Design Patterns

OS Virtuel

Axel Gendillard Audoin De Chantérac

Étudiant ingénieur
Groupe ESEO

16 décembre 2015

Sommaire

Incrément 1 : Singleton

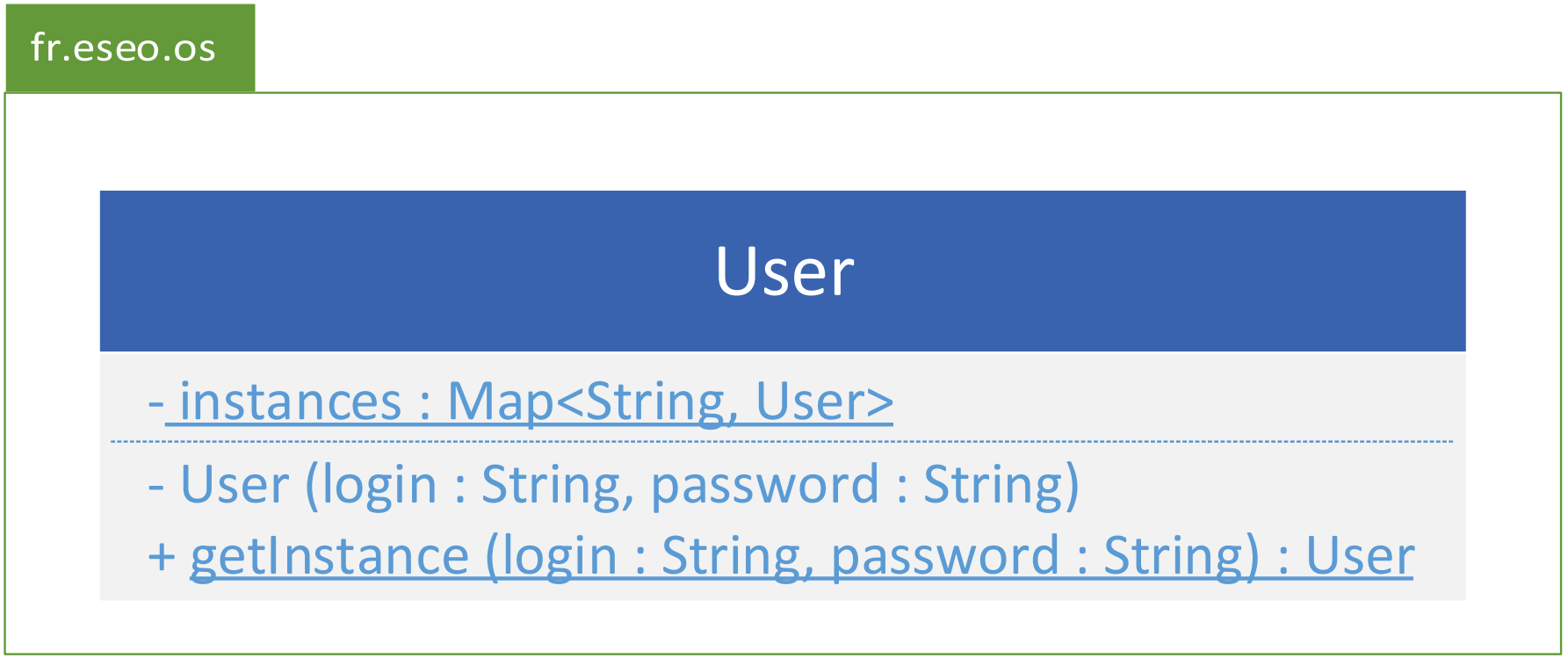
Objectifs

- Créer un nouvel utilisateur à chaque validation ;
- Avoir un utilisateur unique par login ;
- Créer plusieurs terminaux par utilisateur ;
- Interdir la création d'utilisateur hors de la fenêtre.

Patron Singleton

- Garantir l'unicité de l'instance d'une classe ;
- Accéder globalement à l'instance.

