----
1. Operation sur les membres
2. Operation sur les taches
3. Assignation de tache
4. Rechercher les taches assigne a un membre
5. Rechercher les taches en fonction de leur status
6. Sauvegarder Operation d'un Membre
7. Sortir
-----
votre choix : 5

Veuillez entrez le status de la tache a Affiche : terminer

Les taches ayant comme status [terminer] sont :
-----

```

Nom Membre :	Maman Sani	Tache Assigner :	3	Cablage	cablage reseau	terminer
--------------	------------	------------------	---	---------	----------------	----------

**Figure** : Liste des différentes taches en fonction d'un statut

A travers ces captures nous pouvons bien voir qu'il est possible d'assigner plusieurs taches à un membre, tout comme nous pouvons aussi voir qu'une même tache peut être assignée à plusieurs membres.

### ✚ Quelques cas d'exception d'assignation

- **Affectation d'une même tâche au même membre** : Nous pensons qu'il n'est pas possible de ai à effectuer une même tâche a deux reprise. Nous tentons dans cet exemple d'assigner au membre 1 « **Maman Sani** » la tâche qui lui a été déjà assigné « **Tache 3** » la ci-dessous illustre ce cas d'assignation:

```

MENU PRINCIPAL
-----
1. Operation sur les membres
2. Operation sur les taches
3. Assignation de tache
4. Rechercher les taches assigne a un membre
5. Rechercher les taches en fonction de leur status
6. Sauvegarder Operation d'un Membre
7. Sortir
-----
votre choix : 3

Pour assigner une tache a un membre Veuillez entrez

1'Id du membre auquel la tache sera assigner : 1
1'Id de la tache qui sera assigner : 3
Cette tache est deja assigne a ce membre

```

- **Cas de suppression d'un membre après lui avoir assignée une tâche** : Pour illustre ce cas nous allons supprimer le membre 1 et par la suite essayer d'afficher les tâches ayant comme statut « **nouveau** ».

#### ○ Suppression d'un membre :

```

-----
| OPERATIONS SUR LES MEMBRES |
-----
1. Creer et Ajouter un membre
2. Modifier un Membre
3. Supprimer un membre
4. Afficher la liste des membres
5. Sauvegarder Operation Membre
6. Retours au menu principal
-----
Entrez votre choix : 3

Le contenu de la liste est :

Id Membre      Nom Membre
1      Maman Sani
2      Aboubacar Djibo

Veuillez Entrez 1'ID du membre a supprimer : 1

Voulez-vous vraiment supprimer le membre [Maman Sani] ?
Entrez 1 pour confirmer la suppression ou 0 pour annuler : 1

Le membre a bien ete supprimer

```

```

MENU PRINCIPAL
-----
1. Operation sur les membres
2. Operation sur les taches
3. Assignation de tache
4. Rechercher les taches assigne a un membre
5. Rechercher les taches en fonction de leur status
6. Sauvegarder Operation d'un Membre
7. Sortir
-----
votre choix : 5

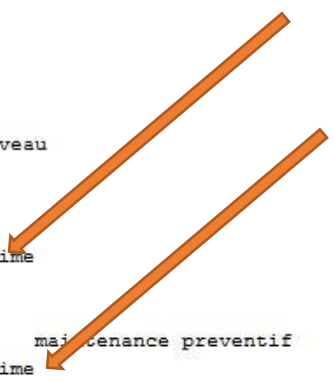
Veuillez entrez le status de la tache a Affiche : nouveau

Les taches ayant comme status [nouveau] sont :
-----
Le membre auquel la tache 1 a ete assigne a ete supprime

Nom Membre :      Aboubacar Djibo
Tache Assigner : 1      Maintenance      maintenance preventif      nouveau
Le membre auquel la tache 2 a ete assigne a ete supprime

Nom Membre :      Aboubacar Djibo
Tache Assigner : 2      Conception      conception application      nouveau

```



- **Cas de l'inexistence d'un membre** : Il est possible d'essayer d'assigner une tache à un membre qui n'existe pas ou le cas contraire (i.e. le membre existe mais la tâche n'existe pas). Nous allons essayer d'assigner au membre que nous venons de supprimer une tache.

