INF2010 - Structures de données et algorithmes
TP01 : Java

Hiver 2019

Objectif du TP: est d'initier l'étudiant aux notions de base de la programation orientée objet (POO) et de pratiquer les notions d'héritage (mots clés **super**, **extends**) et de polymorphisme en Java. Un second volet traite des structure de données séquentielles.

1 Première partie : POO

1. **Classes et objets :**
   Nous désirons écrire un programme permettant de manipuler une collection d'objets et les afficher de manière triée selon différents critères.

   Les objets manipulés sont des étudiants. Un étudiant sera la donnée d'un prénom, d'un nom, d'un matricule et d'une section. Une classe `Etudiant` vous est fournie dans le fichier *Etudiant.java* et il vous revient de compléter certaines de ses méthodes. Fiez-vous aux commentaires pour ce faire.

   La fonction principale (`main`) est fournie dans la classe `Trieuse` du fichier *Trieuse.java*. Il vous revient de compléter la classe Trieuse pour être en mesure d'effectuer les tris désirés. Pour ce faire, vous devez utiliser l'interface `Comparator` et la classe `Arrays` définies dans la bibliothèque `java.util.*`.

2. **Héritage et polymorphisme :**
   La librairie coopérative de l'École Polytechnique (*coopoly*) se spécialise dans la vente de divers produits : livres de génie, de littérature générale, cadre pour diplôme, vêtements, etc. Chaque article possède trois attributs : référence, nom du produit et son prix hors taxes.

On désire prototyper un programme permettant de gérer la vente des produits. On crée une classe produit `Article` dont héritent divers autres produits (`CadreDiplome`, `Livre`, `Vetement`). Un programme prototype est réalisé dans le fichier *Test.java*. Compléter les classes fournies de sorte 'a ce que le programme prototype fonctionne comme attendu.

Travail :

1. Complétez les classes *Etudiant.java* et *Trieuse.java*
2. Complétez les classes *Article.java Livre.java Vetement.java CadreDiplome.java*

2 Seconde partie : Types de données abstraits (ADT)

2.1 Indication

— **Types de données :** Un type de données est un ensemble de valeurs et un ensemble d'opérations sur ces valeurs.
— **Types de données abstraits :** Un type de données abstrait est un type de données dont la représentation interne est masquée au utilisateur finale.

Nous allons définir un type de données modifiables abstrait pour un compteur entier. La partie teste dans la méthode main créent n compteurs et effectuent de manière aléatoire une incrémentation du nombre d'essais. Autrement dit, faire des incrémentations sur des compteurs aléatoires.
Aprés on se propose de simuler le tirage d'une pièce de monnaie équitable n fois à l'aide d'un compteur entier.

2.2 Questions :

1. Complétez le code de *Compteur.java* et *Simulation.java*

3 Troisième partie : Structure de données séquentielles : Sacs, Piles et Files

3.1 Indication

— **MultiEnsemble (Sac) :** est une collection pour laquelle la suppression d'éléments n'est pas prise en charge. Son objectif est de donner la possibilité de collecter des éléments, puis de les parcourir et son implémentation est effectuée à l'aide d'une liste chaînée .

Plusieurs types de données fondamentales impliquent des collections d'objets. Plus précisément, l'ensemble de valeurs est une collection d'objets et les opérations tournent autour de l'ajout, de la suppression ou de la modification d'objets dans la collection. Dans cette partie, nous examinons trois types de structure de données séquentielles, appelés Multiensemble (sac), file et pile. Ils diffèrent par la spécification de l'objet à ajouter ou à supprimer ou à modifier ensuite.

— **Sac.java :** représente d'éléments génériques (*items*). Il prend en charge l'insertion et l'itération sur les éléments dans un ordre arbitraire. Cette implémentation utilise une liste chaînée avec une classe imbriquée statique nœud. Les opérations d'ajout et de test si le multiensemble est vide prendre du temps constant. L'itération prend un temps proportionnellement au nombre d'éléments.
— **File.java :** La classe file représente une file d'attente FIFO (premier entré, premier sorti) d'éléments génériques (*item*). Il prend en charge les opérations habituelles (*enfiler*) et (*defiler*), ainsi que des méthodes permettant d'examiner le premier élément , de vérifier si la file d'attente est vide (*isempty*) et d'effectuer une itération sur les éléments dans l'ordre FIFO.
— **Pile.java :** La classe Pile représente une pile d'éléments génériques (*item*) sur le principe "dernier entré, premier sorti".Il prend en charge les opérations (*empiler*) (*dépiler*), ainsi que des méthodes permettant de renvoyer l'élément de tête sans le dépiler, de tester si la pile est vide et d'effectuer une itération sur les éléments dans l'ordre LIFO.

3.2 Questions

1. Réalisez les classes *Sac.java* , *File.java* , *Pile.java* et *Main.java*

4 Instructions pour la remise

Le travail doit être remis via Moodle :
— 05 Février avant 23h55 pour le groupe 01
— 29 Janvier avant 23h55 pour le groupe 02

Veuillez envoyer dans une archive de type $*$.zip qui portera le nom inf2010_lab1_MatriculeX_ MatriculeY (de sorte que MatriculeX $\prec$ MatriculeY) :
— Vos fichiers .java
Les travaux en retard seront pénalisés de 20 % par jour de retard. Aucun travail ne sera accepté après 4 jours de retard.

5 Barème de correction

6 pts : Première partie : Programmation orientée objet (POO)

6 pts : Seconde partie : Types de données abstraits (ADT)

8 pts : Troisième partie : Structure de données séquentielles : Listes, piles et files