Travail pratique #2

Ce travail doit être fait individuellement.

Notions mises en pratique : respect d'un ensemble de spécifications, implémentation d'un TDA (avec un type générique) en utilisant une liste chainée comme structure de données.

1. Description générale

1.1 DÉFINITION DU TDA RONDE

Il s'agit de concevoir la classe `RondeChainee` qui implémente le TDA Ronde (défini dans l'interface `Ronde` fournie avec l'énoncé de ce TP).

Une ronde est une séquence d'éléments sans début ni fin, mais qui possède un **élément courant** (qui pointe sur un des éléments de la ronde) en tout temps, lorsque celle-ci n'est pas vide. Lorsque la ronde est vide (elle ne contient aucun élément), l'élément courant est égal à `null`.

On peut modifier l'élément courant d'une ronde en **tournant** vers la gauche ou vers la droite dans la ronde, c'est-à-dire en parcourant ses éléments vers la gauche ou vers la droite.

Un **parcours** (dans la ronde) est donné par un nombre entier :

- Un entier `n` positif indique de parcourir `n` éléments vers la droite.
- Un entier `n` négatif indique de parcourir `n` éléments vers la gauche.
- un entier égal a 0 indique de ne parcourir aucun élément

Une **combinaison** est une suite de parcours à effectuer dans la ronde, et est donnée dans un tableau de nombres entiers. Par exemple, la combinaison donnée par le tableau [3, -7, 23] spécifie le parcours de 3 éléments vers la droite, de 7 vers la gauche, suivi de 23 éléments vers la droite.

Une ronde n'accepte pas les éléments `null`, ni les doublons.

Une ronde **vide** est une ronde qui ne contient aucun élément.

Insertion : L'ajout d'un élément dans une ronde non vide se fait toujours à droite de l'élément courant. Dans une ronde vide, l'élément ajouté devient l'élément courant de cette ronde.

Retrait : Le retrait d'un élément consiste à supprimer l'élément courant. Après un retrait, l'élément courant pointe sur l'élément qui était à sa gauche, avant son retrait. Lorsque la ronde ne contient qu'un seul élément avant le retrait, l'élément courant devient `null` après le retrait.

Dans une ronde non vide, il est possible de tourner indéfiniment, que ce soit vers la gauche ou vers la droite.

Les différents services du TDA Ronde sont définis dans l'interface `Ronde` qui vous est fournie avec l'énoncé de ce TP.

Mélanie Lord
lord.melanie@uqam.ca

2. Implémentation

1.1 DÉTAILS ET CONTRAINTES D'IMPLÉMENTATION DE LA CLASSE RONDE CHAINÉE

Vous devez définir la classe `RondeChainee` qui implémente l'interface `Ronde` à l'aide d'une liste chainée de `MaillonRonde` (classe fournie). La classe `MaillonRonde` représente un maillon d'une chaine qui peut être chainée vers la gauche, et vers la droite. Le type générique `T` correspond au type des éléments dans la ronde (le type de l'élément dans le `MaillonRonde`). **L'entête de votre classe doit être exactement** :

```
public class RondeChainee<T> implements Ronde<T>
```

Lisez bien la Javadoc de toutes les méthodes de l'interface `Ronde` pour bien comprendre ce qu'elles doivent faire.

La classe `RondeChainee` doit contenir un constructeur sans argument (par défaut ou non) et LE SEUL attribut permis dans la classe `RondeChainee` est l'attribut d'instance `courant`, qui contient l'élément courant de la ronde (ou `null` si la ronde est vide). **Exemples** :

Suite d'insertions dans une ronde chainée initialement vide :