```

MENU PRINCIPAL
-----
1. Operation sur les membres
2. Operation sur les taches
3. Assignation de tache
4. Rechercher les taches assigne a un membre
5. Rechercher les taches en fonction de leur status
6. Sauvegarder Operation d'un Membre
7. Sortir
-----
votre choix : 3

Pour assigner une tache a un membre Veuillez entrez

1'Id du membre auquel la tache sera assigner : 1
1'Id de la tache qui sera assigner : 1

Le Membre n'existe pas, veuillez le creer afin de lui assigner une tache !

```

#### **Operations de modification et suppression d'un membre et d'une tache**

- **Opération de modification sur Membre** : pour effectuer une modification l'utilisateur doit choisir l'option 1 du menu principal, puis l'option 2 du menu Membre. Une fois cette option choisit une liste des membres est affiche à l'utilisateur afin qu'il puisse

voir l'ID du membre qu'il souhaite modifier. Il est à noter que la modification n'est possible que sur un membre déjà créé. Ci-dessous l'illustration de la modification du membre 2 :

```

-----
| OPERATIONS SUR LES MEMBRES |
-----
1. Creer et Ajouter un membre
2. Modifier un Membre
3. Supprimer un membre
4. Afficher la liste des membres
5. Sauvegarder Operation Membre
6. Retours au menu principal
-----
Entrez votre choix : 2

Le contenu de la liste est :

Id Membre      Nom Membre
2              Aboubacar Djibo

Entrez l'ID du membre a modifier : 1
Le membre que vous voulez modifier n'existe pas

```

Nous constatons à travers cette capture qu'il n'est pas possible de modifier un membre qui n'existe pas.

```

-----
| OPERATIONS SUR LES MEMBRES |
-----
1. Creer et Ajouter un membre
2. Modifier un Membre
3. Supprimer un membre
4. Afficher la liste des membres
5. Sauvegarder Operation Membre
6. Retours au menu principal
-----
Entrez votre choix : 2

Le contenu de la liste est :

Id Membre      Nom Membre
2              Aboubacar Djibo

Entrez l'ID du membre a modifier : 2
Pour le modifier entrer le nouveau ID : 1
Entrez le nouveau Nom : Maman Sani

Voulez-vous vraiment effectuer la modification ?
Choisissez 1 pour confirmer la modification ou 0 pour annuler : 1

Le Membre a ete bien modifier !

```

Affichons la liste pour vérifier la réussite de la modification.



```

-----
| OPERATIONS SUR LES MEMBRES |
-----
1. Creer et Ajouter un membre
2. Modifier un Membre
3. Supprimer un membre
4. Afficher la liste des membres
5. Sauvegarder Operation Membre
6. Retours au menu principal
-----
Entrez votre choix : 4

Le contenu de la liste est :

Id Membre      Nom Membre
1              Maman Sani

```

La modification a bien réussi

- Opération sur la suppression d'un membre : la procédure est la même que précédemment à la différence qu'au lieu de choisir l'option 2 du menu membre, l'utilisateur

dernière

```

-----
| OPERATIONS SUR LES MEMBRES |
-----
1. Creer et Ajouter un membre
2. Modifier un Membre
3. Supprimer un membre
4. Afficher la liste des membres
5. Sauvegarder Operation Membre
6. Retours au menu principal
-----
Entrez votre choix : 3

Le contenu de la liste est :

Id Membre      Nom Membre
1              Maman Sani

Veuillez Entrez l'ID du membre a supprimer : 1

Voulez-vous vraiment supprimer le membre [Maman Sani] ?
Entrez 1 pour confirmer la suppression ou 0 pour annuler : 1

Le membre a bien ete supprimer

```

Affichons la liste pour vérification :

```
-----  
| OPERATIONS SUR LES MEMBRES |  
-----  
1. Creer et Ajouter un membre  
2. Modifier un Membre  
3. Supprimer un membre  
4. Afficher la liste des membres  
5. Sauvegarder Operation Membre  
6. Retours au menu principal  
-----  
Entrez votre choix : 4  
La liste est vide !
```

Membre bien Supprimer

La procédure est la même pour les opérations sur les taches.

## Conclusion

Nous avons conçus au cours de ce TP une application de gestion de tâche qui peut servir à une équipe de travail, et nous pensons au terme de ce TP avoir atteint les objectifs fixés par ce dernier à savoir celui de nous rappeler les de la programmation Orientée Objet avec le langage Java et UML pour la modélisation. Toutes les spécifications du TP ont été respectée et notre programme est fonctionnel, nous avons utilisé l'outil de test JUnit et les tests d'acceptations pour tester et valider notre programme. Cependant des efforts peuvent être fournis dans le sens de la modélisation, du concept de programmation orientée objet et des bonnes pratiques de la programmation afin d'améliorer et de simplifier notre programme. En somme ce TP a été excellent exercice de mise à niveau sur les concepts de base de la programmation Orienté objet et de la modélisation en génie logiciel.

## Annexe

Ci-dessous notre code source :

❖ Classe Principale (Main)

