

Travail Pratique N°1

Génie Logiciel

Rapport du projet Modélisation et programmation orientée-objet avec Java

Auteur : BIAYA TSHIBANGU Emmanuel
Promotion 21

Introduction

La modélisation d'un problème est l'une de partie la plus importante dans la mise en place d'un système d'information que ce soit dans les entreprises ou les Universités. De ce fait, cette pratique reste une technique incontestable pour tout développeur qui souhaite mettre en place son application. Ainsi ce présent rapport vise à mettre en place une application de gestion des tâches qui permettra à un utilisateur d'effectuer automatiquement certaines tâches telle que :

- la modifier;
- la supprimer;
- La modification;
- La recherche.

Description et spécification du logiciel à développer

Nous allons réaliser un petit gestionnaire de tâches pour une équipe de travail. Ce dernier, fournira 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).

Une tâche est composée de ces informations suivantes :

- ID
- Nom
- Une description
- Status : nouveau, en-progrès, terminé.

Un membre est composé de ces informations suivantes :

- ID
- Nom.

Conception

- **Diagramme de cas d'utilisation**

Un diagramme de cas d'utilisation (use case) représente un ensemble de séquences d'actions réalisées par le système et produisant un résultat observable intéressant pour un acteur particulier. Un cas d'utilisation modélise un service rendu par le système. Il exprime les interactions acteurs/système et apporte une valeur ajoutée « notable » à l'acteur concerné.

