KEVIN ALEXANDER AQUINO VASQUEZ

PROGRAMA LISTA ENLAZADA SIMPLE

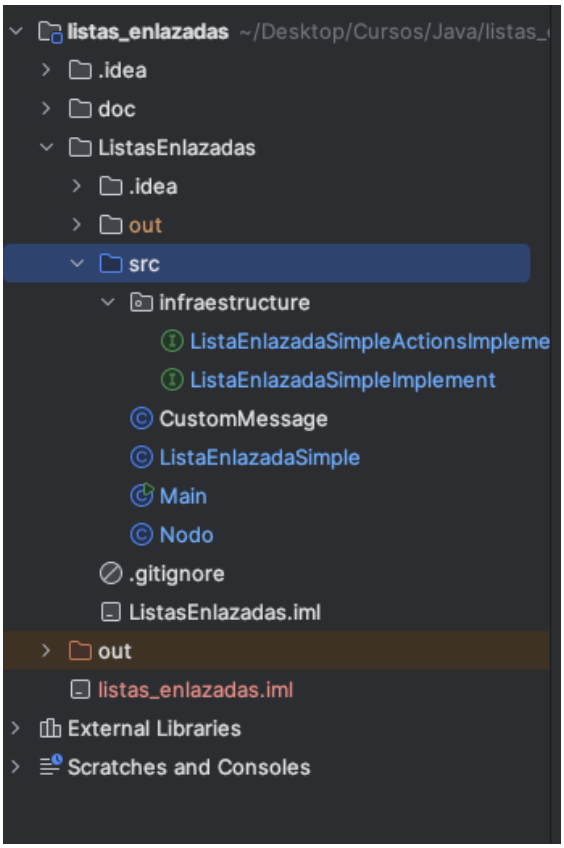
KODIGO

Tabla de contenido

Descripcion de directorio

Como se puede visualizar en la ilustracion 1, tenemos nuestro directorio de archivos, los cuales contiene lo siguiente:

- Clase Nodo
- Clase ListaEnlazadaSimple
- Clase CustomMessage
- Clase Main
- Infraestructure
    - ListaEnlazadaSimpleActionsImplement
    - ListaEnlazadaSimpleImplement

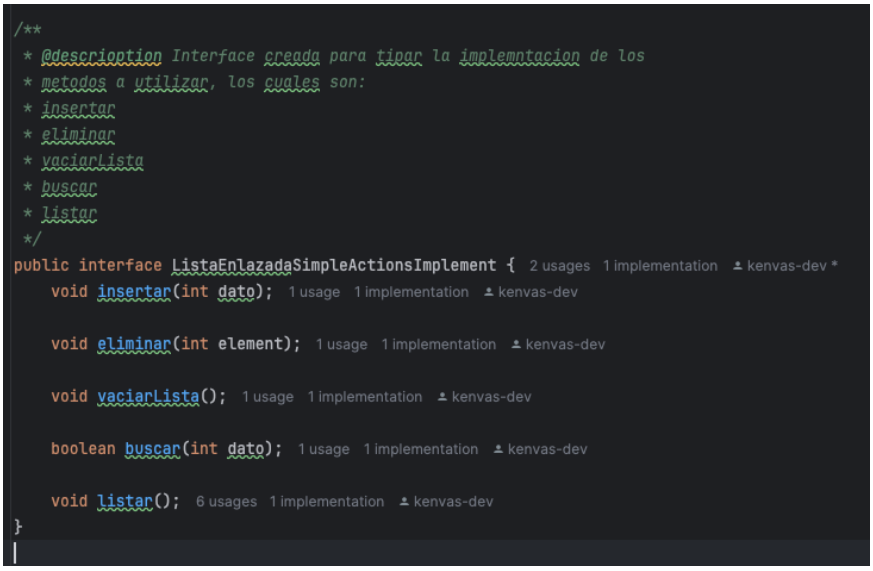


Infraestructure

Interface ListaEnlazadaSimpleActionsImplement

Interface creada para tipar la implemntacion de los, metodos a utilizar, los cuales son:

