Laboratorio No. 4

Tipos de Datos Abstractos - Listas Enlazadas

(Lista enlazada simple y doble)

Introducción

En este laboratorio el/la estudiante deberá desarrollar el TDA Lista Enlazada utilizando estructuras dinámicas con apuntadores

Objetivos

Al finalizar este laboratorio, el/la estudiante deberá ser capaz de:

a. Utilizar estructuras dinámicas para solucionar problemas
b. Implementar todos los métodos de la interface Lista
c. Probar las listas enlazadas simples y dobles
d. Aplicar conocimientos discutidos en clase

Contexto

1. Trabaje con un modelo de n capas (domain, controller, test, uti)
2. Cree un nuevo proyecto llamado “Laboratory4” utilizando la tecnología javaFX, la cual permitirá trabajar en un entorno gráfico.
3. Defina las clases SinglyLinkedList (lista enlazada simple) y DoublyLinkedList (lista doblemente enlazada) e implemente los métodos definidos en la interface List. (publicada en el sitio de mediación del curso)

4. Defina una interface “Person” en la capa domain:

```
public interface Person {
    public boolean equals(Person other);
    public String getName();
    public int getAge();
    public String getAddress();
}
```

5. Defina una clase “Student” que implemente la interface “Person”. Esta clase deberá tener los siguientes objetos:
   Atributos: id(String), name(String), age(int), address
   Métodos: studyHours() retorna age/2, equals(Person other) retorna true si los objetos tienen el mismo id, toString retorna "Student {" + "id=" + id + ", name=" + name + ", age=" + age + ", study hours="+studyHours()+", address="+address+'}';

6. Defina una clase “Course” con los siguientes objetos:
   Atributos: id(String), name(String), credits(int)
   Métodos: getters, setter, toString

7. Compruebe el funcionamiento de la clase “SinglyLinkedList” a través de una clase de testeo, de la siguiente forma:

   a. Cree e instancie un objeto tipo SinglyLinkedList llamado “list”, para agregar los siguientes objetos tipo Student:
      i. Id=1, name=Maria, age=20; address=Cartago
      ii. Id=2, name=Carlos, age=22; address=San José

iii. Id=3, name=Laura, age=20; address=Paraíso
iv. Id=4, name=Paula, age=18; address=Turrialba
v. Id=5, name=Carlos, age=21; address=Limón
vi. Id=6, name=Fabiana, age=19; address=Paraíso
vii. Id=7, name=María, age=23; address=Guanacaste
viii. Id=8, name=Carlos, age=25; address=San Carlos
ix. Id=9, name=Laura, age=20; address=Turrialba
x. Id=10, name=Pedro, age=24; address=Heredia

b. Realice las siguientes pruebas y muestre el resultado por consola:
   i. Utilice el método contains(object) para buscar los siguientes estudiantes:
      ¿Existe Pedro, Id=20? false
      ¿Existe Paula, Id=4? true
      ¿Existe Carlos, Id=5? true
      ¿Existe Carlos, Id=8? true
   ii. Utilice el método getNode(i) para recorrer la lista y mostrar los elementos u objetos almacenados en cada nodo:
      for (int i = 1; i <= list.size(); i++) {
         El elemento en la posicion "+i+" es: "
   iii. Utilice el método indexOf(object) para determinar índice o posición de ciertos elementos u objetos de la lista enlazada:
      El estudiante Carlos con Id=8 se encuentra en la posicion: "
      El estudiante Carlos con Id=100 se encuentra en la posicion: "
   iv. Ordene la lista de estudiantes por nombre, con el método sort()
   v. Suprima los estudiantes con Id=1, 3, 5

c. Cree un método llamado **countNames(SinglyLinkedList list, String name),** que retorne un valor entero que indique el número de veces que se repite el nombre del estudiante indicado en la lista
   Pruebe: Buscamos cuantos Carlos tenemos en la lista: "
   +countNames(list, "Carlos") → Debe retornar: 3

d. Cree un método llamado **findNames(SinglyLinkedList list, String name),** que retorne true si el nombre indicado ha sido agregado a la lista
   Pruebe:
   ¿En la lista existe una estudiante con el nombre Karla? false
   ¿En la lista existe una estudiante con el nombre Fabiana? true