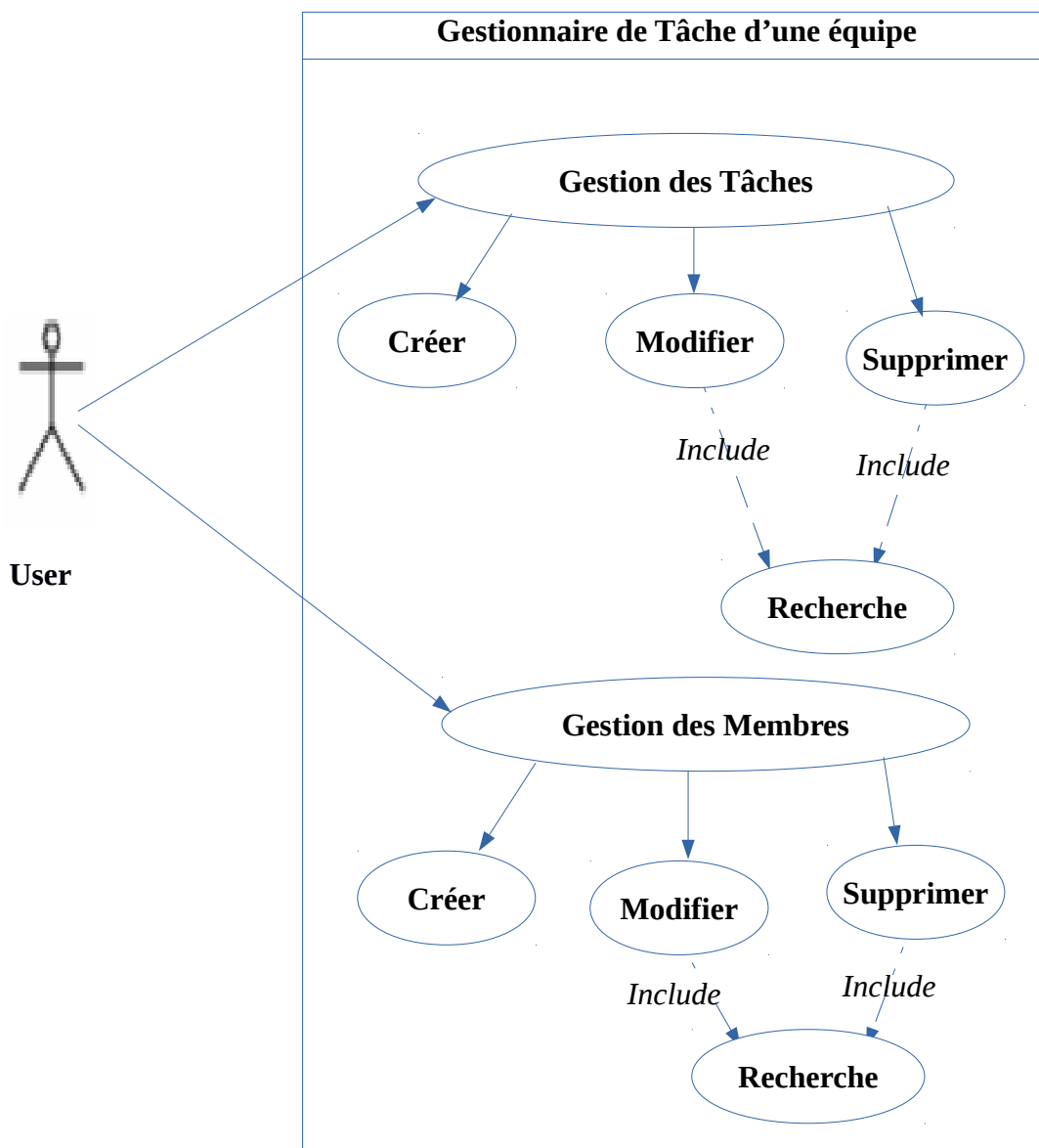


Schéma 1: Diagramme de cas d'utilisation

Le schéma ci-haute demontre comment les opérations peuvent s'effectuer dans le système par un utilisateur. Ici l'acteur principale est le gestionnaire qui a la possibilité de créer un membre ou plusieurs, affecter une tâche à ce membre, ensuite faire la recherche de ce membre pour modifier son comportement soit caremment pour le supprimer et supprimer les tâches qui lui ont été affecté. Nous signalons que l'opération de la suppression et de la modification sont inclus dans la recherche et fonctionne de façon liée.

● Diagramme de Séquence

Le diagramme de séquence montre des interactions entre objets selon un point de vue temporel. Ce type de diagramme sert à modéliser les aspects dynamiques des systèmes temps réels et des scénarios complexes mettant en oeuvre peu d'objets.

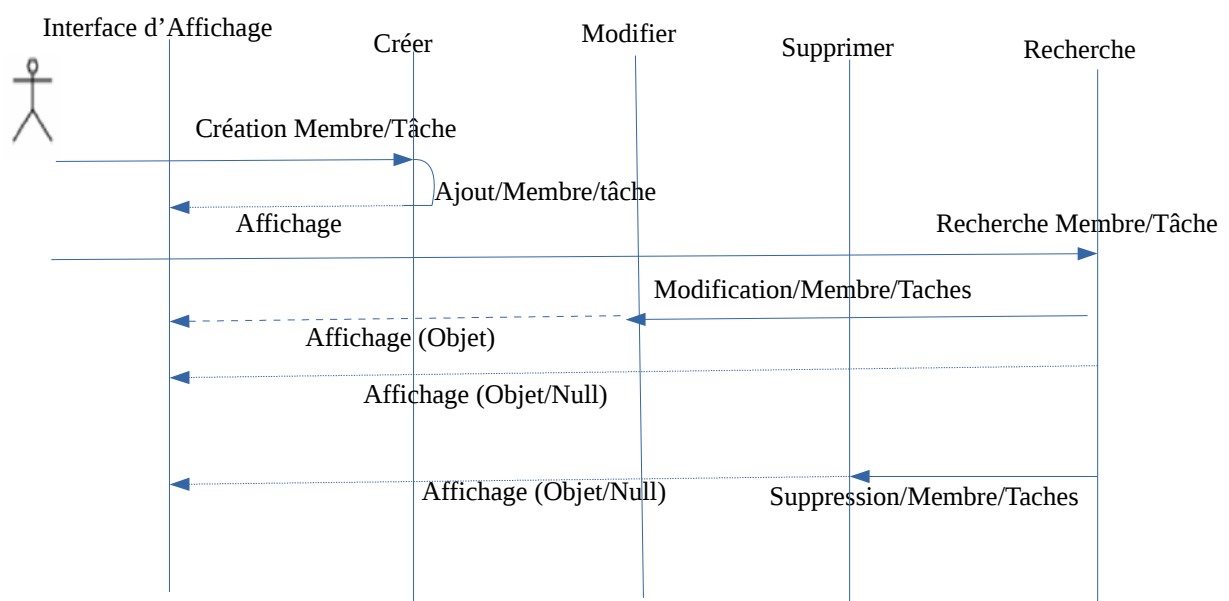


Figure 1: Diagramme de séquence

la figure ci-haute montre comment les opérations s'effectue au sein de notre système:

1. Lorsque nous crayons un membre, il y a un feedback d'affichage d'objet crée et dans la même opération nous pouvons aussi faire un ajout d'un membre;
2. En faisant la recherche de l'objet créer nous pouvons modifier l'objet ou la supprimer.

Diagramme de classe

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet. Alors que le diagramme de cas d'utilisation montre un système du point de vue des acteurs, le diagramme de classes en montre la structure interne. Il contient principalement des classes. Une classe contient des attributs et des opérations. Le diagramme de classes n'indique pas comment utiliser les opérations : c'est une description purement statique d'un système.