- insertar
- eliminar
- vaciarLista
- buscar
- listar



```java
/**
 * @descriqption Interface creada para tipar la implemntacion de los
 * metodos a utilizar, los cuales son:
 * insertar
 * eliminar
 * vaciarLista
 * buscar
 * listar
 */
public interface ListaEnlazadaSimpleActionsImplement {  2 usages  1 implementation  kenvas-dev *
    void insertar(int dato);  1 usage  1 implementation  kenvas-dev

    void eliminar(int element);  1 usage  1 implementation  kenvas-dev

    void vaciarLista();  1 usage  1 implementation  kenvas-dev

    boolean buscar(int dato);  1 usage  1 implementation  kenvas-dev

    void listar();  6 usages  1 implementation  kenvas-dev
}
```

Interface ListaEnlazadaSimpleImplement

Interface creada para tipar la implemntacion de los metodos a utilizar (solicitados en el requerimiento), los cuales son:

- insertarAlInicio
- eliminarAlInicio

Clase CustomMessage

Clase que implementa el uso del System.out.print.

Clase presonalizada para mostrar mensajes.

Dicha clase cuenta con dos metodos estaticos, los cuales son:

- title
- out
- outln

los dos metodos (out, outln) reciben como parametro “texts” de tipo String[].

El metodo title recibe unicamente el parametro “title” de tipo String

Metodo title

Banner creado para mostrar enunciados

Metodo out

Es una personalizacion del syop para un listado de opciones, sin saltos de linea.

```java
/**
 * @name out
 * @description
 * Es una personalizacion del syop
 * para un listado de opciones, sin saltos de linea.
 * @param texts
 */
public static void out(String[] texts) {  kenvas-dev
    for (String text : texts) {
        System.out.print(text+"    ");
    }
}
```

Metodo outln

Es una personalizacion del syop para un listado de opciones, con saltos de linea.

```java
/**
 * @name outln
 * @description
 * Es una personalizacion del syop
 * para un listado de opciones, con saltos de linea.
 * @param texts
 */
public static void outln(String[] texts) {  18 usages  kenvas-dev
    for (String text : texts) {
        System.out.println(text);
    }
}
```

Clase Nodo

Clase creada para almacenar y acceder a valores asignados

Dicha clase, contiene las siguientes propiedades:

- dato de tipo int
- siguiente de tipo Nodo

las dos propiedades cuentan con su metodos de acceso.

```java
/**
 * @name Nodo
 * @description Clase creada para almacenar y acceder a valores asignados
 */
public class Nodo {  12 usages  kenvas-dev *
    /**
     * Propiedad privada de tipo int
     * con sus metodos de accesos
     */
    private int dato;  3 usages

    /**
     * Propiedad privada de tipo Nodo
     * con sus metodos de accesos
     */
    private Nodo siguiente = null;  3 usages

    public Nodo(int dato, Nodo siguiente) {  2 usages  kenvas-dev
        this.dato = dato;
        this.siguiente = siguiente;
    }

    public int getDato() {  4 usages  kenvas-dev
        return dato;
    }

    public void setDato(int dato) {  no usages  kenvas-dev
        this.dato = dato;
    }

    public Nodo getSiguiente() {  12 usages  kenvas-dev
        return siguiente;
    }

    public void setSiguiente(Nodo siguiente) {  2 usages  kenvas-dev
        this.siguiente = siguiente;
    }
}
```

Clase ListaEnlazadaSimple

Clase creada para crear una lista de elementos.

