

UNIVERSITÉ NATIONALE DU VIETNAM À HANOÏ  
INSTITUT FRANCOPHONE INTERNATIONAL



ĐẠI HỌC QUỐC GIA HÀ NỘI  
**VNU**  
*Since 1906*



INSTITUT  
FRANCOPHONE  
INTERNATIONAL

**Rapport du TP1 du cours de Génie Logiciel Avancé**

Classe : Master1

Option : Systèmes Intelligents et Multimédia (SIM)

Par

NIYONKURU Méthode  
Promotion : 21

Nom du Professeur : Dr. Ho Tuong Vinh

Année académique : 2016-2017

## Table des matières

1.Introduction.....	1
2.Spécification.....	1
3.Conception.....	1
3.1.Diagramme de cas d'utilisation.....	1
3.2.Diagramme de classe.....	2
3.3.Diagramme de séquences (méthode Suppression d'un membre).....	3
4.Implémentation et tests.....	4
4.1.Implémentation.....	4
4.2.Tests.....	4
5. Conclusion.....	10
6. Annexes.....	11
Références.....	23

## **1.Introduction**

Pour ce premier travail pratique (TP1) du cours de génie logiciel avancé, nous allons concevoir et réaliser une application de gestion de tâches pour une équipe de travail afin de nous rappeler les concepts de la modélisation avec UML et programmation orientée- objet avec Java. De plus, il a pour but de nous familiariser à un environnement de développement intégré (IDE) « libre » (Open Source) ECLIPSE. Il nous est aussi demandé d'utiliser GitHub (github.com) pour la gestion de code source de notre projet. Le présent rapport montre les étapes de la conception et de la réalisation de ladite application. Le code de différentes tables de notre application est en annexe du présent rapport.

## **2.Spécification**

Les spécifications de l'application à réaliser sont les suivantes :

L'utilisateur gère les membres et les tâches pour une équipe de travail. Ce gestionnaire fournit à l'utilisateur les fonctionnalités suivantes :

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

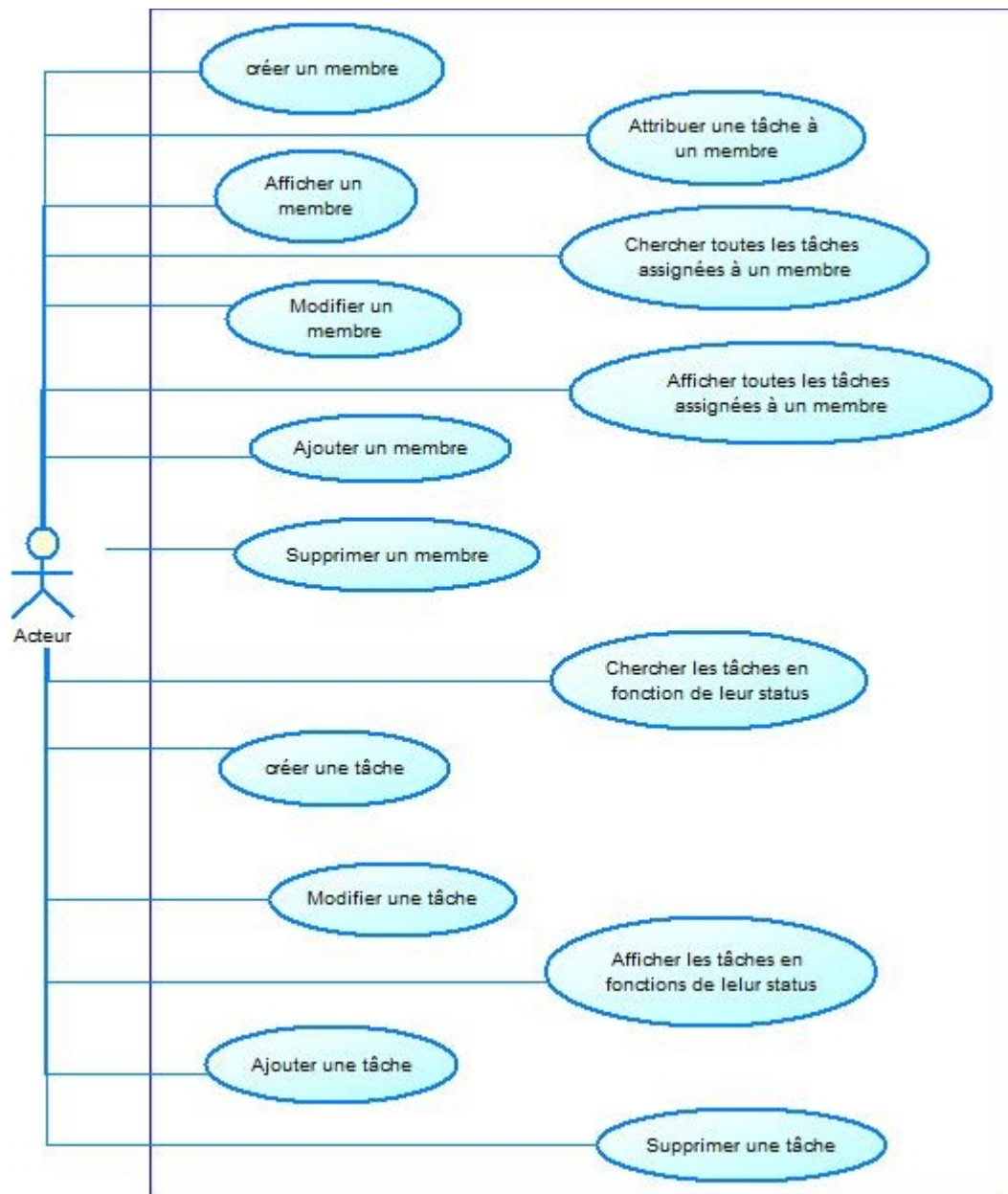
## **3.Conception**

Pour la conception de notre logiciel nous voulons mettre en œuvre un ensemble d'activités qui à partir d'une demande d'informatisation d'un processus (demande qui peut aller de la simple question orale jusqu'au cahier des charges complet) permettent la conception, l'écriture et la mise au point de notre application qui va répondre au besoins de l'utilisateur(acteur en UML) qui sont décrits dans les spécifications ci-haut mentionnées.

Les diagrammes suivants sauront vous montrer d'amples explications.

### **3.1.Diagramme de cas d'utilisation**

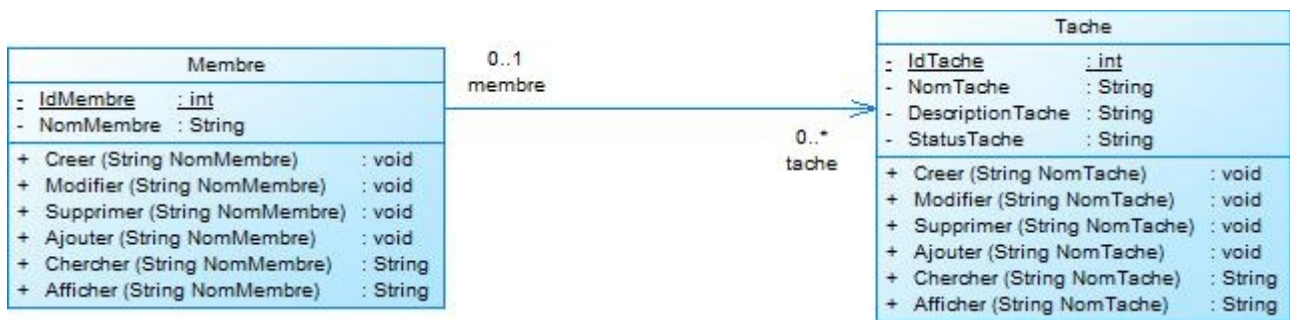
Le diagramme de cas d'utilisation est un modèle de haut niveau destiné à concevoir les besoins et le comportement d'un système. Ce diagramme modélise les fonctionnalités du système telles qu'elles sont perçues par les utilisateurs externes appelés acteurs qui effectuent des tâches définies comme cas d'utilisation.



