Tarea 1 de Estructuras de Datos 2

Marvin Calvo Acuña

mcalvoa@ucenfotec.ac.cr

**RESUMEN:** *La tarea se realizó bajo el lenguaje de programación C#, se implementó las estructuras de datos pila, cola y lista, cada una con sus respectivos funcionamientos. Se desarrolló bajo el patrón decorador, de tal manera usando el polimorfismo y la herencia, pero respetando la abstracción de cada estructura. Se obtuvieron estructuras capaces de cambiar de forma y de poder realizar otras implementaciones.*

**PALABRAS CLAVE**: Cola, estructura de datos, Lista, pila, polimorfismo.

# INTRODUCCIÓN

Este documento describe la implementación en software con respecto a la tarea 1 de la carrera de Ingeniería en Software y el curso Estructura de Datos 2.

Además se describirá cada método utilizado por cada clase que lo implementa, así como los parámetros que reciben y retornan siendo el caso.

# OBJETIVOS

## OBJETIVO GENERAL

Implementar y recordar las estructuras de datos pila, cola y lista, así como su uso.

## OBJETIVO ESPECIFICOS

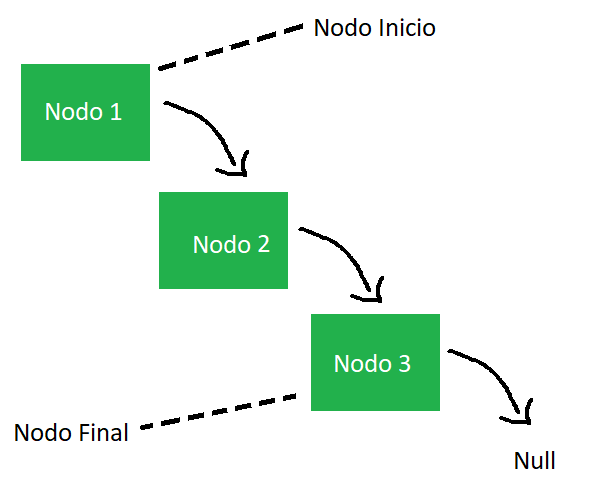
Programar las estructuras pila, cola y lista.

Codificación de los métodos respectivos de cada estructura de base de datos.

# LISTA

## REPRESION ABSTRACTA

<nodo1, nodo2,…, nodo>

, donde nodo sea diferente de nulo.

Los nodos referencian al siguiente y así sucesivamente hasta llegar al nodo que referencia a nada o a nulo.

## INVARIANTE

Todos los nodos son los elementos de la lista y cada nodo puede o no referenciar al siguiente nodo.

## OPERACIONES

InsertAlInicio(Nodo)->Nada (Modificadora)//Inserta al inicio de la lista

InsertAlInicio(int)->Nada (Modificadora)//Inserta al inicio de la lista

InsertOrdenado(int)->Nada (Modificadora)//Inserta ordenado.

InsertAlFinal (int)->Nada (Modificadora)//Inserta al final de la lista

## MANEJO DE ERRORES

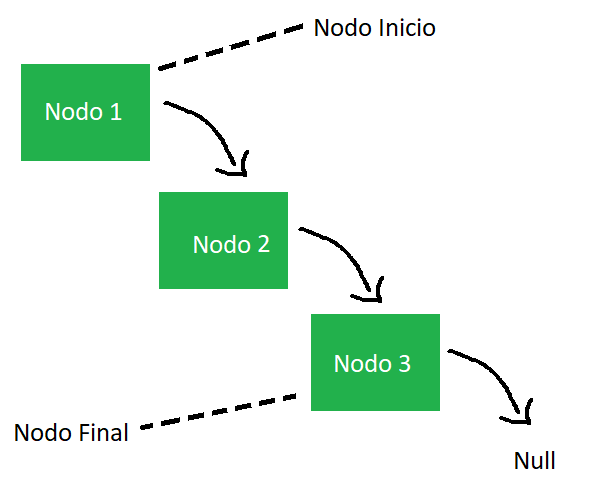
Todas las operaciones de las listas deben de mantener invariante del TAD.

Cuando una operación indebida ocurra el programa debe abortar su ejecución o informar al usuario por medio de audio o un cuadro de texto.

# PILA

## REPRESENTACION ABSTRACTA

<nodo1, nodo2,…, nodo>

, donde nodo sea diferente de nulo.

Los nodos referencian al siguiente y así sucesivamente hasta llegar al nodo que referencia a nada o a nulo.

## INVARIANTE

Todos los nodos son los elementos de la pila y cada nodo puede o no referenciar al siguiente nodo.

## OPERACIONES

InsertAlFinal(Nodo)->Nada (Modificadora)//Inserta al final de la pila.

Push(int)->Nada (Modificadora)//Inserta en la pila.

Push(Nodo)->Nada (Modificadora)//Inserta en la pila.

