Cours 420-306-LI **POO III informatique** **Automne 2015** **Cégep Limoilou** **Département d'informatique** **Professeur :** **Martin Simoneau** **Imed Jaras** **Doudou Camara**	**Laboratoire1** **Collection JDK8** **10% de la session**

Nom : _____.

Consignes :

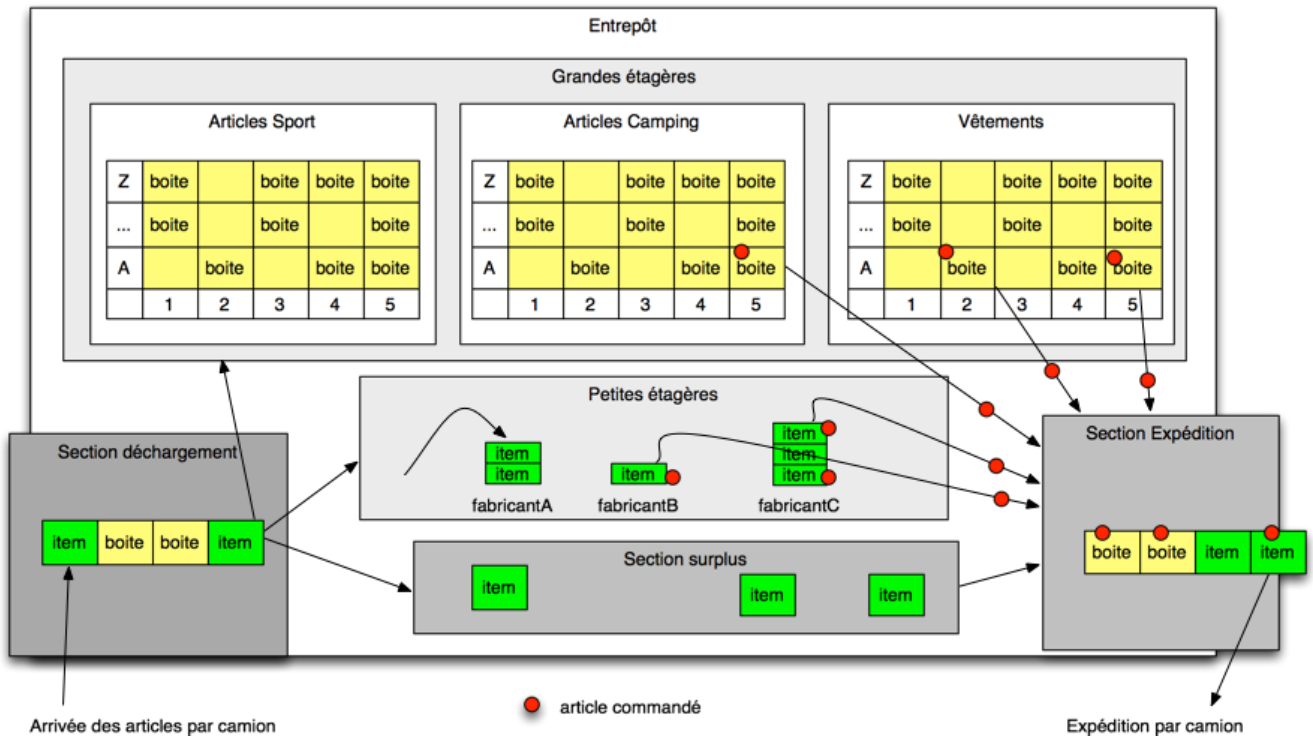
- À faire en équipe de 2
- Remettre à la date indiqué sur LÉA
- Remettre le projet Maven au complet avec le document word.

À remettre :
5% sont accordé pour les exercices 1 et 2

Contexte:
Nous allons programmer un entrepôt d'articles divers. L'entrepôt contient des items divers et des boîtes d'items. Il y a 3 types d'items : ARTICLE_SPORT, ARTICLE_CAMPING et VETEMENT.

- L'entrepôt est principalement constitué de 2 grandes pièces;
  - l'une contenant de grandes étagères destinées à recevoir les boîtes d'articles pour le transport. Cette section est constituée de la manière suivante :
    - Une grande étagère pour chacun des 3 types d'items
    - Chacune des 3 grandes étagères contient plusieurs sous-sections. Chaque sous-section est identifiée avec une lettre de 'A' à Z' (26 sous-sections). On place dans la sous-section les boîtes dont le *nom de produit* de la boîte commence par la lettre identifiant la sous-section.
    - Les boîtes sont placées les unes à côté des autres et elles sont toutes facilement accessibles. Le nombre de boîte n'est pas limité (on suppose une très grande étagère).
    - Les grandes étagères ne contiennent que les boîtes de transport.
  - L'autre pièce contient des items seuls dans des petites étagères.
    - On identifie sur chaque section d'une petite étagère le *nom du fabriquant* dont provient les items. Une section contient donc uniquement des items provenant d'un même fabricant (classement par fabricant). Les items
    - Dans chaque sous-section de fabricant, les items sont empilés les uns par-dessus les autres, la dernière entrée sur le dessus. Une pile peut contenir un nombre d'items inférieur à CAPACITE_MAX_SECTION_PETITE_ETAGERE (configurable).
- En plus des 2 grandes pièces, l'entrepôt contient 3 pièces plus petites.
  - Une section de déchargement où les items (boîte et item seul) sont placés sur un convoyeur qui les transporte vers les 2 grandes pièces. Ils seront récupérés dans l'ordre d'arrivée et placés au bon endroit sur les étagères.
  - Une section de surplus. Lorsqu'une section de la petite étagère est pleine, l'item est automatiquement placé dans le surplus.
  - Une section d'expédition dans laquelle sont placés tous les items qui ont été commandés.