Esta clase implementa dos interfaces, las cuales son:

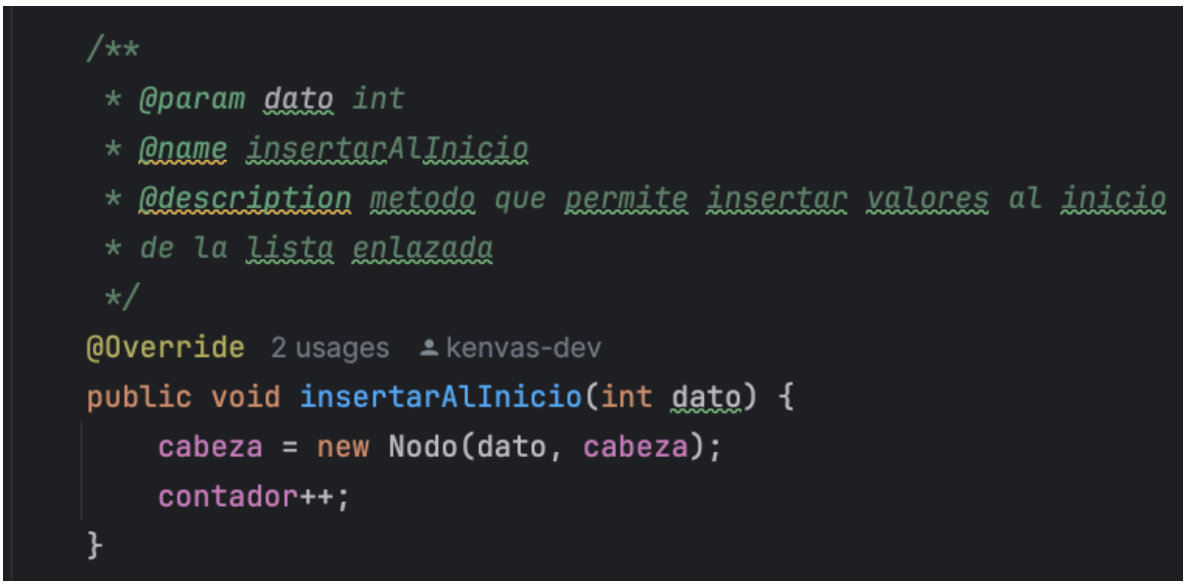
- ListaEnlazadaSimpleImplement
- ListaEnlazadaSimpleActionsImplement

Y cuenta con las siguientes propiedades y metodos:

- Cabeza de tipo Nodo
- Contador de tipo int
- Metodo insertarAlInicio
- Metodo eliminarAlInicio
- Metodo insertar
- Metodo eliminar
- Metodo buscar
- Metodo listar
- Metodo vaciarLista

Metodo insertarAlInicio

Metodo que permite insertar valores al inicio de la lista enlazada



```java
/**
 * @param dato int
 * @name insertarAlInicio
 * @description metodo que permite insertar valores al inicio
 * de la lista enlazada
 */
@Override  2 usages  kenvas-dev
public void insertarAlInicio(int dato) {
    cabeza = new Nodo(dato, cabeza);
    contador++;
}
```

Metodo eliminarAlInicio

Metodo que permite insertar valores al final de la lista enlazada

```java
/**
 * @name eliminarAlInicio
 * @description metodo que permite eliminar valores al inicio
 * de la lista enlazada
 */
@Override 1 usage  kenvas-dev
public void eliminarAlInicio() {
    final int indexInit = 0;
    if (indexInit < contador) {
        cabeza = cabeza.getSiguiente();
        contador--;
    }
}
```

Metodo insertar

Metodo que permite insertar valores al final de la lista enlazada

```java
/**
 * @name insertar
 * @description metodo que permite insertar valores al final
 * de la lista enlazada
 */
@Override 1 usage  kenvas-dev
public void insertar(int dato) {
    Nodo nuevo = new Nodo(dato, siguiente: null);
    if (cabeza == null) {
        cabeza = nuevo;
    } else {
        Nodo actual = cabeza;
        while (actual.getSiguiente() != null) {
            actual = actual.getSiguiente();

        }
        actual.setSiguiente(nuevo);
        contador++;
    }
}
```