Incrément 1 : Singleton - UML



Incrément 2 : Observer

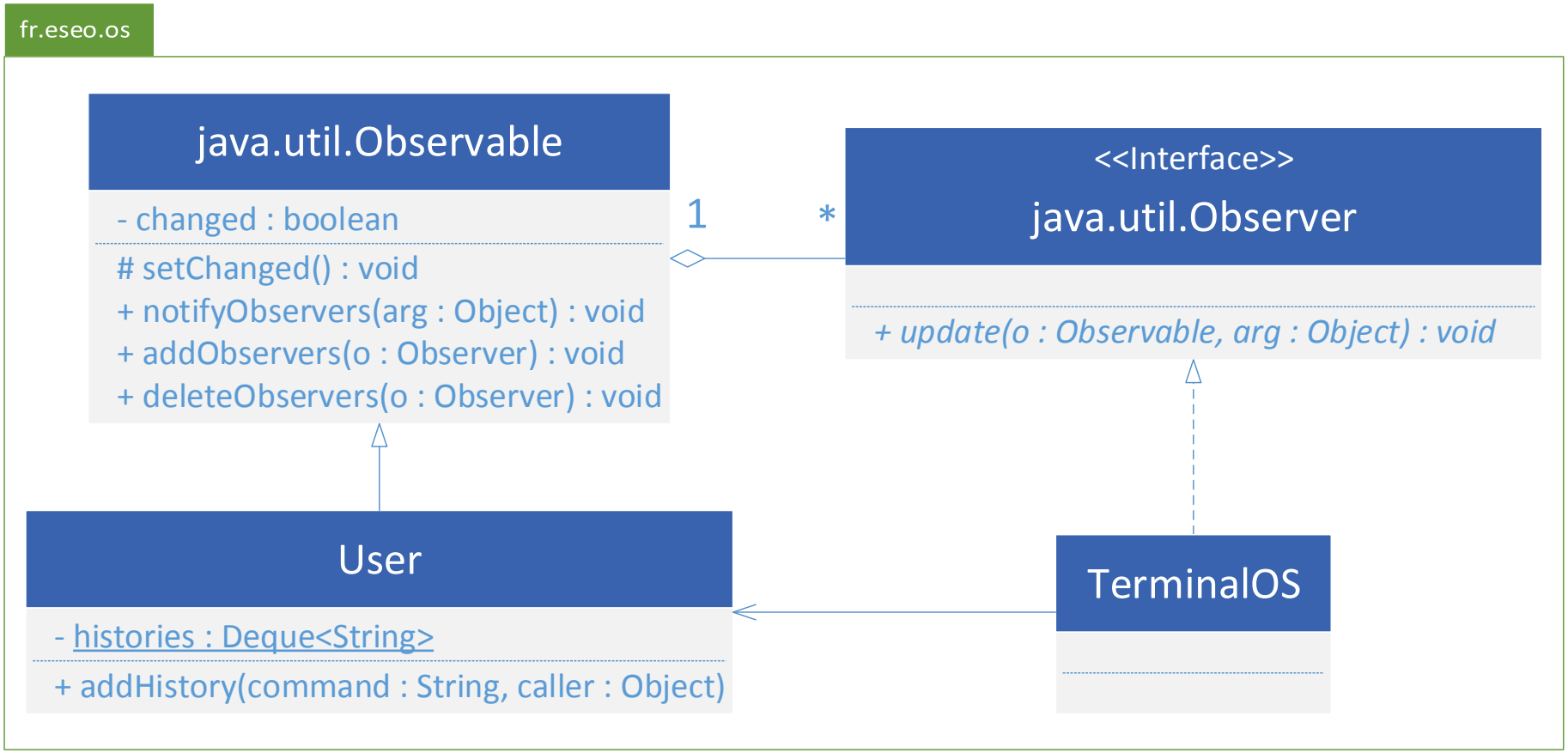
Objectifs

- Synchroniser les historiques des terminaux ;
- Mettre à jour tous les terminaux lors de l'entrée d'une commande.

Patron Observer

- Permet de mettre à jour les objets observés en fonction de l'objet observateur.