```
package gestionnairedetache;

public class GestionnaireDeTache {

    static Scanner keyb = new Scanner(System.in);

    static Membre m = null;

    static Tache t;

    public static void main(String[] args) throws IOException, ClassNotFoundException {

        int choix = 0;

        OperationMembreEtTache c = new OperationMembreEtTache();

        do{

            System.out.println();

            choix=menu();

            switch(choix)

            {

                case 1 : c.operationMembre();

                break;

                case 2 : c.operationTache();

                break;

                case 3 : c.assigntationTache();

                break;

                case 4 : c.affichageTacheAssignerEnFonctionID();

                break;

                case 5 : c.AffichageTacheEnFonctionDuStatus();

                break;

                case 6 : c.afficherListeAssigntation();

                break;

                case 7 : System.out.println("Merci pour votre interaction, Au revoirs !");

                        System.exit(0);

                break;

            }

        }
```

```

    } while(choix != 7);
}

/**menu vers les differentes partie du programme*/
public static int menu()
{
    int choix;

    System.out.println("MENU PRINCIPAL");
    System.out.println("-----");
    System.out.println("1. Operation sur les membres");
    System.out.println("2. Operation sur les taches");
    System.out.println("3. Assignment de tache");
    System.out.println("4. Rechercher les taches assigne a un membre");
    System.out.println("5. Rechercher les taches en fonction de leur status");
    System.out.println("6. Afficher la liste des taches assignees");
    System.out.println("7. Sortir");
    System.out.println("-----");
    System.out.print("votre choix : ");
    choix = Keyboard.getEntier();
    System.out.println();
    return choix;
}
}

```

❖ Classe Membre :

```

public class Membre
{
    Scanner keyb = new Scanner(System.in);
    static Vector<Membre> listeMembre;
    private String nom;
    private int id;
    int nbr;
    boolean isNbr;
    public Membre() {
        System.out.println();
        System.out.print("Entrez l'ID du membre a creer : ");
    }
}

```

```

        id = Keyboard.getEntier();
    do{
        System.out.print("Entrez le Nom du membre a creer : ");
        nom=Keyboard.getString();
    } while(nom.isEmpty());
}

public void afficherMembre()
{
    System.out.println(id+"\t"+nom);
}

/***** Les Methodes Getter et Setter *****/

public String getNom()
{
    return nom;
}

public int getId()
{
    return id;
}

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

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

```

#### ❖ Classe Tache

```

public class Tache {
    Scanner keyb = new Scanner(System.in);
    private Vector<Tache> listeTache;
    private String nom;
    private int id;
    private String description;
}

```

```
private String status;
Membre m;
Tache t;
OperationMembreEtTache c = new OperationMembreEtTache();
public Tache() {
    System.out.print("Entrez l'ID de la tache a creer : ");
    id = Keyboard.getEntier();
    do{
        System.out.print("Nom de la tache : ");
        nom = Keyboard.getString();
    } while(nom.isEmpty());
    do
    {
        System.out.print("Description de la tache : ");
        description=Keyboard.getString();
    } while(description.isEmpty());
    do
    {
        System.out.print("Status de la tache : ");
        status=Keyboard.getString();
    } while(status.isEmpty());
}
public void afficherTache()
{
    System.out.println(id + "\t\t" + nom + "\t\t" + description + "\t\t" + status);
}

/***** LES GETTER & SETTER *****/

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;  
    }  
}
```

#### ❖ Classe Assignment

```
public class Assignment {  
    Scanner keyb = new Scanner(System.in);  
    private int idMembre;  
    private int idTache;  
    public Assignment() {  
        System.out.println("Pour assigner une tache a un membre Veuillez entrez ");  
        System.out.println();  
        System.out.print("l'Id du membre auquel la tache sera assigner : ");  
        idMembre = Keyboard.getEntier();  
        System.out.print("l'Id de la tache qui sera assigner : ");  
        idTache = Keyboard.getEntier();  
    }  
}
```

```
public void afficherAssigination()
{
    System.out.println(idMembre+"\t"+idTache);
}

public int getIdMembre() {
    return idMembre;
}

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

public int getIdTache() {
    return idTache;
}

public void setIdTache(int idTache) {
    this.idTache = idTache;
}
}
```

#### ❖ Classe Keyboard