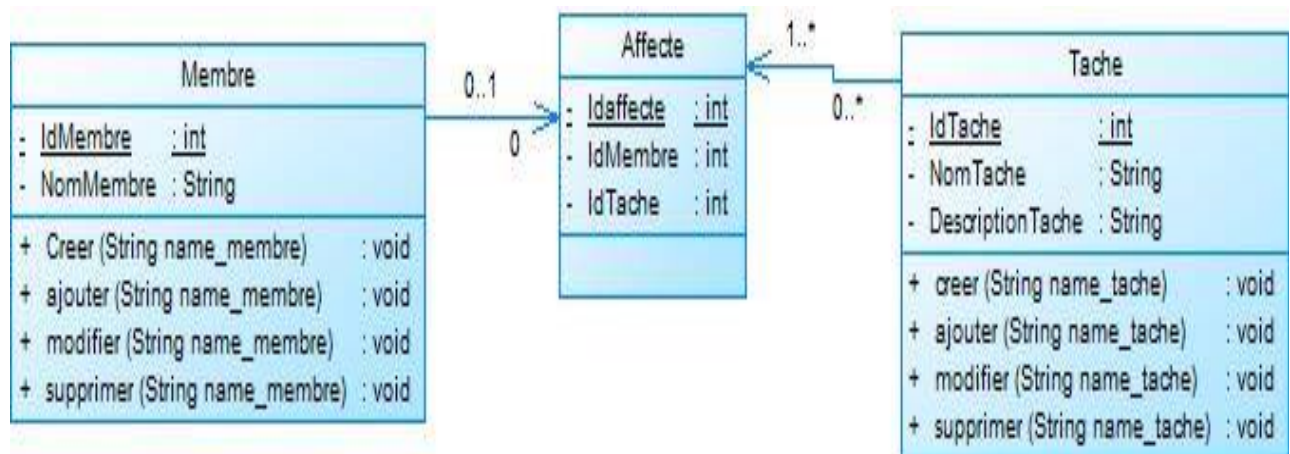


Figure 2 : Diagramme de classe

Implémentation

L'implémentation est la partie de la réalisation d'un projet qui consiste au choix de la technologie à utiliser pour pouvoir coder le programme. Pour notre travail notre choix se porte sur le langage de programmation Java.

La compilation de notre programme se fait de la manière suivante:

1. Nous avons un Menu principal qui nous permet de faire le choix de différentes fonctionnalités

```

Console
PrincipaleMain [Java Application] /usr/lib/jvm/java-8-oracle/bin/java
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 : Affecter une tâche à un membre
Q ou q : Quitter le programme
Veuillez choisir une option :
  
```

2. Lorsque nous choisissons l'option 1 le programme nous demande de créer un membre, nous saisissons ainsi le membre

```

Console
PrincipaleMain [Java Application] /usr/lib/jvm/java-8-oracle/bin/java
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 : Affecter une tâche à un membre
Q ou q : Quitter le programme
Veuillez choisir une option : 1
Veuillez saisir le nom du nouveau membre : emmanuel
  
```

3. Le programme nous crée un membre “emmanuel”

```
Console
PrincipaleMain [Java Application] /usr/lib/jvm/java-8-oracle/bin/ja
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 : Affecter une tâche à un membre
Q ou q : Quitter le programme
Veuillez choisir une option : 1
Veuillez saisir le nom du nouveau membre : emmanuel
Creation du nouveau Membre:emmanuel
```

4. En saisissant l’option 2 de notre Menu, le programme nous demande de faire la recherche de l’objet créer puis nous saisissons le nom “emmanuel” nous constatons que le programme nous retourne le nom “emmanuel” créer

```
Console
PrincipaleMain [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Au
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 : Affecter une tâche à un membre
Q ou q : Quitter le programme
Veuillez choisir une option : 2
Veuillez saisir le nom du membre à rechercher : emmanuel
emmanuel
```

5. Nous allons supprimer l’objet créer “emmanuel”

```
Console
PrincipaleMain [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (
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 : Affecter une tâche à un membre
Q ou q : Quitter le programme
Veuillez choisir une option : 4
Veuillez saisir le nom du membre à supprimer : emmanuel
Suppression reussie
```

6. Création d'une tâche "manger" puis suppression de la tâche "manger"

```
Console
PrincipaleMain [Java Application] /usr/lib/jvm/java-8-ora
Veuillez choisir une option : 5
Veuillez saisir le nom de la tache : manger
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 : Affecter une tâche à un membre
Q ou q : Quitter le programme
Veuillez choisir une option : 6
Veuillez choisir la tache a Supprimer : manger
Suppression de la tâche manger
```

7. Nous allons affecter une tâche "balayer chaque nuit" à un membre "emmanuel"

```
Console
PrincipaleMain [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Aug 19, 2017 11:28:36 PM)
4 : Supprimer un membre
5 : Créer une tâche
6 : Supprimer une tâche
7 : Affecter une tâche à un membre
Q ou q : Quitter le programme
Veuillez choisir une option : 7
Veuillez saisir nom de la tache à assigner à un membre : balayer chaque nuit
Veuillez saisir le membre: emmanuel
**** La tâche : balayer chaque nuit
est assignée à :emmanuel
```