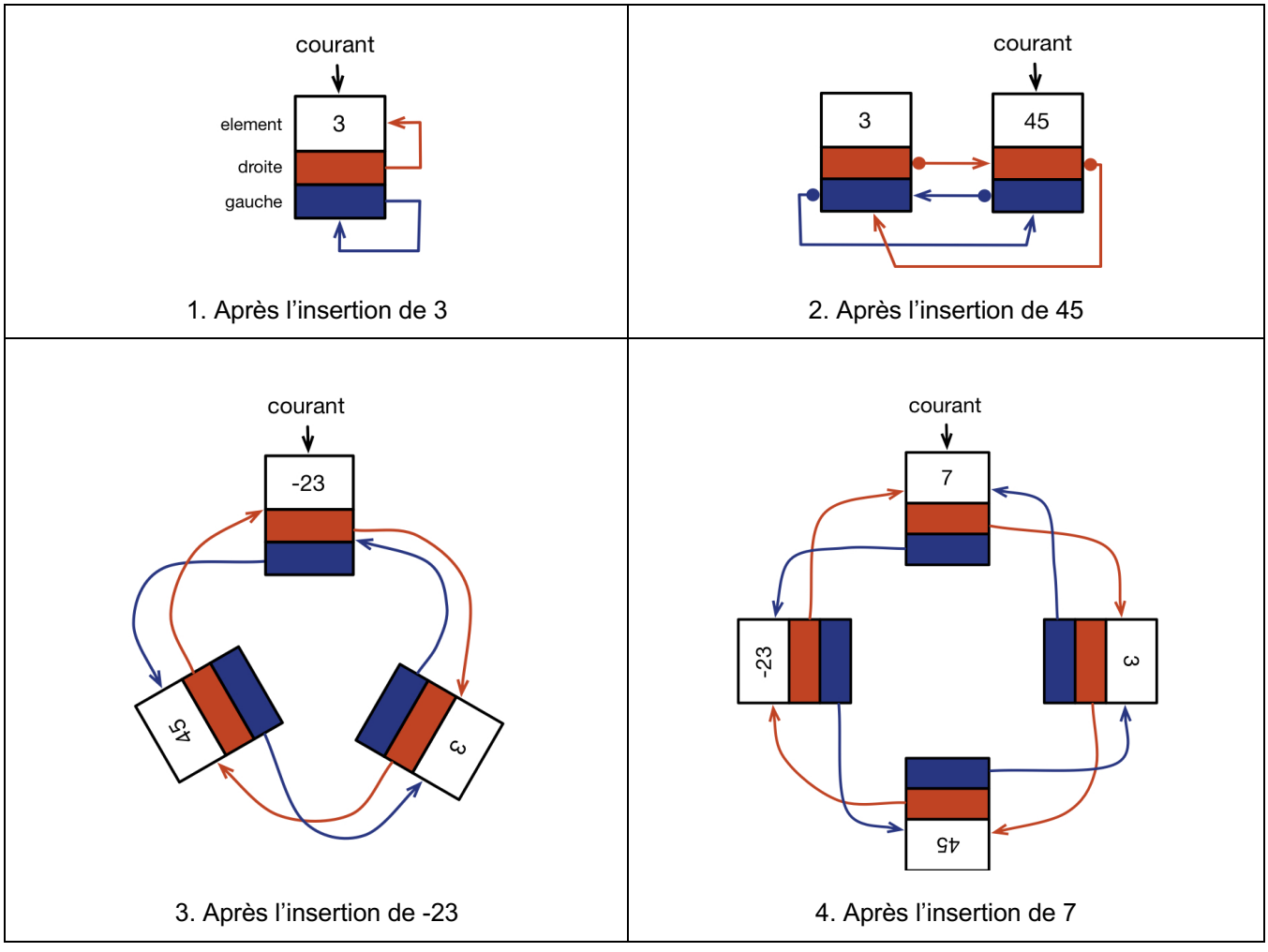
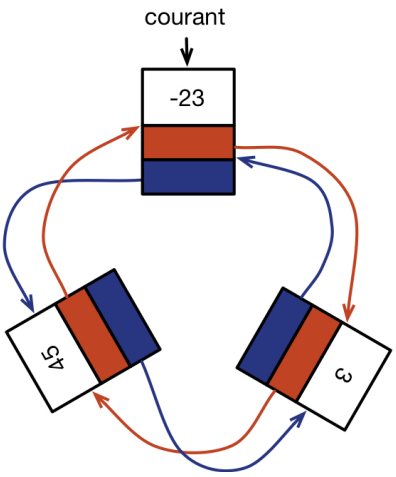
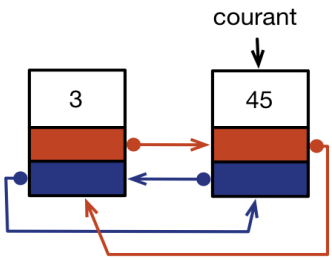


