

---

## **SUD311 : Design pattern et architectures logicielles**

### **Étude de cas 2 : le design pattern Composite**

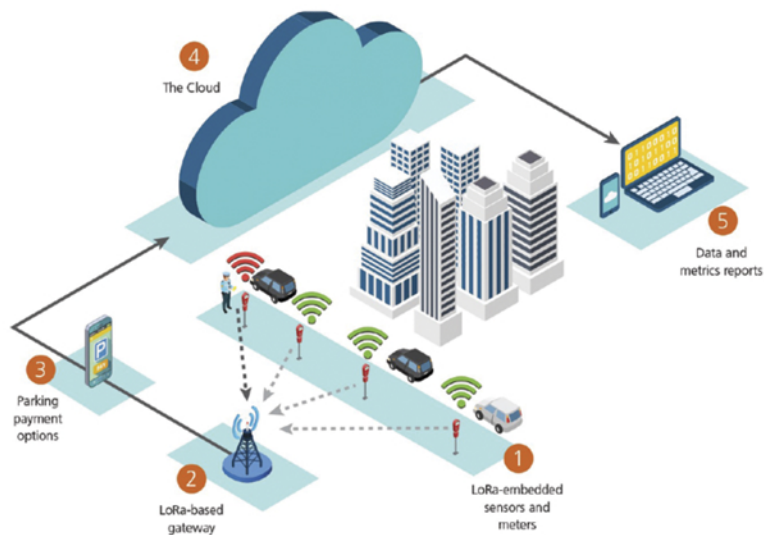
---

*Réalisé par :*

**HAJJAJI Ayyoub**  
**EL MEKKAOUI Fayssal**

*Encadré par :*

**EN-NOUAARY Abdeslam**



## Introduction

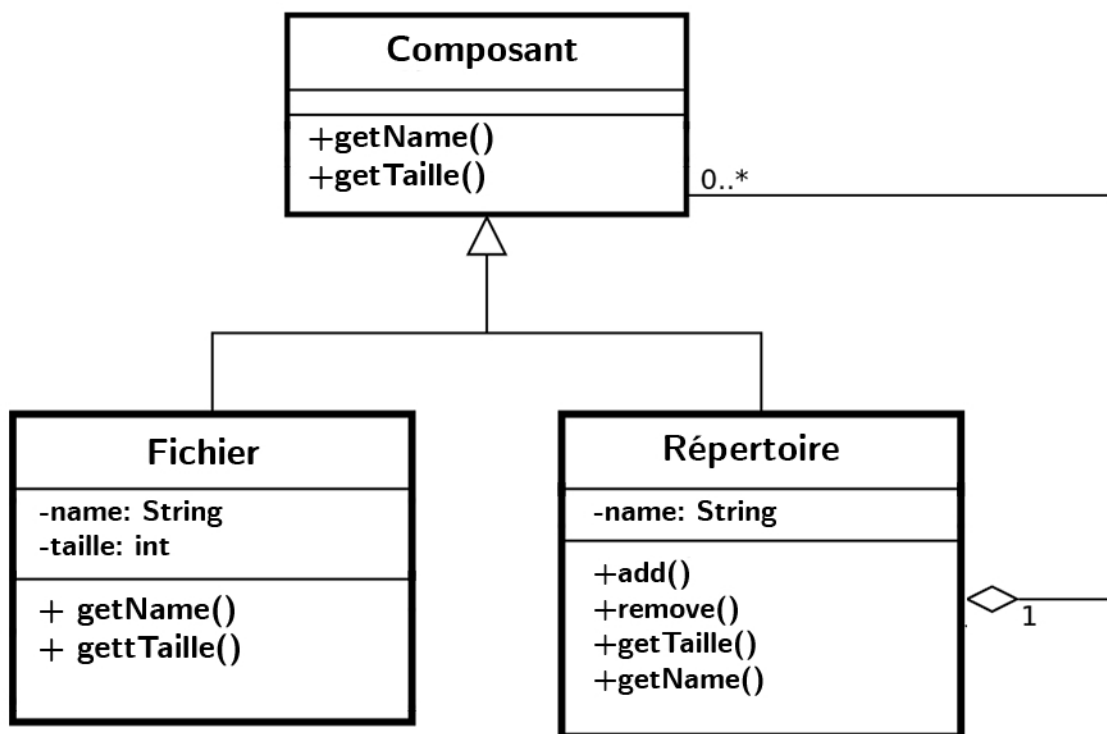
En programmation orientée objet, le patron **composite** propose une structure **récursive** permettant d'implémenter avec la même interface logicielle sur les feuilles et les composites afin qu'ils soient manipulés de la même manière.

On utilise principalement le modèle Composite lorsque l'on souhaite que le client n'ait pas à se préoccuper de la différence entre combinaisons d'objets et objets individuels.

