Departamento de Computación
FCEFQyN, Universidad Nacional de Río Cuarto
Asignatura: Estructuras de Datos y Algoritmos - Algoritmos y Estructuras de Datos II
Primer Cuatrimestre de 2025

Práctica No. 5

Para los siguientes ejercicios proveer test *positivos* y *negativos* para los métodos de las clases para chequear que satisfacen las especificaciones.
Acceder al código de base de los ejercicios de esta práctica aceptando el siguiente assignment: `https://classroom.github.com/a/fDf-EuQP`.

1. Implementa la Interface *Lista* y agregue:
   - una implementación usando arreglos (que cambie dinámicamente el tamaño del arreglo cuando sea necesario) y
   - otra implementación usando listas enlazadas.
   - ¿Cuáles operaciones le parece que son más eficientes en una implementación y cuáles en la otra?
   - Complete la implementación usando listas Doblemente enlazadas.
2. Refactorice el código del proyecto *club* de la práctica 2 para reemplazar la estructura de datos usada (*ArrayList*) por la nueva versión que implementó en el inciso anterior. Tenga en cuenta que no debería ser necesario hacer ningún cambio en las interfaces de las clases (especificaciones y perfiles de los constructores y de sus métodos públicos) para poder modificar la implementación. Verifique que la nueva implementación preserva el comportamiento de la versión anterior ejecutando los tests sobre la nueva versión (todos deberían pasar sin modificaciones).
   - Lanzar excepciones cuando los métodos de las clases del proyecto *club* violen su precondición. Escribir tests negativos para estos métodos.
   - Codifique su solución respecto de la interfaz *Lista*. Pruebe que su código funcione usando dos implementaciones del ejercicio anterior.
3. Implemente en Java el *TAD* pila genérico. Asegúrese que todas las operaciones sean de tiempo constante. (Puede usar alguna implementación del *TAD* lista si lo desea, siempre que las operaciones sean en tiempo constante).
4. Considera que en alguna de las implementaciones de lista implementadas la operación de *concatenación* es de tiempo constante?
5. Implemente en Java el TAD *Cola* genérico. Asegúrese que todas las operaciones sean de tiempo constante. (Puede usar alguna implementación del TAD lista si lo desea, siempre que las operaciones sean en tiempo constante).
6. Utilizando alguna de las implementaciones del TAD Cola, defina un programa que decida si una cadena, ingresada por línea de comandos, es un palíndromo o no.
7. Implementar,en el lenguaje Java,un algoritmo que tome como argumentos una secuencia de paréntesis, corchetes y llaves y determine si la secuencia está balanceada. Por ejemplo, su algoritmo deberá retornar *verdadero* para [()][()()]() y *falso* para [(]).
8. Modifique el proyecto tech-support-teoria-v3 de la práctica anterior para que permita cargar las respuestas por defecto desde un archivo. Investigue cómo leer archivos en Java, y use la biblioteca que considere más apropiada para esta tarea. Maneje apropiadamente las excepciones lanzadas por la biblioteca en caso de errores (ej. archivo inexistente, no hay permisos para acceder al archivo, etc.). Escriba tests negativos para los métodos que realicen la lectura del archivo.

9. (Opcional) Implemente en Java un algoritmo que tome de línea de comandos una expresión aritmética y la evalúe (calculadora de mano). Las operaciones mínimas, que debería poder evaluar la calculadora, son: suma, resta, multiplicación y división. Como una forma de simplificación del problema, puede suponer que:

    - La expresión ingresada es sintácticamente correcta.
    - La expresión ingresada está en notación Postfija.
    - No hay operadores unarios.
    - Los operandos son enteros.