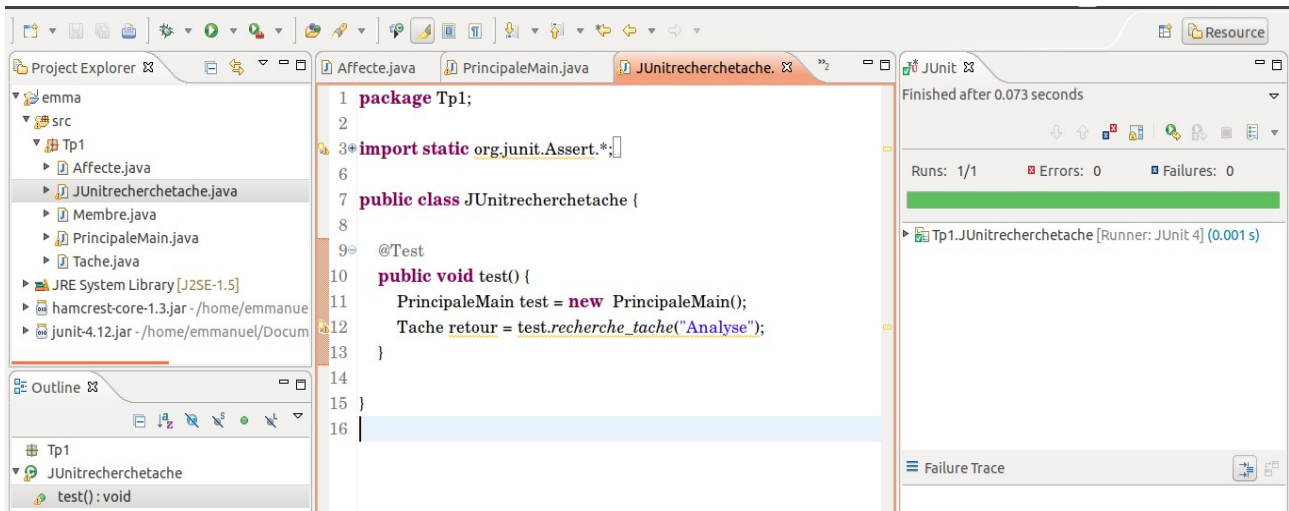
Test JUnit

JUnit est un framework open source pour réaliser des tests unitaires sur du code Java. Le principal intérêt est de s'assurer que le code répond toujours au besoin même après d'éventuelles modifications. Le but est d'automatiser les tests. Ceux ci sont exprimés dans des classes sous la forme de cas de tests avec leurs résultats attendus. JUnit exécute ces tests et les comparent avec ces résultats. Avec Junit, l'unité de tests est une classe dédiée qui regroupe des cas de tests. Ces cas de tests exécutent les tâches suivantes :

- Création d'une instance de la classe et de tout autre objet nécessaire aux tests;
- Appel de la méthode à tester avec les paramètres du cas de test;

➤ Comparaison du résultat obtenu avec le résultat attendu : en cas d'échec, une exception est levée.

Ainsi, pour utiliser JUnit, il faut créer une classe qui va contenir les cas de test. Il faut créer une nouvelle entité de type " Java / JUnit / Scénario de test ". JUnit utilise l'introspection pour exécuter les méthodes commençant par test.



Resultat test Junit

Nous remarquons que notre test a réussi avec succès car en observant l'onglet JUnit à droite de notre capture nous remarquons une ligne verte affichée avec les inscriptions suivante:

- «Failures»: contient la liste des cas de tests qui ont échoués;
- «Error»: contient une arborescence des cas de tests.

Avec les resultats 0(zero) pour failure et 0(zero) pour Error d'où nos classes utilisent des bonnes méthodes.

