Travail pratique 2

1) But

Le but de ce travail est de mettre en pratique les concepts de Type de Données Abstrait (TDA), et de la programmation générique par contrat. En outre, l'étudiant aura à mettre en œuvre :

a) L'implémentation des méthodes déclarées dans une interface `EquipeTda<T>` à l'aide d'une liste simplement chaînée en utilisant la classe `Noeud<T>`.
b) L'utilisation d'un itérateur (`Iterator<T>`) pour le parcours d'une équipe de type `EquipeTda<T>`.
c) Les tests unitaires automatisés (`JUnit`) pour tester les méthodes implémentées.

2) Définition du TDA Équipe

Nous considérons, dans le contexte de ce travail pratique, une équipe comme un ensemble d'opérations pour la gestion des membres appartenant à cette dernière. Les opérations permises sont : l'ajout d'un membre, le retrait d'un membre, le remplacement d'un membre par un autre, l'existence d'un membre, etc.

3) Interface et implémentation

L'interface `EquipeTda<T>` définit le TDA Équipe avec toutes les opérations (méthodes) déclarées. Vous devez implémenter toutes les méthodes de cette interface dans une classe nommée `EquipeImpl<T>`, en utilisant une liste de nœuds simplement chainés comme structure de données.

Vous devez utiliser la classe `Noeud<T>` pour représenter chacun des nœuds de cette liste simplement chainée. Chaque nœud contient un élément (la référence de la donnée stockée) de type `T`, et la référence sur le prochain nœud de type `Noeud<T>`. L'entête de votre classe doit être exactement comme suit :

```
public class EquipeImpl<T> implements EquipeTda<T> {...}
```

Vous trouverez ci-dessous plus de détails sur la description de la classe `EquipeImpl<T>`.

Le paquetage : `ca.uqam.h2024.inf2120.tp2.adt.impl`		
Le nom de la classe : `EquipeImpl<T>`		
Le nom des attributs d'instance	**Type**	**Description**
Une équipe (`uneEquipe`)	`Noeud<T>`	C'est le premier nœud de la liste simplement chainée représentant le premier membre dans l'équipe. Sa valeur doit être `null` lorsqu'il n'y a aucun membre dans l'équipe (équipe vide).
Le nombre de membres (`nbMembres`)	`short`	Le nombre de membres dans l'équipe. Si sa valeur est 0, cela signifie que l'équipe ne contient aucun membre. Cet attribut doit être mis à jour après chaque opération d'ajout ou de retrait d'un membre dans l'équipe.
Le constructeur		
Un seul constructeur sans argument qui crée une équipe vide (les attributs `"uneEquipe"` et `"nbMembres"` sont respectivement `null` et `0`).		
Les méthodes à implémenter		
Toutes les méthodes déclarées dans l'interface `EquipeTda<T>` doivent être implémentées. Assurez-vous de bien lire la `Javadoc` de chaque méthode pour bien comprendre les exigences (les règles d'affaires) de cette dernière. Étant donné que la classe `EquipeImpl<T>` implémente l'interface `EquipeTda<T>` qui hérite de l'interface `Iterable<T>`, vous devez redéfinir la méthode `Iterator<T> iterator()`. Cette méthode doit retourner une instance de type `EquipeIterateur<T>` (fournie avec l'énoncé de ce travail pratique) pour parcourir une équipe afin d'appliquer une opération spécifique sur chaque membre. Aucune autre méthode publique ne doit être ajoutée, cependant, vous pouvez ajouter des méthodes privées que vous jugez nécessaire pour faciliter votre travail.		

4) Les tests

Pour tester votre implémentation, vous devez écrire des tests unitaires basés sur l'approche JUnit 4 vue en classe en créant une classe nommée `EquipeImplTest` qui testera toutes les méthodes de la classe `EquipeImpl<T>`. Utilisez la classe `Membre` (fournie avec l'énoncé de ce travail pratique) comme membre de l'équipe lors de vos tests. Dans la classe de test `EquipeImplTest`, vous devez écrire, au moins, une méthode de test par méthode définie dans la classe `EquipeImpl<T>`.

