INSTITUT FRANCOPHONE INTERNATIONAL (IFI)

****

Rapport TP1
Génie Logiciel
Projet: Un petit gestionnaire de tâches

Rédigé par:

- **Zoumana Mallé**

Professeur:

- Ho Tuong Vinh, IFI

Promotion 21 Hanoï 2017

Sommaire

1. Introduction.....3
2. Les exigences fonctionnelles et non-fonctionnelles de l'application.....3
2.1. Spécification de l'application.....3
2.2. Exigences fonctionnelles.....4
3. La Conception.....6
3.1. Le diagramme de classe.....6
3.2. Les diagrammes de séquence.....7
4. L'implémentation.....7
4.1. Plateforme et langage programmation.....7
4.2. Test Unitaire (Junit)……...8
5. Le test d'acceptation.....9
5.1. Ajouter un membre.....9
5.2. Modifier un membre.....9
5.4. Afficher les membres.....10
5.5. Supprimer.....10
5.6. Ajouter une tâche.....11
5.7. Afficher la liste des tâches……...11
5.8. Modifier une tâche……..……..…….11
5.9. Assigner une tâche……..……..…….12
6. Lister tous les Assignés... ……...…...12
6.1.Lister tous les tâches d'un membre..……...12
6.2. Chercher une tâche par status……..13
7. Conclusion.....13
8. Source code.....14

1. Introduction.

Dans le cadre du cours de Génie Logiciel, nous avions pour travaux pratique la création d'un petit gestionnaire de Tâche d'une équipe afin de nous rappeler les bases de la modélisation sous UML et la programmation orientée objet avec Java , ainsi que notre familiarisation avec l'environnement de développement intégré (IDE) par exemple : ECLIPSE (Open Source).

un gestionnaire de tâche est très utile pour l'ordonnancement et le contrôle des travaux d'une équipe tout en ayant une vue d'ensemble de l'état d'avancement dans une entreprise ou une organisation.

2. Les exigences fonctionnelles et non-fonctionnelles de l'application.

2.1. Spécification de l'application.

l'objectif de ce projet est d'effectué un petit gestionnaire de tâches pour une équipe de travail. Les entrées de ce gestionnaire sont : l'identifiant (**ID**), le **nom** d'une tâche, la D**escription**, le **statut** voir l'état de progression de la tâche (nouveau, en-progrès, terminé) ; pour un membre nous avions l'identifiant (**ID**), le **nom du membre**.

Ce gestionnaire fournit à l'utilisateur les fonctionnalités suivantes:

1.Créer, modifier, supprimer, ajouter une tâche

2.Créer, modifier, supprimer, ajouter un membre

3.Assigner une tâche à un membre

4.Chercher et afficher tous les tâches assignées à un membre (par son ID)

5.Chercher et afficher tous les tâches en fonction de leur statut (avec le nom du assigné)

2.2. Exigences fonctionnelles.

A.Use case diagramme ou Diagramme de cas d'utilisations.

Figure 1 : Diagramme de cas d'Utilisations

NB : Si on veut supprimer ou modifier un membre, on peut chercher et afficher ses informations ou afficher toutes les personnes enregistrées et ensuite choisir un membre pour le supprimer ou le modifier même cas similaire pour les tâches.

B. Les exigences fonctionnelles seront énumérées dans le tableau ci-dessous:

EF 1	**Créer un nouveau membre**
EF 1.1	Le système doit permettre d'ajouter un nouveau membre
EF 1.2	Le système doit demander d'entrer les informations nécessaires d'un membre
EF 1.3	Quand un membre existe déjà, le système ne permet pas d'ajouter
EF 2	**Chercher un membre**
EF 2.1	Le système doit permettre d'ajouter un membre à travers son nom ou ID
EF 2.2	Le système affiche une liste de résultats de recherche avec toutes les informations d'un membre

EF 3	**Supprimer un membre**
EF 3.1	Le système doit permettre de supprimer un membre
EF 3.2	Le système doit permettre de supprimer une liste de membre
EF 3.3	Le système doit demander la confirmation de suppression d'un membre
EF 4	**Modifier un membre**
EF 4.1	Le système doit permettre de chercher un membre et afficher toutes ses informations sur l'écran pour les modifier
EF 4.2	Le système doit permettre d'enregistrer les nouvelles informations d'un membre
EF 5	**Afficher une liste de membre**
EF 5.1	Le système doit permettre d'afficher toutes les personnes enregistrées

EF 6	**Créer une nouvelle tâche**
EF 6.1	Le système doit permettre d'ajouter une nouvelle tâche
EF 6.2	Le système doit demander d'entrer les informations d'une tâche
EF 6.3	Quand une tâche existe déjà, le système ne permet d'en ajouter
EF 7	**Chercher une tâche**
EF 7.1	Le système doit permettre d'ajouter une nouvelle tâche à travers son nom ou ID
EF 7.2	Le système affiche une liste de résultats de recherche avec toutes les informations d'une tâche et voir l'état de son statut
EF 8	**Supprimer une tâche**
EF 8.1	Le système doit permettre de supprimer une tâche
EF 8.2	Le système doit permettre de supprimer une liste de tâche exécutée
EF 8.3	Le système doit demander la confirmation de suppression d'une tâche
EF 9	**Modifier une tâche**
EF 9.1	Le système doit permettre de chercher une tâche et afficher toutes ses informations sur l'écran pour les modifier
EF 9.2	Le système doit permettre d'enregistrer les nouvelles informations d'une tâche
EF 10	**Afficher une liste de tâche**
EF 10.1	Le système doit permettre d'afficher toutes les tâches enregistrées
EF 11	**Assigner une tâche à un membre**
EF 10.1	Chercher et afficher tous les tâches assignées à un membre (par son ID)
EF 10.2	Chercher et afficher tous les tâches en fonction de leur statut (avec le nom du assigné)