**Figure1 : Diagramme de cas d'utilisation**

### 3.2.Diagramme de classe

Les diagrammes de classes[1] représentent les notions d'analyse d'un système (ou bien les notions supportées par un système), avec leurs propriétés et leur relations. D'un point de vue "haut niveau" (conceptuel), les classes représentent les concepts supportés par un système, tandis que d'un point de vue "bas niveau" (physique), elles peuvent représenter les classes implémentées par un langage objet. Un diagramme de classes a pour rôle de définir un ensemble de tous les états possibles et vérifier les contraintes. La figure ci-dessous montre comment nous avons représenté notre diagramme de classe en utilisant le logiciel PowerAMC.



**Figure2 : Diagramme de classes**

### 3.3. Diagramme de séquences (méthode Suppression d'un membre)

Les principales informations contenues dans un diagramme de séquence [2] et [3] sont les messages échangés entre les lignes de vie, présentés dans un ordre chronologique. Les messages échangés entre l'utilisateur et le système sont de trois types : -l'envoi d'un signal ; -l'invocation d'une opération ; -la création ou la destruction d'une instance.

La figure 3 représente le diagramme de séquences pour le cas de suppression d'un membre dans notre application.

#### Représentation des lignes de vie

Une ligne de vie se représente par un rectangle, auquel est accroché une ligne verticale pointillée, contenant une étiquette dont la syntaxe est :

[<nom\_du\_rôle>] : [<Nom\_du\_type>]

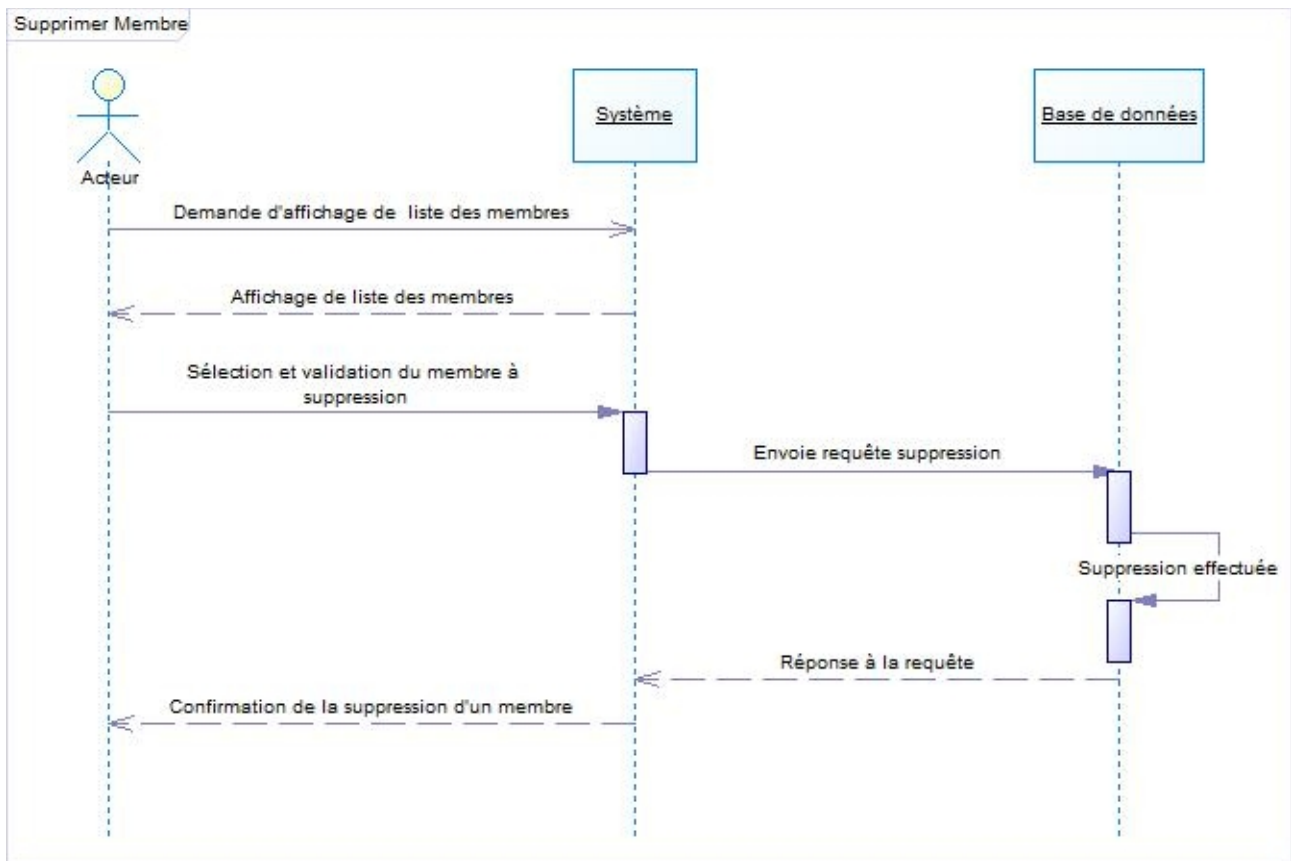
Au moins un des deux noms doit être spécifié dans l'étiquette, les deux points (:) sont, quand à eux, obligatoire.

#### Syntaxe des messages et des réponses

Dans la plupart des cas, la réception d'un message est suivie de l'exécution d'une méthode d'une classe. Cette méthode peut recevoir des arguments et la syntaxe des messages permet de transmettre ces arguments.

- la direction du message est directement spécifiée par la direction de la flèche qui matérialise le message, et non par une flèche supplémentaire au dessus du connecteur reliant les objets ;

La figure 3 suivante représente le diagramme de séquences pour la suppression d'un membre.



**Figure 3 : Diagramme de séquences (méthode Suppression d'un membre)**

signalons que la figure 3 qui est le diagramme de séquence pour notre application a été effectuée tout en supposant qu'on a utilisé une base de données pour stocker les informations. Dans le disque de l'ordinateur. Mais pour les tests du titres (4.2.Tests), nous avons fait l'exécution de notre application dans le mode **console**.

## 4.Implémentation et tests

### 4.1.Implémentation

L'application a été développée sous Linux (16.04) avec le langage de programmation **JAVA** et grâce à l'IDE **Netbeans**.