Test d'acceptation

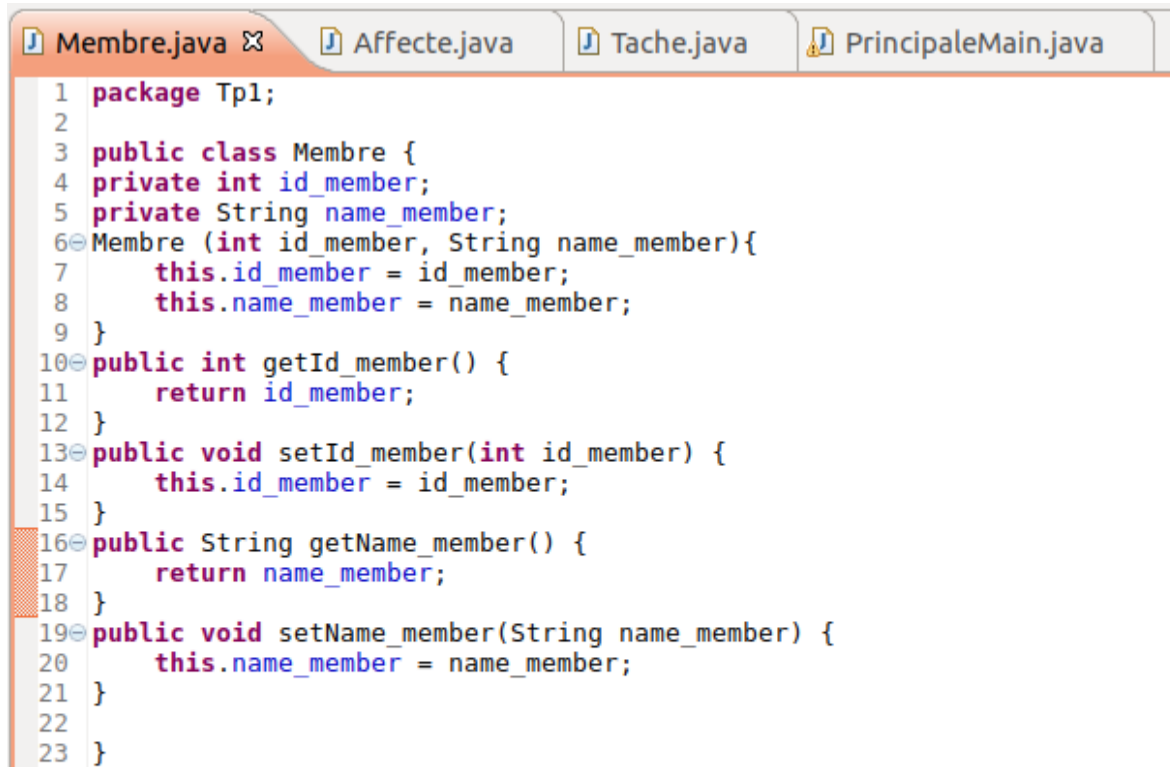
Nous avons fait les tests d'essai de l'ensemble de nos fonctionnalités dans la phase implémentation et nous pouvons tirer la conclusion que ces tests à réussi avec succès car toutes les fonctionnalités de création, modification, suppression et recherche sont belle et bien en fonction dans notre programme.

Conclusion

Au terme de notre travail qui consistait à la réalisation d'une application de gestion de tâches, nous pouvons dire être satisfait des resultats car les objectif établi dès le départ ont trouvé leurs réponses. Aussi par ce travail nous avons appris des nouveaux concepts de la programmation et des tests ainsi que la modélisation en appliquant la méthode UML pour la conception de classes. Nous nous disons être capable d'aborder d'autres sujets de développement avec toute quiétude.

