INF2010 – Structures de données et algorithmes
Automne 2017
Travail Pratique 2
Structures de données séquentielles

Objectifs :

- Implémenter une liste doublement chaînée.
- Implémenter une pile à partir d'un tableau.
- Implémenter un algorithme de résolution d'expressions postfixes à l'aide d'une pile.
- Implémenter une file à l'aide de piles.

Problème 1: Liste doublement chaînée (1.5 points)

Vous devez compléter l'implémentation d'une liste doublement chaînée, c'est-à-dire, une séquence de nœuds chaînés possédant chacun une référence vers le nœud précédent ET le nœud suivant. Une telle liste peut d'ailleurs être utilisée afin d'implémenter toutes deux une file et une pile.

Vous trouverez dans le fichier `DoublyLinkedList.java` la description des méthodes à compléter. Le fichier `ListMain.java` vous permettra de tester cette classe.

Problème 2: Pile (2.25 points)

Une pile est un conteneur qui implémente le protocole dernier entré, premier sorti (LIFO, last in, first-out). C'est l'une des structures de données les plus utilisées en informatique, notamment pour la récursivité, le traitement des expressions arithmétiques et les parcours d'arbres et de graphes.

1) Création de la pile (0.75)

Vous devez ici compléter l'implémentation d'une pile. La pile doit disposer des fonctions suivantes :

- Pop : enlève un élément du sommet de la pile et le renvoi.
- Push : ajoute un élément au sommet de la pile.
- Peek : retourne l'élément au sommet de la pile, sans l'enlever.
- Size : retourne le nombre d'éléments présents dans la pile.
- Empty : indique si la pile est vide.

Vous devez implémenter cette structure à l'aide d'un tableau. Vous trouverez la description des méthodes à implémenter dans le fichier `ArrayStack.java`. Le fichier `StackMain.java` vous permettra de tester votre classe.

2) Test de la pile (0.25)

Écrivez une fonction prenant en argument une chaîne de caractères contenant des espaces, par exemple une phrase, et renvoie la phrase avec les mots inversés. L'entrée « Ceci est un exemple » renvoie « exemple un est Ceci ». Utilisez pour cela la pile que vous avez implémentée. Un squelette de fonction est fourni dans le fichier `StackMain.java`.

3) Solveur d'équations booléennes (1.25)

Vous avez vu dans le cours une méthode utilisant la pile pour implémenter un solveur d'équations postfixes. Dans cet exercice, nous allons implémenter de manière similaire un solveur d'équations booléennes.

Votre solveur doit prendre en entrée une chaîne de caractères représentant une série d'opérations en notation postfixe et retourner le résultat, 0 ou 1. Par exemple, l'entrée « 0 1 or 1 not and » en notation postfixe correspond à « (0 or 1) and (not 1) ».

Les opérations que votre solveur doit reconnaître sont *or*, *and* et *not*. Implantez votre solveur dans la classe `PostfixSolverMain.java`.

Problème 3: File à l'aide de piles (1.25 points)

Vous devez compléter l'implémentation d'une file. Pour rappel, une file est une structure FIFO (first-in-first-out). Un élément peut donc seulement être ajouté à la fin de la file et seul l'élément en tête de file peut être retiré.

Toutefois, de manière à tester votre compréhension du fonctionnement des structures de données FIFO et FILO, **il est demandé d'implémenter la file à l'aide de deux piles**.

***Notez qu'une telle implémentation d'une file n'est pas efficace. Il s'agit ici d'un exercice servant à s'assurer que vous comprenez bien le comportement des files et des piles.

Vous trouverez dans le fichier `StackQueue.java` la description des méthodes à compléter. Le fichier `QueueMain.java` vous permettra de tester cette classe.

Instructions pour la remise

Le travail doit être fait par équipe de 2 personnes et doit être remis via Moodle au plus tard le :

- 2 octobre avant 23h59 pour le groupe 4, soit Mardi (B2)
- 15 octobre avant 23h59 pour le groupe 2, soit Lundi (B2)
- 16 octobre avant 23h59 pour le groupe 3, soit Mardi (B1)
- 22 octobre avant 23h59 pour le groupe 1, soit Lundi (B1)

Veuillez envoyer vos fichiers .java **seulement**, dans un **seul répertoire**, le tout dans une archive de type *.**zip** (et seulement **zip**, pas de **rar**, **7z**, etc.) qui portera le nom : **inf2010_lab2_MatriculeX_MatriculeY.zip**, où MatriculeX < MatriculeY.

Les travaux en retard seront pénalisés de 20 % par jour de retard. Aucun travail ne sera accepté après 4 jours de retard. Si votre dépôt ne respecte pas la nomenclature définie ci-dessus, 0.5 point de pénalité sera appliqué.