Tableau 1 : Des exigences fonctionnelles (**EF**)

C. Les exigences non fonctionnelles seront listées ci-dessous dans le tableau:

ENF 1	Le système doit être accessible pendant toutes les opérations
ENF 2	Temps d'attente à un niveau acceptable (< 6 seconde)
ENF 3	Le système doit être convivial, facile d'utilisation

ENF 4	Le système doit être multi plateformes
ENF 5	Le système ne cause pas de dommage à l'équipement
ENF 6	Lorsque l'erreur survienne, les données sauvegarder doivent être préservées

Tableau 2 : Les exigences non fonctionnelles (**ENF**)

3. La Conception.

3.1. Le diagramme de classe.

Figure 2 : Diagramme de Classe

3.2. Le diagramme de séquence.

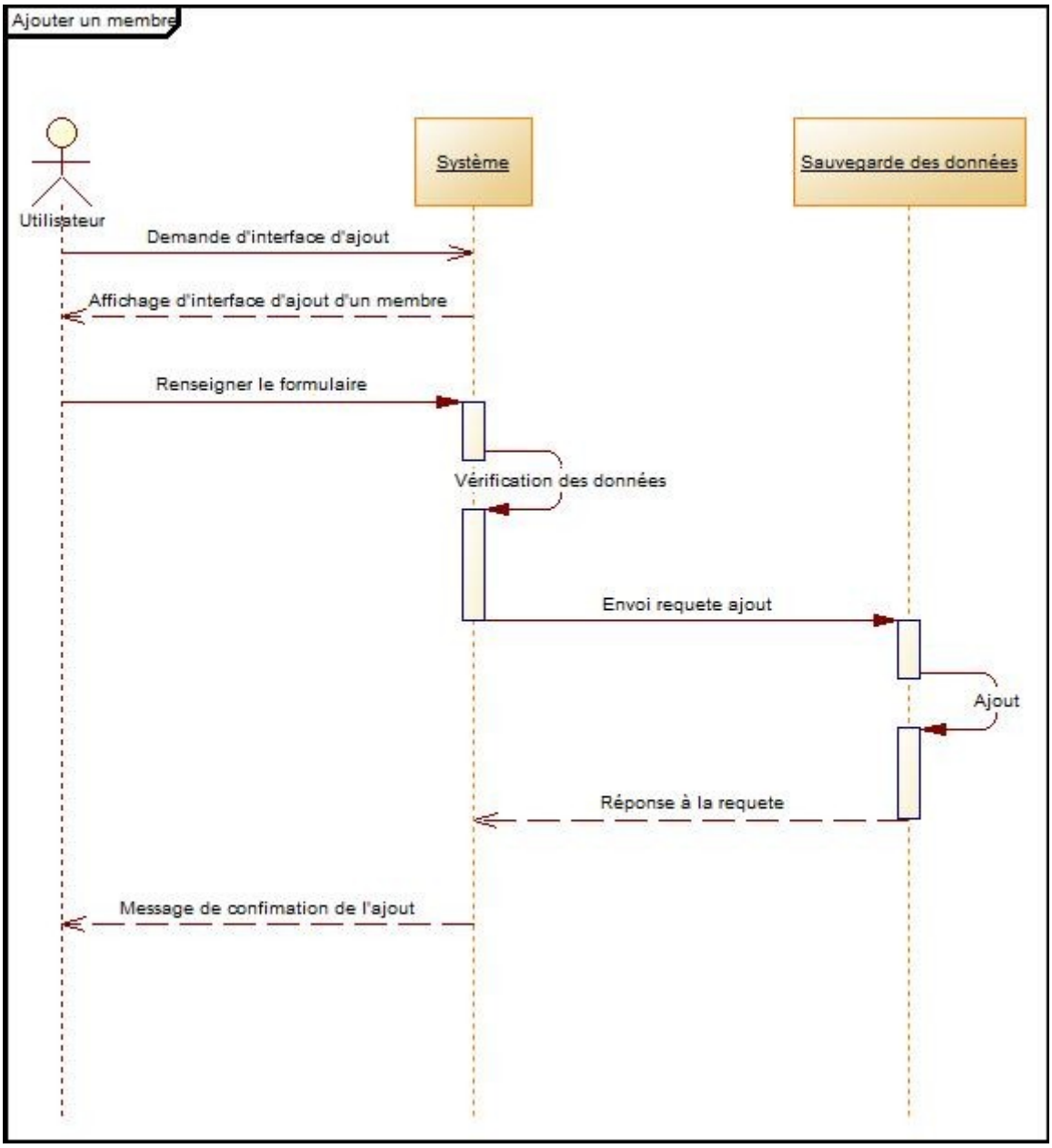


***Figure** 3 :* *Diagramme de séquences (méthode ajouter un membre)*

4. L'implémentation.

4.1. Plateforme et langage de programmation.

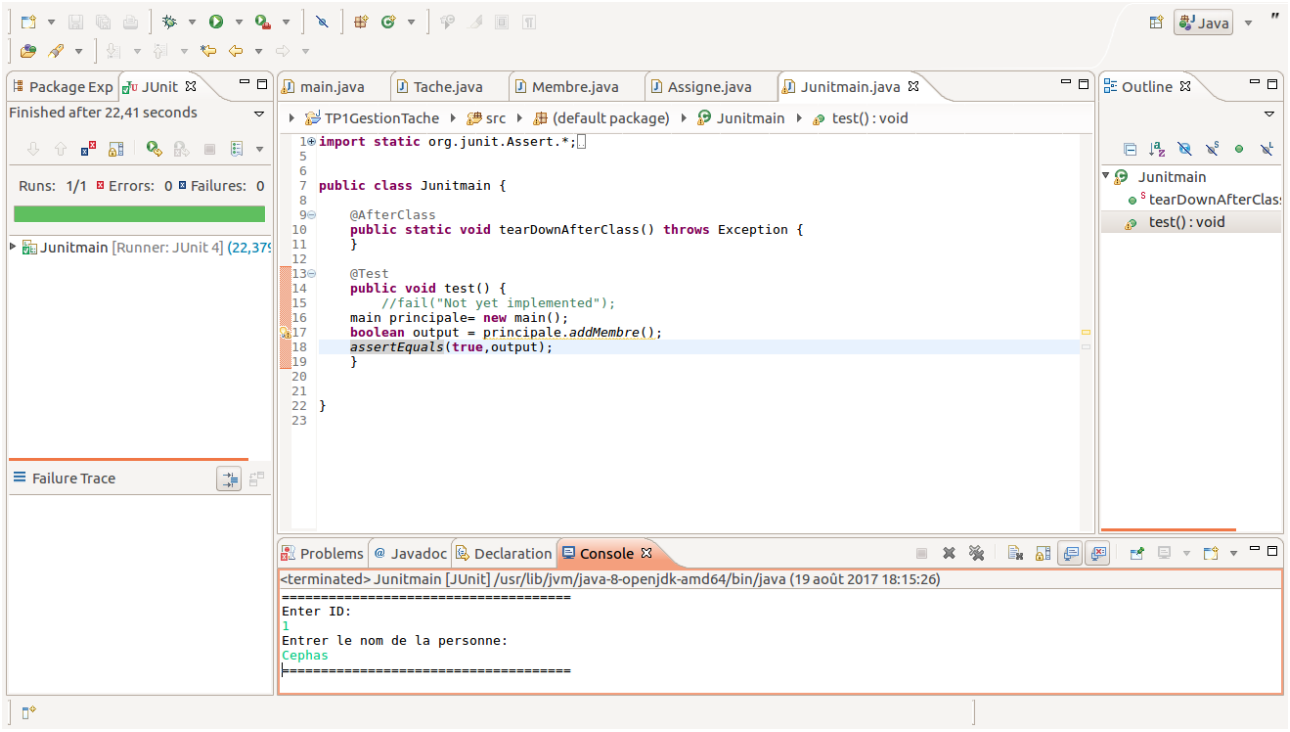
Nous utiliserons le langage java à partir de la plateforme **IDE-Eclipse** pour l'implémentation de cette application. Ce langage nous permettra de développer en orientée objet à travers une bonne structuration des modules.

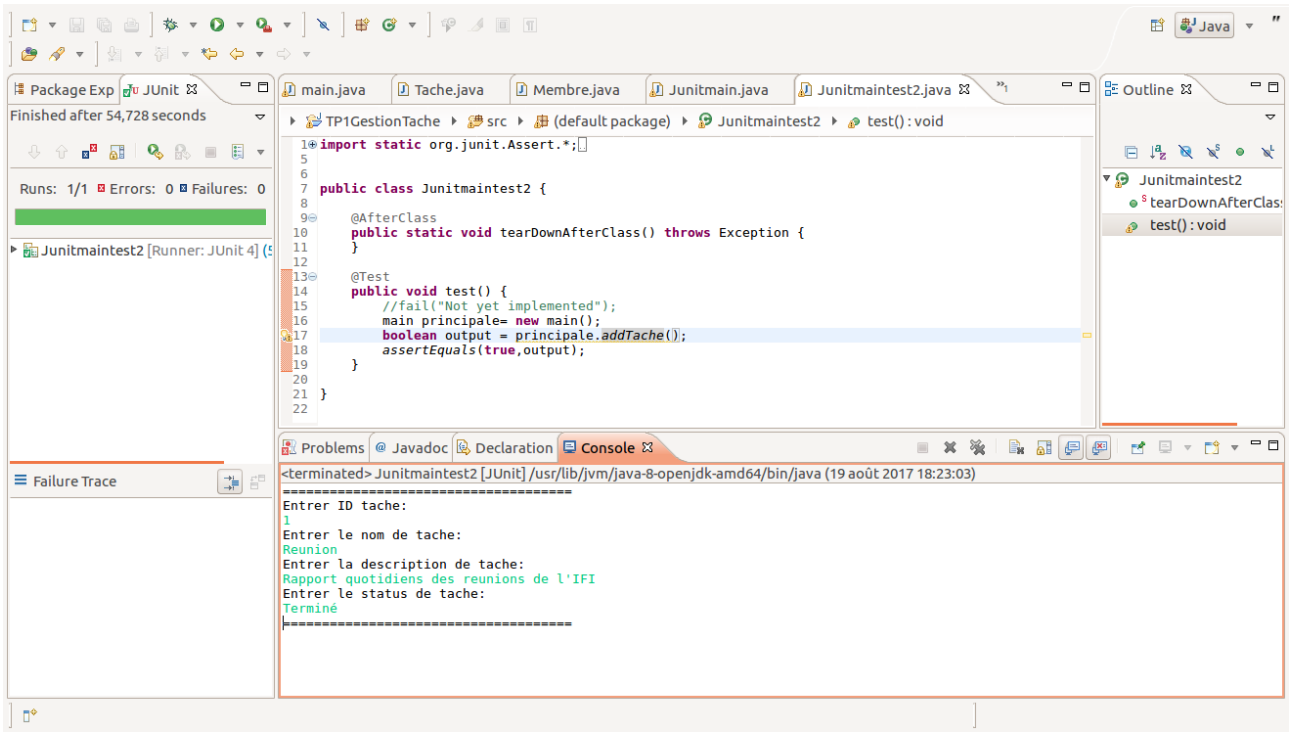
NB : Dans l implémentation de notre application nous avons crée quatre classes qui sont :

