**INF3430 – Méthodes de test et de validation du logiciel**

TP No. 1

Groupe 1 :

1437737 – Maxime Jacob

1539437 – Dan Vatnik

1776903 – Cédrick Busque

Présenté à :

Mlouki, Ons

Polytechnique Montréal

30 janvier 2017

1. **Guide de l’utilisateur**

L’utilisation du programme est très simple. Afin de construire une SuiteP, il suffit d’insérer son constructeur dans la fonction Main() du programme. Pour que l’exécution se fasse correctement, il faudra entrer les paramètres sous la forme suivante :

*SuiteP(opération, entier pair, entier impair, taille maximale de la pile);*

Une fois les paramètres entrés, la pile se construira automatiquement avec les paramètres demandés, et fera l’affichage en console de l’état de la pile après la construction.

**Valeurs acceptées des paramètres :**

-Opération : “addition”, “soustraction”, “multiplication”, “division”

-Entier pair : tout entier pair compris entre -2147483648 et 2147483647

-Entier impaire : tout entier pair compris entre -2147483648 et 2147483647

-Taille maximal de la pile : tout entier compris entre 2 et 10.

**En cas de débordement :**

En cas de débordement de la valeur maximale d’un entier, le programme interrompra prématurément la construction de la pile. Il en va de même pour une situation de division par zéro.

**2) Manuel de conception**

**a) Interface Pile**

L’interface Pile pose les bases de la gestion d’une pile.

**public Object empiler(Object item):**

**public Object dépiler():**

**public Object getHead();**

**public boolean estVide():**

**public int getSize():**

**public void reset():**

**public void print() :**

**a.1) public class Nœud :**

Classe décrivant un élément contenu dans un pile.

**public Noeud(Object val) :**

Construit un nœud contenant la valeur passée en val.

**public Object getVal() :**

Retourne la valeur contenue dans le nœud.

**public void setVal(Object item) :**

Insère une nouvelle valeur dans le nœud.

**public void setNext(Noeud proch) :**

Fixe le nœud suivant parent.

**public Noeud getNext() :**

Retourne le nœud parent

**b) Classe SuiteP**

**public suiteP(String op, int valP, int valI, int tailleMax) :**

Permet de construire une pile avec les paramètres passés. L’opération choisie sera toujours effectuée entre les deux entiers au sommets de la pile. String op représente l’opération à faire sur la pile, valP un entier pair, valI un entier impair, et tailleMax la taille finale de la pile après les opérations.

**public Object empiler(Object item):**

Ajout d’un élément a la pile.

**public Object dépiler():**

Retrait d’un élément de la pile

**public Object getHead();**

Retourne la tête de la pile.

**public boolean estVide():**

Retourne un booléen indiquant si la pile est vide ou pas.

**public int getSize():**

Retourne la taille de la pile

**public void reset():**

Remet la pile à vide.

**public void print() :**

Affiche le contenu de la pile à l’écran, en commençant par le sommet de la pile.

**c) Classe IOpt**

Interface pour les opérations à faire sur des nœuds.

**public int add(int x, int y);**

**public int sub(int x, int y);**

**public int mult(int x, int y);**

**public int div(int x, int y);**

**d) Classe Opt**

Classe décrivant les opérations standards à faire sur notre SuiteP

**public int add(int x, int y) :**

Additionne la valeur x à la valeur y et retourne le résultat.

**public int sub(int x, int y) :**

Soustrait la valeur y à la valeur x et retourne le résultat.

**public int mult(int x, int y) :**

Multiplie la valeur x et y et retourne le résultat.

**public int div(int x, int y) :**

Divise la valeur x par y et retourne le résultat.

**e) Classe Application**

Point d’entrée de notre programme. Contient le main qui peut être édité afin d’accomplir les opérations voulues.

**public static void main(String[] args) :**

Fonction principale du programme et point d’entrée.