Rapport mini projet d'Architecture Logiciel

Rédigé par :

Fotso Tagne Steve didier
Kroun Arezki

Master 1 GIL 2021/2022

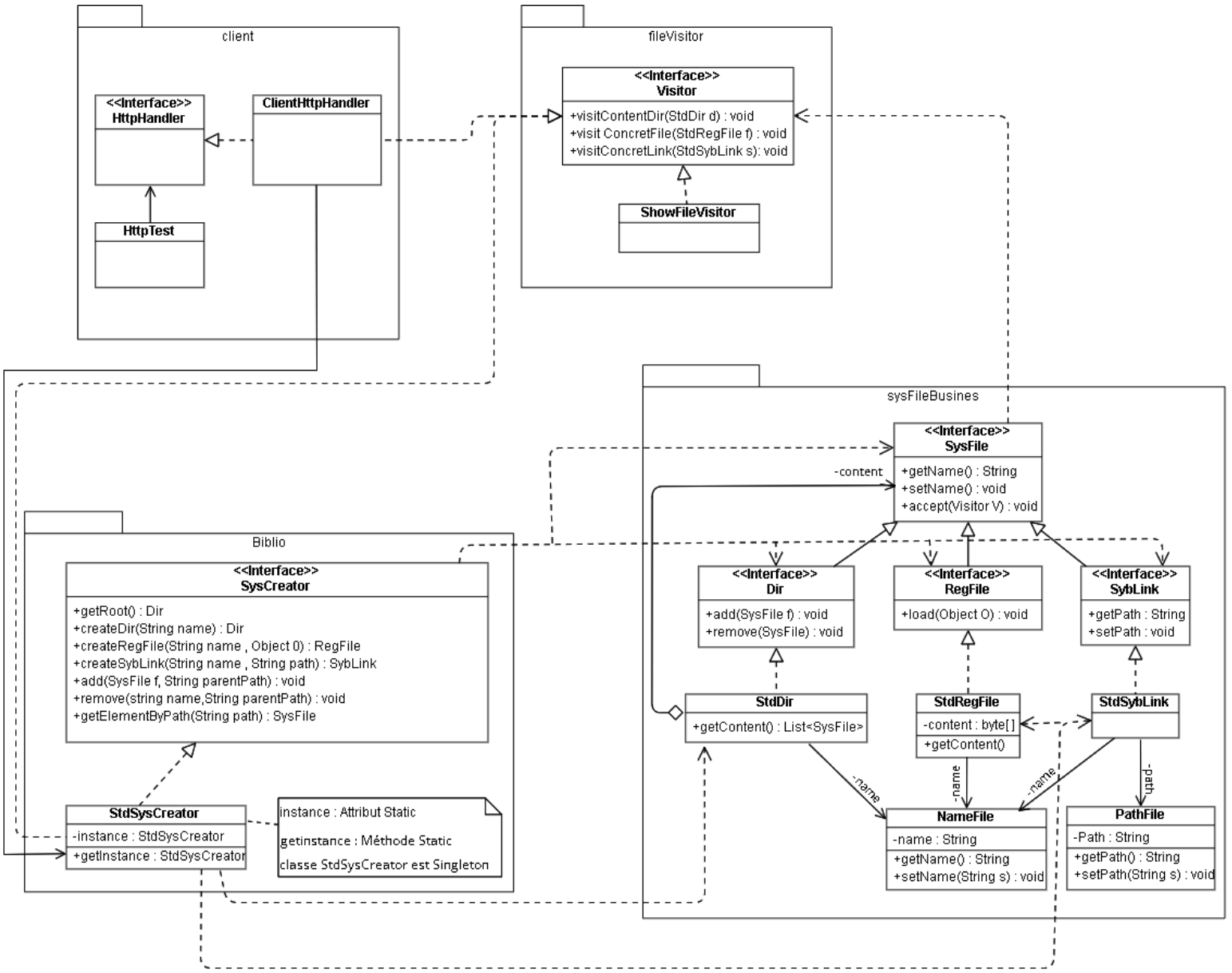
Sommaire :

I. Présentation du projet

Le projet consiste à réaliser un système de fichiers dont le contenu est stocké uniquement dans la mémoire vive d'un ordinateur. Son contenu est perdu, s'il n'est pas explicitement sauvegardé, lors du redémarrage ou de l'arrêt de la machine.
Le système de fichiers sera constitué des trois concepts suivants :
— Les fichiers réguliers qui sont caractérisés par : leur nom, leur taille et leur contenu.
— Les dossiers, caractérisés par leur nom et leur contenu.
— Des liens symboliques, caractérisés par leur nom et une chaîne de caractères représentant le chemin de la cible du lien.

