Grupo 5 - tarea listas enlazadas.

Integrantes del Grupo:

Edwin Romeo Rivas Diaz

Luis Ernesto Figueroa Vásquez

Kevin Alexander Aquino Vásquez

Luis Alexis Velázquez Godoy

Guillermo Alberto Asensio Jiménez

Moisés Roberto Hernández Hernández

Jonathan Alexander Ramirez Vasquez

Este proyecto implementa una lista enlazada simple, permitiendo al usuario realizar diversas operaciones comunes como la inserción, eliminación, y búsqueda de nodos. El código está organizado en varias clases que facilitan la manipulación y prueba de la lista. Aquí se describe brevemente cada una de ellas:

- **Clase Nodo**: representa un nodo de la lista con los atributos dato de tipo entero y siguiente de tipo Nodo
- **Clase ListaEnlazadaSimple:** representa una lista de nodos, posee el atributo cabeza y los métodos para insertar, eliminar, buscar e imprimir una lista
- **Clase ListaEnlazadaSimpleTest:** posee test unitarios de los métodos de insertar, eliminar y buscar usando JUnit
- **Clase Main:** Cuando se ejecuta muestra un menú de opciones que el usuario puede seleccionar para trabajar con lista enlazada simple

Instrucciones para ejecutar el proyecto:

- Sobre la estructura del proyecto hacer clic derecho sobre la clase Main y luego seleccionar Run 'Main.main()'

Instrucciones para ejecutar test unitarios

- Sobre la estructura del proyecto hacer clic derecho sobre la clase ListaEnlazadaSimpleTest y seleccionar Run 'ListaEnlazadaSimpleTest'

Los test que se realizan son los siguientes:

1- Método de insertar al inicio: Insertar 10 y verificar que esté en la lista con método de buscar, Insertar 20 y verificar que esté en la lista usando método buscar

```
lista.insertarAlInicio(10);

assertTrue("El elemento 10 debería estar en la lista.",
lista.buscar(10));

lista.insertarAlInicio(20);

assertTrue("El elemento 20 debería estar en la lista.",
lista.buscar(20));
```

2- Método de insertar al final: Insertar 10 y verificar que esté en la lista con método de buscar, Insertar 20 y verificar que esté en la lista usando método buscar

```
lista.insertarAlFinal(10);

 assertTrue("El elemento 10 debería estar en la lista.",
lista.buscar(10));

lista.insertarAlFinal(20);

assertTrue("El elemento 20 debería estar en la lista.",
lista.buscar(20));
```

3- Método de elimninar al inicio: Insertar 10 y 20 al inicio , invocar eliminar al inicio, verificar que 20 fue eliminado y 10 sigue apareciendo

```
lista.insertarAlInicio(10);

lista.insertarAlInicio(20);

lista.eliminarAlInicio();

assertFalse("El elemento 20 debería haber sido eliminado.",
lista.buscar(20));

assertTrue("El elemento 10 debería estar en la lista.",
lista.buscar(10));
```

4- Método de eliminar al final: Insertar 10 y 20 al final , invocar eliminar al final, verificar que 20 fue eliminado y 10 sigue apareciendo

```
lista.insertarAlFinal(10);

lista.insertarAlFinal(20);

lista.eliminarAlFinal();

assertFalse("El elemento 20 debería haber sido eliminado.",
lista.buscar(20));

assertTrue("El elemento 10 debería estar en la lista.",
lista.buscar(10));
```

5- Test de eliminar al final cuando la lista posee un solo elemento: insertar 10 al final, eliminar al final, verificar que la lista está vacía y no aparecen elementos en la lista

```
lista.insertarAlFinal(10);

lista.eliminarAlFinal();

assertFalse("La lista debería estar vacía después de eliminar el único elemento.", lista.buscar(10));
```

6- Test de eliminar al inicio cuando la lista posee un solo elemento: insertar 10 al inicio, eliminar al inicio, verificar que la lista está vacía y no aparecen elementos en la lista

```
lista.insertarAlInicio(10);

lista.eliminarAlInicio();

assertFalse("La lista debería estar vacía después de eliminar el único elemento.", lista.buscar(10));
```

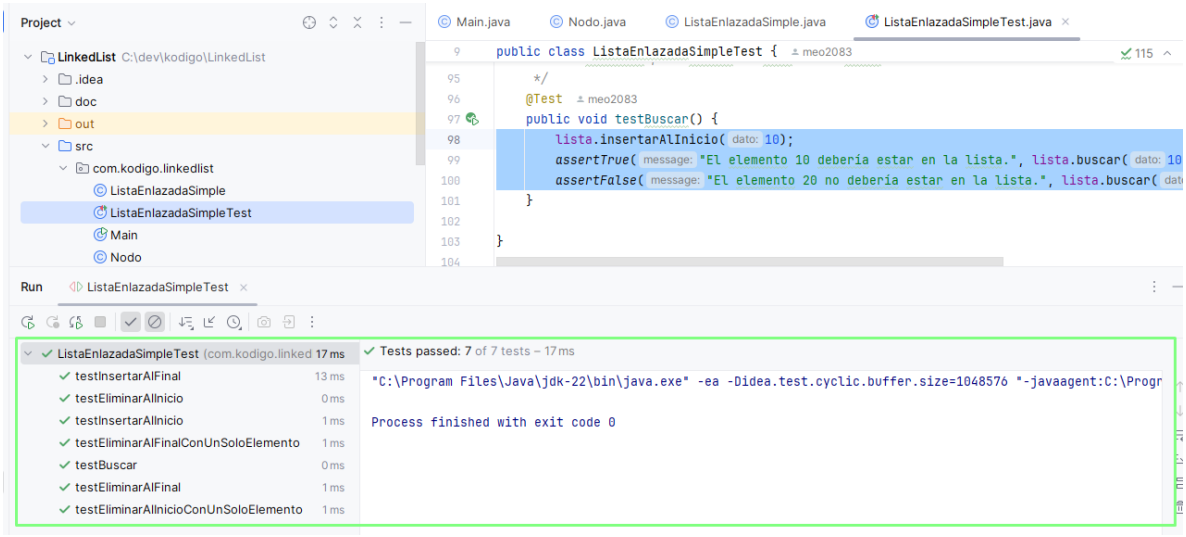
7- Método buscar: Insertar 10 al inicio, verificar que 10 aparece en la lista al buscar y verificar que otro número no aparece al buscar, como ejemplo 20

```
lista.insertarAlInicio(10);

assertTrue("El elemento 10 debería estar en la lista.", lista.buscar(10));

assertFalse("El elemento 20 no debería estar en la lista.", lista.buscar(20));
```

Test de pruebas:



Estas pruebas garantizan que la funcionalidad de la lista enlazada simple opere correctamente y que se manejen adecuadamente los casos de borde, como la eliminación de elementos en listas vacías o con un solo elemento.

Para acceder al repositorio del proyecto y revisar el código fuente completo en el siguiente enlace: https://github.com/meo2083/LinkedList.git