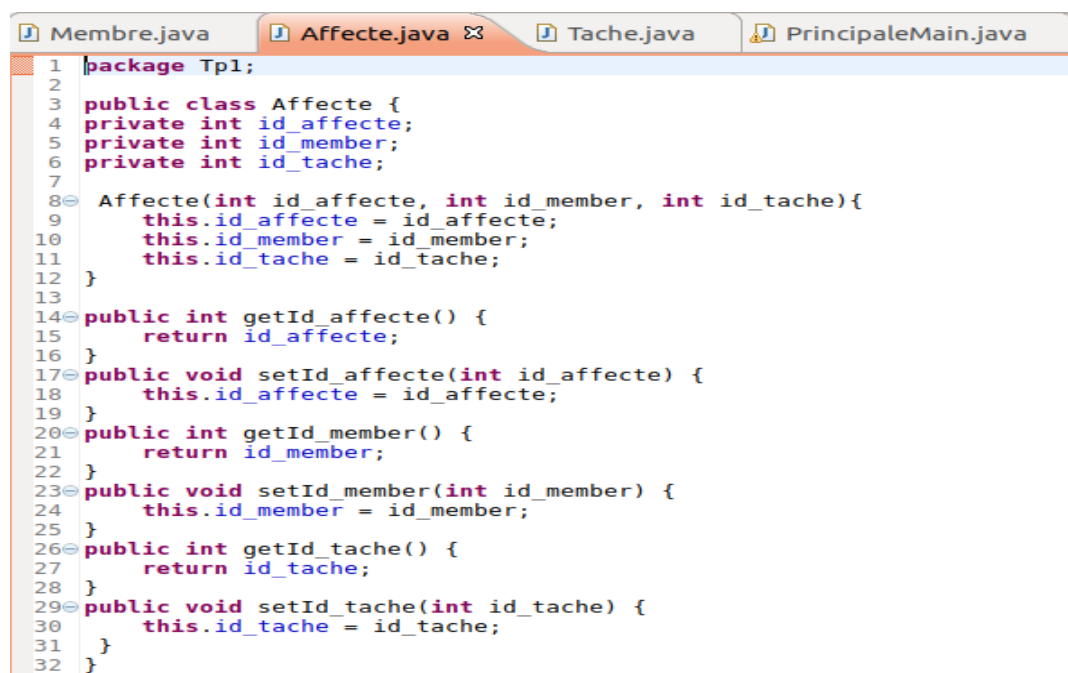
Annexe

Code sources pour chaque Classes



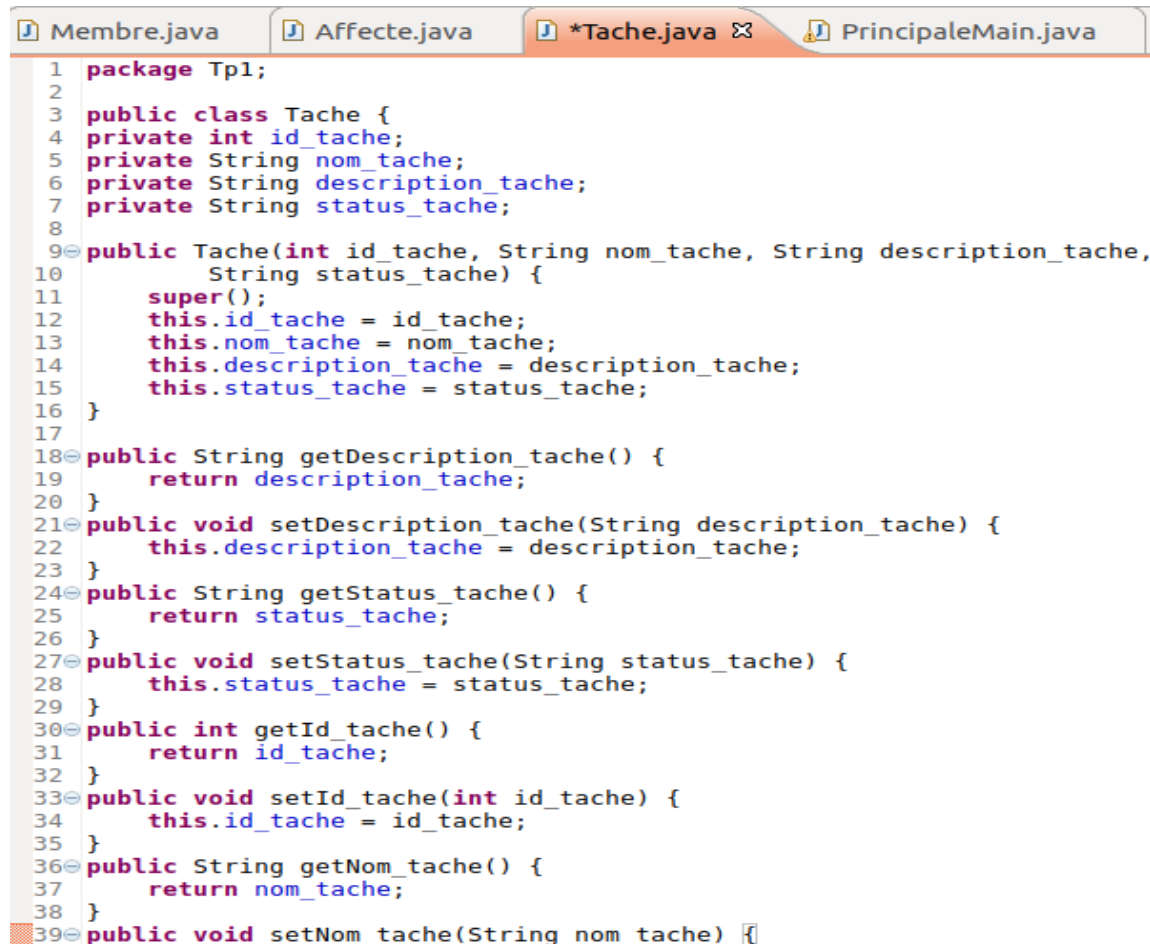
```
1 package Tp1;
2
3 public class Membre {
4     private int id_member;
5     private String name_member;
6     Membre (int id_member, String name_member){
7         this.id_member = id_member;
8         this.name_member = name_member;
9     }
10    public int getId_member() {
11        return id_member;
12    }
13    public void setId_member(int id_member) {
14        this.id_member = id_member;
15    }
16    public String getName_member() {
17        return name_member;
18    }
19    public void setName_member(String name_member) {
20        this.name_member = name_member;
21    }
22 }
23 }
```

