1) But

Le but de ce travail est de mettre en pratique les concepts de type de données abstrait (TDA) et de la programmation par contrat. En outre, l'étudiant aura à mettre en œuvre les notions de classe, l'implantation des services déclarés dans une interface (type de données abstrait) à l'aide d'une liste chaînée sans remorque (Classe `Noeud<T>`) tout en respectant les conventions d'écriture du code Java vues en classe.

2) Travail demandé

Vous devez implémenter l'interface `FileAttenteTda<T>` à travers une classe nommée `FileAttenteChaineeImpl` en utilisant la classe `Noeud<T>`. Votre implantation doit être générique et respecter intégralement l'interface fournie. Aucun CHANGEMENT ne doit être fait dans les interfaces `FileAttenteTda<T>` et `OrdonnableParPrioriteEtDateHeure`.

L'entête de la classe `FileAttenteChaineeImpl` est :

```
public class FileAttenteChaineeImpl<T extends OrdonnableParPrioriteEtDateHeure>
                                                    implements FileAttenteTda<T>{
    ...
}
```

Étant donné que l'interface `FileAttenteTda<T>` hérite de l'interface `Iterable<T>`, vous devez redéfinir la méthode `iterator()` dans votre classe `FileAttenteChaineeImpl`. Cette méthode qui doit retourner un objet `Iterator<T>` à travers lequel on peut parcourir notre file d'attente pour appliquer des traitements spécifiques sur les membres qu'elle contient.

Pour tester votre implémentation, vous devez écrire des tests unitaires basés sur JUnit (Approche JUnit Tests 4 vue en classe) en créant une classe nommée `FileAttenteChaineeImplTest` qui permettra de tester toutes les méthodes de la classe `FileAttenteChaineeImpl`. Utilisez la classe `Membre` comme élément de la file d'attente lors de vos tests. Dans la classe de test `FileAttenteChaineeImplTest`, vous devez écrire, au moins, une méthode de test par méthode définie dans la classe `FileAttenteChaineeImpl`.

Pour une bonne séparation entre les différentes interfaces/classes, créez les paquetages suivants :

a) `ca.uqam.inf2120.tp2.adt` : contiendra les interfaces `FileAttenteTda` et `OrdonnableParPrioriteEtDateHeure`.
b) `ca.uqam.inf2120.tp2.adt.impl` : contiendra les classes `FileAttenteChaineeImpl` et `Noeud`.
c) `ca.uqam.inf2120.tp2.adt.test` : contiendra les classes `FileAttenteChaineeImplTest` et `Membre`.

3) Livrables et date de remise

3.1. Livrables

Via Moodle, vous devez remettre une copie électronique compressée (.zip ou .rar) qui comprend :

a) Le dossier du projet composé des trois (3) paquetages mentionnés précédemment avec toutes les classes/interfaces développées.

b) Une page de présentation (en Word) comprenant votre nom, votre prénom, le sigle du cours, le groupe du cours, le nom de l'enseignant.

3.2. Date de remise

Votre travail doit être remis le mercredi 29 mars 2017 avant 23H55.

Pour tout travail remis en retard, les pénalités seront appliquées selon la formule suivante : **`Nombre de points de pénalité = m / 144`**, où m est le nombre de minutes de retard par rapport à l'heure de remise. Aucun travail ne sera accepté après cinq (5) jours de retard.

4) Pondération

a) **Présentation (2%)** : Page de présentation clairement identifiée.

b) **Code source de l'implantation (50%)**: Clarté du code, indentation, commentaires, choix des identificateurs, respect des spécifications de l'interface, optimisation du code et l'absence de redondance dans le code et bonne utilisation de la généricité.

c) **Code source Programme de test (10%)** : Utilisation de l'approche JUnit Test 4, couverture et pertinence des différents scénarios de tests.

d) **Exécution des tests unitaires de correction (38%)**: Couverture, pertinence des différents scénarios de tests et exactitude des résultats.

Notez-bien que la note 0 sera attribuée à tout programme qui ne compile pas.

Règles de programmation

a) Les identifiants doivent être aussi significatifs que possible et se conformer aux conventions standards en ce qui concerne l'emploi des majuscules et minuscules.

b) Ne déclarez pas de variables que vous n'utilisez pas.

c) N'utilisez pas une même variable pour des usages différents.

d) Ne faites pas de copier-coller des commentaires Javadoc placés au début des méthodes de l'interface dans la classe qui l'implémente.

e) N'employez pas de variables "globales" dans le programme client. Si une méthode nécessite des valeurs, passez ces valeurs en tant que paramètres. Si une méthode produit ou modifie des valeurs, elles doivent être retournées en tant que résultats ou paramètres.

f) Une bonne structure est très importante. Décomposez vos algorithmes ou méthodes en plusieurs algorithmes/méthodes de manière significative en toute occasion où une telle décomposition améliore la clarté. Si nécessaire, n'hésitez pas d'ajouter des méthodes privées pour éviter d'avoir des méthodes publiques dont le nombre de lignes est supérieur à 25.

g) Votre programme doit être indenté de façon systématique et les blocs d'instructions doivent être bien commentés et séparés par une ligne blanche pour permettre une bonne aération.

Le travail est strictement individuel. Le règlement sur le plagiat sera appliqué sans exception. Vous devez ainsi vous assurer de ne pas échanger du code avec des collègues, ni de laisser sans surveillance votre travail au laboratoire. Vous devez également récupérer sans faute toutes vos impressions de programme au laboratoire.