4. Travail demandé

4.1. Organisation de votre projet

a) Vous devez créer votre projet nommé `tp2`, et ajouter les paquetages suivants :
   i. `ca.uqam.h2024.inf2120.grpe30.tp2.adt`
   ii. `ca.uqam.h2024.inf2120.grpe30.tp2.adt.impl`
   iii. `ca.uqam.h2024.inf2120.grpe30.tp2.adt.impl.tests`

b) Vous devez ajouter l'interface `EquipeTda<T>`, et la classe d'exception `PositionException` dans le paquetage `ca.uqam.h2024.inf2120.grpe30.tp2.adt`.

c) Dans le paquetage `ca.uqam.h2024.inf2120.grpe30.tp2.adt.impl`, vous devez créer votre classe `EquipeImpl<T>` qui implémente l'interface `EquipeTda<T>`, et ajouter les classes `Noeud<T>` et `EquipeIterateur<T>`. Vous devez implémenter toutes les méthodes déclarées dans l'interface `EquipeTda<T>`, et la méthode `Iterator<T> iterator()`. Votre implémentation doit être générique, et elle doit respecter intégralement les signatures des méthodes déclarées dans l'interface `EquipeTda<T>`. Autrement dit, aucun CHANGEMENT ne doit être fait dans cette interface.

d) Dans le paquetage `ca.uqam.h2024.inf2120.grpe30.tp2.adt.impl.tests`, vous devez créer votre classe de test `EquipeImplTest`, et ajouter la classe `Membre`.
   i. La classe `EquipeImplTest` doit tester toutes les méthodes implémentées de la classe `EquipeImpl<T>`. Assurez-vous d'ajouter tous les scénarios de test pour chaque méthode implémentée.
   ii. La classe `Membre` remplace le type `T` lors des tests unitaires.

Notez bien que l'interface `EquipeTda<T>`, les classes `PositionException`, `Noeud<T>`, `EquipeIterateur<T>`, et `Membre` sont fournies avec l'énoncé de ce travail pratique.

4.2. Ce que vous devez remettre

La date de remise du travail pratique 2 est le **jeudi 28 mars 2024** avant **23H55**. Via Moodle, vous devez remettre une copie électronique de votre projet `Tp2` compressée (.zip ou .rar) qui contient :

a) L'interface `EquipeTda<T>`.

b) Les classes suivantes : `PositionException`, `Noeud<T>`, `EquipeIterateur<T>`, `EquipeImpl<T>`, `EquipeImplTest`, et `Membre`.

4.3. Pénalités

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.

4.4. Plagiat

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 Java avec des collègues.

5. Pondération.

a) Code source de la classe d'implémentation `EquipeImpl<T>` (50%) : Quelques éléments à considérer pour l'évaluation du code source :

   i. L'implémentation générique en gardant le type `T`.

   ii. Le respect intégral des signatures des méthodes déclarées dans l'interface `EquipeTda<T>`.

   iii. La mise en œuvre de toutes les règles définies dans la Javadoc de chaque méthode.

   iv. Les bonnes structures de contrôle non redondantes.

   v. Des commentaires pertinents.

b) Code source de la classe de test `EquipeImplTest` (18%) : L'utilisation de l'approche JUnit 4, la couverture et la pertinence des différents scénarios de test sont des éléments qui sont considérés lors de l'évaluation de cette partie. La note zéro (0) sera attribuée à ce niveau pour tout programme Java qui ne compile pas ou tout programme Java qui NE contient PAS les différentes classes ci-dessus demandées.

c) Exécution du programme (32%) : La couverture, la pertinence des différents scénarios, et l'exactitude des résultats sont des éléments qui sont considérés lors de l'évaluation de cette partie.