Dans les chapitres suivants on va détailler et expliquer les choix de patrons de conception utilisés pour réaliser ce projet.

II. Architecture globale



III. Patrons mis en oeuvres

1. Patron composite

motivation :

-on voudrait manipuler les objets du système de fichier en mémoire sous forme d'une structure arborescente ou d'arbre.
-cacher au client la différence entre les noeuds de l'arborescence et les feuilles
-uniformiser les traitements entre les objets simples(fichier régulier, lien symbolique) et les objets composés(dossier).

2. Patron Visiteur

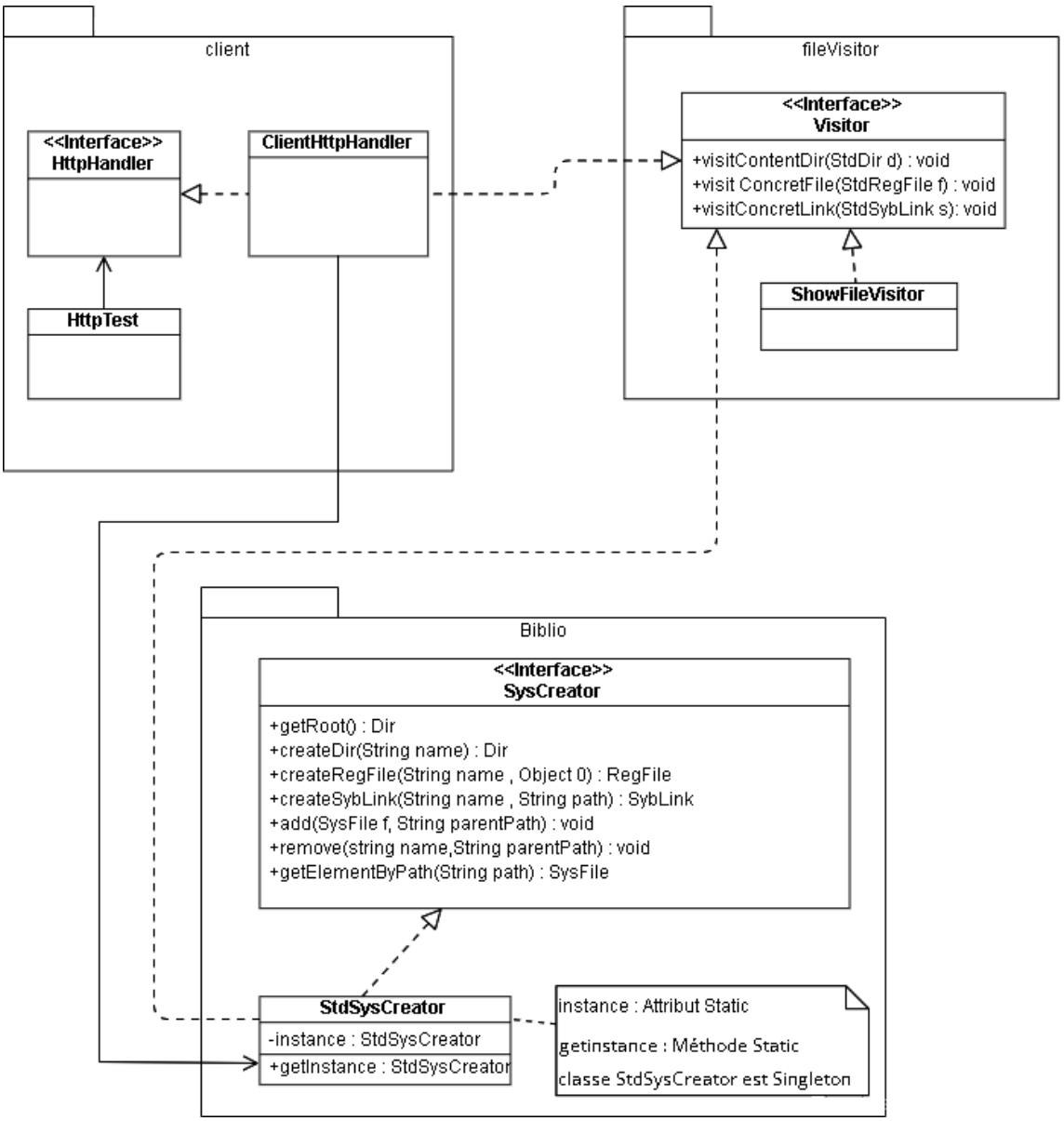
motivation:

-Il vient en aide au patron composite, par l'ajout d'opérations (via la méthode **accept**) qui permettent d'uniformiser l'accès au contenu d'un élément concret du système de fichier lors de l'exécution.

-la classe visiteur concret **ShowFileVisitor** prendra en charge la forme des éléments à l'affichage, par exemple gérer les icônes des éléments du système de fichier à l'affichage.

-la classe **StdSysCreator** implémente également l'interface **Visitor** ce qui lui permet de faire un parcours en profondeur pour rechercher un nœud dans la structure d'arbre d'objets composites.

-la classe **MyHandler** implémente également l'interface **Visitor** ce qui la permet d'appliquer un traitement spécifique concernant le rendu sur chaque élément de la structure composite (téléchargement pour un fichier régulier et affichage du contenu pour un dossier).



3. Patron Factory Method

motivation:

-on veut pouvoir garantir la cohérence entre les types effectifs des implémentations de notre système de fichier et il faut donc définir une interface pour créer un objet et laisser les sous-types décider de la classe à instancier.

l'interface **SysCreator** contient des méthodes de fabriques relative à différentes variantes abstraites d'un fichier (dossier, fichier régulier et lien symbolique) :

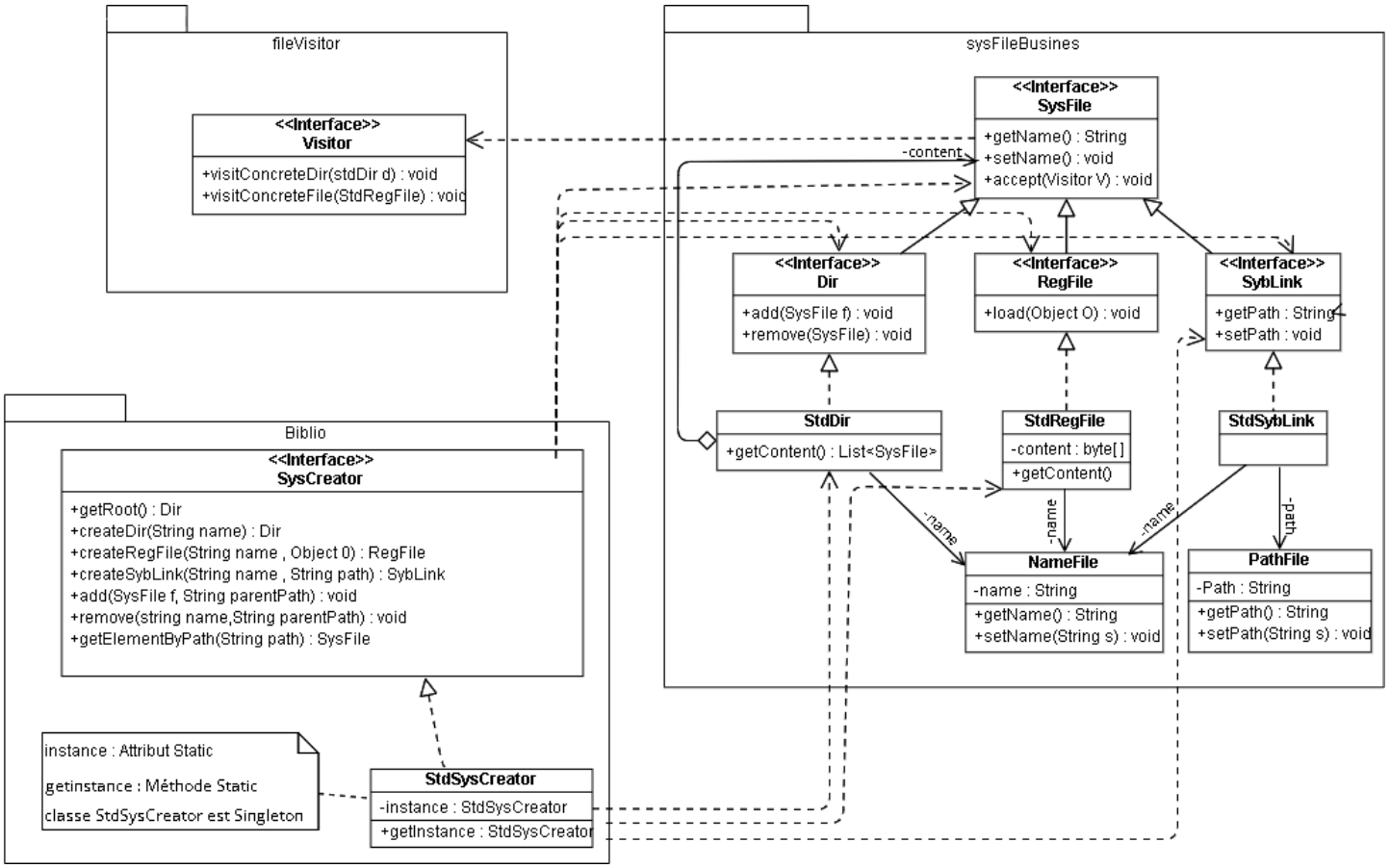
- **createDir**(String name): **Dir** →crée un dossier
- **createRegFile**(String name, Object o): **RegFile** →crée un fichier
- **createSybLink**(String name, String path): **SybLink** →crée un lien à la racine
- **createDir**(String name, String parentPath): **Dir** →crée un au chemin indiqué