```
public class Keyboard {

    public static String getString() // Lire un String
    {
        String string = "";
        char caractere = '\0';
        try {
            while ((caractere = (char) System.in.read()) != '\n')
            {
                if (caractere != '\r') string = string + caractere;
            }
        }
        catch (IOException e)
        {
        }
```



```

        System.out.println("Erreur de saisi : " + e.getMessage());
    }
    return string;
} // fin de S()

public static int getEntier() {
    Scanner keyboard = new Scanner(System.in);
    int identifiant;
    do {
        while (!keyboard.hasNextInt()) {

            System.out.println("Vous n'avez pas entre un entier");
            System.out.print("Veuillez entrer un entier : ");
            keyboard.next();
        }
        identifiant = keyboard.nextInt();
        if(identifiant < 0)
            System.out.print("Veuillez entrez un entier positif : ");

    } while (identifiant < 0);
    return identifiant;
}
}

```

#### ❖ Classe OperationMembreEtTache

```

public class OperationMembreEtTache {
    private Vector<Membre> listeMembre;
    private Vector<Tache> listeTache;
    private Vector<Assigination> listeAssigination;
    Membre m;
    Tache t;
    Scanner keyb = new Scanner(System.in);
    public OperationMembreEtTache() {
        listeMembre = new Vector();
        listeTache = new Vector();
    }
}

```

```
        listeAssignment = new Vector();
    }

    /******* LES OPERATIONS SUR MEMBRE *****/

    // METHODE DE CREATION ET D'AJOUT D'UN MEMBRE
    public void creerEtAjouterMembre()
    {
        int choix;
        Membre newMembre = new Membre();
        Membre verifierExistance;
        verifierExistance = rechercheMembre(newMembre.getId());
        if(verifierExistance == null) {
            System.out.println();
            System.out.println("Voulez-vous enregistrez ce Membre ?");
            System.out.println();
            System.out.print("Choisissez 1 pour oui et 0 pour non :");
            choix = Keyboard.getEntier();
            if(choix == 1) {
                listeMembre.addElement(newMembre);
                System.out.println();
                System.out.println("Le membre a bien ete ajouter");
            }
        }
        else {
            System.out.println();
            System.out.println("Le membre existe deja");
        }
    }

    // METHODE DE RECHERCHE D'UN MEMBRE EN FONCTION DE SON ID
    public Membre rechercheMembre(int id) {
        Membre membreRechercher = null;
        for(int i = 0; i < listeMembre.size(); i++)
        {
            membreRechercher = (Membre)listeMembre.elementAt(i);
        }
    }
}
```

```
        if(id == membreRechercher.getId())
        {
            return membreRechercher;
        }
        else {

        }

    }
    return membreRechercher = null;
}

// METHODE DE MODIFICATION D'UN MEMBRE
public void modifierMembre() {
    int choix, nouveauId, choix2;
    String nouveauNom;
    Membre personneAmodifier;

    System.out.println();
    if(!listeMembre.isEmpty()){
        afficherListeMembre();
        System.out.println();
        System.out.print("Entrez l'ID du membre a modifier : ");
        choix = Keyboard.getEntier();
        personneAmodifier = rechercheMembre(choix);
        if(personneAmodifier != null) {
            System.out.print("Pour le modifier entrer le nouveau ID : ");
            nouveauId = Keyboard.getEntier();
            System.out.print("Entrez le nouveau Nom : ");
            nouveauNom = Keyboard.getString();

            System.out.println();
            System.out.println("Voulez-vous vraiment effectuer la modification ?");
            System.out.print("Choisissez 1 pour confirmer la modification ou 0 pour annuler : ");
            choix2 = Keyboard.getEntier();
            if(choix2 == 1){
                personneAmodifier.setId(nouveauId);
```

```
        personneAmodifier.setNom(nouveauNom);
        System.out.println();
        System.out.println("Le Membre a ete bien modifier !");
    }
    else if(choix2 == 0) {
        System.out.println();
        System.out.println("La Modification a ete annuler !");
    }
}
else {
    System.out.print("Le membre que vous voulez modifier n'existe pas");
}
}
else {
    System.out.println("Impossible d'effectuer une modification"
        + " car aucun membre n'a ete creer !");
}
}