Dans l'implémentation de notre application nous avons créer 4 classes à savoir :

**1- Classe Gestionnaire** : La classe principale permet la compilation pour notre application

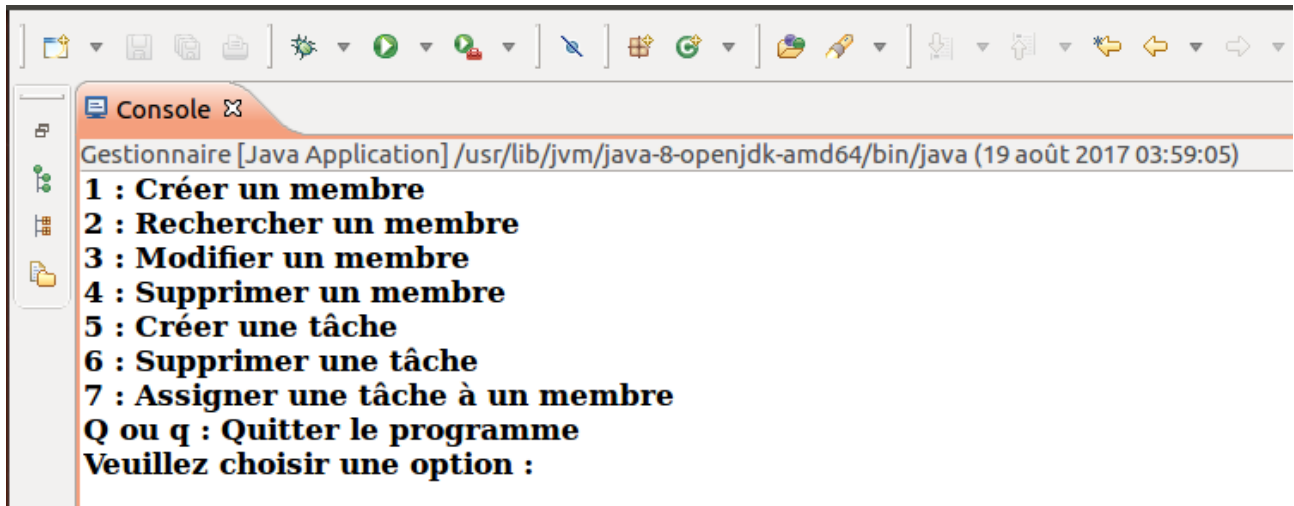
**2- Classe Membre** qui définit les propriétés d'un membre

**3-La classe tâche** tâche gère les propriétés d'une tâches

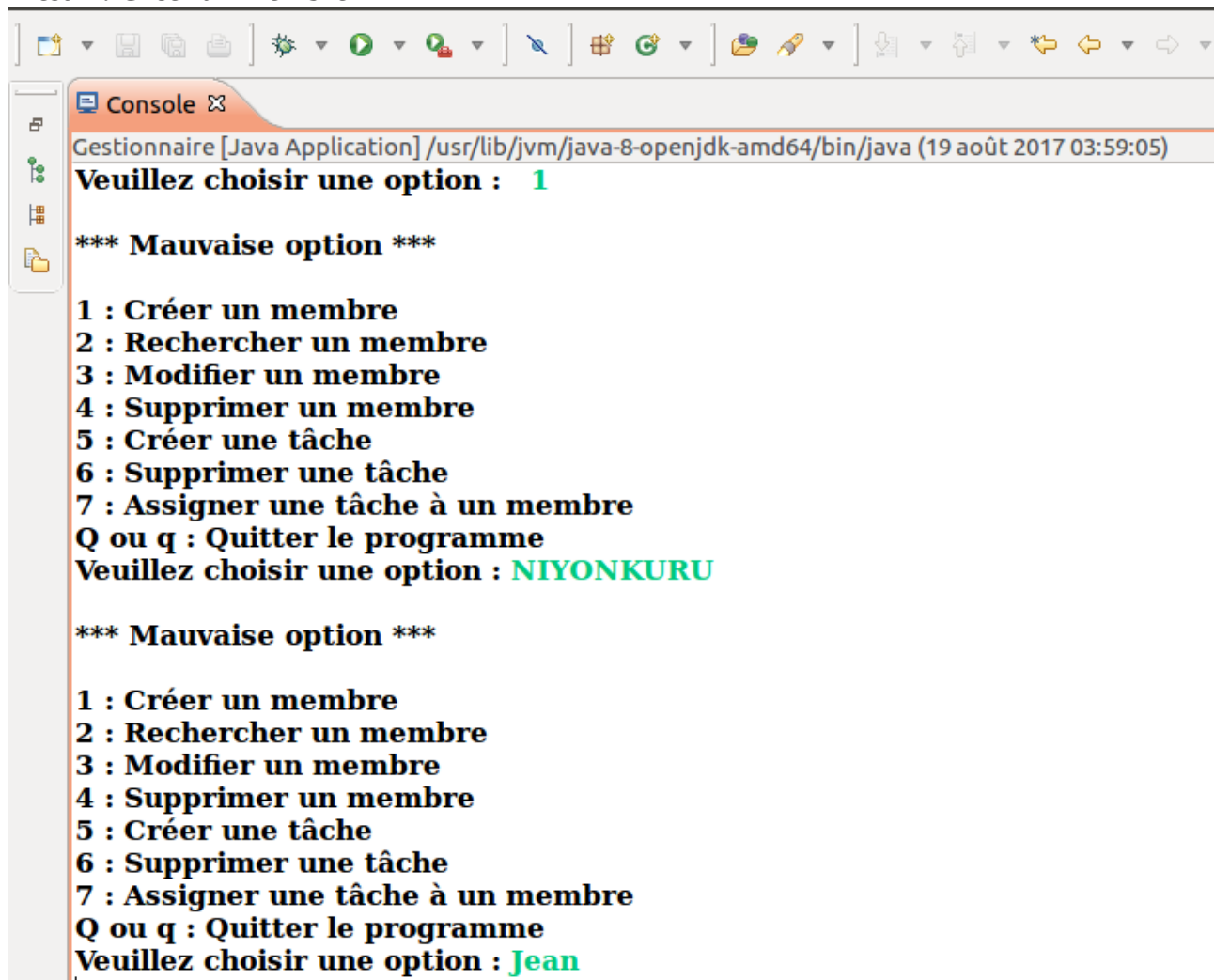
**4-La classe assignation** La classe assignation

## 4.2.Tests

Les tests réalisés sur notre application sont résumés par les captures d'écrans suivantes:  
Avant de commencer les tests prosternements dits, nous aimerions vous présenter les menus de notre application