Metodo eliminar

Metodo que permite eliminar el valor digitado de la lista enlazada

```java
/**
 * @param element
 * @name eliminar
 * @description metodo que permite eliminar el valor digitado
 * de la lista enlazada
 */
@Override 1 usage  kenvas-dev *
public void eliminar(int element) {
    if (cabeza == null)
        CustomMessage.outln(new String[]{"lista vacia"});
    else if (cabeza.getDato() == element) {
        cabeza = cabeza.getSiguiente();
        contador--;
    } else {
        Nodo actual = cabeza;
        while (actual.getSiguiente() != null && actual.getSiguiente().getDato() != element)
            actual = actual.getSiguiente();
        if (actual.getSiguiente() == null)
            CustomMessage.outln(new String[]{"El elemento " + element + " no esta en la lista"});
        else {
            actual.setSiguiente(actual.getSiguiente().getSiguiente());
            contador--;
        }
    }
}
```

Metodo buscar

Metodo que permite buscar el valor digitado de la lista enlazada

```java
/**
 * @param element
 * @name eliminar
 * @description metodo que permite buscar el valor digitado
 * de la lista enlazada
 */
@Override 1 usage  kenvas-dev
public boolean buscar(int element) {
    Nodo aux = cabeza;
    boolean encontrado = false;
    while (aux != null && !encontrado) {
        if (element == aux.getDato()) {
            encontrado = true;
        } else {
            aux = aux.getSiguiente();
        }
    }
    if (!encontrado)
        CustomMessage.outln(new String[]{"El elemento " + element + " no esta en la lista"});
    if (encontrado)
        CustomMessage.outln(new String[]{"El elemento " + element + " se encontro en la lista"});

    return encontrado;
}
```

Metodo listar

Metodo que permite listar la lista enlazada

```java
/**
 * @name listar
 * @description metodo que permite listar la lista enlazada
 */
@Override 6 usages  kenvas-dev *
public void listar() {
    if (cabeza != null) {
        Nodo aux = cabeza;
        int i = 0;
        while (aux != null) {
            CustomMessage.out(new String[]{"| " + aux.getDato() + " |"});
            aux = aux.getSiguiente();
            i++;
        }
    } else {
        CustomMessage.outln(new String[]{"lista vacía"});
    }
    System.out.println();
    System.out.println();
}
```