Figure 3 : Conception de la classe membre



```
1 package Tp1;
2
3 public class Affecte {
4     private int id_affecte;
5     private int id_member;
6     private int id_tache;
7
8     Affecte(int id_affecte, int id_member, int id_tache){
9         this.id_affecte = id_affecte;
10        this.id_member = id_member;
11        this.id_tache = id_tache;
12    }
13
14    public int getId_affecte() {
15        return id_affecte;
16    }
17    public void setId_affecte(int id_affecte) {
18        this.id_affecte = id_affecte;
19    }
20    public int getId_member() {
21        return id_member;
22    }
23    public void setId_member(int id_member) {
24        this.id_member = id_member;
25    }
26    public int getId_tache() {
27        return id_tache;
28    }
29    public void setId_tache(int id_tache) {
30        this.id_tache = id_tache;
31    }
32 }
```

Figure 4: Conception de la classe Affecte

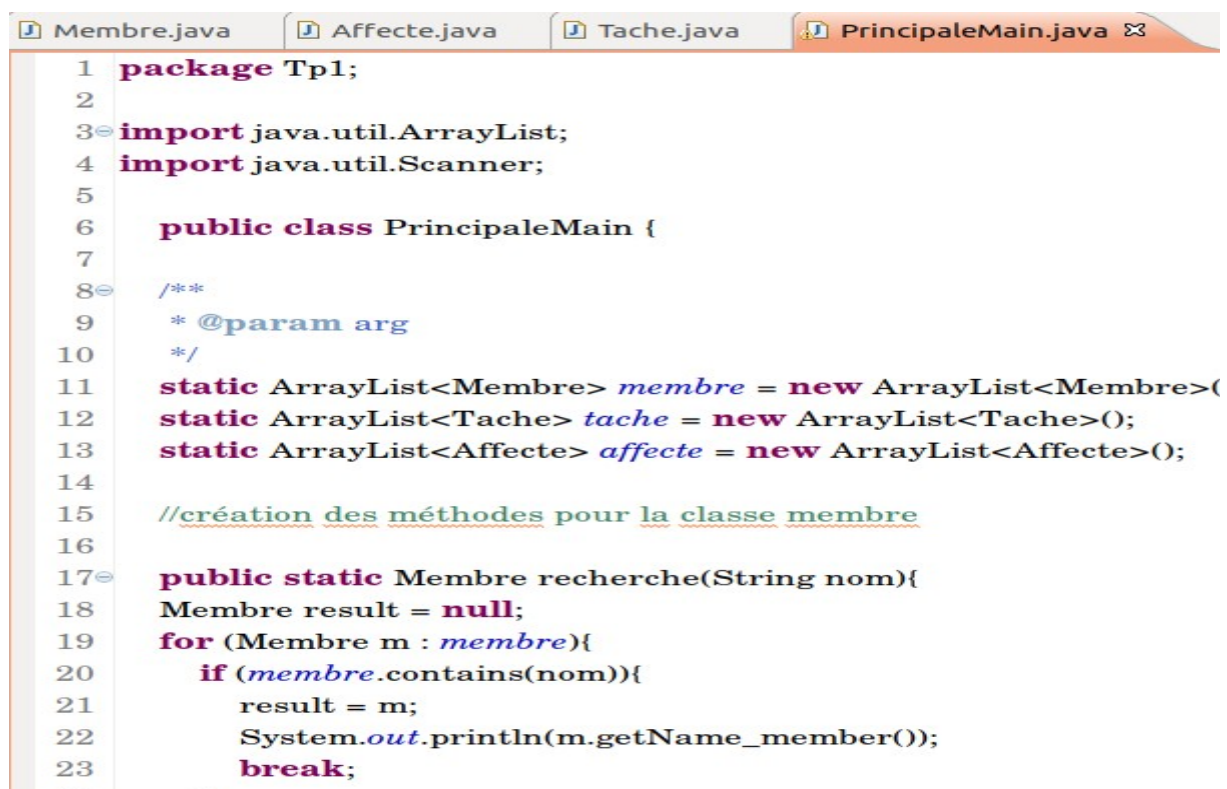


```

1 package Tp1;
2
3 public class Tache {
4     private int id_tache;
5     private String nom_tache;
6     private String description_tache;
7     private String status_tache;
8
9     public Tache(int id_tache, String nom_tache, String description_tache,
10         String status_tache) {
11         super();
12         this.id_tache = id_tache;
13         this.nom_tache = nom_tache;
14         this.description_tache = description_tache;
15         this.status_tache = status_tache;
16     }
17
18     public String getDescription_tache() {
19         return description_tache;
20     }
21     public void setDescription_tache(String description_tache) {
22         this.description_tache = description_tache;
23     }
24     public String getStatus_tache() {
25         return status_tache;
26     }
27     public void setStatus_tache(String status_tache) {
28         this.status_tache = status_tache;
29     }
30     public int getId_tache() {
31         return id_tache;
32     }
33     public void setId_tache(int id_tache) {
34         this.id_tache = id_tache;
35     }
36     public String getNom_tache() {
37         return nom_tache;
38     }
39     public void setNom_tache(String nom_tache) {