**Par exemple** la gestion des fichiers et les répertoires, l'implémentations des formes géométriques, gestion d'emploi (Directeur-Employé)...

Dans notre cas nous allons essayer de trouver une solution orientée objet afin de gérer l'espace dans un disque composé de répertoires et des fichiers.

**Diagramme des classes UML :**



## Partie code (Java):

Les classes utilisées:

```
public class Fichier implements Composant {
    private int taille;
    private String name;
}
```

```
public interface Composant {
    public int getTaille();
    public String getName();
}
```

```
public class Repertoire implements Composant {
    private Collection<Composant> fichiers;
    private String name;
}
```

Pour la classe Repertoire nous allons créer une methode **getTaille()** qui va permettre au client de calculer des tailles des fichiers et les répertoires sans les stocker..

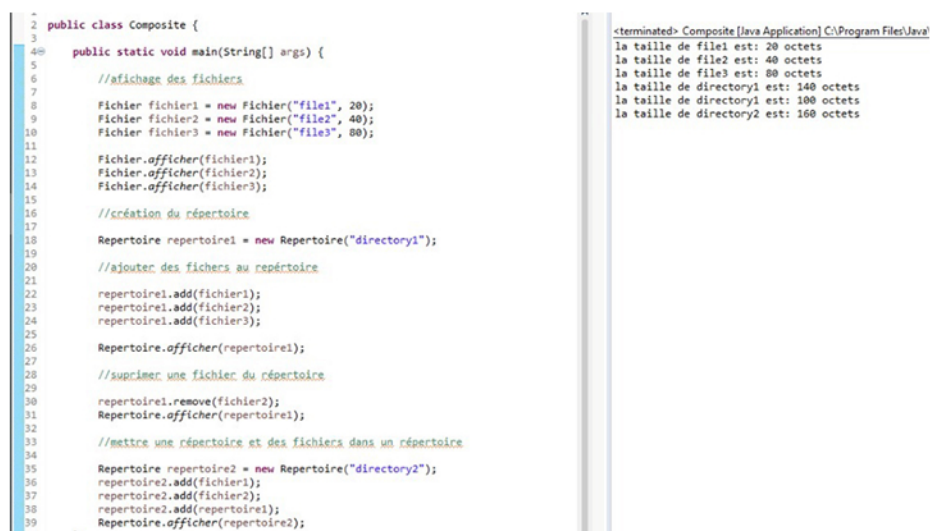
```
public int getTaille() {
    int result = 0;
    for (Iterator<Composant> i = fichiers.iterator(); i.hasNext(); ) {
        Object objet = i.next();

        Composant composant = (Composant)objet;

        result += composant.getTaille();
    }
    return result;
}
```

Le client peut aussi ajouter ou bien supprimer les fichiers et les répertoires grâce aux méthodes **add()** et **remove()**.

Voici un exemple d'exécution de notre code, nous avons crée trois fichiers ( file1, file2, file3), de tailles (20 ,40 ,80), ensuite nous avons ajouté trois répertoires et nous avons essayé par la suite les différents opérations.



```
2 public class Composite {
3
4     public static void main(String[] args) {
5         //affichage des fichiers
6
7         Fichier fichier1 = new Fichier("file1", 20);
8         Fichier fichier2 = new Fichier("file2", 40);
9         Fichier fichier3 = new Fichier("file3", 80);
10
11         Fichier.afficher(fichier1);
12         Fichier.afficher(fichier2);
13         Fichier.afficher(fichier3);
14
15         //création du répertoire
16
17         Repertoire repertoire1 = new Repertoire("directory1");
18
19         //ajouter des fichiers au répertoire
20
21         repertoire1.add(fichier1);
22         repertoire1.add(fichier2);
23         repertoire1.add(fichier3);
24
25         Repertoire.afficher(repertoire1);
26
27         //supprimer un fichier du répertoire
28
29         repertoire1.remove(fichier2);
30         Repertoire.afficher(repertoire1);
31
32         //mettre un répertoire et des fichiers dans un répertoire
33
34         Repertoire repertoire2 = new Repertoire("directory2");
35         repertoire2.add(repertoire1);
36         repertoire2.add(fichier2);
37         repertoire2.add(repertoire1);
38         Repertoire.afficher(repertoire2);
39
40     }
41 }
```

```
<terminated> Composite [Java Application] C:\Program Files\Java\
la taille de file1 est: 20 octets
la taille de file2 est: 40 octets
la taille de file3 est: 80 octets
la taille de directory1 est: 140 octets
la taille de directory1 est: 100 octets
la taille de directory2 est: 160 octets
```

Vous pouvez visiter le lien suivant pour voir le code complet:

<https://github.com/ayy-oub/design-pattern-Composite>