L3 Informatique
Université de Tours

Module : Patrons de Conception
Projet : Refactoring d'un outil d'extraction d'informations d'un fichier texte vers un document HTML

Membres du groupe :

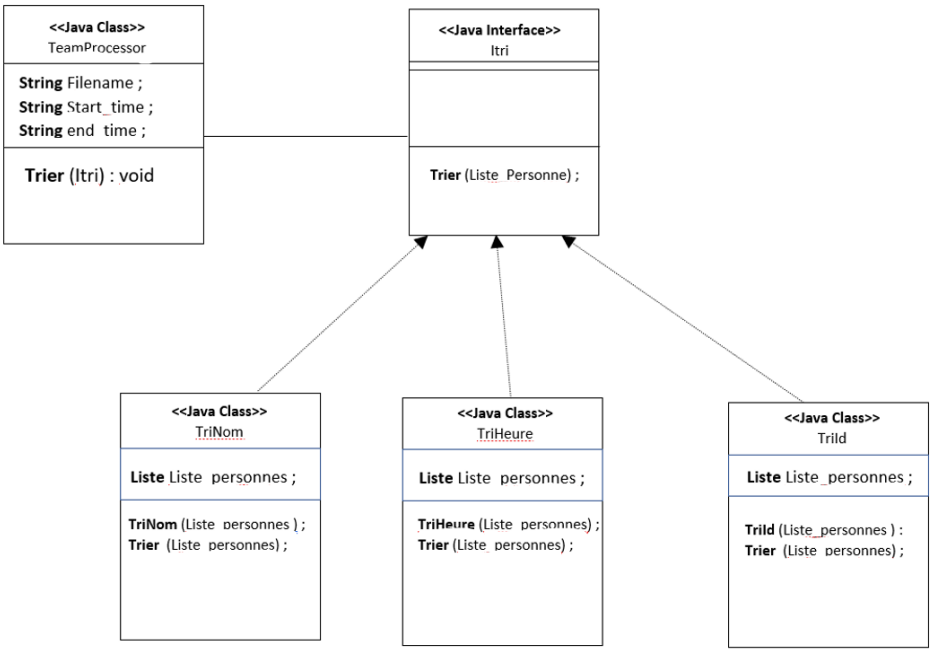
- Abdelouhab Aimad
- Ait Hadji Aghilas
- Bikoli Sorelle Ingrid
- Dahmani Selma

Environnement de développement : IntelliJ IDEA community 2021.1 (jdk 11)

Designs Pattern utilisées :

1.Strategy Pattern : Le Strategy pattern a été utilisé pour implémenter le tri (**Question3**). Vu que pour une même fonctionnalité (tri), les paramètres varient (heure, nom, id, etc.), il est préférable de séparer les détails d'implémentation de chaque Tri personnalisé de son fonctionnement général. Ce qui permet d'avoir une flexibilité lors d'un ajout ou bien de la modification du code d'une des méthodes (**Exemple** : rajouter Tri par date). Aussi, et vu que l'appel de la fonction du Tri se fait dorénavant par l'intermédiaire de **l'interface Itri**, une nouvelle couche d'abstraction est rajoutée entre la classe qui appelle les méthodes de tri et leur implémentation, permettant ainsi de modifier le code de ces méthodes sans changer la syntaxe de l'instruction qui effectue l'appel. Ce qui respecte un des principes SOLID [1] : **le Single Responsibility principle.**

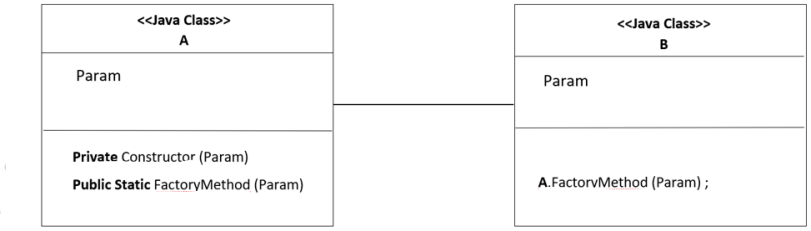
Diagramme de classe correspondant :



[1] SOLID (S.O.L.I.D). C'est un acronyme composé de 5 lettres, de 5 principes de bases pour la Programmation Orientée Objet (POO). En suivant ces principes, cela permettra de développer une application **maintenable, plus fiable et plus robuste.**

2.Static Factory method pattern (Question 4) : Une variante des DP Factory, ce design pattern est utilisé pour organiser la création des objets. Son principe de base consiste en l'instanciation d'une classe par l'intermédiaire d'une méthode static public, qui se substitue au constructeur. Ce dernier aura une visibilité **privée** et ne sera accessible que via la méthode static. La création des objets se fera donc en appelant notre Factory (méthode static), au lieu du constructeur.
L'avantage principal d'une méthode statique est la possibilité qu'elle soit appelée sans être instanciée (**new**). Aussi, et à l'encontre du constructeur une méthode peut avoir un nom, ce qui améliore la lisibilité du code source.
Dans le projet, nous avons implémenté le DP Static Factory method dans les classes **People**, **TEAMSAttendanceList** et **TEAMSAttendanceListAnalyzer**

Diagramme de classe correspondant



Iterator Pattern : Pour parcourir les différents objets du programme nous avons implémenté un Iterator (**Question2**), ce pattern permet de parcourir une structure de données, en faisant abstraction de son type (Liste, collection, etc).