Cours 420-4A6-LI Applications monopostes I Hiver 2020 Cégep Limoilou Département d'informatique	Formatif 3 Structures de données Java (suite) Intro JavaFX

Nom : _____ Gr : _____

Objectif

- Mieux comprendre la structure interne des structures de données.
- Comprendre le principe de classes générique.
- Comprendre l'importance de bien définir « equals() » et « hashCode » lors de l'utilisation des structures de données.
- S'introduire au GUI avec JavaFX

À remettre

- Remettre votre projet Eclipse zippé à votre nom, formatif 3 sur le réseau dans le dépôt « Q:\!Dépôt\JGoulet\420-4A6-H20\Formatif 3 - Code ».

Fonctionnement

- L'exercice peut se faire individuellement ou en équipe.

Avant de commencer

- Créez un nouveau projet ***Formatis 3*** dans Eclispe.

Cas 1 – Une structure de données à base de nœuds chaînés

En vous basant sur les explications dans les notes de cours et le code source des classes pile et file fournies, on vous demande d'implémenter une nouvelle structure de données, le « buffer circulaire ».

Un « buffer circulaire » est une structure de données utilisant un « buffer » de taille fixe et dont le début et la fin sont considérés comme connectés. Les buffers circulaires sont souvent utilisés pour gérer des flux de données ou pour implémenter un comportement de type FIFO.

- Prenez connaissance de son fonctionnement à partir des liens suivants :
  - https://fr.wikipedia.org/wiki/Buffer_circulaire
  - Implémentation statique : http://www.java2s.com/Tutorial/Java/0140__Collections/CircularBuffer.htm
  - Autres explications et implémentation : http://www.mathcs.emory.edu/~cheung/Courses/171/Syllabus/8-List/array-queue2.html
- Créez le package ***cas1***.
- Dans le package ***cas1***, créez la classe ***BufferCirculaire***.
- Votre travail est de réaliser dans la classe ***BufferCirculaire*** une implémentation de cette structure de données en utilisant votre propre structure de nœuds chaînés (code exemple pile et file), plutôt qu'un tableau statique pour contenir les éléments de la structure.
  - La structure a une capacité finie, déterminée lors de sa construction.
  - Toutes les méthodes d'utilisation de base (minimales selon la littérature) doivent être développées.
  - À vous de choisir comment gérer les erreurs (vide, pleine …).
  - La classe du nœud « chaînable » doit être interne à la classe de la structure.
  - Des tests unitaires devraient être faits.
- Faites une classe ***MainBufferCirculaire*** comme preuve de concept de votre nouvelle structure.

Cas 2 – Une structure générique

- Créez le package ***cas2***.
- Dans le package ***cas2***, copiez la classe ***BufferCirculaire*** du précédent cas et renommez la ***GenBufferCirculaire***.
- Votre travail est maintenant de rendre cette classe utilisable de façon générique en ayant les mêmes caractéristiques que le cas 1.
    - Des tests unitaires devraient être faits.
- Faites une classe ***MainGenBufferCirculaire***, utilisant quelques types différents d'objets, comme preuve de concept de votre nouvelle structure générique.

Cas 3 – Classe générique pour faire de la validation de contraintes

On veut faire une classe générique qui va valider les contraintes de définition des méthodes « equals » et « hashCode ». Ces contraintes sont la symétrie, la réflexivité, la transitivité, la consistance entre « hashCode » et « equals » et l'égalité avec la valeur « null ».

On doit aussi ajouter à cette liste de contraintes, les classes qui implémentent l'interface « Comparable ». Elles doivent maintenir une cohérence entre les méthodes « equals() »-« hashCode() » et aussi avec la méthode « **compareTo**() ».

- Prenez connaissance de la théorie à ce sujet à partir des liens suivant :
    - Pour bien comprendre ce qu'impliquent ces contraintes : https://www.jmdoudoux.fr/java/dej/chap-techniques_java.htm#techniques_java-2
    - Autres explications de leur importance : https://www.infoq.com/fr/articles/retour-sur-les-bases-equals-et-hashcode/
- Créez le package ***cas3***.
- Dans le package ***cas3***, créez la classe ***ValiderContEqualsHashCode***.
- La classe générique ainsi produite doit permettre de démontrer que les objets d'une classe quelconque respectent les contraintes d'implémentation des méthodes « equals », « hashCode » et si présente « compareTo ».
    - Affiche pour chaque contrainte un message de *réussite* ou *d'échec*.
    - Pourrait utiliser une liste d'objets à l'entrée.
    - Des tests unitaires devraient être faits.
- Faites une classe ***MainValiderContEqualsHashCode***, utilisant quelques types différents d'objets, comme preuve de concept de votre nouvelle structure générique.

Cas 4 – Utiliser une Map

- Copiez le dossier du package ***cas4,*** fourni sur le réseau avec le formatif, comme nouveau package dans votre projet.
- Modifiez la méthode *produirePermis()* de la classe *MoulinettesCSV* pour que la liste de permis soit une map de permis *Map<String. Permis>* ou la clé est le numéro de permis.
- Modifiez la méthode *main()* de la classe *MoulinettesCSV* pour parcourir la map avec un objet *Map.Entry*.
- Faites les ajustements nécessaires pour que le programme fonctionne toujours correctement.
- Observez si le nombre de lignes dans le fichier « *permis-animaux.csv* » fourni avec le package ***cas4*** (cas4.file) correspond bien avec le nombre d'enregistrements dans la structure de la map. Expliquez vos observations.

Cas 5 – Intro à JavaFX

Avant de vous présenter les premiers concepts au sujet de JavaFx, je vais vous demander de lire les documents suivants qui sont disponibles dans le dossier de cours de la semaine 3.

- S03 - Diapo - ihm1_fx_01 (Introduction aux concepts de base).pdf
- S03 - Diapo - ihm1_fx_03 (Architecture et Concepts techniques).pdf

Copiez et faites fonctionner du code JavaFX.

- Créez le package ***cas5***.
- Copiez et faites fonctionner le code de la classe exemple « Bienvenue », disponible dans vos lectures.