## -Test 1 : Créer un membre



```
Gestionnaire [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (19 août 2017 03:59:05)
Veillez choisir une option : 1

*** Mauvaise option ***

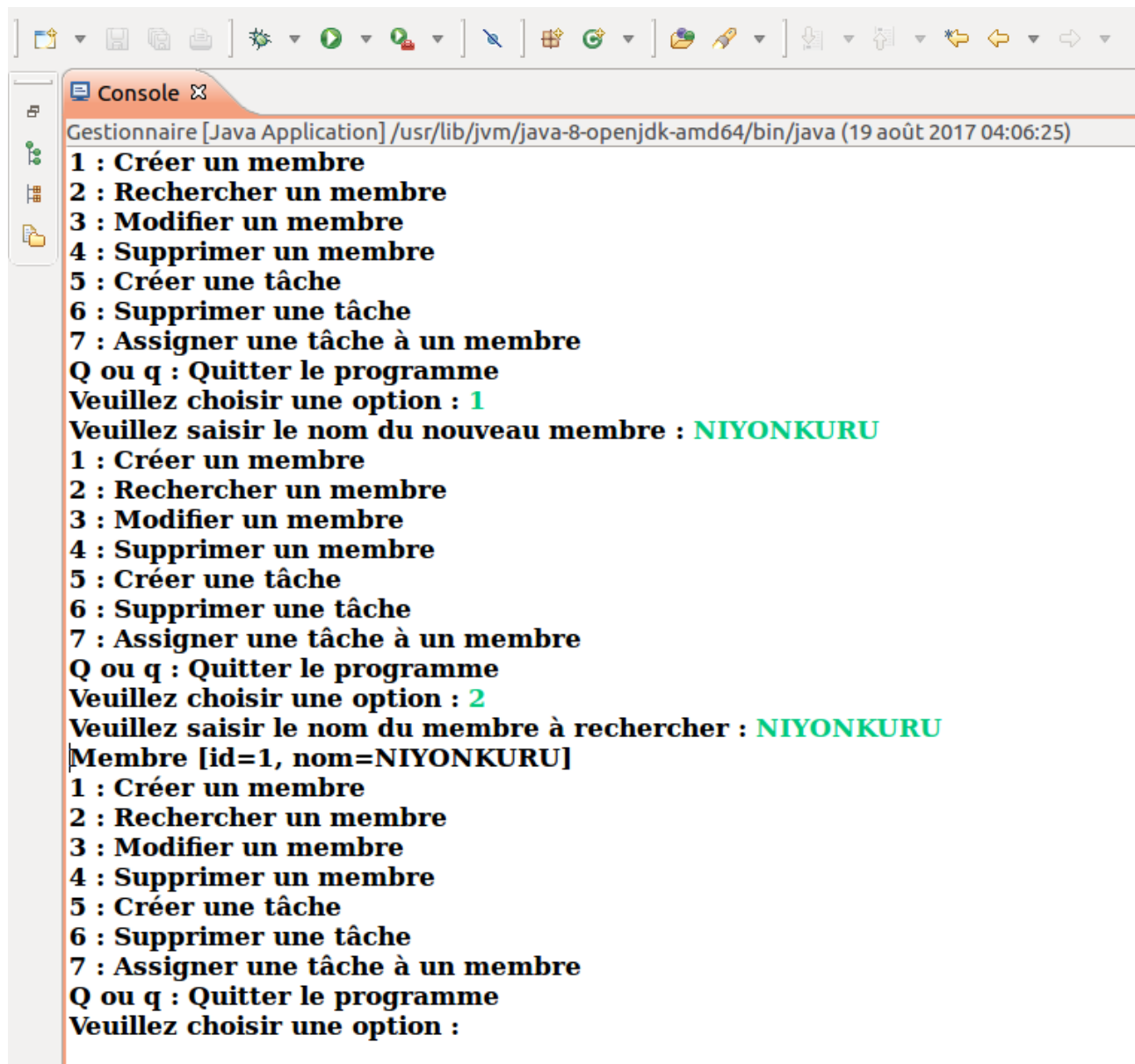
1 : Créer un membre
2 : Rechercher un membre
3 : Modifier un membre
4 : Supprimer un membre
5 : Créer une tâche
6 : Supprimer une tâche
7 : Assigner une tâche à un membre
Q ou q : Quitter le programme
Veillez choisir une option : NIYONKURU

*** Mauvaise option ***

1 : Créer un membre
2 : Rechercher un membre
3 : Modifier un membre
4 : Supprimer un membre
5 : Créer une tâche
6 : Supprimer une tâche
7 : Assigner une tâche à un membre
Q ou q : Quitter le programme
Veillez choisir une option : Jean
```

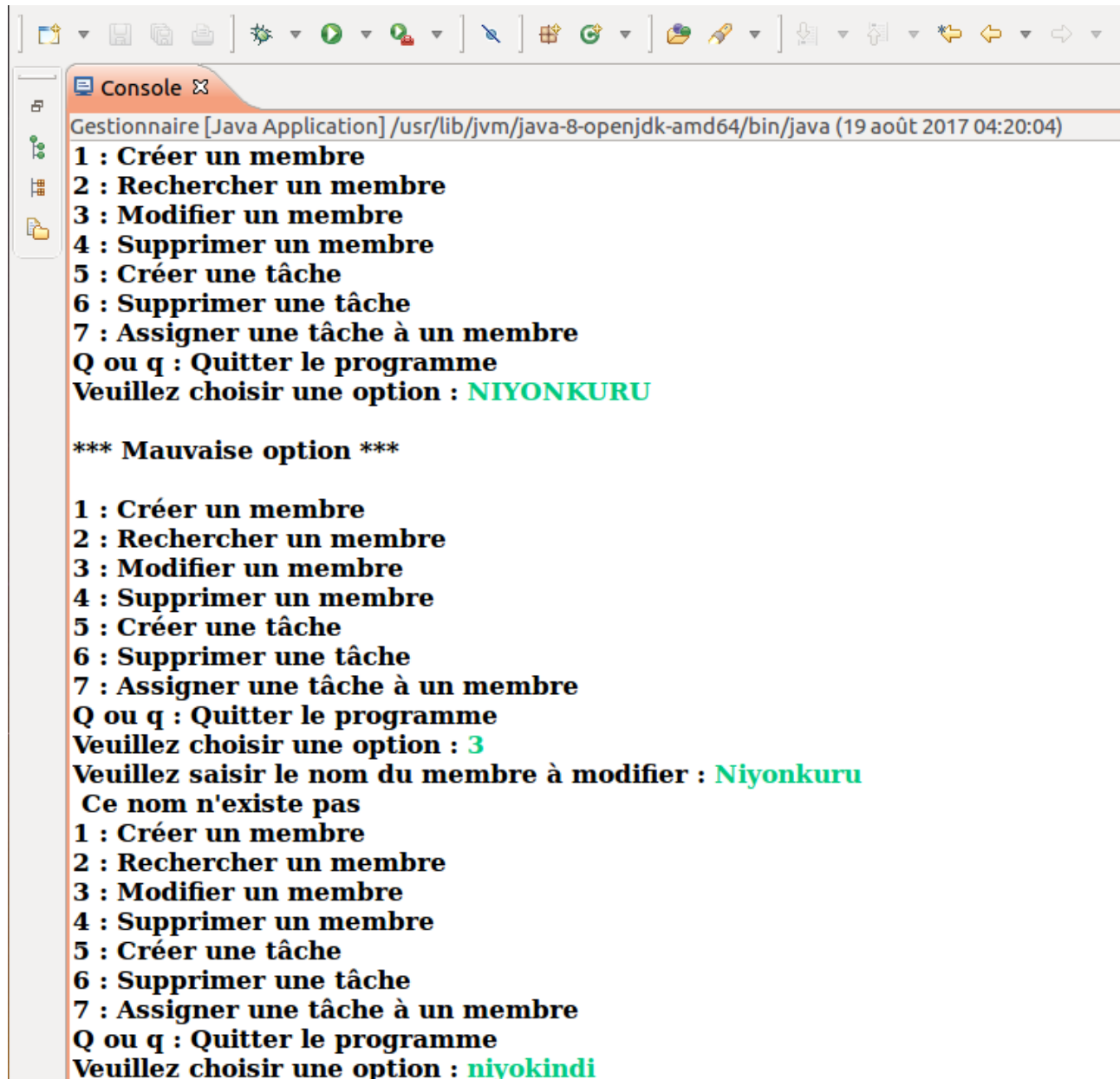


-Test 2 : Rechercher un membre



```
Gestionnaire [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (19 août 2017 04:06:25)
1 : Créer un membre
2 : Rechercher un membre
3 : Modifier un membre
4 : Supprimer un membre
5 : Créer une tâche
6 : Supprimer une tâche
7 : Assigner une tâche à un membre
Q ou q : Quitter le programme
Veuillez choisir une option : 1
Veuillez saisir le nom du nouveau membre : NIYONKURU
1 : Créer un membre
2 : Rechercher un membre
3 : Modifier un membre
4 : Supprimer un membre
5 : Créer une tâche
6 : Supprimer une tâche
7 : Assigner une tâche à un membre
Q ou q : Quitter le programme
Veuillez choisir une option : 2
Veuillez saisir le nom du membre à rechercher : NIYONKURU
[Membre [id=1, nom=NIYONKURU]
1 : Créer un membre
2 : Rechercher un membre
3 : Modifier un membre
4 : Supprimer un membre
5 : Créer une tâche
6 : Supprimer une tâche
7 : Assigner une tâche à un membre
Q ou q : Quitter le programme
Veuillez choisir une option :
```