Incrément 2 : Observer - UML



Incrément 3 : Visiteur

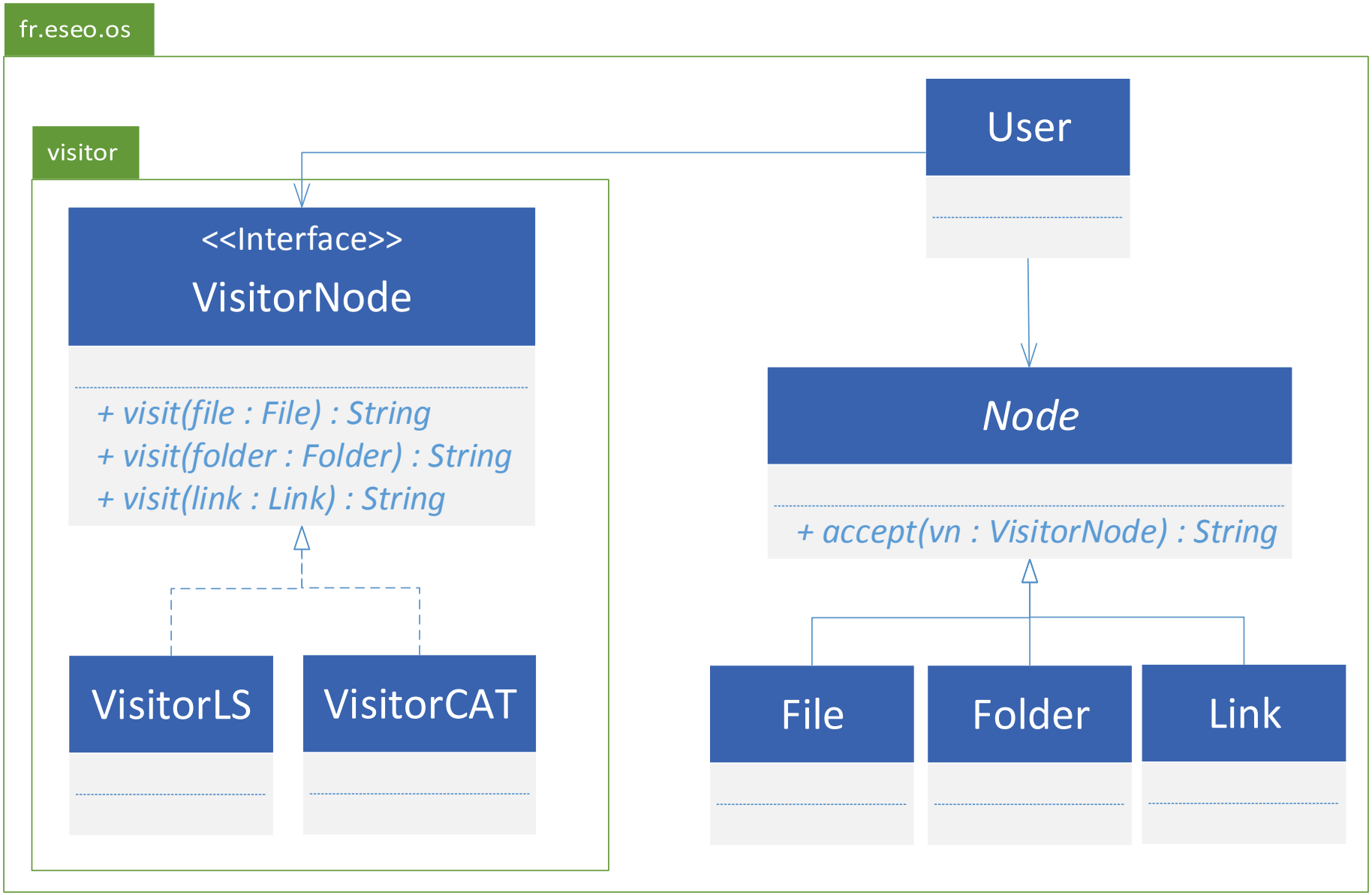
Objectifs

- Interpréter des commandes du terminal (ls et cat) pour chaque type de noeud ;
- Utiliser le patron Visitor.

Patron Visiteur

- Sépare l'algorithme de la structure de données ;
- Chaque élément implémente une méthode d'acceptation de visiteur ;
- Chaque visiteur implémente une méthode *visit()* pour chaque élément (file, folder et link).