Pop InsertAlFinal()->Nada (Modificadora)// Expulsa un nodo de la pila.

## MANEJO DE ERRORES

Todas las operaciones de las pila deben de mantener invariante del TAD.

Cuando una operación indebida ocurra el programa debe abortar su ejecución o informar al usuario por medio de audio o un cuadro de texto.

# COLA

## REPRESION ABSTRACTA

<nodo1, nodo2,…, nodo>

34r, donde nodo sea diferente de nulo.

Los nodos referencian al siguiente y así sucesivamente hasta llegar al nodo que referencia a nada o a nulo.

## INVARIANTE

Todos los nodos son los elementos de la cola y cada nodo puede o no referenciar al siguiente nodo.

## OPERACIONES

.

InsertAlFinal (Nodo)->Nada (Modificadora)//Inserta al final de la cola.

InsertAlFinal (int)->Nada (Modificadora)//Inserta al final de la cola.

## MANEJO DE ERRORES

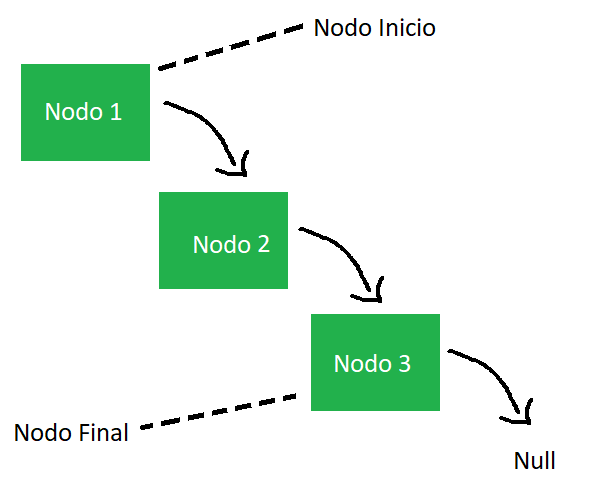
Todas las operaciones de la cola deben de mantener invariante del TAD.

Cuando una operación indebida ocurra el programa debe abortar su ejecución o informar al usuario por medio de audio o un cuadro de texto.

# ESTRUCTURA BASE

## REPRESION ABSTRACTA

<nodo1, nodo2,…, nodo>

, donde nodo sea diferente de nulo.

Los nodos referencian al siguiente y así sucesivamente hasta llegar al nodo que referencia a nada o a nulo.

## INVARIANTE

Todos los nodos son los elementos de la estructura y cada nodo puede o no referenciar al siguiente nodo.

## OPERACIONES

Mostrar ()->string (Salida en pantalla) Muestra en pantalla el contenido de la estructura, si es que tiene contenido.

Pasar (Nodo)->Nada (Modificadora) // Pasa un nodo de una estructura a otra estructura de dato.

EstaVacia ()->bolean (Analizadora) // Retorna a true si esta vacía o de lo contrario false.

GetLongitud ()-> int (Analizadora) // Retorna la longitud de la estructura.

GetNodoInicio ()->Nada (Analizadora) // Retorna el nodo de inicio de la estructura.

SetNodoInicio (Nodo)->Nada (Modificadora) // Modifica el nodo de inicio de la estructura de datos.

GetNodoFinal ()-> Nodo (Analizadora) // Retorna el nodo final de la estructura.

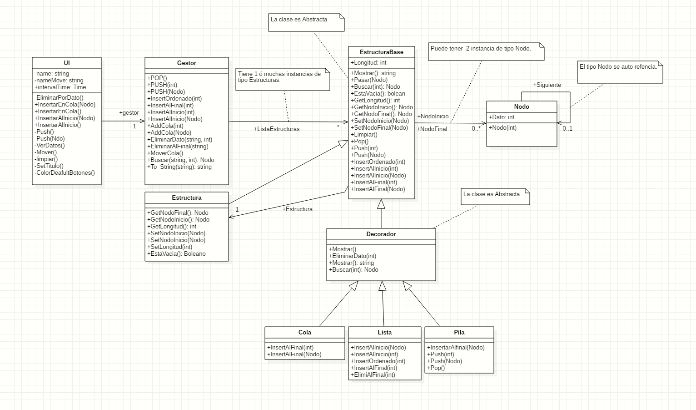
SetNodoFinal (Nodo)->Nada (Modificadora) // Retorna el nodo final de la estructura de datos.

## MANEJO DE ERRORES

Todas las operaciones de las listas deben de mantener invariante del TAD.

Cuando una operación indebida ocurra el programa debe abortar su ejecución o informar al usuario por medio de audio o un cuadro de texto.

# DIAGRAMA DE CLASES



# REFERENCIAS

[1] C. Sibaja, *“Repaso de Estructuras de Base de Datos”*, Cenfotec.SA, 2011.