Universidad del Valle de Guatemala
Facultad de ingeniería
Departamento de Ciencias de la Computación
CC2003-Algoritmos y Estructuras de Datos

Alejandro Gómez 20347
Alejandra Guzmán 20262

Hoja de trabajo No. 4

El programa a realizar para la hoja de trabajo no.4 consiste en la implementación de distintos conceptos, como lo es: la utilización de genéricos, el diseño del ADT para pilas y listas. Para la realización de esta hoja de trabajo se realizó con anterioridad la lectura del capítulo no.9 del libro brindado en este curso respecto al tema de implementaciones de listas. Así como también, se leyó y analizó los conceptos mencionados en el libro de Patrones de Diseño para la correcta implementación de Factory y Singlenton.

Esta hoja se basa en la implementación de distintos patrones de diseño, funcionando de tal forma que el usuario que utilice el programa decida cual implementación quiere usar para el stack.

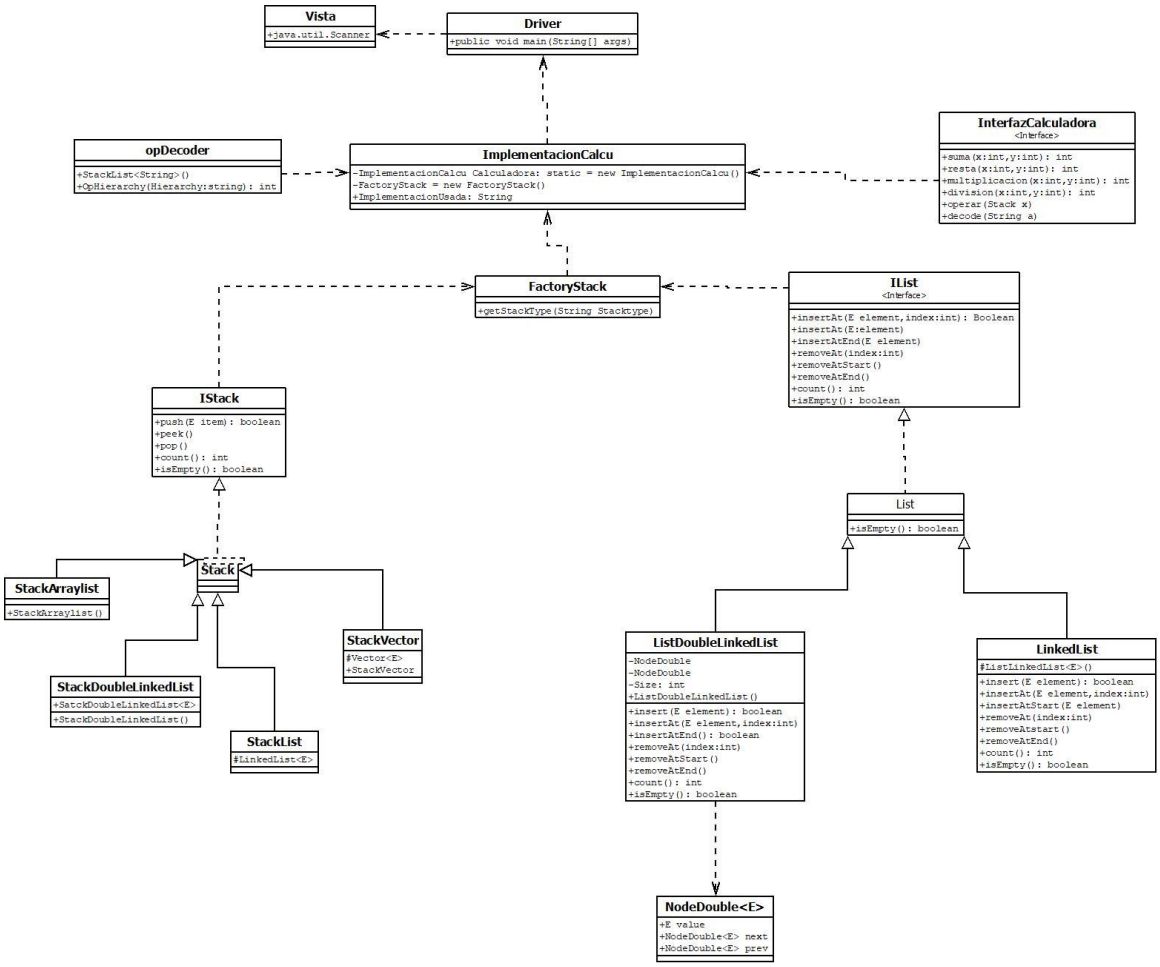
1) Respuesta:
    a) Ventajas y desventajas del patrón Singleton en general:
        i) Ventajas de usar patrón Singleton:
            (1) Es bastante útil debido a que le brinda y asegura que solamente exista una instancia de un objeto.
            (2) Así mismo, por lo anterior mencionado, el objeto sólo se inicializa cuando se accede la primera vez al mismo.