//METHODE DE SUPPRESSION D'UN MEMBRE
public void supprimerMembre(){
    int id, choix;
    Membre membreAsupprimer;
    System.out.println();
    if(!listeMembre.isEmpty()){
        afficherListeMembre();
        System.out.println();
        System.out.print("Veuillez Entrez l'ID du membre a supprimer : ");
        id = Keyboard.getEntier();
        membreAsupprimer = rechercheMembre(id);
        if(membreAsupprimer != null){
            System.out.println();
            System.out.println("Voulez-vous vraiment supprimer le membre [" +
                membreAsupprimer.getNom() + "] ?");
```

```
System.out.print("Entrez 1 pour confirmer la suppression ou 0 pour annuler : ");
choix = Keyboard.getEntier();
if(choix == 1 ){
    listeMembre.removeElement(membreASupprimer);
    System.out.println();
    System.out.println("Le membre a bien ete supprimer");
}
else if (choix == 0){
    System.out.println();
    System.out.println("La suppression a ete annuler");
}
}
else{
    System.out.println();
    System.out.println("Le membre n'exite pas");
}
}
else{
    System.out.println("Impossible d'effectuer la suppression"
        + " car aucun membre n'a ete creer !");
}
}

//METHODE D'AFFICHAGE DE LA LISTE DES MEMBRES
public void afficherListeMembre()
{
    Membre membreAafficher;
    if(listeMembre.size() == 0)
        System.out.println("La liste est vide !");
    else
    {
        System.out.println();
        System.out.println("Id Membre" + "\t Nom Membre");
        for(int i = 0; i < listeMembre.size(); i++)
```

```
        {
            membreAfficher=(Membre)listeMembre.elementAt(i);
            membreAfficher.afficherMembre();
        }
    }
}

// METHODE DE GESTION DU MENU DU MEMBRE
public int menuMembre(){
    int choixMembre;

    System.out.println("-----");
    System.out.println("| OPERATIONS SUR LES MEMBRES |");
    System.out.println("-----");
    System.out.println("1. Creer et Ajouter un membre");
    System.out.println("2. Modifier un Membre");
    System.out.println("3. Supprimer un membre");
    System.out.println("4. Afficher la liste des membres");
    System.out.println("5. Retours au menu principal");
    System.out.println("-----");
    System.out.print("Entrez votre choix : ");
    choixMembre = Keyboard.getEntier();
    return choixMembre;
}

public void operationMembre() throws IOException {
    int choix;
    do{
        System.out.println();
        choix = menuMembre();
        switch(choix)
        {
            case 1 : creerEtAjouterMembre();
            break;
            case 2 : modifierMembre();
```

```
        break;
        case 3 : supprimerMembre();
        break;
        case 4 : afficherListeMembre();
        break;
        case 5 : continue;//System.exit(0);
    }
} while(choix !=5 || choix < 0);
}

/***** LES OPERATIONS SUR LA TACHE *****/
//METHODE DE GESTION DU MENU DE LA TACHE
public int menuTache(){
    int choixTache;
    System.out.println("-----");
    System.out.println("| OPERATION SUR LES TACHES | ");
    System.out.println("-----");
    System.out.println("1. Creer et Ajouter une tache");
    System.out.println("2. Modifier une tache");
    System.out.println("3. Supprimer une tache");
    System.out.println("4. Afficher la liste des taches");
    System.out.println("5. Retours au menu principal");
    System.out.println("-----");
    System.out.print("Entrez votre choix : ");
    choixTache = Keyboard.getEntier();
    System.out.println();
    return choixTache;
}

public void operationTache(){
    int choix;
    do{
        System.out.println();
        choix = menuTache();
        switch(choix)
```

```
{
    case 1 : creerEtAjouterTache();
    break;
    case 2 : modifierTache();
    break;
    case 3 : supprimerTache();
    break;
    case 4 : afficherListeTache();
    break;
    case 5 : continue;//System.exit(0);
}
}while(choix !=5);
}

// METHODE DE CREATION ET D'AJOUT D'UNE TACHE
public void creerEtAjouterTache()
{
    int choix;
    Tache newTache = new Tache();
    Tache verifierExistenceTache, elementListe;
    verifierExistenceTache = rechercherTacheParId(newTache.getId());
    boolean existence = false;
    if(verifierExistenceTache == null) {
        if(!listeTache.isEmpty()){
            for(int index = 0; index < listeTache.size(); index++){
                elementListe = listeTache.elementAt(index);
                if(elementListe.getNom().equals(newTache.getNom())
                    && elementListe.getDescription().equals(newTache.getDescription())
                    && elementListe.getStatus().equals(newTache.getStatus())){
                    existence = true;
                    break;
                }
            }
        }
        if(existence){
```