Figure 1

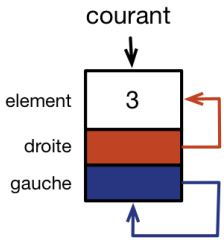
Suite de retraits dans une ronde chainée, en partant de la ronde chainée à l'étape 4 de la figure 1.
Le retrait est toujours celui de l'élément courant.



1. Après le premier retrait



2. Après le deuxième retrait



3. Après le troisième retrait

`courant = null`

4. Après le quatrième retrait

Figure 2

La méthode `toString` suivante vous est fournie et doit être présente dans votre classe `RondeChainee`, car elle sera utilisée pour l'exécution des tests.

Note : une version "texte" de cette méthode (incluant sa Javadoc) se trouve dans le fichier **methodeToString.txt** (pour faciliter le copier-coller).

```
public String toString() {
   MaillonRonde<T> tmp = courant;
   String s = "... ";
   if (courant != null) {
      do {
         s = s + tmp.getElement() + " ";
         tmp = tmp.getDroite();
      } while (tmp != courant);
      s = s + "...";
   }
   return s;
}
```

Les seules méthodes publiques permises dans la classe `RondeChainee` sont :

1) Le constructeur sans argument qui construit une `RondeChainee` vide.
2) La méthode `toString` ci-dessus (fournie)
3) L'implémentation des méthodes de l'interface `Ronde`.

Vous pouvez cependant ajouter des méthodes privées pour bien découper votre code (séparation fonctionnelle). Vous pouvez aussi ajouter des constantes de classe (`static final`) si cela est pertinent, mais rien d'autre.

3. Fichiers fournis

Vous devez utiliser les deux classes et l'interface suivantes, telles quelles, car elles seront utilisées dans les tests de votre code.

1) Classe `ExceptionRondeVide` : classe d'exception utilisée dans l'implémentation de certaines méthodes définies dans `Ronde`.
2) Classe `MaillonRonde` : classe représentant un maillon de la liste chainée utilisée comme structure de données pour l'implémentation de la classe `RondeChainee`.
3) Interface `Ronde` : L'interface qui définit le TDA `Ronde`.
4) Fichier *methodeToString.txt* (contient la méthode `toString` à copier dans votre classe `RondeChainee`)
5) Classe `TestsPartiels` : classe de tests PARTIELS pour tester la classe `RondeChainee`.

IMPORTANT : Les éléments fournis seront utilisés tels quels dans les tests de votre programme. **Il est donc important de ne pas les modifier**.

4. Précisions

- Vous devez respecter le **principe d'encapsulation** des données.
- Vous NE DEVEZ PAS utiliser la classe `ArrayList` (ou tout autre type de collections) ni les tableaux (sauf pour les tableaux `int [] combinaison` donnés comme paramètre de certaines méthodes).
- N'oubliez pas d'écrire la **Javadoc** (pour TOUTES les méthodes).
- **Aucune variable globale** (autre que l'attribut d'instance requis) n'est permise. Vous pouvez cependant ajouter des constantes de classe (`final static`) si vous le jugez pertinent.
- Réutilisez votre code autant que possible. Vous pouvez (et devriez) aussi faire des **méthodes privées** pour bien structurer/découper votre code (séparation fonctionnelle). ATTENTION : toute méthode qui n'est pas dans l'interface `Ronde` (autre que `toString`) doit être privée (`private`).
- Il ne doit y avoir aucun affichage dans vos méthodes (pas de `System.out`)
- Votre classe doit se trouver dans le **paquetage par défaut**.
- **L'utilisation des expressions lambda et des streams (qu'on n'a pas vu(e)s encore) est interdite.**
- Vos fichiers à remettre doivent être encodés en UTF8.
- **Votre code doit compiler et s'exécuter avec le JDK 8**.
- Vous DEVEZ respecter le style Java.
- Vos fichiers à remettre doivent être encodés en UTF8.
- Notez que ce travail ne comporte pas d'application Java, mais vous devrez composer des tests (qui ne sont pas à remettre) pour vous assurer du bon fonctionnement de vos méthodes.