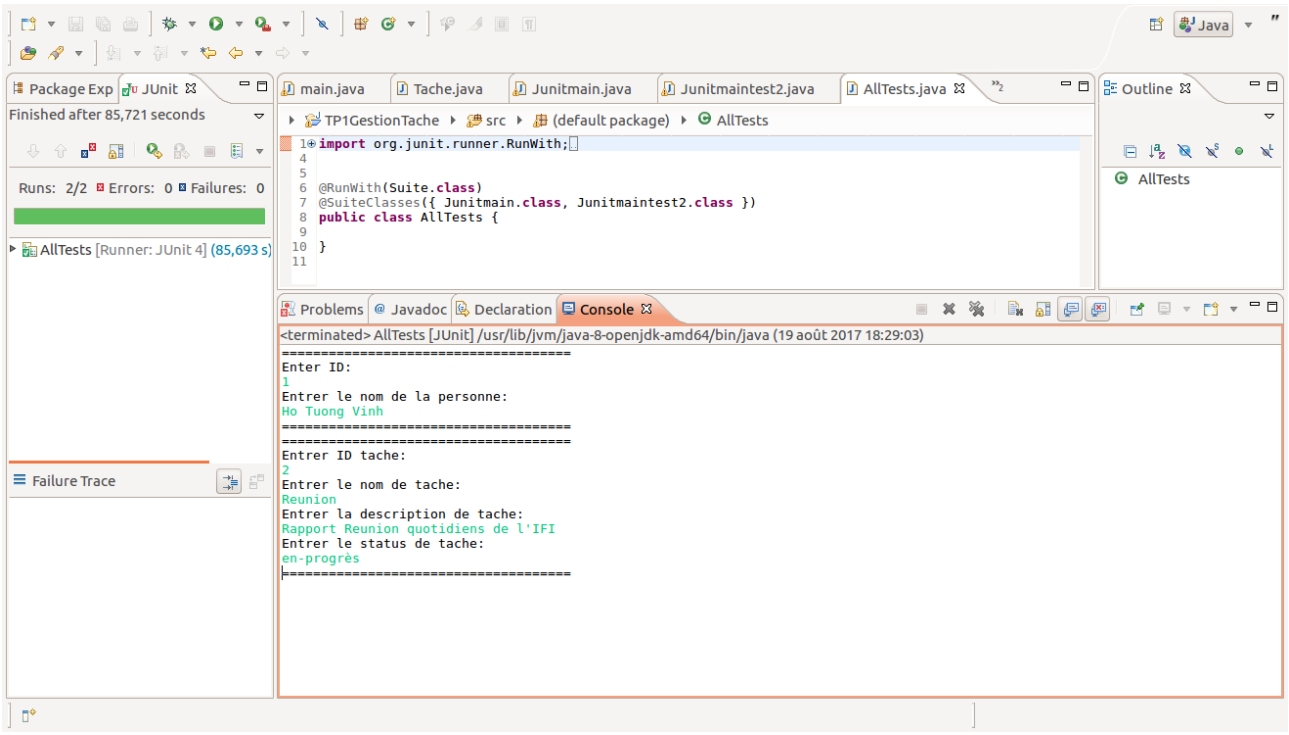
* la classe **Membre** qui définit les propriétés d'un membre ;
* la classe **Assigne** qui permet de gérer toutes les tâches assignées à un membre à travers son Identifiant (ID) ;
* la classe **Tache** qui définit les propriétés d'une tâche ;
* la classe **Main** qui joue le rôle de classe principale servant à l'exécution de notre application et permet de gérer toutes les opérations .

Les détails et le fonctionnement de ces classes sont disponible en annexes.