### -Test 3 : Modifier un membres



```
Gestionnaire [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (19 août 2017 04:20:04)
1 : Créer un membre
2 : Rechercher un membre
3 : Modifier un membre
4 : Supprimer un membre
5 : Créer une tâche
6 : Supprimer une tâche
7 : Assigner une tâche à un membre
Q ou q : Quitter le programme
Veillez choisir une option : NIYONKURU

*** Mauvaise option ***

1 : Créer un membre
2 : Rechercher un membre
3 : Modifier un membre
4 : Supprimer un membre
5 : Créer une tâche
6 : Supprimer une tâche
7 : Assigner une tâche à un membre
Q ou q : Quitter le programme
Veillez choisir une option : 3
Veillez saisir le nom du membre à modifier : Niyonkuru
Ce nom n'existe pas
1 : Créer un membre
2 : Rechercher un membre
3 : Modifier un membre
4 : Supprimer un membre
5 : Créer une tâche
6 : Supprimer une tâche
7 : Assigner une tâche à un membre
Q ou q : Quitter le programme
Veillez choisir une option : niyokindi
```

-Test4 : Suppression d'un membre

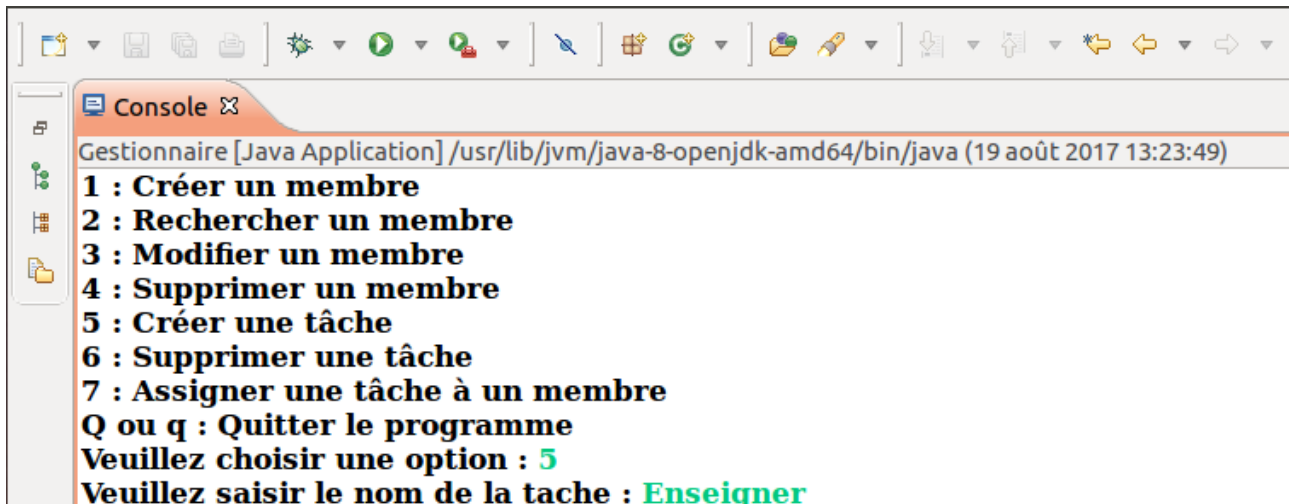
```

Gestionnaire [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (19 août 2017 04:29:00)
7 : Assigner une tâche à un membre
Q ou q : Quitter le programme
Veillez choisir une option : 4
Veillez saisir le nom du membre a supprimer: Niyonkuru
Ce nom n'existe pas déjà
1 : Créer un membre
2 : Rechercher un membre
3 : Modifier un membre
4 : Supprimer un membre
5 : Créer une tâche
6 : Supprimer une tâche
7 : Assigner une tâche à un membre
Q ou q : Quitter le programme
Veillez choisir une option : Niyonkuru

*** Mauvaise option ***

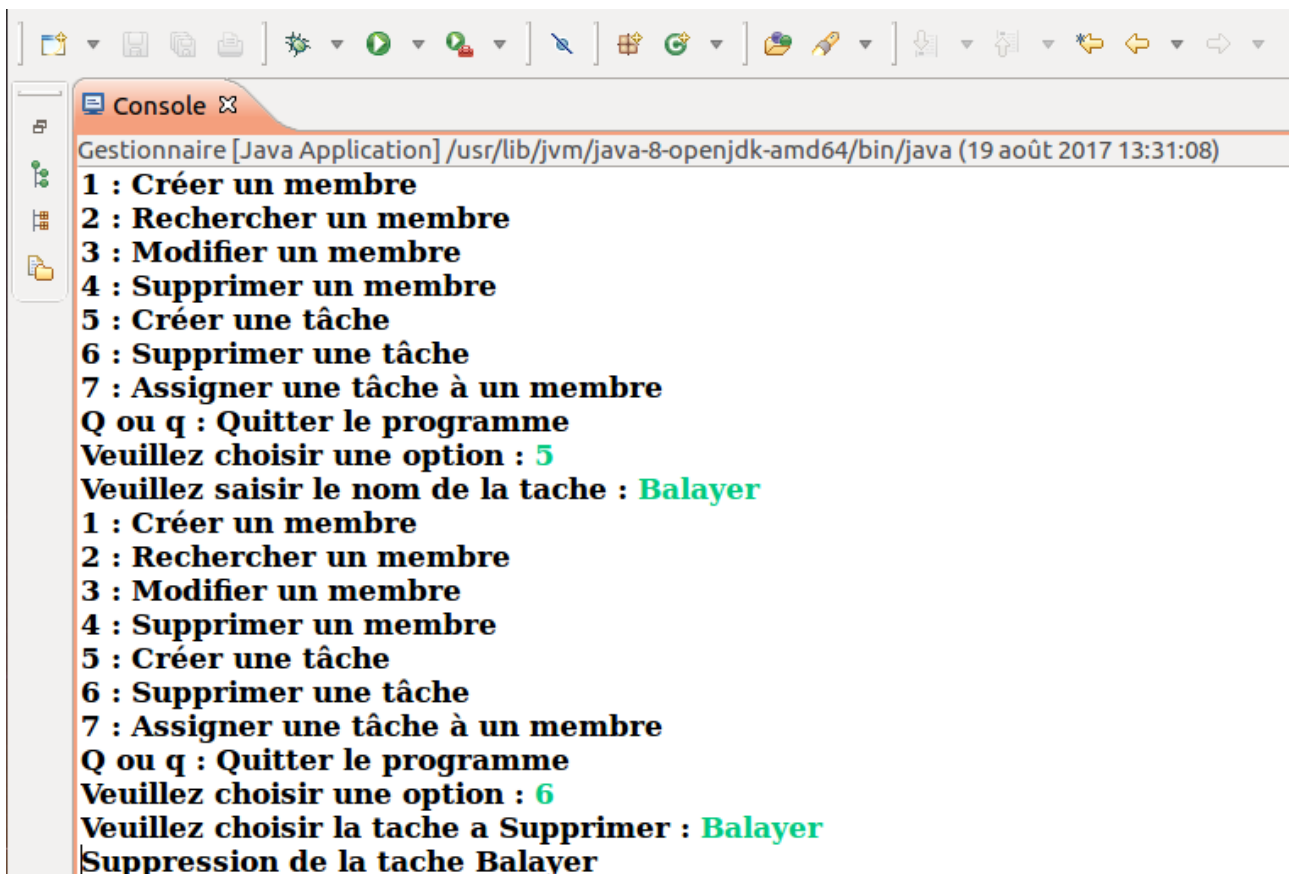
1 : Créer un membre
2 : Rechercher un membre
3 : Modifier un membre
4 : Supprimer un membre
5 : Créer une tâche
6 : Supprimer une tâche
7 : Assigner une tâche à un membre
Q ou q : Quitter le programme
Veillez choisir une option : 2
Veillez saisir le nom du membre à rechercher : Niyonkuru
Ce nom n'existe pas!
1 : Créer un membre
2 : Rechercher un membre
3 : Modifier un membre
4 : Supprimer un membre
5 : Créer une tâche
6 : Supprimer une tâche
7 : Assigner une tâche à un membre
Q ou q : Quitter le programme
Veillez choisir une option :
```