- N'oubliez pas d'écrire (entre autres) votre nom complet et votre code permanent dans l'entête de vos classes.

Le non-respect de toute spécification ou consigne se trouvant dans l'énoncé du TP est susceptible d'engendrer une perte de points.

NOTE : Si quelque chose est ambigu, obscure, s'il manque de l'information, si vous ne comprenez pas les spécifications, si vous avez des doutes… vous avez la responsabilité de vous informer auprès de votre enseignante.

5. Détails sur la correction

5.1 LA QUALITÉ DU CODE (35 POINTS)

Concernant les critères de correction du code, lisez attentivement le document "Critères généraux de correction du code Java". De plus, votre code sera noté sur le respect des conventions de style Java vues en classe (dont un résumé se trouve dans le document "Conventions de style Java"). **Ces deux documents peuvent être téléchargés dans la section TRAVAUX PRATIQUES (ET BOITES DE REMISE) sur la page Moodle du cours**.

Note : Votre code sera corrigé sur la totalité ou une partie des critères de correction indiqués ci-dessus, ainsi que sur le respect des spécifications / consignes mentionnées dans ce document.

5.2 L'EXÉCUTION (65 POINTS)

Un travail qui ne compile pas se verra attribuer la note 0 pour l'exécution.

Votre code sera testé en tout ou en partie. Les points seront calculés sur les tests effectués. Ceci signifie que si votre code fonctionne dans un cas x et que ce cas x n'est pas testé, vous n'obtiendrez pas de points pour ce cas. Il est donc dans votre intérêt de vous assurer du bon fonctionnement de votre programme dans tous les cas possible.

Une classe de tests PARTIELS vous est fournie (servez-vous-en), mais vous devrez aussi composer vos propres tests pour tester correctement (complètement) votre classe. Les tests qui seront utilisés pour la correction seront différents de ceux fournis.

6. Date et modalités de remise

6.1 REMISE

Date de remise : Au plus tard le **8 juillet 2022** à 23h59.

Le fichier à remettre : `RondeChainee.java` (**PAS** dans une archive zip, rar, etc.)

Remise via Moodle uniquement.

Vous devez remettre (téléverser) votre fichier sur le site du cours (Moodle). Vous trouverez la boîte de remise dans la section **TRAVAUX PRATIQUES (ET BOITES DE REMISE)**.

VÉRIFIEZ BIEN QUE VOUS AVEZ REMIS LE BON FICHIER SUR MOODLE (le .java et non le .class, par exemple).

6.2 POLITIQUE CONCERNANT LES RETARDS

Une pénalité de 10% de la note finale, par jour de retard, sera appliquée aux travaux remis après la date limite. La formule suivante sera utilisée pour calculer la pénalité pour les retards : `Nbr points de pénalité = m / 144,` où `m` est le nombre de minutes de retard par rapport à l'heure de remise. Ceci donne 10 points de pénalité pour 24 heures de retard, 1.25 point de pénalité pour 3 heures, etc.

Aucun travail ne sera accepté après 1 jour (24 h) de retard, et la note attribuée sera 0.

6.3 REMARQUES GÉNÉRALES

- Aucun fichier reçu par courriel ne sera accepté. **En d'autres termes, un travail reçu par courriel sera considéré comme non remis.**
- Vous avez la responsabilité de conserver des copies de sauvegarde de votre travail (sur disque externe, Moodle, Dropbox, Google Drive, One Drive, etc.). La perte d'un travail due à un vol, un accident, un bris… n'est pas une raison valable pour obtenir une extension pour la remise de votre travail.
- **N'oubliez pas d'écrire (entre autres) votre nom complet, et votre code permanent dans l'entête de la classe à remettre.**
- **N'attendez pas à la dernière minute pour commencer le travail, vous allez fort probablement rencontrer des problèmes inattendus!**

Ce 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 rapidement toutes vos impressions de programme au laboratoire.

BON TRAVAIL !