**Sujet d'exercice**

Vous travaillez pour une société qui développe un système de gestion de fichiers et dossiers dans le cloud. Les composants principaux sont les Dossiers et les Fichiers. Un Dossier peut contenir plusieurs Dossiers ou Fichiers.

L'objectif est de fournir une fonctionnalité pour appliquer différentes opérations sur ces objets, telles que la Compression et le Chiffrement.

Les classes de base que vous devez développer sont :

* IDossier
* IFichier
* Dossier
* Fichier

Les classes de design pattern que vous devez développer sont :

* ProxyDossier
* ProxyFichier
* VisiteurCompression
* VisiteurChiffrement
* CompositeDossier

**Conception UML**

UML est une représentation visuelle qui aide à comprendre comment les différentes classes interagissent entre elles. Voici une représentation UML pour cet exercice :



**Solution avec code C#**

**1. Interfaces**

Définir les interfaces IDossier et IFichier.

csharp

public interface IDossier

{

void Accept(IVisiteur visiteur);

void Add(IFichier fichier);

void Remove(IFichier fichier);

}

public interface IFichier

{

void Accept(IVisiteur visiteur);

}

**2. Classes de base**

csharp

public class Dossier : IDossier

{

private List<IFichier> \_fichiers = new List<IFichier>();

public void Accept(IVisiteur visiteur)

{

visiteur.Visit(this);

foreach (var fichier in \_fichiers)

{

fichier.Accept(visiteur);

}

}

public void Add(IFichier fichier)

{

\_fichiers.Add(fichier);

}

public void Remove(IFichier fichier)

{

\_fichiers.Remove(fichier);

}

}

public class Fichier : IFichier

{

public void Accept(IVisiteur visiteur)

{

visiteur.Visit(this);

}

}

**3. Classes de design pattern**

csharp

public class ProxyDossier : IDossier

{

private Dossier \_dossier = new Dossier();

public void Accept(IVisiteur visiteur)

{

\_dossier.Accept(visiteur);

}

public void Add(IFichier fichier)

{

\_dossier.Add(fichier);

}

public void Remove(IFichier fichier)

{

\_dossier.Remove(fichier);

}

}

public class ProxyFichier : IFichier

{

private Fichier \_fichier = new Fichier();

public void Accept(IVisiteur visiteur)

{

\_fichier.Accept(visiteur);

}

}

public class VisiteurCompression : IVisiteur

{

public void Visit(Dossier dossier)

{

Console.WriteLine("Compresser le dossier.");

}

public void Visit(Fichier fichier)

{

Console.WriteLine("Compresser le fichier.");

}

}

public class VisiteurChiffrement : IVisiteur

{

public void Visit(Dossier dossier)

{

Console.WriteLine("Chiffrer le dossier.");

}

public void Visit(Fichier fichier)

{

Console.WriteLine("Chiffrer le fichier.");

}

}

public class CompositeDossier : IDossier

{

private List<IDossier> \_dossiers = new List<IDossier>();

public void Accept(IVisiteur visiteur)

{

foreach (var dossier in \_dossiers)

{

dossier.Accept(visiteur);

}

}

public void Add(IFichier fichier)

{

foreach (var dossier in \_dossiers)

{

dossier.Add(fichier);

}

}

public void Remove(IFichier fichier)

{

foreach (var dossier in \_dossiers)

{

dossier.Remove(fichier);

}

}

public void AddDossier(IDossier dossier)

{

\_dossiers.Add(dossier);

}

public void RemoveDossier(IDossier dossier)

{

\_dossiers.Remove(dossier);

}

}

Remarque : Ce code est une illustration simpliste de l'utilisation des motifs de conception et ne représente pas une application réelle. Vous devrez l'adapter pour répondre aux besoins spécifiques de votre projet.