        ii) Desventajas de usar patrón Singleton:
            (1) De conformidad con los principios aprendidos durante el curso de POO, se puede observar que viola los principios de responsabilidad única, o por sus siglas en inglés *SOLID*.
            (2) Debido a la forma de uso del mismo, puede llegar a dificultar el testeo del programa, además que con este patrón es fácil ocultar un diseño mal hecho.

b) ¿El uso del patrón Singlenton es adecuado en este programa?

Al analizar el patrón Singlenton y su posible implementación en el programa se deben tomar en cuenta distintos aspectos. Como ya se sabe, en este patrón de diseño se puede instanciar una clase operaciones. Haciendo que estas se conformen de los objetos que necesite como las operaciones. (ej: suma, resta, multiplicación, etc.) Se pueden aislar los diferentes tipos de objetos, los cuales pueden administrar y ser utiles en la realización de este programa

2) Diagrama UML de clases:



3) GitHub:

a) Link del repositorio en GitHub: https://github.com/aleg001/HDT4

4) Evidencia del correcto funcionamiento del programa:
    a) Fotos:
        i) Al ingresar un archivo correcto:

```
Bienvenido a la calculadora Infix-Postfix

1. Usar calculadora
2. Salir

Ingrese su opcion:
1
Leyendo el archivo... datos.txt

Cual implementacion de Stack usara?
1. Vector
2. ArrayList
3. List
4. Salir
```

        ii) Al ingresar un archivo que no existe:

```
Ingrese su opcion:
2
ERROR: Archivo no se encontro
```

Bibliografía:

https://halfstuck.com/2020/09/patron-singleton/

https://devexperto.com/principio-responsabilidad-unica/

https://es.slideshare.net/poloparedesrodriguez/patron-creacional-singleton

Referencia:

Infix to Postfix algorithm using Stack

Inputs

1. The infix expresión
   a. 1+2*9
   b. (1+2)*9

Output

1. The postfix expresión
   a. 1 2 9 * +
   b. 1 2 + 9 *

Algorithm

```
1. Begin
2.    initially push some special character say # into the stack
3.    for each character ch from infix expression, do
4.       if ch is alphanumeric character, then
5.          add ch to postfix expresión (queue or list)
6.       else if ch = opening parenthesis (, then
7.          push ( into stack
8.       else if ch = ^, then //any operator of higher precedence
9.          push ^ into the stack
10.      else if ch = closing parenthesis ), then
11.         while stack is not empty and stack top ≠ (, do
12.            pop and add item from stack to postfix expression
13.         done
14.
15.          pop ( also from the stack
16.       else
17.               while stack is not empty AND precedence of ch <=
   precedence of stack top element, do
18.                  pop and add into postfix expression
19.               done
20.
21.               push the newly coming character.
22.         done
23.
24.         while the stack contains some remaining characters, do
25.            pop and add to the postfix expression
26.         done
27.         return postfix
28.      End
```