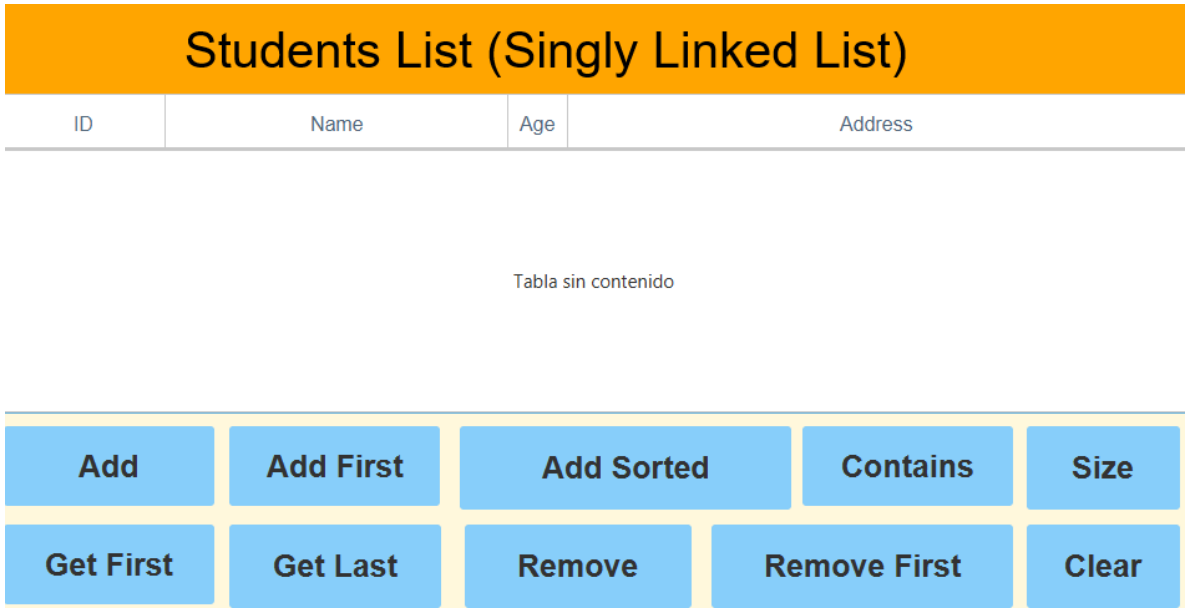
8. Compruebe el funcionamiento de la clase “DoublyLinkedList” a través de una prueba unitaria (unit testing), de la siguiente forma:
   e. Cree e instancie un objeto tipo DoublyLinkedList llamado “list”, para agregar objetos tipo Course:
      i. id=IF-3001, name=Algoritmos y Estructuras de Datos, credits=4;
      ii. id=IF-4001, name=Sistemas Operativos, credits=4;
      iii. id=IF-2000, name=Programación 1, credits=4;
      iv. id=IF-3000, name=Programación 2, credits=4;
      v. id=IF-4000, name=Arquitectura, credits=3;
      vi. id=IF-5000, name=Redes, credits=4;
      vii. id=IF-5100, name=Bases de Datos, credits=4;
      viii. id=IF-4101, name=Lenguajes app Comerciales, credits=4;
      ix. id=IF-3100, name=Sistemas de Información, credits=3;

   f. Realice las siguientes pruebas y muestre el resultado por consola:
      i. Utilice el método contains(object) para buscar los siguientes cursos:
         ¿Existe Informática Aplicada, Id=IF-6201? false

¿Existe Algoritmos y Estructuras de Datos, Id=IF-3001? true
¿Existe Sistemas Operativos, Id=IF-4001? true
¿Existe Análisis y Diseño de Sistemas, Id=IF-6100? false

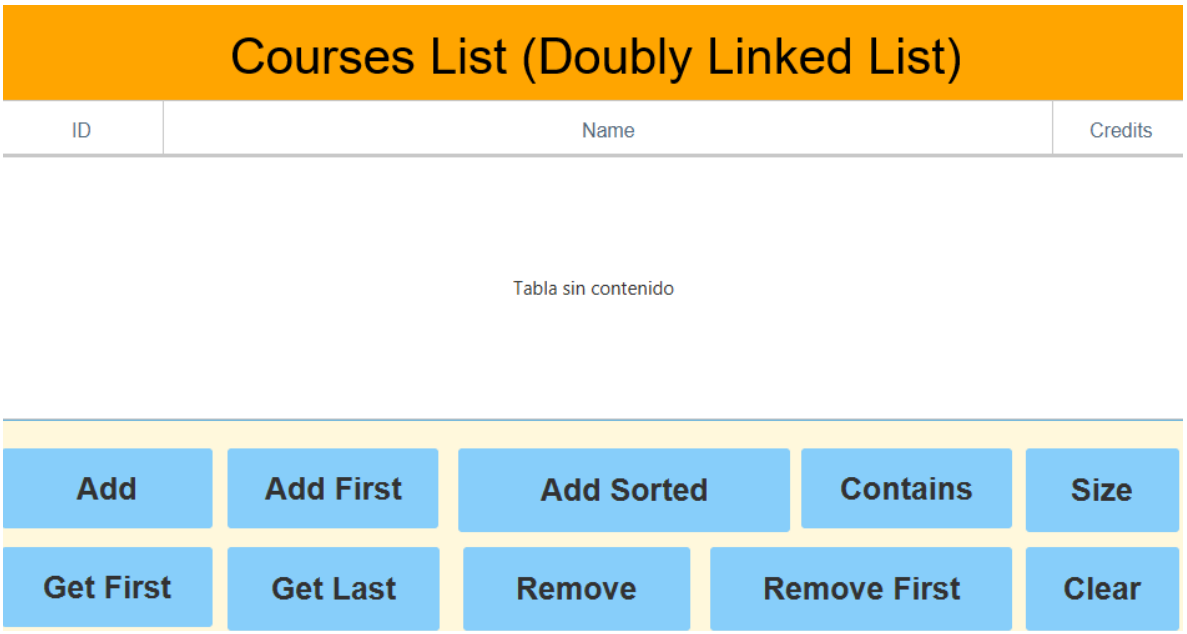
ii. Utilice el método getNode(i) para recorrer la lista y mostrar los elementos u objetos almacenados en cada nodo:
    for (int i = 1; i <= list.size(); i++) {
        El elemento en la posicion "+i+" es: "
iii. Utilice el método indexOf(object) para determinar índice o posición de ciertos elementos u objetos de la lista enlazada:
    El curso Algoritmos y Estructuras de Datos Id=IF-3001 se encuentra en la posición: "
    El curso Análisis y Diseño de Sistemas con Id=IF-6100 se encuentra en la posición: "
iv. Ordene la lista de cursos por nombre, con el método sort()
v. Suprima los cursos con Id= IF-5000, IF-5100