Incrément 3 : Visiteur - UML



Incrément 4 : Commande

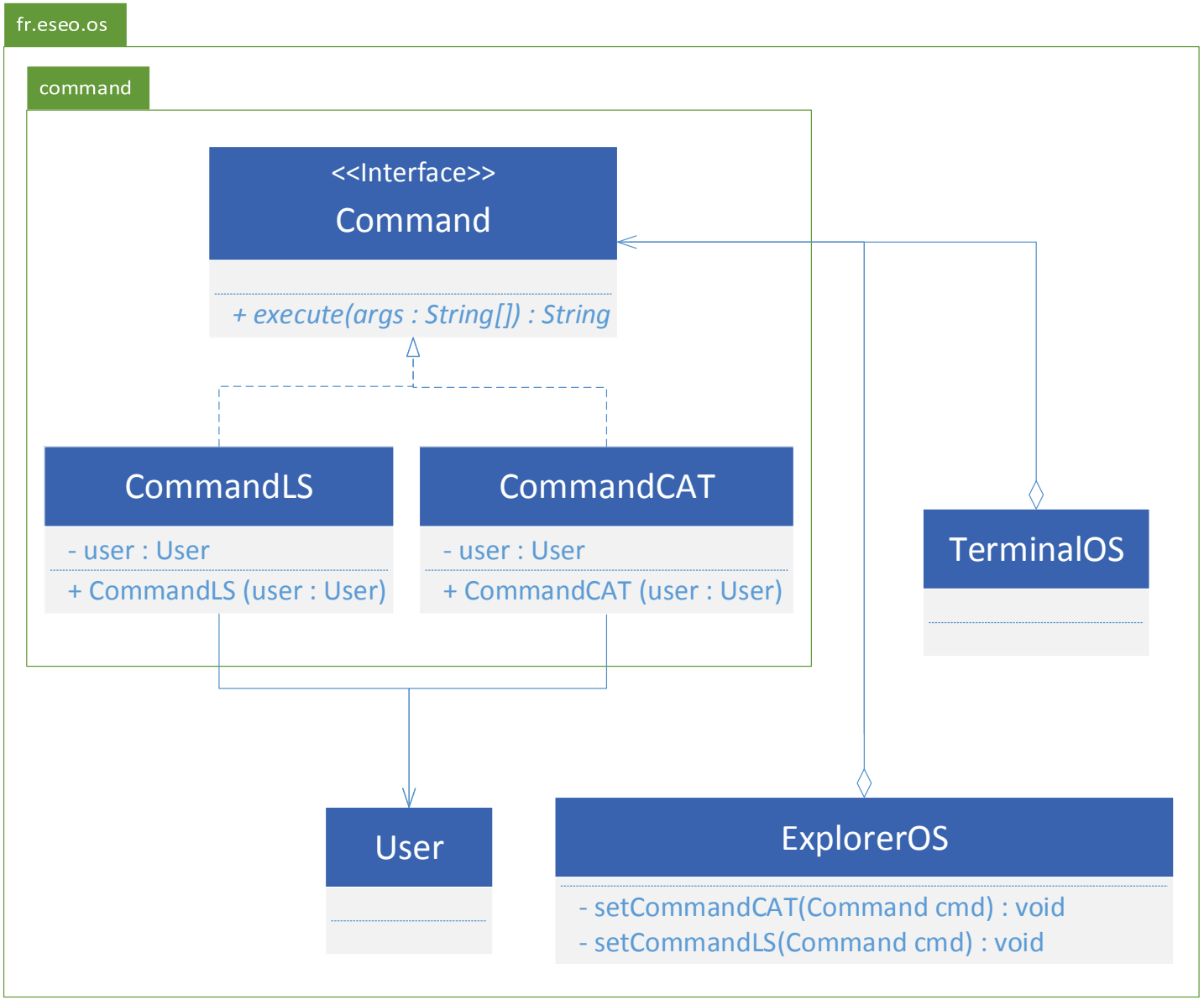
Objectifs

- Réutiliser les traitements interprétés par le terminal ;
- Exprimer de manière unique le traitement des commandes ls et cat.

Patron Commande

- Isole une requête ;
- Ces requêtes peuvent provenir de plusieurs émetteurs (*TerminalOS* et *ExplorerOS*) ;
- Ces émetteurs doivent produire la même requête ;
- Ces requêtes doivent pouvoir être annulées.