-Test 5 : Créer une tâche



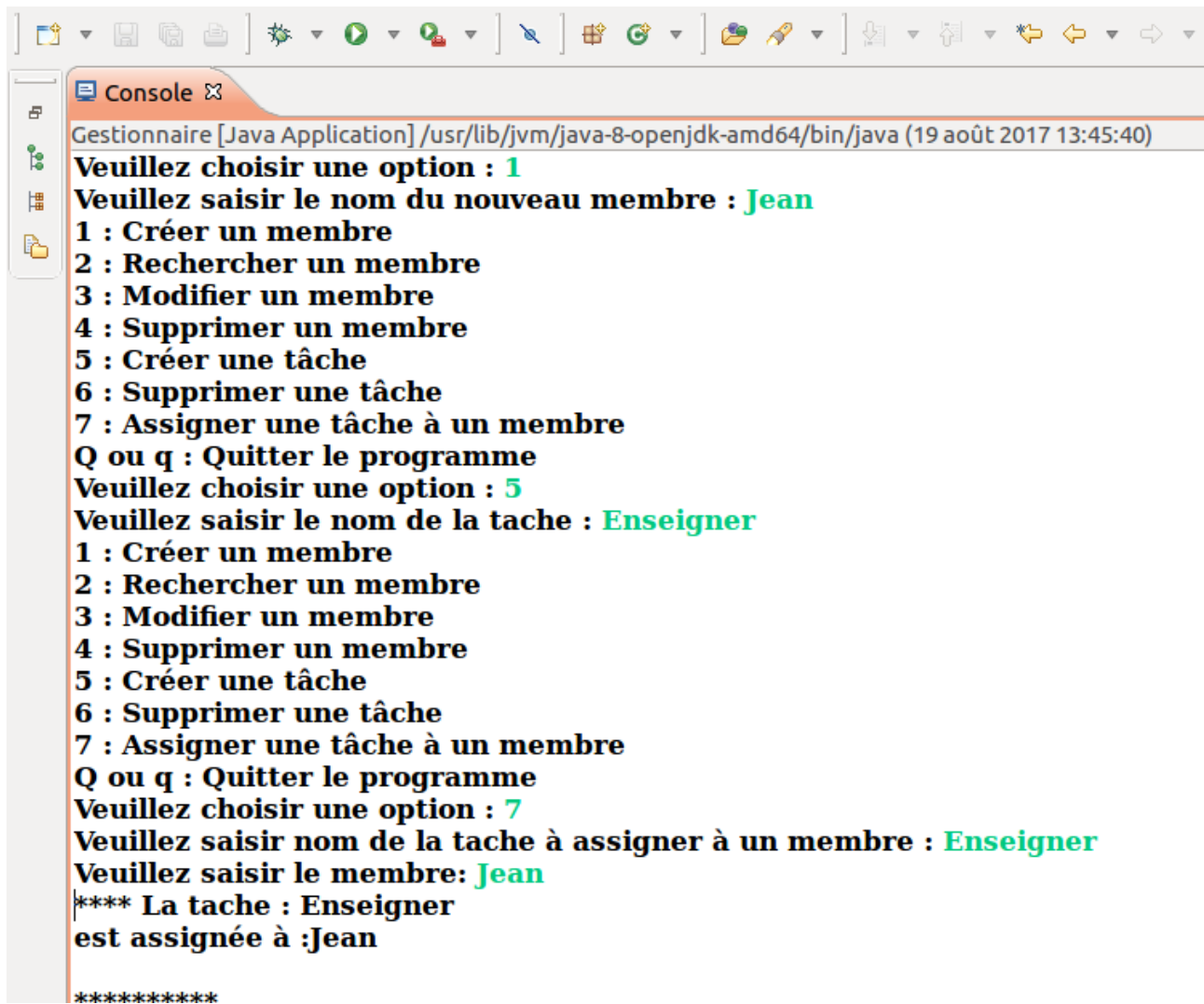
```
Gestionnaire [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (19 août 2017 13:23:49)
1 : Créer un membre
2 : Rechercher un membre
3 : Modifier un membre
4 : Supprimer un membre
5 : Créer une tâche
6 : Supprimer une tâche
7 : Assigner une tâche à un membre
Q ou q : Quitter le programme
Veuillez choisir une option : 5
Veuillez saisir le nom de la tache : Enseigner
```

-Test 6: Suppression d'une tâche



```
Gestionnaire [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (19 août 2017 13:31:08)
1 : Créer un membre
2 : Rechercher un membre
3 : Modifier un membre
4 : Supprimer un membre
5 : Créer une tâche
6 : Supprimer une tâche
7 : Assigner une tâche à un membre
Q ou q : Quitter le programme
Veuillez choisir une option : 5
Veuillez saisir le nom de la tache : Balayer
1 : Créer un membre
2 : Rechercher un membre
3 : Modifier un membre
4 : Supprimer un membre
5 : Créer une tâche
6 : Supprimer une tâche
7 : Assigner une tâche à un membre
Q ou q : Quitter le programme
Veuillez choisir une option : 6
Veuillez choisir la tache a Supprimer : Balayer
Suppression de la tache Balayer
```