```
System.out.println();

System.out.println("La Tache existe deja, il n'est pas possible d'avoir"
    + " deux taches de meme caracteristiques. ");
}else{
    System.out.println();
    System.out.println("Voulez-vous Ajouter cette Tache ?");
System.out.println();
    System.out.print("Choisissez 1 pour oui et 0 pour non :");
    choix = Keyboard.getEntier();
    if(choix == 1) {
        listeTache.addElement(newTache);
        System.out.println();
        System.out.println("La tache a bien ete ajouter !");
    }
    else if(choix == 0) {
        System.out.println("Vous avez annuler l'enregistrement !");
    }
}
}else{
    System.out.println();
    System.out.println("Voulez-vous Ajouter cette Tache ?");
System.out.println();
    System.out.print("Choisissez 1 pour oui et 0 pour non :");
    choix = Keyboard.getEntier();
    if(choix == 1) {
        listeTache.addElement(newTache);
        System.out.println();
        System.out.println("La tache a bien ete ajouter !");
    }
    else if(choix == 0) {
        System.out.println("Vous avez annuler l'enregistrement !");
    }
}
}
```

```
    }  
    else {  
        System.out.println();  
        System.out.println("La tache existe deja");  
    }  
}
```

// METHODE DE RECHERCHE D'UNE TACHE EN FONCTION DE SON ID

```
public Tache rechercherTacheParId(int idTache)  
{  
    Tache tacheRechercher = null;  
    for(int i = 0; i < listeTache.size(); i++)  
    {  
        tacheRechercher = (Tache)listeTache.elementAt(i);  
        if(idTache == tacheRechercher.getId())  
        {  
            return tacheRechercher;  
        }  
    }  
    return tacheRechercher = null;  
}
```

// METHODE DE RECHERCHE D'UNE TACHE EN FONCTION DE SON STATUS

```
public Tache rechercherTacheParStatus(String statusTache)  
{  
    Tache tacheRechercher = null;  
    for(int i = 0; i < listeTache.size(); i++)  
    {  
        tacheRechercher = (Tache)listeTache.elementAt(i);  
        if(statusTache.equals(tacheRechercher.getStatus()))  
        {  
            return tacheRechercher;  
        }  
    }  
}
```

```
        return tacheRechercher = null;
    }

    // METHODE DE MODIFICATION D'UNE TACHE
    public void modifierTache() {
        int choix, nouveauId, choix2;
        String nouveauNom = "";
        String nouvelleDescription, nouveauStatus;
        Tache tacheAmodifier;
        if(!listeTache.isEmpty()){
            afficherListeTache();
            System.out.println();
            System.out.print("Entrez l'ID de la tache a modifier :");
            choix = Keyboard.getEntier();
            tacheAmodifier = rechercherTacheParId(choix);
            if(tacheAmodifier != null) {
                System.out.print("Pour la modifier entrer le nouveau ID : ");
                nouveauId = Keyboard.getEntier();
                System.out.print("Entrez le nouveau Nom : ");
                nouveauNom = Keyboard.getString();
                System.out.print("Entrez la nouvelle Description : ");
                nouvelleDescription = Keyboard.getString();
                System.out.print("Entrez le nouveau Status : ");
                nouveauStatus = Keyboard.getString();
                System.out.println();
                System.out.println("Voulez-vous vraiment effectuer la modification ?");
                System.out.print("Choisissez 1 pour confirmer la modification ou 0 pour annuler : ");
                choix2 = Keyboard.getEntier();
                if(choix2 == 1){
                    tacheAmodifier.setId(nouveauId);
                    tacheAmodifier.setNom(nouveauNom);
                    tacheAmodifier.setDescription(nouvelleDescription);
                    tacheAmodifier.setStatus(nouveauStatus);
                    System.out.println();
                }
            }
        }
    }
}
```

```
        System.out.println("La Tache a ete bien modifier !");
    }
    else if(choix2 == 0) {
        System.out.println();
        System.out.println("La Modification a ete annuler !");
    }
}
else {
    System.out.println("La Tache que vous voulez modifier n'existe pas");
}
}
else {
    System.out.println("Impossible de d'effectuer une modification"
        + " car aucune tache n'a ete creer !");
}
}