Incrément 4 : Commande - UML



Incrément 5 : Commande ++

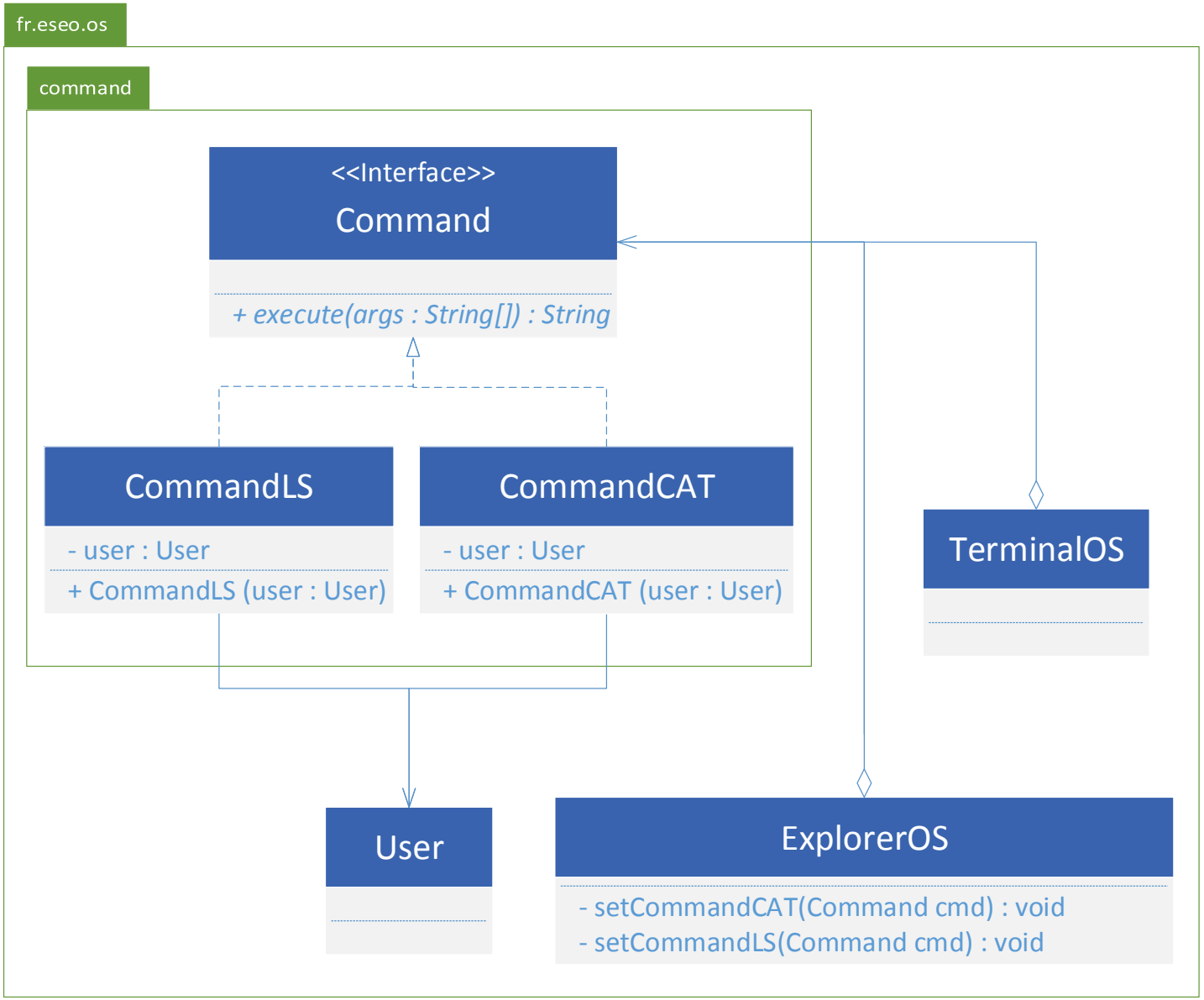
Objectifs

- RM : Supprimer un dossier, un fichier ou un lien ;
- MKDIR : Créer un dossier ;
- TOUCH : Créer un fichier ;
- LN : Créer un lien.

Patron Commande

- Isole une requête ;
- Ces requêtes peuvent provenir de plusieurs émetteurs (*TerminalOS* et *ExplorerOS*) ayant le même focntionnement ;
- Ces requêtes doivent pouvoir être annulées.