## -Test 7 : Assigner une tâche à un membre



```
Gestionnaire [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (19 août 2017 13:45:40)
Veillez choisir une option : 1
Veillez saisir le nom du nouveau membre : Jean
1 : Créer un membre
2 : Rechercher un membre
3 : Modifier un membre
4 : Supprimer un membre
5 : Créer une tâche
6 : Supprimer une tâche
7 : Assigner une tâche à un membre
Q ou q : Quitter le programme
Veillez choisir une option : 5
Veillez saisir le nom de la tache : Enseigner
1 : Créer un membre
2 : Rechercher un membre
3 : Modifier un membre
4 : Supprimer un membre
5 : Créer une tâche
6 : Supprimer une tâche
7 : Assigner une tâche à un membre
Q ou q : Quitter le programme
Veillez choisir une option : 7
Veillez saisir nom de la tache à assigner à un membre : Enseigner
Veillez saisir le membre: Jean
**** La tache : Enseigner
est assignée à :Jean
*****
```

## 5. Conclusion

Les objectifs fixés pour ce premier travail pratique (TP1) du cours de génie logiciel avancé ont été atteints. Les spécifications énumérées plus haut ont été respectées et notre application est fonctionnelle. Nous avons ainsi amélioré notre compréhension de la programmation orientée objet avec java et les concepts de modélisation avec UML. De plus, nous nous sommes familiarisé à un environnement de développement intégré (IDE) « libre » (Open Source) ECLIPSE. L'utilisation de GitHub (github.com) pour la gestion de code source de notre projet a été mise en place. Nous devons continuer à fournir des efforts afin d'améliorer la réflexion de la programmation orientée objet en général et de la maîtrise du langage java en particulier.

## 6. Annexes

### 1- Classe Gestionnaire

//La classe principale Gestionnaire permet la compilation pour notre application

**import** java.util.ArrayList;

**import** java.util.Scanner;

**public class** Gestionnaire {

/\*\*

\* @param args

\*/

**static** ArrayList<Membre> *membres* = **new** ArrayList<>();

**static** ArrayList<Tache> *taches* = **new** ArrayList<>();

**private static** Scanner *sc*;

// les methodes de la classe membre

// méthode de recherche d'un membre

**public static** Membre recherche\_membre(String nom) {

    Membre res = **null**;

**for** (Membre m : *membres*) {

**if** (m.getNom().equalsIgnoreCase(nom)) {

            res = m;

**break**;

        }

    }

**return** res;

}

// méthode de modification d'un membre

**public static void** editMembre(**int** index, String nom) {

    Membre m3 = *recherche\_membre*(nom);

**if** (m3 != **null**) {

        System.out.println("Ce nom existe déjà");

    } **else** {

*membres*.set(index, m3);

    }

}

// méthode d'ajout d'un membre

**public static void** addMembre(Membre m) {

    Membre m2 = *recherche\_membre*(m.getNom());

**if** (m2 != **null**) {

        System.out.println("Ce nom existe déjà");

```

    } else {
        m.setId(membres.size()+1);
        membres.add(m);
    }
}

// méthode de suppression d'un membre
public static void delMembre(String nom) {
    Membre m3 = recherche_membre(nom);
    if (m3 != null) {
        membres.remove(m3);
    } else {
        System.out.println("Ce nom n'existe pas déjà");
    }
}

// Debut des Methodes de la classe tache

// Recherche d'une tache
public static Tache recherche_tache(String nom_tache) {
    Tache res = null;
    for (Tache t : taches) {
        if (t.getNomt().equalsIgnoreCase(nom_tache)) {
            res = t;
            break;
        }
    }
    return res;
}

// méthode de modification d'une tache
public static void editTache(int index, String nomt) {
    Tache t = recherche_tache(nomt);
    if (t != null) {
        System.out.println("Cette tache existe déjà");
    } else {
        taches.set(index, t);
    }
}

// méthode d'ajout d'une tache
public void addTache(Tache t) {
    Tache t2 = recherche_tache(t.getNomt());
    if (t2 != null) {
        System.out.println("Cette tache existe déjà");
    } else {

```



```

        taches.add(t);
    }
}

// méthode de suppression d'un membre
public void delTache(String nom) {
    Membre t3 = recherche_membre(nom);
    if (t3 != null) {
        taches.remove(t3);
    } else {
        System.out.println("Suppression de la tache avec
succès");
    }
}

public static void menu() {
    System.out.println("1 : Créer un membre");
    System.out.println("2 : Rechercher un membre");
    System.out.println("3 : Modifier un membre");
    System.out.println("4 : Supprimer un membre");
    System.out.println("5 : Créer une tâche");
    System.out.println("6 : Supprimer une tâche ");
    System.out.println("7 : Assigner une tâche à un membre");
    System.out.println("Q ou q : Quitter le programme");
}

// / fin des methodes de la classe tache

public static void main(String[] args) {
    // TODO Auto-generated method stub
    String nom = "";
    String choix = "";
    sc = new Scanner(System.in);
    do {
        menu();
        //Ajouter un membre
        System.out.print("Veuillez choisir une option : ");
        choix = sc.nextLine();
        if (choix.equals("1")) {
            System.out.print("Veuillez saisir le nom du
nouveau membre : ");
            nom = sc.nextLine();
            addMembre(new Membre(0, nom));
            // Rechercher et afficher un membre
        } else if (choix.equals("2")) {

```



```

        System.out.print("Veuillez saisir le nom du
membre à rechercher : ");
        nom = sc.nextLine();
        Membre m2 = recherche_membre(nom);
        if (m2 != null)
            System.out.println(m2.toString());
        else
            System.out.println("Ce nom n'existe pas!");

        // Modifier un membre
    } else if (choix.equals("3")) {
        System.out.print("Veuillez saisir le nom du
membre à modifier : ");
        nom = sc.nextLine();
        Membre m3 = recherche_membre(nom);
        if (m3 != null) {
            System.out.println(" veuillez entrer les
modifications");

            nom = sc.nextLine();
            System.out.println(" modifications
enregistrees");
        } else {
            System.out.println(" Ce nom n'existe pas");
        }
    }

    else if (choix.equals("4")) {
        System.out.print("Veuillez saisir le nom du
membre a supprimer: ");
        nom = sc.nextLine();
        delMembre(nom);
    }

    // Ajout tache
    else if (choix.equals("5")) {
        System.out.print("Veuillez saisir le nom de la tache
: ");

        nom = sc.nextLine();
    }

    else if (choix.equals("6")) {
        String tache_name;
        System.out.print("Veuillez choisir la tache a
Supprimer : ");

```

```

        tache_name = sc.nextLine();
        if (tache_name != null) {
            System.out.print("Suppression de la tâche " +
tache_name);
            System.out.print("\n*****\n");
        }
        else {
            System.out.print(" Cette tâche n' existe
pas");
            System.out.println("\n***** \n");
        }
    }
    else if (choix.equals("7")) {
        String tache2;
        String member1;
        System.out.print("Veuillez saisir nom de la tache à
assigner à un membre : ");
        member1 = sc.nextLine();
        System.out.print("Veuillez saisir le membre: ");
        tache2 = sc.nextLine();
        System.out.println("**** La tache : " +member1);
        System.out.println("est assignée à : " +tache2);
        System.out.println("\n***** \n");
    }

    else{
        System.out.println("\n*** Mauvaise option ***\n");
    }
} while (!choix.toUpperCase().equals("Q")); // q == pour
quitter le programme
}

}

```

## 2- classe Membre

//La Classe Membre qui gère les propriétés d'un membre

```

public class Membre {

    private int IdMembre;
    private String NomMembre;