4.2. Test Unitaire (Junit).







5. Le test d'acceptation.

Les tests réalisés sur notre application sont résumés par les captures d'écrans ci-dessous :

5.1. Ajouter un membre.

```
===>Gestion des Taches<===
1. Ajouter un membre.
2. Supprimer un membre
3. Modifier un membre
4. Lister tous les membres
5. Ajouter une tache.
6. Supprimer une tache
7. Modifier une tache
8. Lister tous les taches
9. Assigner une tache
10. Lister tous les assigne
11. Lister tous les taches d'un membre
12. Chercher une tache par status
Choisissez une option ci-dessus:
1
Ajouter un membre:
=====================================
Enter ID:
2
Entrer le nom de la personne:
Ho Tuong Vinh
=====================================
```

5.2. Modifier un membre

```
===>Gestion des Taches<===
1. Ajouter un membre.
2. Supprimer un membre
3. Modifier un membre
4. Lister tous les membres
5. Ajouter une tache.
6. Supprimer une tache
7. Modifier une tache
8. Lister tous les taches
9. Assigner une tache
10. Lister tous les assigne
11. Lister tous les taches d'un membre
12. Chercher une tache par status
Choisissez une option ci-dessus:
3
Modifier un membre
=====================================
Entrer ID membre pour la modification:
2
Changer le nom de la personne:
Daniel Medou Magloire
=====================================
```

5.4. Afficher les membres

```
===>Gestion des Taches<===
1. Ajouter un membre.
2. Supprimer un membre
3. Modifier un membre
4. Lister tous les membres
5. Ajouter une tache.
6. Supprimer une tache
7. Modifier une tache
8. Lister tous les taches
9. Assigner une tache
10. Lister tous les assigne
11. Lister tous les taches d'un membre
12. Chercher une tache par status
Choisissez une option ci-dessus:
4
Lister tous les membres:
====================================
Membre [Id_Membre=2, NomMembre=Daniel Medou Magloire]
------------------------
====================================
```

5.5. Supprimer:

```
===>Gestion des Taches<===
1. Ajouter un membre.
2. Supprimer un membre
3. Modifier un membre
4. Lister tous les membres
5. Ajouter une tache.
6. Supprimer une tache
7. Modifier une tache
8. Lister tous les taches
9. Assigner une tache
10. Lister tous les assigne
11. Lister tous les taches d'un membre
12. Chercher une tache par status
Choisissez une option ci-dessus:
2
Supprimer un membre:
====================================
Entrer ID membre pour l'élimination:
2
====================================
```

```
===>Gestion des Taches<===
1. Ajouter un membre.
2. Supprimer un membre
3. Modifier un membre
4. Lister tous les membres
5. Ajouter une tache.
6. Supprimer une tache
7. Modifier une tache
8. Lister tous les taches
9. Assigner une tache
10. Lister tous les assigne
11. Lister tous les taches d'un membre
12. Chercher une tache par status
Choisissez une option ci-dessus:
4
Lister tous les membres:
====================================
====================================
```

5.6. Ajouter une tâche:

```
===>Gestion des Taches<===
1. Ajouter un membre.
2. Supprimer un membre
3. Modifier un membre
4. Lister tous les membres
5. Ajouter une tache.
6. Supprimer une tache
7. Modifier une tache
8. Lister tous les taches
9. Assigner une tache
10. Lister tous les assigne
11. Lister tous les taches d'un membre
12. Chercher une tache par status
Choisissez une option ci-dessus:
5
Ajouter une tache:
====================================
Entrer ID tache:
3
Entrer le nom de tache:
Maintenance des ordinateurs
Entrer la description de tache:
Reinstallations des Systèmes d'exploitation et update
Entrer le status de tache:
Nouveau
====================================
```

```
===>Gestion des Taches<===
1. Ajouter un membre.
2. Supprimer un membre
3. Modifier un membre
4. Lister tous les membres
5. Ajouter une tache.
6. Supprimer une tache
7. Modifier une tache
8. Lister tous les taches
9. Assigner une tache
10. Lister tous les assigne
11. Lister tous les taches d'un membre
12. Chercher une tache par status
Choisissez une option ci-dessus:
5
Ajouter une tache:
====================================
Entrer ID tache:
4
Entrer le nom de tache:
Reunion
Entrer la description de tache:
Rapport des reunions quotidiens à l'IFI
Entrer le status de tache:
en-progrès
====================================
```

5.7. Afficher la liste des tâches

```
Choisissez une option ci-dessus:
8
Lister tous les taches:
Tache [Id_tache=3, Nom=Maintenance des ordinateurs , Description=Reinstallations des Systèmes d'exploitation et update , Status=Nouveau]
--------------------------
Tache [Id_tache=4, Nom=Reunion , Description=Rapport des reunions quotidiens à l'IFI , Status=en-progrès]
--------------------------
```

5.8. Modifier une tâche

```
Choisissez une option ci-dessus:
7
Modifier une tache:
====================================
Entrer ID pour la modification:
4
Entrer le nouveau nom de tache:
Reunion
Entrer la nouvelle description de tache:
Rapport des reunions quotidiens à l'IFI
Entrer le nouveau status de tache:
terminé
====================================
```

```
===>Gestion des Taches<===
1. Ajouter un membre.
2. Supprimer un membre
3. Modifier un membre
4. Lister tous les membres
5. Ajouter une tache.
6. Supprimer une tache
7. Modifier une tache
8. Lister tous les taches
9. Assigner une tache
10. Lister tous les assigne
11. Lister tous les taches d'un membre
12. Chercher une tache par status
Choisissez une option ci-dessus:
8
Lister tous les taches:
Tache [Id_tache=4, Nom=Reunion , Description=Rapport des reunions quotidiens à l'IFI , Status=terminé]
--------------------------
```

5.9. Assigner une Tâche

```
Choisissez une option ci-dessus:
3
Modifier un membre
====================================
Entrer ID membre pour la modification:
2
Ce membre n'existe pas
====================================
```

```
Choisissez une option ci-dessus:
9
Assigner une tache:
====================================
Entrer ID Assigne:
1
Entrer ID tache:
1
Entrer ID du membre:
3
====================================
```