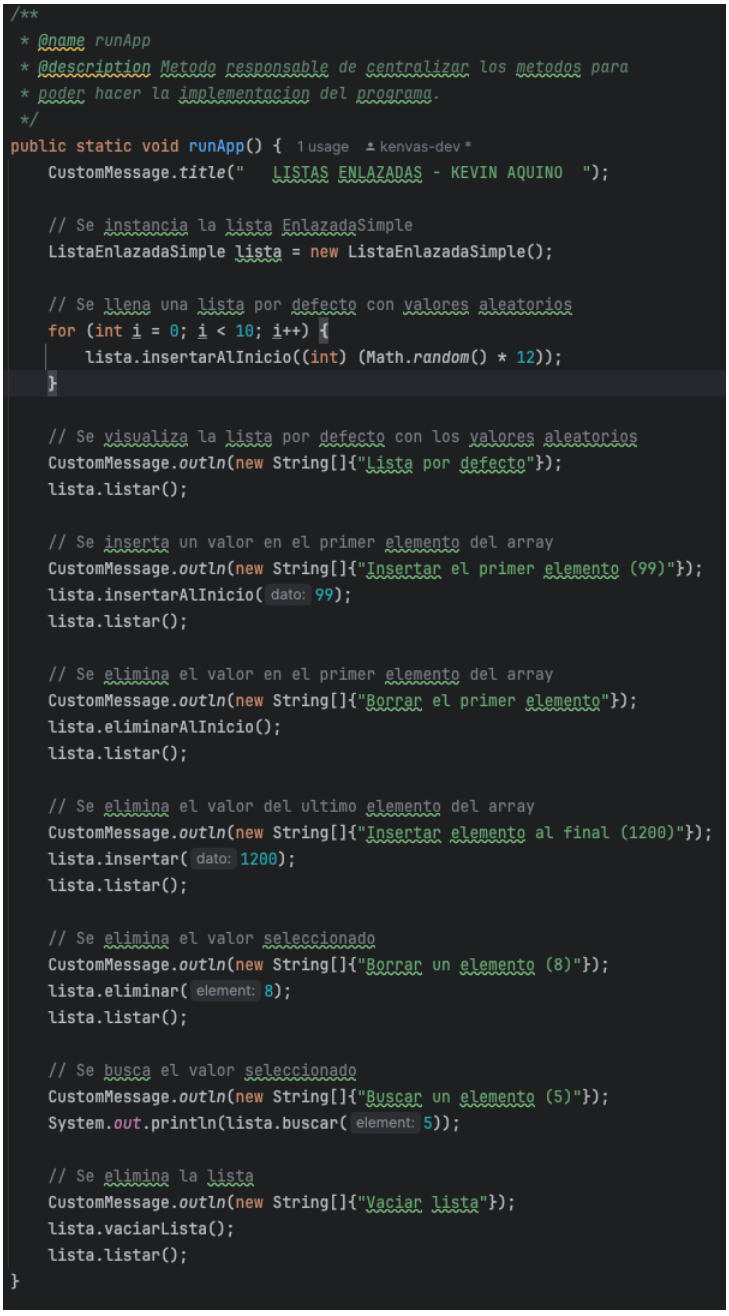
Metodo vaciarLista

Metodo que permite vaciar la lista enlazada

```java
/**
 * @name vaciarLista
 * @description metodo que permite vaciar la lista enlazada
 */
@Override 1 usage  kenvas-dev *
public void vaciarLista() {
    cabeza = null;
    contador = 0;
}
```