Sommaire du fonctionnement :



1. Des **items** et **boîtes** sont ajoutés dans la **section déchargement** de l'entrepôt (méthode ajouteItems())
2. Toutes les boîtes de la section déchargement sont ensuite transférées vers les **grandes** étagères et les items vers la section des **petites étagères** selon les critères énoncés plus tôt. (méthode traiteEntrepot)
3. À L'aide d'une fonction de **recherche,** on peut obtenir toutes les boîtes et tous les items de l'entrepôt. La fonction de recherche sera donc utilisée pour sélectionner les éléments qui seront commandés. On commande un élément en lui ajoutant un objet commande qui contient la date de la commande.
4. Tous les items marqués pour la commande sont **transférés** dans la section expédition
5. La section expédition est vidée. Chaque item est écrit dans un fichier de sérialisation pour constituer un **historique** de tout ce qui est sorti de l'entrepôt.

Partie 1 – *Collection*

Récupérez l'archive qui vous a été transmise avec cet énoncé, vous y trouverez le projet « Entrepot.depart ». Dans le fichier « pom.xml » vous devez changer 2 balises. Inscrivez vos initiales où c'est demandé. Ne changer rien d'autre.

<groupId>climoilou.**vos_initiales**</groupId>

<artifactId>Entrepot.depart.**vos_initiales**</artifactId>

Pour inclure le projet dans votre espace de travail eclipse, utilisez *import Maven project au lieu de import existing project,*

La première tâche consiste à donner le meilleur type possible aux attributs qui représentent les différentes sections de l'entrepôt

Le programme entrepôt est principalement réalisé dans la classe *Entrepot*. Vous devez programmer les fonctionnalités suivantes. ATTENTION vous devez également programmez pour chaque méthode au moins un test unitaire avec Junit, afin de montrer que la méthode réalise bien ce qui est demandé (dans EntrepotTest.java).

- Constructeur(moyen)

  Vous devez construire les objets requis. Attention les sous-sections des fabricants des petites étagères doivent être construites au besoin (lazy).

- commandeItem() (court)

  On marque tous les items reçus comment étant commandé en leur attribuant une commande en date du moment où la méthode est appelée. On peut donc prendre tenir pour acquis qu'un item qui a l'attribut *commande* égale à null n'a pas été commandé.

- prepareExpedition(long)

  1. On prend toutes les boîtes de la grande étagère qui sont commandées, et on les transfère dans la section expédition. On doit alors mettre la date d'expédition dans la commande.
  2. On transfère également tous les items de la petite étagère qui ont été marqués pour la commande dans la section expédition.

  CONTRAINTE: ici vous ne devez utiliser que les méthode pop(), push() et peek() pour jouer avec la Stack.

- videDechargement (moyen)

  Extrait tous les articles (boîtes et items seuls) de la section déchargement. S'il s'agit d'une boîte, elle est ajoutée dans les grandes étagères, s'il s'agit d'un item il va dans les petites étagères.

  Si la section de la petite étagère est pleine, l'item ira dans la section surplus.

- getToutItemDansEntrepot (moyen)

  Sort tous les items et boîtes qui sont dans toutes les étagères et dans la section de déchargement. (ne pas mettre les éléments de la section expédition). Cette méthode sera utilisée pour la commande.

Partie 2 – *Serialisation*

L'entrepôt note tous les items et boîtes qui sont expédiés dans des fichiers de sérialisation. Il doit y avoir un fichier par appel à la méthode déchargement. Le nom des fichiers est donné dans les constantes. À chaque nouveau fichier le numéro est incrémenté

Exemple: *expedition0.ser, expedition1.ser, expedition2.ser*…

- String expedie()

  prendre chaque item ou boîte dans la section expédition et l'ajouter dans le fichier de sérialisation. Retourne le nom du fichier utiliser à l'appelant. N'oubliez pas de retirer les éléments de la section expédition au fur et à mesure qu'ils sont notés dans le fichier de sérialisation.

- unserializeHistory(String)

lire tous les items et boîtes dans le fichier de sérialisation correspondant au nom reçu en paramètre. Vous devez lire tous les items… comment savoir quand vous arrêter? Rappelez-vous l'EOFException.

Partie 3 – *Moteur de recherche*

Faites diagramme de classe UML du package Item (mettre toutes les classes, relations et multiplicités)

Faites diagramme de classe UML du package recherche. (mettre toutes les classes, relations et multiplicités)

Mettre votre projet visual paradigm dans le dossier *modele* que vous placerez à la racine du projet. Remettre un document word contenant vos 2 diagrammes de classes ainsi qu'un cours texte expliquant le fonctionnement du moteur de recherche ().

Programmation :

Dans le package moteur de programmation vous devez complétez les portions de code indiquées avec le commentaire //*TODO* (voir la vue task) afin que les tests unitaires fonctionnent.

Dans le document Word, dites si la solution avec les *MemeIDCritereCollection* est plus efficace que *MemeIDCritere* et expliquez pourquoi.