trois opérations métiers supplémentaires:

- **getRoot()** qui retourne respectivement la **racine** du système de fichier.
- **getElementByPath(String path)** un élément du système de fichier connaissant son chemin.
- **add**(SysFile f, String parentPath); ajoute un fichier au chemin indiqué.

StdSysCreator est un **Singleton** en raison des contraintes d'unicité du **répertoire racine(root)**,

En gros le dossier racine sera instancié au tout premier appel de méthode **getInstance** de la classe **StdSysCreator,** ce qui nous garantira l'unicité du répertoire root et aussi la cohérence de notre système de fichier en mémoire.



IV. Liste des patrons écartés

Éventuellement, une liste des patrons que nous avons hésité à utiliser et les raisons pour lesquelles nous les avions écartés.

1. Patron Builder

La logique de construction d'un objet du système de fichier n'est à priori pas complexe et n'inclut pas de couplage avec une bibliothèque extérieure.

2. Patron Abstract Factory

Abstract Factory n'a pas de rôle métier. Elle ne sert qu'à produire des objets or dans notre cas la fabrique doit contenir des opérations métiers comme l'ajout et suppression de fichier dans un chemin spécifique par exemple.

V. Annexe

Cette annexe du rapport, contient la liste complète, par paquetage, des entités de notre application. en donnant en face du nom de chaque entité sa responsabilité dans l'application en une phrase.

Package sysFileBusines: Il spécifie les éléments du système de fichier à produire.
ces entités sont:

- **L'interface SysFile:** l'interface qui spécifie un élément du système de fichier.
- **L'interface Dir:** l'interface qui spécifie un dossier.
- **L'interface RegFile:** l'interface qui spécifie les fichiers réguliers.
- **L'interface SybLink:** l'interface qui spécifie les liens symboliques.
- **La classe StdDir:** est une implémentation de l'interface Dir où le contenu est une List.
- **la classe StdRegFile:** est une implémentation de l'interface RegFile où le contenue est un tableau de bytes.
- **la classe StdSybLink:** est une implémentation de l'interface SybLink.
- **la classe PathFile:** généré le chemin de stockage du fichier.
- **la classe NameFile:** généré le nom d'un fichier en respectant les contraintes du nom.