// METHODE DE SUPPRESSION D'UNE TACHE
public void supprimerTache() {
    int id, choix;
    Tache tacheAsupprimer;
    System.out.println();
    if(!listeTache.isEmpty()) {
        afficherListeTache();
        System.out.println();
        System.out.print("Veuillez Entrez l'ID de la tache a supprimer : ");
        id = Keyboard.getEntier();
        tacheAsupprimer = rechercherTacheParId(id);
        if(tacheAsupprimer != null) {
            System.out.println();
            System.out.println("Voulez-vous vraiment supprimer la tache [" +
                tacheAsupprimer.getNom() + "] ?");
            System.out.print("Entrez 1 pour confimer la suppression ou 0 pour annuler : ");
            choix = Keyboard.getEntier();
        }
    }
}
```

```
        if(choix == 1 ){
            listeTache.removeElement(tacheASupprimer);
            System.out.println();
            System.out.println("La tache a bien ete supprimer");
        }
        else if (choix == 0){
            System.out.println();
            System.out.println("La suppression a ete annuler");
        }
    }
}
else{
    System.out.println();
    System.out.println("La tache n'exite pas");
}
}
else{
    System.out.println("Impossible de supprimer une tache car aucune tache n'a ete creer !");
}
}

// METHODE D'AFFICHAGE DE LA LISTE DE TOUTES LES TACHES
public void afficherListeTache()
{
    Tache tacheAafficher;
    if(listeTache.size() == 0)
        System.out.println("Liste vide !");
    else
    {
        System.out.println("Id Tache" + "\t Nom Tache" + "\t Description" + " \t Status");
        for(int i = 0; i < listeTache.size(); i++)
        {
            //System.out.println("Le nous avons "+listeTache.size()+" tache dans la liste");
            tacheAafficher=(Tache)listeTache.elementAt(i);
            tacheAafficher.afficherTache();
        }
    }
}
```

```

    }
}
}

// METHODE D'ASSIGNATION D'UNE TACHE A UN MEMBRE
public void assignationTache() {
    if(!listeMembre.isEmpty()) && !listeTache.isEmpty()) {
        System.out.println("Veuillez vous refferez aux deux listes "
            + "ci-dessous pour realiser votre assignation");
        System.out.println();
        System.out.println("Liste des membres : ");
        afficherListeMembre();
        System.out.println();
        System.out.println("-----");
        System.out.println("Liste des taches : ");
        afficherListeTache();
        System.out.println();
        Assignation tacheAssigner = new Assignation();
        Assignation element = null;
        Membre verifierExistanceMembre;
        Tache verifierExistanceTache;
        boolean existance = false;

        verifierExistanceMembre = rechercheMembre(tacheAssigner.getIdMembre());
        verifierExistanceTache = rechercherTacheParId(tacheAssigner.getIdTache());
        if(verifierExistanceMembre == null) {
            System.out.println();
            System.out.println("Le Membre n'existe pas, veuillez le creer afin de lui assigner une
tache !");
        }
        else if(verifierExistanceTache == null) {
            System.out.println();
            System.out.println("La Tache n'existe pas, veuillez la creer afin de l'assigner");
        }
    }
}

```

```
else {
    if(listeAssignment.isEmpty()){
        listeAssignment.addElement(tacheAssigner);
        System.out.println();
        System.out.println("La Tache a bien ete assigner");
    }
    else{
        for(int i = 0; i < listeAssignment.size(); i++){
            element = listeAssignment.elementAt(i);
            if((element.getIdMembre() == tacheAssigner.getIdMembre()) &&
                element.getIdTache() == tacheAssigner.getIdTache()){
                existence = true;
                break;
            }
        }
        if(existence == true){
            System.out.println("Cette tache est deja assigne a ce membre");
        }
        else{
            listeAssignment.addElement(tacheAssigner);
            System.out.println();
            System.out.println("La Tache a bien ete assigner");
        }
    }
}

} else if(!listeMembre.isEmpty()){
    System.out.println("Vous ne pouvez pas effectuer cette operation car "
        + "la liste des taches est vides. ");
}
else{
    System.out.println("Vous ne pouvez pas effectuer cette operation car "
        + "la liste des membres est vides. ");
}
```