6. Lister tous les Assignés

```
Choisissez une option ci-dessus:
10
Lister tout les assignes:
ID Assigne: 1
Membre: Membre [Id_Membre=3, NomMembre=Daniel Medou Magloire]
Tache: Tache [Id_tache=1, Nom=Reunion , Description=Rapport des reunions quotidiens de l'IFI , Status=Terminé]

--------------------------
```

6.1. Lister tous les tâches d’un membre

```
Choisissez une option ci-dessus:
11
Lister tous les taches d'un membre:
Entrer ID membre pour la recherche:
3
====================================
Les taches du membre ID: 3
Tache [Id_tache=1, Nom=Reunion , Description=Rapport des reunions quotidiens de l'IFI , Status=Terminé]
====================================
```

6.2. Chercher une tache par status

```
===>Gestion des Taches<===
1. Ajouter un membre.
2. Supprimer un membre
3. Modifier un membre
4. Lister tous les membres
5. Ajouter une tache.
6. Supprimer une tache
7. Modifier une tache
8. Lister tous les taches
9. Assigner une tache
10. Lister tous les assigne
11. Lister tous les taches d'un membre
12. Chercher une tache par status
Choisissez une option ci-dessus:
12
Chercher une tache par status:
Entrer le status pour la recherche:
Terminé
====================================
Les taches avec status (Terminé):
Tache [Id_tache=1, Nom=Reunion , Description=Rapport des reunions quotidiens de l'IFI , Status=Terminé]
====================================
```

```
Choisissez une option ci-dessus:
12
Chercher une tache par status:
Entrer le status pour la recherche:
en-progrès
====================================
Les taches avec status (en-progrès):
====================================
```

```
Choisissez une option ci-dessus:
12
Chercher une tache par status:
Entrer le status pour la recherche:
Nouveau
====================================
Les taches avec status (Nouveau):
====================================
```

7. Conclusion :

Au cours de ce projet de TP nous avions puis atteindre les objectifs à fortiori fixés tout en ayant amélioré notre compréhension de la programmation orientée objet avec le langage Java. Nous avions puis élaborés les spécifications ci-haut énumérées , qui ont toutes été bien coordonnées en adéquation avec les exigences ; notre gestionnaire de tâche est fonctionnelle. Même si beaucoup d'efforts restent à fournir dans l'optique d'approfondir nos connaissances de l'orientée objet et de nous rendre apte dans la maîtrise du langage Java et la mise en œuvre des bonnes techniques de programmation .

Nous retenons que ce Travaux Pratique a été un très bon moyens pour l'assimilation des concepts de base du cours de génie logiciel , les étapes du processus de développement d'un logiciel et de savoir comment utiliser UML dans la création d'un logiciel ainsi que le test de Junit.