```

```

// constructeurs
public Membre() {

}

public Membre(int id, String nom) {
    this.IdMembre = id;
    this.NomMembre = nom;
}

public int getId() {
    return IdMembre;
}

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

public String getNom() {
    return NomMembre;
}

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

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

```

### 3-La classe tache

//La Classe tache gère les propriétés d'une taches

```
public class Tache {  
  
    private int IdTache;  
    private String NomTache;  
    private String DescriptionTache;  
    private String StatusTache;  
  
    // constructeurs  
    public Tache() {  
  
    }  
  
    public Tache(int Idt, String Nomt, String Descriptiont, String  
Statust) {  
        this.IdTache = Idt;  
        this.NomTache = Nomt;  
        this.DescriptionTache = Descriptiont;  
        this.StatusTache = Statust;  
    }  
  
    public int getIdt() {  
        return IdTache;  
    }  
  
    public void setIdt(int Idt) {  
        this.IdTache = Idt;  
    }  
  
    public String getNomt() {  
        return NomTache;  
    }  
  
    public void setNomt(String Nomt) {  
        this.NomTache = Nomt;  
    }  
  
    public String getDescr() {  
        return DescriptionTache;  
    }  
  
    public void setDescr(String Descr) {
```

```

        this.DescriptionTache = Descr;
    }

    public String getStatus() {
        return StatusTache;
    }

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

    @Override
    public String toString() {
        return "Tache [Idt=" + IdTache + ", Nomt=" + NomTache +
        ", Descriptiont=" + DescriptionTache + ", Statust=" + StatusTache +
        "];"
    }

}

```

#### 4-La classe assignation

//La classe assignation

```

import java.util.ArrayList;
import java.util.Scanner;

public class Gestionnaire {

    /**
     * @param args
     */
    static ArrayList<Membre> membres = new ArrayList<>();
    static ArrayList<Tache> taches = new ArrayList<>();
    private static Scanner sc;

    // / les methodes de la classe membre
    // méthode de recherche d'un membre
    public static Membre recherche_membre(String nom) {
        Membre res = null;
        for (Membre m : membres) {
            if (m.getNom().equalsIgnoreCase(nom)) {

```

```

        res = m;
        break;
    }
}
return res;
}

// méthode de modification d'un membre
public static void editMembre(int index, String nom) {
    Membre m3 = recherche_membre(nom);
    if (m3 != null) {
        System.out.println("Ce nom existe déjà");
    } else {
        membres.set(index, m3);
    }
}

// méthode d'ajout d'un membre
public static void addMembre(Membre m) {
    Membre m2 = recherche_membre(m.getNom());
    if (m2 != null) {
        System.out.println("Ce nom existe déjà");
    } else {
        m.setId(membres.size()+1);
        membres.add(m);
    }
}

// méthode de suppression d'un membre
public static void delMembre(String nom) {
    Membre m3 = recherche_membre(nom);
    if (m3 != null) {
        membres.remove(m3);
    } else {
        System.out.println("Ce nom n'existe pas déjà");
    }
}

// Debut des Methodes de la classe tache

// Recherche d'une tache
public static Tache recherche_tache(String nom_tache) {
    Tache res = null;
    for (Tache t : taches) {
        if (t.getNomt().equalsIgnoreCase(nom_tache)) {
            res = t;
        }
    }
}

```

```

        break;
    }
}
return res;
}

// méthode de modification d'une tâche
public static void editTache(int index, String nomt) {
    Tache t = recherche_tache(nomt);
    if (t != null) {
        System.out.println("Cette tâche existe déjà");
    } else {
        taches.set(index, t);
    }
}

// méthode d'ajout d'une tâche
public void addTache(Tache t) {
    Tache t2 = recherche_tache(t.getNomt());
    if (t2 != null) {
        System.out.println("Cette tâche existe déjà");
    } else {
        taches.add(t);
    }
}

// méthode de suppression d'un membre
public void delTache(String nom) {
    Membre t3 = recherche_membre(nom);
    if (t3 != null) {
        taches.remove(t3);
    } else {
        System.out.println("Suppression de la tâche avec succès");
    }
}

public static void menu() {
    System.out.println("1 : Créer un membre");
    System.out.println("2 : Rechercher un membre");
    System.out.println("3 : Modifier un membre");
    System.out.println("4 : Supprimer un membre");
    System.out.println("5 : Créer une tâche");
    System.out.println("6 : Supprimer une tâche ");
    System.out.println("7 : Assigner une tâche à un membre");
    System.out.println("Q ou q : Quitter le programme");
}

```

```

}

// / fin des methodes de la classe tache

public static void main(String[] args) {
    // TODO Auto-generated method stub
    String nom = "";
    String choix = "";
    sc = new Scanner(System.in);
    do {
        menu();
        //Ajouter un membre
        System.out.print("Veuillez choisir une option: ");
        choix = sc.nextLine();
        if (choix.equals("1")) {
            System.out.print("Veuillez saisir le nom du
nouveau membre: ");
            nom = sc.nextLine();
            addMembre(new Membre(0, nom));
            // Rechercher et afficher un membre
        } else if (choix.equals("2")) {
            System.out.print("Veuillez saisir le nom du
membre à rechercher: ");
            nom = sc.nextLine();
            Membre m2 = recherche_membre(nom);
            if (m2 != null)
                System.out.println(m2.toString());
            else
                System.out.println("Ce nom n'existe pas!");

            // Modifier un membre
        } else if (choix.equals("3")) {
            System.out.print("Veuillez saisir le nom du membre à
modifier: ");
            nom = sc.nextLine();
            Membre m3 = recherche_membre(nom);
            if (m3 != null) {
                System.out.println(" veuillez entrer les
modifications");

                nom = sc.nextLine();
                System.out.println(" modifications
enregistrees");
            } else {
                System.out.println(" Ce nom n'existe pas");
            }
        }
    } while (sc.nextLine().equals("q"));
}

```



```

    }
    else if (choix.equals("4")) {
        System.out.print("Veuillez saisir le nom du
membre a supprimer: ");
        nom = sc.nextLine();
        delMembre(nom);
    }

    // Ajout tache
    else if (choix.equals("5")) {
        System.out.print("Veuillez saisir le nom de la
tache: ");
        nom = sc.nextLine();
    }
    else if (choix.equals("6")) {
        System.out.print("Veuillez choisir la tache a
Supprimer: ");
        nom = sc.nextLine();
    }
    else if (choix.equals("7")) {
        String nom2;
        String nom1;
        System.out.print("Veuillez saisir nom de la tache à
assigner à un membre : ");
        nom1 = sc.nextLine();
        System.out.print("Veuillez saisir du membre à
assigner la tache : ");
        nom2 = sc.nextLine();
        System.out.println("**** La tache:" +nom1);
        System.out.println("est assignee a:" +nom2);
        System.out.println("*****");
    }
    else{
        System.out.println("\n*** Mauvaise option ***\n");
    }
} while (!choix.toUpperCase().equals("Q")); // q == pour
quitter le programme
}
}

```

## Références

- [1] <http://selsek.free.fr/eloker/miage/cours%20M1/uml/UML-ClassesObjets.pdf>
- [2] <https://sites.google.com/site/developpementdelinformatique/uml/diagramme-de-sequence>
- [3] <http://uml.free.fr/cours/i-p19.html>
- [4] [http://www.fr.w3eacademy.com/eclipse/eclipse\\_overview.htm](http://www.fr.w3eacademy.com/eclipse/eclipse_overview.htm)