```
}

//METHODE D'AFFICHAGE DE LA LISTE DES TACHES ASSIGNEES

public void afficherListeAssignment()
{
    Assignment assignmentAafficher;
    if(listeAssignment.size() == 0)
        System.out.println("Liste vide !");
    Else{
        System.out.println("La liste contient "+listeAssignment.size()+" assignment");
        System.out.println();
        System.out.println("Id Membre"+"\\t Id Tache");
        for(int i = 0; i < listeAssignment.size(); i++)
        {
            assignmentAafficher=(Assignment)listeAssignment.elementAt(i);
            assignmentAafficher.afficherAssignment();
        }
    }
}

// METHODE D'AFFICHAGE DE LA TACHE ASSIGNEE A UN MEMBRE EN FONCTION DE SON ID

public void affichageTacheAssignerEnFonctionID() {
    Membre existenceMembre;
    int idTache, idMembre;
    Assignment assignment;
    Tache tacheAafficher;
    System.out.print("Veuillez entrez l'ID du membre : ");
    idMembre = Keyboard.getEntier();
    System.out.println();
    existenceMembre = rechercheMembre(idMembre);
    if(existenceMembre != null) {
        System.out.println("la tache assigner a " + existenceMembre.getNom()
            + " est :");
        System.out.println("-----");
        for(int i = 0; i < listeAssignment.size(); i++) {
```



```

        assignation = (Assignation) listeAssignation.elementAt(i);
        if(assignation.getIdMembre() == idMembre) {
            idTache = assignation.getIdTache();
            tacheAfficher = rechercherTacheParId(idTache);
            if(tacheAfficher != null) {
                tacheAfficher.afficherTache();
            }
        }
    }
}
else{
    System.out.println("Le membre n'existe pas");
}
}

```

// METHODE D'AFFICHAGE D'UNE TACHE EN FONCTION DE STATUS ET DU MEMBRE AUQUEL ELLE EST ASSIGNEE

```

public void AffichageTacheEnFonctionDuStatus(){
    int idTache, idMembre;
    String status;
    Membre existenceMembre, membre;
    Tache existenceTache, tacheAfficher;
    Assignation tacheAssigner;
    System.out.print("Veuillez entrez le status de la tache a Affiche : ");
    status = Keyboard.getString();
    existenceTache = rechercherTacheParStatus(status);
    if(existenceTache != null){
        System.out.println();
        System.out.println("Les taches ayant comme status [" + status + "] sont : " );
        System.out.println("-----");
        for(int i = 0; i < listeTache.size(); i++){
            tacheAfficher = (Tache) listeTache.elementAt(i);
            if(tacheAfficher.getStatus().equals(status)){
                idTache = tacheAfficher.getId();
            }
        }
    }
}

```

```

for(int j = 0; j < listeAssignation.size(); j++){
    tacheAssigner = (Assignation) listeAssignation.elementAt(j);
    if(tacheAssigner.getIdTache() == idTache){
        idMembre = tacheAssigner.getIdMembre();
        existenceMembre = rechercheMembre(idMembre);
        if(existenceMembre != null){
            for(int k = 0; k < listeMembre.size(); k++){
                membre = (Membre) listeMembre.elementAt(k);
                if(idMembre == membre.getId()){
                    System.out.println();
                    System.out.println("Nom Membre : " + "\t" + membre.getNom());
                    System.out.print("Tache Assigner : " );
                    tacheAfficher.afficherTache();
                    System.out.println("-----");
                }
            }
        }
        else{
            System.out.println("Le membre auquel la tache "+ tacheAssigner.getIdTache()
                +" a ete assigne a ete supprime");
        }
    }
}

}

}

}

}

else{
    System.out.println("La tache n'existe pas ");
}
}
}

```

## References

- [1] Coursera. (2017). *Coursera / Online Courses From Top Universities. Join for Free*. [online] Available at: <https://www.coursera.org/learn/init-prog-java/home/welcome> [Accessed 19 Aug. 2017].
- [2] Coursera. (2017). *Coursera / Online Courses From Top Universities. Join for Free*. [online] Available at: <https://www.coursera.org/learn/java-poo/home/welcome> [Accessed 19 Aug. 2017].
- [3] Herby, C. (2017). *Apprenez à programmer en Java @OpenClassrooms*. [online] OpenClassrooms. Available at: <https://openclassrooms.com/courses/apprenez-a-programmer-en-java> [Accessed 19 Aug. 2017].
- [4] Roels, C. (2017). *Débutez l'analyse logicielle avec UML @OpenClassrooms*. [online] OpenClassrooms. Available at: <https://openclassrooms.com/courses/debutez-l-analyse-logicielle-avec-uml> [Accessed 19 Aug. 2017].