8. Annexes :Code Source

1. Classe main

```
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import java.util.Scanner;

import javax.management.openmbean.ArrayType;

public class main {

	static ArrayList<Membre> arrMembre = new ArrayList<Membre>();
	static ArrayList<Tache> arrTache = new ArrayList<Tache>();
	static ArrayList<Assigne> arrAssigne = new ArrayList<Assigne>();

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

	// Ajouter un membre

	public static boolean addMembre() {
		System.out.println("====================================");
		System.out.println("Enter ID: ");
		int id = 0;
		id = sc.nextInt();
		sc.nextLine();
		System.out.println("Entrer le nom de la personne: ");
		String nom = sc.nextLine();
		Membre mem = new Membre(id, nom);
		arrMembre.add(mem);

		System.out.println("====================================");
return true;
	}

	// Supprimer un membre

	public static void deleteMembre(int id) {
		for (int i = 0; i < arrMembre.size(); i++) {
			if (arrMembre.get(i).getId_Membre() == id) {
				arrMembre.remove(i);
			} else {
				System.out.println("Ce membre n'existe pas");
				break;
			}
		}
		System.out.println("====================================");

	}

	// Modifier un membre

	public static void modifierMembre() {
		System.out.println("====================================");
		System.out.println("Entrer ID membre pour la modification: ");
		int id = sc.nextInt();
		sc.nextLine();
		for (int i = 0; i < arrMembre.size(); i++) {
			if (arrMembre.get(i).getId_Membre() == id) {
				System.out.println("Changer le nom de la personne: ");
				String nom = sc.nextLine();
```

```
                        arrMembre.get(i).setNomMembre(nom);
                } else {
                        System.out.println("Ce membre n'existe pas");
                        break;
                }
        }
        System.out.println("=====================================");
}

// Lister tous les membres

public static void showAllMembre() {
        System.out.println("=====================================");
        for (int i = 0; i < arrMembre.size(); i++) {
                System.out.println(arrMembre.get(i).toString());
                System.out.println("-------------------------");
        }
        System.out.println("=====================================");
}

// Ajouter une tache

public static boolean addTache() {
        System.out.println("=====================================");
        System.out.println("Entrer ID tache: ");
        int id = sc.nextInt();
        sc.nextLine();
        System.out.println("Entrer le nom de tache: ");
        String nom = sc.nextLine();
        System.out.println("Entrer la description de tache: ");
        String description = sc.nextLine();
        System.out.println("Entrer le status de tache: ");
        String status = sc.nextLine();
        Tache ta = new Tache(id, nom, description, status);
        arrTache.add(ta);


        System.out.println("=====================================");
        return true;
}

// Supprimer une tache

public static void deleteTache(int id) {
        System.out.println("=====================================");
        System.out.println("Entrer ID de tache: ");
        for (int i = 0; i < arrTache.size(); i++) {
                if (arrTache.get(i).getId_tache() == id) {
                        arrTache.remove(i);
                } else {
                        System.out.println("Ce tache n'existe pas");
                        break;
                }
        }
        System.out.println("=====================================");

}

// Modifier une tache

public static void modifierTache() {
        System.out.println("=====================================");
```

```
            System.out.println("Entrer ID pour la modification: ");
            int id = sc.nextInt();
            sc.nextLine();
            for (int i = 0; i < arrTache.size(); i++) {
                  if (arrTache.get(i).getId_tache() == id) {
                        System.out.println("Entrer le nouveau nom de tache: ");
                        String nom = sc.nextLine();
                        System.out.println("Entrer la nouvelle description de
tache: ");
                        String description = sc.nextLine();
                        System.out.println("Entrer le nouveau status de tache:
");
                        String status = sc.nextLine();
                        arrTache.get(i).setNom(nom);
                        arrTache.get(i).setDescription(description);
                        arrTache.get(i).setStatus(status);
                  } else {
                        System.out.println("Ce tache n'existe pas");
                        break;
                  }
            }
            System.out.println("====================================");
                  }

      public static void showAllTache() {

            for (int i = 0; i < arrTache.size(); i++) {
                  System.out.println(arrTache.get(i).toString());
                  System.out.println("--------------------------");
            }
      }

      public static void assigneTache() {
            System.out.println("====================================");
            System.out.println("Entrer ID Assigne: ");
            int idAssigne = sc.nextInt();

            System.out.println("Entrer ID tache: ");
            int idTache = sc.nextInt();
            Tache ta = new Tache();

            for (int i = 0; i < arrTache.size(); i++) {
                  if (arrTache.get(i).getId_tache() == idTache) {
                        ta.setId_tache(idTache);
                        ta.setNom(arrTache.get(i).getNom());
                        ta.setDescription(arrTache.get(i).getDescription());
                        ta.setStatus(arrTache.get(i).getStatus());

                  }
            }
            System.out.println("Entrer ID du membre: ");
            int idMembre = sc.nextInt();

            Membre mem = new Membre();

            for (int i = 0; i < arrMembre.size(); i++) {
                  if (arrMembre.get(i).getId_Membre() == idMembre) {
                        mem.setId_Membre(idMembre);
                        mem.setNomMembre(arrMembre.get(i).getNomMembre());

                  }
            }
```

```
		Assigne ass = new Assigne(idAssigne, mem, ta);
		arrAssigne.add(ass);
		System.out.println("====================================");
	}

	public static void showAllAssigne() {

		for (int i = 0; i < arrAssigne.size(); i++) {
			System.out.println(arrAssigne.get(i).show());
			System.out.println("--------------------------");
		}
	}

	// Lister les taches d'une personne

	public static void show_Own_Tache() {
		System.out.println("Entrer ID membre pour la recherche: ");
		int id = sc.nextInt();
		System.out.println("====================================");
		System.out.println("Les taches du membre ID: " + id);
		for (int j = 0; j < arrAssigne.size(); j++) {
			if (id == arrAssigne.get(j).getMembre().getId_Membre()) {
				System.out.println(arrAssigne.get(j).getTache());
			}
		}
		System.out.println("====================================");
	}

	// Chercher une tache par status

	public static void searchTache() {
		System.out.println("Entrer le status pour la recherche: ");
		String status_tache;
		status_tache = sc.nextLine();
		System.out.println("====================================");
		System.out.println("Les taches avec status " + "(" + status_tache +
"): ");
		for (int j = 0; j < arrTache.size(); j++) {
			if (status_tache.compareTo(arrTache.get(j).getStatus()) == 0)
{
				System.out.println(arrTache.get(j).toString());
				break;
			}
		}
		System.out.println("====================================");
	}

	public static String menu() {
		System.out.println("===>Gestion des Taches<===");
		System.out.println("1. Ajouter un membre.");
		System.out.println("2. Supprimer un membre");
		System.out.println("3. Modifier un membre");
		System.out.println("4. Lister tous les membres");
		System.out.println("5. Ajouter une tache.");
		System.out.println("6. Supprimer une tache");
		System.out.println("7. Modifier une tache");
		System.out.println("8. Lister tous les taches");
		System.out.println("9. Assigner une tache");
		System.out.println("10. Lister tous les assigne");
		System.out.println("11. Lister tous les taches d'un membre");
		System.out.println("12. Chercher une tache par status");
```

```
		System.out.println("Choisissez une option ci-dessus:");

		return sc.nextLine();
	}

	public static boolean verifyInput(Object val) {
		boolean rep = true;
		try {
			int v = Integer.parseInt(val.toString());
			return rep = true;
		} catch (Exception ex) {
			System.out.println("Valeur Saisie invalide !");
			return rep = false;
		}
	}

	public void lancer() {
		String choix = "";
		do {
			choix = menu();
			if (verifyInput(choix))
				operation(Integer.parseInt(choix));
			else
				choix = "";

		} while((choix != "1") || (choix != "2") || (choix != "3")  ||
(choix != "4")|| (choix != "5")|| (choix != "6")|| (choix != "7")|| (choix !=
"8")|| (choix != "9")|| (choix != "10")|| (choix != "11") || (choix != "12"));
	}

	public static void operation(int key) {
		switch (key) {
		case 1:
			System.out.println("Ajouter un membre:");
			addMembre();
			break;
		case 2:
			System.out.println("Supprimer un membre:");
			System.out.println("====================================");
			System.out.println("Entrer ID membre pour l'élimination: ");
			deleteMembre(sc.nextInt());
			break;
		case 3:
			System.out.println("Modifier un membre");
			modifierMembre();
			break;
		case 4:
			System.out.println("Lister tous les membres:");
			showAllMembre();
			break;
		case 5:
			System.out.println("Ajouter une tache:");
			addTache();
			break;
		case 6:
			System.out.println("Supprimer une tache:");
			deleteTache(sc.nextInt());
			break;
		case 7:
			System.out.println("Modifier une tache:");
			modifierTache();
			break;
```

```
			case 8:
				System.out.println("Lister tous les taches:");
				showAllTache();
				break;
			case 9:
				System.out.println("Assigner une tache:");
				assigneTache();
				break;
			case 10:
				System.out.println("Lister tout les assignes:");
				showAllAssigne();
				break;
			case 11:
				System.out.println("Lister tous les taches d'un membre:");
				show_Own_Tache();
				break;
			case 12:
				System.out.println("Chercher une tache par status:");
				searchTache();
				break;
			case 13:
				break;
			}
		}

		public static void main(String[] args) {
			String choix = "";
			do {
				choix = menu();
				if (verifyInput(choix))
					operation(Integer.parseInt(choix));
				else
					choix = "";

			} while ((choix != "1") || (choix != "2") || (choix != "3")  ||
(choix != "4")|| (choix != "5")|| (choix != "6")|| (choix != "7")|| (choix !=
"8")|| (choix != "9")|| (choix != "10")|| (choix != "11") || (choix != "12"));
		}

}
```

