Compte rendu du projet programmation fonctionnel

Introduction :

La **Queue** classe est une structure de données premier entré, premier sorti (FIFO) implémentée en tant que classe de cas dans Scala. Il a deux listes : une **in** liste et une **out** liste. La **enqueue** méthode ajoute un élément au début de la **in** liste, et la **dequeue** méthode supprime et renvoie le dernier élément de la **out** liste. Si la **out** liste est vide, la **in** liste est inversée et utilisée comme **out** liste.

Développement :

La **Queue** classe a deux listes : **in** et **out** . La liste **in** est utilisée pour stocker les nouveaux éléments ajoutés à la file d'attente, et la **out** liste est utilisée pour stocker les éléments qui ont déjà été supprimés de la file d'attente. Lorsqu'un élément est supprimé de la **out** liste, il est supprimé de la file d'attente. Lorsque la **out** liste est vide, la **in** liste est inversée et utilisée comme **out** liste pour permettre aux éléments d'être supprimés dans le bon ordre.
Nous avons implémenté plusieurs méthodes sur la **Queue** classe :

- **enqueue(x:T)**: Cette méthode ajoute un élément **x** au début de la file d'attente. Il crée une nouvelle file d'attente avec la même **out** liste, mais avec **x** ajouté au début de la **in** liste.
- **dequeue():(T,Queue[T])**: Cette méthode supprime et renvoie le dernier élément de la file d'attente. Si la **out** liste n'est pas vide, il renvoie la tête de la **out** liste et une nouvelle file d'attente avec la queue de la **out** liste. Si la **out** liste est vide, il inverse la **in** liste et l'utilise comme **out** liste pour la nouvelle

file d'attente. Si les deux listes sont vides, il lance un **NoSuchElementException**.

- **headOption():Option[T]**: Cette méthode renvoie le premier élément de la file d'attente, s'il existe. Il cherche **out** d'abord la tête de la liste, et s'il n'est pas trouvé, il cherche la tête de la **in** liste inversée.
- **isEmpty:Boolean:** Cette méthode retourne **true** si les listes inet sont vides, et sinon.out **false**
- l**ength():Int**: Cette méthode renvoie la longueur de la file d'attente, qui est la somme des longueurs des listes inet .out
- **rearOption():Option[T]:** Cette méthode renvoie le dernier élément de la file d'attente, s'il existe. Il cherche **in** d'abord la tête de la liste, et s'il n'est pas trouvé, il cherche la tête de la **out** liste inversée.
- **toList():List[T]:** Cette méthode convertit la file d'attente en une liste à liens simples. Il concatène la **in** liste inversée et la **out** liste.
- **map[B](f:T=>B):Queue[B]:** Cette méthode applique une fonction **f** à chaque élément de la file d'attente et renvoie une nouvelle file d'attente avec les résultats. Il crée la nouvelle file d'attente en mappant la fonction sur les listes inet .out
- **foldLeft[B](z:B)(f:(B,T)=>B):B:** Cette méthode applique une fonction **f** à chaque élément de la file d'attente et un accumulateur, en commençant par la valeur initiale de l'accumulateur **z**. Il renvoie la valeur finale de l'accumulateur. Il convertit d'abord la file d'attente en liste, puis utilise la **foldLeft** méthode sur la liste.
- **dequeueTotal():(Option[T],Queue[T]):** Cette méthode est similaire à **dequeue()**, mais elle renvoie un **Option[T**] au lieu de a T et a **Queue[T]**. Si la file d'attente est vide, elle renvoie **(None, this)**. Sinon, il renvoie le résultat de **dequeue()** sous la forme d'un tuple enveloppé dans Some.
- **isEmptyWithMatch:Boolean:** Cette méthode renvoie **true** si la file d'attente est vide, et **false** sinon. Il utilise la correspondance de modèle pour vérifier si la file d'attente est égale à **Queue(Nil, Nil).**
- Pour tester toutes les méthodes de la **Queue** classe à l'aide de la bibliothèque de tests **MUnit**, vous pouvez définir une suite de tests qui étend **munit.FunSuite** et inclut des cas de test pour chaque méthode.

Pour utiliser la bibliothèque de test MUnit dans votre projet Scala, on doit l'ajouter en tant que dépendance dans votre fichier de configuration de construction. Si vous utilisez sbt, vous pouvez ajouter la ligne suivante à votre **build.sbt** fichier :

```
libraryDependencies += "org.scalameta" %% "munit" % "0.7.29" % Test
```

Cela ajoutera la bibliothèque MUnit en tant que dépendance pour la test configuration de votre projet.

Si vous utilisez un outil de build différent tel que Maven ou Gradle, vous devrez ajouter la configuration de dépendance appropriée pour votre outil de build. Consultez la documentation de MUnit pour plus d'informations.

Conclusion:

La **Queue** classe fournit plusieurs méthodes pour manipuler une structure de données de file d'attente FIFO. Il est implémenté sous la forme d'une classe de cas avec deux listes : une **in** liste et une **out** liste. La **enqueue** méthode ajoute un élément au début de la **in** liste, et la **dequeue** méthode supprime et renvoie le dernier élément de la **out** liste. Les autres méthodes proposent différentes manières d'accéder à la file d'attente et de la transformer, telles que **headOption**, **isEmpty**, **length**, **rearOption**, **toList**, **map**, **foldLeft**, **dequeueTotalet** et **isEmptyWithMatch**. Aucune difficulté particulière n'a été rencontrée dans la mise en œuvre de ces méthodes.