**Compte rendu du projet**

**d’Architecture Logicielle**

**FRAMEWORK**

**DE LOGGING**



*Enseignant :* Nicolas Andre

Sommaire

[Introduction 3](#_Toc412733350)

[I). Dossier utilisateur 4](#_Toc412733351)

[II). Dossier de maintenance 5](#_Toc412733352)

[Architecture des classes : 5](#_Toc412733353)

[Conclusion 7](#_Toc412733354)

## Introduction

Dans le cadre du cours d’Architecture Logicielle, il nous a été demandé de concevoir et de développer un framework de logging.

Le but est de créer un système permettant de gérer des messages d’information, de debug ou d’erreur afin de mieux comprendre le fonctionnement d’un programme.

Ces messages générés pourront être afficher ou sauvegarder dans une console, dans un fichier ou dans une base de données.

## I). Dossier utilisateur

## II). Dossier de maintenance

### Architecture des classes :

* Package fr.esiea.loggingfw
  + Ourlogger : classe principale du logger.
    - Champs :
      * name : pour différencier les différents logs. généralement la source du log
      * level : le niveau de log (debug, error...)
      * target : la liste des cibles qu'on va vouloir logger
      * formatter : permet l'uniformisation de l'affichage en console
    - Méthodes :
      * Les deux constructeurs qui instancie le logger
      * Les trois méthodes de debug, d'erreur et d'info qui définissent de quel level sera le log créé.
      * La méthode printLog() qui effectuera le log dans les différentes cibles du logger.
      * Les getters et setters pour le niveau de log
      * Les méthodes setTarget(String), setTarget(AbstractTarget), addTarget(AbstractTarget), removeTargets() permettent d'ajouter des cibles au log ou au contraire à n'en définir qu'une seule.
  + ReadPropertiesFile : classe permettant de récupérer et d'utiliser le fichier properties du logger
    - Champs :
      * config : de type Properties, contient les propriétés à charger
    - Méthodes :
      * La méthode readProperties permettant de charger et lire les informations du fichier config.properties.
      * getter du champ config
* Package fr.esiea.loggingfw.format
  + LoggerFormatter : classe permettant d'uniformiser l'affichage en console des logs
    - Champs : aucun
    - Méthodes :
      * Deux méthodes, l'une affichant le name, le level et le message du log et l'autre y ajoutant la date.
* Package fr.esiea.loggingfw.levels
  + LoggerLevel : énumération contenant les différents niveaux de logs
    - Enum : DEBUG, INFO, ERROR
    - Champs :
      * valeurLevel : pour récupérer la bonne valeur de l'énum
    - Méthodes :
      * getter et setter du champ
* Package fr.esiea.loggingfw.log
  + Log : classe permettant de créer un log avec ses paramètres.
    - Champs :
      * name : pour différencier les différents logs. généralement la source du log
      * level : le niveau de log (debug, error...)
      * message : contenu du log donnant l'information
      * date : date et heure du log
    - Méthodes :
      * deux constructeurs, un avec tous les champs, l'autre sans la date

## Conclusion

Les parties qui ont posé problème :

-

Les points à améliorer :

-