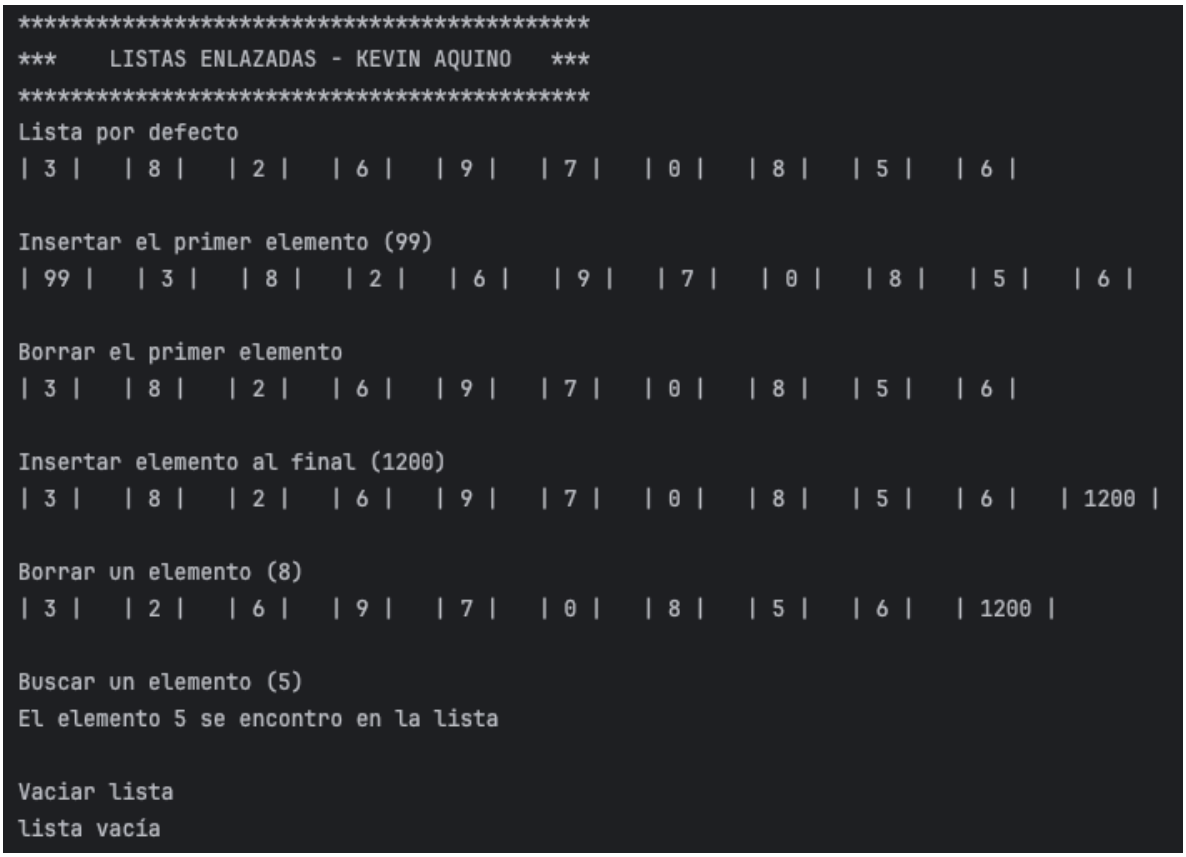
Clase Main

La implementacion de la solucion de llama en esta clase, en la cual se ha creado un metodo llamado “runApp” el cual centraliza la invicacion de las demas clases, asi siendo este unico metodo que se llama al “main”



```java
/**
 * @name runApp
 * @description Metodo responsable de centralizar los metodos para
 * poder hacer la implementacion del programa.
 */
public static void runApp() {  1 usage   kenvas-dev *
    CustomMessage.title("   LISTAS ENLAZADAS - KEVIN AQUINO  ");

    // Se instancia la lista EnlazadaSimple
    ListaEnlazadaSimple lista = new ListaEnlazadaSimple();

    // Se llena una lista por defecto con valores aleatorios
    for (int i = 0; i < 10; i++) {
        lista.insertarAlInicio((int) (Math.random() * 12));
    }

    // Se visualiza la lista por defecto con los valores aleatorios
    CustomMessage.outln(new String[]{"Lista por defecto"});
    lista.listar();

    // Se inserta un valor en el primer elemento del array
    CustomMessage.outln(new String[]{"Insertar el primer elemento (99)"});
    lista.insertarAlInicio( dato: 99);
    lista.listar();

    // Se elimina el valor en el primer elemento del array
    CustomMessage.outln(new String[]{"Borrar el primer elemento"});
    lista.eliminarAlInicio();
    lista.listar();

    // Se elimina el valor del ultimo elemento del array
    CustomMessage.outln(new String[]{"Insertar elemento al final (1200)"});
    lista.insertar( dato: 1200);
    lista.listar();

    // Se elimina el valor seleccionado
    CustomMessage.outln(new String[]{"Borrar un elemento (8)"});
    lista.eliminar( element: 8);
    lista.listar();

    // Se busca el valor seleccionado
    CustomMessage.outln(new String[]{"Buscar un elemento (5)"});
    System.out.println(lista.buscar( element: 5));

    // Se elimina la lista
    CustomMessage.outln(new String[]{"Vaciar lista"});
    lista.vaciarLista();
    lista.listar();
}
```

Implementacion de la solucion



Como se puede observar en la imagen, se muestra una lista inicial, que a lo largo de la ejecucion y en las llamadas de los metodos como el “insertar el primer elemento (99)” se insertar en el primer indice de la lista. Tambien se observa que se elimina dicho elemento de la lista, a su ves se puede observar como se puede insertar elementos al final de la lista. Ademas, se han agregados metodos tales como, borrar un elemento del array, busqueda, y vaciado de la lista enlazada.