Incrément 5 : Commande - UML



Incrément 6 : Proxy

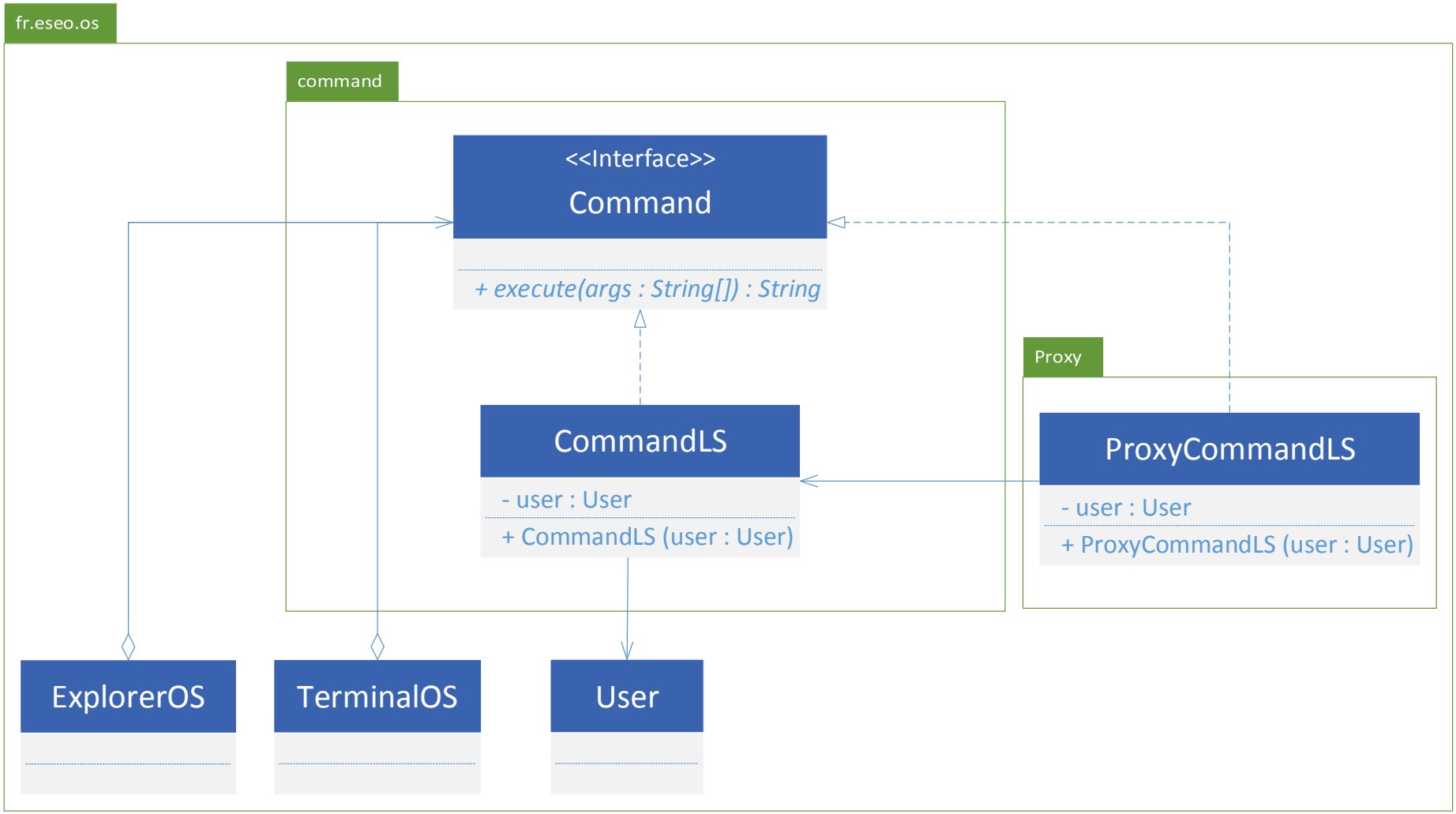
Objectifs

- Lecture seule : Pas de modification des données des utilisateurs ls et cat sont autorisées ;
- Lecture/écriture : Les commandes mkdir, touch, ln et rm sont autorisées.

Patron Proxy

- Rôle d'aiguilleur en fonction des droits de l'utilisateur ;
- Couche d'abstraction entre le client et la commande.