Package FileVisitor: permet d'uniformiser l'accès au contenu d'un élément concret du système de fichier lors de l'exécution
ces entités sont:

- **L'interface Visitor:** Définit les méthodes de visite pour chaque classe du modèle qui implémente.
- **la classe ShowFileVisitor:** prendra en charge la forme des éléments à l'affichage par

exemple gérer les icônes des éléments du système de fichier à l'affichage.

Package client: Il spécifie les éléments du système de fichier à produire.
ces entités sont:

- **L'interface HttpHandler:** l'interface java utilisée contenant les méthodes permettant de gérer les requêtes http.
- **La classe ClientHttpHandler:** cette classe implémente l'interface **HttpHandler** et l'interface **Visitor** .
- **La classe HttpTest:** est la classe cliente .

Package Biblio:
ces entités sont:

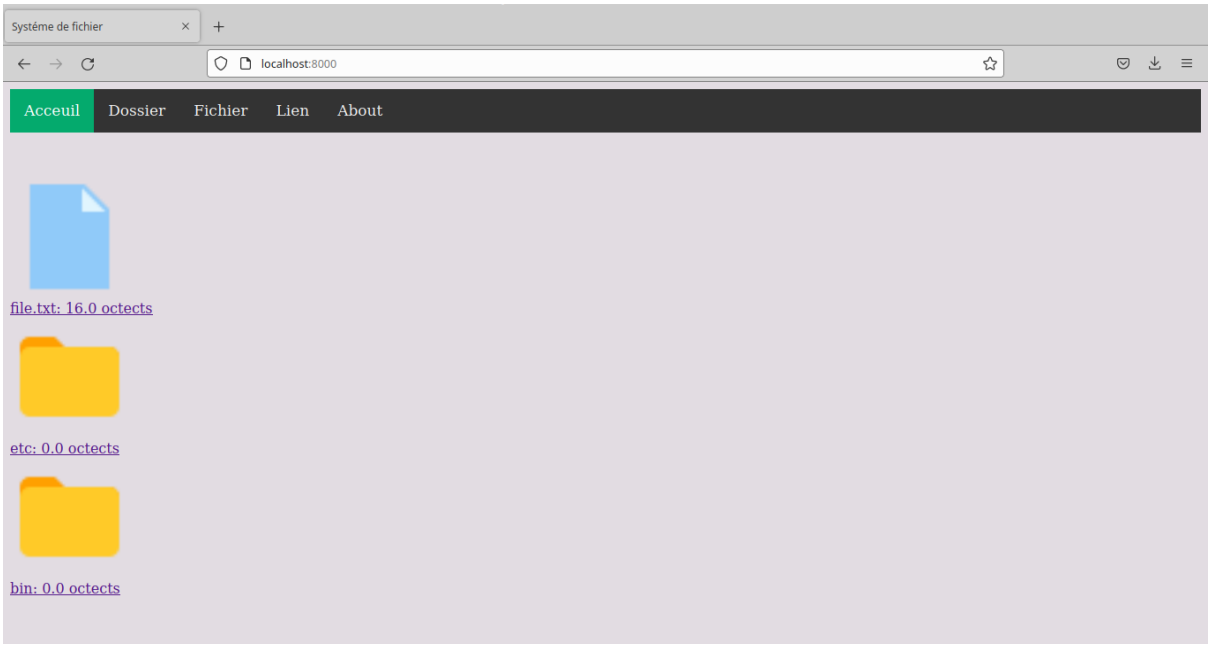
- **L'interface SysCreator:** contient des méthodes de fabriques relative à différentes variantes abstraites d'un fichier(dossier, fichier régulier et lien symbolique).
- **La classe StdSysCreator:** est une implémentation de l'interface SysCreator et l'interface Visitor.

VI. Javadoc

Nous avions produit la javadoc de l'ensemble de notre application, après l'avoir lue, dans un sous-dossier intitulé javadoc de notre archive.

VII. Interfaces Web

1. page Accueil



2. page “creer dossier”

VIII. Conclusion

A travers ce projet, nous avons eu l'occasion de mettre en œuvre toutes nos connaissances et les acquis du module vu en cours et en TP.

Cependant nous avons bien compris que pour arriver à la réalisation propre d'un logiciel on doit bien organiser l'architecture du logiciel tout en respectant les différents patrons.

La solution que nous avons proposée reste une idée générale de notre conception, mais certainement des modifications et des améliorations peuvent être apportées.