9. Utilice la tecnología “javaFX” para crear un entorno gráfico que muestre un menú principal y permita probar las listas enlazadas (simples y dobles), de la siguiente forma:
    a. Students (Singly Linked List)
    Permitirá llevar un control de estudiantes, por lo que deberá contar con al menos las siguientes opciones:
        i. Agregar un estudiante a la lista. Se deberá solicitar la información del estudiante (cédula, nombre, edad, dirección)
        ii. Buscar un estudiante para determinar si se ha agregado a la lista
        iii. Eliminar un estudiante de la lista
        iv. Mostrar todos los estudiantes existentes en la lista enlazada. Para ello deberá utilizar un objeto visual tipo tabla (TableView)

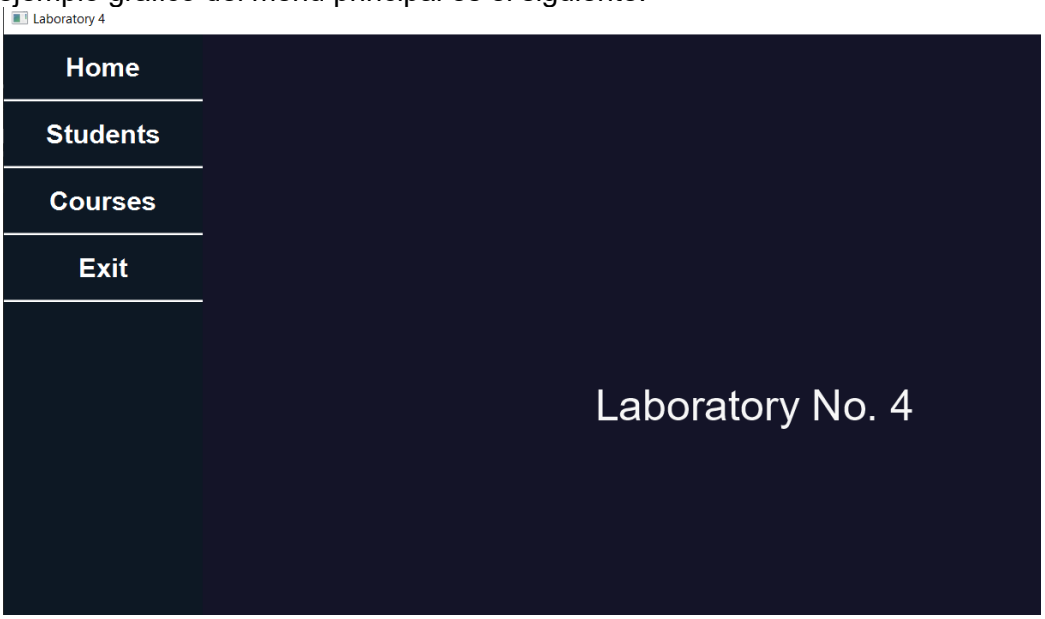


b. Courses (Doubly Linked List)
Permitirá llevar un control de cursos, por lo que deberá contar con al menos las siguientes opciones:

i. Agregar un curso a la lista. Se deberá solicitar la información del curso (sigla, nombre, créditos)
ii. Buscar un curso para determinar si se ha agregado a la lista
iii. Eliminar un curso de la lista
iv. Mostrar todos los cursos existentes en la lista enlazada. Para ello deberá utilizar un objeto visual tipo tabla



Un ejemplo gráfico del menú principal es el siguiente:



Resuelva y publique el laboratorio en el entorno del curso de la plataforma de mediación virtual (METICS). Verifique la fecha límite para el envío del informe.
URL: https://mv1.mediacionvirtual.ucr.ac.cr/course/view.php?id=7513