2. Classe Membre

```
public class Membre {

		private int Id_Membre;
		private String NomMembre;


		public Membre() {

		}

			public Membre(int Id_Membre, String NomMembre) {
				this.Id_Membre = Id_Membre;
				this.NomMembre = NomMembre;
		}

			public int getId_Membre() {
				return Id_Membre;
		}

			public void setId_Membre(int Id_Membre) {
```

```
		this.Id_Membre = Id_Membre;
	}

		public String getNomMembre() {
			return NomMembre;
	}

		public void setNomMembre(String NomMembre) {
			this.NomMembre = NomMembre;
	}

	@Override
		public String toString() {
			return "Membre [Id_Membre=" + Id_Membre + ", NomMembre=" +
NomMembre + "]";
	}


}
```

3. Classe Tache

```
public class Tache {

	private int Id_tache;
	private String Nom;
	private String Description;
	private String Status;


		public Tache() {

	}

		public Tache(int Id_tache, String Nom, String Description, String
Status) {
			this.Id_tache = Id_tache;
			this.Nom = Nom;
			this.Description = Description;
			this.Status = Status;
	}

		public int getId_tache() {
			return Id_tache;
	}

		public void setId_tache(int Id_tache) {
			this.Id_tache = Id_tache;
	}

		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;
	}

	@Override
	public String toString() {
		return "Tache [Id_tache=" + Id_tache + ", Nom=" + Nom + " ,
Description=" + Description + " , Status=" + Status + "]";
	}


}
```

4. Classe Assigne

```
public class Assigne {

	private int ID_Assigne;
	private Membre membre;
	private Tache tache;

	public Assigne(int iD_Assigne, Membre membre, Tache tache) {
		super();
		ID_Assigne = iD_Assigne;
		this.membre = membre;
		this.tache = tache;
	}

	public Assigne() {
		super();
	}

	public int getID_Assigne() {
		return ID_Assigne;
	}

	public void setID_Assigne(int iD_Assigne) {
		ID_Assigne = iD_Assigne;
	}

	public Membre getMembre() {
		return membre;
	}

	public void setMembre(Membre membre) {
		this.membre = membre;
	}

	public Tache getTache() {
		return tache;
	}

	public void setTache(Tache tache) {
```

```
		this.tache = tache;
	}

	public String show(){
		return  "ID Assigne: " + ID_Assigne  + "\n" +
				"Membre: " + membre.toString() + "\n" +
				"Tache: " + tache.toString() + "\n";

	}
}
```

5. Classe Junitmain

```
import static org.junit.Assert.*;

import org.junit.AfterClass;
import org.junit.Test;


public class Junitmain {

	@AfterClass
	public static void tearDownAfterClass() throws Exception {
	}

	@Test
	public void test() {
		//fail("Not yet implemented");
	main principale= new main();
	boolean output = principale.addMembre();
	assertEquals(true,output);
	}


}
```

6. Classe Junitmaintest2

```
import static org.junit.Assert.*;

import org.junit.AfterClass;
import org.junit.Test;


public class Junitmaintest2 {

	@AfterClass
	public static void tearDownAfterClass() throws Exception {
	}

	@Test
	public void test() {
		//fail("Not yet implemented");
		main principale= new main();
		boolean output = principale.addTache();
```

```
            assertEquals(true,output);
        }

}
```

7. Classe AllTests

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


@RunWith(Suite.class)
@SuiteClasses({ Junitmain.class, Junitmaintest2.class })
public class AllTests {

}
```