Incrément 6 : Proxy - UML



Bonus : Gestion de l'historique au clavier

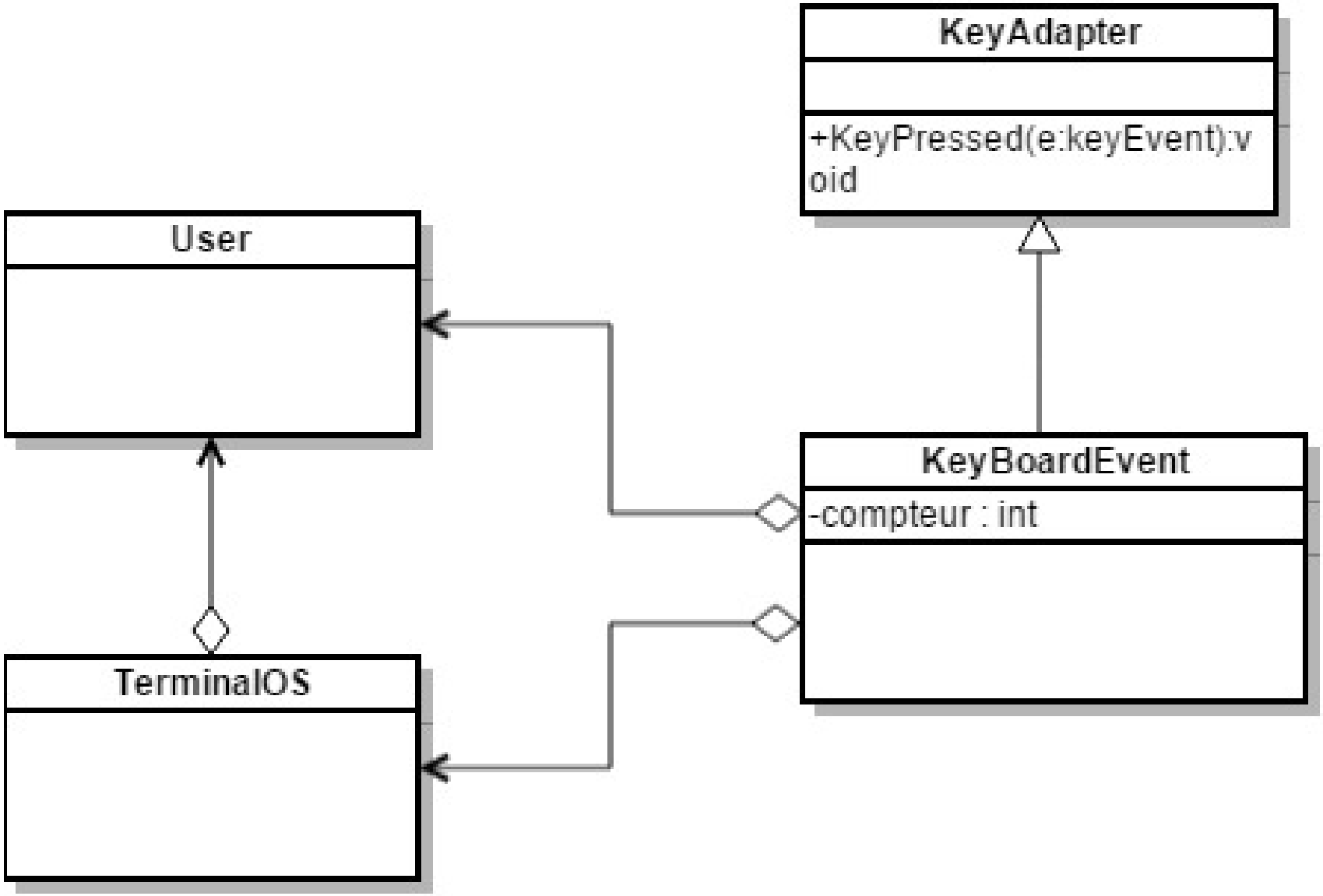
Objectifs

- Implémenter un gestionnaire d'historique au clavier (flèche du haut et du bas)
- Utiliser la classe *java.awt.event.KeyAdapter*

Implémentation

- Création classe *KeyBoardListener* qui hérite de *KeyAdapter*
- Création d'un *listener* sur les événement du clavier
- La méthode *handleCommand()* est appelée via la touche *Enter*.

Bonus : UML


KeyAdapter
+KeyPressed(e:keyEvent):void
User
KeyBoardEvent
-compteur : int
TerminalOS

Bonus : Un peu de code pour finir

```
@Override
public void keyPressed(KeyEvent e) {
    if (e.getKeyCode()==KeyEvent.VK_UP) {

        if(user.getHistories().size()>0 && user.getHistories().size()>=compteur){
            --compteur;
            this.terminal.update(user, e);
            currentHistories= this.user.getHistories().toString();
            System.out.println(currentHistories);
            if(compteur<=0){
                compteur=0;
            }
            command=this.user.getCommandHistories().get(compteur);
            terminal.getCommandTA().setText(currentHistories+command);
```