```

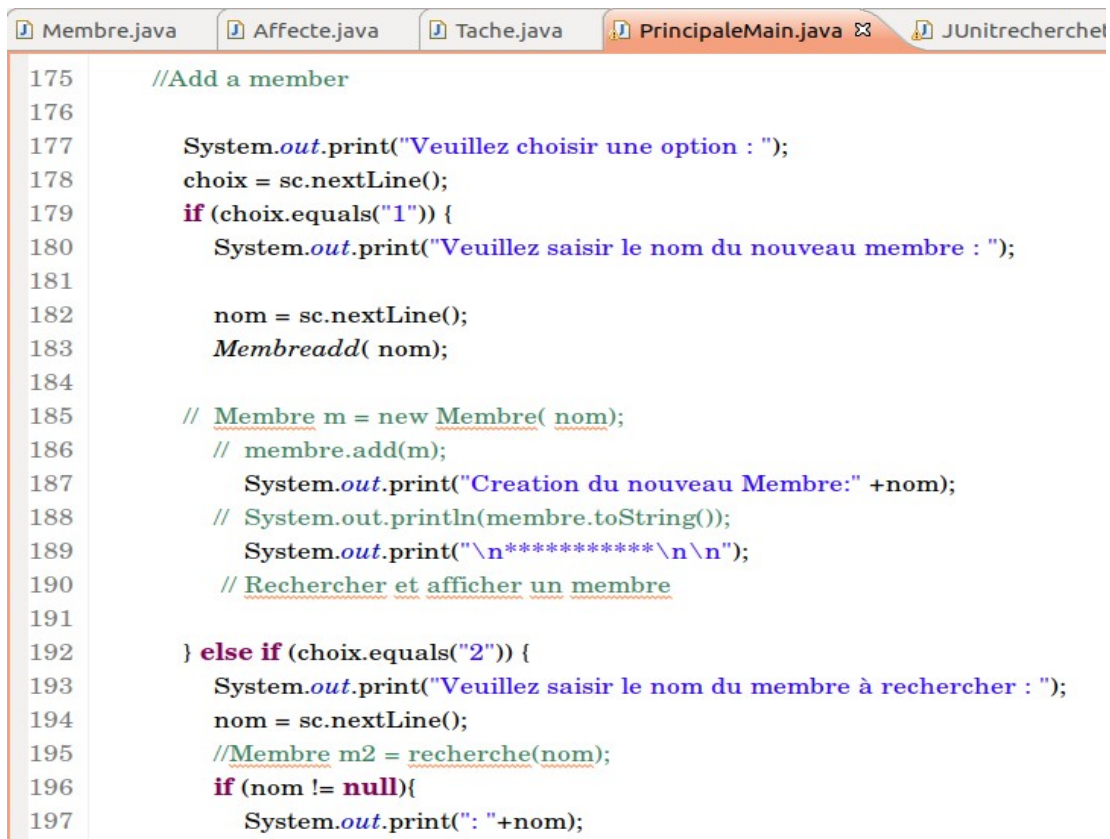
Figure 5: Conception de la classe Tâche



```

1 package Tp1;
2
3 import java.util.ArrayList;
4 import java.util.Scanner;
5
6 public class PrincipaleMain {
7
8     /**
9      * @param arg
10     */
11     static ArrayList<Membre> membre = new ArrayList<Membre>();
12     static ArrayList<Tache> tache = new ArrayList<Tache>();
13     static ArrayList<Affecte> affecte = new ArrayList<Affecte>();
14
15     //création des méthodes pour la classe membre
16
17     public static Membre recherche(String nom){
18         Membre result = null;
19         for (Membre m : membre){
20             if (membre.contains(nom)){
21                 result = m;
22                 System.out.println(m.getName_member());
23                 break;

```



```
175 //Add a member
176
177 System.out.print("Veuillez choisir une option : ");
178 choix = sc.nextLine();
179 if (choix.equals("1")) {
180     System.out.print("Veuillez saisir le nom du nouveau membre : ");
181
182     nom = sc.nextLine();
183     Membreadd( nom);
184
185     // Membre m = new Membre( nom);
186     // membre.add(m);
187     System.out.print("Creation du nouveau Membre:" +nom);
188     // System.out.println(membre.toString());
189     System.out.print("\n*****\n\n");
190     // Rechercher et afficher un membre
191
192 } else if (choix.equals("2")) {
193     System.out.print("Veuillez saisir le nom du membre à rechercher : ");
194     nom = sc.nextLine();
195     //Membre m2 = recherche(nom);
196     if (nom != null){
197         System.out.print(": " +nom);
```

Figure 6: Conception de la classe PrincipaleMain